1. 温湿度传感器硬件



图1-1 温湿度传感器原理图

温湿度传感器需要将DATA引脚和SCK引脚与单片机相连，所以使用单片机的P0\_6和P0\_7引脚分别连接到DATA和SCK口，并且给两个引脚发送相应的时序就能控制温湿度传感器。

* SHT10的DATA口接CC2531的P0\_6   //定义通讯数据端口
* SHT10的SCK口接CC2531的P0\_7    //定义通讯时钟端口

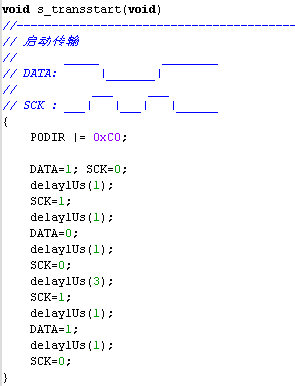
1. 温湿度传感器软件
   1. 主要代码

要驱动SHT10进行采样必须发送如下命令：首先，向SHT10发送“启动传输”时序，完成数据传输的初始化。如图4-3所示，时序包括当SCK时钟高电平时，DATA发转为低电平；紧接着SCK变为低电平，随后在SCK时钟高电平时，DATA翻转为高电平。初始化之后，单片机便可以向SHT10发送命令。通常的命令包括3个地址位（目前只支持“000”）和5个命令位，具体将在后面的代码中进行介绍。SHT10会以下述方式表示已正确的接收到命令：在第8个SCK时钟的下降沿之后，将DATA下拉为低电平，并且在第9个SCK时钟的下降沿之后，将DATA位恢复为高电平。



图2-1 启动传输时序

启动传输程序：



启动程序完成之后，SHT10便会以串行数据的方式与单片机进行通信，时序图如图4-4所示。DATA三态门用于数据的读取，DATA在SCK时钟下降沿之后改变状态，并仅在SCK时钟上升沿有效。数据传输期间，在SCK时钟高电平时，DATA必须保持稳定。为避免信号冲突，单片机应驱动DATA在低电平。

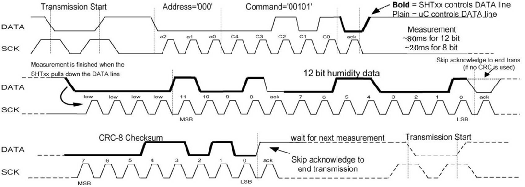
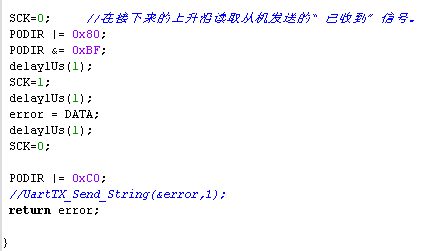


图2-2 RH测量时序举例：“0000’1001’0011’0001”=2353=75.79%RH

(未包含温度补偿)

给传感器发送数据需要注意时序，在上升沿之前把数据写入，上升沿时数据有效，在下降沿时把数据发送给传感器。数据发送的相关代码：





从传感器读数据，同样也要注意时序，只有在上升沿数据有效之后才能读。读数据的相关代码：



温湿度测量的相关代码：



* 1. 数据结果换算
  + 计算温度的公式为： Temperature = a + b \* X

式中，a，b为参数；X为传感器采集的温度数据。

对于本实验采用3V电压、14位采样精度，查芯片手册可知：a = --39.67，b = 0.01。带入计算即可得到以摄氏度为单位的温度值。在极端工作条件下测量温度时，可使用进一步的补偿算法以获取高精度。可参阅应用说明“相对湿度与温度的非线性补偿”。

* + 计算湿度的公式为： Humidity = c + d \* Y + e\* Y^2

式中，c，d，e为参数；Y为传感器采集的湿度数据。

对于本实验采用12位采样精度，查芯片手册可知：c = -4，d = 0.0405，e = -2.8 \* 10^-6。

1. 源码注释

Main.c

#include "ioCC2530.h"

