STM32 MCU Development

STM32 单片机开发

-- I2C 协议之 SHT30 温湿度采样

CopyRight(C) 2021 凌云物网智科实验室

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

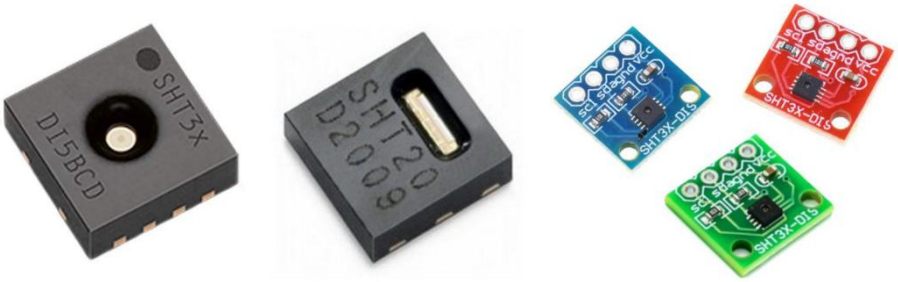
目录

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

1. SHT30 传感器介绍

1.1. SHT30 简介

SHT30 数字温湿度传感器采用业内知名的瑞士 Sensirion 公司推出的新一代 SHT30 温湿度传感器芯片，它能够提供极高的可靠性和出色的长期稳定性，具有功耗低、反应快、抗干扰能力强等优点。IIC 通讯，兼容 3.3V/5V，可以非常容易的集成到智能楼宇、天气站、仓库存储、养殖、孵化等应用场景中，其中小米的温湿度传感器使用的也是 SHT30。



SHT30 在 SHT3x 系列中属于入门版，相比上一代精度更高。传感器在 10%RH~90%RH（25℃时）误差仅为±2%RH，传感器在 0℃-65℃（典型值）误差仅为±0.2℃。其特性为：

- 高精度，内部自动校准，数字输出
- 低功耗、响应速度快、抗干扰能力强
- 兼容 3.3V/5V 控制器

1.2. SHT30 工作原理

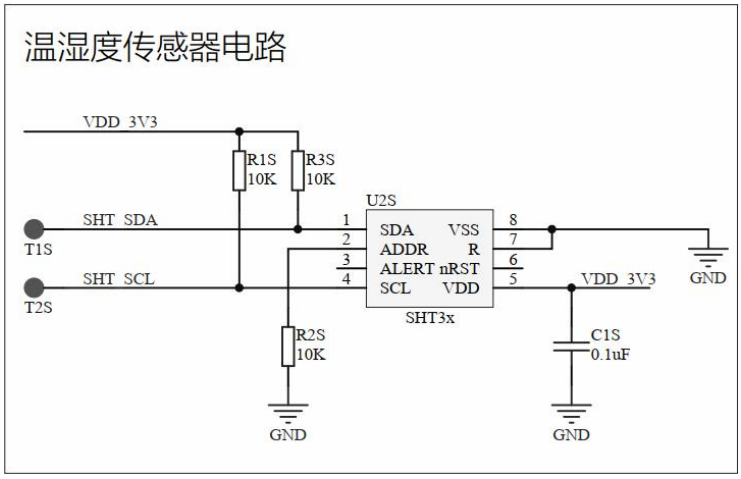
SHT30 芯片有八个引脚，利用 I2C 进行数据传输，具有两个可选地址，宽电源电压从 2.4V 到 5.5V。下面是引脚说明：

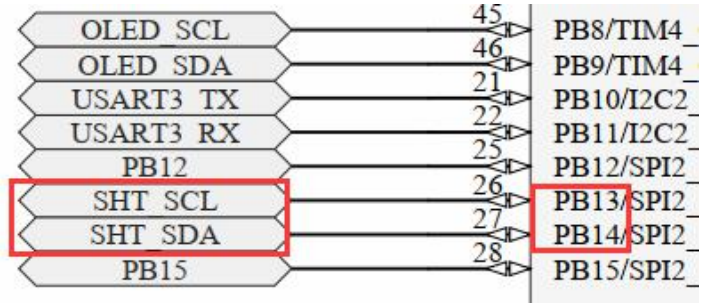
管脚	名称	备注
1	SDA	I2C 数据引脚，输入/输出
2	ADDR	地址引脚，输入
3	ALENT	报警引脚，输出；不使用时悬空
4	SCL	I2C 时钟引脚，输入/输出
5	VDD	电源引脚，输入
6	nRESET	复位引脚，低电平有效，输入
7	R	无用引脚，接地
8	VSS	接地引脚

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

下面是小熊座 NB-IoT 开发板上 SHT30 传感器的原理图，其中 I2C 接口连到了 MCU 的 PB13 和 PB14 两个引脚上，这两个引脚可以设置为 I2C2 模式工作。

