第三十四章 三轴加速度传感器实验

自从有了 Iphone，各种新技术的普及程度越来越快，人们喜欢的不再是摔不坏的诺基亚，而是用户体验极佳的 Iphone。

本章，我们介绍一种当今智能手机普遍具有的传感器：加速度传感器。在手机上，这个功能可以用来：自动切换横竖屏、玩游戏和切歌等。ALIENTEK 战舰 STM32 开发板自带了加速度传感器：ADXL345。本章我们将使用 STM32 来驱动 ADXL345，读取 3 个方向的重力加速度值，并转换为角度，显示在 TFTLCD 模块上。本章分为如下几个部分：

34.1 ADXL345 简介

34.2 硬件设计

34.3 软件设计

34.4 下载验证

34.1 ADXL345 简介

ADXL345 是 ADI 公司的一款 3 轴、数字输出的加速度传感器。ADXL345 是 ADI 公司推出的基于 iMEMS 技术的 3 轴、数字输出加速度传感器。该加速度传感器的特点有：

- 分辨率高。最高 13 位分辨率。
- 量程可变。具有+/-2g，+/-4g，+/-8g，+/-16g 可变的测量范围。
- 灵敏度高。最高达 3.9mg/LSB，能测量不到 1.0°的倾斜角度变化。
- 功耗低。40~145uA 的超低功耗，待机模式只有 0.1uA。
- 尺寸小。整个 IC 尺寸只有 3mm*5mm*1mm，LGA 封装。

ADXL 支持标准的 I2C 或 SPI 数字接口，自带 32 级 FIFO 存储，并且内部有多种运动状态检测和灵活的中断方式等特性。ADXL345 传感器的检测轴如图 34.1.1 所示：

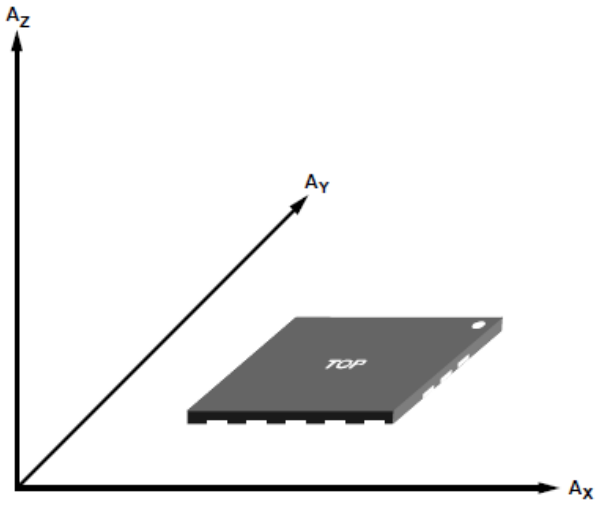

图 34.1.1 ADXL345 的三个检测轴

当 ADXL345 沿检测轴正向加速时，它对正加速度进行检测。在检测重力时用户需要注意，当检测轴的方向与重力的方向相反时检测到的是正加速度。图 33.1.2 所示为输出对重力的响应。

