基于 ARM-Linux 的微惯性单元数据采集与处理

孙 凯,刘瑞华
(中国民航大学 新航行系统研究所,天津 300300)

摘 要:在 Linux 下通过串口编程对 MEMS IMU 数据采集和解算,实现了其高速实时采集。其中,设计的 IIR 低通滤波器有效消除了信号中的噪声成分,并通过 Qt 编程设计了应用程序窗口将 MEMS IMU 的输出数据动态显示在 ARM 开发板上。该设计在小体积、低功耗、低成本的惯性测量中具有重要的工程应用意义,可广泛应用于动态测量、动态控制、辅助导航等领域。

关键词:ARM-Linux;MEMS IMU;惯性测量;数据采集;IIR 滤波器
中图分类号:TP277 **文献标识码**:A **文章编号**:1674-7720(2011)04-0083-03

MEMS IMU data acquisition and processing based on ARM-Linux

SUN Kai, LIU Ruihua
(Institution of New Navigation System, Civil Aviation University of China, Tianjin 300300, China)

Abstract: In the way of serial port programming the real-time data of MEMS IMU was speedly acquired and decoded. Analysing the data, the paper designed an IIR lowpass filter to reduce the influence and interference of noise. And an output window was designed to display the data that got on the screen of the ARM development board. The design, with its valuable characters of small size, low power consumption and low cost, can be widely applicated in many areas such as dynamic measurement, dynamic controlling and auxiliary navigation.

Key words: ARM-Linux; MEMS IMU; inertial measurement; data acquisition; IIR filter

微机械惯性器件是集微型精密机械、微电子学、半导体集成电路等新技术于一身的世界前沿新技术。随着微电子技术的发展,目前微机械惯性器件凭借其价格低、可靠性高、尺寸小、重量轻等特点引起了国内惯性技术及微电子技术领域的广泛关注。以陀螺仪和加速度计为核心部件的惯性导航系统已成为现代飞机、大型舰只和潜艇的一种重要导航设备,在其他一些民用领域中也有着十分广泛和重要的应用。以惯性系统为基础发展起来的惯性测量和惯性定位系统,可以用于大地测量、地图绘制、海洋调查、地球物理勘探、管道铺设选线、石油钻井定位和机器人等需要大范围测量及精确定位的场合[1]。

本文在嵌入式 Linux 环境下使用 ARM9 开发板实现了对 IMU 输出数据的采集和动态显示,为进一步的工程应用打下基础。

1 微惯性测量单元 IMU

微惯性测量单元由 6 个传感器组成,包括 3 个微机械陀螺仪和 3 个微机械加速度计,配置在立方体的 3 个正交平面上。其基本原理为古典的牛顿力学原理,由三根轴的陀螺确定载体的姿态,安装在三根轴上的加速度计测出载体的加速度值,积分得到速度,再积分得到位移[1,2]。

本文所采用的惯性测量单元为 XW-IMU5200。它以 DSP 为核心处理器,采用 16 bit 高精度多通道并行 A/D 转换。其 A/D 转换器的采样率至少是惯性传感器带宽的 4 倍,能够保持惯性传感器的固有频率。6 路并行采集通道可实时接收加速度计和陀螺仪以及温度传感器的信号,保证了数据采集的一致性。图 1 所示为本文所采用的 XW-IMU5200 的外观。

图 1 XW-IMU5200

XW-IMU5200 内部有一个 0.8 μs 的计时器。计时器计数从 0~2^{16},然后开始新的周期(高位溢出后继续)。每个数据周期中,在读取内部计时器之前和之后,分别对两组惯性测量数据进行采样,然后将得到的数据做数字滤波并封装;惯性测量数据、计时器数据、温度数据通过 XW-IMU5200 的 RS232 口

送出。

如图2所示，Ti是周期的起点。从Ti到$Ti1$为0.1 ms，实现第一组数据的采样；从$Ti1$到$Ti2$为1 ms，读取内部计时器；从$Ti2$到$Ti3$为0.1 ms，实现第二组数据的采样；从$Ti3$到$Ti4$进行滤波并封装数据，对IMU为1.8 ms，对IMU而言，最大数据输出速率约为100 Hz，波特率为115 200 b/s。

