第3章

蓝牙(Bluetooth)通信

3.1 项目任务

在本项目中要完成以下任务。

(1) Bluetooth 通信硬件模块及接口分析；

(2) Bluetooth 通信软件程序及接口分析；

(3) Bluetooth 主从通信模式设计；

(4) 通信速率及通道设计。

具体任务指标如下：

完成基于 Bluetooth 无线通信模式下的传感器采集数据通信应用系统。

3.2 项目的提出

“基于 Bluetooth 无线通信模式下的传感器采集数据通信应用系统”是以 Bluetooth 通信为基础，采用 32 位高性能低功耗的 STM32F103C8 处理器为核心处理器，其上位机 Windows 开发环境使用的是嵌入式集成开发环境 IAR Embedded Workbench for ARM 5.41，采用蓝牙模块 BF10-I Bluetooth Module 与 STM32 硬件连接，实现数据的串口通信。

3.3　实施项目的预备知识

预备知识的重点内容：

(1) 理解 Bluetooth 技术的概念、技术特点；

(2) 了解 Bluetooth 的设备类型、硬件的连接使用；

(3) 重点掌握使用 IAR 开发环境设计程序，实现蓝牙主从设备连接，获取传感器数据。

关键术语：

Bluetooth 技术[①]：作为一种短距离无线技术标准，具有开放的技术规范，可实现固定设备、移动设备和楼宇个人域网之间的短距离的无线语音和数据通信(使用 2.4～2.485GHz 的 ISM 波段的 UHF 无线电波)。蓝牙技术最初由电信巨头爱立信公司于 1994 年创制，当时是作为 RS-232 数据线的替代方案。蓝牙可连接多个设备，克服了数据同步的难题。

微微网(Piconet)是由采用蓝牙技术的设备以特定方式组成的网络。微微网的建立是由两台设备(如便携式电脑和蜂窝电话)的连接开始的，最多由 8 台设备构成。所有的蓝牙设备都是对等的，以同样的方式工作。然而，当一个微微网建立时，只有一台为主设备，其他均为从设备，而且在一个微微网存在期间将一直维持这一状况。

分布式网络(Scatternet)是由多个独立、非同步的微微网形成的。

主设备(Master Unit)是指在微微网中，如果某台设备的时钟和跳频序列用于同步其他设备，则称它为主设备。从设备(Slave Unit)是指非主设备的设备均为从设备。

MAC 地址(MAC Address)是用 3 比特表示的地址，用于区分微微网中的设备。休眠设备(Parked Units)在微微网中只参与同步，但没有 MAC 地址的设备。监听及保持方式(Sniff and Hold Mode)指微微网中从设备的两种低功耗工作方式。

预备知识的内容结构：

- 蓝牙(Bluetooth)通信
  - 蓝牙(Bluetooth)技术概述
    - 蓝牙(Bluetooth)技术起源
    - 蓝牙(Bluetooth)技术的特点
  - 软件安装——嵌入式开发环境 IAR EWSTM8 1.30 安装
  - 蓝牙(Bluetooth)硬件技术
    - 蓝牙模块 BF10-I
    - STM32 模块与蓝牙模块的连接原理

① 引自 http://wenku.baidu.com/view/ba5b2fde6529647d272852a0.html?from=search

预备知识：

3.3.1 蓝牙(Bluetooth)技术概述

1. 蓝牙(Bluetooth)技术起源

Bluetooth 取自 10 世纪丹麦国王 Harald Bluetooth 的名字。它孕育着颇为神奇的前景：对手机而言，与耳机之间不再需要连线；在个人计算机，主机与键盘、显示器和打印机之间可以摆脱纷乱的连线；在更大范围内，电冰箱、微波炉和其他家用电器可以与计算机网络连接，实现智能化操作。

1994 年，爱立信移动通信公司开始研究在移动电话及其附件之间实现低功耗、低成本无线接口的可行性。随着项目的进展，爱立信公司意识到短距无线通信(Short Distance Wireless Communication)的应用前景无限广阔。爱立信将这项新的无线通信技术命名为蓝牙(Bluetooth)。

1998 年 5 月，爱立信联合诺基亚(Nokia)、英特尔(Intel)、IBM、东芝(Toshiba)这 4 家公司一起成立了蓝牙特殊利益集团(Special Interest Group，SIG)，负责蓝牙技术标准的制定、产品测试，并协调各国蓝牙的具体使用。

2. 蓝牙(Bluetooth)技术的特点

蓝牙作为一种短距无线通信的技术规范，最初的目标是取代现有的掌上电脑、移动电话等各种数字设备上的有线电缆连接。

在制定蓝牙规范之初，就建立了统一全球的目标，向全球公开发布，工作频段为全球统一开放的 2.4GHz 工业、科学和医学(Industrial，Scientific and Medical，ISM)频段。

从目前的应用来看，由于蓝牙体积小、功率低，其应用已不局限于计算机外设，几乎可以被集成到任何数字设备之中，特别是那些对数据传输速率要求不高的移动设备和便携设备。

蓝牙技术的特点可归纳为如下几点。

(1) 全球范围适用，蓝牙工作在 2.4GHz 的 ISM 频段，全球大多数国家 ISM 频段的范围是 2.4～2.4835GHz，使用该频段无须向各国的无线电资源管理部门申请许可证。

(2) 同时可传输语音和数据：蓝牙采用电路交换和分组交换技术，支持异步数据信道、三路语音信道以及异步数据与同步语音同时传输的信道。每个语音信道数据速率为 64kb/s，语音信号编码采用脉冲编码调制(PCM)或连续可变斜率增量调制(CVSD)方法。当采用非对称信道传输数据时，速率最高为 721kb/s，反向为 57.6kb/s；当采用对称信道传输数据时，速率最高为 342.6kb/s。蓝牙有两种链路类型：异步无连接(ACL)链路和同步面向连接(SCO)链路。

(3) 可以建立临时性的对等连接：根据蓝牙设备在网络中的角色，可分为主设备(Master)与从设备(Slave)。主设备是组网连接主动发起连接请求的蓝牙设备，几个蓝牙设备连接成一个皮网(Piconet)时，其中只有一个主设备，其余的均为从设备。皮

网是蓝牙最基本的一种网络形式，最简单的皮网是一个主设备和一个从设备组成的点对点的通信连接。

(4) 具有很好的抗干扰能力。工作在 ISM 频段的无线电设备有很多种，如家用微波炉、无线局域网(WLAN)Home RF 等产品，为了很好地抵抗来自这些设备的干扰，蓝牙采用了跳频(Frequency Hopping)方式来扩展频谱(Spread Spectrum)，将 2.402～2.48GHz 频段分成 79 个频点，相邻频点间隔 1MHz。蓝牙设备在某个频点发送数据之后，再跳到另一个频点发送，而频点的排列顺序则是伪随机的，每秒钟频率改变 1600 次，每个频率持续 625μs。

(5) 蓝牙模块体积很小、便于集成。由于个人移动设备的体积较小，嵌入其内部的蓝牙模块体积就应该更小，如爱立信公司的蓝牙模块 ROK101008 的外形尺寸仅为 32.8mm×16.8mm×2.95mm。

(6) 低功耗。蓝牙设备在通信连接(Connection)状态下，有 4 种工作模式：激活 Active 模式；呼吸 Sniff 模式；保持 Hold 模式；休眠 Park 模式。Active 模式是正常的工作状态，另外三种模式是为了节能所规定的低功耗模式。

(7) 开放的接口标准。SIG 为了推广蓝牙技术的使用，将蓝牙的技术标准全部公开，全世界范围内的任何单位和个人都可以进行蓝牙产品的开发，只要最终通过 SIG 的蓝牙产品兼容性测试，就可以推向市场。

(8) 成本低。随着市场需求的扩大，各个供应商纷纷推出自己的蓝牙芯片和模块，蓝牙产品价格飞速下降。

3.3.2　软件安装

1. 嵌入式集成开发环境 IAR EWARM 安装

嵌入式开发环境 IAR Embedded Workbench for ARM 5.41 安装：IAR Embedded Workbench for ARM 5.41 是上位机 Windows 的嵌入式集成开发环境，该开发环境针对目标处理器集成了良好的函数库和工具支持，其安装过程如下。

