

ST公司 Lis2dh12 三轴加速度传感器，计算加速度值转成角度值

原创Ch_champion2019-11-21 14:30:1837826收藏12

分类专栏：# 传感器

版权

目录

概述

项目上使用了一款Lis2dh12三轴加速度传感器。开发前要准备的工作。

1、原理图：

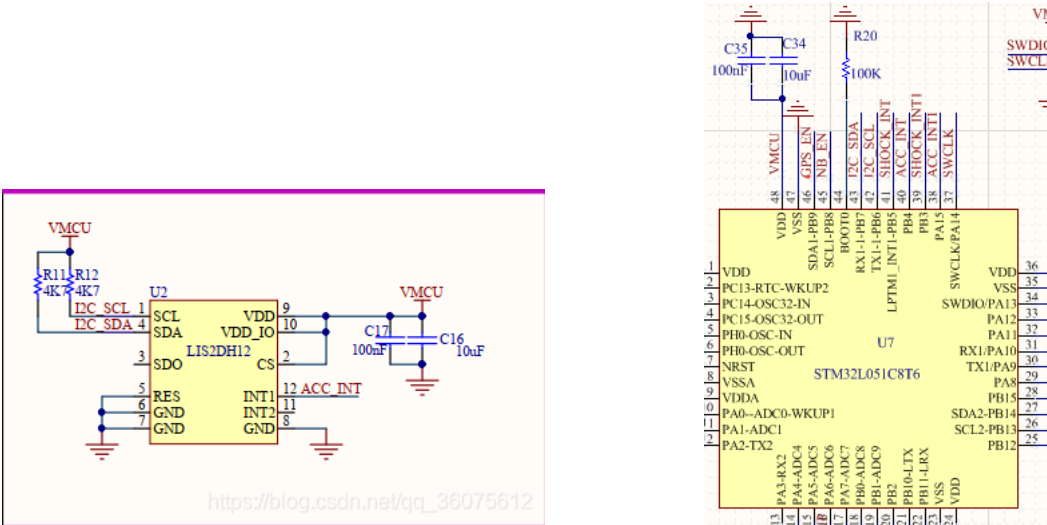
- 1.1、创建lis2dh12.c文件
- 1.2、在此重点说明，如果想调传感器的中断灵敏度，注意：关注1、INT1_THS(32h)，2、INT1_DURATION(33h)这两个寄存器即可。
- 1.3、计算阈值：主要是看配置中断的量程是哪个。
- 1.4、创建lis2dh12.h文件
- 1.5、main.c文件
- 1.6、demo运行结果如下：

