嵌入式 Linux 下温湿度传感器的设计与实现①

Design and Implementation of Temperature and Humidity Sensor Based on Embedded Linux

陈　博　刘锦高　(华东师范大学 电子科学技术系 上海 200241)

摘　要： 在嵌入式应用领域，需要测量周围环境的质量对生产和工作进行监控和预警。通过比较设计方案，提出在嵌入式 Linux 下，基于 PXA310 平台温湿度传感器的设计与实现方法。在 Linux 操作系统下通过对驱动程序接口调用，完成温湿度数据读取和预警，并对 Linux 驱动程序编写进行比较。实验表明，本方案硬件和软件设计切实可行，提高了环境测量的准确度和系统性能的实时性。

关键词： Linux PXA310 驱动程序 实时性

1　引言

在工业控制和工业生产领域中，传感器对于工业控制和生产环境的监控作用不言而喻。传统的传感器监控系统大都采用单片机控制，其监控的准确度和实时性不太令人满意。本文寻找到一套切实可行的传感器设计方案，其利用温湿度传感器芯片，基于 PXA310 硬件平台和 Linux 操作系统，能有效监控现场温湿度变化。在周围环境发生变化，不能满足工作要求时，可以获取监控数据并提出预警，提高生产和工作环境检测的可靠性及实时性。

2　温湿度传感器电路设计

比较了一些传感器应用设计方案后，选用 SHT10 芯片为嵌入式温湿度传感器的核心部件。它外围电路简便，相比其他传感器芯片(DS18B20)有其独到优势[1]。STH10 每秒可进行 3 次温湿度测量，数据精度 14 bit 并且工作稳定。其测量采用 CMOSens 专利[2]，所以在测量效率和精度上要好于 DS18B20。DS18B20 采用单总线控制方案(1-wire)，大约每秒测量一次，9 位数字式温度数据；只提供温度测量。其在生产环境检测要求严格时，就显得精度和功能有些不足。

2.1 SHT10 简介

SHT10 是一款高度集成的温湿度传感器芯片，提供全量程标定数字输出。传感器包括一个电容性聚合体湿度敏感元件和一个用能隙材料制成的温度敏感元件，他们与一个 14 位 A/D 转换器以及一个串行接口电路设计在同一个芯片上面。其通过标定得到校准系数以程序形式储存在芯片 OTP 内存中，并利用两线制串行接口与内部电压调整，使外围系统集成变得快速而简单。

2.2 SHT10 工作原理

SHT10 芯片电源 3.3V。传感器上电后，等待 11ms 来完成“休眠”状态。通信复位和启动传输命令后，发送一组测量命令(‘00000101’表示相对湿度 RH，‘00000011’表示温度 T)，控制器要等待测量结束。这个过程需要大约 11/55/210ms ，分别对应 8/12/14bit 测量。SHT10 通过下拉 DATA 至低电平，表示测量结束。控制器触发 SCK 时钟前，必须等待这个“数据备妥”信号才能将测量数据正确读入。测量和通讯结束后，SHT10 自动转入休眠模式。数据传送采用两线制串行接口(与 I2C 接口不兼容)。

2.3 SHT10 电路原理图

SHT10 采用 LCC 封装，其 DATA 和 SCK 引脚分别连接到 PXA310 的 GPIO78 和 GPIO79。PXA310 通过模拟时序方式实现对外围温湿度传感器的控制和数据读写操作。由于 SHT10 对于温湿度灵敏度很高，

① 基金项目:上海市科委重点项目(075115002)
收稿时间:2009-03-10

在系统集成时应尽量远离发热源(如 MCU、LCD 等)，否则测量结果会有所偏离；为 SHT10 布线时，周围应尽量铺地减少周围器件对其的干扰。SHT10 电路原理图如图 1 所示。

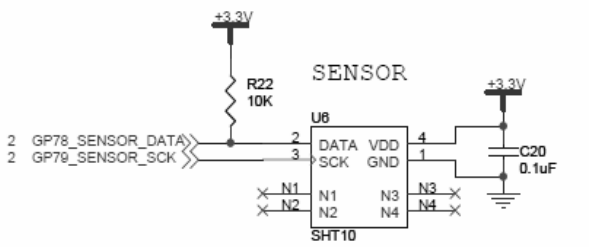

图 1　SHT10 电路原理图

3　Linux温湿度传感器驱动程序实现

单片机控制的传感器设备中，单片机通常是单线程运行。在进行温湿度测量时，单片机需要等待测试结果返回，其方法阻碍了其他测试和操作的同步执行。在嵌入式 Linux 系统中，驱动程序将测试任务送入任务队列，交出 CPU 控制权，继而进行其他实时任务运行，待内核空闲再进入任务队列完成传感器的测量，以此提高系统执行的效率和实时性。