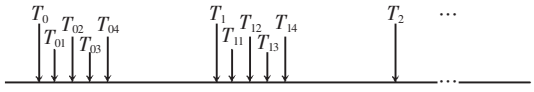

图2 XW-IMU5200的时序

XW-IMU5200的测量数据包为19 bit，其定义如表1所示，数据为Little Endian格式，即低位(LSB)字节在先。

表1 XW-IMU5200 IMU测量数据格式

字节	类型	定义
0	unsigned char	字头，0xAA
1	unsigned char	FLAG(类型 & 量程)
2~3	unsigned short	计时器
4~5	Short	陀螺仪x轴
6~7	Short	陀螺仪y轴
8~9	Short	陀螺仪z轴
10~11	Short	加速度计x轴
12~13	Short	加速度计y轴
14~15	Short	加速度计z轴
16~17	unsigned short	温度
18	unsigned char	检验和

2 Arm-linux环境下的MEMS IMU数据采集实现

2.1 交叉编译环境的建立

本设计所采用的ARM9开发板为S3C2440处理器，内嵌Linux系统。

因嵌入式开发一般需要在PC机上进行，需要在宿主机建立交叉编译环境，以下给出建立交叉编译环境的步骤：

(1)安装linux环境；

(2)下载交叉编译文件包 cross-2.95.3.tar.bz2；

(3)建立交叉编译环境目录 /usr/local/arm/；

(4)复制安装包到目录下，在此目录下解包文件，命令为：tar jxvf cross-2.95.3.tar.bz2；

(5)配置环境变量，修改bashrc文件，在文件最后一行添加环境变量，代码为export。

PATH=/usr/local/arm/2.95.3/bin: $PATH，保存退出后重启Linux。

至此，交叉编译环境建立完成。

2.2 串口通信的实现

本设计所采用的微惯性单元数据通过RS232串口输出，同时所使用的ARM开发板上也附有串口，因此完成数据采集必须通过串口通信。

串口通信是仪器仪表设备通用的通信方式，它用于ASCII码的字符传输，主要由地线、发送和接收数据线3根数据线完成，其他线用于握手。

串口通信的最重要的参数配置是：波特率、数据位和奇偶校验位，在进行串口通信时，必须正确设置参数。Linux中所有的设备一般位于/dev下，串口1和2的名称分别为/dev/ttyS0和/dev/ttyS1. 通过对struct termios结构体的各成员值的设置来进行串口设置，如下：

```
#include<termios.h>
Struct termio
{
unsigned short c_iflag;          /* 输出控制模式标志 */
unsigned short c_oflag;          /* 输出模式标志 */
unsigned short c_cflag;          /* 控制模式标志 */
unsigned short c_lflag;          /* 本地模式标志 */
unsigned char c_line;            /* 行标志 */
unsigned char c_cc[NCC;          /* 控制字符 */
};
```

其中，c_cflag包含对数据传输率、字符大小、数据位、停止位、奇偶校验位和硬件流控的设置。

串口配置主函数如下：

```
int main(void)
{
    int fd;
    int nread,i;
    char buff[512]="0";
    if((fd=open_port(fd,1))<0)
    {
      perror("open_port error");
      return;
    }
    if((i=set_opt(fd,38400,8,'N',1))<0)
    {
      perror("set_opt error");
      return;
    }
    printf("fd=%d\n",fd);
      while(1)
    {
      while((nread = read(fd,buff,sizeof(buff)))>0)
    {
      if(buff[i]==0xaa&&buff[i+1]==0x18);
      printf("header found\n ");
      }
    }
      close(fd);
      return;
}
```

串口波特率为9 600 b/s，数据位为8 bit，无奇偶校验位，1 bit停止位。对于串口的操作同读写文件，使用

read、write 函数。如上串口调通后，根据产品的解码进行数据的解算，其算法可表示为：

```
IMU_meas.gyro[i]=IMU_data.gyro[i]*G_S/SCALAR;
//陀螺仪输出数据解算,i 取值 1、2、3 分别代表正交方向三路陀螺仪
IMU_meas.acc [i]=IMU_data.acc [i]*A_S/SCALAR; //加速度计输出数据解算,i 取值 1、2、3,分别代表三路加速度计
```

本文忽略温度信息只考虑 6 路传感器信息，其中 G_S 为陀螺仪的角度范围，A_S 为加速度计测量范围，scalar 为常值 2^{15}。

2.3 数字低通滤波

低通滤波属于经典滤波的范畴，它通过一定的运算关系改变输入信号频率成分的相对比例或滤除某些频率成分，对 MEMS IMU 进行滤波的目的就是尽量滤除信号中的各种噪声成分，因为 MEMS IMU 输出信号的有用成分基本位于低频段，加之实时性能的要求，这里只选择相比 FIR 滤波器阶次低得多的 IIR 滤波器[3]。