概述

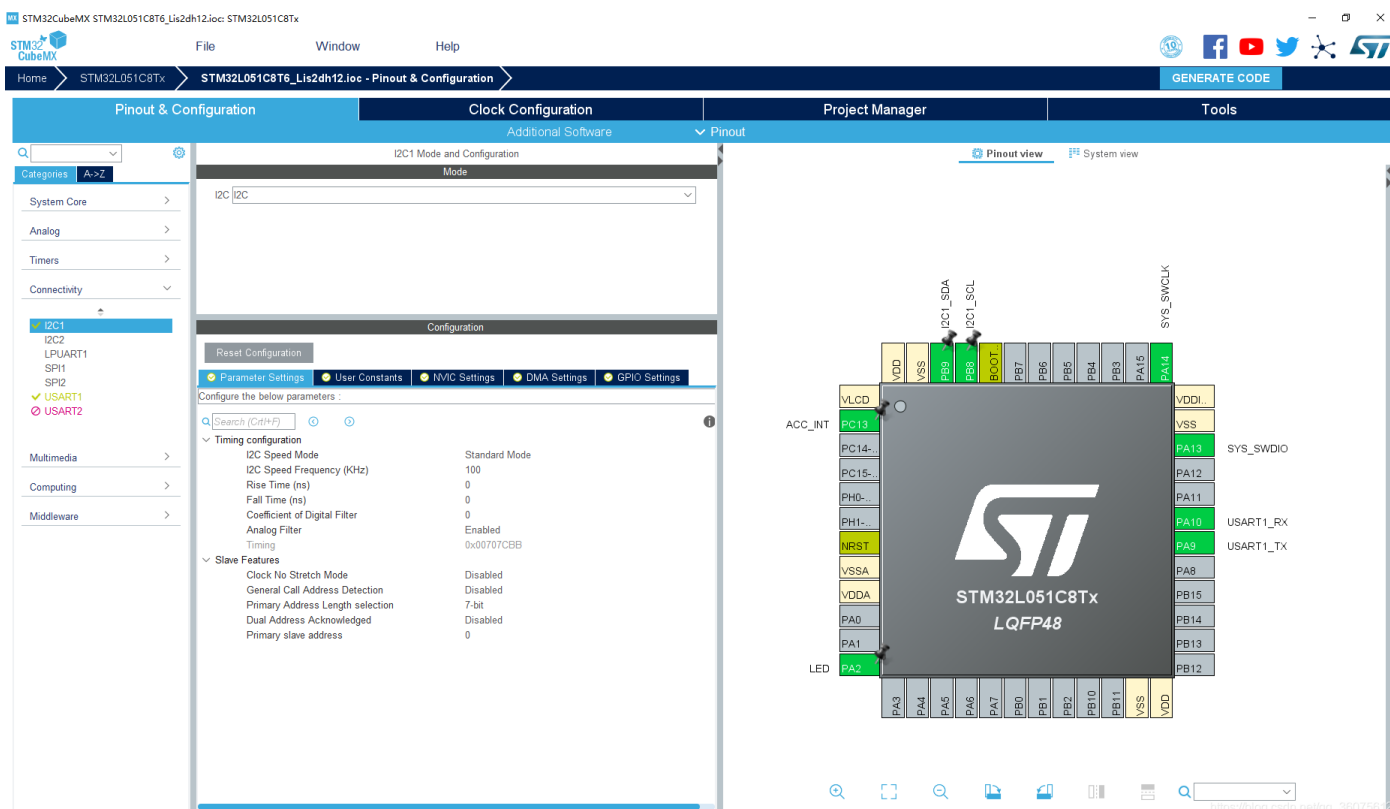
项目上使用了一款Lis2dh12三轴加速度传感器。开发前要准备的工作。

- 1) 要引用官方提供SDK中的lis2dh12_reg.h文件、lis2dh12_reg.c文件，才行。
- 2) 我使用的主控芯片是STM32L051C8T6，硬件IIC与lis2dh12通讯。
- 3) 本次写的是Demo项目，使用工具STM32Cube IDE开发

1、原理图：



.ioc文件创建



6.1.1 I²C operation

The transaction on the bus is started through a START (ST) signal. A START condition is defined as a HIGH-to-LOW transition on the data line while the SCL line is held HIGH. After this has been transmitted by the master, the bus is considered busy. The next byte of data transmitted after the start condition contains the address of the slave in the first 7 bits and the eighth bit tells whether the master is receiving data from the slave or transmitting data to the slave. When an address is sent, each device in the system compares the first seven bits after a start condition with its address. If they match, the device considers itself addressed by the master.

The Slave Address (SAD) associated to the LIS2DH12 is 001100xb. The **S_{DO}/S_{AO}** pad can be used to modify the less significant bit of the device address. If the S_{AO} pad is connected to the voltage supply, LSB is '1' (address 0011001b), else if the S_{AO} pad is connected to ground, the LSB value is '0' (address 0011000b). This solution permits to connect and address two different accelerometers to the same I²C lines.

Data transfer with acknowledge is mandatory. The transmitter must release the SDA line during the acknowledge pulse. The receiver must then pull the data line LOW so that it remains stable low during the HIGH period of the acknowledge clock pulse. A receiver which has been addressed is obliged to generate an acknowledge after each byte of data received.