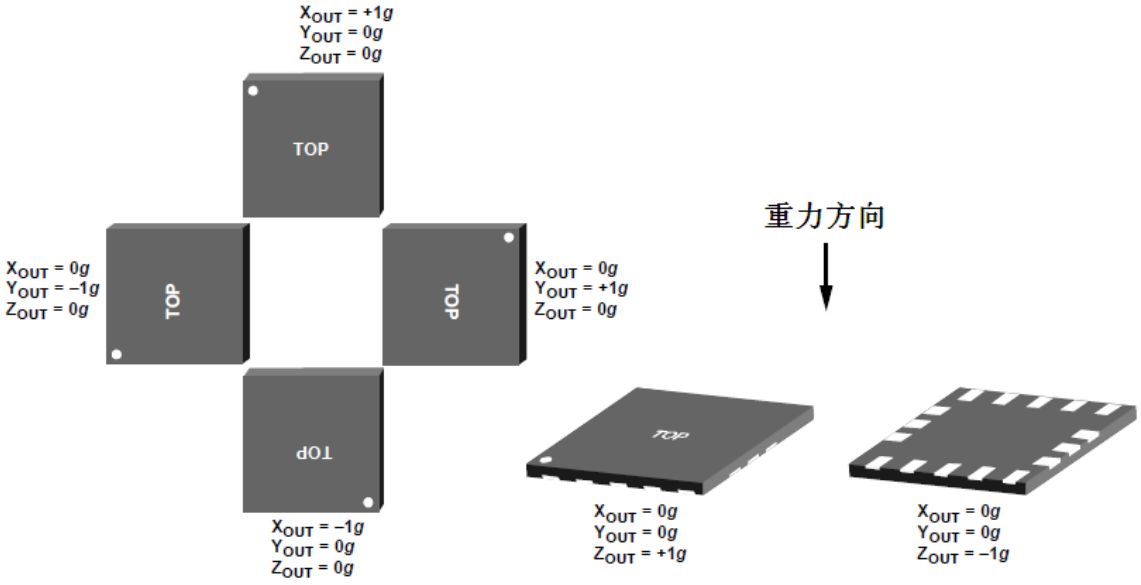

图 34.1.2 ADXL345 输出对重力的响应

图 34.1.2 列出了 ADXL345 在不同摆放方式时的输出，以便后续分析。接下来我们看看 ADXL345 的引脚图，如图 34.1.3 所示：

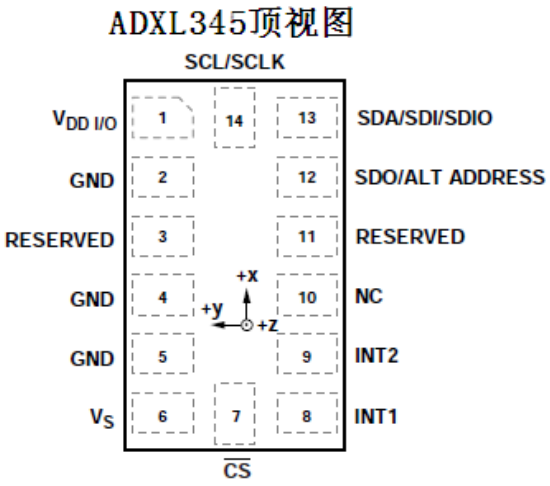

图 34.1.3 ADXL345 引脚图

ADXL345 支持 SPI 和 IIC 两种通信方式，为了节省 IO 口，战舰 STM32 开发板采用的是 IIC 方式连接，官方推荐的 IIC 连接电路如图 34.1.4 所示：

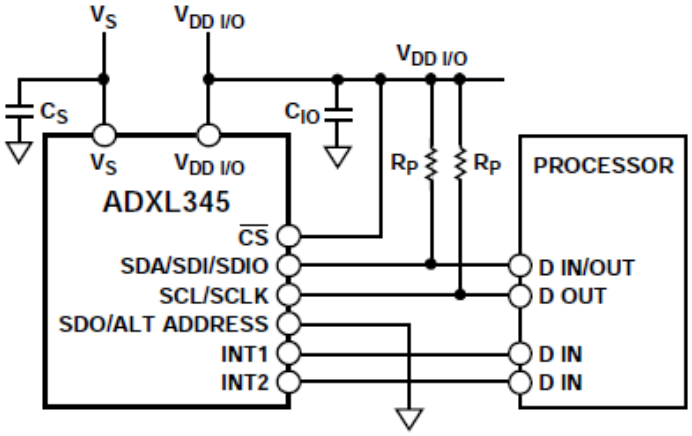

图 34.1.4 ADXL345 IIC 模式连接电路

从上图可看出，ADXL345 的连接十分简单，外围需要的器件也极少（就 2 个电容），如上连接（SDO/ALT ADDRESS 接地），则 ADXL345 的地址为 0X53（不含最低位），如果 SDO/ALT ADDRESS 接高，那么 ADXL345 的地址将变为 0X1D（不含最低位）。IIC 通信的时序我们在之前已经介绍过（第二十七章，IIC 实验），这里就不再细说了。