从原理图上可以看到 ADDR 线拉到了 GND 上，这样 SHT30 传感器的从设备地址为

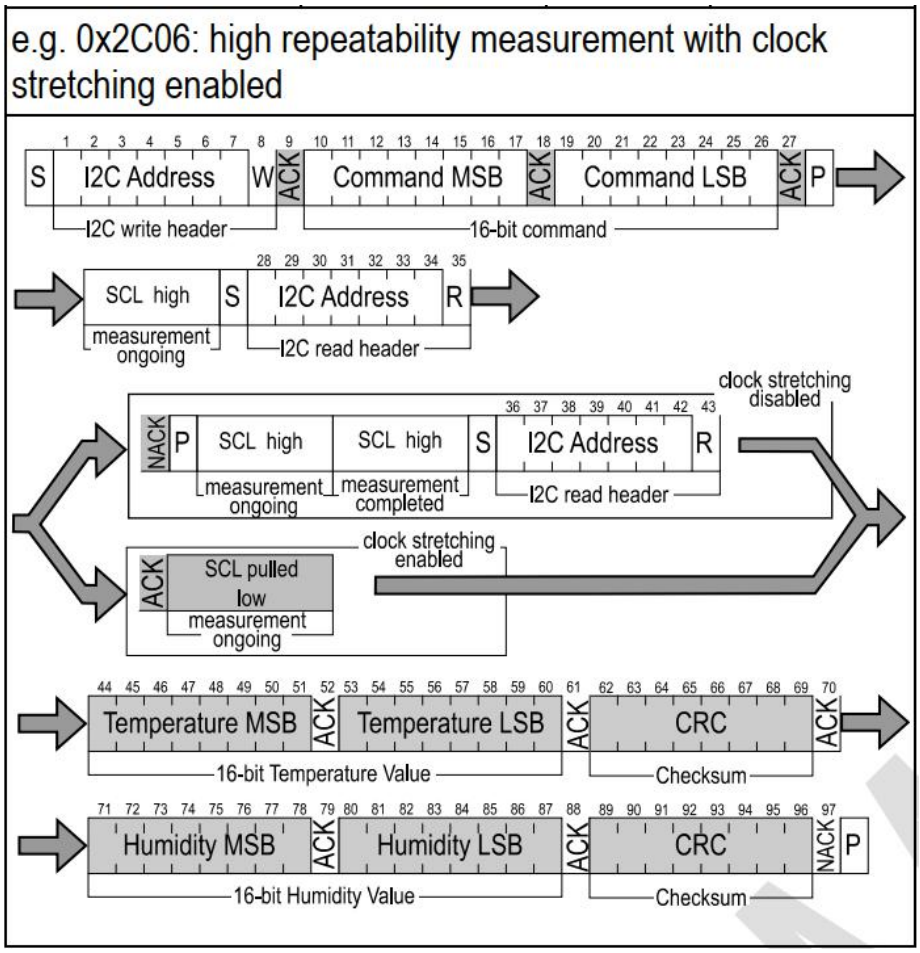




另外，SHT30 可以通过改变 ADDR 引脚的电压实现传感器的地址改变。默认地址是 ADDR 连接 VSS 时地址为 0x44，如果拉成高电平则可以修改为 0x45。从原理图上可以看到 ADDR 接地，则 SHT30 的 I2C 从设备地址为 0x44。

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

1.3. SHT30 通信时序

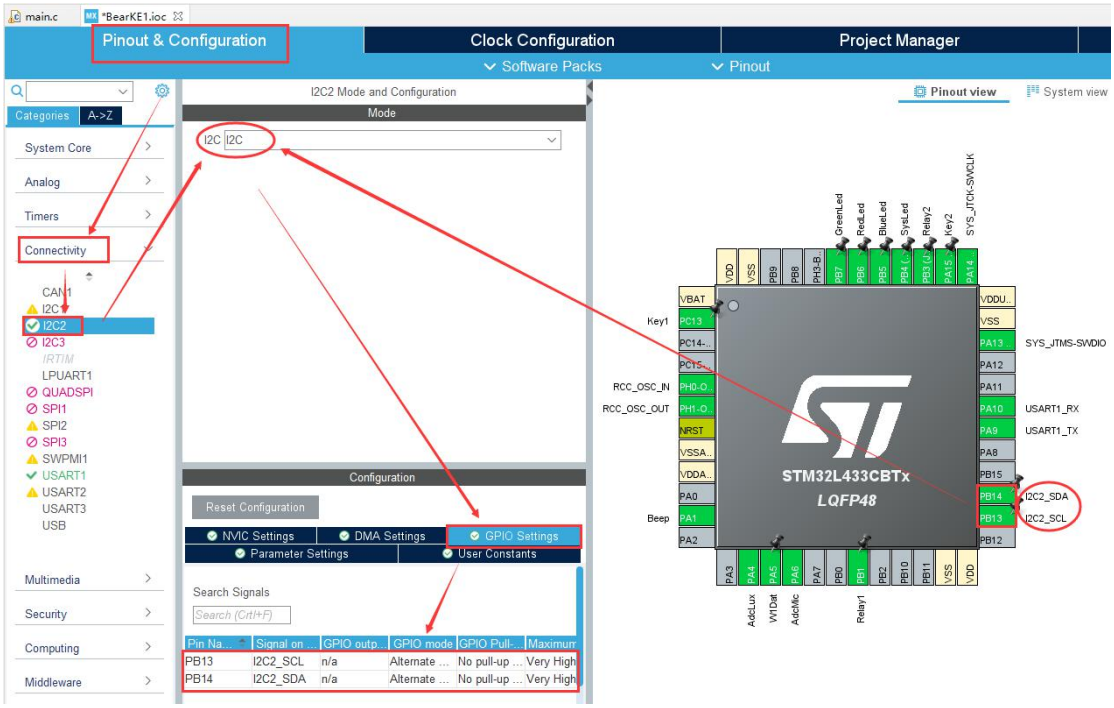


版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

2. 基于 HAL 库 I2C 采样实现

2.1. STM32CubeMX 配置

先配置 SHT30 连接的 I2C 管脚 PB13 和 PB14 为 I2C 模式，此时因为 I2C 功能并没有使能，管脚状态为黄色。接下来再在 Connectivity 里选择 I2C2 并配置为 I2C 模式，这是 I2C 功能配置完成。配置好之后按 Ctrl+S 将会自动生成 I2C 总线初始化代码。



