



STM32 MCU Development

STM32 单片机开发

-- 一线协议之 DHT11 温湿度传感器采样

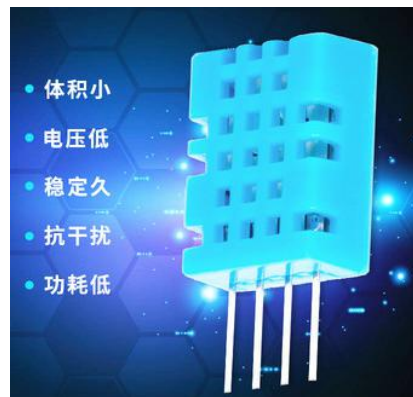
目录

1. DHT11 传感器介绍.....	3
1.1. DHT11 简介.....	3
1.2. DHT11 工作原理.....	3
1.3. DHT11 通信时序.....	4
2. DHT11 温湿度采样实现.....	5
2.1. 硬件连接.....	5
2.2. STM32CubeMX 配置.....	5
2.3. DHT11 源文件.....	6
2.4. DHT11 头文件.....	11
2.5. DHT11 测试代码.....	12
2.6. 运行测试.....	13

1. DHT11 传感器介绍

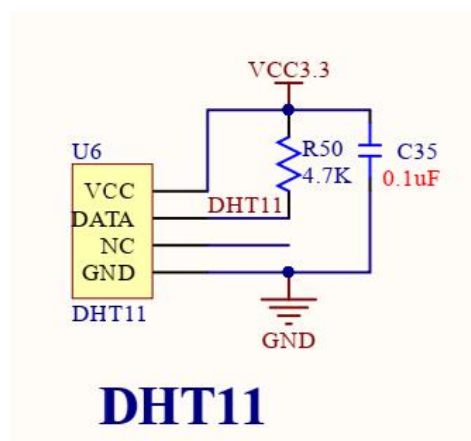
1.1. DHT11 简介

DHT11 温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器，它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性和卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比极高等优点。



1.2. DHT11 工作原理

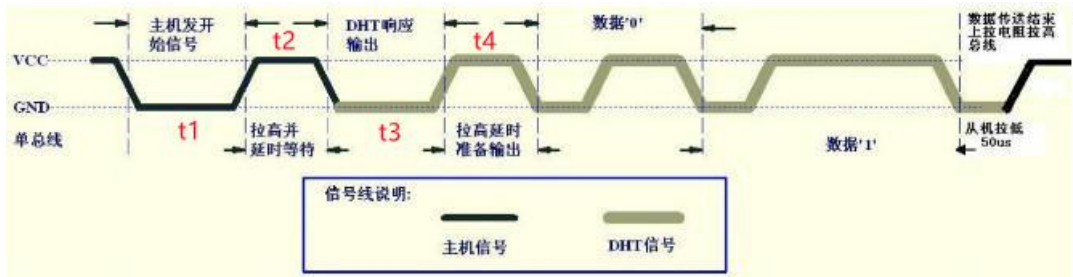
DHT11 器件采用简化的单总线通信。单总线即只有一根数据线，系统中的数据交换、控制均由单总线完成。单总线通常要求外接一个约 5.1k Ω 的上拉电阻，这样，当总线闲置时，其状态为高电平。由于它们是主从结构，只有主机呼叫从机时，从机才能应答，因此主机访问器件都必须严格遵循单总线序列，如果出现序列混乱，器件将不响应主机。



1.3. DHT11 通信时序

数据总时序

用户主机（STM32 单片机）发送一次开始信号后，DHT11 从低功耗模式转换到高速模式，待主机开始信号结束后，DHT11 发送响应信号，并送出 5 个字节的采样数据，之后结束本次采集任务。



主机发送起始信号

首先单片机将连接 DHT11 DATA 引脚的 GPIO 口输出低电平，且低电平保持时间不能小于 18ms (t_1)，然后拉高数据线 20~40 μs (t_2)，等待读取 DHT11 的响应信号。

检测从机应答信号

DHT11 的 DATA 引脚检测到外部信号有低电平(t_1)，并等待外部低电平信号结束(t_2)，之后 DHT11 开始输出 80 μs (t_3) 的低电平作为应答信号，紧接着输出 80 μs (t_4) 的高电平通知主机准备接收数据。

