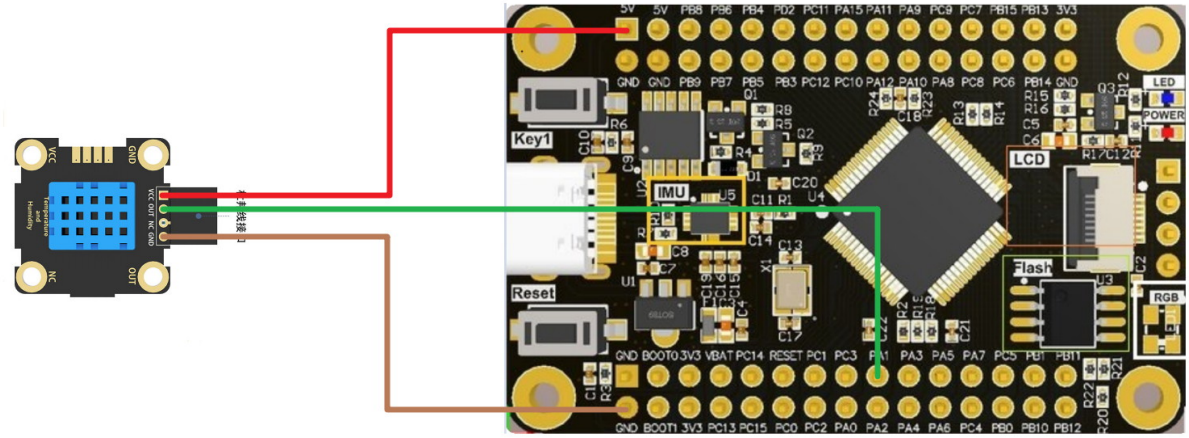
Temperature and humidity module: single-bus digital signal output

Hardware wiring



DHT11 temperature and humidity module	STM32F103RCT6
VCC	5V/3.3V
OUT	PA1
NC	
GND	GND

Brief principle

Communication and synchronization between STM32F103RCT6 and DHT11 in a single-bus data format.

A full data transfer is 40bit, high first-out.

Data format:

8bit humidity integer data 8bit humidity decimal data 8bi temperature integer data 8bit temperature decimal data 8bit checksum

When the data is transmitted correctly,

Checksum = 8-bit humidity integer data + 8-bit humidity decimal data + 8bi temperature integer data + 8-bit temperature decimal data

After the user MCU sends a start signal, DHT11 switches from low-power mode to high-speed mode, waits for the host start signal to end, DHT11 sends a response signal, sends 40bit data, and triggers a signal acquisition, and the user can choose to read part of the data.

In slave mode, DHT11 receives a start signal to trigger a temperature and humidity acquisition, if it does not receive the host to send a start signal, DHT11 will not actively collect temperature and humidity, and after collecting data, switch to low-speed mode.

Main code

main.c

```
#include "stm32f10x.h"
#include "UART.h"
#include "SysTick.h"
#include "DHT11.h"

int main(void)
{

    SysTick_Init();//滴答定时器初始化
    UART1_Init();//UART1初始化
    DHT11_Init();//DHT11温湿度传感器初始化 PA1

    printf("Temperature and humidity measurement begins!\n");

    while(1)
    {
        DHT11_Read();//读取DHT11完整信息
        printf("Temperature = %d.%d℃\n", Tem_H, Tem_L);//打印温度信息
        printf("humidity = %d.%d %%RH\n", Hum_H, Hum_L);//打印湿度信息
        Delay_us(500000);
    }
```

SysTick.c

```
#include "SysTick.h"

unsigned int Delay_Num;

void SysTick_Init(void)//滴答定时器初始化
{
    while(SysTick_Config(72));//设置重装载值 72 对应延时函数为微秒级
    //若将重装载值设置为72000 对应延时函数为毫秒级
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void Delay_us(unsigned int NCount)//微秒级延时函数
{
    Delay_Num = NCount;
    SysTick->CTRL |= (1 << 0);//开启定时器
    while(Delay_Num);
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
```

```
}

void SysTick_Handler(void)
{
    if(Delay_Num != 0)
    {
        Delay_Num--;
    }
}
```

SysTick.h

```
#ifndef __SYSTICK_H__
#define __SYSTICK_H__

#include "stm32f10x.h"

void SysTick_Init(void);//滴答定时器初始化
void Delay_us(unsigned int NCount);//微秒级延时函数

#endif
```

UART.c

```
#include "UART.h"

void UART1_Init(void)//UART1初始化
{
    USART_InitTypeDef USART_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIO clock */
    /* 使能GPIOA AFIO时钟 TXD(PA9) RXD(PA10) */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABLE);

    /* Enable USART1 Clock */
    /* 使能串口1 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 , ENABLE);

    /* Configure USART1 Rx as input floating */
    /* 配置RXD引脚 PA10 浮空输入模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USART1 Tx as alternate function push-pull */
    /* 配置TXD引脚 PA9 复用推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
```

```
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* USART1 configured as follow:
    - BaudRate = 115200 baud
    - Word Length = 8 Bits
    - One Stop Bit
    - No parity
    - Hardware flow control disabled (RTS and CTS signals)
    - Receive and transmit enabled
    */
    USART_InitStructure.USART_BaudRate = 115200;//波特率设置
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//8位数据位
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//1位停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;//无需硬件流控
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//全双工 即发送也接
受
    USART_Init(USART1, &USART_InitStructure);

    /* Enable the USART1 */
    /* 使能USART1 */
    USART_Cmd(USART1, ENABLE);
}

void USART_SendString(USART_TypeDef* USARTx, char *pt)//给指定串口发送字符串
{
    while(*pt)
    {
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
        USART_SendData(USARTx,*pt);
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
        pt++;
    }
}

int fputc(int c, FILE *pt)//printf重定向
{
    USART_TypeDef* USARTx = USART1;
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
    USART_SendData(USARTx, c);
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
    return 0;
}
```