随机采集一组数据进行滤波说明，如图 3 为转台静止状态下 y 轴 MEMS 陀螺仪输出的 5 000 点数据，采样频率为 100 Hz。

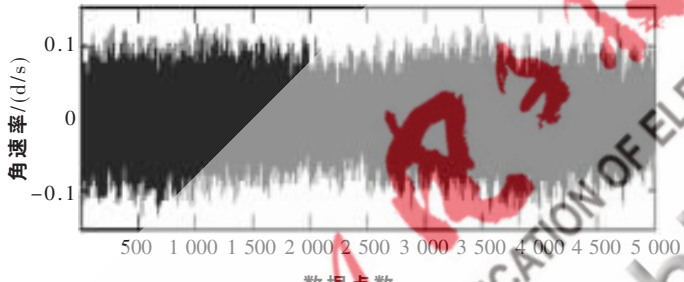

图 3 转台静止转台下陀螺仪 y 轴采样数据

为了确定滤波器的通带截止频率，对这组数据进行 Yule Walker 功率谱密度分析[5]，为了获得较高的精度，此处取 AR 模型的阶数为 30，功率谱密度分析结果如图 4 所示。

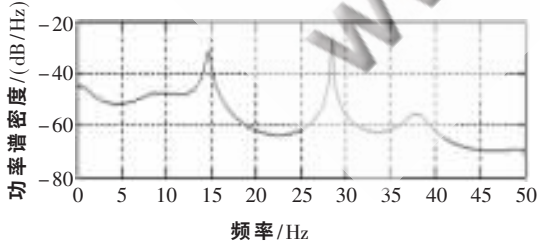

图 4 功率谱密度分析结果

从图 4 中可以看出陀螺仪输出信号中存在两个明显的尖峰，分别位于 14.6 Hz 和 28.5 Hz 处，所以通带的截止频率必须小于 14.6 Hz。这两个尖峰是指陀螺仪输出数据随机误差的正弦成分，由图中可知它们对陀螺仪输出数据的零点漂移起主导作用。考虑到过渡带宽的因素，截止频率的设置最大也应为 7 Hz 左右。但这一滤波器的任务除了要滤除正弦成分外，也应该滤除大部分的近似白噪声统计特性的噪声成分，并且在没有噪声情况下陀螺仪的输出数据应为常数，所以截止频率设得越低越好。

设计 IIR 数字低通滤波器必须确定的另一个重要参数是滤波器的阶数。阶数低时，滤波时延较小，但过渡带宽过大，滤波效果不明显，阶数高时，过渡带宽较小，但滤波时延较大。为了既能获得较好的滤波效果，又能够最大限度地满足实时应用，在选择滤波器的阶数时需要做折中考虑。基于以上分析，选择具有单调下降幅频特性的巴特沃斯滤波器。

采用巴特沃斯直接型结构，系统函数为：

$$H(z)=\frac{\sum_{r=0}^{M} b_r z^{-r}}{1+\sum_{k=1}^{N} a_k z^{-k}}$$

通过分析比较，当通带截止频率设为 4 Hz 左右、滤波器阶数设为 4 时，能得到理想的综合滤波效果。图 5 所示为滤波后的数据。

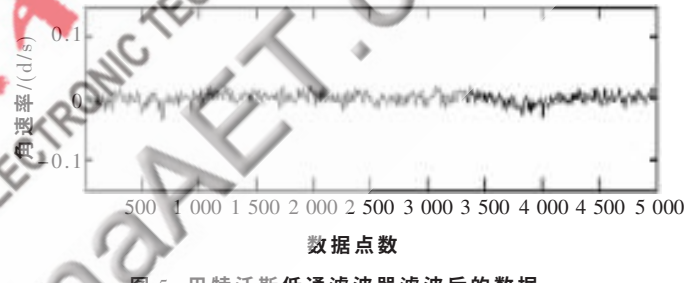

图 5 巴特沃斯低通滤波器滤波后的数据

对滤波后的数据进行 Yule Walker 功率谱密度分析，结果如图 6 所示。

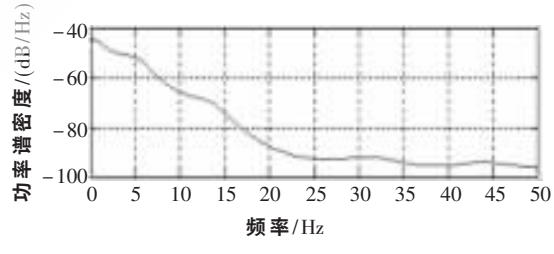

图 6 巴特沃斯低通滤波器滤波后数据功率谱密度

比较图 4 和图 6 可以看出，滤波后的功率谱密度原来的两个尖峰完全消失了，除 0 Hz 附近整体变得十分平坦。这正是期望的滤波效果，说明设计的滤波器达到了预期的滤波目的。计算滤波前后的均值与方差，结果如表 2 所示。