The I²C embedded inside the LIS2DH12 behaves like a slave device and the following protocol must be adhered to. After the start condition (ST) a slave address is sent, once a slave acknowledgement (SAK) has been returned, an 8-bit sub-address (SUB) is transmitted: the 7 Lsb represent the actual register address while the MSb enables address auto increment. If the MSb of the SUB field is '1', the SUB (register address) is automatically increased to allow multiple data read/writes.

The slave address is completed with a Read/Write bit. If the bit is '1' (Read), a repeated START (SR) condition must be issued after the two sub-address bytes; if the bit is '0' (Write) the master will transmit to the slave with direction unchanged. [Table 16](#) explains how the



1.1、创建lis2dh12.c文件

```
#include "lis2dh12.h"
#include "lis2dh12_reg.h"
#include "i2c.h"
#include "stdlib.h"
#include "stdio.h"
#include <string.h>
#include <math.h>
```

```

#define LIS2DH12_FROM_FS_2g_HR_TO_mg(lsb)  (float)((int16_t)lsb>>4) * 1.0f
#define LIS2DH12_FROM_FS_4g_HR_TO_mg(lsb)  (float)((int16_t)lsb>>4) * 2.0f
#define LIS2DH12_FROM_FS_8g_HR_TO_mg(lsb)  (float)((int16_t)lsb>>4) * 4.0f
#define LIS2DH12_FROM_FS_16g_HR_TO_mg(lsb) (float)((int16_t)lsb>>4) * 12.0f
#define LIS2DH12_FROM_LSB_TO_degC_HR(lsb)  (float)((int16_t)lsb>>6) / 4.0f+25.0f

axis_info_t acc_sample;
filter_avg_t acc_data;

/* =====
func name   : lis2_Delay_us
discription : lis2 微妙延时
param      : us
return     :
Revision   :
===== */
void lis2_Delay_us(uint32_t us)
{
    int n = 11;
    while(--us)
    {
        while(--n);
        n = 11;
    }
}

/* =====
func name   : lis2_Delay_ms
discription : lis2 毫秒延时
param      : ms
return     :
Revision   :
===== */
void lis2_Delay_ms(uint32_t ms)
{
    int i = 0;
    for(;i<ms;i++)
    {
        lis2_Delay_us(1000);
    }
}

/* =====
func name   : lis2dh12_iic_read_byte
discription : 写一个字节
param      : reg:寄存器地址, data:寄存器对应的值
return     :
Revision   :
===== */
int32_t lis2dh12_iic_write_byte(uint8_t reg, uint8_t data)
{
    #ifdef HARDWARE_IIC
        HAL_I2C_Mem_Write(&hi2c1, LIS2DH12_I2C_ADD_H, reg,I2C_MEMADD_SIZE_8BIT, &data, 1, 1000);
    #endif

    #ifdef SIMULATED_IIC
        Lis2DH12_IIC_Write_Byte(LIS2DH12_I2C_ADD_H, reg, data);
    #endif
    return 1;
}

