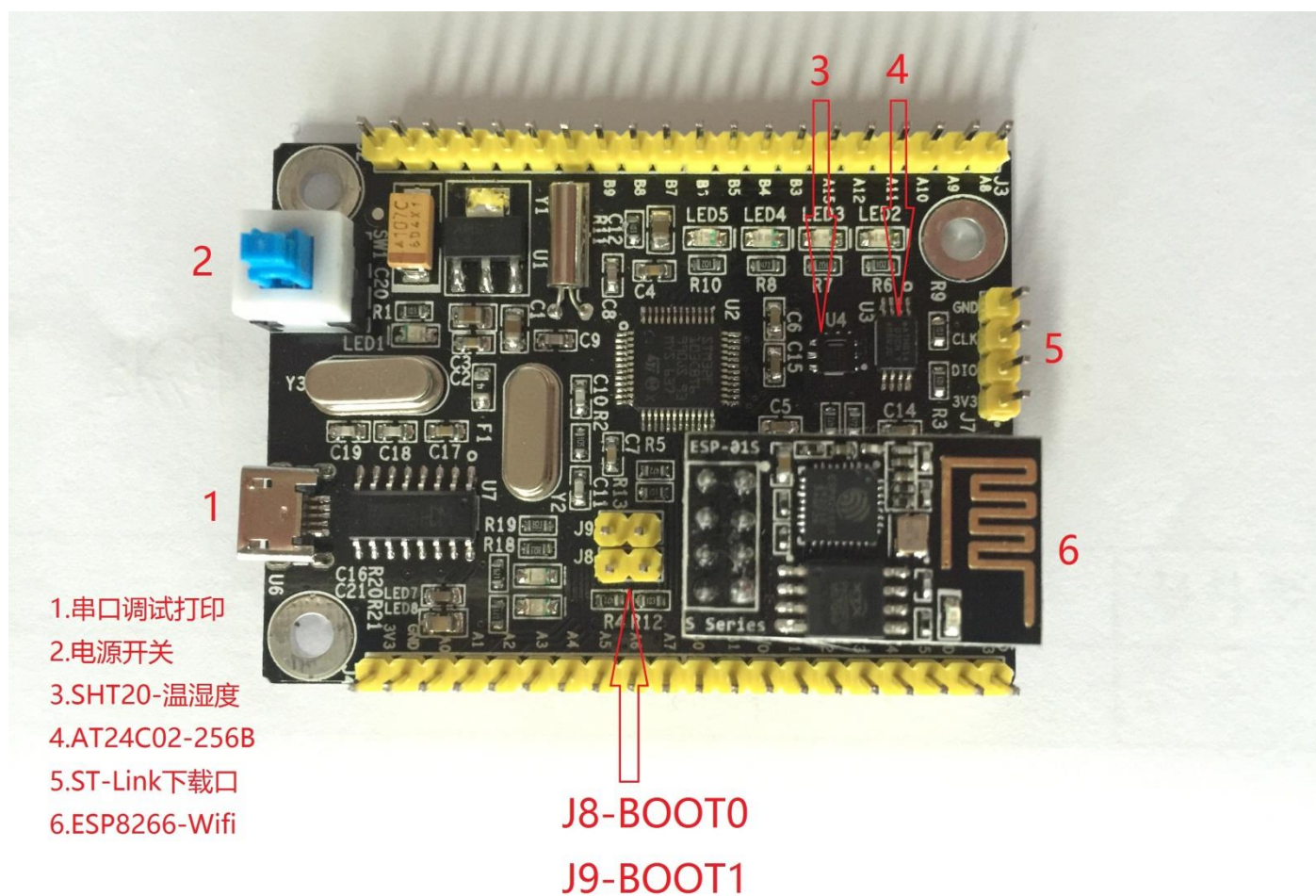


麒麟座 Mini 基础例程使用手册

2017 年 1 月 12 日

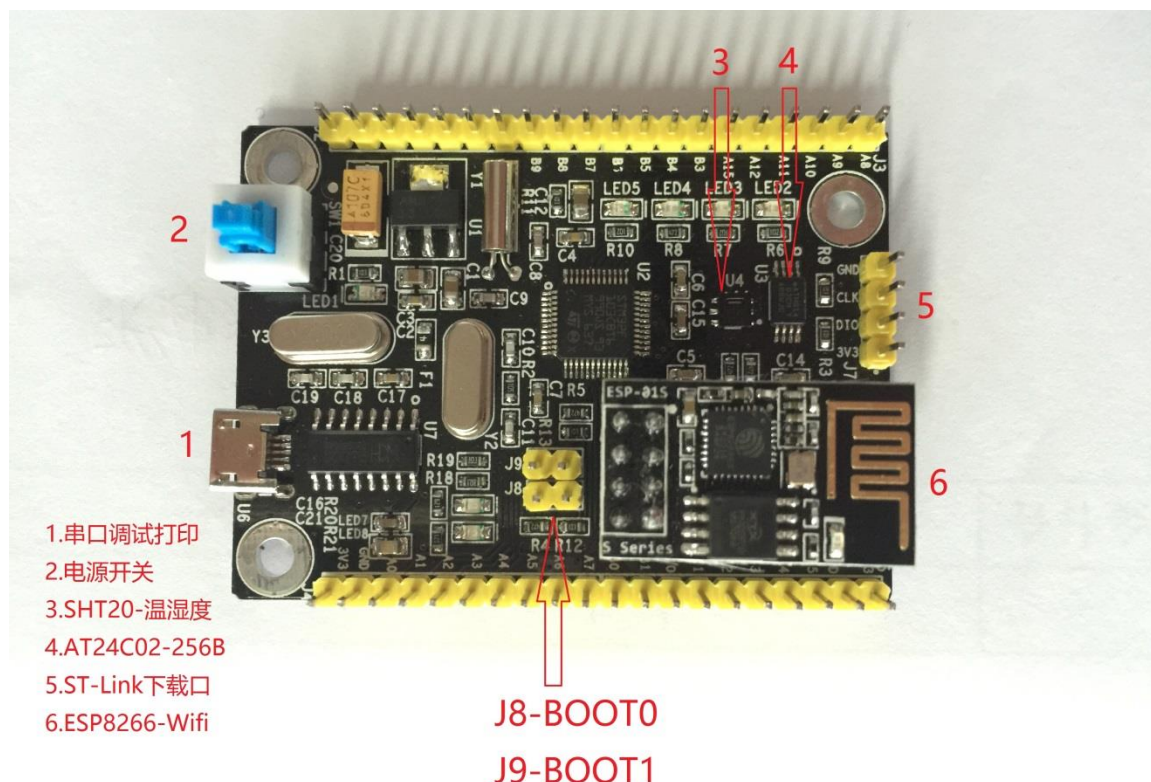


目录

第一章 开发板简介	3
1. 开发板介绍	3
2. MCU 介绍	3
第二章 基础例程学习	4
1. LED 流水灯	4
2. 串口打印	6
3. 串口打印-接收中断	9
4. AT24C02	12
5. SHT20-温湿度传感器	15