最后，我们介绍一下 ADXL345 的初始化步骤。ADXL345 的初始化步骤如下：

1）上电

2）等待 1.1ms

3）初始化命令序列

4）结束

其中上电这个动作发生在开发板第一次上电的时候，在上电之后，等待 1.1ms 左右，就可以开始发送初始化序列了，初始化序列一结束，ADXL345 就开始正常工作了。这里的初始化序列，最简单的只需要配置 3 个寄存器，如表 34.1.1 所示：

步骤	寄存器地址	寄存器名字	寄存器值	功能描述
1	0X31	DATA_FORMAT	0X0B	±16g，13 位模式
2	0X2D	POWER_CTL	0X08	测量模式
3	0X2E	INT_ENABLE	0X80	使能 DATA_READY 中断

表 34.1.1 ADXL345 最简单的初始化命令序列

发送以上序列给 ADXL345 以后，ADXL345 即开始正常工作。

ADXL345 我们就介绍到这里，详细的介绍，请参考 ADXL345 的数据手册。

34.2 硬件设计

本实验采用 STM32 的 3 个普通 IO 连接 ADXL345，本章实验功能简介：主函数不停的查询 ADXL345 的转换结果，得到 x、y 和 z 三个方向的加速度值（读数值），然后将其转换为与自然系统坐标的角度，并将结果在 LCD 模块上显示出来。DS0 来指示程序正在运行，通过按下 WK_UP 按键，可以进行 ADXL345 的自动校准（DS1 用于提示正在校准）。

所要用到的硬件资源如下：

1） 指示灯 DS0、DS1

2） WK_UP 按键

3） TFTLCD 模块

4） ADXL345

前 3 个，在之前的实例已经介绍过了，这里我们仅介绍 ADXL345 与战舰 STM32 开发板的连接。该接口与 MCU 的连接原理图如 34.2.1 所示：

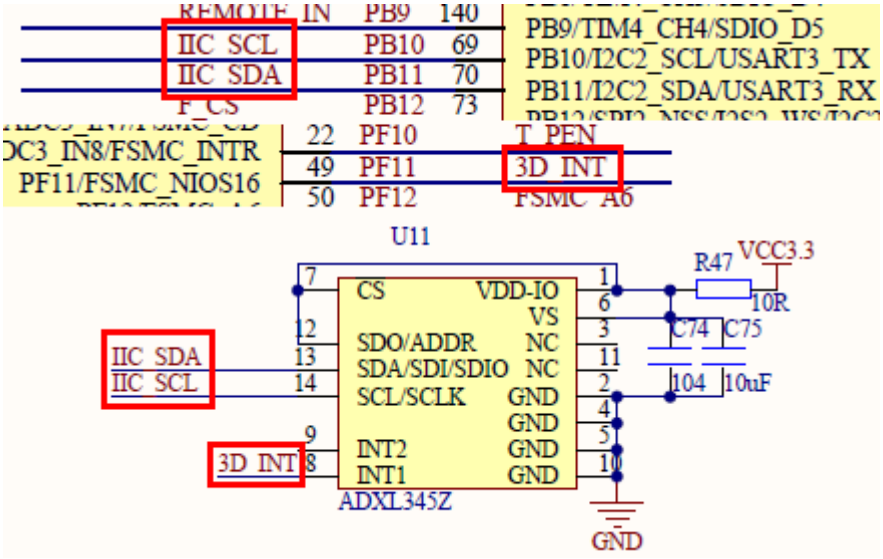

图 34.2.1 ADXL345 与 STM32 的连接电路图

从上图可以看出，ADXL345 通过三根线与 STM32 开发板连接，其中 IIC 总线时和 24C02 以及 RDA5820 共用，接在 PB10 和 PB11 上面。ADXL345 的两个中断输出，这里我们只用了一个，连接在 STM32 的 PF11 脚，另外这里的地址线是接 3.3V，所以 ADXL345 的地址是 0X1D，转换为 0X3A 写入，0X3B 读取。

