学校：天津现代职业技术学院　　参赛队编号：J09

2013 全国大学生电子设计竞赛

设计报告

题目:电磁控制运动装置（J 题）

电磁控制运动装置设计报告

摘要

随着科技的发展，近年来单片机等微型处理器在控制方面的应用越来越多。加之其易于使用、性价比高，所以用该类型芯片开发的产品成本低廉且使用方便。

本电磁控制运动装置系统是一个兼具手动控制、自动控制盒实时显示功能的控制系统。该系统是以 MSP430 处理器为核心，通过 PWM（Pulse Width Modulation,脉冲宽度调制）技术把直流电压变成电压脉冲列，并通过控制电压脉冲宽度和脉冲列的周期来达到变压、变频目的的一种控制技术，使电磁铁的磁性变化，“推动”摆杆摆动，用键盘设定摆杆摆角幅度和周期，通过角度传感器 ADXL345 将实时采集到的角度数据以 SPI 串行接口的形式传递给模数转换器，经由单片机处理使电磁铁吸引摆杆在规定的角度内摆动，形成一个电磁控制运动系统使摆杆能更精准的转动规定的角度，当达到设定要求时可进行声光提示，并可通过按键来操控摆杆的摆动的幅度以及周期。以该系统摆动角度的准确性以及周期来判断该设计的完成指标。

关键词：MSP430；PWM；电磁铁；ADXL345；电磁控制

目录

任务与要求

设计并制作一套电磁控制运动装置，该装置由电磁控制装置、摆杆等部分构成。

（1）按下启动按钮，由静止点开始，控制摆杆摆动。

（2）由静止点开始，控制摆杆在指定的摆角（10°～45°范围内）连续摆动，摆动摆角绝对误差≤5°，响应时间≤15s。

（3）由静止点开始，按指定周期（0.5s～2s 范围内）控制摆杆连续摆动，摆动周期绝对误差值≤0.2s，响应时间≤15s。

（4）在摆杆连续摆动的情况下，按下停止按钮，控制摆杆平稳地停在静止点上，停止时间≤10s。

（5）摆杆摆角幅度能在 10°～45°范围内预置，预置步进值为 5°，摆角幅度绝对误差值≤3°，响应时间≤10s。

（6）摆杆的周期能在 0.5s～2s 范围内预置，预置步进值 0.5s，周期绝对误差值≤0.1s，响应时间≤10s。

（7）摆杆摆角幅度和周期在上述范围内可同时预置，由静止点开始摆动，摆角幅度值和周期相对误差要求均和发挥部分中的（1）、（2）相同。当摆杆稳定运行 20 秒后发出声、光提示，并在 5s 内平稳停在静止点上。

1.系统方案比较和选择

本次实验利用 MSP430 单片机接收按键采集信号，通过控制电磁铁的磁性有无以及强弱是摆杆摆动所要求的角度，通过角度传感器 MMA7361L 将实时采集到的角度数据以模拟电压的形式传递给模数转换器，经由单片机处理使电磁铁吸引摆杆在规定的角度内控制摆杆的运转角度。

总体方案设计

根据实际的要求，系统可分为控制模块、电源模块、角度检测模块、电磁驱动模块、按键及显示模块、声光提示模块五部分组成，系统框图如下图1所示：

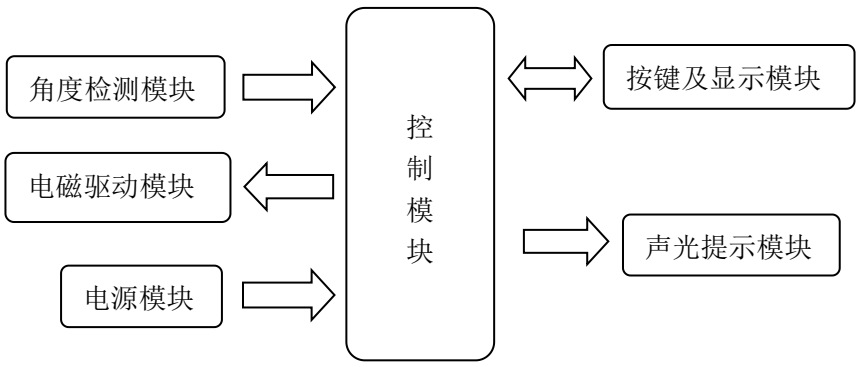


图 1 系统框图

根据要求，我们对以下三部分进行了方案论证与比较：

1.1 控制模块的比较与选择

方案一：选用 FPGA 作为核心控制器，采用光电计数器作为角度检测传感器，对光电计数器的输出进行计数，确定摆杆摆动角度并进行实时检测。电机驱动采用 PWM 控制方式。

优点：资源丰富，接口众多，运算速度快，控制精度高

缺点：价格较高、程序编写复杂，同时因随机摆动的可能会造成光电计数器读值不准而造成控制误差

方案二：采用普通 51 单片机作为核心控制器，采用精密单圈电位器作为角度检测传感器，经过 ADC 转换之后送人 MCU，采用控制输出电压的方式控制电机输出速度。

优点：成本低、程序编写简单

缺点：控制精度低、处理速度较慢、功耗高。

方案三：采用 TI 公司的 MSP430 作为核心控制器，它是一个 16 位的功能强大的超低功耗微处理器，特别适合于电池应用的场合，而且其集成度高，不需要加入其他的芯片进行辅助，就能完成本次设计的各项要求，同时 MSP430 有丰富的不同类型器件可供选择，给我们的设计带来了很大的灵活性，而且其性价比高。

优点：处理能力强、运算速度快、超低功耗、片内资源丰富、方便高效的开发环境

综和上比较以上三种方案中控制器的优缺点，第三种方案具有更大的优越性，我们选择方案三进行设计。

1.2 显示模块的比较和选择

方案一：采用 LED 数码管显示，利用 74HC373 驱动数码管动态显示，控制比较简单，但占用较多 I/O 口，不能实现资源的有效利用，而且只能显示一些简单的字符，显示信息量有限且不能达到题目要求。

方案二：采用 Cry12864LCD 液晶显示，LCD 微功耗、超薄轻小，可以灵活显示图片文字、字迹清晰，因此具有友好的人机交流显示界面，特别适合智能系统的可编程人性化显示。

综合以上论述，选择方案二。

1.3 按键模块的比较和选择

方案一：采用行列式键盘，这种键盘的特点是行线、列线分别接输入线、输出线。按键设置在行、列线的交叉点上，利用这种矩阵结构只需 m 根行线和 n 根列线就可组成 m×n 个按键的键盘，因此矩阵式键盘适用于按键数量较多的场合。但此种键盘的软件结构较为复杂。

方案二：采用独立式键盘，这种键盘硬件连接和软件实现简单，并且各按键

相互独立，每个按键均有一端接地，另一端接到输入线上。按键的工作状态不会影响其它按键上的输入状态。但是由于独立式键盘每个按键需要占用一根输入口线，所以在按键数量较多时，I/O 口浪费大，故此键盘只适用于按键较少或操作速度较高的场合。