版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

2.2. SHT30 源文件

在 Core/Src 下创建并编写 SHT30 温湿度传感器的驱动源文件 sht30.c

```
#include <stdio.h>
#include "stm32l4xx_hal.h"
#include "i2c.h"
#include "sht30.h"

//#define CONFIG_SHT30_DEBUG

#ifdef CONFIG_SHT30_DEBUG
#define sht30_print(format,args...) printf(format, ##args)
#else
#define sht30_print(format,args...) do{} while(0)
#endif

static int sht30_send_cmd(SHT30_CMD cmd)
{
    uint8_t buf[2];

    buf[0] = cmd >> 8;
    buf[1] = cmd & 0xFF;

    return HAL_I2C_Master_Transmit(&hi2c2, SHT30_ADDR_WR, (uint8_t*)buf, 2, 0xFFFF);
}

static void sht30_soft_reset(void)
{
     sht30_send_cmd(SOFT_RESET_CMD);
     HAL_Delay(1);
}

static int sht30_single_shot_measurement(uint8_t *buf, uint8_t buf_size)
{
         uint16_t     cmd = HIGH_ENABLED_CMD;  /* High with clock stretching */
         uint8_t      rv;

         if( !buf || buf_size<SHT30_DATA_SIZE )
         {
           sht30_print("%s(): Invalid input arguments\n", __func__);
           return -1;
         }
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
        rv = sht30_send_cmd(cmd);
        if( rv )
        {
          sht30_print("ERROR: SHT30 send measurement command failure, rv=%d\n", rv);
          sht30_soft_reset();
          return -2;
        }

        rv = HAL_I2C_Master_Receive(&hi2c2, SHT30_ADDR_RD, buf, SHT30_DATA_SIZE, 0xFFFF);
        if(rv)
        {
          sht30_print("ERROR: SHT30 read measurement result failure, rv=%d\n", rv);
          return -3;
        }

        return 0;
}


static uint8_t sht30_crc8(const uint8_t *data, int len)
{
    const uint8_t     POLYNOMIAL = 0x31;   /* SHT30 CRC8 Polynomial:  0x31=x8 + x5 + x4 + 1  */
    uint8_t           crc = 0xFF;         /* SHT30 CRC8 Initialization: 0xFF */
    int               i, j;

    for (i=0; i<len; ++i)
    {
        crc ^= *data++;

        for (j=0; j<8; ++j)
        {
          crc = ( crc & 0x80 )? (crc << 1) ^ POLYNOMIAL: (crc << 1);
        }
    }

    return crc;
}

int SHT30_SampleData(float *temperature, float *humidity)
{
        uint8_t            buf[SHT30_DATA_SIZE];
        int                rv;
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
        uint16_t          temp;
        uint16_t          humd;
        uint8_t           crc;

        if(!temperature || !humidity)
        {
           sht30_print("%s(): Invalid input arguments\n", __func__);
           return -1;
        }

        rv = sht30_single_shot_measurement(buf, SHT30_DATA_SIZE);
        if( rv )
        {
           sht30_print("SHT30 Single Short measurement failure, rv=%d\n", rv);
           return -2;
        }

#ifdef CONFIG_SHT30_DEBUG
        {
           int         i;

           sht30_print("SHT30 get %d bytes sample data: \n", SHT30_DATA_SIZE);
           for(i=0; i<SHT30_DATA_SIZE; i++)
           {
              sht30_print("0x%02x ", buf[i]);
           }
           sht30_print("\n");
        }
#endif

        /* byte[0-1] is temperature value, and byte[2] is temperature CRC */
        crc = sht30_crc8(buf, 2);
        sht30_print("SHT30 temperature Cal_CRC: [%02x] EXP_CRC: [%02x]\n", crc, buf[2]);
        if( crc != buf[2])
        {
           sht30_print("SHT30 measurement temperature got CRC error\n");
           return -3;
        }


        /* byte[3-4] is humidity value, and byte[5] is humidity CRC */
        crc = sht30_crc8(&buf[3], 2);
        sht30_print("SHT30 humidity Cal_CRC: [%02x] EXP_CRC: [%02x]\n", crc, buf[5]);
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
        if( crc != buf[5])
        {
           sht30_print("SHT30 measurement temperature got CRC error\n");
           return -4;
        }

        temp = (buf[0]<<8) | buf[1];
        humd = (buf[3]<<8) | buf[4];

        *temperature = -45 + 175*((float)temp/65535);
        *humidity = 100 * ((float)humd / 65535);

        return 0;
}
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

2.3. SHT30 头文件

在 Core/Irc 下创建并编写 SHT30 的驱动头文件 sht30.h：