(1) 打开 IAR 安装包进入安装界面，如图 3.1 所示。

图 3.1　打开安装包

(2) 选择 Install IAR Embedded Workbench 安装选项,如图 3.2 和图 3.3 所示。

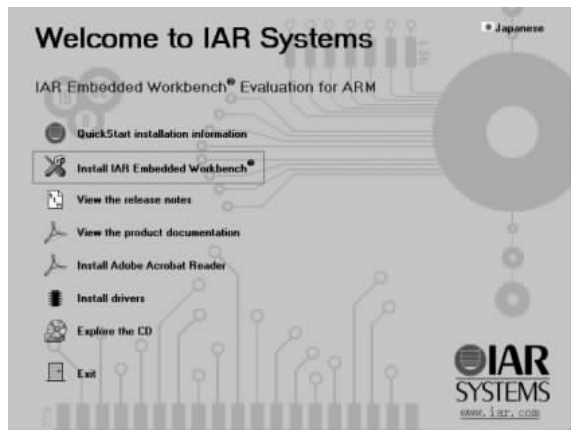

图 3.2 安装 IAR(1)

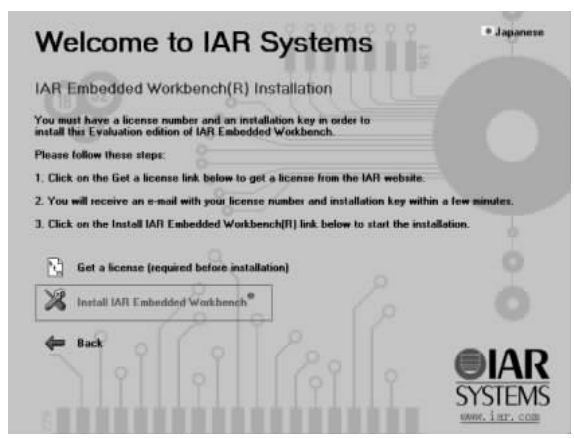

图 3.3 安装 IAR(2)

(3) 进入 IAR 安装过程,单击 Next 按钮,如图 3.4 所示。

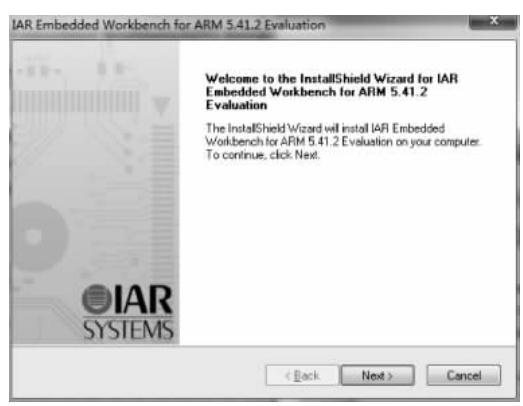

图 3.4 安装 IAR(3)