根据上面两种方案的论述，由于本次设计的系统硬件只需要五个按键，所以采用方案二独立键盘进行设计。

1.4 角度检测模块的比较和选择

方案一：采用光电编码器，它是一种角度检测装置，将角度变化量，利用光电转换原理转换成相应的电脉冲或数字量，具有体积小、工作可靠、接口数字化等优点；其缺点是无法输出轴转动的绝对位置信息，达不到所需精度。

方案二：采用高精度单轴 SCA61T 倾角传感器，它具有双输出接口；模拟电压输出和数字接口 SPI 接口。还内置了温度传感器和 EEPROM 存储器，用于给倾角传感器做温度补偿和存储补偿数据使用；此外它还具有自我检测/验证引脚的功能，能通过此引脚（SELF TEST）检测传感器是否正常工作。此外 SCA61T 倾角传感器，具有分辨率高、噪声低、抗冲击能力强、反应迅速等等优点。但是本次设计是要精准的监测单摆所转动的角度，如若选用该传感器，并不能很好的完成本次实验。

方案三：采用高精度的模拟三轴加速度传感器 MMA7361L，MMA7361L 是一种超低功耗、小型电容式的微机械加速度传感器。一种可以对物体运动过程中的加速度进行测量的电子设备，典型互动应用中加速度传感器可以用来对物体的运动方向进行监测，根据物体运动和方向改变输出信号的电压值。精密多圈电位器直接把角度变化转化为电阻值的变化，控制器利用其内部 AD 转换将其转化为数字量，通过计算得出角度值，软件编程灵活，硬件电路简单，实现比较容易。

综合以上论述，选择方案三。

1.5 电磁驱动模块的比较和选择

方案一：采用伺服电机控制滚珠丝杠使电磁铁在规定的位置范围内滑动，滚珠丝杠具有很小的摩擦阻力、具高精度、可逆性和高效率的特点。传动的原理就是通过伺服马达的带动，丝杆做高速运动，通过螺母来带动固定在螺母上面的电磁铁，特点是滚珠丝杆的精度比较高，误差小。由于伺服电机的灵敏度太高不易控制，磁铁摆杆摆动的角度范围有限，因此不能完成此次实验的要求。

方案二：采用自制电磁铁，在磁铁摆杆正下方排成一排，通过对单个或多个电磁铁通电起到吸引或者排斥的作用，控制摆杆摆动，由 MMA7361L 角度传感器实时检测磁铁摆杆摆动的角度以达到之前摆杆摆动设定的角度以及周期，电磁铁的磁性可通过 PWM（脉冲宽度调制）来调节，控制起来方便简洁，易实现。

综合以上论述，选择方案二。

1.6 电源模块的比较和选择

方案一：采用+5V 和+24 两个电源供电。将伺服电机驱动电源与处理器以及其周电路电源完全隔离，利用三极管 9013 传输 PWM 信号。这样可以使直流轴流风扇驱动所造成的干扰彻底消除，但是电路复杂，容易造成短路。

方案二：采用单一电源供电。电源直接给伺服电机供电，因伺服电机启动瞬间电流较大，会造成电源电压波动，因而控制与检测等其他部分电路通过集成稳压块供电。其供电电路比较简单，但干扰太大，不易提高精度。

方案三：采用开关电源，开关电源开关电源一般由脉冲宽度调制（PWM）控制 IC 和 MOSFET 构成，体积小、重量轻，由于没有工频变压器，所以体积和重量只有线性电源的 20～30%，功耗小、效率高，功率晶体管工作在开关状态，所以晶体管上的功耗小，转化效率高，继电器易选择，电路构成简单，集成度高。

综合以上论述，选择方案三。

系统各模块的最终方案

经过仔细分析和论证，决定了系统各模块的最终方案如下：

（1）控制模块：采用 TI 公司的 MSP430

（2）显示模块：采用 Cry12864LCD 液晶显示

（3）按键模块：采用独立式键盘

（4）角度检测模块：采用高精度的模拟三轴加速度传感器 MMA7361L

（5）电磁驱动模块：采用自制电磁铁

（6）电源模块：采用开关电源

2.系统理论分析与参数计算

2.1 PWM 驱动的基本原理及特点

PWM（Pulse Width Modulation,脉冲宽度调制）技术是利用半导体开关器件的导通和关断，把直流电压变成电压脉冲列，并通过控制电压脉冲宽度和脉冲列的周期来达到变压、变频目的的一种控制技术[2]。也就是用脉冲宽度不等的一系列等幅值的矩形脉冲去逼近一个所需要的电流或电压信号。

PWM 驱动电路，是广泛应用于高精度控制系统的驱动形式。这种电路能够实现宽范围的速度和位置控制，较之常规驱动方式具有无可比拟的优点。 PWM 驱动电路线路简单、快速性好、线性度好、效率高的优点，使其广泛应用于测量、通信、功率控制与变换的许多领域中。本设计利用 PWM 驱动电路所需大功率可控器件少、调速范围宽、快速性好、效率高，功耗低的特点，用 C8051F005 单片机直接输出的 PWM 信号经过驱动电路，然后配合合适的控制算法（PID 算法或模糊控制算法等）去控制比例电磁铁，可实现离合器的精确控制，对于电控离合器控

制系统的研究有很好的参考价值。

2.2 大幅度摆动单摆的高精度周期

一个质量为 m 的小球由一根轻质的长度为 L 的刚性细绳悬挂在一个固定的支架上(小球半球远远小于细绳长度),小球在重力的作用下可在垂直平面内来回摆动(不考虑空气阻力),如图 2-2 所示,根据受力分析重力对该系统提供外力矩作用,由转动定律可得小球运动的微分方程为:

$$\frac{d^2\theta}{dt^2}+\frac{g}{L}\sin\theta=0 \qquad (1)$$

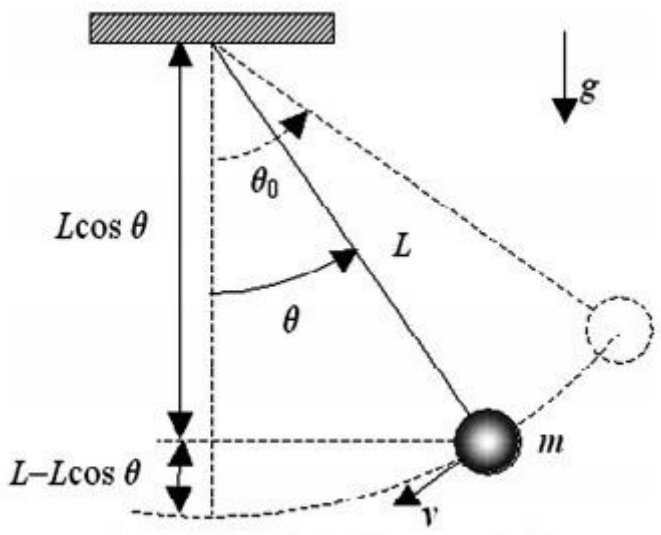


图 2-2 单摆任意角圆周运动示意图

式(1)中 θ 为离开平衡位置的角位移、g 为重力加速度。若给定初始条件,式(1)的任意精度的数值解是可以求出来的,但要求出单摆周期的解析解还应采取一些近似处理。如果我们采用小角度近似 θ<5° ,sin θ ≌ θ ,式(1)便成为一个线性微分方程,相应的摆动便是简谐振动。在这个角度限制下单摆振动周期为 T0=2π 。事实上,T0 并不能表示任意摆动幅度的周期,由于在小角度近似条件下我们几乎觉察不到周期有什么不同。超出这个小角度限制,随着摆角增大 T 0 越来越不能描述单摆的精确周期,这时式(1)可以通过数值模拟求解。

