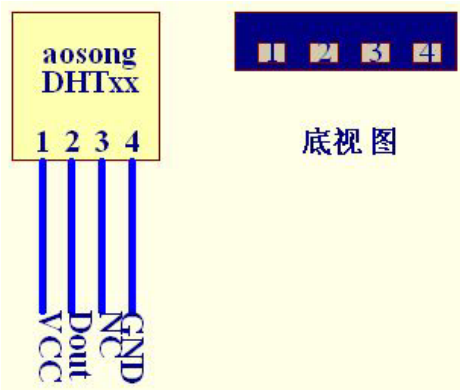
DHT11 数字湿温度传感器的原理和应用范例

概述：DHT11 是广州奥松有限公司生产的一款湿温度一体化的数字传感器。该传感器包括一个电阻式测湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。通过单片机等微处理器简单的电路连接就能够实时的采集本地湿度和温度。DHT11 与单片机之间能采用简单的单总线进行通信，仅仅需要一个 I/O 口。传感器内部湿度和温度数据 40Bit 的数据一次性传给单片机，数据采用校验和方式进行校验，有效的保证数据传输的准确性。DHT11 功耗很低，5V 电源电压下，工作平均最大电流 0.5mA。

性能指标和特性如下：

- 工作电压范围：3.5V-5.5V
- 工作电流　　：平均 0.5mA
- 湿度测量范围：20－90%RH
- 温度测量范围：0－50℃
- 湿度分辨率　：1%RH　8 位
- 温度分辨率　：1℃　　8 位
- 采样周期　　：1S
- 单总线结构
- 与 TTL 兼容（5V）

管脚排列如下：



引脚说明：

Vcc　正电源
Dout　输出
NC　空脚
GND　地

1、应用电路连接说明

DHT11 数字湿温度传感器连接方法极为简单。第一脚接电源正，第四脚接电源地端。数据端为第二脚。可直接接主机（单片机）的 I/O 口。为提高稳定性，建议在数据端和电源正之间接一只 4.7K 的上拉电阻。第三脚为空脚，此管脚悬空不用。

2、DHT11 数据结构

DHT11数字湿温度传感器采用单总线数据格式。即，单个数据引脚端口完成输入输出双向传输。其数据包由5Byte（40Bit）组成。数据分小数部分和整数部分,具体格式在下面说明。

一次完整的数据传输为40bit，高位先出。

数据格式：8bit湿度整数数据+8bit湿度小数数据
+8bit温度整数数据+8bit温度小数数据
+8bit校验和

校验和数据为前四个字节相加。

传感器数据输出的是未编码的二进制数据。数据(湿度、温度、整数、小数)之间应该分开处理。如果，某次从传感器中读取如下5Byte数据：

byte4	byte3	byte2	byte1	byte0
00101101	00000000	00011100	00000000	01001001
整数	小数	整数	小数	校验和
湿度		**温度**		**校验和**

由以上数据就可得到湿度和温度的值，计算方法：

humi (湿度)= byte4 . byte3=45.0 (%RH)

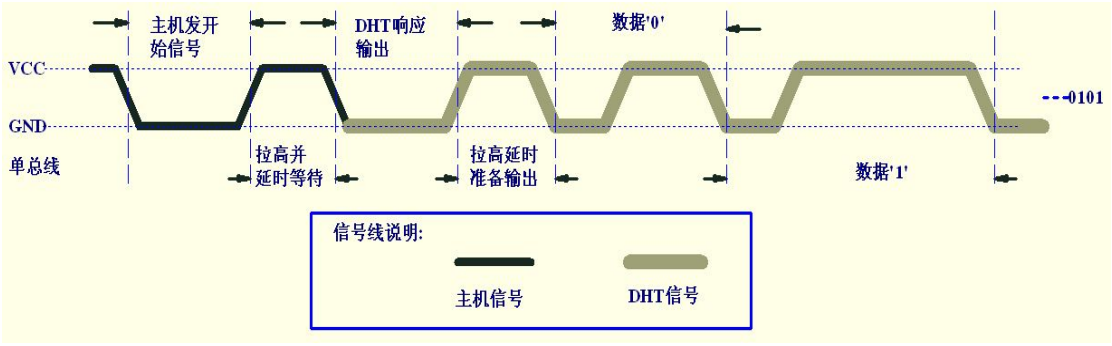
temp (温度)= byte2 . byte1=28.0 (℃)

jiaoyan(校验)= byte4+ byte3+ byte2+ byte1=73(=humi+temp)(校验正确)