3.1 Linux 温湿度传感器设备加载[3]

温湿度传感器使用 Linux 内核的 Miscdevice 数据结构在驱动程序初始化时将设备注册到内核。Miscdevice 是字符设备，其主设备号为 10，设备及设备接口函数定义如下所示。

```
static struct file_operations sht10_fops = {
  owner:THIS_MODULE,        //所属的设备模块
  read:  sht10_read,        //数据读取操作
};
static struct miscdevice my_sht10 ={
  .minor=4,                 //次设备号为 4
  .name="SHT10",            //设备名称 SHT10
  .fops=&sht10_fops,        //设备可用相关操作
};
```

驱动程序加载设备时将调用内核的注册函数。在 Linux2.4 和 2.6 内核中，几乎所有 Linux 驱动程序都依靠如下函数加载模块[4]。

```
static int __init sht10_init(void)
{
misc_register(&my_sht10);     //注册 SHT10 设备
return 0;                     //操作成功返回 0
}
```

驱动程序初始化完成后，上层应用程序可以调用 sht10_fops 中的 sht10_read 函数进行温湿度的读取操作。

3.2 Linux 温湿度传感器设备操作

进行数据读取前，首先要在驱动程序中开辟 4 个字节的数据空间，用于存放温度和湿度测量值。这里定义全局变量数据缓冲区为 unsigned char buf[4]。

读取 SHT10 温湿度数据前，需要进行端口初始化和 SHT10 复位操作，然后将任务送于任务队列并阻塞线程[5]，当任务完成返回后再唤醒线程，将读到数据传递给上层应用程序做进一步处理。程序流程图和实现函数如图 2 所示。

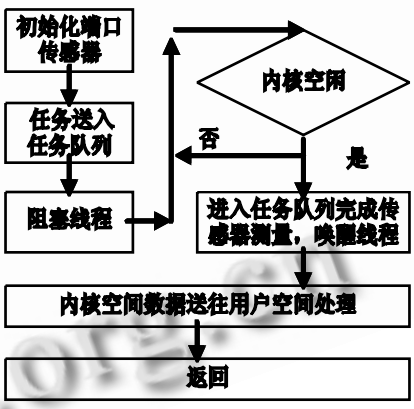

图 2　驱动程序流程图

```
static int measure_sht10(u8 checksum,u8 mode)
{
unsigned char error=0;   //设备无应答，标识清 0
int i=0;                 //用来指示数据存放位置
start_trans();           //模拟时序，启动传输
switch(mode)             //选择测量方式
{
case TEMP:error+=write_byte (MEASURE_TEMP);
```

```
i=1;break;          //测量温度，指示存放位置
case HUMI:error+=write_byte(MEASURE_HUMI);
break;              //测量湿度
}
while(1){           //等待 SHT10 应答，退出
if(read_data()==0) break;
}
if(i){              //查看 i，存放数据
buf[0]=read_byte(ACK);   //将测量温度数据存放于
buf[1]=read_byte(ACK);   //buf[0]和 buf[1],并应答
}
else{
buf[2]=read_byte(ACK);   //将测量湿度数据存放于
buf[3]=read_byte(ACK);   //buf[2]和 buf[3]并应答
}                        //最后读效验，无应答
checksum=read_byte(noACK);
return error;       //返回错误标识
}
```

上述函数中 start_trans; write_byte; read_data; read_byte 分别利用 PXA310 引脚模拟时序完成启动传输、写字节，读一位数据和读字节的操作。

```
static ssize_t sht10_read(struct file *file,char *buffer,size_t count,loff_t *ppos)
{
port_init();            //初始化 PXA310 端口
reset_sht10();          //复位 SHT10
tasklet_schedule(&sht10_tasklet);
                        //将任务送于任务队列
wait_for_completion(&comp);
                        //阻塞线程，等待完成
copy_to_user(buffer,(char *)&buf,sizeof(buf));
                        //将读到的数据返回用户
return 0;           //空间,退出
}
```