当单摆的摆动角度>5° ,由于系统的机械能守恒,从能量的观点出发也可以求解单摆周期的精确解,这样就不需要详细讨论式(1)非线性微分方程。给出非常简单的单摆周期 T3
的公式,

$$T_3=T_0\frac{1}{\sqrt{\cos(\theta_0/2)}} \qquad (2)$$

式(2)简单实用,由式(2)可以计算出,当摆角为 57° 左右,其相对误差为 0.1%,摆角为 90° 相对误差还不到 0.75%。

2.3 角度测量原理

角度测量采用倾角传感器 MMA7361L 的传感轴安装在与摆杆同一平面内，采用双轴测量值合成来计算倾斜角，在小倾角测量时，具有高分辨率和高精度的特点。

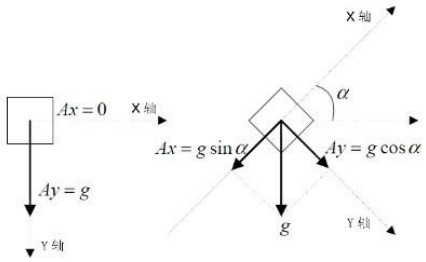


图 2-3 角度测量原理图

由于加速度传感器在静止放置时受到重力作用，因此会有1g 的重力加速度。利用这个性质，通过测量重力加速度在加速度传感器的 X 轴和 Y 轴上的分量，可以计算出其在垂直平面上的倾斜角度。如上图所示。有

$$Ax = g\sin\alpha, \quad Ay = g\cos\alpha, \quad \text{则}\ \tan\alpha = Ax / Ay$$

$$\text{即}\ \alpha = \arctan\left(\frac{Ax}{Ay}\right)$$

这样，根据以上原理一个2轴加速度传感器可以测量在 X-Y 平面上的倾斜角度。这个公式就是本文中用来测量物体倾斜角度的基本原理。需要说明的是，这里利用的是物体在静止时受到重力的性质，如果物体同时也有运动加速度的话，那么这个公式将不再准确。所以必须为公式增加一个限制条件，即

$$\begin{cases} \alpha = \arctan\left(\dfrac{Ax}{Ay}\right) \\ \sqrt{Ax^2 + Ay^2} = 1g \end{cases}$$

角度测量模块采用角度传感器 MMA7361L，利用了电阻分压原理。当帆板带动传感器转动时，角度的变化使得传感器内部电阻的阻值发生变化，从而导致电压值的变化，控制器对电压信号进行采集，通过 AD 转换，将其变换成数字量，经过内部计算，转化成角度值。

2.4 电磁铁磁性控制

控制器从角度传感器采集到角度值，和预设的角度值作比较，经过运算，调整 PWM 脉宽，PWM 调速是使加在直流电机两端的电压为方波形式,通过改变方波占空比实现对电机转速调节。

在自动控制系统中，PID 算法使用比较广泛。它具有原理简单，易于实现，适用面广，控制参数相互独立，参数的选定比较简单等优点；而且在理论上可以

证明，对于过程控制的典型对象——“一阶滞后＋纯滞后”与“二阶滞后＋纯滞后”的控制对象，PID 控制器是一种最优控制。PID 调节规律是连续系统动态品质校正的一种有效方法，它的参数整定方式简便，结构改变灵活。本系统中，摆杆的角度要比较稳定，这就需要将角度传感器检测的值与设定值比较，然后调整占空比控制电磁铁磁力，从而调整摆杆角度。

3.电路设计与程序设计

根据题目要求，将角度检测信号送入单片机进行数字化处理，键盘进行设定值输入，显示其作为各种信息如设定值、操作界面等的显示。经过数字化处理的角度值由单片机与设定值进行比较，控制风扇转速达到控制摆杆角度的目的。同时，单片机可完成一些其他功能。

3.1 硬件电路的设计

由 TI 公司的 MSP430 单片机作为整个系统的控制核心。采用独立式键盘：使用角度传感器采集摆杆转角大小，使用伺服电机控制滚珠丝杠达到电磁铁的滑动。各个检测信号、控制信号、显示信号等由单片机的 I/O 口进行控制，同时可以方便直观的进行系统设置，并由程序保证系统抗干扰的能力。

传感器采集到信号后，送人单片机内进行 A/D 转换，使用键盘设定摆杆摆角，通过 PWM 驱动伺服电机转动及调速，此时摆杆摆角会相应变化，角度传感器继续采集信号至达到键盘设定的摆动角度为止。

硬件设计框图如下图 3-1 所示：

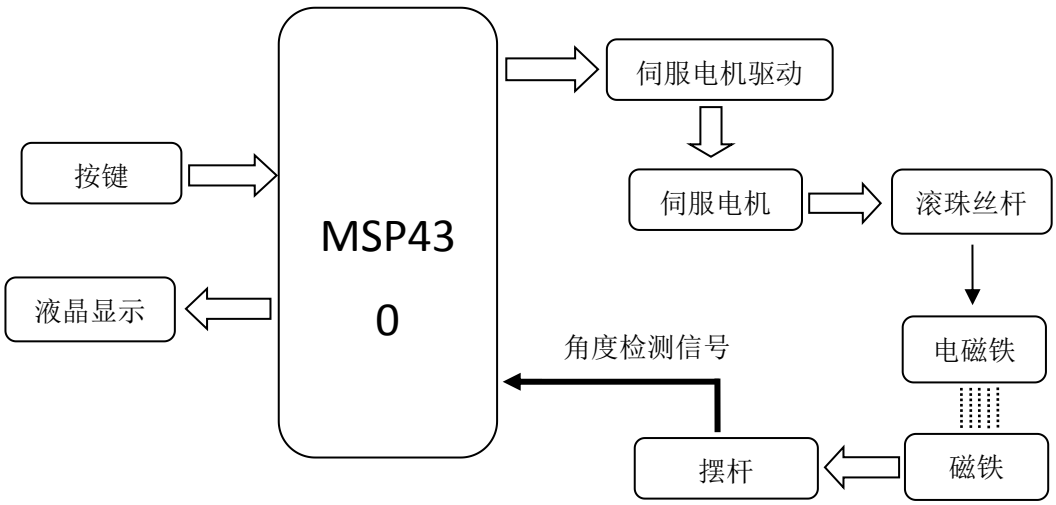


图 3-1 硬件设计框架

3.1.1 主控电路

以 MSP430 为控制核心，P1.0、P1.1、P1.3 口作为按键中断所用，P1.2 作为 PWM 电机转速控制，P2 口控制 Nokia5110lcd 显示，P3 口作为声光提示所用，实现帆板控制系统的自动控制。原理图如下图：