注意：DHT11一次通讯时间最大3ms，主机连续采样间隔建议不小于100ms。

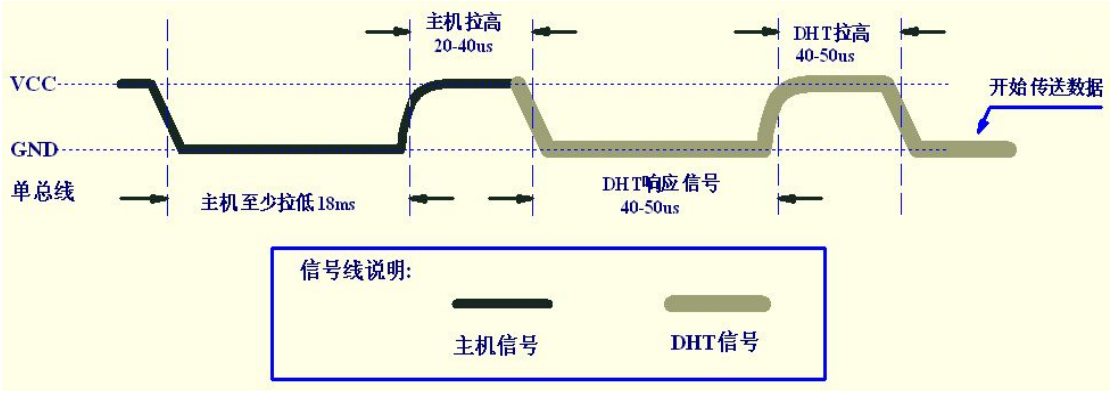
3、DHT11的传输时序

3.1、DHT11 开始发送数据流程

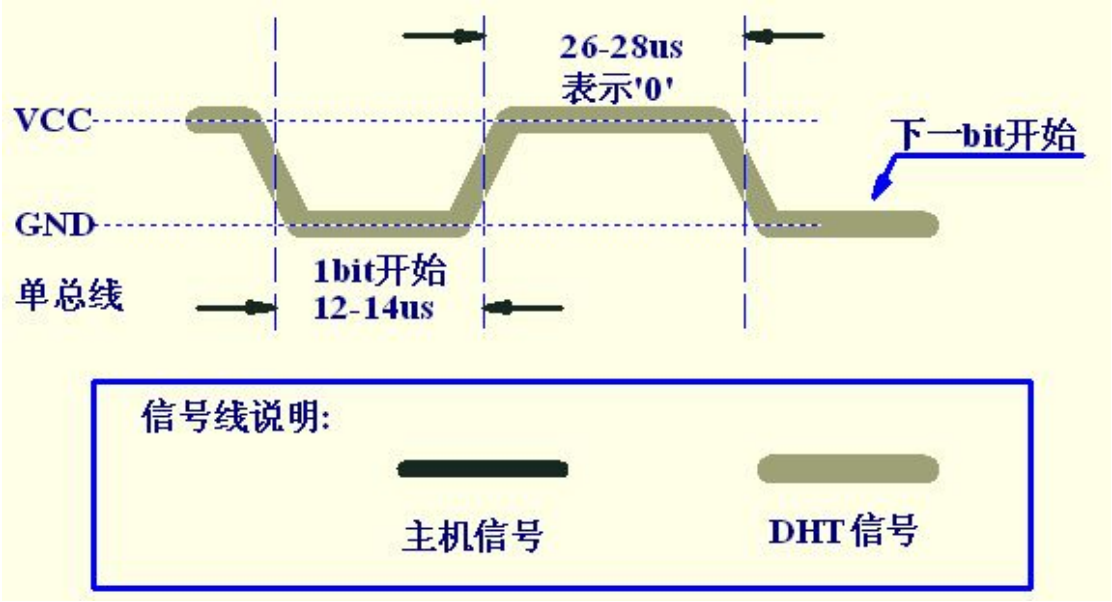


主机发送开始信号后,延时等待 20us-40us 后读取 DH11T 的回应信号，读取总线为低电平,说明 DHT11 发送响应信号，DHT11 发送响应信号后，再把总线拉高,准备发送数据,每一 bit 数据都以低电平开始,格式见下面图示。如果读取响应信号为高电平,则 DHT11 没有响应,请检查线路是否连接正常。