可以看出，滤波后的均值与滤波前的均值相比在误差允许的范围内可以认为是相等的，滤波后的方差比滤波前的方差小两个数量级。这说明合理地选择巴特沃斯数字低通滤波器的截止频率和阶数，MEMS 陀螺仪输出

表 2 巴特沃斯数字低通滤波器滤波前后的均值与方差

	均值	方差
滤波前	2.2380e-004	3.3000e-003
滤波后	2.3225e-004	6.8304e-005

的数据可产生较好的滤波效果。通过计算得出低通滤波器的权系数 a_k 和 b_r，便可根据差分方程编制 C 语言程序。

2.4 输出数据基于 Qt 的界面显示

Qt 是一个跨平台的 C++图形界面库，主要通过汇集 C++类的形式来实现应用程序界面开发所需要的一切，包括 Qt/X11、Qt Embedded、Qtdesigner 和 Qt linguist 等[4]。Qt 是基于面向对象的 C++语言，它提供了 signal(信号)和 slot(槽)的对象通信机制，具有可查询和设计的属性以及强大的事件和事件过滤器。本文主要用到其面向嵌入式开发的 Qt Embedded 及其设计器 Qtdesigner。

Qt 界面开发通常有以下步骤：

(1)用 Qt 生成 file.ui 和 main.cpp；

(2)用 uic 生成 file.h 和 file.cpp；

(3)用 qmake 生成 file.pro；

(4)通过./setenv 命令设置环境变量；

(5)用 tmake 生成二进制代码。

主函数部分如下：

```
int main( int argc, char ** argv)
{
    QApplication a(argc, argv);
    IMU_display w;
    QTimer *t = new QTimer(&w );
a.connect(t,SIGNAL(timeout()),&w,SLOT(imu()));
    t->start( 10, FALSE);
    w.show();
a.connect(&a,SIGNAL(lastWindowClosed()),&a, SLOT(quit()) );
    return a.exec();
}
```

IMU 数据输出速率为 100 Hz，将 Qtimer 定时器设置为 10 ms 刷新一次，保证数据输出的完整性。IMU 数据的具体输出可以进行定制。在上位机开发完系统之后，可以利用虚拟帧缓存技术(qvfb)技术在 PC 机上测试 Qt/Embedded 程序。经过反复的测试修改，测试成功之后制作图标和桌面启动器，通过 minicom 拷贝到 ARM-Linux 系统下的指定目录，便完成了程序的开发工作，需要注意的是交叉编译前需要将程序中上位机的串口名改为 ARM 开发板的指定串口名，否则程序将不会正常运行。

本文实现了 ARM-Linux 环境下对于 MEMS IMU 的数据采集处理，功耗低、成本低、体积小，可广泛应用到各种惯性测量领域。文中所述 IIR 数字低通滤波器，设计简单，适合运用于有用信号和噪声的频带不重叠的非高速变化运动的场合。其不足之处在于虽然可以取得较为理想的滤波效果，但同时产生了一定的延迟，所以不适合对于实时性要求很高的惯性测量场合，但可以通过改进滤波算法来实现。

参考文献

[1] 刘俊,石云波,李杰.微惯性技术[M].北京：电子工业出版社,2005.

[2] 牛徐明,王田苗,梁建宏.基于 ARM 与 MEMS 器件的微惯性测量装置设计[J].单片机与嵌入式系统应用,2007,7(3)：62-64.

[3] 丁杨斌,王新龙,王缜,等.数字滤波在光纤陀螺数据处理中应用研究[J].传感器世界,2005,11(11)：13-16.

[4] 倪继利.Qt 及 Linux 操作系统窗口设计[M].北京：电子工业出版社,2006.

[5] 张贤达.现代信号处理(第二版)[M]..北京：清华大学出版社,2002.

(收稿日期：2010-09-20)

作者简介：

孙凯，男，1985 年生，硕士，主要研究方向：通信与信息系统、惯性导航。

刘瑞华，男，1965 年生，博士，教授，主要研究方向：导航、制导与控制、组合导航信息融合。