```
#ifndef INC_SHT30_H_
#define INC_SHT30_H_

#include "stm32l4xx_hal.h"

/* chip 7-bits hardware address */
#define SHT30_ADDR        0x44    /* ADDR connected to GND */

/* I2C protocol communication address */
#define SHT30_ADDR_WR     (SHT30_ADDR<<1)          /* address bit[0]=0 is write */
#define SHT30_ADDR_RD     ((SHT30_ADDR<<1) | 0x01)   /* address bit[0]=1 is read  */

#define SHT30_DATA_SIZE    6  /* 2B temperature + 1B CRC, 2B humidity + 1B CRC  */

typedef enum
{
    /* Soft reset command  */
    SOFT_RESET_CMD = 0x30A2,

    /* Measurement Commands for Single Shot Data Acquisition Mode:
     * Repeatability (low,medium and high) and clock stretching (enabled ordisabled)
     */
    HIGH_ENABLED_CMD    = 0x2C06,
    MEDIUM_ENABLED_CMD  = 0x2C0D,
    LOW_ENABLED_CMD     = 0x2C10,
    HIGH_DISABLED_CMD   = 0x2400,
    MEDIUM_DISABLED_CMD = 0x240B,
    LOW_DISABLED_CMD    = 0x2416,

    /*  Measurement Commands for Periodic Data Acquisition Mode
     *  Data acquisition frequency: 0.5, 1, 2, 4 & 10 measurements per second, mps
     */
    HIGH_0_5_CMD   = 0x2032,
    MEDIUM_0_5_CMD = 0x2024,
    LOW_0_5_CMD    = 0x202F,
    HIGH_1_CMD     = 0x2130,
    MEDIUM_1_CMD   = 0x2126,
    LOW_1_CMD      = 0x212D,
    HIGH_2_CMD     = 0x2236,
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
    MEDIUM_2_CMD   = 0x2220,
    LOW_2_CMD      = 0x222B,
    HIGH_4_CMD     = 0x2334,
    MEDIUM_4_CMD   = 0x2322,
    LOW_4_CMD      = 0x2329,
    HIGH_10_CMD    = 0x2737,
    MEDIUM_10_CMD  = 0x2721,
    LOW_10_CMD     = 0x272A,

        /* Readout of Measurement Results for Periodic Mode */
        READOUT_FOR_PERIODIC_MODE = 0xE000,
} SHT30_CMD;


extern int SHT30_SampleData(float *temperature, float *humidity);

#endif /* INC_SHT30_H_ */
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

2.4. SHT30 测试代码

修改 main.c 文件，将之前的 DHT11 温湿度传感器采样函数改成 SHT30 采样函数，实现每隔 3s 采样当前的温湿度值并采用 JSON 格式上报。

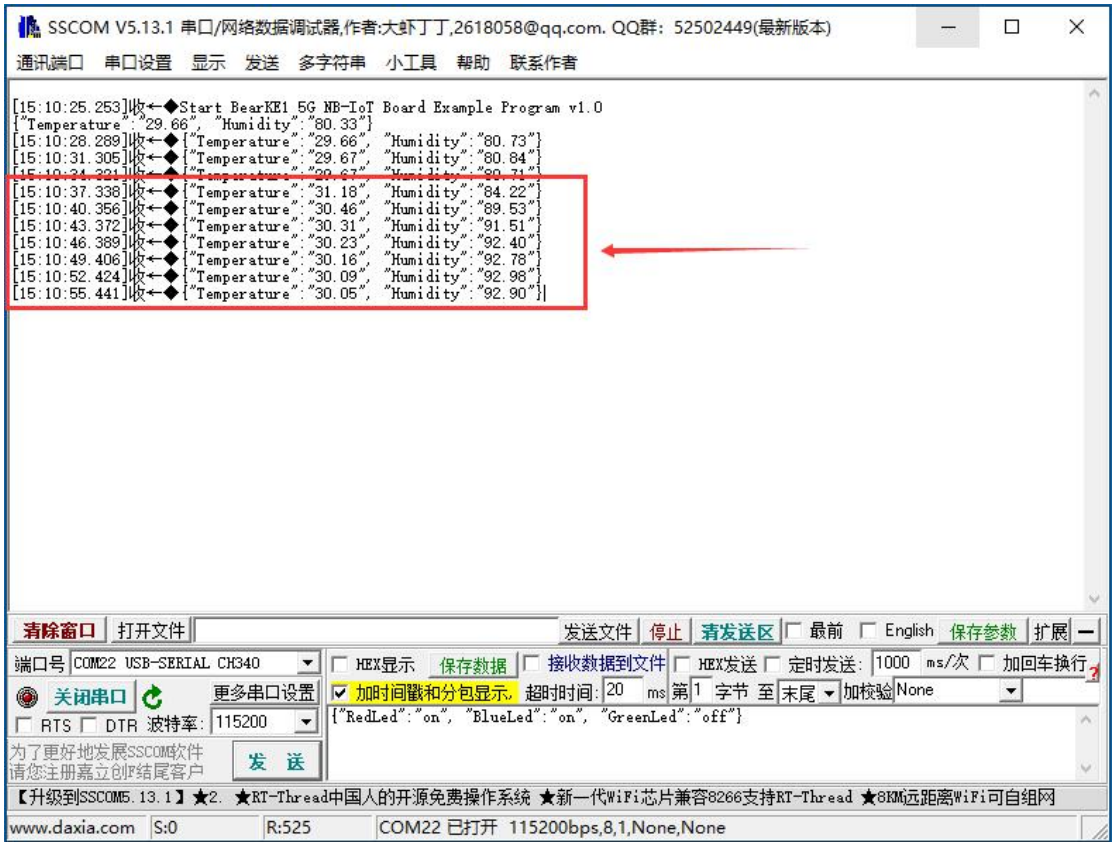
```
... ...

在文件开始处添加头文件 sht30.h ：
/* USER CODE BEGIN Includes */
#include <string.h>
#include "dht11.h"
#include "sht30.h"
#include "core_json.h"
/* USER CODE END Includes */

... ...

int main(void)
{
  ... ...

  /* 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)
  {
      report_tempRH_json();
      HAL_Delay(3000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */  }
 ... ...
}

... ...

修改 report_tempRH_json()函数，将 DHT11 采样改成 SHT30 温湿度传感器采样 ：
int report_tempRH_json(void)
{
    char         buf[128];
    float        temperature, humidity;
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
    if ( SHT30_SampleData(&temperature, &humidity) < 0 )
    {
        printf("ERROR: SHT30 Sample data failure\n");
        return -1;
    }

    memset(buf, 0, sizeof(buf));
    snprintf(buf, sizeof(buf), "{\"Temperature\":\"%.2f\", \"Humidity\":\"%.2f\"}", temperature,
humidity);

    HAL_UART_Transmit(&huart1 , (uint8_t *)buf, strlen(buf), 0xFFFF);

    return 0;
}
```