第一章 开发板简介

1. 开发板介绍



为了满足广大的物联网用户的需求、且帮助大家连接 OneNET 开放云平台，我们在麒麟座开发板的基础上减少部分传感器与硬件，以开发板采用底板+核心板这样的结构，开发出了麒麟座 mini 开发板。这样可以方便的更改开发板 MCU 的类型。开发板的 MCU 采用应用广泛的 STM32F103 以及 STC12LE5A60S2，两者可以交替使用。mini 开发板有 ESP8266、SHT20 温湿度传感器等硬件。

2. MCU 介绍

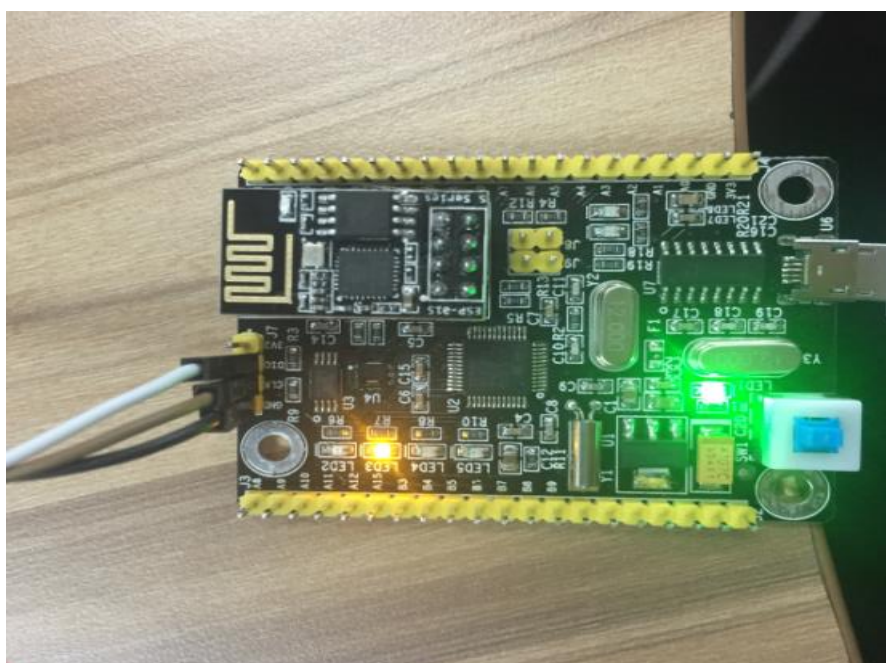
STM32F103xx 增强型系列使用高性能的 ARM Cortex-M3 32 位的 RISC 内核，工作频率为 72MHz，内置高速存储器(高达 128K 字节的闪存和 20K 字节的 SRAM)，丰富的增强 I/O 端口和联接到两条 APB 总线的外设。所有型号的器件都包含 2 个 12 位的 ADC、3 个通用 16 位定时器和一个 PWM 定时器，还

包含标准和先进的通信接口：多达 2 个 I2C 和 SPI、3 个 USART、一个 USB 和一个 CAN。STM32F103xx 增强型系列工作于 -40°C 至 $+105^{\circ}\text{C}$ 的温度范围，供电电压 2.0V 至 3.6V，一系列的省电模式保证低功耗应用的要求。完整的 STM32F103xx 增强型系列产品包括从 36 脚至 100 脚的五种不同封装形式；根据不同的封装形式，器件中的外设配置不尽相同。

第二章 基础例程学习

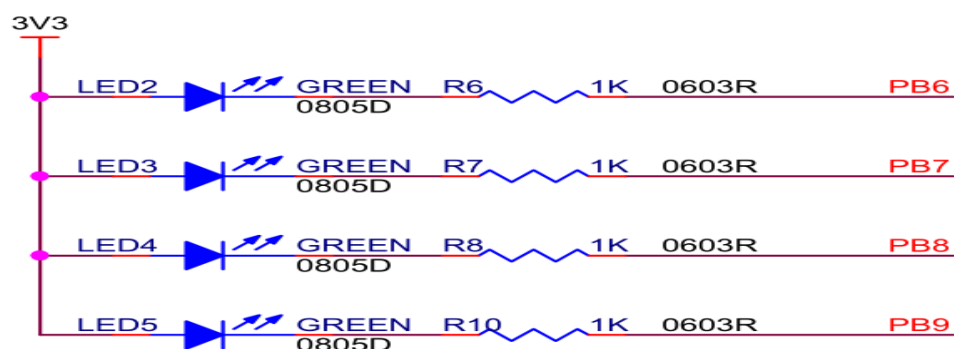
1 LED 流水灯

1.1 例程功能



LED 灯的流水闪烁效果。

1.2 硬件连接



LED 是集成在我们的 MINI 开发板上的，根据 mini 开发板的原理图，我们得知它的 IO 口便可以进行编程。

1.3 程序设计

a) 打开工程

名称	修改日期	类型	大小
core	2017/1/11 18:26	文件夹	
fwlib	2017/1/11 18:22	文件夹	
hardware	2017/1/11 18:22	文件夹	
listing	2017/1/12 11:40	文件夹	
output	2017/1/12 11:40	文件夹	
user	2017/1/11 18:29	文件夹	
keilkill.bat	2016/11/8 8:23	Windows 批处理...	1 KB
stm32f10x-demo.uvgui.admin007	2017/1/12 8:23	ADMIN007 文件	72 KB
stm32f10x-demo.uvopt	2017/1/11 18:35	UVOPT 文件	10 KB
stm32f10x-demo.uvproj	2017/1/11 18:35	Keilvision4 Project	17 KB
stm32f10x-demo_LED.dep	2017/1/12 11:40	DEP 文件	14 KB