数据传输

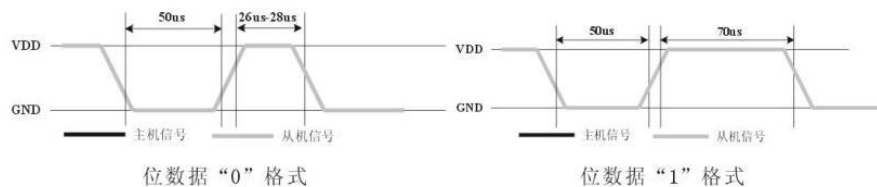
DHT11 在传输数据时，一次传输 4 字节温湿度值数据和 1 字节数据校验。其数据格式为：

1B 湿度整数数据 + 1B 湿度小数数据 + 1B 温度整数数据 + 1B 温度小数数据 + 1B 校验位。

在发送每个字节的 8 个位时，采用高位优先方式（MSB），其中对于数据位 0/1 的电平定义如下：

数据位“0”： 50 微秒的低电平加 26-28 微秒的高电平；

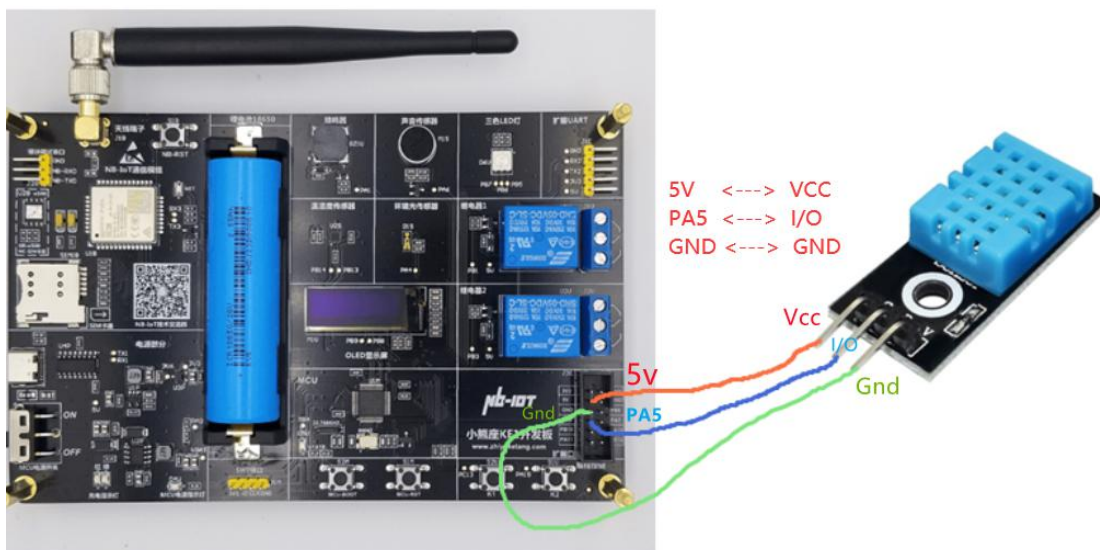
数据位“1”： 50 微秒的低电平加 70 微秒的高电平；



单片机在处理数据接收时可以先等待低电平过去，即等待数据线拉高，再延时 40 μs （因为 40 μs 大于 28 μs 且小于 70 μs ），再检测此时数据线是否为高，如果为高，则数据判定为 1，否则为 0。