#include <string.h>

#include "sht10.h"

#include <stdio.h>

//#define uint unsigned int

//#define uchar unsigned char

//#define Uint16 unsigned int

//定义控制灯的端口

#define led1 P1\_0

#define led2 P1\_1

//函数声明

void Delay1(uint);

void initUARTtest(void);

void UartTX\_Send\_String(char \*Data,int len);

void initLED(void);

union

{ unsigned int i;

float f;

}humi\_val,temp\_val; //定义两个共同体，一个用于湿度，一个用于温度

char temp\_result[15] = {0}; //温度读取结果

float temp\_value = 0; //温度数值

char humi\_result[15] = {0}; //湿度读取结果

float humi\_value = 0; //湿度数值

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数功能 ：主函数

\*入口参数 ：无

\*返 回 值 ：无

\*说 明 ：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

unsigned char error,checksum;

unsigned char HUMI,TEMP;

HUMI=0X01;

TEMP=0X02;

initUARTtest(); //初始化串口

s\_connectionreset(); //复位

while(1)

{

error=0;

error+=s\_measure((unsigned char\*) &humi\_val.i,&checksum,HUMI); //湿度测量

error+=s\_measure((unsigned char\*) &temp\_val.i,&checksum,TEMP); //温度测量

temp\_value = temp\_val.i \* 0.01 - 39.6; //温度值换算

sprintf(temp\_result,"%s","temperature: "); //打印为字符串

UartTX\_Send\_String(temp\_result, 13); //传出字符串

sprintf(temp\_result,"%9.2f",temp\_value); //打印为浮点数

temp\_result[9] = '\n'; //添加换行

UartTX\_Send\_String(temp\_result, 10); //串口传出结果

humi\_value = humi\_val.i \* 0.0367 - 2.0468; //湿度值换算

sprintf(humi\_result,"%s","humidity: "); //打印为字符串

UartTX\_Send\_String(humi\_result, 10); //传出字符串

sprintf(humi\_result,"%9.4f",humi\_value); //打印为浮点数

humi\_result[9] = '\n'; //添加换行

UartTX\_Send\_String(humi\_result, 10); //串口传出结果

if(error!=0)

{

s\_connectionreset() ; //如果发生错误，系统复位

led1 = !led1; //修改led1状态

led2 = !led2; //修改led2状态

}

else

{

humi\_val.f=(float)humi\_val.i; //转换为浮点数

temp\_val.f=(float)temp\_val.i; //转换为浮点数

calc\_sth11(&humi\_val.f,&temp\_val.f); //修正相对湿度及温度

//dew\_point=calc\_dewpoint(humi\_val.f,temp\_val.f); //计算绝对湿度值

}

// UartTX\_Send\_String(&error,1);

Delay1(50000); //延时

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数功能 ：延时

\*入口参数 ：定性延时

\*返 回 值 ：无

\*说 明 ：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay1(uint n)

{

uint i;

for(i=0;i<n;i++);

for(i=0;i<n;i++);

for(i=0;i<n;i++);

for(i=0;i<n;i++);

for(i=0;i<n;i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数功能 ：初始化串口1

\*入口参数 ：无

\*返 回 值 ：无

\*说 明 ：57600-8-n-1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initUARTtest(void)

{

CLKCONCMD &= ~0x40; //晶振

while(!(SLEEPSTA & 0x40)); //等待晶振稳定

CLKCONCMD &= ~0x47; //TICHSPD128分频，CLKSPD不分频

SLEEPSTA |= 0x04; //关闭不用的RC振荡器

PERCFG = 0x01; //位置1 串口0

P1SEL |= 0x30; //P1用作串口

U0CSR |= 0x80; //UART方式

U0GCR |= 8; //baud\_e

U0BAUD |= 59; //波特率设为9600

UTX0IF = 1;