34.3 软件设计

打开上一章的工程，首先在 HARDWARE 文件夹下新建一个 ADXL345 的文件夹。然后新建一个 adxl345.c 和 adxl345.h 的文件保存在 JOYPAD 文件夹下，并将这个文件夹加入头文件包含路径。

打开 adxl345.c 文件，输入如下代码：

```
#include "adxl345.h"
#include "sys.h"
#include "delay.h"
#include "math.h"
//初始化 ADXL345.
//返回值:0,初始化成功;1,初始化失败.
u8 ADXL345_Init(void)
{
    IIC_Init();                                    //初始化 IIC 总线
    if(ADXL345_RD_Reg(DEVICE_ID)==0XE5)   //读取器件 ID
    {
        ADXL345_WR_Reg(DATA_FORMAT,0X2B);
        //低电平中断输出,13 位全分辨率,输出数据右对齐,16g 量程
        ADXL345_WR_Reg(BW_RATE,0x0A);          //数据输出速度为 100Hz
        ADXL345_WR_Reg(POWER_CTL,0x28);        //链接使能,测量模式
        ADXL345_WR_Reg(INT_ENABLE,0x00);       //不使用中断
        ADXL345_WR_Reg(OFSX,0x00);
        ADXL345_WR_Reg(OFSY,0x00);
        ADXL345_WR_Reg(OFSZ,0x00);
        return 0;
    }
    return 1;
}
//写 ADXL345 寄存器
//addr:寄存器地址
//val:要写入的值
//返回值:无
void ADXL345_WR_Reg(u8 addr,u8 val)
{
    IIC_Start();
    IIC_Send_Byte(ADXL_WRITE);        //发送写器件指令
    IIC_Wait_Ack();
    IIC_Send_Byte(addr);                   //发送寄存器地址
    IIC_Wait_Ack();
    IIC_Send_Byte(val);                    //发送值
    IIC_Wait_Ack();
```

```
    IIC_Stop();                         //产生一个停止条件
}
//读 ADXL345 寄存器
//addr:寄存器地址
//返回值:读到的值
u8 ADXL345_RD_Reg(u8 addr)
{
    u8 temp=0;
    IIC_Start();
    IIC_Send_Byte(ADXL_WRITE); //发送写器件指令
    temp=IIC_Wait_Ack();
    IIC_Send_Byte(addr);            //发送寄存器地址
    temp=IIC_Wait_Ack();
    IIC_Start();                    //重新启动
    IIC_Send_Byte(ADXL_READ);  //发送读器件指令
    temp=IIC_Wait_Ack();
    temp=IIC_Read_Byte(0);          //读取一个字节,不继续再读,发送 NAK
    IIC_Stop();                     //产生一个停止条件
    return temp;                    //返回读到的值
}
//读取 ADXL 的平均值
//x,y,z:读取 10 次后取平均值
void ADXL345_RD_Avval(short *x,short *y,short *z)
{
    short tx=0,ty=0,tz=0;
    u8 i;
    for(i=0;i<10;i++)
    {
        ADXL345_RD_XYZ(x,y,z);
        delay_ms(10);
        tx+=(short)*x; ty+=(short)*y; tz+=(short)*z;
    }
    *x=tx/10; *y=ty/10; *z=tz/10;
}
//自动校准
//xval,yval,zval:x,y,z 轴的校准值
void ADXL345_AUTO_Adjust(char *xval,char *yval,char *zval)
{
    short tx,ty,tz;
    u8 i;
    short offx=0,offy=0,offz=0;
    ADXL345_WR_Reg(POWER_CTL,0x00);     //先进入休眠模式.
    delay_ms(100);
```

```
        ADXL345_WR_Reg(DATA_FORMAT,0X2B);
        //低电平中断输出,13 位全分辨率,输出数据右对齐,16g 量程
        ADXL345_WR_Reg(BW_RATE,0x0A);          //数据输出速度为 100Hz
