@TOC

一、 任务

学习I2C总线通信协议，使用STM32F103完成基于I2C协议的AHT20温湿度传感器的数据采集，并将采集的温度-湿度值通过串口输出。具体任务：

1）解释什么是“软件I2C”和“硬件I2C”？ （阅读野火配套教材的第23章“I2C--读写EEPROM”原理章节）

2）阅读AHT20数据手册，编程实现：每隔2秒钟采集一次温湿度数据，并通过串口发送到上位机（win10）。

二、 准备工作

① stm32F103c8t6
② AHT20温湿度传感器
③ TTL转串口
④ AHT20封装函数
可以使用以下代码：
AHT20.h

```
// #ifndef INC_AHT20_H_
#define INC_AHT20_H_

#include "i2c.h"

//****************************************
// 定义 AHT20 命令
//****************************************
#define AHT20_Write             0x00    //读取
#define AHT20_Read              0x01    //写入



//****************************************
// 定义 AHT20 地址
//****************************************
#define AHT20_SLAVE_ADDRESS     0x70        /* I2C从机地址 */



//****************************************
// 定义 AHT20 命令
//****************************************
#define AHT20_INIT_COMD         0xBE    //初始化 寄存器地址
#define AHT20_SoftReset         0xBA    //软复位 单指令
#define AHT20_TrigMeasure_COMD  0xAC    //触发测量 寄存器地址

// 存储AHT20传感器信息的结构体

typedef struct m_AHT20
{
    uint8_t alive;  // 0-器件不存在; 1-器件存在
    uint8_t flag;   // 读取/计算错误标志位。0-读取/计算数据正常; 1-读取/计算设备失败
```

```
    uint32_t HT[2]; // 湿度、温度 原始传感器的值(20Bit).

    float RH;       // 湿度，转换单位后的实际值，标准单位%
    float Temp;     // 温度，转换单位后的实际值，标准单位°C
}AHT20_StructureTypedef;

extern  AHT20_StructureTypedef  Humiture;

uint8_t AHT20_Init(void);
void    AHT20_Start_Init(void);
uint8_t AHT20_ReadHT(uint32_t *HT);
uint8_t StandardUnitCon(AHT20_StructureTypedef *aht);
uint8_t AHT20_Get_Value(AHT20_StructureTypedef *p);

#endif /* INC_AHT20_H_ */
```