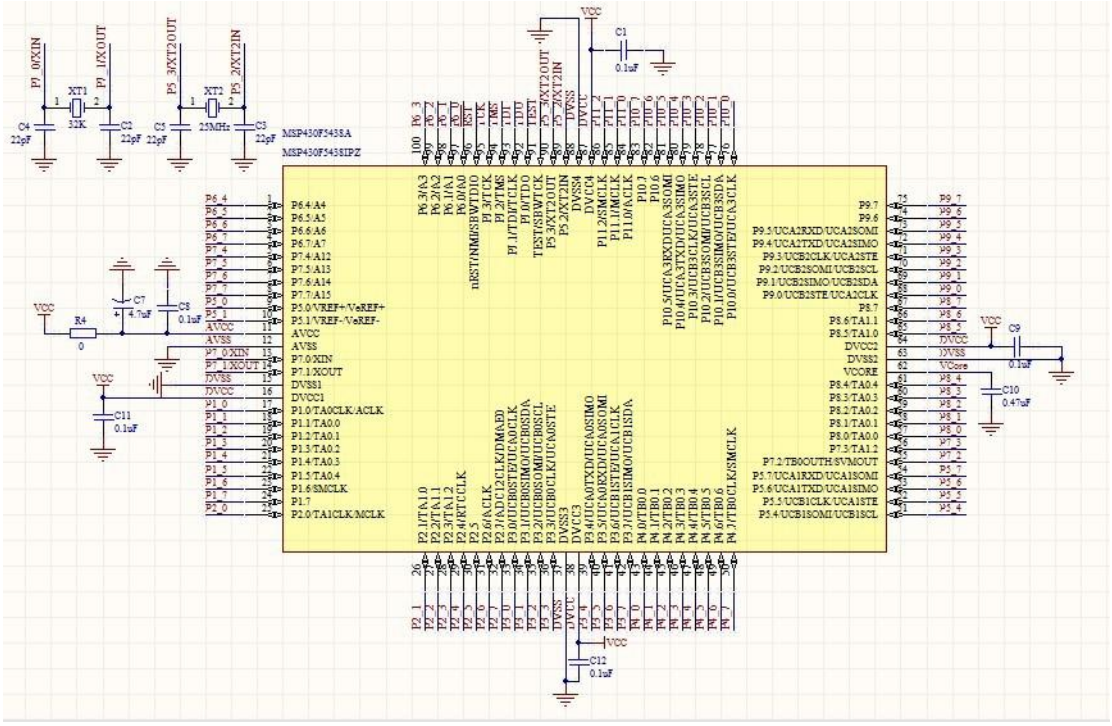


图 3-1-1 主控电路

3.1.2 角度检测子系统框图与电路原理图

为了实现帆板倾斜角度的精确测量，设计中选用飞思卡尔 MMA7361L 三轴加速度传感器作为角度测量的核心器件，MMA7361L 具有三路模拟量输出信号，Xout、Yout、Zout 脚分别是 X 轴、Y 轴和 Z 轴方向倾角的模拟电压输出脚，该设计仅测量一个方向的的倾角，因此仅需用 ADC0804 对一个输出脚的数据进行模数转换，将测得的数字量回传到单片机经过计算即可得到一个方向的倾斜角度
图 3-1-2(a)是角度检测模块的系统框图，图 3-1-2（b）是角度检测模块的硬件电路设计。

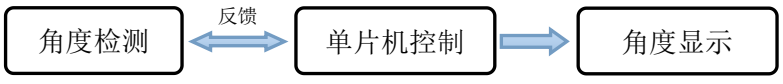


图 3-1-2(a)角度检测模块的系统框图

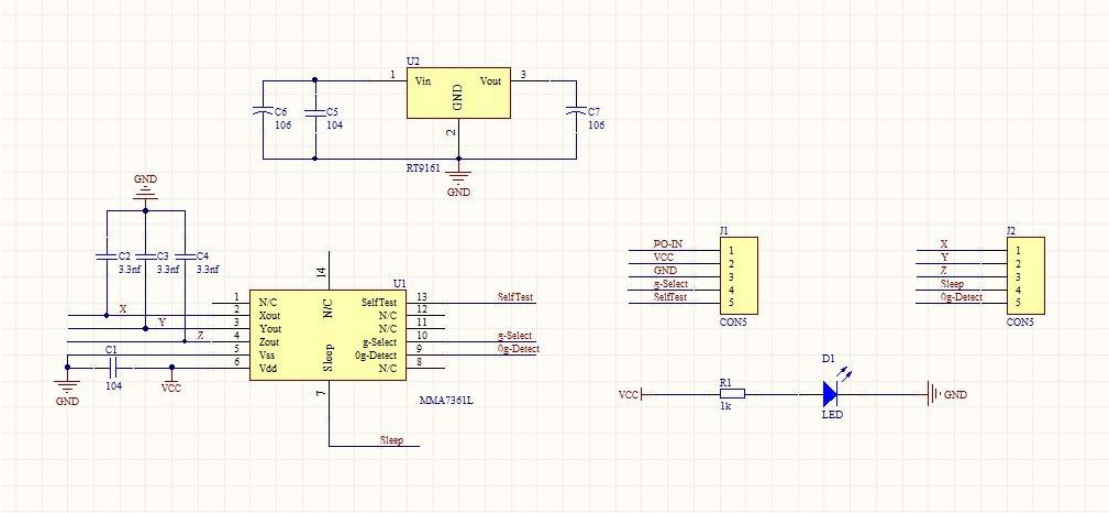


图 3-1-2（b）角度检测模块的硬件电路设计

3.1.3 按键及显示电路

由按键设定单片机的工作模式及帆板转动角度，并由液晶显示，液晶采用Cry12864LCD，其微功耗、超薄轻小，具有强大的显示功能。
图 3-1-3（a）为显示与声光控制的子系统框图，图 3-1-3（b）为液晶显示接口电路原理图