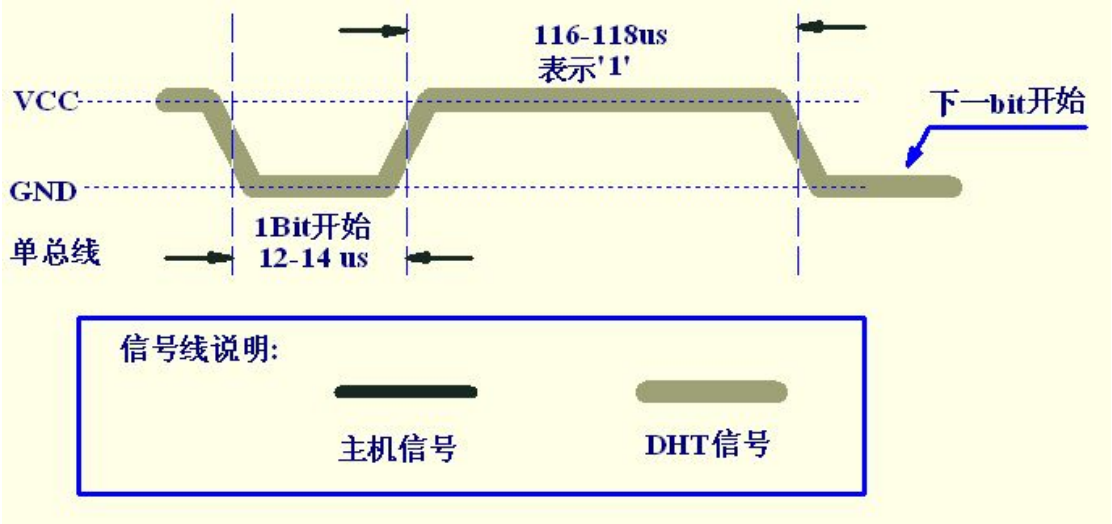
3.2 主机复位信号和 DHT11 响应信号



3.3 数字‘0’信号表示方法



3.4 数字‘1’信号表示方法



4、实例应用

4.1 硬件描述

本项目采用两个三位一体共阳数码管做为湿度，温度显示模块。采用 ATMEL 公司的 ATmega8 做为主控芯片，详细如图 4.1 所示。

4.2 管脚分配

设备		管脚
传感器(HDT11)		PC5
S2(温度显示)	LED[1..8]	PB[0..7]
	P3	PC3
	P4	PD5
	P5	PD4
S1(湿度显示)	LED[1..8]	PB[0..7]
	P1	PD7
	P2	PD6
	Pm3	PD0