ATH20.c

```
#include "AHT20.h"

AHT20_StructureTypedef  Humiture;


/**
  * @brief  读AHT20 设备状态字
  * @param  void
  * @retval uint8_t 设备状态字
  */
static uint8_t AHT20_ReadStatusCmd(void)
{
    uint8_t tmp = 0;
    HAL_I2C_Master_Receive(&hi2c1, AHT20_SLAVE_ADDRESS, &tmp, 1, 0xFFFF);
    return tmp;
}

/**
  * @brief  读AHT20 设备状态字 中的Bit3: 校准使能位
  * @param  void
  * @retval uint8_t 校准使能位：1 - 已校准; 0 - 未校准
  */
static uint8_t AHT20_ReadCalEnableCmd(void)
{
    uint8_t tmp = 0;
    tmp = AHT20_ReadStatusCmd();
    return (tmp>>3)&0x01;
}

/**
  * @brief  读AHT20 设备状态字 中的Bit7: 忙标志
  * @param  void
  * @retval uint8_t 忙标志：1 - 设备忙; 0 - 设备空闲
  */
static uint8_t AHT20_ReadBusyCmd(void)
{
    uint8_t tmp = 0;
    tmp = AHT20_ReadStatusCmd();
    return (tmp>>7)&0x01;
```

```c
}

/**
  * @brief  AHT20 芯片初始化命令
  * @param  void
  * @retval void
  */
static void AHT20_IcInitCmd(void)
{
    uint8_t tmp = AHT20_INIT_COMD;
    HAL_I2C_Master_Transmit(&hi2c1, AHT20_SLAVE_ADDRESS, &tmp, 1, 0xFFFF);
}

/**
  * @brief  AHT20 软复位命令
  * @param  void
  * @retval void
  */
static void AHT20_SoftResetCmd(void)
{
    uint8_t tmp = AHT20_SoftReset;
    HAL_I2C_Master_Transmit(&hi2c1, AHT20_SLAVE_ADDRESS, &tmp, 1, 0xFFFF);
}

/**
  * @brief  AHT20 触发测量命令
  * @param  void
  * @retval void
  */
static void AHT20_TrigMeasureCmd(void)
{
    static uint8_t tmp[3] = {AHT20_TrigMeasure_COMD, 0x33, 0x00};
    HAL_I2C_Master_Transmit(&hi2c1, AHT20_SLAVE_ADDRESS, tmp, 3, 0xFFFF);
}


/**
  * @brief  AHT20 设备初始化
  * @param  void
  * @retval uint8_t：0 - 初始化AHT20设备成功; 1 - 初始化AHT20失败，可能设备不存在或器件已损坏
  */
uint8_t AHT20_Init(void)
{
    uint8_t rcnt = 2+1;//软复位命令 重试次数，2次
    uint8_t icnt = 2+1;//初始化命令 重试次数，2次

    while(--rcnt)
    {
        icnt = 2+1;

        HAL_Delay(40);//上电后要等待40ms
        // 读取温湿度之前，首先检查[校准使能位]是否为1
        while((!AHT20_ReadCalEnableCmd()) && (--icnt))// 2次重试机会
        {
            HAL_Delay(1);
            // 如果不为1，要发送初始化命令
            AHT20_IcInitCmd();
```

```
            HAL_Delay(40);//这个时间手册没说，按上电时间算40ms
        }

        if(icnt)//[校准使能位]为1,校准正常
        {
            break;//退出rcnt循环
        }
        else//[校准使能位]为0,校准错误
        {
            AHT20_SoftResetCmd();//软复位AHT20器件，重试
            HAL_Delay(20);//这个时间手册标明不超过20ms.
        }
    }

    if(rcnt)
    {
        return 0;// AHT20设备初始化正常
    }
    else
    {
        return 1;// AHT20设备初始化失败
    }
}

/**
  * @brief  AHT20 寄存器复位
  * @param  void
  * @retval void
  */
static void AHT20_Register_Reset(uint8_t addr)
{
  uint8_t  iic_tx[3] = {0}, iic_rx[3] = {0};

  iic_tx[0] = addr;
  HAL_I2C_Master_Transmit(&hi2c1, AHT20_SLAVE_ADDRESS, iic_tx, 3, 0xFFFF);
  HAL_Delay(5);
  HAL_I2C_Master_Receive(&hi2c1, AHT20_SLAVE_ADDRESS, iic_rx, 3, 0xFFFF);
  HAL_Delay(10);
  iic_tx[0] = 0xB0 | addr;
  iic_tx[1] = iic_rx[1];
  iic_tx[2] = iic_rx[2];
  HAL_I2C_Master_Transmit(&hi2c1, AHT20_SLAVE_ADDRESS, iic_tx, 3, 0xFFFF);
  HAL_Delay(10);
}


/**
  * @brief  AHT20 设备开始初始化
  * @param  void
  * @retval void
  */
void AHT20_Start_Init(void)
{
  static uint8_t    temp[3] = {0x1B, 0x1C, 0x1E}, i;
  for(i = 0; i < 3; i++)
  {
    AHT20_Register_Reset(temp[i]);
  }
```

```
}



/**
  * @brief  AHT20 设备读取 相对湿度和温度（原始数据20Bit）
  * @param  uint32_t *HT：存储20Bit原始数据的uint32数组
  * @retval uint8_t：0-读取数据正常；1-读取设备失败，设备一直处于忙状态，不能获取数据
  */
uint8_t AHT20_ReadHT(uint32_t *HT)
{
    uint8_t cnt=3+1;//忙标志 重试次数，3次
    uint8_t tmp[6];
    uint32_t RetuData = 0;

    // 发送触发测量命令
    AHT20_TrigMeasureCmd();

    do{
        HAL_Delay(75);//等待75ms待测量完成，忙标志Bit7为0
    }while(AHT20_ReadBusyCmd() && (--cnt));//重试3次

    if(cnt)//设备闲，可以读温湿度数据
    {
        HAL_Delay(5);
        // 读温湿度数据
        HAL_I2C_Master_Receive(&hi2c1, AHT20_SLAVE_ADDRESS, tmp, 6, 0XFFFF);
        // 计算相对湿度RH。原始值，未计算为标准单位%。
        RetuData = 0;
        RetuData = (RetuData|tmp[1]) << 8;
        RetuData = (RetuData|tmp[2]) << 8;
        RetuData = (RetuData|tmp[3]);
        RetuData = RetuData >> 4;
        HT[0] = RetuData;

        // 计算温度T。原始值，未计算为标准单位°C。
        RetuData = 0;
        RetuData = (RetuData|tmp[3]) << 8;
        RetuData = (RetuData|tmp[4]) << 8;
        RetuData = (RetuData|tmp[5]);
        RetuData = RetuData&0xfffff;
        HT[1] = RetuData;

        return 0;
    }
    else//设备忙，返回读取失败
    {
        return 1;
    }
}

/**
  * @brief  AHT20 温湿度信号转换（由20Bit原始数据，转换为标准单位RH=%，T=°C）
  * @param  struct m_AHT20* aht：存储AHT20传感器信息的结构体
  * @retval uint8_t：0-计算数据正常；1-计算数据失败，计算值超出元件手册规格范围
  */
uint8_t StandardUnitCon(AHT20_StructureTypedef *aht)
{
```

```
    aht->RH = (double)aht->HT[0] *100 / 1048576;//2^20=1048576 //原式：
(double)aht->HT[0] / 1048576 *100，为了浮点精度改为现在的
    aht->Temp = (double)aht->HT[1] *200 / 1048576 -50;

    //限幅,RH=0~100%; Temp=-40~85°C
    if((aht->RH >=0)&&(aht->RH <=10000) && (aht->Temp >=-4000)&&(aht->Temp
<=8500))
    {
        aht->flag = 0;
        return 0;//测量数据正常
    }
    else
    {
        aht->flag = 1;
        return 1;//测量数据超出范围，错误
    }
}


/**
  * @brief  AHT20 温湿度信号转换（由20Bit原始数据，转换为标准单位RH=%，T=°C）
  * @param  struct m_AHT20* aht：存储AHT20传感器信息的结构体
  * @retval uint8_t：0-计算数据正常；1-计算数据失败，计算值超出元件手册规格范围
  */
uint8_t AHT20_Get_Value(AHT20_StructureTypedef *p)
{
  uint8_t   temp = 0;

  temp = AHT20_ReadHT(p->HT);

  if(temp == 0)
  {
    temp = StandardUnitCon(p);
  }

  return temp;
}
```

三、 问题解决

Ⅰ 了解I2C总线协议

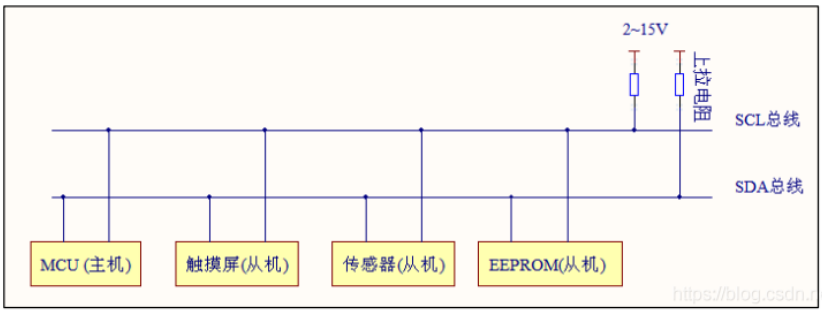
① 什么是I2C协议

I2C 通讯协议(Inter - Integrated Circuit)是由 Phiilps 公司开发的，由于它引脚少，硬件实现简单，可扩展性强，不需要 USART、CAN 等通讯协议的外部收发设备，现在被广泛地使用在系统内多个集成电路(IC)间的通讯。