内核 tasklet_schedule() 调度执行指定的 tasklet，在获得运行机会之前只会调度一次，如果在运行时被调度，则完成后会被再次运行[6]。wait_for_completion()这个函数进行一个不可打断的等待,如果有代码调用它，并且没有完成这个任务，结果会是一个不可杀死的进程。copy_to_user()将内核空间数据传向上层用户空间，并让上层测试程序做进一步处理。

3.3 Linux 温湿度传感器设备阻塞操作

由于温湿度传感器测量需要一定时间，为提高系统运行效率和实时性，在驱动程序中阻塞线程，交出内核控制权，等待操作完成后唤醒线程，提高系统利用率。complete()在函数中就是唤醒一个等待的读取线程。任务队列实现函数如下所示。

```
static int sht10_do_tasklet(void)
{
unsigned int error=0;     //无应答,标识清 0
unsigned char checksum=0;  //效验清 0
error+=measure_sht10(checksum,TEMP);
error+=measure_sht10(checksum,HUMI);
complete(&comp);          //完成测量，唤醒线程
if(error!=0){             //测量有误，提示
printk("wrong in measure error==>%d\n",error);
}
else{
printk("data correct!\n");  //测量无误输出提示
}
return error;             //返回错误标识
}
```

Tasklet 可以使测量操作在系统负荷不重时被调用，或是被立即执行，但始终不会晚于下一个 CPU clock。Tasklet 始终在中断期间运行，并且在调度他的同一 CPU 上运行。对比单片机系统，在单线程情况下，一般在 sht10_read()中调用 2 次 measure_ sht10()来等待测量完成，测量效率依赖 2 次测量消耗的时间；但在 Linux 驱动程序中，使用 Tasklet 方式操作，2 次测量过程不会对其他线程产生影响，在有其他实时事件需要及时处理时(如网络，视频)，可以更有效提高驱动运行效率，降低对其他实时处理产生的影响。

4　温湿度传感器测试与验证

驱动程序完成以后，需要相应测试程序验证驱动程序编写的正确性。由于驱动程序中不能对数据进行浮点数运算，所以测试程序必须将驱动程序传递来的数据进行浮点数运算才能得到相应的温湿度值。

4.1 温湿度传感器测试环境

在实验室常温下，测试程序多次调用驱动程序中读温湿度的函数接口获得测试数据，来验证设计的正确和可靠。并考虑实验室内常温下，相对湿度与温度具有非线性关系，计算湿度值时需要考虑温度的补偿关系，其关系如图 3 所示。