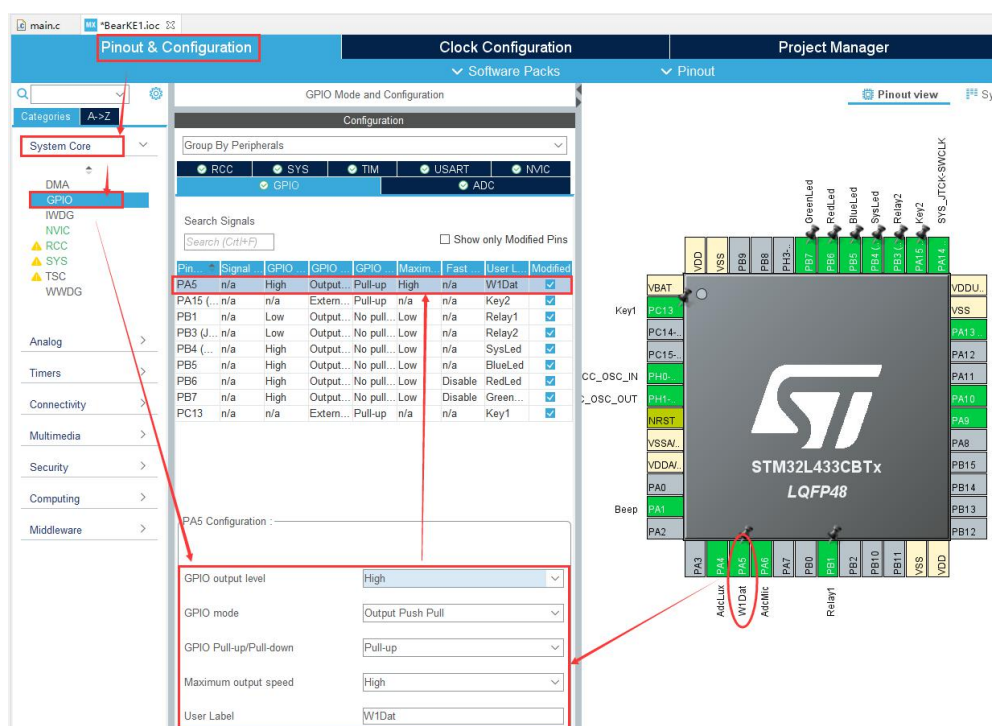
2. DHT11 温湿度采样实现

2.1. 硬件连接



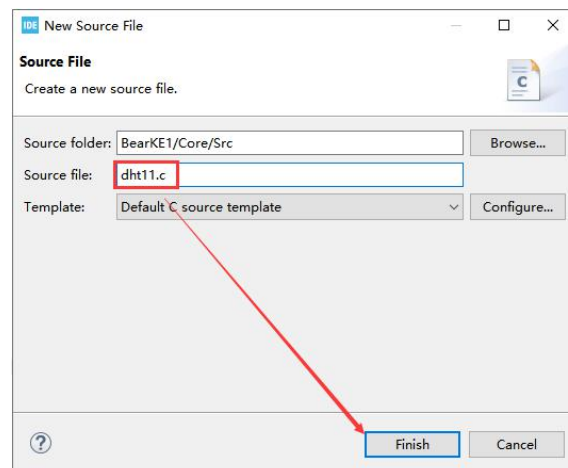
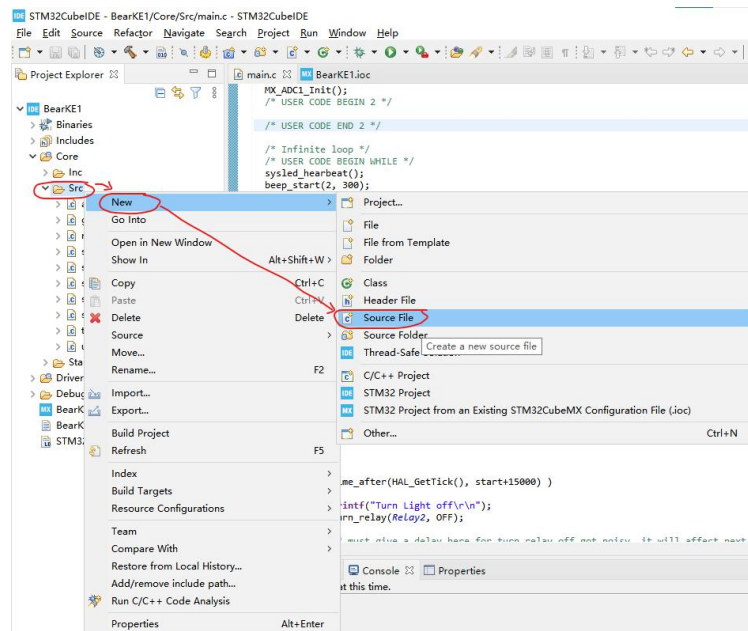
2.2. STM32CubeMX 配置

配置 DHT11 连接的 GPIO 管脚(PA5)，默认设置为 GPIO 输出模式，配置使能内部上拉电路并设置翻转速率为高速。配置好后，任意位置按 Ctrl+S 键自动生成代码。