② I2C协议的物理层和协议层

1. 物理层
   I2C是一个支持设备的总线。可连接多个 I2C 通讯设备，支持多个通讯主机及多个通讯从机。对于 I2C 总线，只使用两条总线线路，一条双向串行数据线(SDA)，一条串行时钟线(SCL)。

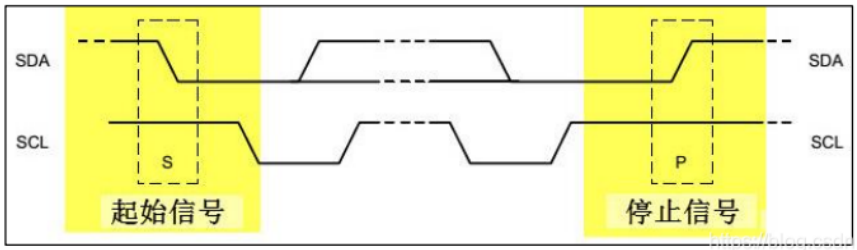
I2C 通讯设备常用连接方式（引用野火资料中的图）



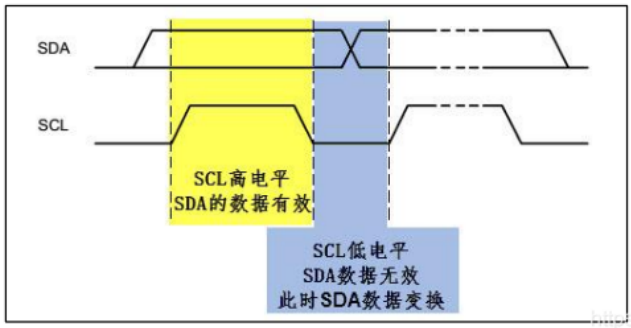
2. 协议层

> 主要是定义了通讯的起始和停止信号、数据有效性、响应、仲裁、时钟同步和地址广播等。

通讯的起始和停止信号

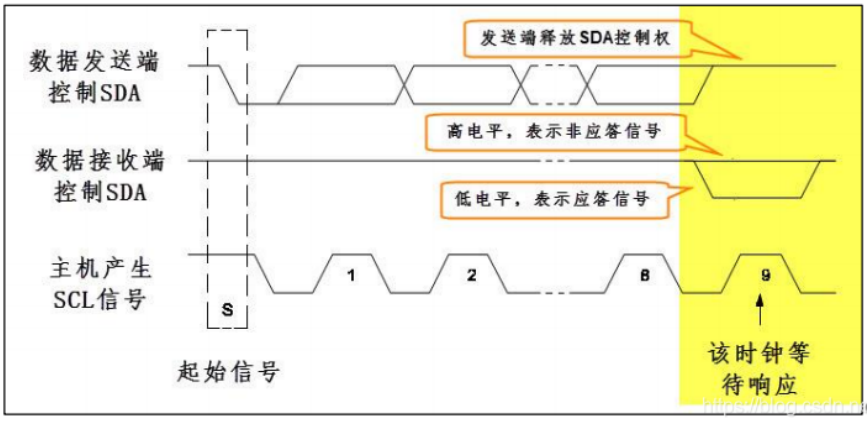


数据有效性



从图中可以看出I2C在通讯的时候，只有在SCL处于高电平时，SDA的数据传输才是有效的。SDA 信号线是用于传输数据，SCL 信号线是保证数据同步。

响应



当SDA传输数据后，接收方对接受到的数据进行一个应答。如果希望继续进行传输数据，则回应应答信号（低电平），否则回应非应答信号（高电平）。

③ I2C的两种方式——硬件I2C和软件I2C

1. 硬件I2C
   直接利用 STM32 芯片中的硬件 I2C 外设。

> **硬件I2C的使用**
> 只要配置好对应的寄存器，外设就会产生标准串口协议的时序。在初始化好 I2C 外设后，只需要把某寄存器位置 1，此时外设就会控制对应的 SCL 及 SDA 线自动产生 I2C 起始信号，不需要内核直接控制引脚的电平。

2. 软件I2C
   直接使用 CPU 内核按照 I2C 协议的要求控制 GPIO 输出高低电平，从而模拟I2C。

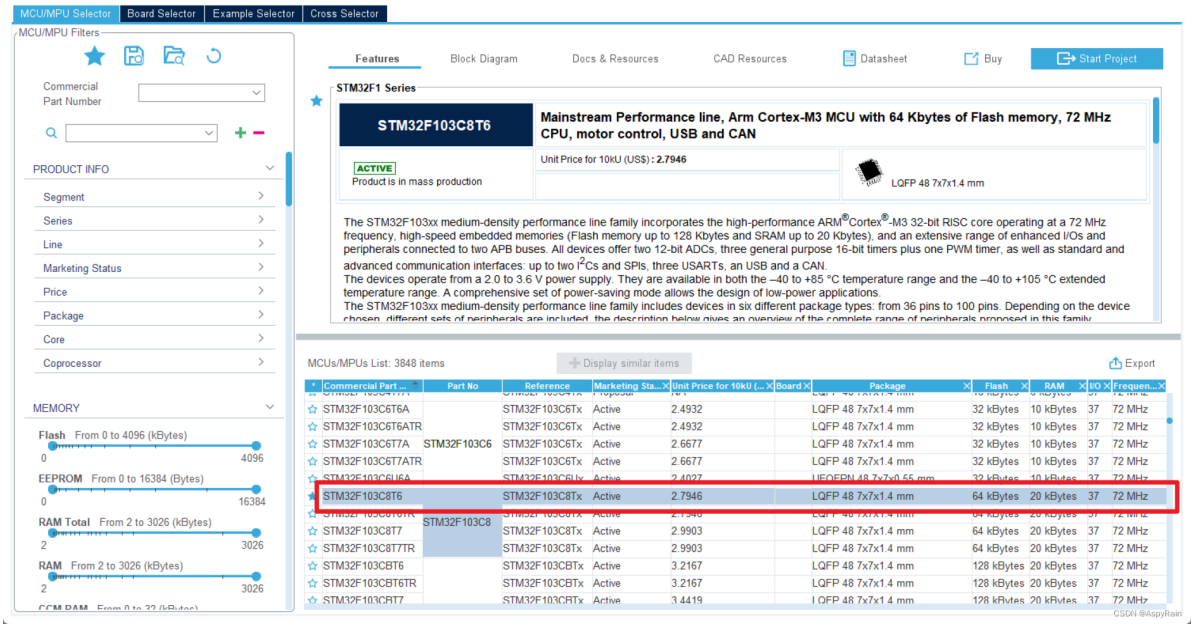
> **软件I2C的使用**
> 需要在控制产生 I2C 的起始信号时，控制作为 SCL 线的 GPIO 引脚输出高电平，然后控制作为 SDA 线的 GPIO 引脚在此期间完成由高电平至低电平的切换，最后再控制SCL 线切换为低电平，这样就输出了一个标准的 I2C 起始信号。