b) 初始化硬件

```

31  * 函数名称: Hardware_Init
32  * 函数功能: 硬件初始化
33  * 入口参数: 无
34  * 返回参数: 无
35  * 说明: 初始化单片机功能以及外接设备
36  * 说明: 初始化单片机功能以及外接设备
37  * 说明: 初始化单片机功能以及外接设备
38  * 说明: 初始化单片机功能以及外接设备
39  * 说明: 初始化单片机功能以及外接设备
40  * 说明: 初始化单片机功能以及外接设备
41  * 说明: 初始化单片机功能以及外接设备
42  * 说明: 初始化单片机功能以及外接设备
43  * 说明: 初始化单片机功能以及外接设备
44  * 说明: 初始化单片机功能以及外接设备
45  * 说明: 初始化单片机功能以及外接设备
46  * 说明: 初始化单片机功能以及外接设备
47  * 说明: 初始化单片机功能以及外接设备
48  * 说明: 初始化单片机功能以及外接设备
49  * 说明: 初始化单片机功能以及外接设备
50  * 说明: 初始化单片机功能以及外接设备
51  * 说明: 初始化单片机功能以及外接设备
52  * 说明: 初始化单片机功能以及外接设备
53  * 说明: 初始化单片机功能以及外接设备

```

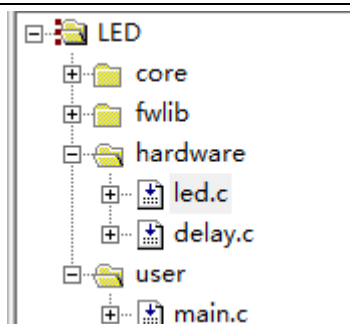
控制 LED 的闪烁需要对接口、延时、中断控制器的分组设置，在执行 main 函数之前，我们先对其初始化。同时我们打开设置初始化程序：

```

35  * 函数名称: Led_Init
36  * 函数功能: LED初始化
37  * 入口参数: 无
38  * 返回参数: 无
39  * 说明: LED4-PB6 LED5-PB7 LED6-PB8 LED7-PB9
40  * 说明: 高电平关灯 低电平开灯
41  * 说明: 高电平关灯 低电平开灯
42  * 说明: 高电平关灯 低电平开灯
43  * 说明: 高电平关灯 低电平开灯
44  * 说明: 高电平关灯 低电平开灯
45  * 说明: 高电平关灯 低电平开灯
46  * 说明: 高电平关灯 低电平开灯
47  * 说明: 高电平关灯 低电平开灯
48  * 说明: 高电平关灯 低电平开灯
49  * 说明: 高电平关灯 低电平开灯
50  * 说明: 高电平关灯 低电平开灯
51  * 说明: 高电平关灯 低电平开灯
52  * 说明: 高电平关灯 低电平开灯
53  * 说明: 高电平关灯 低电平开灯
54  * 说明: 高电平关灯 低电平开灯
55  * 说明: 高电平关灯 低电平开灯
56  * 说明: 高电平关灯 低电平开灯
57  * 说明: 高电平关灯 低电平开灯
58  * 说明: 高电平关灯 低电平开灯
59  * 说明: 高电平关灯 低电平开灯
60  * 说明: 高电平关灯 低电平开灯
61  * 说明: 高电平关灯 低电平开灯
62  * 说明: 高电平关灯 低电平开灯
63  * 说明: 高电平关灯 低电平开灯
64  * 说明: 高电平关灯 低电平开灯
65  * 说明: 高电平关灯 低电平开灯
66  * 说明: 高电平关灯 低电平开灯
67  * 说明: 高电平关灯 低电平开灯
68  * 说明: 高电平关灯 低电平开灯
69  * 说明: 高电平关灯 低电平开灯
70  * 说明: 高电平关灯 低电平开灯
71  * 说明: 高电平关灯 低电平开灯
72  * 说明: 高电平关灯 低电平开灯
73  * 说明: 高电平关灯 低电平开灯
74  * 说明: 高电平关灯 低电平开灯
75  * 说明: 高电平关灯 低电平开灯
76  * 说明: 高电平关灯 低电平开灯
77  * 说明: 高电平关灯 低电平开灯
78  * 说明: 高电平关灯 低电平开灯
79  * 说明: 高电平关灯 低电平开灯
80  * 说明: 高电平关灯 低电平开灯
81  * 说明: 高电平关灯 低电平开灯
82  * 说明: 高电平关灯 低电平开灯
83  * 说明: 高电平关灯 低电平开灯
84  * 说明: 高电平关灯 低电平开灯
85  * 说明: 高电平关灯 低电平开灯
86  * 说明: 高电平关灯 低电平开灯
87  * 说明: 高电平关灯 低电平开灯
88  * 说明: 高电平关灯 低电平开灯
89  * 说明: 高电平关灯 低电平开灯
90  * 说明: 高电平关灯 低电平开灯
91  * 说明: 高电平关灯 低电平开灯
92  * 说明: 高电平关灯 低电平开灯
93  * 说明: 高电平关灯 低电平开灯
94  * 说明: 高电平关灯 低电平开灯
95  * 说明: 高电平关灯 低电平开灯
96  * 说明: 高电平关灯 低电平开灯
97  * 说明: 高电平关灯 低电平开灯
98  * 说明: 高电平关灯 低电平开灯
99  * 说明: 高电平关灯 低电平开灯
100 * 说明: 高电平关灯 低电平开灯