```

```

/* ===== | func name : lis2dh12_iic_read_byte
discription :  读一个字节
param       :  reg:寄存器地址, data:寄存器对应的值
return      :
Revision    :
===== */
int32_t lis2dh12_iic_read_byte(uint8_t reg, uint8_t* data)
{
    #ifdef HARDWARE_IIC
        HAL_I2C_Mem_Read(&hi2c1, LIS2DH12_I2C_ADD_H, reg, I2C_MEMADD_SIZE_8BIT, data, 1, 1000);
    #endif

    #ifdef SIMULATED_IIC
        Lis2DH12_IIC_Read_Byte(LIS2DH12_I2C_ADD_H, reg, data);
    #endif

    return 1;
}

/* =====
func name : Lis2dh12_Init
discription : 配置检测阈值与中断
param       : void
return      :
Revision    :
===== */
int32_t Lis2dh12_Init(void)
{
    /* Initialization of sensor */
    lis2dh12_iic_write_byte(0x20, 0x37); /* CTRL_REG1(20h): 关闭sensor, 设置进入掉电模式 ODR 25HZ */
    // lis2dh12_iic_write_byte(0x20, 0x57); /* CTRL_REG1(20h): 关闭sensor, 设置进入低功耗模式 ODR 100HZ */
    lis2dh12_iic_write_byte(0x21, 0x03); /* CTRL_REG2(21h): IA1、IA2 开启高通滤波 bc */
    lis2dh12_iic_write_byte(0x22, 0xc0); /* CTRL_REG3(22h): 0x80 使能单击中断到INT_1 INT_2 */
    lis2dh12_iic_write_byte(0x23, 0x88); /* CTRL_REG4(23h): 使能快, 数据更新, 全量程+/-2G, 非常精度模式 */
    // lis2dh12_iic_write_byte(0x25, 0x00); /* CTRL_REG6(25h): 高电平(上升沿)触发中断 */

    /* INT1 翻转检测, 中断*/ /*0x6a
    lis2dh12_iic_write_byte(0x30, 0x7f); /* INT1_CFG(30h): 使能, 6D X/Y/Z任一超过阈值中断 */
    // lis2dh12_iic_write_byte(0x30, 0x4f); /* INT1_CFG(30h): 使能, 6D X/Y任一超过阈值中断 */
    // lis2dh12_iic_write_byte(0x31, 0x20); /* INT1_SRC(31h): 设置中断源 */

    lis2dh12_iic_write_byte(0x32, 0x02); /* INT1_THS(32h): 设置中断阈值 0x10: 16*2(FS) 0x20: 32*16(FS) */

    // lis2dh12_iic_write_byte(0x33, 0x02); /* INT1_DURATION(33h): 1LSB=1/ODR 如果ODR=25HZ 那么1LSB=40ms 设置延时 1s, 对应25->0x1
    // lis2dh12_iic_write_byte(0x33, 0x03); /* INT1_DURATION(33h): 1LSB=1/ODR 如果ODR=50HZ 那么1LSB=20ms 设置延时 1s, 对应50->0x32
    lis2dh12_iic_write_byte(0x33, 0x03); /* INT1_DURATION(33h): 1LSB=1/ODR 如果ODR=100HZ 那么1LSB=10ms 设置延时 1s, 对应100->0x6

    // /* INT2 单击中断 */
    lis2dh12_iic_write_byte(0x24, 0x01); /* CTRL_REG5(24h): */
    // lis2dh12_iic_write_byte(0x25, 0xa0); /* CTRL_REG6(25h): Click interrupt on INT2 pin */
    //
    // lis2dh12_iic_write_byte(0x38, 0x15); /* CLICK_CFG (38h): 单击识别中断使能 */
    lis2dh12_iic_write_byte(0x39, 0x10);
    // lis2dh12_iic_write_byte(0x3a, 0x7f); /* CLICK_THS (3Ah): 单击阈值 */
    // lis2dh12_iic_write_byte(0x3b, 0xff); /* TIME_LIMIT (3Bh): 时间限制窗口6 ODR 1LSB=1/ODR 1LSB=1/100HZ, 10ms, 设置延时1s, 对应100->
    // lis2dh12_iic_write_byte(0x3c, 0xff); /* TIME_LATENCY (3Ch): 中断电平持续时间1 ODR=10ms */
    // lis2dh12_iic_write_byte(0x3d, 0x01); /* TIME_WINDOW (3Dh): 单击时间窗口 */

    /* Start sensor */
    // lis2dh12_iic_write_byte(0x20, 0x37);
    lis2dh12_iic_write_byte(0x20, 0x5f); /* CTRL_REG1(20h): Start sensor at ODR 100Hz Low-power mode */

    return 0;
}