(4) 进入 License 输入界面,如图 3.5 所示。

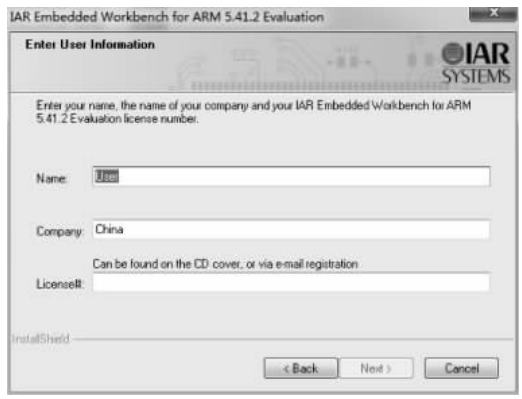

图 3.5　输入用户信息和 License

(5) 输入 Key,如图 3.6 所示。

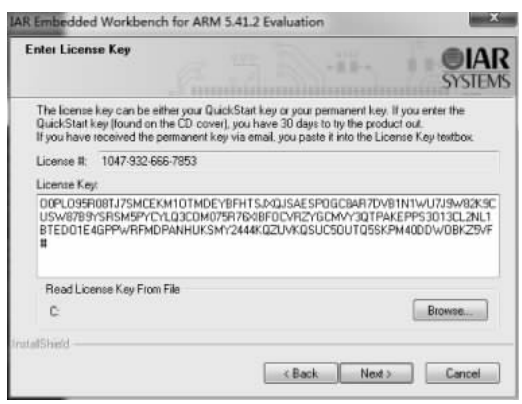

图 3.6　输入 Key

(6) 选择 IAR 软件安装路径,这里选择默认的 C 盘 Program Files,建议默认安装,如图 3.7 所示。

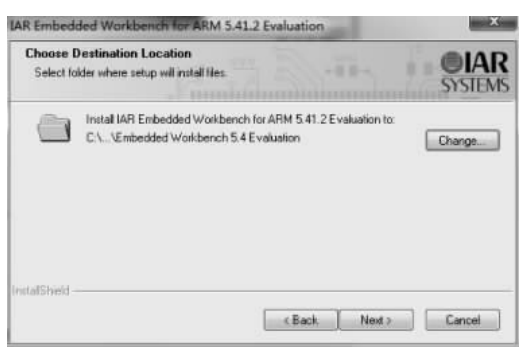

图 3.7　安装到 C 盘默认路径

(7) 进入安装过程界面,如图 3.8 所示。

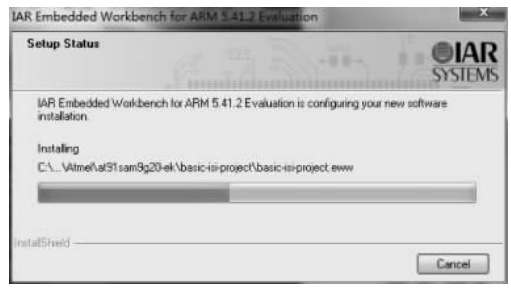

图 3.8 开始安装

(8) 安装完成,如图 3.9 所示。

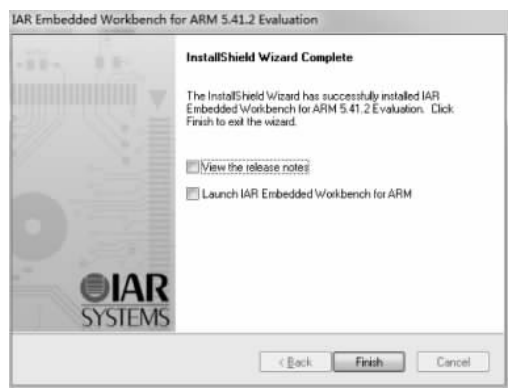

图 3.9 安装完成

2. Jlink 4.20 驱动程序安装过程

(1) 运行安装程序 Setup_JLinkARM_V420p 安装包,并单击 Yes 按钮,如图 3.10 所示。

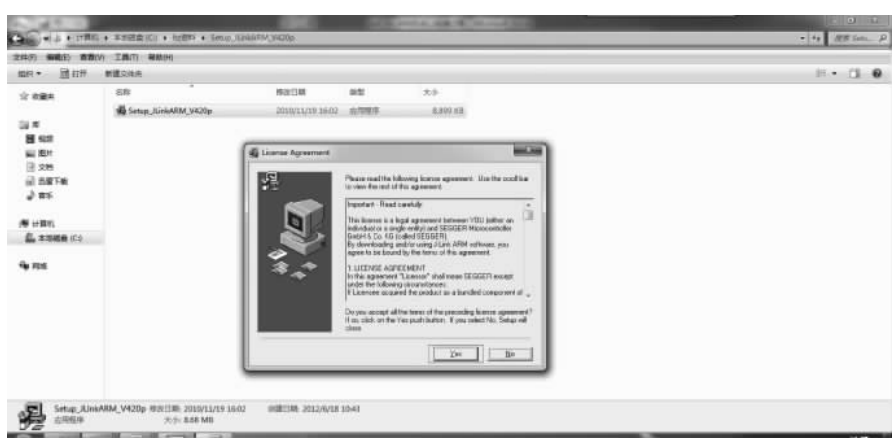

图 3.10 安装仿真器驱动

(2) 单击 Next 按钮,继续安装过程,如图 3.11 所示。

图 3.11　安装下一步

(3) 选择驱动安装路径,单击 Next 按钮,如图 3.12 所示。

图 3.12　默认安装路径

(4) 选择在桌面创建快捷方式,单击 Next 按钮,如图 3.13 所示。

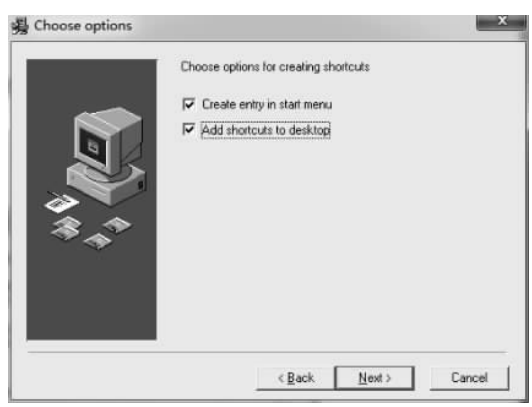

图 3.13　创建图标

(5) 进入安装状态,如图 3.14 所示。

图 3.14 开始安装

(6) 进入 SEGGER J-Link DLL Updater V4.20p 界面,勾选相应的 IAR 版本,单击 Next 按钮,如图 3.15 所示。

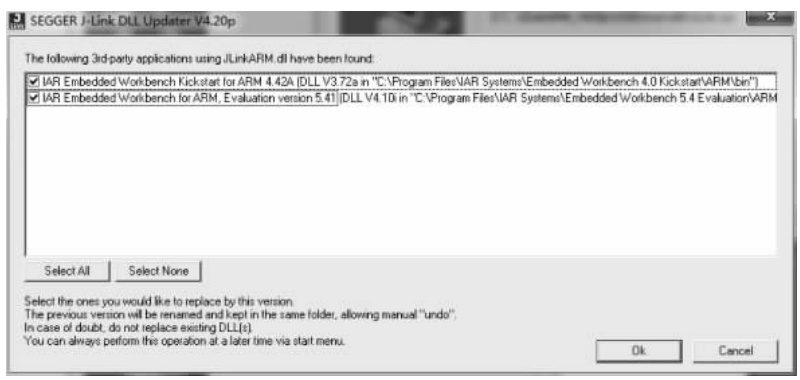

图 3.15 选择第三方环境支持

(7) 完成 Jlink 驱动程序安装,如图 3.16 所示。

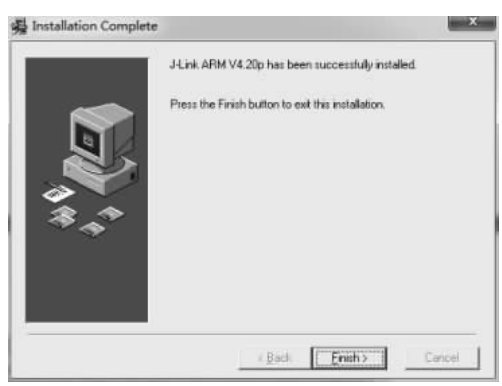

图 3.16 完成安装

3. Emberznet-4.3.0 协议栈安装过程

(1) 运行 EmberZnet 协议栈安装程序，如图 3.17 所示。

图 3.17　协议栈安装包

(2) 进入安装界面，单击 Next 按钮，如图 3.18 所示。接受协议，如图 3.19 所示。

图 3.18　安装

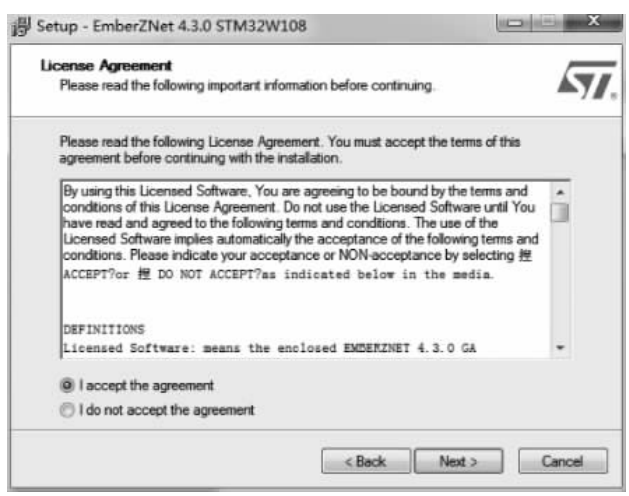

图 3.19　接受协议

(3) 选择安装路径，这里选择默认 C 盘 Program Files 文件夹下，并单击 Next 按钮，如图 3.20～图 3.22 所示。

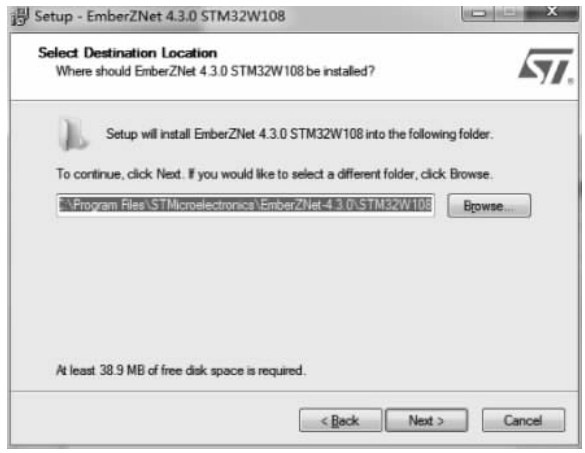

图 3.20 默认 C 盘安装路径

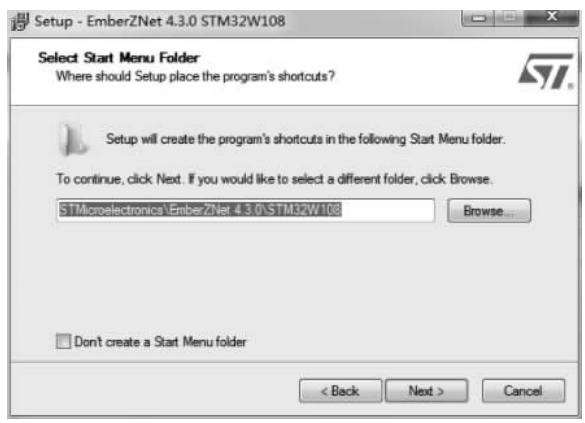

图 3.21 安装父目录

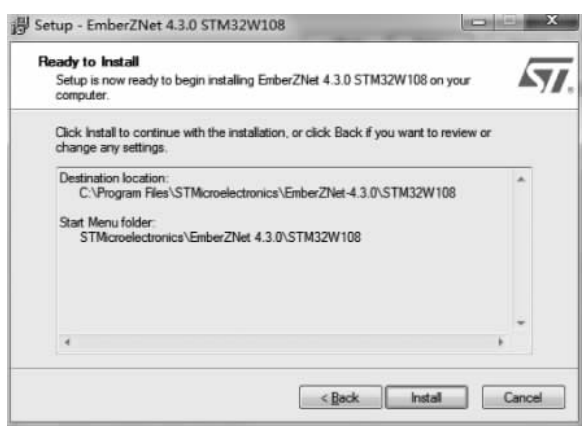

图 3.22 开始安装

(4) 完成安装,如图 3.23 所示。

图 3.23　完成安装

3.3.3　蓝牙(Bluetooth)硬件技术

蓝牙(Bluetooth)模块由 STM32 处理器和 BF10-I(蓝牙通信模块)两个模块搭建而成,RFID 射频卡采用 MF1S50。具体工作原理如下。

1. 蓝牙模块 BF10-I

BF10-I(蓝牙通信模块)是一款智能型的无线数据传输产品,支持 64 通道蓝牙替代串口线;具有 1200～1 382 400b/s 等多种接口波特率;支持主从模式,灵活用在不同领域;SPP 蓝牙串行服务,非常方便和手机、PC 等连接;通过蓝牙模块的串口管脚 UART_TX 和 UART_RX,发送 AT 指令即可完成对蓝牙模块的配置和控制。电路接口如图 3.24 所示。

该模块主要用于短距离的数据无线传输领域。可以方便地和 PC(PDA 手机)的蓝牙设备相连,也可以两个模块之间的数据互通,避免烦琐的线缆连接,能直接替代现有的串口线。

替代串口线透明数据需要两个 BF10-I 模块,一个模块工作在主模式下,一个模块工作在从模式下。当两模块设置为相同的波特率,相同的通道(不能为通道 64),上电之后,主从模块则自动连接形成串口透明。此时的数据传输则是全双工的。串口线透明数据模式应用原理如图 3.25 所示。①设置主模块的 PIO0 为高或悬空,从模块的 PIO0 为低。②设置两个模块的 PIO2～PIO5 高低到对应的波特率,具体参考设置串口通信波特率。③设置两个模块的 PIO6～PIO11 相同的通道,不能为通道 6、4(即全高电平)。具体参考设置模块通道。④模块上电,主模块则自动去查找该通道的从模块,此时主模块和从模块的 PIO1 脚都是输出为高低脉冲。若连接成功之后,主从模