```

LED 初始化的子文件



同理，分别对其他两个子程序进行编写即可。

c) Main 函数

```
int main(void)
{
    Hardware_Init();           //硬件初始化

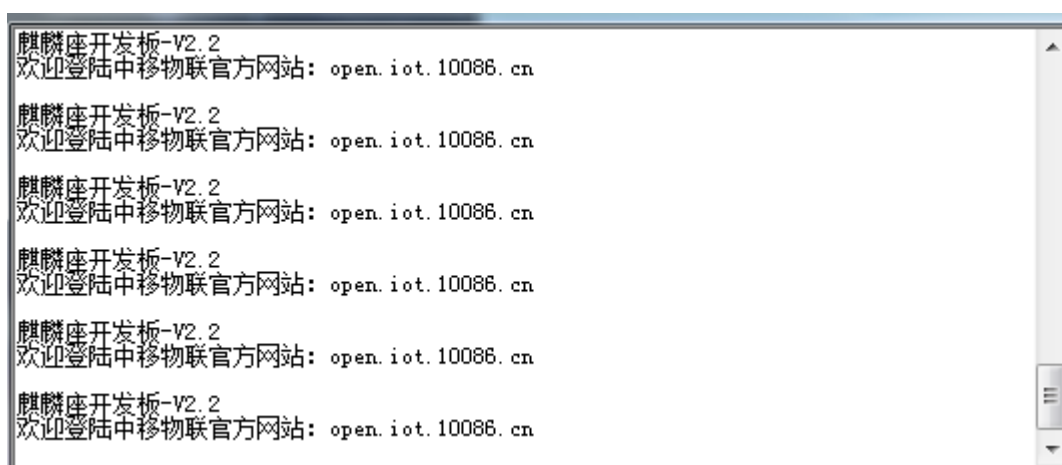
    while(1)
    {
        Led2_Set(LED_ON);DelayXms(500);Led2_Set(LED_OFF); //点亮LED4，并延时500ms，然后熄灭LED4
        Led3_Set(LED_ON);DelayXms(500);Led3_Set(LED_OFF); //点亮LED5，并延时500ms，然后熄灭LED5
        Led4_Set(LED_ON);DelayXms(500);Led4_Set(LED_OFF); //点亮LED6，并延时500ms，然后熄灭LED6
        Led5_Set(LED_ON);DelayXms(500);Led5_Set(LED_OFF); //点亮LED7，并延时500ms，然后熄灭LED7
    }
}
```

Main 函数的内容起始很简短，在硬件初始化完成后，分别让几个 LED 点亮熄灭，在加一个 while（1）死循环就能实现相应效果了。整个例程可分别以下几步：

- 硬件初始化
- 初始化设置
- 循环控制

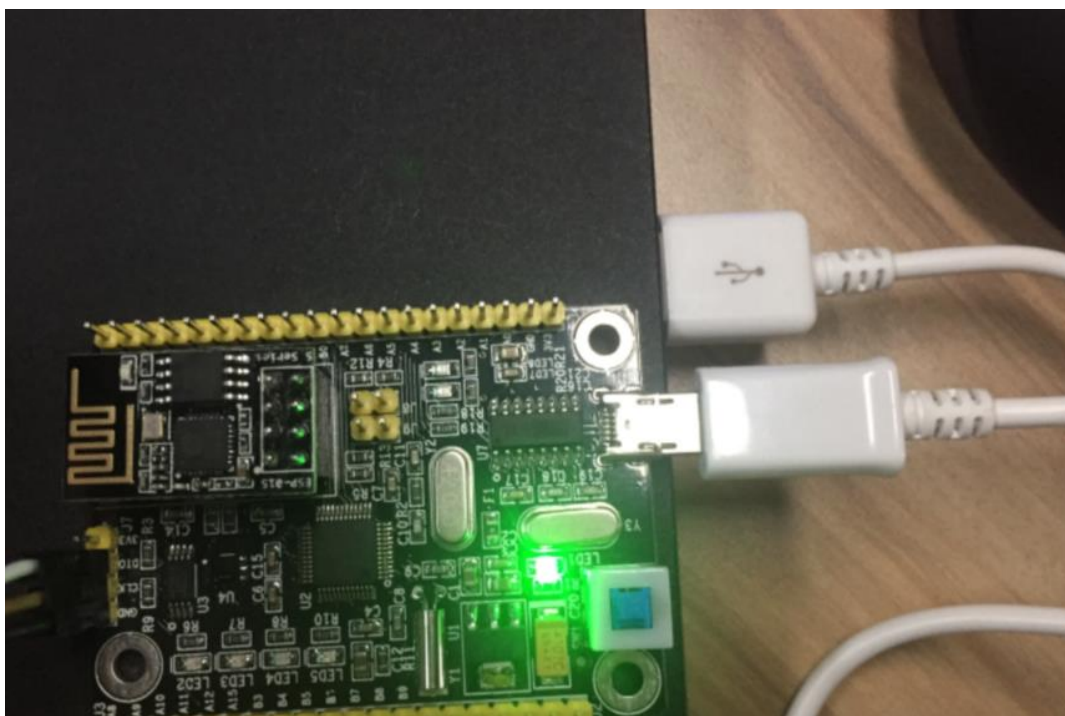
2 串口打印

2.1 例程功能



通过串口的连接。让 MCU 向串口发送指定数据。

2.2 硬件连接



使用通用的 micro USB 数据线与电脑 USB 接口相连，并安装串口驱动与串口助手。

2.3 程序设计

a) 打开工程

名称	修改日期	类型	大小
core	2017/1/11 18:26	文件夹	
fwlib	2017/1/11 18:22	文件夹	
hardware	2017/1/11 18:22	文件夹	
listing	2017/1/12 11:40	文件夹	
output	2017/1/12 11:40	文件夹	
user	2017/1/11 18:29	文件夹	
keilkill.bat	2016/11/8 8:23	Windows 批处理...	1 KB
stm32f10x-demo.uvgui.admin007	2017/1/12 8:23	ADMIN007 文件	72 KB
stm32f10x-demo.uvopt	2017/1/11 18:35	UVOPT 文件	10 KB
stm32f10x-demo.uvproj	2017/1/11 18:35	Keil uVision4 Project	17 KB
stm32f10x-demo_LED.dep	2017/1/12 11:40	DEP 文件	14 KB

b) 初始化硬件

```

30  /*
31  ****
32  * 函数名称:   Hardware_Init
33  *
34  * 函数功能:   硬件初始化
35  *
36  * 入口参数:   无
37  *
38  * 返回参数:   无
39  *
40  * 说明:       初始化单片机功能以及外接设备
41  ****
42  */
43  void Hardware_Init(void)
44  {
45
46      NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); //中断控制器分组设置
47
48      Delay_Init(); //SysTick初始化, 用于普通的延时
49
50      Usart1_Init(115200); //初始化串口1, 波特率115200
51
52  }

```

由于本例程的功能是使用串口发送特定数据, 所以需要初始化串口, 串口的 IO 口我们可以从 mini 开发板的原理图中了解。如下图:



所以在初始化过程中, 我们需要对相应的 I/O 口进行编程。即:

```

46  void Usart1_Init(unsigned int baud)
47  {
48
49      GPIO_InitTypeDef gpioInitStruct;
50      USART_InitTypeDef usartInitStruct;
51      NVIC_InitTypeDef nvicInitStruct;
52
53      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
54      RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
55
56      //PA9 TXD
57      gpioInitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
58      gpioInitStruct.GPIO_Pin = GPIO_Pin_9;
59      gpioInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
60      GPIO_Init(GPIOA, &gpioInitStruct);
61
62      //PA10 RXD
63      gpioInitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
64      gpioInitStruct.GPIO_Pin = GPIO_Pin_10;
65      gpioInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
66      GPIO_Init(GPIOA, &gpioInitStruct);
67
68      usartInitStruct.USART_BaudRate = baud;
69      usartInitStruct.USART_HardwareFlowControl = USART_HardwareFlowControl_None; //无硬件流控
70      usartInitStruct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; //接收和发送
71      usartInitStruct.USART_Parity = USART_Parity_No; //无校验
72      usartInitStruct.USART_StopBits = USART_StopBits_1; //1位停止位
73      usartInitStruct.USART_WordLength = USART_WordLength_8b; //8位数据位
74      USART_Init(USART1, &usartInitStruct);

```

c) Main 函数


```

int main(void)
{
    Hardware_Init();           //硬件初始化

    while(1)
    {
        UsartPrintf(USART1, "\r\n麒麟座开发板-V2.2\r\n"); //打印
        UsartPrintf(USART1, "欢迎登陆中移物联官方网站: open.iot.10086.cn\r\n");
        DelayMs(2500);
    }
}

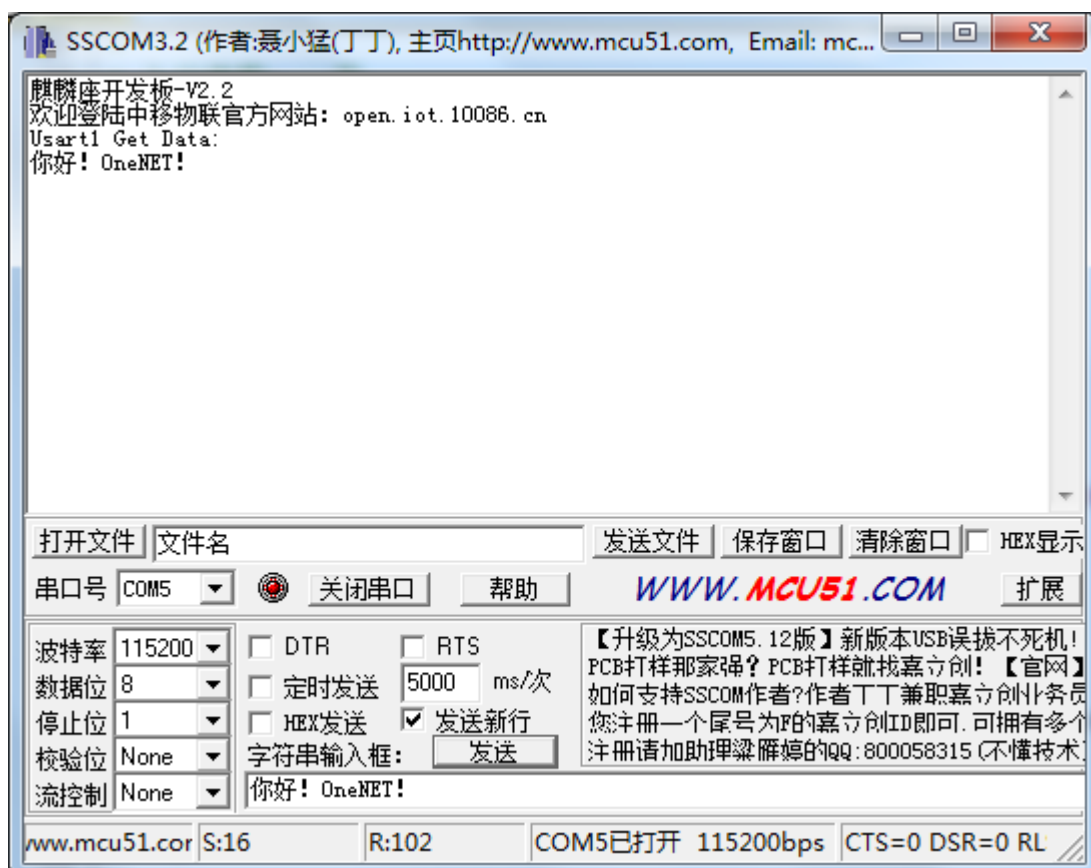
```

Main 函数的结构与例程一几乎相同，即可总结为以下几点：

- 硬件初始化
- 串口一发送数据
- 延时 2500ms
- While 死循环

3 串口打印-接收中断

3.1 例程功能



3.2 硬件连接

同例程二。

3.3 程序设计

a) 打开工程

名称	修改日期	类型	大小
core	2017/1/11 18:26	文件夹	
fwlib	2017/1/11 18:22	文件夹	
hardware	2017/1/11 18:22	文件夹	
listing	2017/1/12 11:40	文件夹	
output	2017/1/12 11:40	文件夹	
user	2017/1/11 18:29	文件夹	
keilkill.bat	2016/11/8 8:23	Windows 批处理...	1 KB
stm32f10x-demo.uvgui.admin007	2017/1/12 8:23	ADMIN007 文件	72 KB
stm32f10x-demo.uvopt	2017/1/11 18:35	UVOPT 文件	10 KB
stm32f10x-demo.uvproj	2017/1/11 18:35	Keil uVision4 Project	17 KB
stm32f10x-demo_LED.dep	2017/1/12 11:40	DEP 文件	14 KB

b) 初始化硬件

```

30  /*
31  *****
32  * 函数名称:   Hardware_Init
33  *
34  * 函数功能:   硬件初始化
35  *
36  * 入口参数:   无
37  *
38  * 返回参数:   无
39  *
40  * 说明:       初始化单片机功能以及外接设备
41  *****
42  */
43  void Hardware_Init(void)
44  {
45
46      NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);    //中断控制器分组设置
47
48      Delay_Init();                                     //SysTick初始化,用于普通的延时
49
50      Usart1_Init(115200);                             //初始化串口1,波特率115200
51
52  }

```

由于本例程的功能是使用串口发送特定数据,所以需要初始化串口,串口的IO口我们可以从 mini 开发板的原理图中了解。如下图:



所以在初始化过程中，我们需要对相应的 I/O 口进行编程。即：

```

46 void Usart1_Init(unsigned int baud)
47 {
48     GPIO_InitTypeDef gpioInitStruct;
49     USART_InitTypeDef usartInitStruct;
50     NVIC_InitTypeDef nvicInitStruct;
51
52     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
53     RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
54
55     //PA9 TXD
56     gpioInitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
57     gpioInitStruct.GPIO_Pin = GPIO_Pin_9;
58     gpioInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
59     GPIO_Init(GPIOA, &gpioInitStruct);
60
61     //PA10 RXD
62     gpioInitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
63     gpioInitStruct.GPIO_Pin = GPIO_Pin_10;
64     gpioInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
65     GPIO_Init(GPIOA, &gpioInitStruct);
66
67     usartInitStruct.USART_BaudRate = baud;
68     usartInitStruct.USART_HardwareFlowControl = USART_HardwareFlowControl_None; //无硬件流控
69     usartInitStruct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; //接收和发送
70     usartInitStruct.USART_Parity = USART_Parity_No; //无校验
71     usartInitStruct.USART_StopBits = USART_StopBits_1; //1位停止位
72     usartInitStruct.USART_WordLength = USART_WordLength_8b; //8位数据位
73     USART_Init(USART1, &usartInitStruct);
74

```

c) Main 函数

```

202 /*
203 *****
204 * 函数名称:  USART1_IRQHandler
205 *
206 * 函数功能:  串口1收发中断
207 *
208 * 入口参数:  无
209 *
210 * 返回参数:  无
211 *
212 * 说明:
213 *****
214 */
215 void USART1_IRQHandler(void)
216 {
217     if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET) //接收中断
218     {
219         if(usart1Len >= 64) //防止数据过多，导致内存溢出
220             usart1Len = 0;
221         usart1Buf[usart1Len++] = USART1->DR;
222         USART_ClearFlag(USART1, USART_FLAG_RXNE);
223     }
224 }
225

```

USART1 接收中断。将接收到的数据一个一个放入一个缓存区，并对缓存区做响应的保护，防止内存溢出导致死机。

```

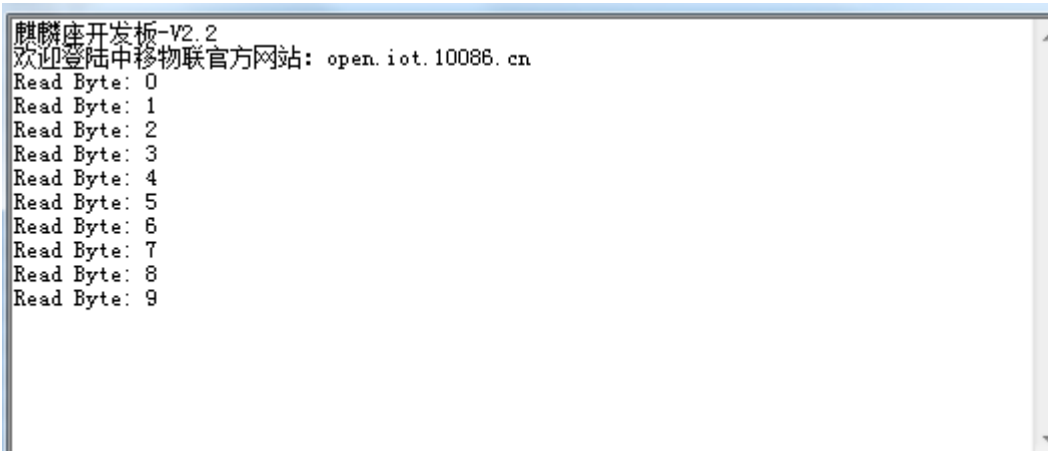
62 /*
63 ****
64 *   函数名称:   main
65 *
66 *   函数功能:   USART1打印数据
67 *
68 *   入口参数:   无
69 *
70 *   返回参数:   0
71 *
72 *   说明:
73 ****
74 */
75 int main(void)
76 {
77
78     Hardware_Init();           //硬件初始化
79
80     UsartPrintf(USART1, "\r\n麒麟座开发板-V2.2\r\n");           //打印
81     UsartPrintf(USART1, "欢迎登陆中移物联官方网站: open.iot.10086.cn\r\n");
82
83     while(1)
84     {
85
86         if(usart1Len > 0)
87         {
88             UsartPrintf(USART1, "Usart1 Get Data: \r\n%s\r\n", usart1Buf);
89
90             memset(usart1Buf, 0, sizeof(usart1Buf));
91             usart1Len = 0;
92         }
93
94         DelayXms(100);
95     }
96 }
97
98 }

```

间隔 100ms 检测一下串口缓存是否有数据，如果有则打印出接收到的数据，并清空接收缓存和接收计数。

4 AT40C20

4.1 例程功能



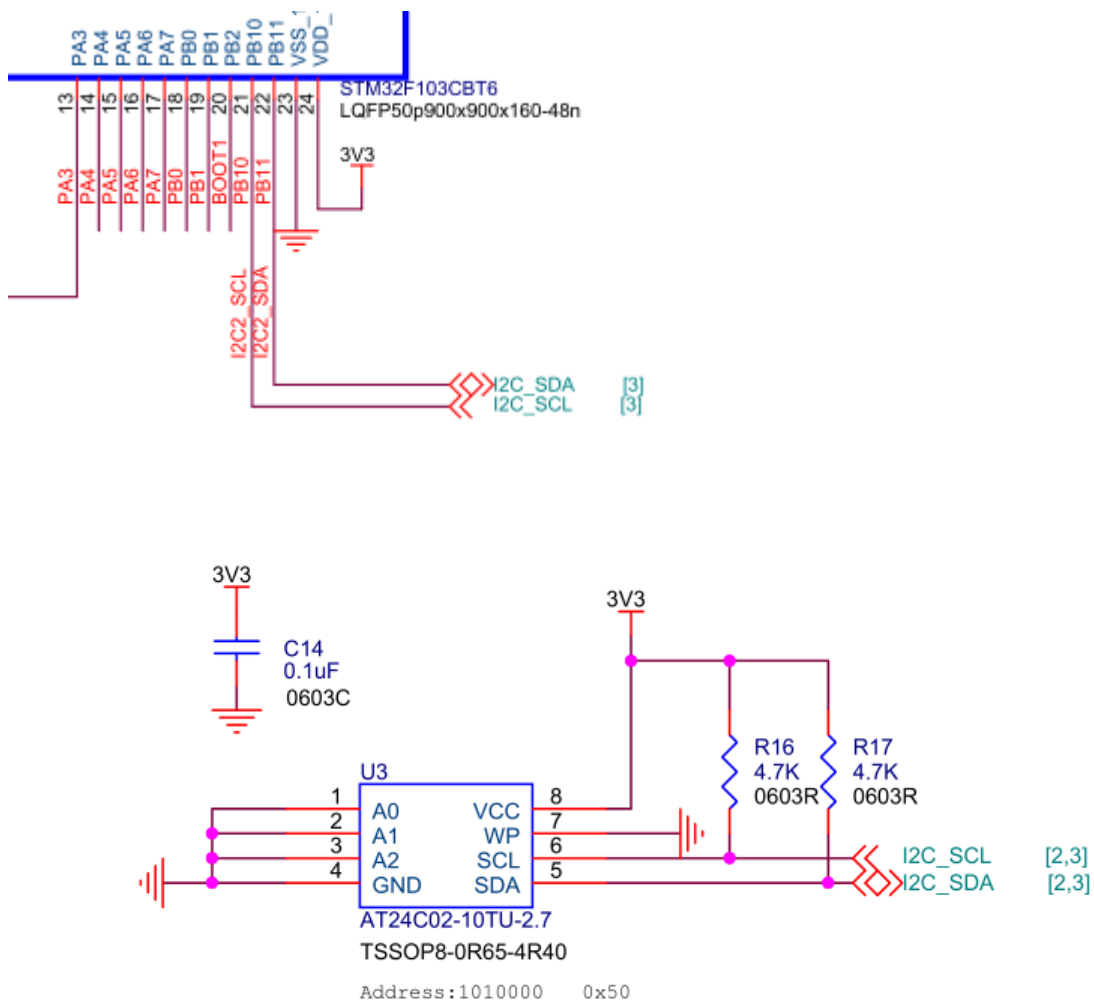
```

麒麟座开发板-V2.2
欢迎登陆中移物联官方网站: open.iot.10086.cn
Read Byte: 0
Read Byte: 1
Read Byte: 2
Read Byte: 3
Read Byte: 4
Read Byte: 5
Read Byte: 6
Read Byte: 7
Read Byte: 8
Read Byte: 9