```

```

/* =====
func name : get_acc_value
discription : 获取加速度值
param : axis_info_t *sample
return : void
Revision :
===== */
void get_acc_value(axis_info_t *sample)
{
    uint8_t i = 0;
    uint8_t data[6];
    for (i=0; i<6; i++){
        lis2dh12_iic_read_byte(0x28+i, data+i); //获取X、y、z轴的数据
        //printf("data[i] %d, \r\n", data[i]);
    }
    // sample->x = abs((int)(LIS2DH12_FROM_FS_2g_HR_TO_mg(*(int16_t*)data)));
    // sample->y = abs((int)(LIS2DH12_FROM_FS_2g_HR_TO_mg(*(int16_t*)(data+2))));
    // sample->z = abs((int)(LIS2DH12_FROM_FS_2g_HR_TO_mg(*(int16_t*)(data+4))));
    sample->x = LIS2DH12_FROM_FS_2g_HR_TO_mg(*(int16_t*)data);
    sample->y = LIS2DH12_FROM_FS_2g_HR_TO_mg(*(int16_t*)(data+2));
    sample->z = LIS2DH12_FROM_FS_2g_HR_TO_mg(*(int16_t*)(data+4));
}

/* =====
func name : filter_calculate
discription : 均值滤波器---滤波
            读取xyz数据存入均值滤波器, 存满进行计算, 滤波后样本存入sample
param : filter_avg_t *filter, axis_info_t *sample
return : void
Revision :
===== */
void filter_calculate(filter_avg_t *filter, axis_info_t *sample)
{
    uint8_t i = 0;
    short x_sum = 0, y_sum = 0, z_sum = 0;

    for (i=0; i<FILTER_CNT; i++)
    {
        get_acc_value(sample);

        filter->info[i].x = sample->x;
        filter->info[i].y = sample->y;
        filter->info[i].z = sample->z;

        x_sum += filter->info[i].x;
        y_sum += filter->info[i].y;
        z_sum += filter->info[i].z;

        printf("acc_x:%d, acc_y:%d, acc_z:%d \n", filter->info[i].x, filter->info[i].y, filter->info[i].z);
    }
    sample->x = x_sum / FILTER_CNT;
    sample->y = y_sum / FILTER_CNT;
    sample->z = z_sum / FILTER_CNT;
    printf("\r\n acc_info.acc_x:%d, acc_info.acc_y:%d, acc_info.acc_z:%d \r\n", sample->x, sample->y, sample->z);
}

/* =====
func name : new_angle_calculate
discription : 计算新角度
param : axis_info_t *sample
return : void
Revision :
===== */
void new_angle_calculate(axis_info_t *sample)

```

```
{
    |
    |      sample->new_angle_x = atan((short)sample->x/(short)sqrt(pow(sample->y, 2)+pow(sample->z, 2))) * DEGREE_CAL;
sample->new_angle_y = atan((short)sample->y/(short)sqrt(pow(sample->x, 2)+pow(sample->z, 2))) * DEGREE_CAL;
sample->new_angle_z = atan((short)sample->z/(short)sqrt(pow(sample->x, 2)+pow(sample->y, 2))) * DEGREE_CAL;
if (sample->new_angle_z < 0)
{
    sample->new_angle_x = 180-sample->new_angle_x;
    sample->new_angle_y = 180-sample->new_angle_y;
}
printf("sample->new_angle_x:%d, sample->new_angle_y:%d, sample->new_angle_z:%d \r\n",sample->new_angle_x, sample->new_angle_

}

/* =====
func name      :  old_angle_calculate
discription   :  计算旧角度
param         :  axis_info_t *sample
return        :  void
Revision      :
===== */
void old_angle_calculate(axis_info_t *sample)
{
    sample->old_angle_x = atan((short)sample->x/(short)sqrt(pow(sample->y, 2)+pow(sample->z, 2))) * DEGREE_CAL;
    sample->old_angle_y = atan((short)sample->y/(short)sqrt(pow(sample->x, 2)+pow(sample->z, 2))) * DEGREE_CAL;
    sample->old_angle_z = atan((short)sample->z/(short)sqrt(pow(sample->x, 2)+pow(sample->y, 2))) * DEGREE_CAL;
    if (sample->old_angle_z < 0)
    {
        sample->old_angle_x = 180-sample->old_angle_x;
        sample->old_angle_y = 180-sample->old_angle_y;
    }
    printf("sample->old_angle_x:%d, sample->old_angle_y:%d, sample->old_angle_z:%d \r\n",sample->old_angle_x, sample->old_angle_

}
```