2.5. 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，我们也会发现温湿度传感器会有明显变化。



版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

3. 基于 GPIO 模拟 I2C 采样实现

ST 公司的单片机(包括 STM8、STM32 系列) I2C 接口普遍存在总线死锁在 BUSY 状态无法恢复的现象，有些时候 MCU 复位也无法恢复，只有断电才行。产生 I2C 总线故障的方法简单而粗暴：在 I2C 总线工作过程中，用镊子把 SCL 和 SDA 两个信号短路一下，很容易进入 BUSY 死锁状态，长时间短路也可能产生超时。HAL_I2C_Init()、HAL_I2C_Master_Transmit()、HAL_I2C_Master_Receive()等函数返回值分别为 HAL_BUSY（0x02）、HAL_TIMEOUT（0x03）。

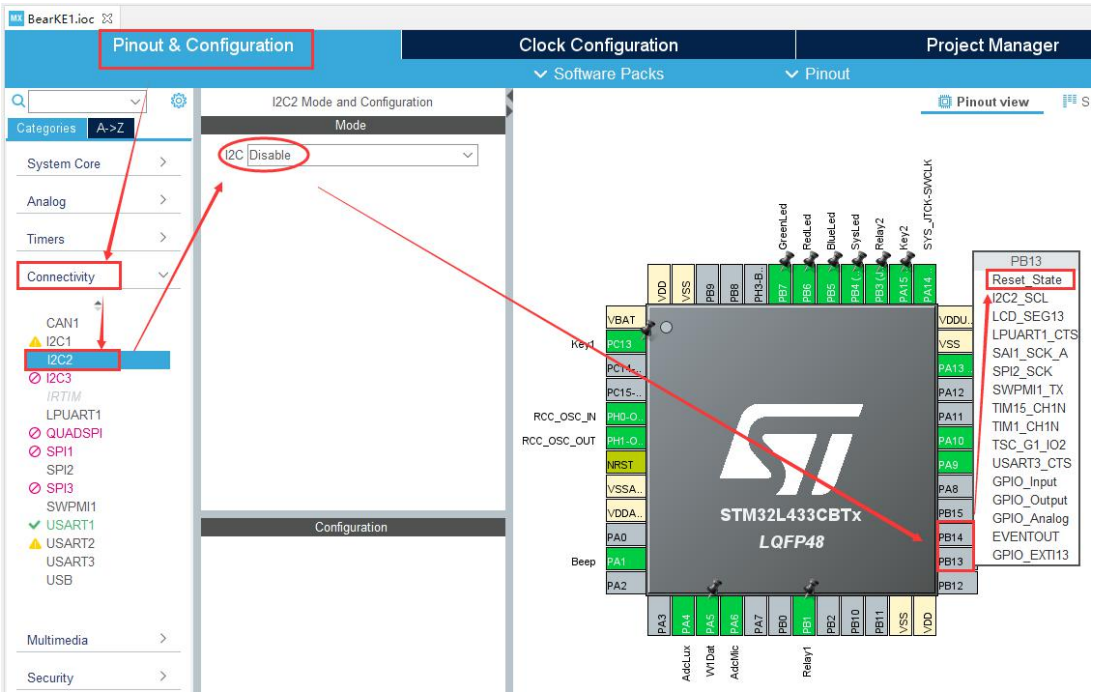
I2C 总线锁死时，用万用表测试 I2C 信号电压，SCL、SDA 均为低电平。如果调用函数：HAL_I2C_DeInit(&hi2c1)，会函数释放 IO 口回到 GPIO 的默认状态（Input），此时再测 SCL、SDA 电压，均为高电平。这说明总线是被 MCU 这边的 Master 拉低的，而不是被 Slave 拉低的，当然也存在 Slave 刚好输出低电平拉低 SDA 的可能。