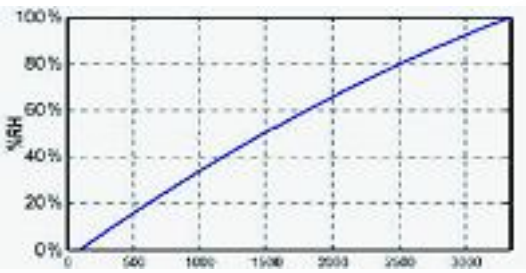

图 3　SORH 转换到相对湿度

为补偿湿度传感器的非线性以获取准确数据，并考虑实际温度与测试参考温度(25℃)不同，使用如下公式修正读数。

$RH_{linear}=C_1+C_2x\ SO_{RH}+C_3\ x\ SO^2RH$

$RH_{true}=(t_c-25)\ x\ (t_1+t_2x\ SORH)+RH_{linear}$

RH_{linear} 是温度修正系数，RHtrue 是相对湿度，SORH 是传感器返回的湿度值。进行 12bit 湿度检测时，参数取值如下表所示。

表 1　湿度转换系数与温度补偿系数

SO_{RH}	C_1	C_2	C_3	t_1	t_2
12bit	-4	0.0405	$-2.8*10^{-6}$	0.01	0.00008

由于能隙材料研发的温度传感器具有极好线性,14bit 温度值参考如下公式。

$$Temperature = d_1+d_2\ x\ SOT$$

温度转换系数取值如下表所示，SOT 是传感器返回的温度值。

表 2　温度转换系数

VDD	d_1[℃]	SO_T	d_2[℃]
3.3v	-39.66	14bit	0.01

利用上述温湿度转换公式和系数可以得出温湿度测量值。

4.2 温湿度传感器测试途径与效率验证

在测试程序中，考虑上述测量环境下温湿度之间的非线性，调用驱动程序的 sht10_read 函数将读到的温湿度数据返回上层测试程序进行浮点数运算，将计算值通过串口输出，达到测试验证的目的。测试程序的实现如下所示。

```
static void calc_sht10(float *humi, float *temp)
{
float rh=*humi;
float t=*temp;
float rh_line;
float rh_true;
t=t*d2+d1;                    //温度转换公式
rh_line=C3*rh*rh+C2*rh+C1; //相对湿度转换公式
rh_true=(t-25)*(t1+t2*rh)+rh_line;
                              //相对湿度温度补偿
if(rh_true>100)rh_true=100; //超出范围
if(rh_true<0.1)rh_true=0.1;
printf("Humidity     is: %.2f%RH\n",rh_true);
printf("Temperature is: %.2f'C\n",t);
}
int main(int argc, char *argv[]) //主函数
{
int fd;
float temp,humi;              //温湿度数据
char buffer[4];               //数据缓冲
fd = open("/dev/sht10", 0);     //打开文件
if (fd < 0) {                 //打开失败，退出
perror("open device /dev/sht10");
exit(1);
}
read(fd,buffer,sizeof(buffer));//读取温湿度值
temp=(float)((buffer[0]<<8)|buffer[1]);
humi=(float)((buffer[2]<<8)|buffer[3]);
calc_sht10(&humi, &temp);     //温湿度数值转换
```

```
close(fd);                    //关闭文件
return 0;                     //退出
}
```

测试完成后，考察驱动程序运行效率，即在驱动程序的 tasklet_schedule 和 copy_to_user 前分别对 PXA310 的 OSCR 时间计数寄存器进行时间读取，计算此次温湿度测量所用时间。计算公式如下所示。

$$Time=(OSCR_2-OSCR_1)/OSCR\_FREQ$$

$OSCR_2$ 是唤醒线程后的时间，$OSCR_1$ 是进入任务队列前的时间。OSCR_FREQ 是 PXA310 内部时钟频率 3.25MHz。这样就可以计算出每次温湿度读取消耗的时间，以此对比 SHT10 开发文档中理论测量时间值，确定实际驱动程序运行的效率。

5 实验结果与分析

超级终端中插入驱动模块，运行测试程序，可以在终端上看到测试结果(如图 4)。

```
[root@Linux /root]#insmod sht10.ko
Using sht10.ko
[root@Linux /root]#./sht10
data correct!
Humidity          is:     37.45%RH
Temperature       is:     23.72'C
[root@Linux /root]#
```

图 4　超级终端测试结果

系统功能实现后，利用上述 Time 计算公式计算驱动程序中温湿度测量消耗的时间，实际测试结果如表 3 所示。

表 3　驱动程序中实际测量消耗的时间

测量	1	2	3	4	5	平均
温湿度	1.29s	1.24s	1.33s	1.31s	1.3s	1.29s

上表的测试结果不仅和传感器的响应速度有关，而且还与系统中其他运行的线程有关。当系统中有高一级任务到来或其他实时事件需要处理时，实际测量时间会大于上表中的测量时间，并且随着任务的增加测量时间也会相应的增加，完成的时间也受到外界中断的影响。内核会在任务不繁忙时完成测量操作。上表测试结果并未受到系统中其他驱动程序和中断的影响。对比开发手册中理论测量时间可以看到，使用任务队列的方法对改善系统处理能力与实时性效果明显。

此外，实现温湿度传感器驱动程序还需要清楚了解 SHT10 读写时序，读取温度和湿度所需要的时间不同。如果应用程序中得出的温湿度值超过预期值，就可以打开 GPIO 驱动模块，触发系统板上的蜂鸣器达到预警效果。

6 结语

此设计方案已经应用于嵌入式无声交互控制系统的检测，并且运行正常。实践证明，该嵌入式 Linux 温湿度传感器设计方案可行有效，线程阻塞提高系统运行效率，在环境测量准确度和系统实时性方面得到了令人满意的效果。由于此方案基于 Linux 操作系统和 PXA310 平台，其在多任务、实时快速处理上具有一定的优势。

参考文献

1 清风电子.18B20.pdf,2007.5.http://www.mcubbs.net

2 Sensirion. C-Datasheet_SHTxx_2.04.pdf,2005.5. http://www.sensi- rion.com/humidity

3 王粉花.基于 LINUX 设备驱动程序的设计与实现.计算机工程, 2006,32(23):278－280.

4 刘淼.嵌入式系统接口设计与 Linux 驱动程序开发.北京:北京航空航天大学出版社, 2006.5.

5 宋宝华.LINUX 设备驱动开发详解.北京:人民邮电出版社, 2008.

6 魏永明,狄岳,钟书毅.LINUX 设备驱动程序.第 3 版.北京:中国电力出版社, 2006.