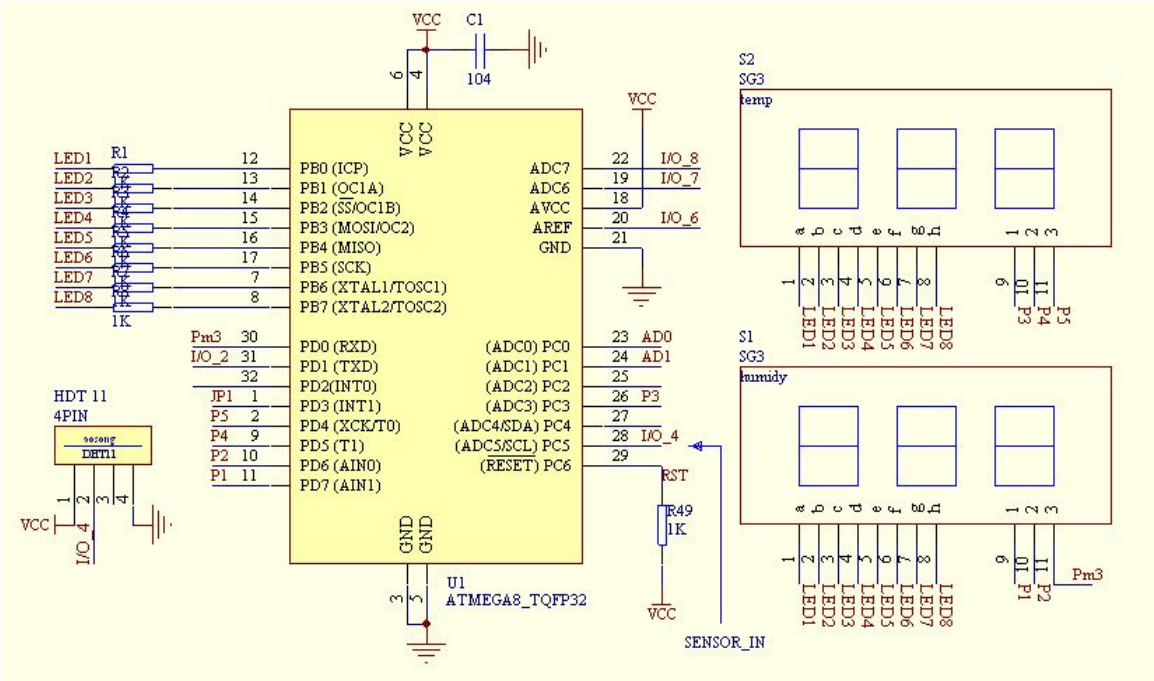


图 4.1　典型实例原理图

/*原程序清单如下所示，本程序在 ICCAVR　6.31A 下测试通过。*/

```
/*****************************************************************************
 编译环境 ：ICCAVR
 功能：读取湿温度传感器 DTH11 数据并显示
 公司：奥松电子有限公司
 芯片：Mega 8
 晶振：内部 8.0000MHz
 创建人：木工
 修改 ：宁佐文
*****************************************************************************/
#include <iom8v.h>
#include <macros.h>
typedef unsigned   char   U8;       // defined for unsigned 8-bits integer variable
typedef signed     char   S8;       // defined for signed 8-bits integer variable
typedef unsigned   int    U16;      // defined for unsigned 16-bits integer variable
typedef signed     int    S16;       // defined for signed 16-bits integer variable
typedef unsigned   long   U32;      // defined for unsigned 32-bits integer variable
typedef signed     long   S32;      // defined for signed 32-bits integer variable
typedef float             F32;      // single precision floating point variable (32bits)
typedef double            F64;     // double precision floating point variable (64bits)
/*****************************************************************************
                          数码管显示赋值区
*****************************************************************************/
#define       _a        0x01
#define       _b        0x02
#define       _c        0x04
#define       _d        0x08
#define       _e        0x10
#define       _f        0x20
#define       _g        0x40
#define       _dp       0x80

#define       num0          _g                    //灭的数码管
#define       num1          _a|_d|_e|_f|_g
#define       num2          _c|_f
#define       num3          _e|_f
#define       num4          _a|_d|_e
#define       num5          _b|_e
#define       num6          _b
#define       num7          _d|_e|_f|_g
#define       num8          0x00
#define       num9          _e
#define       num_           _a|_b|_c|_d|_e|_f|_dp
```

```
#define     num_11      _a|_d|_e|_f
#define     num_E       _b|_c|_dp                   // Err 代码
#define     num_r       _a|_b|_c|_d|_f|_dp

#define     dc2         0x04
#define     dc3         0x08
#define     dd          0x03
#define     dd4         0x10
#define     dd5         0x20
#define     dd6         0x40
#define     dd7         0x80
#define     dd2         0x04
#define     dd3         0x08            //数码管控制端口

#define     led1        dc3             //使用时要先将数码管全熄灭掉
#define     led2        dd5
#define     led3        dd4
#define     led4        dd7
#define     led5        dd6
#define     led6        dd2

U8 PLAY[6]={0,0,0,0,0,0};
unsigned char Flag = 0;
unsigned int u16ReadDownStreamTimer;
const U8 LED[14]={num0,num1,num2,num3,num4,num5,num6,num7,num8,num9,
                  num_,num_11,num_E,num_r };        //数码管笔形码数组
//函数声明：
void init_devices(void);
void time1_start(void);
void port_init(void);
void timer0_init(void);
void timer1_init(void);
void Delay(unsigned int i);
void DelayNS (U16 dly) ;
//*****************************************************************************
//                              显示模块
//*****************************************************************************
void Display (void)
  {
    U8 i;
    for(i=0;i<=5;i++)
      {
        DDRD |= 0XFC;
        PORTD&=0x03;
```

```
            DDRC |= 0X0C;
            PORTC&=0xF3;
            Delay(10);