2.3. DHT11 源文件

2.3.1. 创建 DHT11 温湿度传感器的驱动源文件 dht11.c



2.3.2. 编写 DHT11 的驱动源文件 dht11.c

```
#include "tim.h"
#include "gpio.h"
#include "main.h"

#define DHT11_In_init() \
{ \
    GPIO_InitTypeDef GPIO_InitStructure = {0}; \
    GPIO_InitStructure.Pin = W1Dat_Pin; \
    GPIO_InitStructure.Mode = GPIO_MODE_INPUT; \
    GPIO_InitStructure.Pull = GPIO_PULLUP; \
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH; \
    HAL_GPIO_Init(W1Dat_GPIO_Port, &GPIO_InitStructure); \
}

#define DHT11_Out_init() \
{ \
    GPIO_InitTypeDef GPIO_InitStructure = {0}; \
    GPIO_InitStructure.Pin = W1Dat_Pin; \
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP; \
    GPIO_InitStructure.Pull = GPIO_NOPULL; \
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH; \
    HAL_GPIO_Init(W1Dat_GPIO_Port, &GPIO_InitStructure); \
}

#define DHT11_DQ_OUT(N) HAL_GPIO_WritePin(W1Dat_GPIO_Port, \
                                             W1Dat_Pin, N==1?GPIO_PIN_SET:GPIO_PIN_RESET)
#define DHT11_DQ_IN()  HAL_GPIO_ReadPin(W1Dat_GPIO_Port, W1Dat_Pin)

/* 主机发送起始信号 */
static void DHT11_StartSignal(void)
{
    DHT11_Out_init();

    /* 主机拉低 >= 18ms */
    DHT11_DQ_OUT(0);
    HAL_Delay(20);

    /* 主机拉高 >= 20~40us */
    DHT11_DQ_OUT(1);
    delay_us(30);
}
```

```
        DHT11_In_init();
    }

uint8_t DHT11_RespondSignal(void)
{
    uint8_t retry = 0;

    /* 总线变为低电平说明从设备发送了响应信号：80us */
    while( DHT11_DQ_IN() && retry <100)
    {
        retry++;
        delay_us(1);
    }

    /* 超时没有收到响应信号 */
    if(retry >= 100)
        return 1;

    /* 从设备再把总线拉高表示从设备要发送数据了：80us */
    retry = 0;
    while( !DHT11_DQ_IN() && retry <100)
    {
        retry++;
        delay_us(1);
    }

    /* 超时没有收到数据开始信号 */
    if(retry >= 100)
        return 1;

    return 0;
}

uint8_t DHT11_ReadBit(void) //读取一个位
{
    uint8_t retry = 0;

    /* 从设备回复的每个位数据以低电平标置开始：50us */
    while( DHT11_DQ_IN() && retry<100 )
    {
        retry++;
        delay_us(1);
    }
}
```



```
/* 数据位都用高电平表示，但高电平的长短决定了数据是 1 or 0 */
retry = 0;
while( !DHT11_DQ_IN() && retry<100 )
{
    retry++;
    delay_us(1);
}

/* 判断数据位是 1(70us) or 0(26~28us)*/
delay_us(40);
if( DHT11_DQ_IN() )
    return 1;
else
    return 0;
}

uint8_t DHT11_ReadByte(void)    //读取一个字节返回值位采集值
{
    uint8_t    i,dat;

    dat = 0;
    for(i=0; i<8; i++)
    {
        dat <= 1;
        dat |= DHT11_ReadBit();    //每读取到一个位放到最后一位
    }

    return dat;
}

int DHT11_SampleData(float *temperature, float *humidity)
{
    uint8_t    humi_H8bit;
    uint8_t    humi_L8bit;
    uint8_t    temp_H8bit;
    uint8_t    temp_L8bit;
    uint8_t    check_sum;

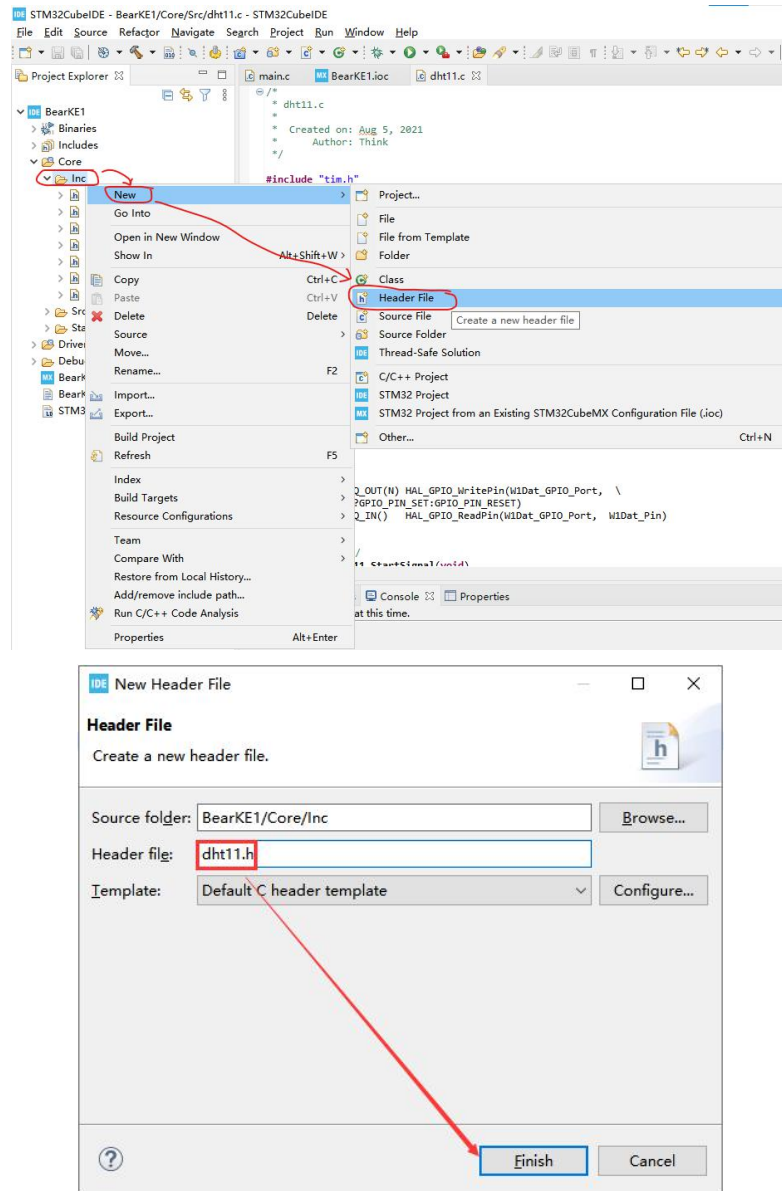
    if( !temperature || !humidity )
        return -1;

    /* 主机发起起始信号并等到从设备的响应信号 */
    DHT11_StartSignal();
```

```
if( 0 != DHT11_RespondSignal() )  
    return -2;  
  
humi_H8bit = DHT11_ReadByte();  
humi_L8bit = DHT11_ReadByte();  
temp_H8bit = DHT11_ReadByte();  
temp_L8bit = DHT11_ReadByte();  
check_sum = DHT11_ReadByte();  
  
if( (humi_H8bit+humi_L8bit+temp_H8bit+temp_L8bit) != check_sum )  
    return -3;  
  
*humidity = (humi_H8bit*100 + humi_L8bit) / 100.00;  
*temperature = (temp_H8bit*100 + temp_L8bit) / 100.00;  
  
return 0;  
}
```