        ADXL345_WR_Reg(POWER_CTL,0x28);        //链接使能,测量模式
        ADXL345_WR_Reg(INT_ENABLE,0x00);       //不使用中断
        ADXL345_WR_Reg(OFSX,0x00);
        ADXL345_WR_Reg(OFSY,0x00);
        ADXL345_WR_Reg(OFSZ,0x00);
        delay_ms(12);
        for(i=0;i<10;i++)
        {
            ADXL345_RD_Avval(&tx,&ty,&tz);
            offx+=tx; offy+=ty; offz+=tz;
        }
        offx/=10; offy/=10; offz/=10;
        *xval=-offx/4; *yval=-offy/4; *zval=-(offz-256)/4;
        ADXL345_WR_Reg(OFSX,*xval);
        ADXL345_WR_Reg(OFSY,*yval);
        ADXL345_WR_Reg(OFSZ,*zval);
    }
    //读取 3 个轴的数据
    //x,y,z:读取到的数据
    void ADXL345_RD_XYZ(short *x,short *y,short *z)
    {
        u8 buf[6],i;
        IIC_Start();
        IIC_Send_Byte(ADXL_WRITE); //发送写器件指令
        IIC_Wait_Ack();
        IIC_Send_Byte(0x32);           //发送寄存器地址(数据缓存的起始地址为 0X32)
        IIC_Wait_Ack();
        IIC_Start();                   //重新启动
        IIC_Send_Byte(ADXL_READ);  //发送读器件指令
        IIC_Wait_Ack();
        for(i=0;i<6;i++)
        {
            if(i==5)buf[i]=IIC_Read_Byte(0); //读取一个字节,不继续再读,发送 NACK
            else buf[i]=IIC_Read_Byte(1);    //读取一个字节,继续读,发送 ACK
        }
        IIC_Stop();                    //产生一个停止条件
        *x=(short)(((u16)buf[1]<<8)+buf[0]);
        *y=(short)(((u16)buf[3]<<8)+buf[2]);
        *z=(short)(((u16)buf[5]<<8)+buf[4]);
    }
```

```
//读取 ADXL345 的数据 times 次,再取平均
//x,y,z:读到的数据
//times:读取多少次
void ADXL345_Read_Average(short *x,short *y,short *z,u8 times)
{
	u8 i;
	short tx,ty,tz;
	*x=0; *y=0; *z=0;
	if(times)//读取次数不为 0
	{
		for(i=0;i<times;i++)//连续读取 times 次
		{
			ADXL345_RD_XYZ(&tx,&ty,&tz);
			*x+=tx; *y+=ty; *z+=tz;
			delay_ms(5);
		}
		*x/=times; *y/=times; *z/=times;
	}
}
//得到角度
//x,y,z:x,y,z 方向的重力加速度分量(不需要单位,直接数值即可)
//dir:要获得的角度.0,与 Z 轴的角度;1,与 X 轴的角度;2,与 Y 轴的角度.
//返回值:角度值.单位 0.1°.
short ADXL345_Get_Angle(float x,float y,float z,u8 dir)
{
	float temp,res=0;
	switch(dir)
	{
		case 0://与自然 Z 轴的角度
			temp=sqrt((x*x+y*y))/z;
			res=atan(temp);
			break;
		case 1://与自然 X 轴的角度
			temp=x/sqrt((y*y+z*z));
			res=atan(temp);
			break;
		case 2://与自然 Y 轴的角度
			temp=y/sqrt((x*x+z*z));
			res=atan(temp);
			break;
	}
	return res*1800/3.14;
}
```

该部分代码总共有 8 个函数，这里我们仅介绍其中 4 个。首先是 ADXL345_Init 函数，该函数用来初始化 ADXL345，和前面我们提到的步骤差不多，不过本章我们而是采用查询的方式来读取数据的，所以在这里并没有开启中断。另外 3 个偏移寄存器，都默认设置为 0。