3. 区别
   硬件 I2C 直接使用外设来控制引脚，可以减轻 CPU 的负担。不过使用硬件I2C 时必须使用某些固定的引脚作为 SCL 和 SDA，软件模拟 I2C 则可以使用任意 GPIO 引脚，相对比较灵活。对于硬件I2C用法比较复杂，软件I2C的流程更清楚一些。如果要详细了解I2C的协议，使用软件I2C可能更好的理解这个过程。在使用I2C过程，硬件I2C可能通信更加快，更加稳定。

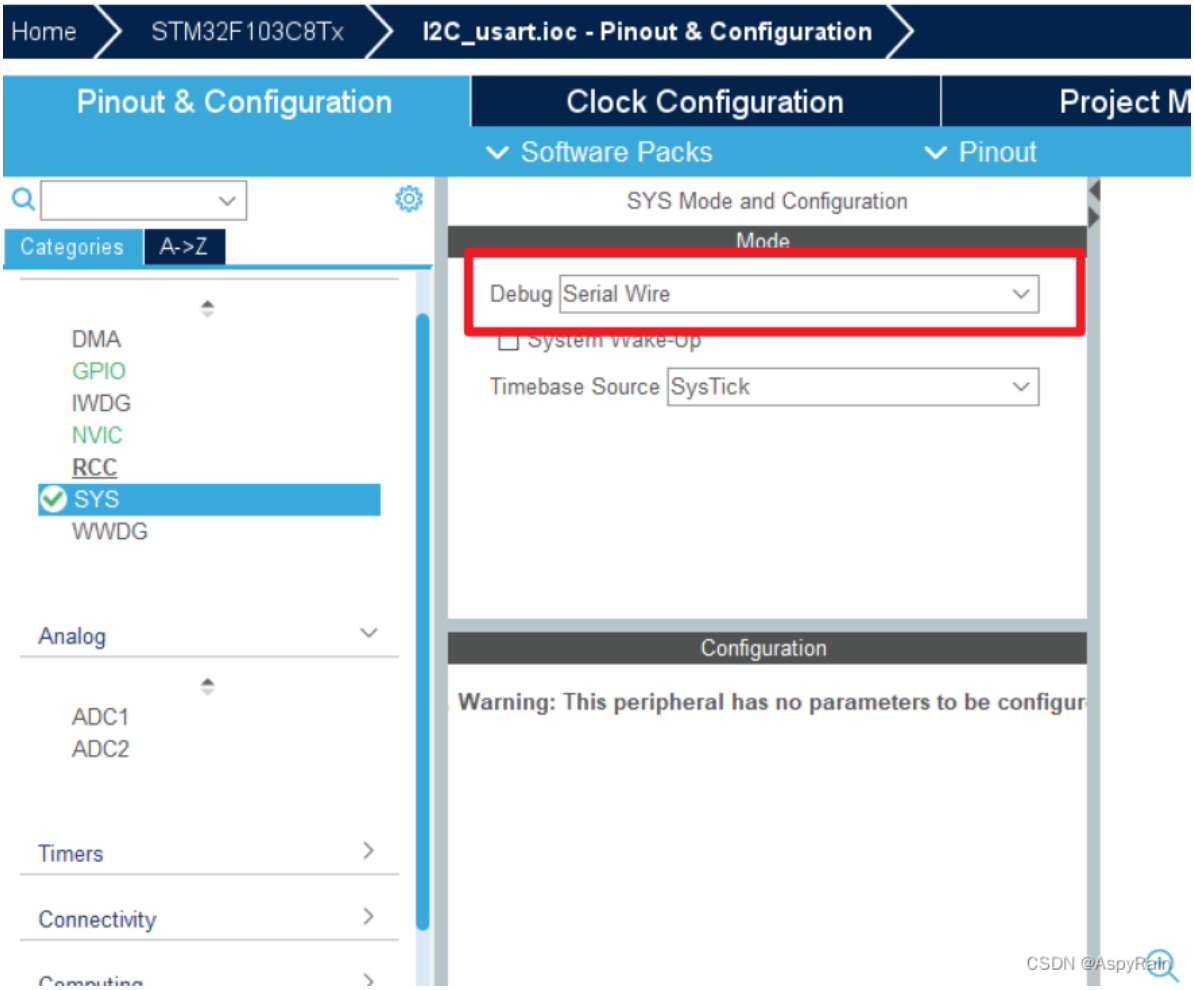
II 通过AHT20采集温湿度数据

① 新建工程

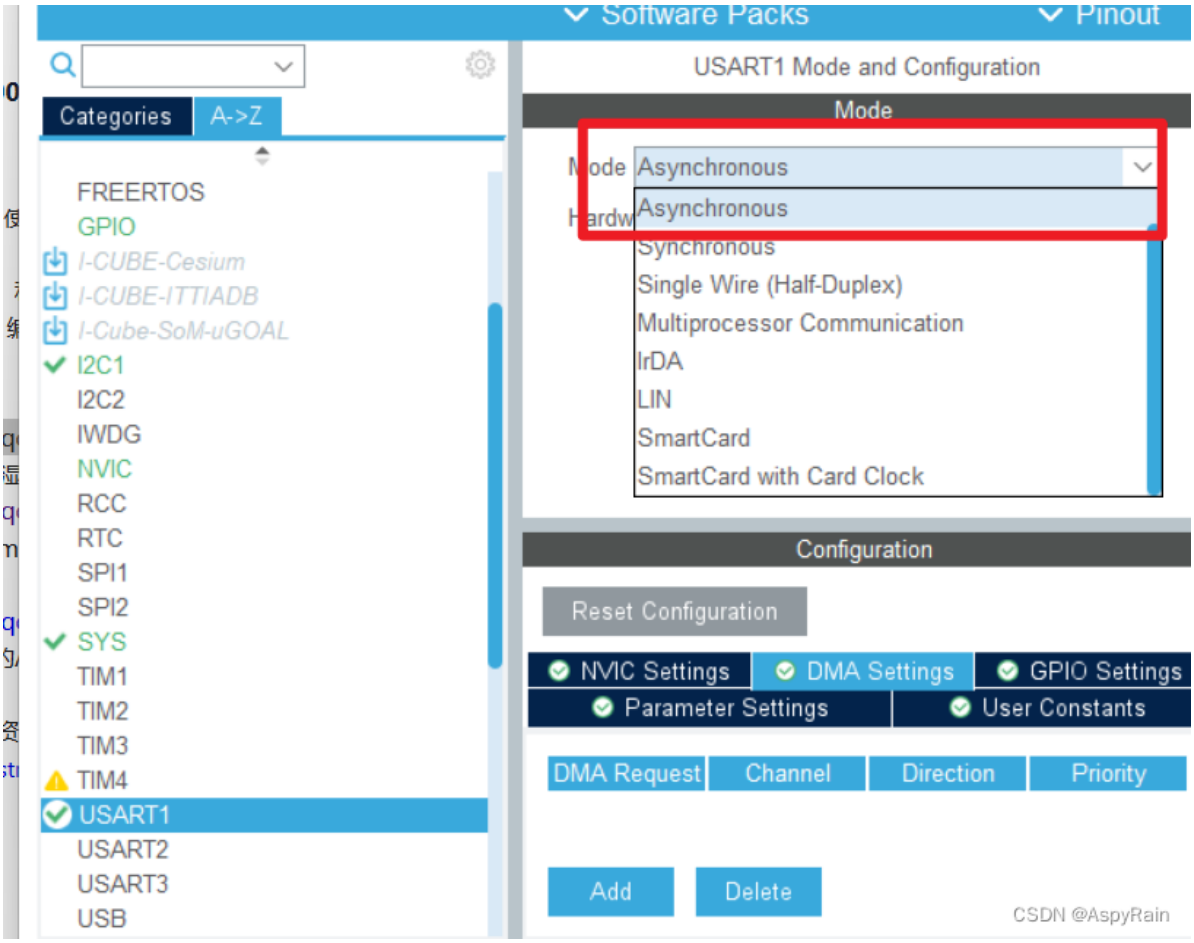
1. 打开cubeMX，新建工程，选择F103c8t6:



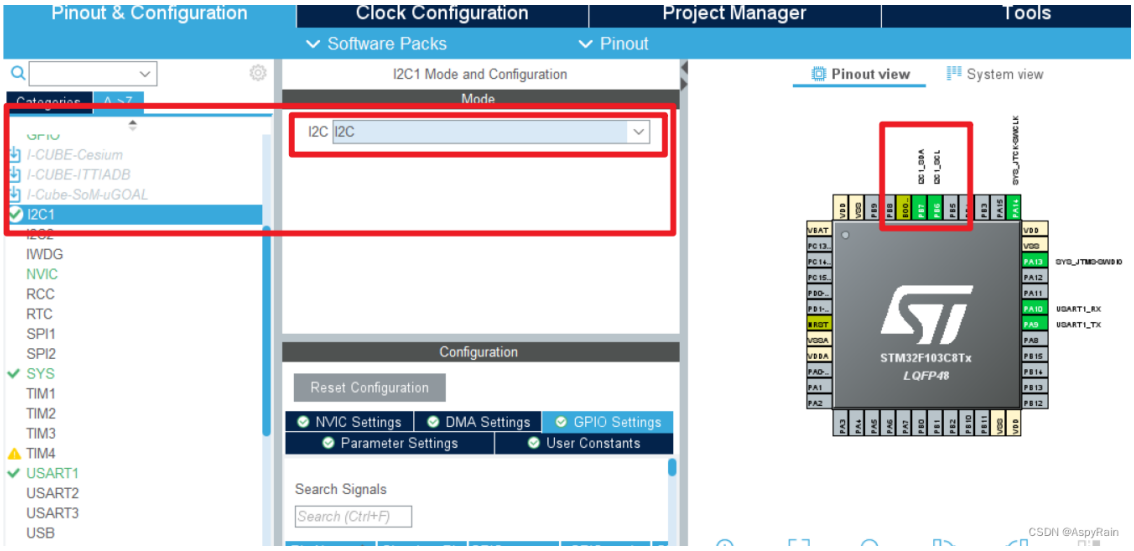
2. 在SYS里面选择


Home
STM32F103C8Tx
I2C_usart.ioc - Pinout & Configuration
Pinout & Configuration
Clock Configuration
Project M
Software Packs
Pinout
Categories
A->Z
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
ADC1
ADC2
Timers
Connectivity
SYS Mode and Configuration
Mode
Debug
Serial Wire
Timebase Source
SysTick
Configuration
Warning: This peripheral has no parameters to be configur
CSDN @AspyRain