总之，ST 公司的单片机我们在做项目应用时通常都不会直接使用硬件 I2C 控制器实现，而是通过软件编程使用 GPIO 模拟 I2C 总线时序来实现。另外，一般 CPU 上的 I2C 总线个数有限制，很多引脚容易冲突导致没有 I2C 接口使用，此时我们也可以直接拿两个 GPIO 口来模拟 I2C 总线。

本章我们将讲解如何通过任意两个 GPIO 管脚来模拟实现 I2C 接口，并实现 SHT30 温湿度传感器采样。

3.1. STM32CubeMX 配置

首先取消 I2C 控制器功能模式，然后配置 SHT30 连接的 I2C 管脚 PB13 和 PB14 为复位状态，接下来我们将会在 GPIO 模拟 I2C 的代码中控制它们。配置好之后按 Ctrl+S 将会自动删除 I2C 总线初始化代码。



版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

3.2. GPIO 模拟 I2C 源文件

在 Core/Src 下创建并编写 GPIO 模拟 I2C 源文件 gpio_i2c_sht30.c

```
/*****************************************************************************
*   Copyright: (C)2021 LingYun IoT System Studio <www.weike-iot.com>
*      Author: GuoWenxue<guowenxue@gmail.com> QQ: 281143292
* Description: BearKE1 NB-IoT Board GPIO simulate I2C bus source code
*
*      Notice: Must implement delay_us() by timer in tim.h first.
*
*   ChangeLog:
*       Version    Date       Author           Description
*       V1.0.0  2021.08.10  GuoWenxue      Release initial version
*****************************************************************************/

#include <stdio.h>
#include "stm32l4xx_hal.h"
#include "tim.h"   /* delay_us() implement */
#include "gpio.h"
#include "gpio_i2c_sht30.h"

/* comment follow macro will disable I2C bus clock stretching support
 * reference: http://www.i2c-bus.org/clock-stretching/
 */
#define I2C_CLK_STRETCH_TIMEOUT    50

/* uncomment follow macro will enable debug print */
//#define CONFIG_GPIO_I2C_DEBUG
#ifdef CONFIG_GPIO_I2C_DEBUG
#define i2c_print(format,args...) printf(format, ##args)
#else
#define i2c_print(format,args...) do{} while(0)
#endif

/* GPIO Simulate I2C Bus pins */
typedef struct i2c_gpio_s
{
        GPIO_TypeDef       *group;
        uint16_t            scl; /* SCL */
        uint16_t            sda; /* SDA */
} i2c_gpio_t;
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
static i2c_gpio_t   i2c_pins = { GPIOB, GPIO_PIN_13/*SCL*/, GPIO_PIN_14/*SDA*/ };

#define SDA_IN()   do{  GPIO_InitTypeDef GPIO_InitStruct = {0}; \
                            GPIO_InitStruct.Pin = i2c_pins.sda; \
                            GPIO_InitStruct.Mode = GPIO_MODE_INPUT; \
                            GPIO_InitStruct.Pull = GPIO_PULLUP; \
                            GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH; \
                            HAL_GPIO_Init(i2c_pins.group, &GPIO_InitStruct); \
                        }while(0)

#define SDA_OUT()  do{  GPIO_InitTypeDef GPIO_InitStruct = {0}; \
                            GPIO_InitStruct.Pin = i2c_pins.sda; \
                            GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; \
                            GPIO_InitStruct.Pull = GPIO_PULLUP; \
                            GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH; \
                            HAL_GPIO_Init(i2c_pins.group, &GPIO_InitStruct); \
                        }while(0)

#define SCL_OUT()  do{  GPIO_InitTypeDef GPIO_InitStruct = {0}; \
                            GPIO_InitStruct.Pin = i2c_pins.scl; \
                            GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP; \
                            GPIO_InitStruct.Pull = GPIO_PULLUP; \
                            GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH; \
                            HAL_GPIO_Init(i2c_pins.group, &GPIO_InitStruct); \
                        }while(0)

#define SCL_H()         HAL_GPIO_WritePin(i2c_pins.group, i2c_pins.scl, GPIO_PIN_SET)
#define SCL_L()         HAL_GPIO_WritePin(i2c_pins.group, i2c_pins.scl, GPIO_PIN_RESET)
#define SDA_H()         HAL_GPIO_WritePin(i2c_pins.group, i2c_pins.sda, GPIO_PIN_SET)
#define SDA_L()         HAL_GPIO_WritePin(i2c_pins.group, i2c_pins.sda, GPIO_PIN_RESET)

#define READ_SDA()      HAL_GPIO_ReadPin(i2c_pins.group, i2c_pins.sda)
#define READ_SCL()      HAL_GPIO_ReadPin(i2c_pins.group, i2c_pins.scl)

static inline uint8_t I2c_WaitWhileClockStretching(uint16_t timeout)
{
        while( timeout-- >  0 )
        {
             if( READ_SCL() )
                  break;
             delay_us(1);
        }
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
        return timeout ? NO_ERROR : BUS_ERROR;
}

/* StartCondition(S) */
uint8_t I2c_StartCondition()
{
        uint8_t          rv = NO_ERROR;

        SDA_OUT();
        SCL_OUT();

        /* StartCondition(S): A high to low transition on the SDA line while SCL is high.
          _______
    SCL:         |___
          _____
    SDA:       |_____
        */
        SDA_H();
        delay_us(1);
        SCL_H();
        delay_us(1);

#ifdef I2C_CLK_STRETCH_TIMEOUT
        rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
        if( rv  )
        {
         i2c_print("ERROR: %s() I2C bus busy\n", __func__);
         return rv;
        }
#endif

        SDA_L();
        delay_us(2);

        SCL_L();
        delay_us(2);

        return rv;
}
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
/* StopCondition(P) */
uint8_t I2c_StopCondition(void)
{
        uint8_t          rv = NO_ERROR;

        SDA_OUT();

        /* StopCondition(P): A low to high transition on the SDA line while SCL is high.