其次，我们介绍 ADXL345_RD_XYZ 函数，该函数用于从 ADXL345 读取数据，通过该函数可以读取 ADXL345 的转换结果，得到三个轴的加速度值（仅是数值，并没有转换单位）。

接着，我们介绍 ADXL345_AUTO_Adjust 函数，该函数用于 ADXL345 的校准，ADXL345 有偏移校准的功能，该功能的详细介绍请参考 ADXL345 数据手册的第 29 页，偏移校准部分。这里我们就不细说了，如果不进行校准的话，ADXL345 的读数可能会有些偏差，通过校准，我们可以讲这个偏差减少甚至消除。

最后，我们看看 ADXL345_Get_Angle 函数，该函数根据 ADXL345 的读值，转换为与自然坐标系的角度。计算公式如下：

加速度传感器 Z 轴与自然坐标系 Z 轴夹角：$\angle 1 = \tan^{-1}(\frac{\sqrt{A^2_x + A^2_y}}{A_z})$；

加速度传感器 X 轴与自然坐标系 X 轴夹角：$\angle 2 = \tan^{-1}(\frac{A_x}{\sqrt{A^2_y + A^2_z}})$；

加速度传感器 Y 轴与自然坐标系 Y 轴夹角：$\angle 3 = \tan^{-1}(\frac{A_y}{\sqrt{A^2_x + A^2_z}})$；

其中 Ax，Ay，Az 分别代表从 ADXL345 读到的 X，Y，Z 方向的加速度值。通过该函数，我们只需要知道三个方向的加速度值，就可以将其转换为对应的弧度值，再通过弧度角度转换，就可以得到角度值了。

其他函数，我们就不介绍了，也比较简单。保存 adxl345.c，然后把该文件加入 HARDWARE 组下。接下来打开 adxl345.h 在该文件里面加入如下代码：

```
#ifndef __ADXL345_H
#define __ADXL345_H
#include "myiic.h"
#define DEVICE_ID          0X00     //器件 ID,0XE5
#define THRESH_TAP         0X1D     //敲击阀值
……省略部分寄存器定义
#define FIFO_STATUS        0X39

//0X0B TO OX1F Factory Reserved
//如果 ALT ADDRESS 脚(12 脚)接地,IIC 地址为 0X53(不包含最低位).
//如果接 V3.3,则 IIC 地址为 0X1D(不包含最低位).
//开发板接 V3.3,所以转为读写地址后,为 0X3B 和 0X3A(如果接 GND,则为 0XA7 和 0XA6)
#define ADXL_READ      0X3B
#define ADXL_WRITE     0X3A
u8 ADXL345_Init(void);                                  //初始化 ADXL345
void ADXL345_WR_Reg(u8 addr,u8 val);                    //写 ADXL345 寄存器
u8 ADXL345_RD_Reg(u8 addr);                             //读 ADXL345 寄存器
```

```
void ADXL345_RD_XYZ(short *x,short *y,short *z);                    //读取一次值
void ADXL345_RD_Avval(short *x,short *y,short *z);                  //读取平均值
void ADXL345_AUTO_Adjust(char *xval,char *yval,char *zval);         //自动校准
void ADXL345_Read_Average(short *x,short *y,short *z,u8 times);//连续读取 times 次,取平均
short ADXL345_Get_Angle(float x,float y,float z,u8 dir);
#endif
```