3. 选择串口 通信

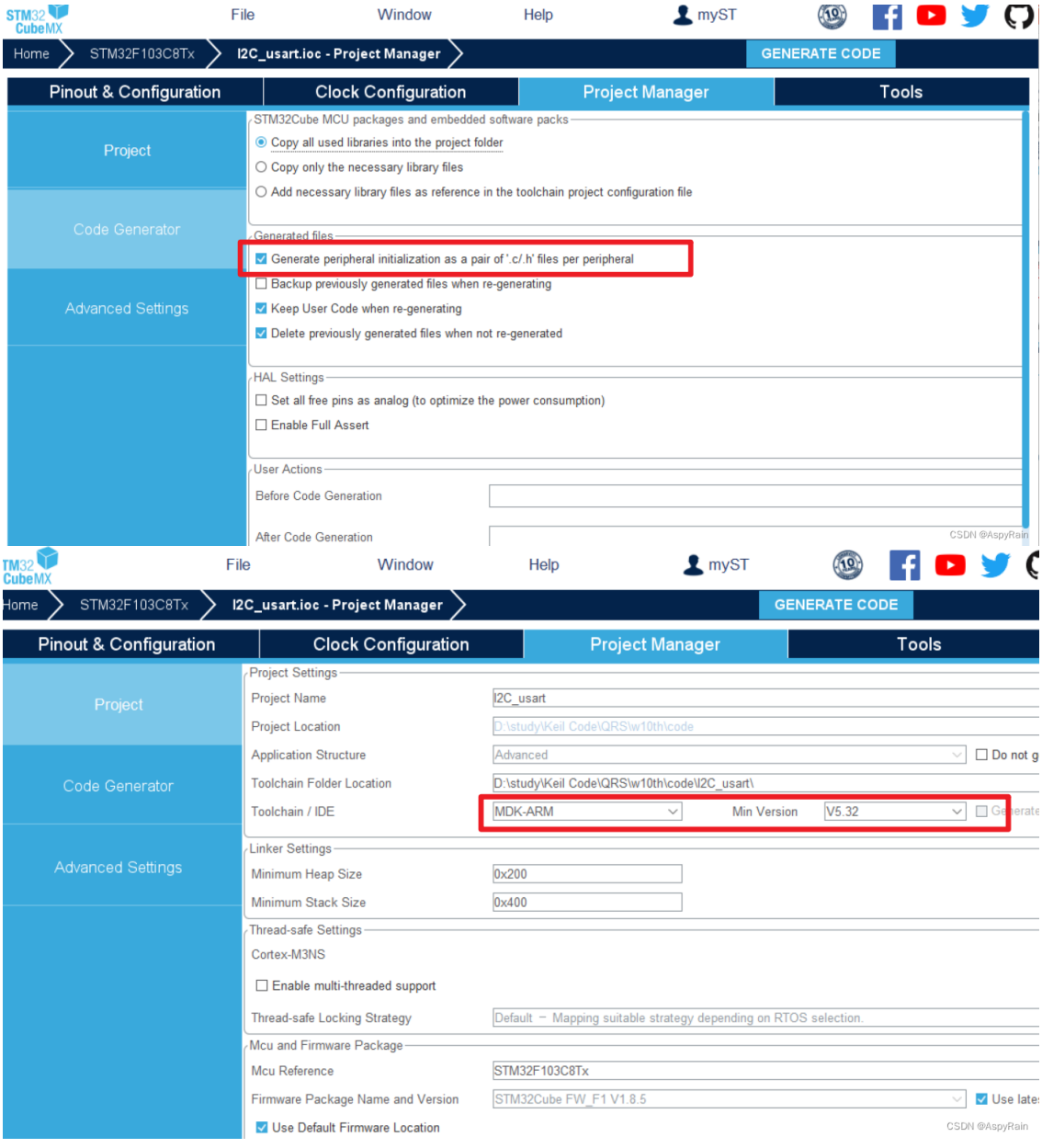

Software Packs
Pinout
Categories
A->Z
FREERTOS
GPIO
I-CUBE-Cesium
I-CUBE-ITTIADB
I-Cube-SoM-uGOAL
I2C1
I2C2
IWDG
NVIC
RCC
RTC
SPI1
SPI2
SYS
TIM1
TIM2
TIM3
TIM4
USART1
USART2
USART3
USB
USART1 Mode and Configuration
Mode
Asynchronous
Asynchronous
Synchronous
Single Wire (Half-Duplex)
Multiprocessor Communication
IrDA
LIN
SmartCard
SmartCard with Card Clock
Configuration
Reset Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
DMA Request
Channel
Direction
Priority
Add
Delete
CSDN @AspyRain