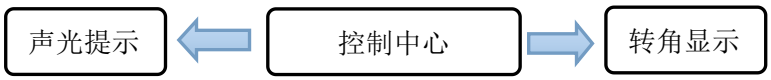


图 3-1-3（a）显示与声光控制的子系统框图

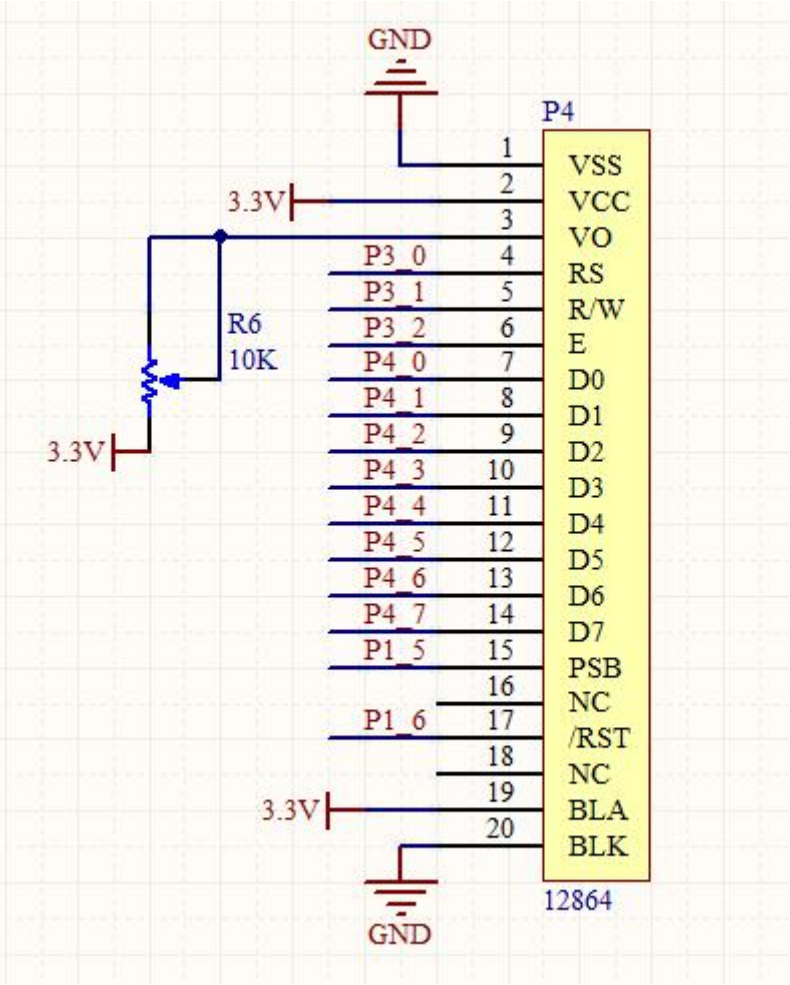


图 3-1-3（b）液晶显示接口电路原理图

3.1.4 总体电路图

见附录一

3.2 软件程序设计

3.2.1 程序功能描述与设计思路

1）按下启动按钮，由静止点开始，控制摆杆摆动。

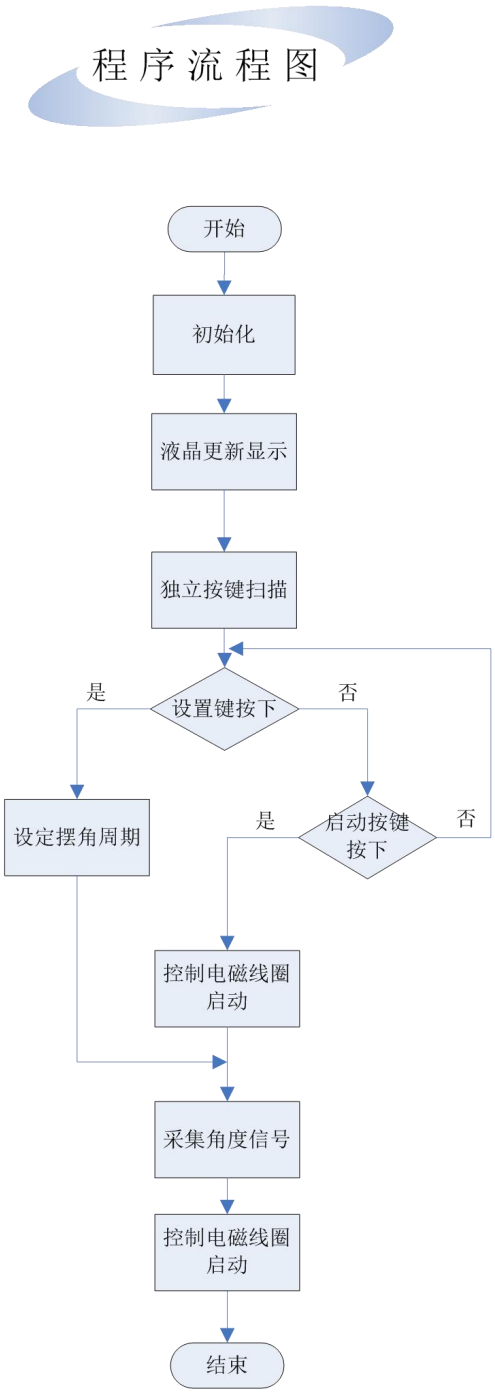
2）可以在固定摆角、摆动周期。

3）提供 12864 液晶显示和 5 位独立按键提供人机交互。

设计思路：

根据摆杆上的角度传感器接收角度信号并传送给单片机进行信号处理，通过处理后的信号对电磁线圈进行控制，从而控制摆杆的摆动。

3.2.2 程序流程图



3.2.3 源程序

见附录二

4.系统测试及结果分析

4.1 测试方案

4.1.1 硬件测试

1）硬件调试时，可先检查印制板及焊接的质量是否符合要求，有无虚焊点及线路间有无短路、断路。然后用万用表检测，检查无误后，可通电检查数码管以及键盘电路的好坏。

2）电磁驱动电路的测试，用数字示波器检测 PWM 信号是否正常，占空比是否可调，是否和轴流风扇转速相对应。

3）角度检测电路的测试，当转动角度传感器时，用数字示波器检测其输出是否有方波脉冲。

4.1.2 软件测试

软件调试是在 Keil 编译器下进行，源程序编译及仿真调试应分段或以子程序为单位逐个进行，最后结合硬件实时调试。子程序调试包括：PWM 脉宽调制子程序，角度检测子程序，显示电路子程序，键扫描子程序等。

4.2 测试条件与仪器

4.2.1 测试条件

检查多次，仿真电路和硬件电路必须与系统原理图完全相同，并且检查无误，硬件电路保证无虚焊。

4.2.2 测试仪器

本系统需要使用以下仪器进行功能调试与性能测试。如下表4.2.2所列：

表4.2.2测试使用仪器

仪器名称	用途	数量	备注
数字万用表	检查电路是否短路	1	
数字示波器	检查输出波形是否正确	1	
量角器	测量角度	1	
直尺	测量距离	1	量程0—30cm
秒表	测量时间	1	

4.3 测试结果与分析

4.3.1 测试结果（数据）

测试数据如下表 4.3.1（a）（b）所列（角度设以及摆动周期为随机值）：

表 4.3.1（a）摆杆在指定的摆角摆动

设置角度大小(单位:度)	10	13	19	21	30	35	40	42	45
实际角度大小(单位:度)	10	13	19	21	30	36	41	43	46
摆角绝对误差	0	0	0	0	0	+1	+1	+1	+1
响应时间（单位：秒）	5	6	6	6	9	9	10	10	12