1.2、在此重点说明，如果想调传感器的中断灵敏度，注意：关注1、INT1_THS(32h)，2、INT1_DURATION(33h)这两个寄存器即可。

INT1_THS(32h)

8.21 INT1_THS (32h)

Table 58. INT1_THS register

0	THS6	THS5	THS4	THS3	THS2	THS1	THS0
---	------	------	------	------	------	------	------

Table 59. INT1_THS description

THS[6:0]	Interrupt 1 threshold. Default value: 000 0000 1 LSB = 16 mg @ FS = 2 g 1 LSB = 32 mg @ FS = 4 g 1 LSB = 62 mg @ FS = 8 g 1 LSB = 186 mg @ FS = 16 g
----------	--

1.3、计算阈值：主要是看配置中断的量程是哪个。

(1) 如果量程 FS=2g 每1LSB=16mg
那么寄存器配置INT1_THS=2 那么阈值=2*16mg
如果INT1_THS=5 那么阈值=5*16mg

(2) 如果量程FS=4g，那么每1LSB=32mg
INT1_THS =5 阈值就是5*32mg=160mg

8.22 INT1_DURATION (33h)

Table 60. INT1_DURATION register

0	D6	D5	D4	D3	D2	D1	D0
---	----	----	----	----	----	----	----

Table 61. INT1_DURATION description

D[6:0]	Duration value. Default value: 000 0000 1 LSB = 1/ODR
--------	--

The D[6:0] bits set the minimum duration of the Interrupt 2 event to be recognized. Duration steps and maximum values depend on the ODR chosen.
Duration time is measured in N/ODR, where N is the content of the duration register.

看配置的ODR是哪个, 来计算延时的时间。

1LSB=1/ODR 如果ODR=50HZ 那么1LSB=20ms

```
/*
500mg = 30°
250mg = 15°
160mg = 10°
我现在对应的量程是FS=2g, INT1_THS=10, 那么阈值=10*16mg

1LSB=1/ODR 如果ODR=50HZ 那么1LSB=20ms
50LSB = 1s
100LSB = 2s
*/
```

1.4、创建lis2dh12.h文件

```
#ifndef __LIS2DH12_H
#define __LIS2DH12_H

#ifdef __cplusplus
extern "C"
{
#endif

#include "stdint.h"

#define DEGREE_CAL 180.0/3.1416
#define FILTER_CNT 4

typedef struct {
    short x;
    short y;
    short z;
    short new_angle_x;
    short new_angle_y;
    short new_angle_z;
    short old_angle_x;
    short old_angle_y;
    short old_angle_z;
}axis_info_t;

typedef struct filter_avg{
    axis_info_t info[FILTER_CNT];
    unsigned char count;
}filter_avg_t;

extern axis_info_t acc_sample;
extern filter_avg_t acc_data;

void filter_calculate(filter_avg_t *filter, axis_info_t *sample);
void old_angle_calculate(axis_info_t *sample);
void new_angle_calculate(axis_info_t *sample);

#endif
```

1.5、main.c文件

```
/* USER CODE BEGIN Header */
```

```

/* USER CODE END Header */

/* Includes -----*/
#include "main.h"
#include "i2c.h"
#include "usart.h"
#include "gpio.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "stdio.h"

#include "lis2dh12.h"
/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