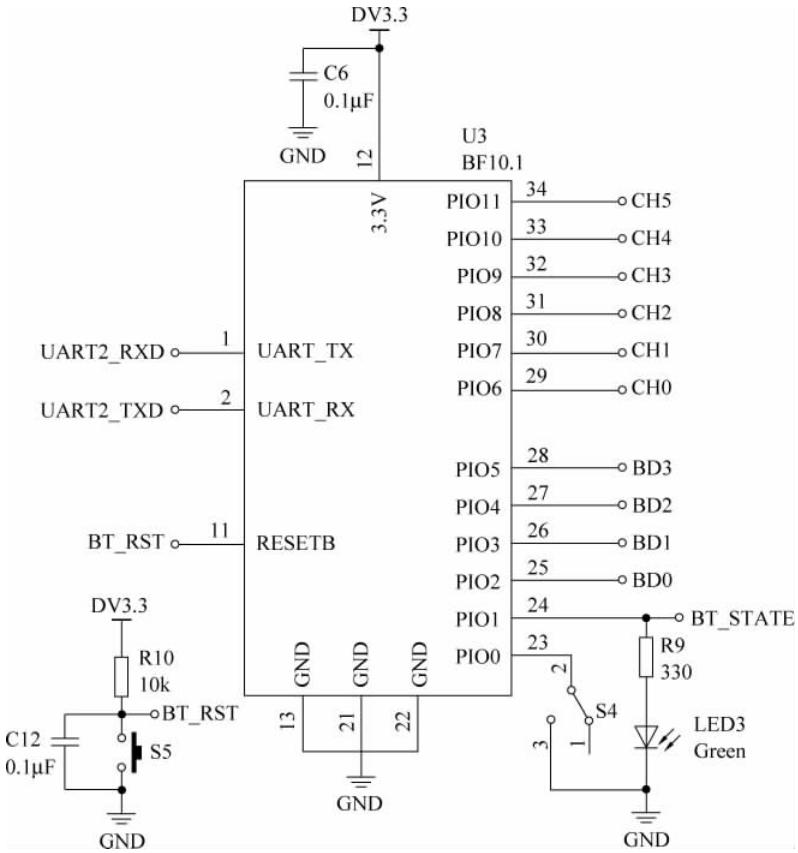

图 3.24 BF10-I(蓝牙通信模块)电路接口

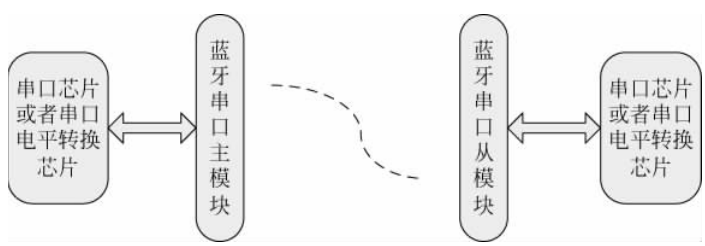

图 3.25 串口线透明数据模式

块的 PIO1 管脚输出为高电平。可以连接一个 LED 进行显示状态。⑤连接成功之后,两个模块两端就能进行串口数据全双工通信了。

从客户端模式是用在被计算机的蓝牙适配器、PDA、手机等通用蓝牙设备连接进行数据传输的情况。操作方式如下:①将 PIO0 接地,设置为从模式;②将 PIO6～PIO11 悬空或者置高,设置为 64 通道;③设置 PIO2～PIO5 为对应需要的波特率;④给模块上电,等待 PC 蓝牙适配器、PDA 等主机设备连接该模块;⑤连接成功后,

PIO1 脚都是输出为高低脉冲。若连接成功之后，PIO1 管脚输出为高电平，可以连接一个 LED 进行显示状态。

2. STM32 模块与蓝牙模块的连接原理

STM32 的 UART2 与蓝牙模块的串口管脚相连接，STM32 硬件接口方式如图 3.26 所示。其中，PA9、PA10、PA2、PA3 蓝牙模块的串口管脚 UART_TX 和 UART_RX，发送 AT 指令即可完成对蓝牙模块的配置和控制。

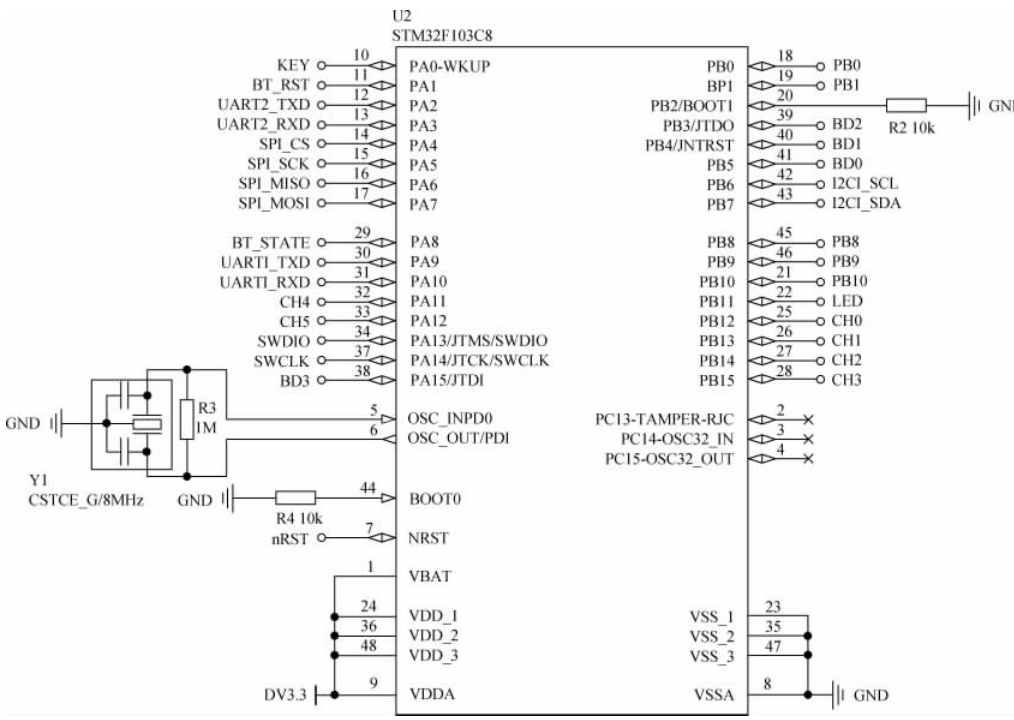

图 3.26　STM32 硬件接口方式

查询/设置串口工作波特率 AT 指令，如表 3.1 所示。

表 3.1　查询/设置波特率 AT 指令

指　令	应　答	参　数
查询：AT+BAUD	OK+Get:[para1]	Para1：0～8 0=9600；1=19200； 2=38400；3=57600； 4=115200；5=4800； 6=2400；7=1200； 8=230400； Defaut：0(9600)
设置：AT+BAUD[para1]	OK+Set:[para1]	

查询/设置模块主从模式指令,如表 3.2 所示。

表 3.2 查询/设置主从模式 AT 指令

指　令	应　答	参　数
查询：AT+ROLE?	OK+Get:[para1]	Para1:0～1 1:主设备 0:从设备 Default:0
设置：AT+ROLE[para1]	OK+Set:[para1]	

查询/设置主模式下搜索时是否仅搜索 HM 模块,如表 3.3 所示。

表 3.3 查询/设置主模式下搜索时是否仅搜索 HM 模块

指　令	应　答	参　数
查询：AT+FILT?	OK+Get:[Para]	无
设置：AT+FILT[Para]	OK+Set:[Para]	Para：0～1 0:搜索所有 BLE 从设备 1:仅搜索 HM 模块

3.4 项目实施

3.4.1 任务 1：Bluetooth 组网配置

利用 STM32 处理器和 BF10-I 模块两片芯片构成蓝牙组网设备,完成主从蓝牙模块的透明传输。

替代串口线透明数据需要两个蓝牙模块,一个模块工作在主模式下,一个模块工作在从模式下。当两模块设置为相同的波特率,上电之后,主从模块则自动连接形成串口透明。此时的数据传输则是全双工的。①发送 AT 指令,设置主、从模块相同的波特率。②发送 AT 指令,设置主模块为主模式,从模块为从模式。③发送 AT 指令,设置主模块为搜索所有从设备。④模块上电,主模块则自动去查找该通道的从模块,此时主模块和从模块的 PIO1 脚都是输出为高低脉冲。若连接成功之后,主从模块的 PIO1 管脚输出为高电平,连接一个 BT LED 进行显示状态。⑤连接成功之后,两个模块两端就能进行串口数据全双工通信了。