                ______
     SCL:   ___|
                 ______
     SDA:   _______|
        */
        SCL_L();
        SDA_L();
        delay_us(2);

        SCL_H();
        delay_us(2);

#ifdef I2C_CLK_STRETCH_TIMEOUT
        rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
        if( rv  )
        {
         i2c_print("ERROR: %s() I2C bus busy\n", __func__);
        }
#endif

        SDA_H();
        delay_us(2);

        return rv;
}

uint8_t I2c_WriteByte(uint8_t byte)
{
        uint8_t rv = NO_ERROR;
        uint8_t mask;

        /* Data line changes must happened when SCL is low */
        SDA_OUT();
        SCL_L();
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
        /* 1Byte=8bit, MSB send: bit[7]-->bit[0] */
        for(mask=0x80; mask>0; mask>>=1)
        {
           if((mask & byte) == 0)
              SDA_L();
           else
              SDA_H();

           delay_us(1);    // data set-up time (t_SU;DAT)

           SCL_H();
           delay_us(5);    // SCL high time (t_HIGH)

#ifdef I2C_CLK_STRETCH_TIMEOUT
           rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
           if( rv  )
           {
               i2c_print("ERROR: %s() I2C bus busy\n", __func__);
               goto OUT;
           }
#endif

           SCL_L();
           delay_us(1);    // data hold time(t_HD;DAT)
        }

        /* clk #9 wait ACK/NAK from slave */
        SDA_IN();
        SCL_H();    // clk #9 for ack
        delay_us(1); // data set-up time (t_SU;DAT)

#ifdef I2C_CLK_STRETCH_TIMEOUT
        rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
        if( rv  )
        {
         i2c_print("ERROR: %s() I2C bus busy\n", __func__);
         goto OUT;
        }
#endif

        /* High level means NAK */
        if( READ_SDA() )
             rv = ACK_ERROR;
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
OUT:
        SCL_L();
        delay_us(20);

        return rv;
}



uint8_t I2c_ReadByte(uint8_t *byte, uint8_t ack)
{
        uint8_t rv = NO_ERROR;
        uint8_t mask;

        *byte = 0x00;

        SDA_IN();

        /* 1Byte=8bit, MSB send: bit[7]-->bit[0] */
        for(mask = 0x80; mask > 0; mask >>= 1)
        {
          SCL_H();    // start clock on SCL-line
          delay_us(1); // clock set-up time (t_SU;CLK)

#ifdef I2C_CLK_STRETCH_TIMEOUT
          rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
          if( rv  )
          {
              i2c_print("ERROR: %s() I2C bus busy\n", __func__);
              goto OUT;
           }
#endif

           if(READ_SDA())
             *byte |= mask; // read bit

           SCL_L();
           delay_us(1); // data hold time(t_HD;DAT)
        }
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
        /* clk #9 send ACK/NAK to slave */
        if(ack == ACK)
        {
          SDA_OUT();
          SDA_L(); // send Acknowledge if necessary
        }
        else if( ack == NAK )
        {
          SDA_OUT();
          SDA_H(); // send NotAcknowledge if necessary
        }

        delay_us(1); // data set-up time (t_SU;DAT)
        SCL_H();     // clk #9 for ack
        delay_us(2); // SCL high time (t_HIGH)

#ifdef I2C_CLK_STRETCH_TIMEOUT
        rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
        if( rv  )
        {
         i2c_print("ERROR: %s() I2C bus busy\n", __func__);
        }
#endif

OUT:
        SCL_L();
        delay_us(2);  // wait to see byte package on scope

  return rv;
}


uint8_t I2c_SendAddress(uint8_t addr)
{
        return I2c_WriteByte(addr);
}
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
int I2C_Master_Receive(uint8_t addr, uint8_t *buf, int len)
{
  int      i;
  int      rv = NO_ERROR;
  uint8_t   byte;

  I2c_StartCondition();

  rv = I2c_SendAddress(addr);
  if( rv )
  {
          i2c_print("Send I2C read address failure,  rv=%d\n", rv);
          goto OUT;
  }

#ifdef I2C_CLK_STRETCH_TIMEOUT
  /* wait while clock streching */
  rv = I2c_WaitWhileClockStretching(I2C_CLK_STRETCH_TIMEOUT);
  if( rv )
  {
          i2c_print("ERROR: %s() I2C wait clock stretching failure, rv=%d\n", __func__, rv);
          return rv;
  }
#endif

  for (i=0; i<len; i++)
  {
     if( !I2c_ReadByte(&byte, ACK) )
          buf[i] = byte;
     else
          goto OUT;
  }

OUT:
  I2c_StopCondition();
  return rv;
}
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
int I2C_Master_Transmit(uint8_t addr, uint8_t *data, int bytes)
{
        int      i;
        int      rv = NO_ERROR;

        if(!data)
        {
          return PARM_ERROR;
        }

        i2c_print("I2C Mastr start transimit [%d] bytes data to addr [0x%02x]\n", bytes, addr);
        I2c_StartCondition();

        rv = I2c_SendAddress(addr);
        if( rv )
        {
          goto OUT;
        }

        for (i=0; i<bytes; i++)
        {
          if( NO_ERROR != (rv=I2c_WriteByte(data[i])) )
          {
             break;
          }
        }

OUT:
        I2c_StopCondition();
        return rv;
}
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

3.3. GPIO 模拟 I2C 头文件

在 Core/Inc 下创建并编写 GPIO 模拟 I2C 头文件 gpio_i2c_sht30.h