U0CSR |= 0X40; //允许接收

IEN0 |= 0x84; //开总中断，接收中断

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*函数功能 ：串口发送字符串函数

\*入口参数 : data:数据

\* len :数据长度

\*返 回 值 ：无

\*说 明 ：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UartTX\_Send\_String(char \*Data,int len)

{

int j;

for(j=0;j<len;j++)

{

U0DBUF = \*Data++;

while(UTX0IF == 0); //每次发送完数据串口发送中断位硬件自动设置为1

UTX0IF = 0; //将软件设置为0

}

}

SHT10.C

#include <math.h>

#include "sht10.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn delay1Us

\*

\* @brief wait for x us. @ 32MHz MCU clock it takes 32 "nop"s for 1 us delay.

\*

\* @param x us. range[0-65536]

\*

\* @return 延时约为0.4us

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay1Us(Uint16 microSecs)

{

while(microSecs--)

{

/\* 32 NOPs == 1 usecs \*/

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

延时 1uS 带参数(int)子程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay (unsigned int time){

unsigned int a;

for(a=0;a<time;a++)

{

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop");

}

}

//----------------------------------------------------------------------------------

void s\_connectionreset(void)

//----------------------------------------------------------------------------------

// 连接复位;

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

// DATA: |\_\_\_\_\_\_\_|

// \_ \_ \_ \_ \_ \_ \_ \_ \_ \_\_\_ \_\_\_

// SCK : \_\_| |\_\_| |\_\_| |\_\_| |\_\_| |\_\_| |\_\_| |\_\_| |\_\_| |\_\_\_\_\_\_| |\_\_\_| |\_\_\_\_\_\_

{

unsigned char i;

P0DIR |= 0xC0;

DATA=1; SCK=0; //准备

for(i=0;i<9;i++) //DATA保持高，SCK时钟触发9次，发送启动传输，通迅即复位

{ SCK=1;

SCK=0;

}

s\_transstart(); //启动传输

}

//----------------------------------------------------------------------------------

void s\_transstart(void)

//----------------------------------------------------------------------------------

// 启动传输

// \_\_\_\_\_ \_\_\_\_\_\_\_\_

// DATA: |\_\_\_\_\_\_\_|

// \_\_\_ \_\_\_

// SCK : \_\_\_| |\_\_\_| |\_\_\_\_\_\_

/\*首先，向SHT10发送“启动传输”时序，完成数据传输的初始化。如图4-3所示，时序包括当SCK时钟高电平时，DATA发转为低电平；紧接着SCK变为低电平，随后在SCK时钟高电平时，DATA翻转为高电平。初始化之后，单片机便可以向SHT10发送命令。通常的命令包括3个地址位（目前只支持“000”）和5个命令位，具体将在后面的代码中进行介绍。SHT10会以下述方式表示已正确的接收到命令：在第8个SCK时钟的下降沿之后，将DATA下拉为低电平，并且在第9个SCK时钟的下降沿之后，将DATA位恢复为高电平。\*/

{

P0DIR |= 0xC0;

DATA=1; SCK=0;

delay1Us(1);

SCK=1;

delay1Us(1);

DATA=0;

delay1Us(1);

SCK=0;

delay1Us(3);

SCK=1;

delay1Us(1);

DATA=1;

delay1Us(1);

SCK=0;

}

//----------------------------------------------------------------------------------

char s\_measure(unsigned char \*p\_value, unsigned char \*p\_checksum, unsigned char mode)

//----------------------------------------------------------------------------------

// 进行温度或者湿度转换，由参数mode决定转换内容；

{

unsigned error=0;

s\_transstart(); //启动传输

switch(mode){

case 0x02 : error+=s\_write\_byte(MEASURE\_TEMP); break;

case 0x01 : error+=s\_write\_byte(MEASURE\_HUMI); break;

default : break;