1. 源码实现

蓝牙主机主函数：

```
int main(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    u8 i = 0;
    NVIC_Configuration();
```

```
    CLI();
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, DISABLE);

    //JTAG_Remap
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_
APB2Periph_AFIO, ENABLE);
    //JTAG-DP Disabled and SW-DP Enabled
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
    Systick_Init(72);

    LED_Init();
    LED_USER_On();

    UART1_Configuration();
    UART2_Configuration();

    //1ms 中断
    TIM3_Configuration();
    BT_Init();
    BT_Reset();
    delay_ms(5000);
    for(i = 0; i < 26;i++)
        tx_buf[i] = 0;
    for(i = 0; i < 14;i++)
        rx_buf[i] = 0;
    tx_buf_init();
    BT_State = 0;
    BT_Cnt = 0;
    Uart_RecvFlag = 0;
    rx_counter = 0;
    Uart_RecvFlag1 = 0;
    rx_counter1 = 0;
    /* Open Global Interrupt */
    SEI();
    AT_Cmd();
    while(1)
    {
    }

}
```

发送 AT 指令函数：

```
void AT(char * str,int len)
{
  at_len = len;
  AT_FLAG = 0;
  UART2_PutString(str);
  while(!AT_FLAG);
}
```

蓝牙主模块配置函数：

```
void AT_Cmd(void)
{
  Uart_RecvFlag = 1;
  disc = 0;
        / ******** AT CONFIG ********* /
   AT("AT + BAUD0",8);                    //设置波特率为 9600
   AT("AT + IMME0", 8);                   //上电即复位
   AT("AT + ROLE1", 8);                   //查询 设置主模式
   AT("AT + FILT0",8);                    //搜索所有从设备
  UART2_SendString("AT + RESET", 8);      //重启
  delay_ms(5000);
}
```

蓝牙从机主函数：

```
int main(void)
{

    GPIO_InitTypeDef GPIO_InitStructure;
    u8 i = 0;
    NVIC_Configuration();
    CLI();
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, DISABLE);

    //JTAG_Remap
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_
APB2Periph_AFIO, ENABLE);
    //JTAG-DP Disabled and SW-DP Enabled
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);

    Systick_Init(72);
    LED_Init();
    LED_USER_On();
    UART1_Configuration();
    UART2_Configuration();

    //1ms 中断
    TIM3_Configuration();

    BT_Init();
    BT_Reset();
    delay_ms(5000);

    BT_State = 0;
    BT_Cnt = 0;
    Uart_RecvFlag = 0;
    rx_counter = 0;

    Uart_RecvFlag1 = 1;
    rx_counter1 = 0;

    /* Open Global Interrupt */
    SEI();
    /********** AT CONFIG **********/
    AT_Cmd();
    Uart_RecvFlag1 = 0;
   /*************************************/
     while(1)
    {
    }
}
```

蓝牙从模块配置函数：

```
void AT_Cmd(void)
{
  AT("AT+BAUD0",8);                    //设置波特率为 9600
  AT("AT+IMME0", 8);                   //上电即复位

  AT("AT+ROLE0", 8);                   //查询 设置从模式
  BT_Reset();                          //重启
  delay_ms(5000);
}
```

2. 硬件连接

用 J-Link 连接 PC 与实验箱,用实验箱配套的电源给实验箱供电,并给模块上电。

3. 建立工程,编译运行

在 IAR Embedded Workbench for ARM 5.41 软件环境中,打开工程,将工程进行编译。具体方法可以选择 Project 中的 Rebuild All 或者选中工程栏中的工程文件然后右击选择 Rebuild All 进行编译,如图 3.27 所示。

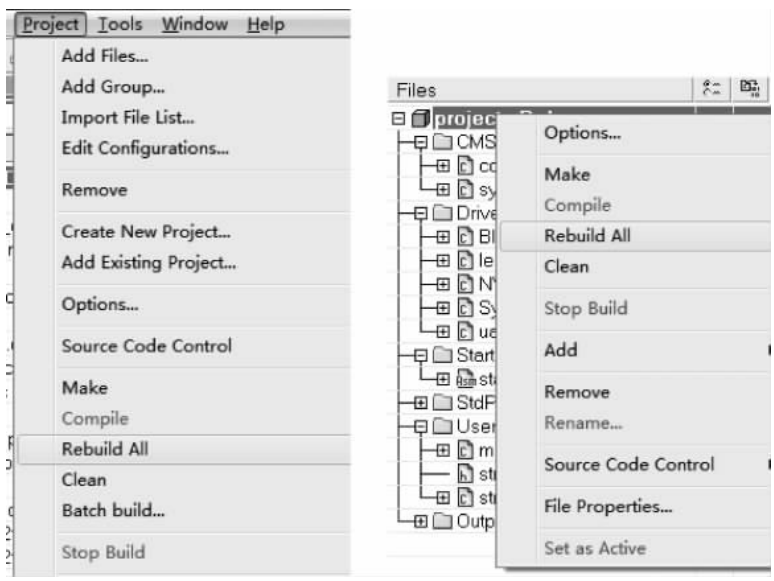

图 3.27 IAR Embedded Workbench for ARM 5.41 软件环境

用板载的“+”、“-”按键分别选中主、从机模块,将 Master 和 Slaver 程序分别烧录到蓝牙主、从机模块里,并重启模块或者使用 RST 键复位模块。

4. 通信测试

观察设备上的 BT LED 灯的情况,判断两个蓝牙模块是否连接,若 BT LED 指示灯长亮,则表示连接成功,反之连接不成功。因为蓝牙主模块设置的是搜索所有从模块,如果同时打开多个蓝牙从模块,主模块会随机与其中一个进行连接,建议使用的时候只打开一个蓝牙从模块。

3.4.2 任务 2:基于 Bluetooth 无线通信传感器采集数据通信设计

BF10-I 蓝牙模块通过 AT 指令设置相同的波特率,设置成透传模式,分别设置主从模块,主从模块连接成功后,即可进行两个模块的通信。通过蓝牙模块,获取传感器的数据。完成基于 Bluetooth 无线通信模式下的传感器采集数据通信应用系统的设计。

蓝牙主模块与蓝牙从模块相连接时，传感器模块将采集到的数据经过处理后通过串口发送给 STM32，STM32 从串口 1 接到传感器的数据经过处理串口 2 发送给蓝牙从模块，蓝牙模块通过无线的方式将数据发送给蓝牙主模块，蓝牙主模块间接到的数据通过串口将数据发送给 STM32，STM32 将数据通过串口 2 接收蓝牙主模块数据，经过处理后由串口 1 发出，如图 3.28 所示。

蓝牙模块已经内置应用程序，用户只需要通过蓝牙模块的串口管脚 UART_TX 和 UART_RX，发送 AT 指令即可完成对蓝牙模块的配置和控制。

上一个项目介绍了蓝牙模块和 STM32 连接的电路图，及 STM32 的 UART2 与蓝牙模块的串口管脚连接方法。

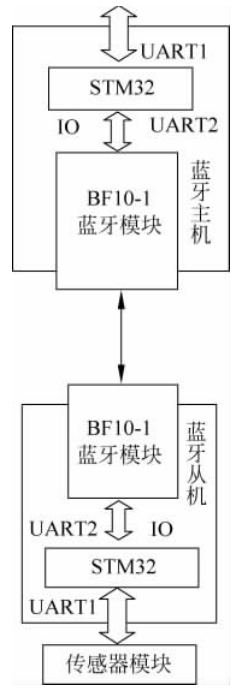

图 3.28　蓝牙传感器工作原理图

1. 软件工作原理

查询/设置串口工作波特率 AT 指令如表 3.4 所示。

表 3.4　查询/设置波特率 AT 指令

指　令	应　答	参　数
查询：AT+BAUD	OK+Get:[para1]	Para1：0～8 0=9600；1=19200； 2=38400；3=57600； 4=115200；5=4800； 6=2400；7=1200； 8=230400； Defaut：0(9600)
设置：AT+BAUD[para1]	OK+Set:[para1]	

查询/设置主模式下搜索时是否仅搜索 HM 模块，如表 3.5 所示。

表 3.5　查询/设置主模式下仅搜索 HM 模块指令

指　令	应　答	参　数
查询：AT+FILT?	OK+Get:[Para]	无
设置：AT+FILT[Para]	OK+Set:[Para]	Para：0～1 0：搜索所有 BLE 从设备 1：仅搜索 HM 模块