表 4.3.1（b）摆杆在指定的摆角摆动

设置周期值(单位:秒)	0.5	0.8	1.2	1.6	1.8	2
实际周期值(单位:秒)	0.8	0.13	1.5	1.9	2	2.5
摆动周期绝对误差	+0.3	+0.5	+0.3	+0.3	+0.2	+0.5
响应时间（单位：秒）	8	9	7	10	10	12

4.3.2 测试分析

误差分析：通过以上两张表格的数据分析可知，输入的角度与实际测出的角度存在一定的误差，但误差的大小控制在范围之内。产生误差的可能因素存在很多，可能是人为产生的误差，可能是转轴与横杠之间的摩擦，也可能是整体系统构架上存在误差。

结论：本控制系统的基本部分和发挥部分的要求都能够实现，测出的实际数据与理论值存在误差，误差的范围在要求之内。通过系统的整体分析，难点在于对帆板实际偏转的角度反馈给单片机，然后单片机再分析，从而控制电磁铁的磁性大小。这部分的程序设计需要用到 PID 算法，PID 算法的实现对 PWM 的调制起着关键的作用，PID 程序算法与 PWM 的结合是完成系统设计的难点，通过这几天的调试，软件程序与硬件系统达到了很好的结合，所以也就比较顺利完成了系统设计要求。

测试结果与分析如表 4.3.2 所列。

表 4.3.2 测试结果与分析

调试项目	调试结果
由静止点开始，控制摆杆在指定的摆角（10°～45°范围内）连续摆动，摆动摆角绝对误差≤5°，响应时间≤15s。	能够在指定的摆角内摆动，摆动摆角绝对误差均在允许范围内
在摆杆连续摆动的情况下，按下停止按钮，控制摆杆平稳地停在静止点上，停止时间≤10s。	可以在 10 s 内平稳地停在静止点

5.总结

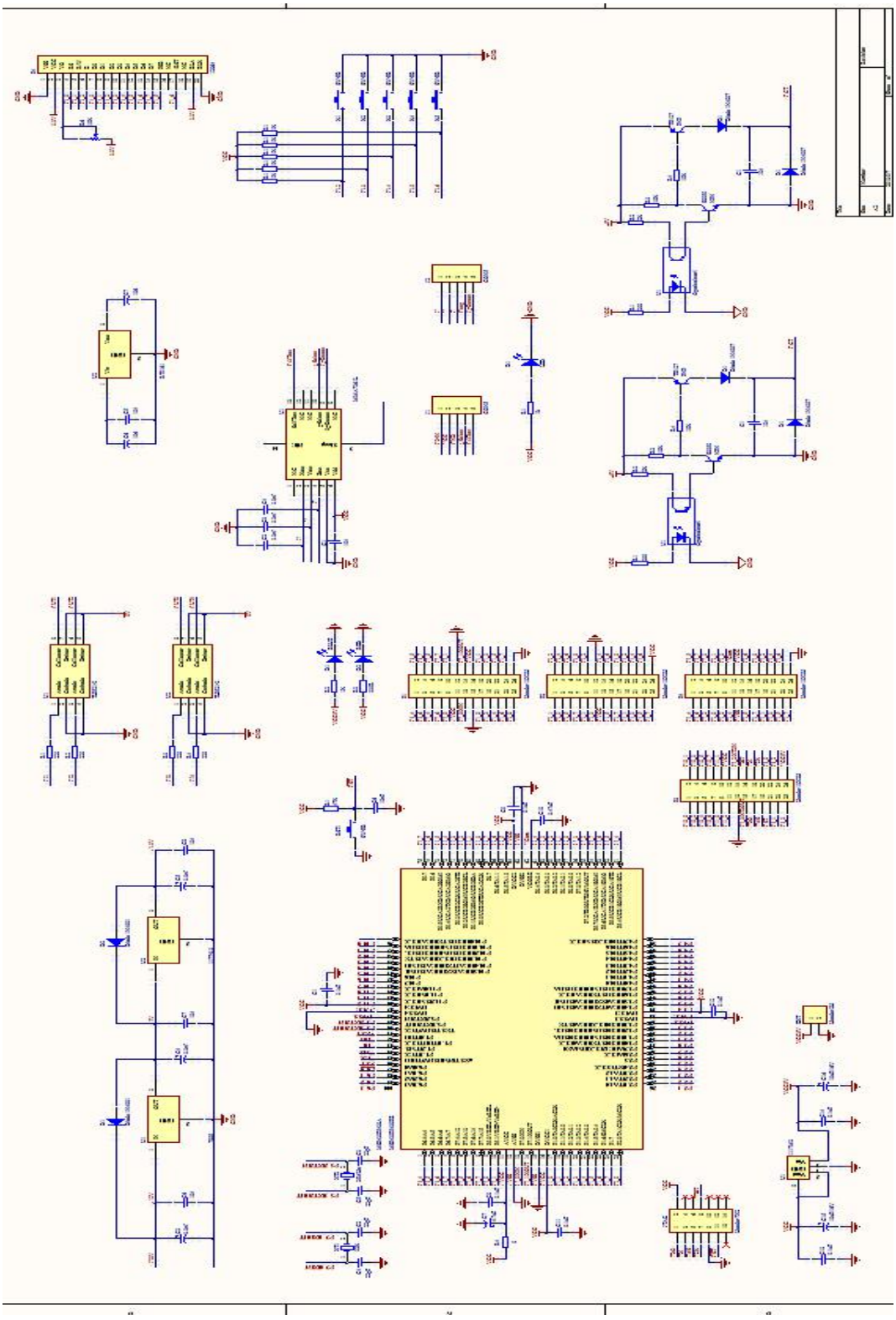
综合上述的测试结果，本设计完成了题目的基本要求，使用的 MSP430 和 MMA7361 角度传感器大大的提高了本设计的灵敏度和精准度。经过紧张激烈的四天三夜我们终于完成了本次设计，通过这次比赛，我们学到了很多东西，对以后的人生有着相当重要的意义。而且提高了我们的创新精神，动手能力，团队协作和竞争意识，这些在今后的人生道路上将是一笔宝贵的财富。充分发挥团队合作精神，工作进展顺利。

最后，十分感谢学校的培养和全国电子设计竞赛组委会给予我们这次锻炼的机会。这必将是我们人生中最宝贵的“财富”。

参考文献：

[1]张正伟.传感器原理与应用.北京：中央广播电视大学出版社,1991
[2]黄智伟.全国大学生电子设计竞赛电路设计第二版,2011
[3]沈建华 杨艳琴. MSP430系列16位超低功耗单片机原理与实践,2008
[4]黄根春 周立青 张望先.全国大学生电子设计竞赛教程—基于 TI 器件设计方法.电子工业出版社,2011.4
[4]胡寿松.自动控制原理第四版.北京：科学出版社,2001
[5]杨素行.模拟电子技术基础简明教程.北京：高等教育出版社,1998