上面的代码省略了部分寄存器的定义，其他部分比较简单，我们不作介绍。保存 adxl345.h，然后在 test.c 里面修改代码如下：

```
//x,y:开始显示的坐标位置
//num:要显示的数据
//mode:0,显示加速度值;1,显示角度值;
void Adxl_Show_Num(u16 x,u16 y,short num,u8 mode)
{
    if(mode==0)  //显示加速度值
    {
        if(num<0)
        {
            LCD_ShowChar(x,y,'-',16,0);          //显示负号
            num=-num;                            //转为正数
        }else LCD_ShowChar(x,y,' ',16,0);        //去掉负号
        LCD_ShowNum(x+8,y,num,4,16);             //显示值
    }else           //显示角度值
    {
        if(num<0)
        {
            LCD_ShowChar(x,y,'-',16,0);          //显示负号
            num=-num;                            //转为正数
        }else LCD_ShowChar(x,y,' ',16,0);        //去掉负号
        LCD_ShowNum(x+8,y,num/10,2,16);          //显示整数部分
        LCD_ShowChar(x+24,y,'.',16,0);           //显示小数点
        LCD_ShowNum(x+32,y,num%10,1,16);         //显示小数部分
    }
}
int main(void)
{
    u8 key;
    u8 t=0;
    short x,y,z;
    short angx,angy,angz;
    Stm32_Clock_Init(9);        //系统时钟设置
    uart_init(72,9600);         //串口初始化为 9600
    delay_init(72);             //延时初始化
    LED_Init();                 //初始化与 LED 连接的硬件接口
```

```
    LCD_Init();                     //初始化 LCD
    usmart_dev.init(72);            //初始化 USMART
    KEY_Init();                     //按键初始化
    POINT_COLOR=RED;//设置字体为红色
    LCD_ShowString(60,50,200,16,16,"WarShip STM32");
    LCD_ShowString(60,70,200,16,16,"3D TEST");
    LCD_ShowString(60,90,200,16,16,"ATOM@ALIENTEK");
    LCD_ShowString(60,110,200,16,16,"2012/9/12");
    LCD_ShowString(60,130,200,16,16,"KEY0:Auto Adjust");
    while(ADXL345_Init()) //3D 加速度传感器初始化
    {
        LCD_ShowString(60,150,200,16,16,"ADXL345 Error");
        delay_ms(200);
        LCD_Fill(60,150,239,150+16,WHITE);
        delay_ms(200);
    }
    LCD_ShowString(60,150,200,16,16,"ADXL345 OK");
    LCD_ShowString(60,170,200,16,16,"X VAL:");
    LCD_ShowString(60,190,200,16,16,"Y VAL:");
    LCD_ShowString(60,210,200,16,16,"Z VAL:");
    LCD_ShowString(60,230,200,16,16,"X ANG:");
    LCD_ShowString(60,250,200,16,16,"Y ANG:");
    LCD_ShowString(60,270,200,16,16,"Z ANG:");
    POINT_COLOR=BLUE;//设置字体为红色
    while(1)
    {
        if(t%10==0)//每 100ms 读取一次
        {
            //得到 X,Y,Z 轴的加速度值(原始值)
            ADXL345_Read_Average(&x,&y,&z,10);  //读取 X,Y,Z 三个方向的加速度值
            Adxl_Show_Num(60+48,170,x,0);       //显示加速度原始值
            Adxl_Show_Num(60+48,190,y,0);
            Adxl_Show_Num(60+48,210,z,0);
            //得到角度值,并显示
            angx=ADXL345_Get_Angle(x,y,z,1);
            angy=ADXL345_Get_Angle(x,y,z,2);
            angz=ADXL345_Get_Angle(x,y,z,0);
            Adxl_Show_Num(60+48,230,angx,1);    //显示角度值
            Adxl_Show_Num(60+48,250,angy,1);
            Adxl_Show_Num(60+48,270,angz,1);
        }
        key=KEY_Scan(0);
        if(key==KEY_UP)
```

```
            {
                LED1=0;//绿灯亮,提示校准中
                ADXL345_AUTO_Adjust((char*)&x,(char*)&y,(char*)&z);//自动校准
                LED1=1;//绿灯灭,提示校准完成
            }
            delay_ms(10);
            t++;
            if(t==20)
            {
                t=0;
                LED0=!LED0;
            }
        }
    }
```

此部分代码除了 main 函数，还有一个 Adxl_Show_Num 函数，该函数用于数据显示，因为在 ILI93xx.c 里面，没有提供可以显示小数和负数的函数，所以我们这里编写了该函数，来实现小数和负数的显示，以满足本章要求。