查询/设置模块主从模式指令,如表 3.6 所示。

表 3.6 查询/设置主从模式 AT 指令

指　令	应　答	参　数
查询：AT+ROLE?	OK+Get：[para1]	Para1：0～1 1：主设备 0：从设备 Default：0
设置：AT+ROLE[para1]	OK+Set：[para1]	

设置模块工作模式,如表 3.7 所示。

表 3.7 设置模块工作模式

指　令	应　答	参　数
查询：AT+MODE?	OK+Get：[para1]	无
设置：AT+MODE[para1]	OK+Set：[para1]	Para：0～2 0：透传模式 1：PIO 采集+远控+透传 2：透传+远程控制 Default：0

透传模式：即普通的串口透明传输,蓝牙通到数据后转发至串口,同时也转发串口收到的数据到远端蓝牙。替代串口线透明数据需要两个蓝牙模块,一个模块工作在主模式下,一个模块工作在从模式下。当两模块设置为相同的波特率,工作模式设置为透传模式,同时主模块设置搜索所有 BLE 从设备,上电之后,主从模块则自动连接形成串口透明。此时的数据传输则是全双工的。给两个模块分别发送 AT+BAUD0,波特率设置为 9600;给两个模块分别发送 AT+MODE0,工作模式设置为透传模式;给主模块发送 AT+FILT0,设置为搜索所有 BLE 从设备。

模块上电,主模块则自动去查找所有的从模块,此时主模块和从模块的 PIO1 脚都是输出为高低脉冲。若连接成功之后,主从模块的 PIO1 管脚输出为高电平,BT LED 指示灯长亮。

连接成功之后,两个模块两端就能进行串口数据全双工通信了。

2. CYB 蓝牙串口通信协议

接口：UART 波特率 9600,如下,一帧数据为定长 46 字节。

```
u8 DataHeadH;                //包头 0xEE
u8 DataDeadL;                //包头 0xCC
u8 NetID;            //所属网络标识 01(ZigBee) 02(IPv6)03(WiFi)04(Bluetooth)05(RFID)
u8 NetInfoChnanelList[8];   //蓝牙名称
4u8 NetInfoPanID[2];         //蓝牙服务 UUID(0xFFE0/0xFFE1)
5u8 NodeIEEEAddress[8];      //节点 MAC 地址(48b 占数组低 6 字节)
u8 NodeNwkAddress[4];        //保留
```

```
u8 NodeFamilyAddress[4];          //保留
u8 NodeType;                      //节点类型(0:主节点,1:从节点)
u8 NodeState;                     //节点状态(0:掉线,1:在线)
u8 NodeDepth;                     //CMD (0、自动上报 1、搜索 2、连接 3、断开 4、控制)
u8 NodeLinkRSSI;                  //保留
u8 NodePosition;                  //节点位置(同 ZigBee 部分)
u8 SensorType;                    //传感器类型
u8 SensorIndex;                   //传感器 ID
u8 SensorCMD;                     //传感器命令序号
u8 Sensordata1;                   //传感器数据 1
u8 Sensordata2;                   //穿管器数据 2
u8 Sensordata3;                   //传感器数据 3
u8 Sensordata4;                   //传感器数据 4
u8 Sensordata5;                   //传感器数据 5
u8 Sensordata6;                   //传感器数据 6
u8 DataResv1;                     //保留字节 1
u8 DataResv2;                     //保留字节 2
U8 DataEnd;                       //节点包尾 0xFF
```

CBT 传感器说明如表 3.8 所示。

表 3.8　CBT 传感器说明

传感器名称	传感器类型编号	传感器输出数据说明
磁检测传感器	0x01	1-有磁场；0-无磁场
光照传感器	0x02	1-有光照；0-无光照
红外对射传感器	0x03	1-有障碍；0-无障碍
红外反射传感器	0x04	1-有障碍；0-无障碍
结露传感器	0x05	1-有结露；0-无结露
酒精传感器	0x06	1-有酒精；0-无酒精
人体检测传感器	0x07	1-有人；0-无人
三轴加速度传感器	0x08	XH，XL，YH，YL，ZH，ZL
声响检测传感器	0x09	1-有声音；0-无声音
温湿度传感器	0x0A	HH，HL，TH，TL
烟雾传感器	0x0B	1-有烟雾；0-无烟雾
振动检测传感器	0x0C	1-有振动；0-无振动
传感器扩展板	0xFF	用户自定义

CBT 传感器底层协议如表 3.9 所示。

表 3.9　CBT 传感器底层协议

传感器模块	发　　送	返　　回	意义
磁检测传感器	CC EE 01 NO 01 00 00 FF 查询是否有磁场	EE CC 01 NO 01 00 00 00 00 00 00 00 00 FF	无人
		EE CC 01 NO 01 00 00 00 00 00 01 00 00 FF	有人

续表

传感器模块	发　　送	返　　回	意义
光照传感器	CC EE 02 NO 01 00 00 FF 查询是否有光照	EE CC 02 NO 01 00 00 00 00 00 00 00 00 FF	无光照
		EE CC 02 NO 01 00 00 00 00 00 01 00 00 FF	有光照
红外对射传感器	CC EE 03 NO 01 00 00 FF 查询红外对射传感器是否有障碍	EE CC 03 NO 01 00 00 00 00 00 00 00 00 FF	无障碍
		EE CC 03 NO 01 00 00 00 00 00 01 00 00 FF	有障碍
红外反射传感器	CC EE 04 NO 01 00 00 FF 查询红外反射传感器是否有障碍	EE CC 04 NO 01 00 00 00 00 00 00 00 00 FF	无障碍
		EE CC 04 NO 01 00 00 00 00 00 01 00 00 FF	有障碍
结露传感器	CC EE 05 NO 01 00 00 FF 查询是否有结露	EE CC 05 NO 01 00 00 00 00 00 00 00 00 FF	无结露
		EE CC 05 NO 01 00 00 00 00 00 01 00 00 FF	有结露
酒精传感器	CC EE 06 NO 01 00 00 FF 查询是否检测到酒精	EE CC 06 NO 01 00 00 00 00 00 00 00 00 FF	无酒精
		EE CC 06 NO 01 00 00 00 00 00 01 00 00 FF	有酒精
人体检测传感器	CC EE 07 NO 01 00 00 FF 查询是否检测到人	EE CC 07 NO 01 00 00 00 00 00 00 00 00 FF	无人
		EE CC 07 NO 01 00 00 00 00 00 01 00 00 FF	有人
三轴加速度传感器	CC EE 08 NO 01 00 00 FF 查询 XYZ 轴加速度	EE CC 08 NO 01 XH XL YH YL ZH ZL 00 00 FF	XYZ 轴加速度
声响检测传感器	CC EE 09 NO 01 00 00 FF 查询是否有声响	EE CC 09 NO 01 00 00 00 00 00 00 00 00 FF	无声响
		EE CC 09 NO 01 00 00 00 00 00 01 00 00 FF	有声响
温湿度传感器	CC EE 0A NO 01 00 00 FF 查询湿度和温度	EE CC 0A NO 01 00 00 HH HL TH TL 00 00 FF	湿度和温度值
烟雾传感器	CC EE 0B NO 01 00 00 FF 查询是否检测到烟雾	EE CC 0B NO 01 00 00 00 00 00 00 00 00 FF	无烟雾
		EE CC 0B NO 01 00 00 00 00 00 01 00 00 FF	有烟雾
振动检测传感器	CC EE 0C NO 01 00 00 FF 查询是否检测到振动	EE CC 0C NO 01 00 00 00 00 00 00 00 00 FF	无振动
		EE CC 0C NO 01 00 00 00 00 00 01 00 00 FF	有振动

3. 源码实现

主机模块蓝牙初始化函数：

```
void AT_Cmd(void)
{
    Uart_RecvFlag = 1;
    disc = 0;
       / ******** AT CONFIG ********* /
    AT("AT + BAUD0",8);                          //设置波特率为 9600
    AT("AT + IMME0", 8);                         //上电即复位
    AT("AT + MODE0",8);                          //设置通透传输
    AT("AT + ROLE1", 8);                         //查询 设置主模式
    AT("AT + FILT0",8);                          //搜索所有从设备
    UART2_SendString("AT + RESET", 8);           //重启
    delay_ms(5000);
}
```