```

通过串口发送相关数据。

4.2 硬件连接



4.3 程序设计

a) 打开工程

名称	修改日期	类型	大小
core	2017/1/11 18:26	文件夹	
fwlib	2017/1/11 18:22	文件夹	
hardware	2017/1/11 18:22	文件夹	
listing	2017/1/12 11:40	文件夹	
output	2017/1/12 11:40	文件夹	
user	2017/1/11 18:29	文件夹	
keilkill.bat	2016/11/8 8:23	Windows 批处理...	1 KB
stm32f10x-demo.uvgui.admin007	2017/1/12 8:23	ADMIN007 文件	72 KB
stm32f10x-demo.uvopt	2017/1/11 18:35	UVOPT 文件	10 KB
stm32f10x-demo.uvproj	2017/1/11 18:35	Microvision4 Project	17 KB
stm32f10x-demo_LED.dep	2017/1/12 11:40	DEP 文件	14 KB

b) 初始化硬件

```

*****
* 函数名称:   Hardware_Init
*
* 函数功能:   硬件初始化
*
* 入口参数:   无
*
* 返回参数:   无
*
* 说明:       初始化单片机功能以及外接设备
*****
*/
void Hardware_Init(void)
{
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);    //中断控制器分组设置

    Delay_Init();                                     //Systick初始化, 用于普通的延时

    Usart1_Init(115200);                             //初始化usart1, 波特率115200

    IIC_Init();                                       //I2C总线初始化

    UsartPrintf(USART1, "\r\n麒麟座开发板-V2.2\r\n"); //打印
    UsartPrintf(USART1, "欢迎登陆中移物联官方网站: open.iot.10086.cn\r\n");
}

```

这里只需要初始化 IIC 接口就行。

```

55  /*
56  *****
57  * 函数名称:   IIC_Init
58  *
59  * 函数功能:   软件IIC总线IO初始化
60  *
61  * 入口参数:   无
62  *
63  * 返回参数:   无
64  *
65  * 说明:       使用开漏方式, 这样可以不用切换IO口的输入输出方向
66  *****
67  */
68  void IIC_Init(void)
69  {
70
71      GPIO_InitTypeDef gpioInitStruct;
72
73      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //打开GPIOB的时钟
74
75      gpioInitStruct.GPIO_Mode = GPIO_Mode_Out_OD;           //开漏, 这样不用去切换输出输入方向
76      gpioInitStruct.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_11;
77      gpioInitStruct.GPIO_Speed = GPIO_Speed_50MHz;
78      GPIO_Init(GPIOB, &gpioInitStruct);
79
80      IIC_SpeedCtl(2);                                       //IIC速度控制, 单位: 微秒
81
82      SDA_H;                                                 //拉高SDA线, 处于空闲状态
83      SCL_H;                                                 //拉高SCL线, 处于空闲状态
84
85  }

```

将 IIC 接口初始化为 OD 模式, 然后在硬件上外接上拉电阻, 这样对于 stm32 这类型单片机来说, 就不用频繁的切换输入输出方向了。

注意: 所有例程均使用软件 IIC。

c) Main 函数

```

4  int main(void)
5  {
6
7      unsigned char writeByte = 0, readByte = 0;
8
9      Hardware_Init();           //硬件初始化
10
11     while(1)
12     {
13
14         AT24C02_WriteByte(0x00, writeByte++);
15         DelayXms(10);
16         AT24C02_ReadByte(0x00, &readByte);
17
18         UartPrintf(USART1, "Read Byte: %d\r\n", readByte);
19
20         DelayMs(1000);
21     }
22 }

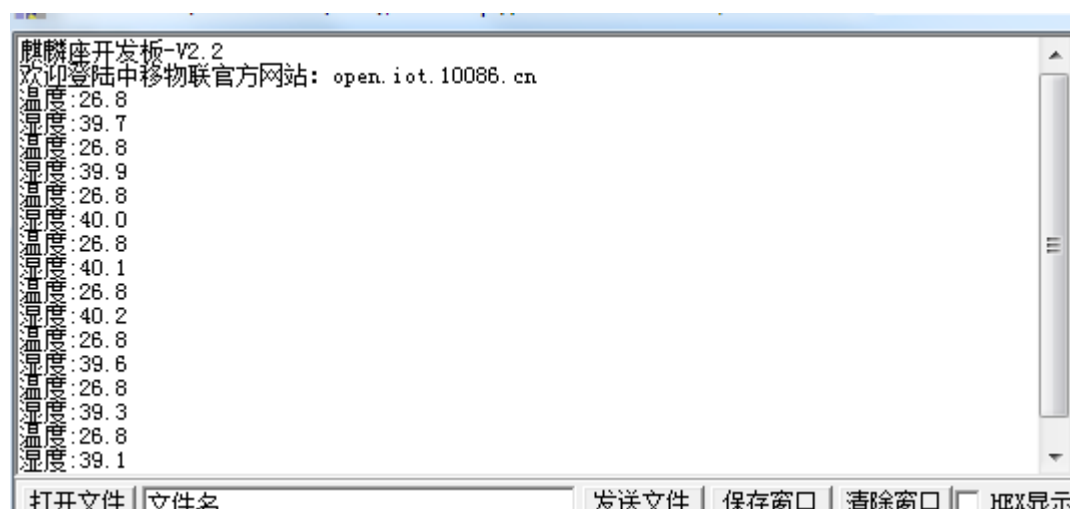
```