UART.h

```
#ifndef __UART_H__
#define __UART_H__

#include "stm32f10x.h"
#include "stdio.h"

void UART1_Init(void);//UART1初始化
void USART_SendString(USART_TypeDef* USARTx, char *pt);//给指定串口发送字符串
int fputc(int c, FILE *pt);//printf重定向

#endif
```

DHT11.c

```
#include "DHT11.h"

uint8_t Hum_H, Hum_L, Tem_H, Tem_L;

void DHT11_Init(void)//DHT11 OUT 引脚初始化 PA1
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIO clock */
  /* 使能GPIOA时钟 PA1 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    /* Configure PA1 in output pushpull mode */
  /* 配置PA1 推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void DHT11_Mode_Out_PP(void)//PA1输出模式
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Configure PA1 in output pushpull mode */
  /* 配置PA1 推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void DHT11_Mode_IPU(void)//PA1输入模式
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Configure PA1 in Pull-up input mode */
```

```
  /* 配置PA1 上拉输入模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void DHT11_Start(void)//DHT11开始信号
{
    uint16_t Time;

    /* 主机拉低：PA1输出低电平信号不低于18ms */
    DHT11_Mode_Out_PP();//PA1输出模式
    DHT11_PA1_OUT(0);//PA1输出低电平
    Delay_us(20000);//拉低信号20ms

    /* 主机拉高：PA1输出高电平信号20-40ms */
    DHT11_PA1_OUT(1);//PA1输出高电平
    Delay_us(30);//拉高信号30ms


    DHT11_Mode_IPU();//PA1输入模式

    /* DHT11响应信号 */
    /*
        80us低电平信号
    */
    Time = 80;
    while((!GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1)) && (Time > 0))
    {
        Time--;
        Delay_us(1);
    }
    /*
        80us高电平信号
    */
    Time = 80;
    while(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) && (Time > 0))
    {
        Time--;
        Delay_us(1);
    }
}

uint8_t DHT11_Read_Byte(void)//读取DHT11数据
{
    uint16_t i;
    uint8_t Data = 0;
    uint16_t Time;

    DHT11_Mode_IPU();//PA1输入模式

    /* 接收一个Byte数据 */
```

```
    for(i = 0; i < 8; i++)
    {
        while(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) == 0);//等待50us低电平间隙信号发送完成

        Delay_us(40);//区分数据为是'0'还是'1'

        Data <<= 1;
        if(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1))//如果40us过后还是高电平，则数据为1;反之,数据为0
            Data += 1;

        Time = 50;
        while(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) && (Time > 0))//等待上一个数据发送完毕
        {
            Time--;
            Delay_us(1);
        }
    }

    return Data;
}

void DHT11_Read(void)//读取DHT11完整信息
{
    uint8_t Hum_H_temp, Hum_L_temp, Tem_H_temp, Tem_L_temp, Check_Sum;

    DHT11_Start();//DHT11开始信号

    Hum_H_temp = DHT11_Read_Byte();//湿度整数
    Hum_L_temp = DHT11_Read_Byte();//湿度小数
    Tem_H_temp = DHT11_Read_Byte();//温度整数
    Tem_L_temp = DHT11_Read_Byte();//温度小数
    Check_Sum  = DHT11_Read_Byte();//校验和

    DHT11_Mode_Out_PP();//PA1输出模式
    DHT11_PA1_OUT(1);//拉高电平 进入空闲状态

    if((Hum_H_temp + Hum_L_temp + Tem_H_temp + Tem_L_temp) == Check_Sum)//数据校验 获取正确数据
    {
        Hum_H = Hum_H_temp;
        Hum_L = Hum_L_temp;
        Tem_H = Tem_H_temp;
        Tem_L = Tem_L_temp;
    }
}
```

DHT11.h

```
#ifndef __DHT11_H__
#define __DHT11_H__

#include "stm32f10x.h"
#include "SysTick.h"

#define DHT11_PA1_OUT(A)    GPIO_WriteBit(GPIOA, GPIO_Pin_1, (BitAction)(A))

extern uint8_t Hum_H, Hum_L, Tem_H, Tem_L;

void DHT11_Init(void);///DHT11 OUT 引脚初始化 PA1
void DHT11_Read(void);////读取DHT11完整信息

#endif
```

Phenomenon

After downloading the program, press the Reset key once, and the downloaded program will run.

At this time, the serial port will continuously print the temperature and humidity information measured by the temperature and humidity sensor.