蓝牙主机模块主函数：

```
int main(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    u8 i = 0;
    NVIC_Configuration();
    CLI();
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, DISABLE);

    //JTAG_Remap
     RCC _ APB2PeriphClockCmd ( RCC _ APB2Periph _ GPIOA | RCC _ APB2Periph _ GPIOB | RCC _
APB2Periph_AFIO, ENABLE);
    //JTAG - DP Disabled and SW - DP Enabled
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
    Systick_Init(72);
    LED_Init();
    LED_USER_On();

    UART1_Configuration();
```

```
    UART2_Configuration();

    //1ms 中断
    TIM3_Configuration();
    TIM2_Configuration();
    BT_Init();
    BT_Reset();
    delay_ms(5000);

    for(i = 0; i < 26;i++)
        tx_buf[i] = 0;
    for(i = 0; i < 14;i++)
        rx_buf[i] = 0;
    tx_buf_init();
    BT_State = 0;
    BT_Cnt = 0;
    Uart_RecvFlag = 0;
    rx_counter = 0;
    Uart_RecvFlag1 = 0;
    rx_counter1 = 0;
    /* Open Global Interrupt */
    SEI();
    AT_Cmd();
    while(1)
    {
       BluetoothHandle();
    }
}
```

主机 BluetoothHandle()关键函数：

```
int BluetoothHandle()
{
    int j = 0;;
        Uart_RecvFlag = 1;                  //不允许串口接收数据
        delay_ms(500);
        if(BT_State == 1)                   //已连接成功
        {
            AT_FLAG = 1;
            Uart_RecvFlag = 0;              //允许串口接收数据
          //while(Uart_RecvFlag == 0);      //等待接收数据
            while((BT_State == 1) && (Uart_RecvFlag == 0));    //等待接收数据
            //处理并发送数据
            tx_buf[29] = SLAVER;
            tx_buf[30] = 0x01;
            tx_buf[33] = rx_buf[2];         //position
            tx_buf[34] = rx_buf[3];
```

```
            tx_buf[35] = rx_buf[4];
            tx_buf[36] = rx_buf[5];
            tx_buf[37] = rx_buf[6];
            tx_buf[38] = rx_buf[7];
            tx_buf[39] = rx_buf[8];
            tx_buf[40] = rx_buf[9];
            tx_buf[41] = rx_buf[10];
            tx_buf[42] = rx_buf[11];
            tx_buf[3] = rx_buf[12];
            tx_buf[4] = rx_buf[15];
            tx_buf[5] = rx_buf[16];       //蓝牙名称;
            tx_buf[6] = rx_buf[17];
            tx_buf[7] = rx_buf[18];
            tx_buf[8] = rx_buf[19];
            tx_buf[9] = rx_buf[20];
            tx_buf[10] = rx_buf[21];
            tx_buf[11] = rx_buf[22];      //UUID
            tx_buf[12] = rx_buf[23];
            tx_buf[13] = 0x00;
            tx_buf[14] = 0x00;
            tx_buf[15] = rx_buf[24];      //mac addr
            tx_buf[16] = rx_buf[25];
            tx_buf[17] = rx_buf[6];
            tx_buf[18] = rx_buf[27];
            tx_buf[19] = rx_buf[28];
            tx_buf[20] = rx_buf[29];
            UART1_SendString(tx_buf, 46);
            LED_USER_Toggle();
        }
        else                              //未连接成功
        {
            UART1_PutString("ERROR!\n");

        }
        return 0;
}
```

从机模块蓝牙初始化函数：

```
void AT_Cmd(void)
{
    AT("AT+BAUD0",8);              //设置波特率为 9600
    AT("AT+IMME0", 8);             //上电即复位
    AT("AT+MODE0",8);              //设置通透传输
    AT("AT+ROLE0", 8);             //查询 设置从模式
    BT_Reset();                    //重启
    delay_ms(5000);
}
```

从机模块主函数：

```
int main(void)
{

    GPIO_InitTypeDef GPIO_InitStructure;
    u8 i = 0;
    NVIC_Configuration();
    CLI();
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, DISABLE);
    //JTAG_Remap
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_
APB2Periph_AFIO, ENABLE);
    //JTAG-DP Disabled and SW-DP Enabled
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
    Systick_Init(72);
    LED_Init();
    LED_USER_On();
    UART1_Configuration();
    UART2_Configuration();

    //1ms 中断
    TIM3_Configuration();
    BT_Init();
    BT_Reset();
    delay_ms(5000);
    BT_State = 0;
    BT_Cnt = 0;
    Uart_RecvFlag = 0;
    rx_counter = 0;

    Uart_RecvFlag1 = 1;
    rx_counter1 = 0;

    /* Open Global Interrupt */
    SEI();
    AT_Cmd();
    Uart_RecvFlag1 = 0;
```

```
    /*************************************/
     while(1)
     {

      LED_USER_On();
      delay_ms(5000);
         //判断是否与主机连接
         if(BT_State == 1)
         {
             delay_ms(100);
             LED_USER_Toggle();
             delay_ms(500);
           if(Uart_RecvFlag)                    //接收 完整一帧 传感器信息
           {
               for(i = 0; i < 14;i++)
                   tx_buf[i] = rx_buf[i];
              tx_buf[1] = 0xcc;
              tx_buf[0] = 0xee;
              delay_ms(100);
              Uart_RecvFlag = 0;
              UART2_SendString(tx_buf, 14);

           }
         }
     }
}
```

4. 硬件连接

用 J-Link 连接 PC 与实验箱，用实验箱配套的电源给实验箱供电，并给模块上电。

5. 建立工程，编译运行

在 IAR Embedded Workbench for ARM 5.41 软件环境中，打开工程，将工程进行编译。具体方法可以选择 Project 中的 Rebuild All 或者选中工程栏中的工程文件然后右击选择 Rebuild All 进行编译，如图 3.29 所示。

用板载的“＋”、“－”按键分别选中主、从机模块，将 Master 和 Slaver 程序分别烧录到蓝牙主、从机模块里。烧写完毕需要复位 Debug，单击“选择”旁边的“复位”即可。

6. 通信测试

通过蓝牙主机的拨码开关选择到 Debug UART，并用 9 针的串口线把实验板上的 Debug UART 串口和 PC 端的串口进行连接，打开串口调试软件，并设置波特率为 9600，校验位为 NONE，数据位为 8，停止位为 1，然后观察结果。图 3.30 中显示的串口中的数据，不代表所有传感结果，实际结果请以实际为准。

对照蓝牙串口通信协议和传感器底层协议，分析串口收到信息的含义，并与实际情况做比较。注意：因为蓝牙主模块设置的是搜索所有从模块，如果同时打开多个蓝牙从模块，主模块会随机与其中一个进行连接，建议使用的时候只打开一个蓝牙从模块。