附录 1：总体电路图



附录 2：源程序

```
/************************************************************
程序功能：在 12864 液晶上显示 ASCII 常用字符
------------------------------------------------------------
测试说明：观察液晶显示
************************************************************/
#include   "msp430x14x.h"
#include   "Cry12864.h"
#include   "KEY_SCAN.h"
#include   <math.h>
#define CPU_F ((double)8000000)        //定义 CPU 的时钟频率为 8M
#define  delay_us(x) __delay_cycles((long)(CPU_F*(double)x/1000000.0))// 定义延时 1us
#define delay_ms(x) __delay_cycles((long)(CPU_F*(double)x/1000.0))     // 定义延时 1ms


#define uchar unsigned char
#define uint   unsigned int

//ADXL345 控制口，SPI3 线传输
#define csout P5DIR |= BIT0
#define cs1 P5OUT |= BIT0
#define cs0 P5OUT &= ~BIT0
#define sclout P5DIR |= BIT2
#define scl1 P5OUT |= BIT2
#define scl0 P5OUT &= ~BIT2
#define sdaout P5DIR |= BIT1
#define sdain P5DIR &= ~BIT1
#define sda1 P5OUT |= BIT1
#define sda0 P5OUT &= ~BIT1
#define sdai P5IN&BIT1
int mr[2];   //读取数据储存
#define Magnetic_Pole_H
#define Magnetic_Pole_L

/************************************************************///默认显示值
uchar Test_Code[16] = {"      .                  "};
uchar First_Line[16] ={"电磁控制运动装置"};
uchar Second_Line[16]={"设定角度: 20   度"};     //10 11
uchar Third_Line[16] ={"设定周期: 1.0  秒"};     //10 12
```

```
uchar Fourth_Line[16]={"运行状态: 停止   "};     //10-13
uchar OFF[]={"停止"};
uchar ON[] ={"运行"};
uchar Num_Code[]={"0123456789"};
/************************************************************/
int Angle;                              // 摆角 单位 0.1°  100<-->10 °
int   Angle_temp;
int Angle_temp2;
uchar Cycle;                             // 周期 单位 0.1 秒 100<-->10.0 S
uchar Power_Status,cai;                    //工作状态 1：ON 运行   0：OFF 停止

uint Set_Angle;             //设定角度 范围 100-450
uchar Set_Cycle;             //设定周期   范围 5-20

uchar Setting_Mode;
uchar Key_Change;

uchar Angle_DIR;   //角度方向   左转偏转为 1      右侧偏转为 2      静止为 0
int PWM_A=0;
/************************************************************/
void Init_Disp();                       //初始化显示
void UPdat_Dis();                        //显示数据更新
void Key_Scan();                         //按键扫描
void Init_Main();                        //主初始化


void Start();                           //通用启动程序
void Stop();                             //通用停止程序
/************************************************************/
void clock_8m()     //转换晶振 8mhz
{
   uchar i;
   BCSCTL1 &= ~XT2OFF;
   BCSCTL2 |= SELM1+SELS;
   do
   {
      IFG1 &= ~OFIFG;
      for(i=0;i<100;i++)
         _NOP();
   }
   while((IFG1&OFIFG)!=0);
   IFG1 &= ~OFIFG;
}
```

```
void Init_Disp()
{uchar i;
      Write_Cmd(0x80);                    //写第一行的显示地址
        for(i = 0; i < 16; i++)
              Write_Data(First_Line[i]);
  //    Write_Data(Test_Code[i]);
            Delay_Nms(1);
      Write_Cmd(0x90);                    //写第二行的显示地址
        for(i = 0; i < 16; i++)
             Write_Data(Second_Line[i]);        //显示 0x40~0x4f 对应的字符
      Delay_Nms(1);
      Write_Cmd(0x88);                    //写第三行的显示地址
        for(i = 0; i < 16; i++)
            Write_Data(Third_Line[i]);        //显示 0x50~0x5f 对应的字符
      Delay_Nms(1);
      Write_Cmd(0x98);                    //写第四行的显示地址
        for(i = 0; i < 16; i++)
            Write_Data(Fourth_Line[i]);        //显示 0x60~0x6f 对应的字符
      Delay_Nms(1);
}
void UPdat_Dis()
{
   if(Key_Change)
   {        Key_Change=0;
      Second_Line[10]=Num_Code[Set_Angle%100/10];        //10 角度显示位
      Second_Line[11]=Num_Code[Set_Angle%10];             //1 角度显示位
      Third_Line[10] =Num_Code[Set_Cycle%100/10];        //1 秒显示位
      Third_Line[12] =Num_Code[Set_Cycle%10];             //0.1 秒显示位
      if(Power_Status)                                       //运行状态
        {
          Fourth_Line[10]=ON[0];
          Fourth_Line[11]=ON[1];
          Fourth_Line[12]=ON[2];
          Fourth_Line[13]=ON[3];
        }
      else
        {
          Fourth_Line[10]=OFF[0];
          Fourth_Line[11]=OFF[1];
          Fourth_Line[12]=OFF[2];
          Fourth_Line[13]=OFF[3];
        }

        Init_Disp();//显示数据更新
```

```
   }
}
void Key_Scan()
{uchar Key_Val;

   Key_Val=Scan();
   if(Key_Val)
     {
        switch(Key_Val)
        {
        case 1: Start();
           break;   //启动
        case 2: Power_Status=0;
           break;//停止
        case 3:
           {
              if(!Power_Status)
              {   Setting_Mode++;
              if(Setting_Mode>=3)
                 Setting_Mode=0;
              }

           }
           break;//设置     0 未设置模式   1  设置角度模式     2  设置周期模式
        case 4:
           {
              if(Setting_Mode==1)
                    Set_Angle+=5;
              else if(Setting_Mode==2)
                    Set_Cycle+=1;

              if(Set_Angle>=45)
                    Set_Angle=45;
              if(Set_Cycle>=20)
                    Set_Cycle=20;

           }
           break; //增加
        case 5:
           {
              if(Setting_Mode==1)
                    Set_Angle-=5;
              else if(Setting_Mode==2)
                    Set_Cycle-=1;
```

```
            if(Set_Angle<=5)
                Set_Angle=5;
            if(Set_Cycle<=5)
                Set_Cycle=5;

          }
          break; //减少
        }
      Key_Change=1;
      }

UPdat_Dis();
   Delay_Nms(5);
          if(Setting_Mode==1)
            { Write_Cmd(0x0f);
              Write_Cmd(0x95);
                        //整体显示开，游标开，游标位置开
            }
          else if(Setting_Mode==2)
            {
              Write_Cmd(0x0f);
              Write_Cmd(0x8d);