```
/*****************************************************************************
*   Copyright: (C)2021 LingYun IoT System Studio <www.weike-iot.com>
*      Author: GuoWenxue<guowenxue@gmail.com> QQ: 281143292
* Description: BearKE1 NB-IoT Board GPIO simulate I2C bus source code
*
*   ChangeLog:
*       Version    Date       Author           Description
*       V1.0.0  2021.08.10  GuoWenxue      Release initial version
*****************************************************************************/
#ifndef INC_GPIO_I2C_SHT30_H_
#define INC_GPIO_I2C_SHT30_H_

/* I2C Bus ERROR Number */
enum
{
  NO_ERROR       = 0x00, // no error
  PARM_ERROR     = 0x01, // parameter out of range error
  ACK_ERROR      = 0x02, // no acknowledgment error
  CHECKSUM_ERROR = 0x04, // checksum mismatch error
  TIMEOUT_ERROR  = 0x08, // timeout error
  BUS_ERROR      = 0x10, // bus busy
};

enum
{
         ACK_NONE, /* Without ACK/NAK Reply */
         ACK,      /* Reply with ACK */
         NAK,      /* Reply with NAK */
};

extern int I2C_Master_Receive(uint8_t addr, uint8_t *buf, int len);

extern int I2C_Master_Transmit(uint8_t addr, uint8_t *data, int bytes);

#endif /* INC_GPIO_I2C_SHT30_H_ */
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

3.4. SHT30 源码更新

修改 SHT30 驱动文件 sth30.c 文件，将之前的 HAL 库里的硬件 I2C 接口函数改成我们 GPIO 模拟实现的 I2C 接口函数。

```
#include <stdio.h>
#include "stm32l4xx_hal.h"
#include "sht30.h"

文件开始处修改头文件的包含，由宏的条件编译控制
/* 通过该宏控制是使用 HAL 库里的 I2C 接口还是使用 GPIO 模拟串口的接口 */
#define CONFIG_GPIO_I2C

#ifdef CONFIG_GPIO_I2C
#include "gpio_i2c_sht30.h"
#else
#include "i2c.h"
#endif

//#define CONFIG_SHT30_DEBUG

#ifdef CONFIG_SHT30_DEBUG
#define sht30_print(format,args...) printf(format, ##args)
#else
#define sht30_print(format,args...) do{} while(0)
#endif

static int sht30_send_cmd(SHT30_CMD cmd)
{
    uint8_t buf[2];

    buf[0] = cmd >> 8;
    buf[1] = cmd;

#ifdef CONFIG_GPIO_I2C
    return I2C_Master_Transmit(SHT30_ADDR_WR, (uint8_t*)buf, 2);
#else
    return HAL_I2C_Master_Transmit(&hi2c2, SHT30_ADDR_WR, (uint8_t*)buf, 2, 0xFFFF);
#endif
}

... ...
```

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

```
static int sht30_single_shot_measurement(uint8_t *buf, uint8_t buf_size)
{
        uint16_t     cmd = HIGH_ENABLED_CMD;  /* High with clock stretching */
        uint8_t      rv;

        if( !buf || buf_size<SHT30_DATA_SIZE )
        {
          sht30_print("%s(): Invalid input arguments\n", __func__);
          return -1;
        }

        rv = sht30_send_cmd(cmd);
        if( rv )
        {
          sht30_print("ERROR: SHT30 send measurement command failure, rv=%d\n", rv);
          sht30_soft_reset();
          return -2;
        }

#ifdef CONFIG_GPIO_I2C
        rv = I2C_Master_Receive(SHT30_ADDR_RD, buf, SHT30_DATA_SIZE);
#else
        rv = HAL_I2C_Master_Receive(&hi2c2, SHT30_ADDR_RD, buf, SHT30_DATA_SIZE, 0xFFFF);
#endif
        if(rv)
        {
          sht30_print("ERROR: SHT30 read measurement result failure, rv=%d\n", rv);
          return -3;
        }

        return 0;
}
```

其它代码保持不变，只需要修改这三处替换掉相应的接口函数即可。

版权声明： 本文档由凌云实验室郭工编著，仅供实验室内部教学使用，未经许可任何人不得以任何形式分享或传播！
CopyRight(C) 2021 凌云物网智科实验室 www.weike-iot.com 郭工：QQ 281143292

3.5. 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。