4. 选择I2C

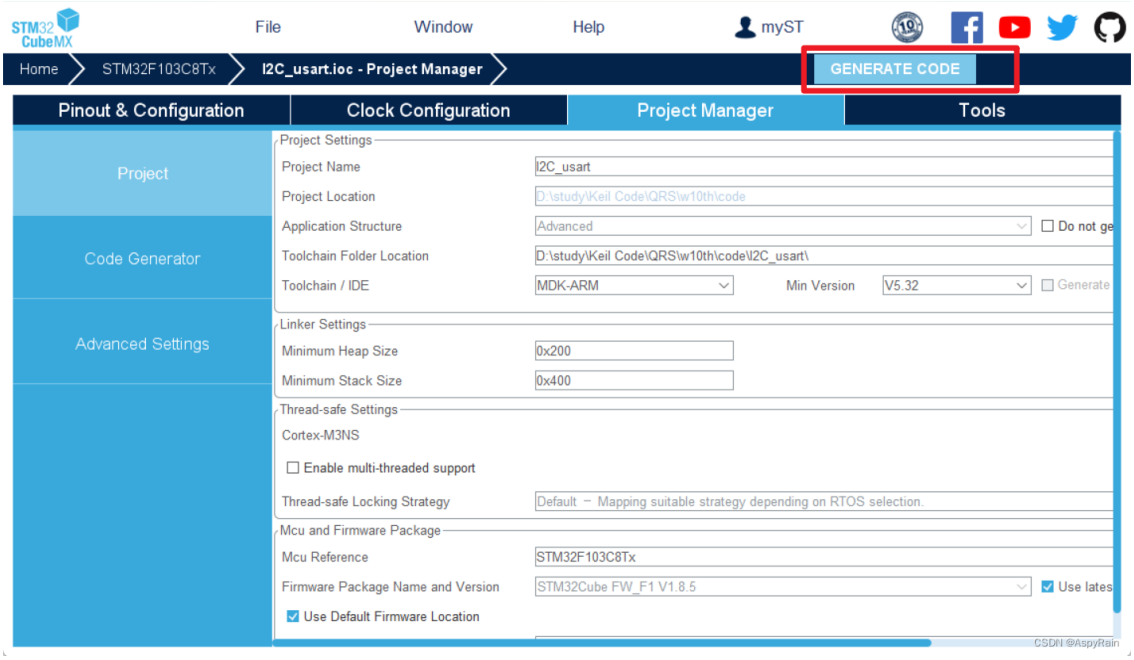


可以看到I2C对应的管脚

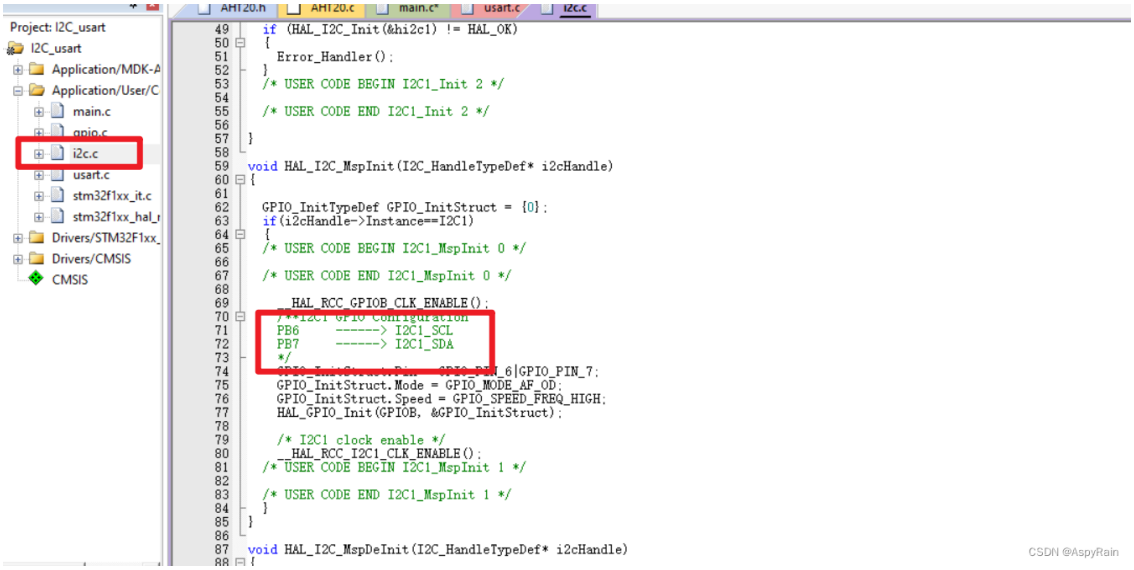
5. 做好项目配置



6. 生成项目



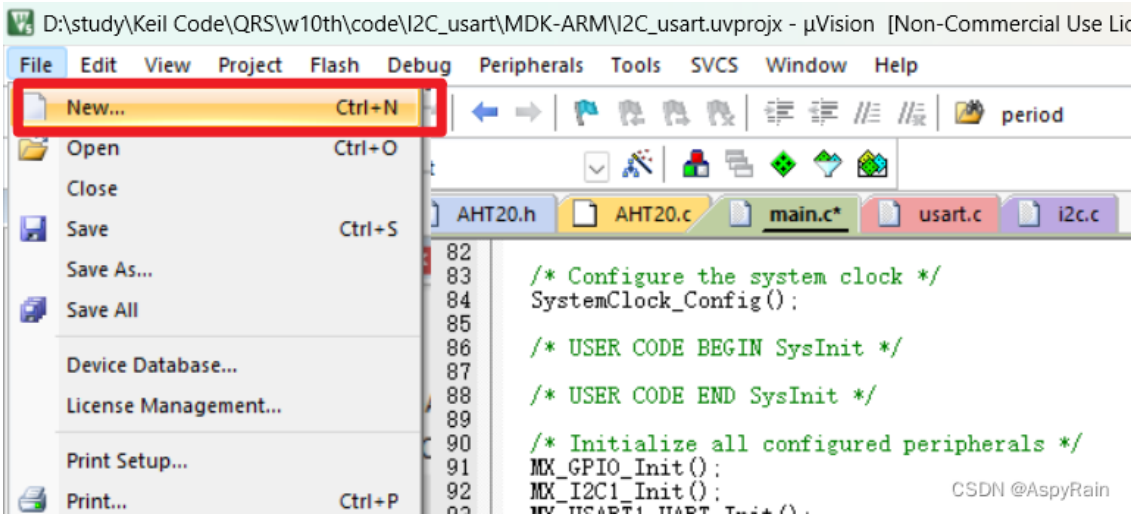
也可以在I2C.c里面看到管脚配置

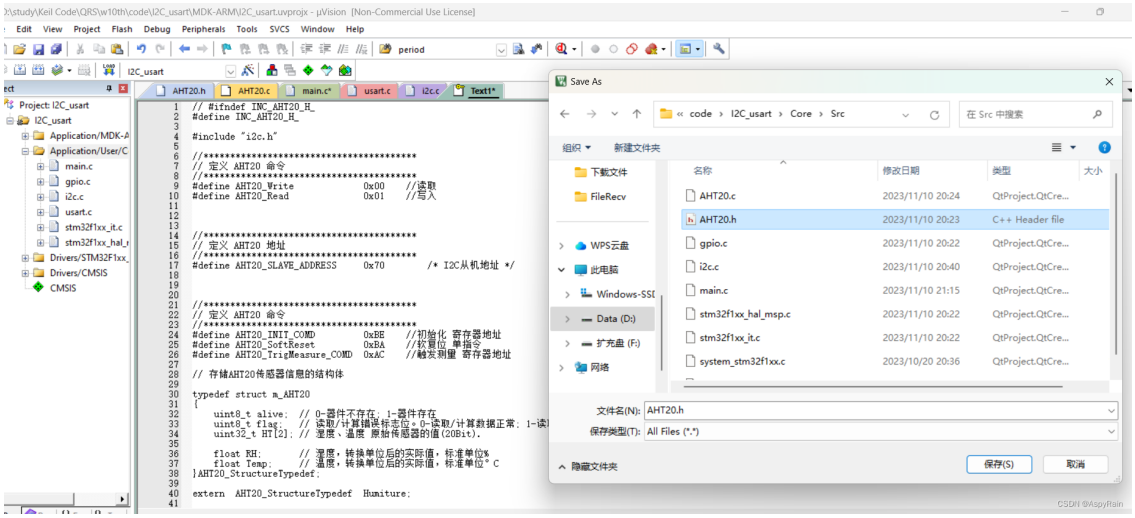


关于mdk的其余基础设置省略，于平时烧录时设置一致就行

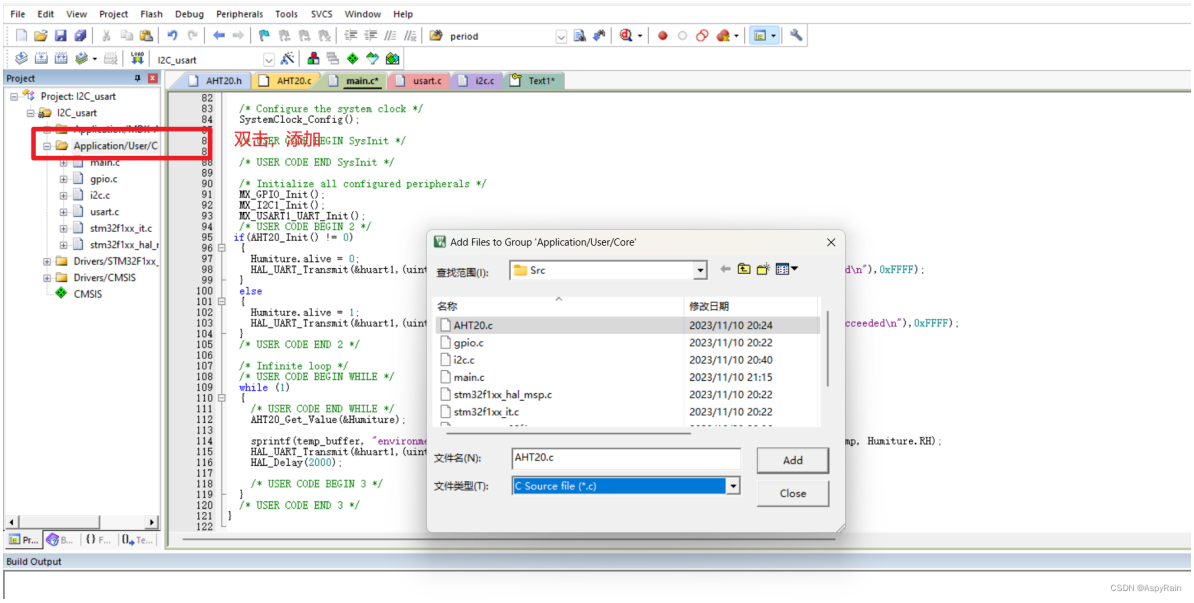
② 编辑项目

1. 新建ATH.h和ATH.c（代码在最开头）





2. 添加至勾选文件夹（添加c即可，但是要让ATH.H与ATH.c于同一文件夹）



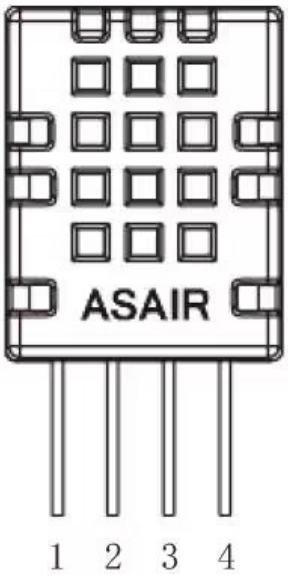
3. 编写main.c

```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */
```

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
 if(AHT20_Init() != 0)
  {
    Humiture.alive = 0;
        HAL_UART_Transmit(&huart1,(uint8_t *)"AHT20 Initialization
failed\n",strlen("AHT20 Initialization failed\n"),0xFFFF);
  }
  else
  {
    Humiture.alive = 1;
        HAL_UART_Transmit(&huart1,(uint8_t *)"AHT20 Initialization
succeeded\n",strlen("AHT20 Initialization succeeded\n"),0xFFFF);
  }
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    AHT20_Get_Value(&Humiture);

    sprintf(temp_buffer, "environment temperature is : %.2f;\nenvironment