                          //整体显示开，游标开，游标位置开
            }
            else
          {
                        //位址归位，游标回到原点
              Write_Cmd(0x0c);                  //整体显示开，游标关，游标位置
关
           }

}

void trans_data(uchar adds,uchar dat,uchar n)   // 3 线 SPI 传输（地址，数据，输
出位数）
{
   uchar i,j,d;                                          //读数据时，数据为 0，n!=0,
写 0x00 时，
                                                            //dat=0,n=0
   csout;sclout;sdaout;
   cs0;scl1;
   delay_us(2);
   d=adds;
```

```
if((dat==0)&&(n!=0))
   d += 0x80;
if(n>1)
   d += 0x40;
for(i=0;i<8;i++)
{
   scl0;
   if((d&0x80)==0x80)
      sda1;
   else
      sda0;
   delay_us(2);
   scl1;
   d<<=1;
   delay_us(2);
}
d=dat;
if((dat==0)&&(n!=0))
{
   sdain;
   for(j=0;j<n;j++)
   {
      for(i=0;i<8;i++)
      {
        scl0;
        delay_us(2);
          d<<=1;
          if(sdai)
              d++;
          scl1;
          delay_us(2);
      }
      mr[j]=d;
   }
}
else
{
   d=dat;
   for(i=0;i<8;i++)
   {
      scl0;
      delay_us(2);
      if((d&0x80)==0x80)
          sda1;
```

```
            else
               sda0;
            scl1;
            d<<=1;
            delay_us(2);
        }
    }
    delay_us(5);
    cs1;
}
void Angle_Convert()
{


    float Q,Roll;
    int Roll_temp;//左侧偏转为 1      右侧偏转为 2      静止为 0

       trans_data(0x34,0,2);            //数据传输  ,接收数据只使用 Y 轴数据

       mr[0]=mr[0]+(mr[1]<<8);                    //合并数据
       mr[0]*=3.9;
       mr[0]-=50;                                  //误差补偿   1000<-->1.000

    if(mr[0]<0)                                   //符号处理
       {
            Angle_DIR=2;
            Test_Code[0]='-';
       }
    else
    {
            Angle_DIR=1;
            Test_Code[0]=' ';
    }

    Q=(float)mr[0]/1000.0;            // 数值转换

    if(Q>1)                     //限位
          Q=1;
    else if(Q<-1)
          Q=-1;

    if(Q<0)
          Q=-Q;
```

```
    Roll=(float)(asin(Q)*180.0/3.141592653);



    Roll*=10.0;
    Roll_temp= (unsigned int) Roll;
    Test_Code[1]=Roll_temp%1000/100+48;
    Test_Code[2]=Roll_temp%100/10+48;
    Test_Code[4]=Roll_temp%10/1+48;

    Test_Code[15]=Num_Code[Angle_DIR];

     if(Angle_DIR==2)   //右侧
        Angle=Roll_temp;
     else
          Angle=-Roll_temp;

  //   Key_Change=1;
  //   delay_ms(100);

}

void Init_PWM_OUT()
{

   TA0CTL |= TASSEL_1 + TACLR; //ACLK + CLR;
   TA0CCR0 = 200-1; //PWM 周期
   TA0CCTL1 |= OUTMOD_6;//输出模式 2

   TA0CCR1 = PWM_A ;//占空比
   TA0CTL |= MC_1;



}
void Init_Main ()
{
     P1DIR |= 0X60;          //液晶显示复位
     P1OUT |= 0X60;

     Set_Angle=20;                        // 摆角 范围 5-45 单位 °      默认
值 10
     Set_Cycle=10;                        // 周期 范围 5-20 单位 0.1 秒    默
认 0.5
```

```
    Power_Status=0;                    //工作状态 1：ON 运行   0：OFF 停止 默
认 OFF
    Setting_Mode=0;
    Key_Change=0;

    P1DIR|=BIT2;
    P1OUT&=~BIT2;

    delay_us(50);
}

void Run()
{
  uchar TEMP;

/* if(cai ==0)
  Angle_temp=Angle;
  Angle_Convert();
 if(cai++>1)
 {
    Angle_temp2=Angle;
    cai=0;
  }
 */

  Angle_temp=Angle;
  Angle_Convert();
  Angle_temp2=Angle;



    if(Angle_temp2-Angle_temp>10)
      TEMP=1;   //1 为左趋向
    else if (Angle_temp2-Angle_temp<-10)
      TEMP=2;   //2 为右趋向
    else TEMP=0;

    if((Angle<10&&Angle>-10) &&TEMP == 0)
    {P1SEL|=BIT2;}   // up
     else if (TEMP==0)
     {P1SEL&=~BIT2;P1OUT&=~BIT2;}


      if ((Angle>160)||(Angle<-160))
```

```
            {P1SEL&=~BIT2;P1OUT&=~BIT2;}      //down
          else    if((TEMP==1&&Angle>40)||(TEMP==2&&Angle<-40))
           {P1SEL|=BIT2;}    // up

       // else      P1OUT&=~BIT2;//down


}
void Start()
{unsigned int Temp;

      Power_Status=1;Setting_Mode=0;//数据 初始化进行

      if(Set_Angle<35)
      {
         Temp=(Set_Angle-4);
         Temp=Temp*21;
         Temp=Temp+30;
         Temp/=10;
         PWM_A=Temp;


      }
      else
         PWM_A=Set_Angle*2+8;                    //down time 180 度


    if(PWM_A>200)
           PWM_A=200;
    if(PWM_A<14)
           PWM_A=14;
      TA0CCR1=PWM_A;
 //       TA0CCR1=200;
      P1DIR|=BIT2;
      P1SEL&=~BIT2;
      P1OUT|=BIT2;
      delay_us(10000);
}
void Stop()
{
      P1DIR|=BIT2;
      P1OUT&=~BIT2;
      P1SEL&=~BIT2;
```

```
//反推力程序待定
}
void Work()
{
   if(Power_Status==1)     //运行状态
   {
      Run();
   }
   else     //运行状态
   {
      Stop();
   }

}
   /*
void Start()
{
   uchar TEMP;
   Angle_temp=Angle;
      Angle_Convert();

     if(Angle-Angle_temp>0)
        TEMP=1;   //1 为左趋向
      else
        TEMP=2;   //2 为右趋向

        if(TEMP==1&&Angle>0)
          P1OUT|=BIT2;
        else   if(TEMP==1&&Angle<0)
              P1OUT&=~BIT2;
                 else   if(TEMP==2&&Angle>0)
                       P1OUT&=~BIT2;
                          else   if(TEMP==2&&Angle<0)
                             P1OUT|=BIT2;
                                  else
                                     P1OUT&=~BIT2;

}*/
/*
void Start()
{
      if(Angle<50)
        P1OUT|=BIT2;
```

```
      else
         P1OUT&=~BIT2;

}*/
/*************************主函数*************************/
void main( void )
{

    cai= 0;
      WDTCTL = WDTPW + WDTHOLD;       //关狗
      clock_8m();
      Init_Main();                          //I/0  变量初始化
      Init_Lcd();                          //初始化液晶
      Init_Disp();                         //初始化显示数据
      Init_PWM_OUT();
  trans_data(0x31,0x6b,0);//0110 1011
  trans_data(0x2d,0x08,0);
  trans_data(0x2e,0x80,0);
  Angle_Convert();
      while(1)
      {
         Key_Scan();

        Work();

      }

}
```