```



```

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

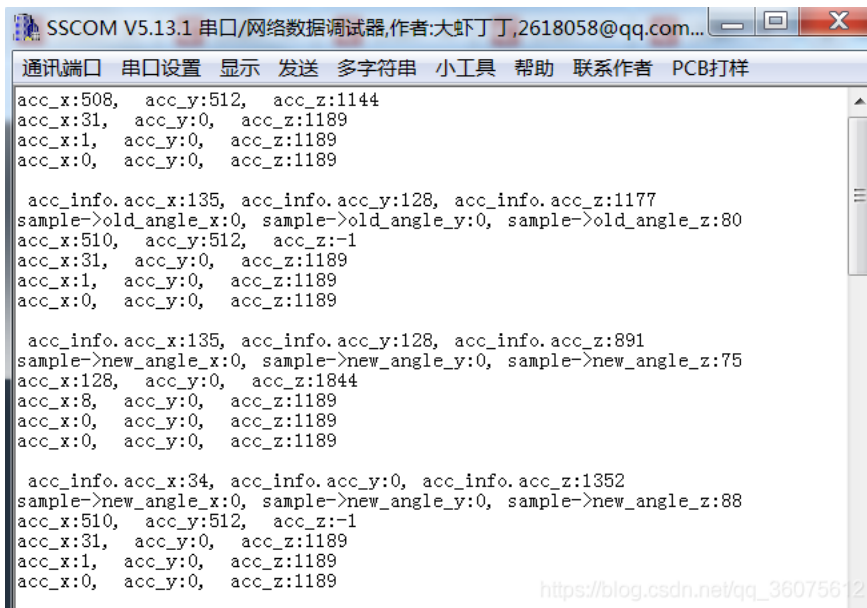
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_I2C1_Init();
MX_USART1_UART_Init();
/* USER CODE BEGIN 2 */
filter_calculate(&acc_data, &acc_sample);
old_angle_calculate(&acc_sample);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_Delay(1000);
    filter_calculate(&acc_data, &acc_sample);
    new_angle_calculate(&acc_sample);
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

```

1.6、demo运行结果如下:



The screenshot shows a serial terminal window titled "SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁,2618058@qq.com...". The window has tabs for "通讯端口", "串口设置", "显示", "发送", "多字符串", "小工具", "帮助", "联系作者", and "PCB打样". The "显示" tab is active, showing the following output:

```

acc_x:508, acc_y:512, acc_z:1144
acc_x:31, acc_y:0, acc_z:1189
acc_x:1, acc_y:0, acc_z:1189
acc_x:0, acc_y:0, acc_z:1189

acc_info.acc_x:135, acc_info.acc_y:128, acc_info.acc_z:1177
sample->old_angle_x:0, sample->old_angle_y:0, sample->old_angle_z:80
acc_x:510, acc_y:512, acc_z:-1
acc_x:31, acc_y:0, acc_z:1189
acc_x:1, acc_y:0, acc_z:1189
acc_x:0, acc_y:0, acc_z:1189

acc_info.acc_x:135, acc_info.acc_y:128, acc_info.acc_z:891
sample->new_angle_x:0, sample->new_angle_y:0, sample->new_angle_z:75
acc_x:128, acc_y:0, acc_z:1844
acc_x:8, acc_y:0, acc_z:1189
acc_x:0, acc_y:0, acc_z:1189
acc_x:0, acc_y:0, acc_z:1189

acc_info.acc_x:34, acc_info.acc_y:0, acc_info.acc_z:1352
sample->new_angle_x:0, sample->new_angle_y:0, sample->new_angle_z:88
acc_x:510, acc_y:512, acc_z:-1
acc_x:31, acc_y:0, acc_z:1189
acc_x:1, acc_y:0, acc_z:1189
acc_x:0, acc_y:0, acc_z:1189

```

At the bottom right of the terminal window, there is a URL: https://blog.csdn.net/qq_36075612.

代码: Git