其他部分，我们就不多说了。至此，我们的软件设计部分就结束了。

34.4 下载验证

在代码编译成功之后，我们通过下载代码到 ALIENTEK 战舰 STM32 开发板上，可以看到 LCD 显示如图 34.4.1 所示的内容：

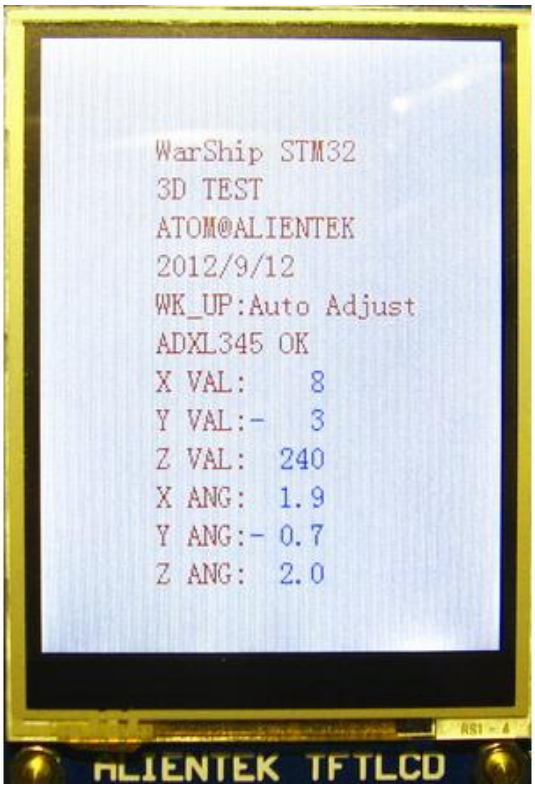

图 34.4.1 程序运行时 LCD 显示内容

可以看到，X 方向和 Z 方向的角度有些大（最佳值是 0），所以我们按下 WK_UP 键，进行一次校准（注意校准的时候保持开发板水平，并且稳定），校准后如图 34.4.2 所示：

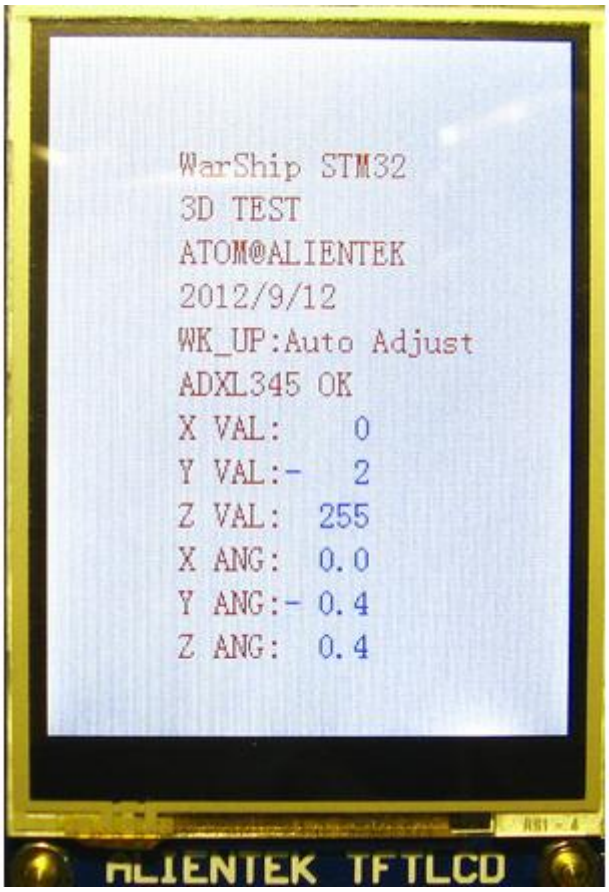

图 34.4.2 校准后

可以看到，校准后，比未校准前好了很多，此时我们移动开发板到不同角度，可以看到 X、Y、Z 的数值和角度也跟着变化。