}

P0DIR |= 0x80;

P0DIR &= 0xBF;

while(DATA); //等待测量结束；

if(DATA) error+=1; // 如果长时间数据线没有拉低，说明测量错误

\*(p\_value+1) =s\_read\_byte(ACK); //读第一个字节，高字节 (MSB)

\*(p\_value)=s\_read\_byte(ACK); //读第二个字节，低字节 (LSB)

\*p\_checksum =s\_read\_byte(noACK); //read CRC校验码

return error;

}

//----------------------------------------------------------------------------------

char s\_write\_byte(unsigned char value)

//----------------------------------------------------------------------------------

// 写字节函数

{

char i;

char error=0;

P0DIR |= 0xC0;

SCK=0;

DATA=0;

for(i=0;i<8;i++) //发送8位数据，丛机将在上升沿读取数据

{

SCK=0;

if(value&(0x80>>i))

DATA=1;

else

DATA=0;

delay1Us(1);

SCK=1;

delay1Us(1);

}

SCK=0; //在接下来的上升沿读取从机发送的“已收到”信号。

P0DIR |= 0x80;

P0DIR &= 0xBF;

delay1Us(1);

SCK=1;

delay1Us(1);

error = DATA;

delay1Us(1);

SCK=0;

P0DIR |= 0xC0;

//UartTX\_Send\_String(&error,1);

return error;

}

//----------------------------------------------------------------------------------

char s\_read\_byte(unsigned char ack)

//----------------------------------------------------------------------------------

// 读数据；

{

unsigned char i,val=0;

//DATA=1; //数据线为高

P0DIR |= 0x80;

P0DIR &= 0xBF;

SCK=0;

for (i=0x80;i>0;i>>=1) //右移位

{

SCK=1; //在上升沿数据有效之后才能读

delay1Us(1);

if (DATA)

val=(val | i); //读数据线的值

SCK=0;

delay1Us(1);

}

P0DIR |= 0xC0;

DATA=!ack; //如果是校验，读取完后结束通讯；

SCK=1;

delay1Us(3);

SCK=0;

DATA=1; //释放数据线

return val;

}

//----------------------------------------------------------------------------------------

void calc\_sth11(float \*p\_humidity ,float \*p\_temperature)

//----------------------------------------------------------------------------------------

// 补偿及输出温度和相对湿度

{ const float C1=-4.0; // for 12 Bit 湿度修正公式

const float C2=+0.0405; // for 12 Bit 湿度修正公式

const float C3=-0.0000028; // for 12 Bit 湿度修正公式

const float T1=+0.01; // for 14 Bit @ 5V 温度修正公式

const float T2=+0.00008; // for 14 Bit @ 5V 温度修正公式

float rh=\*p\_humidity;

float t=\*p\_temperature;

float rh\_lin;

float rh\_true;

float t\_C;

t\_C=t\*0.01 - 39.66; //补偿温度

rh\_lin=C3\*rh\*rh + C2\*rh + C1; //相对湿度非线性补偿

rh\_true=(t\_C-25)\*(T1+T2\*rh)+rh\_lin; //相对湿度对于温度依赖性补偿

if(rh\_true>100)rh\_true=100; //湿度最大修正

if(rh\_true<0.1)rh\_true=0.1; //湿度最小修正

\*p\_temperature=t\_C; //返回温度结果

\*p\_humidity=rh\_true; //返回湿度结果

//UartTX\_Send\_String((int\*)&t\_C,1);

//UartTX\_Send\_String((int\*)&rh\_true,1);

}

//--------------------------------------------------------------------

float calc\_dewpoint(float h,float t)

//--------------------------------------------------------------------

// 计算绝对湿度值

{ float logEx,dew\_point;

logEx=0.66077+7.5\*t/(237.3+t)+(log10(h)-2);

dew\_point = (logEx - 0.66077)\*237.3/(0.66077+7.5-logEx);

return dew\_point;

}