2.4. DHT11 头文件

2.4.1. 创建 DHT11 温湿度传感器的驱动头文件 dht11.h



2.4.2. 编写 DHT11 的驱动源文件 dht11.c

```
#ifndef INC_DHT11_H_
#define INC_DHT11_H_

extern int DHT11_SampleData(float *temperature, float *humidity);

#endif /* INC_DHT11_H_ */
```

2.5. DHT11 测试代码

修改 main.c 文件，实现每隔 3s 采样当前的温湿度值并打印输出

```
... ..

在文件开始处添加用户头文件 dht11.h :
/* USER CODE BEGIN Includes */
#include "dht11.h"
/* USER CODE END Includes */
... ..

在 main 开始处相应位置添加两个用来保存采样温度和相对湿度值的变量：
int main(void)
{
    /* USER CODE BEGIN 1 */
    float      temperature, humidity;
    /* USER CODE END 1 */

    ... ..

    /* USER CODE BEGIN WHILE */
    sysled_hearbeat();
    beep_start(2, 300);
    printf("Start BearKE1 5G NB-IoT Board Example Program v1.0\r\n");
    while (1)
    {
        if( DHT11_SampleData(&temperature, &humidity) < 0 )
        {
            printf("ERROR: DHT11 Sample Data failure\r\n");
        }
        else
        {
            printf("DHT11 Sample Temperature: %.3f   Relative Humidity: %.3f\r\n", temperature,
humidity);
        }

        HAL_Delay(3000);
    }
    ... ..
}
```

2.6. 运行测试

重新编译并烧录运行程序，使用串口调试助手监控串口输出，这时候会发现每隔 3s 单片机就会打印输出当前的温湿度值。对着温湿度传感器哈一下气，我们会发现温湿度传感器会有明显变化。