humidity is : %.2f\n",Humiture.Temp, Humiture.RH);
    HAL_UART_Transmit(&huart1,(uint8_t
*)temp_buffer,strlen(temp_buffer),0xFFFF);
    HAL_Delay(2000);

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

③ 编译烧录

连接配置：
首先了解AHT20的管脚



模块正面有孔

注意管脚的编号和定义。接错会炸！！！

引脚	名称	描述
1	VDD	接电源（2.2~5.5V）
2	SDA	串行数据，双向
3	GND	电源地
4	SCL	串行时钟，双向


```
__HAL_RCC_GPIOB_CLK_ENABLE();
/**I2C1 GPIO Configuration
PB6     ------> I2C1_SCL
PB7     ------> I2C1_SDA
*/
GPIO_InitStruct.Pin = GPIO_PIN
```


依据上图进行连接

结果：

四、 学习总结

学习I2C总线通信协议，并在STM32F103微控制器上完成基于I2C协议的AHT20温湿度传感器数据采集以及通过串口输出的任务，让我对嵌入式系统开发有了更深刻的理解。

首先，软件I2C和硬件I2C是两种实现I2C通信协议的方式。软件I2C通过软件模拟I2C协议，适用于一些不支持硬件I2C的情况，但速度较慢。硬件I2C则是使用微控制器内置的硬件模块来直接处理I2C通信，速度更快且占用更少的处理器资源。

在实际操作中，通过阅读AHT20数据手册，我学会了如何配置STM32F103的I2C硬件模块，设置AHT20传感器的地址和寄存器，以及如何周期性地获取温湿度数据。使用串口输出将数据发送到上位机，我成功地实现了数据的可视化监测。

这次学习让我深入了解了I2C通信协议、传感器配置和STM32F103的相关编程技巧。通过不断调试和优化，我更加熟悉了嵌入式系统开发的工作流程，同时也提高了解决实际问题的能力。这是一次实践性很强的学习过程，为我今后在嵌入式系统领域的发展打下了坚实的基础。

参考链接:
[1]https://blog.csdn.net/qq_43279579/article/details/111597278
[2]https://blog.csdn.net/zhengwenbang/article/details/124285882