            switch(i)
              {
                case 0: PORTC|=led1;break;
                case 1: PORTD|=led2;break;
                case 2: PORTD|=led3;break;
                case 3: PORTD|=led4;break;
                case 4: PORTD|=led5;break;
                case 5: PORTD|=led6;break;
                default :break;
              }
           PORTB=LED[PLAY[i]] ;              //送数据
           if((i==0)|(i==2)|(i==3)|(i==5))       //消小数点
             {
               PORTB|=0x80;
             }
           DelayNS(1);
         }
       PORTD&=0x03;
       PORTC&=0xF3;
   }
//*****************************************************************************
//                  delay()
//*****************************************************************************
void DelayNS (U16 dly)                      //延时子程序       1.010ms
   {
      U16 i;
      for ( ; dly>0; dly--)WDR();
      for (i=0; i<1131; i++);
   }
void Delay(unsigned int i)
   {
      while(i--);
   }
void delay1N(unsigned int Dcount)
{
   while(Dcount--)
   {
     NOP();NOP();NOP();
   }
}
void Timer2_Delay150ms(void)         // Timer 2
```

```
  {
    TCCR2 = 0x00;
    TCNT2 = 0xDE;
    TIFR |= 0X40;
    TCCR2 = 0x02;
  }
//*****************************************************************************
//                          系统初始化函数
//*****************************************************************************
void init_devices(void)
  {
      CLI();
      port_init();
      timer0_init();
      TIMSK = 0x05;
      SEI();
  }
//*****************************************************************************
//                          端口初始化
//*****************************************************************************
void port_init(void)
  {
      DDRB  = 0xFF;
      PORTB = 0x00;
      DDRC  = 0x0C;
      PORTC = 0x03;
      DDRD  = 0xF0;
      PORTD = 0x00;
}
//*****************************************************************************
//TIMER0 initialize - prescale:256
// desired value: 1mSec
// actual value:   0.992mSec (0.8%)
//*****************************************************************************
void timer0_init(void)
  {
       TCCR0 = 0x00;
       TCNT0 = 0xE1;
       TCCR0 = 0x04;
  }
//*****************************************************************************
//TIMER2 initialize - prescale:8
// WGM: Normal
// desired value: 35uSec
```

```
// actual value: 34.000uSec (2.9%)
//*****************************************************************************
void timer2_init(void)
  {
      TCCR2 = 0x00;
      ASSR   = 0x00;
      TCNT2 = 0xDE;
      OCR2   = 0x22;
      TCCR2 = 0x02;
}
//*****************************************************************************
//                        定时器  TIMER0  中断
//*****************************************************************************
#pragma interrupt_handler timer0_ovf_isr:10
void timer0_ovf_isr(void)
  {
      TCNT0 = 0xE1;
      u16ReadDownStreamTimer++;
  }
//*****************************************************************************
//                                  main() function
//*****************************************************************************
void main(void)
  {
    U16   u16Flag;          // 超时计数
    unsigned char iiii;
    unsigned long u32I;     // 接收数据临时变量
    unsigned long ReceiveValue = 0;   //   接收数据包变量