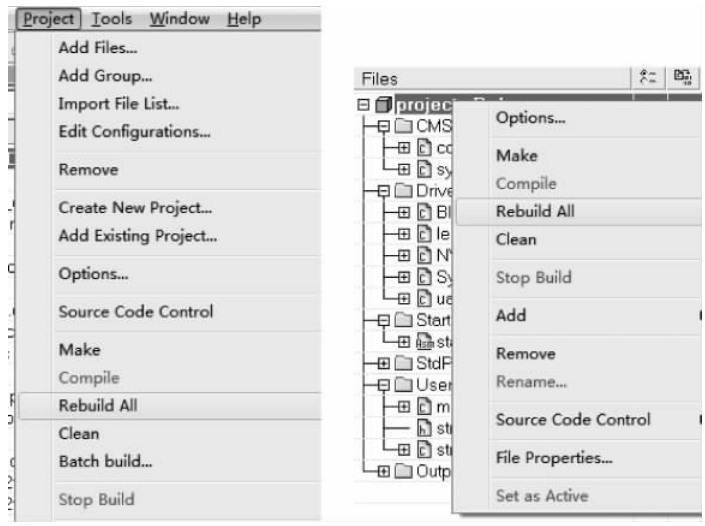

图 3.29 IAR Embedded Workbench for ARM 5.41 软件环境

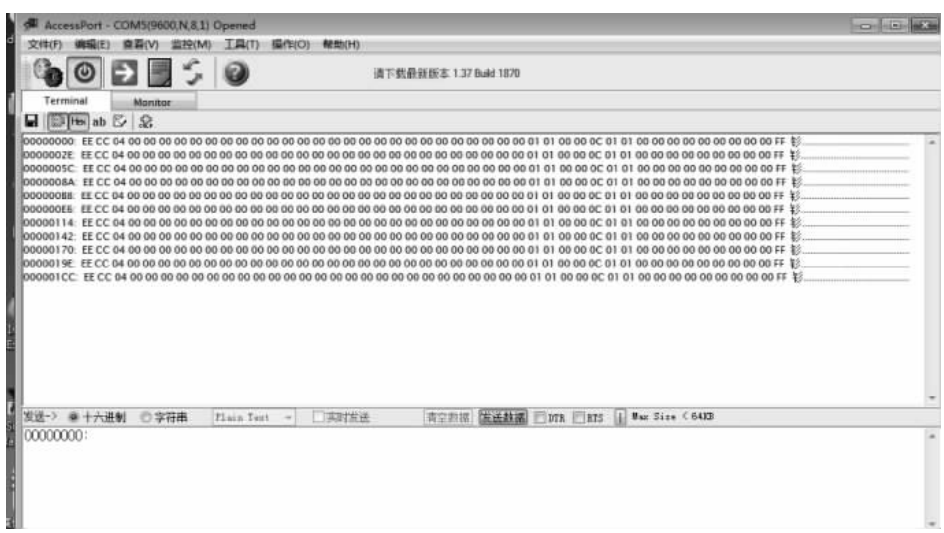

图 3.30 串口接收图

3.4.3 扩展任务：蓝牙设备与手机(PC)通信连接

BF10-I 蓝牙模块支持 1200～1 382 400b/s 等多种接口波特率，通过 AT 指令设置相同的波特率，设置成透传模式，分别设置主从模块，主从模块连接成功后，利用 SPP 蓝牙串行服务，实现模块与手机和 PC 的通信连接。

BT_RST 为蓝牙模块复位引脚，低电平复位，S5 为蓝牙模块复位按键。BT_STATE 为连接状态指示引脚，连接成功后输出高电平，LED 点亮；否则输出脉冲电平，LED

闪烁。UART_TX、UART_RX 为串口通信接口，与 STM32 的 UART2 连接，通过向蓝牙模块发送 AT 指令即可完成对蓝牙模块的配置和控制。

1. 软件工作原理

查询/设置模块主从模式指令，如表 3.10 所示。

表 3.10　查询/设置主从模式 AT 指令

指　令	应　答	参　数
查询：AT＋ROLE?	OK＋Get：[para1]	Para1：0～1 1：主设备 0：从设备 Default：0
设置：AT＋ROLE[para1]	OK＋Set：[para1]	

注：用手机(PC)连接蓝牙模块的时候只需要把蓝牙模块设置成从机模式。

2. 源码实现

主机模块蓝牙初始化函数：

```
int main(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    u8 i = 0;
    NVIC_Configuration();
    CLI();
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB |
                           RCC_APB2Periph_GPIOC, DISABLE);
    //JTAG_Remap
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_
APB2Periph_AFIO, ENABLE);
    //JTAG-DP Disabled and SW-DP Enabled
    GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable, ENABLE);
    Systick_Init(72);
    LED_Init();
    LED_USER_On();

    UART1_Configuration();
    UART2_Configuration();
```

```
    //1ms 中断
    TIM3_Configuration();
    TIM2_Configuration();
    BT_Init();
    BT_Reset();
    delay_ms(5000);

    for(i = 0; i < 26;i++)
        tx_buf[i] = 0;
    for(i = 0; i < 14;i++)
        rx_buf[i] = 0;
    tx_buf_init();
    BT_State = 0;
    BT_Cnt = 0;
    Uart_RecvFlag = 0;
    rx_counter = 0;
    Uart_RecvFlag1 = 0;
    rx_counter1 = 0;
    /* Open Global Interrupt */
    SEI();
    AT("AT+ROLE0", 8);                    //设置从模式
    while(1)
    {
    }

}
```

蓝牙重设复位函数：

```
void BT_Reset(void)
{
    GPIO_ResetBits(GPIOA, GPIO_Pin_1);
    delay_ms(10);
    GPIO_SetBits(GPIOA, GPIO_Pin_1);
    delay_ms(1000);
}
```

4. 硬件连接

用 J-Link 连接 PC 与实验箱，用实验箱配套的电源给实验箱供电，并给模块上电。

5. 建立工程，编译运行

在 IAR Embedded Workbench for ARM 5.41 软件环境中，打开工程，将工程进行编译。具体方法可以选择 Project 中的 Rebuild All 或者选中工程栏中的工程文件然后右击选择 Rebuild All 进行编译，如图 3.31 所示。

将编译好的代码下载到选中的模块中，按下模块上的 BT——RST 按键，将模块

复位一次，如图 3.32 所示。

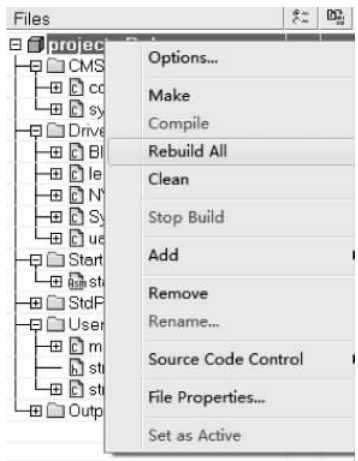

图 3.31　IAR Embedded Workbench for ARM 5.41 软件环境

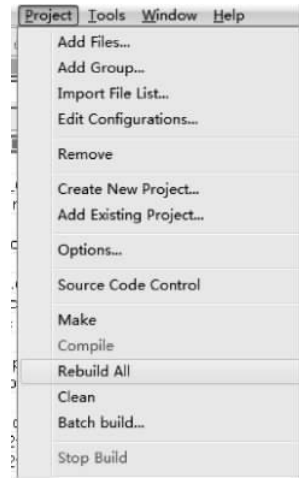

图 3.32　模块复位

5. 通信测试

打开手机(PC)的蓝牙设备，然后搜索设备，搜索的设备中有 Cyb-Bot 时，连接，无须输入密码，即可完成配对如图 3.33 所示。

注意：关于无线蓝牙的使用环境，无线信号包括蓝牙应用都受周围环境的影响很大，如树木、金属等障碍物会对无线信号有一定的吸收，从而在实际应用中，数据传输的距离受一定的影响。

计算机蓝牙驱动问题，在从模式情况下，计算机上使用蓝牙适配器，通用的有 WIDCOMM IVT Windows 自带的驱动。推荐采用 Windows 自带的驱动，如图 3.34 所示。

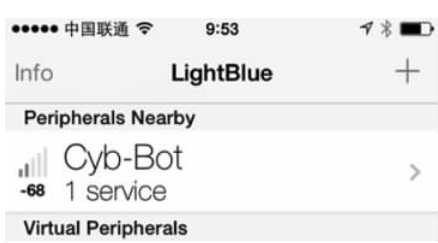

图 3.33　搜索到蓝牙模块

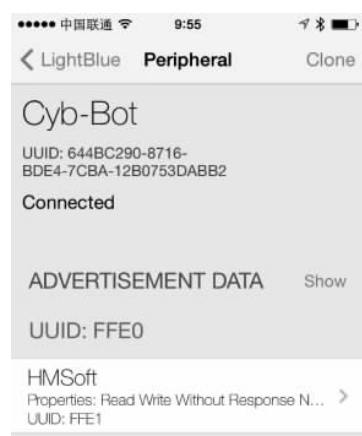

图 3.34　蓝牙模块连接成功

BLE4.0 蓝牙模块连接手机(PC)项目常见问题如下。

1. 什么是 BLE

从蓝牙 4.0 开始有两个分支：经典 4.0 和 BLE4.0。经典 4.0 就是传统的 3.0 蓝牙升级而成，向下兼容。而 BLE 4.0 是一个新的分支，不向下兼容。BLE 是低功耗蓝牙的缩写，顾名思义，其功耗较低。

2. 哪些设备支持 BLE

iPhone 4s/5/5c/5s/iPad 3/4/mini 等都支持 BLE，无须做 MFI 认证。装配了蓝牙 4.0 的 Android 手机并且升级到 Android 4.3 的系统。

3. 为什么计算机上不支持 BLE

计算机上如果装配了 4.0 双模的蓝牙适配器(双模指经典 4.0 和 BLE 4.0)在硬件上是支持 BLE 的，只不过目前的现状比较尴尬，找不到配套的软件去动这个适配器。目前计算机上的代用产品是 HM-15，可以实现计算机支持 BLE 通信。

4. 为什么在系统蓝牙界面下找不到 BLE 设备

手机蓝牙默认工作在经典模式下，需要通过程序来实现搜索，配对连接和通信的整个过程。iOS 系统，如果没有开发者证书请从苹果商店下载 LightBlue，www.jnhuamao.cn 下载中心，有 LightBlue 使用说明。Android 系统，请在 www.jnhuamao.cn 下载 BLE 串口助手，或者从 Android 市场搜索 BLE 串口助手。