使用两个变量，一个作为写，一个作为读。

先将写变量写入 EEPROM 的 0 地址处，然后自增，等待一会，保证写入成功；然后将 EEPROM 的 0 地址处的值读到读变量里，然后分别答应到串口上，然后等待 1s，循环操作

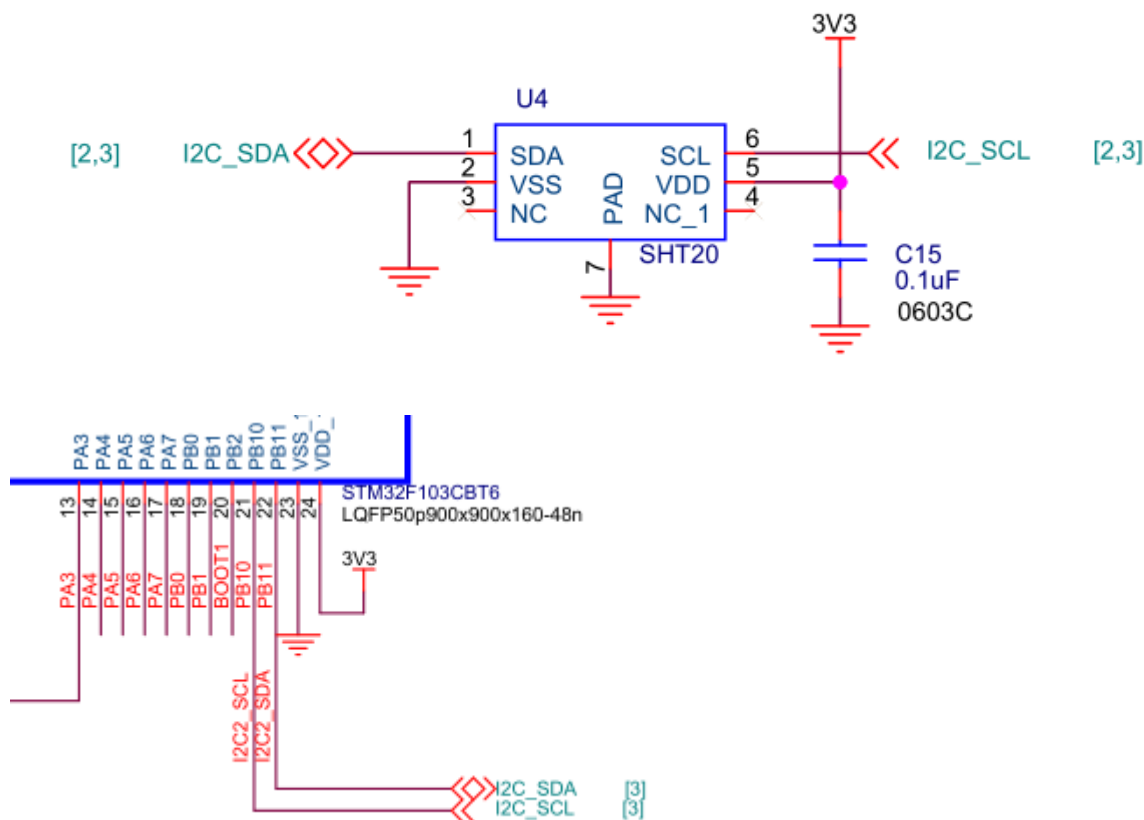
5 SHT20-温湿度传感器

5.1 例程功能



通过串口打印出温湿度。

5.2 硬件连接



5.3 程序设计

a) 打开工程

名称	修改日期	类型	大小
core	2017/1/11 18:26	文件夹	
fwlib	2017/1/11 18:22	文件夹	
hardware	2017/1/11 18:22	文件夹	
listing	2017/1/12 11:40	文件夹	
output	2017/1/12 11:40	文件夹	
user	2017/1/11 18:29	文件夹	
keilkill.bat	2016/11/8 8:23	Windows 批处理...	1 KB
stm32f10x-demo.uvgui.admin007	2017/1/12 8:23	ADMIN007 文件	72 KB
stm32f10x-demo.uvopt	2017/1/11 18:35	UVOPT 文件	10 KB
stm32f10x-demo.uvproj	2017/1/11 18:35	Keil uVision4 Project	17 KB
stm32f10x-demo_LED.dep	2017/1/12 11:40	DEP 文件	14 KB

b) 初始化硬件

```

32  /*
33  ****
34  * 函数名称:   Hardware_Init
35  *
36  * 函数功能:   硬件初始化
37  *
38  * 入口参数:   无
39  *
40  * 返回参数:   无
41  *
42  * 说明:       初始化单片机功能以及外接设备
43  ****
44  */
45  void Hardware_Init(void)
46  {
47
48      NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);    //中断控制器分组设置
49
50      Delay_Init();                                     //SysTick初始化, 用于普通的延时
51
52      Usart1_Init(115200);                             //初始化usart1, 波特率115200
53
54      IIC_Init();                                       //I2C总线初始化
55
56      UsartPrintf(USART1, "\r\n麒麟座开发板-V2.2\r\n"); //打印
57      UsartPrintf(USART1, "欢迎登陆中移物联官方网站: open.iot.10086.cn\r\n");
58
59  }

```

SHT20 不需要进行初始化, 初始化 IIC 接口即可。

c) Main 函数

```

int main(void)
{
    Hardware_Init();                                   //硬件初始化

    while(1)
    {
        SHT20_GetValue();

        UsartPrintf(USART1, "温度:%0.1f\r\n湿度:%0.1f\r\n", sht20Info.tempreture, sht20Info.humidity);

        DelayMs(200);
    }
}

```

通过 SHT20 读取出温湿度, 直接通过串口打印即可。