    unsigned long ReceiveValue1 = 0;   // 接收校验和变量
    unsigned int ReceiveHighByte;       // 分解数据包     湿度
    unsigned int ReceiveLowByte;       //   分解数据包    温度
    WDR();
    DelayNS(2);                                        //提高稳定性的延时
    init_devices();
    timer2_init();
    u16ReadDownStreamTimer = 0;
    DDRC &= ~0X20;
    PORTC |= 0X20;
    while(1)
      {
        Display();     //显示
        WDR();
            if(u16ReadDownStreamTimer > 1000)          //1s  采集一次
              {
```

```
if(iiii == 0)
  {
    DDRC |= 0X20;
    PORTC &= ~0X20;   //发送 0
    iiii++;
    u16ReadDownStreamTimer = 991;
  }
else
  {
    iiii = 0;
    delay1N(200);          //延时等待对方响应
    DDRC &= ~0X20;   //输入
    PORTC |= 0X20;     //拉高
    delay1N(40);          //延时等待对方响应
    if(!(PINC&0X20))   //检测对方是否响应
      {
        u16Flag = 0;
        while((!(PINC&0X20)) && ((u16Flag++)<50000));
        //等待响应低电平拉高，拉高之后开始传输数据
        ReceiveValue = 0;
        ReceiveValue1 = 0;
        u16Flag = 0;
        while((PINC&0X20) && ((u16Flag++)<50000));
        // 等待数据 第一个'0'出现
        for(u32I=0x80000000; u32I>0; u32I>>=1)
        // 4 个字节的数据   湿度(8+8)+温度(8+8)=32(Bit)
          {
            u16Flag = 0;
            while((!(PINC&0X20))&& ((u16Flag++)<30));
            Timer2_Delay150ms();
            u16Flag = 0;
            while((PINC&0X20) && ((u16Flag++)<300));
            if(TIFR&0x40)
              {
                  ReceiveValue |= u32I;
              }
          }
        for(u32I=0x80; u32I>0; u32I>>=1)       // 1 BYTE 的校验字节
          {
            u16Flag = 0;
            while((!(PINC&0X20))&& ((u16Flag++)<30));
            Timer2_Delay150ms();
            u16Flag = 0;
            while((PINC&0X20) && ((u16Flag++)<300));
```

```
                    if(TIFR&0x40)
                      {
                        ReceiveValue1 |= u32I;
                      }
                  }
                DDRC   |= 0X20;   //输出
                PORTC |= 0X20;    //高上拉
                ReceiveLowByte =(unsigned int )( ReceiveValue&0x0000FFFF);
                ReceiveHighByte = ReceiveValue>>16;
                if((U8)(*((U8*)(&ReceiveHighByte)+1)+*((U8*)(&ReceiveLowByte)+1))
                  ==*(U8 *)(&ReceiveValue1))
                  {      //校验和
                    PLAY[3] = *((U8*)(&ReceiveHighByte) + 1)/10;
                    PLAY[4] = *((U8*)(&ReceiveHighByte) + 1)%10;
                    PLAY[2] = 0;
                    PLAY[0] = *((U8*)(&ReceiveLowByte) + 1)/10;
                    PLAY[1] = *((U8*)(&ReceiveLowByte) + 1)%10;
                    PLAY[5] = 0;
                  }
                else    //校验和不正确显示 6 个 8
                  {
                    PLAY[0] = 8; PLAY[1] = 8;
                    PLAY[2] = 8; PLAY[3] = 8;
                    PLAY[4] = 8; PLAY[5] = 8;
                  }
              }
            else       //没有检测到信号   发送 Err 到显示
              {
                PLAY[0] = 12;PLAY[1] = 13;PLAY[2] = 13;
                PLAY[3] = 12;PLAY[4] = 13;PLAY[5] = 13;
              }
            DDRC   |= 0X20;
            PORTC |= 0X20;
            u16ReadDownStreamTimer = 0;   //清定时计数位
          }
        }
      else    // 还没到，等待
        {
          DelayNS(1);DelayNS(1);
          DelayNS(1);DelayNS(1);
        }
    }
  }
/************************************END*************************************/
```