目录

青风带你玩蓝牙 nRF52832 系列教程

----------作者：青风

出品论坛：　www.qfv8.com　青风电子社区

作者： 青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：346518370
硬件平台：青云 QY-nRF52832 开发板

第十八章 蓝牙 BLE 温湿度检测

很多读者希望能够把通过蓝牙把检测的环境温湿度发送到手机上，手机通过 APP 接收数据进行显示。通过这样的应用为智能家居做必要的准备。本章将来讨论如何实现蓝牙温湿度的采集，使用的温湿度模块为 DHT11.

那么大体思路有两种方法，方法一：使用已经搭建好的蓝牙串口工程，发送控制指令后开始采集温湿度，温湿度采集完成后通过蓝牙发送给蓝牙串口 APP 接收。第二种方法：采用类似蓝牙按键通知这章的方法，建立一个私有任务，通过空中属性通知的方式发送给手机显示温湿度的值。这两种方法将在本章进行详细的讲解。

这里我们通过一个简单的例子：蓝牙 BLE 温湿度采集，来进行一个简单的思路验证。

18.1 温湿度 DHT11 驱动

DHT11 是一款湿温度一体化的数字传感器。该传感器包括一个电阻式测湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。可以通过单片机等微处理器简单的电路连接就能够实时的采集本地湿度和温度。DHT11 与单片机之间能采用简单的单总线进行通信，仅仅需要一个 I/O 口。传感器内部湿度和温度数据 40Bit 的数据一次性传给单片机，数据采用校验和方式进行校验，有效的保证数据传输的准确性。DHT11 功耗很低，5V 电源电压下，工作平均最大电流 0.5mA，非常适合用于低功耗设备。DHT11 的管脚排列如图 18. 1 所示：

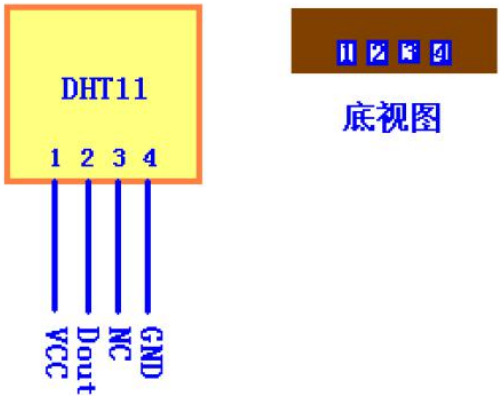

图 18.1：DHT11 管脚排列图

虽然 DHT11 与 DS18B20 类似，都是单总线访问，但是 DHT11 的访问，相对 DS18B20 来说要简单很多。下面我们先来看看 DHT11 的数据结构。DHT11 数字湿温度传感器采用单总线数据格式。即，单个数据引脚端口完成输入输出双向传输。其数据包由 5Byte（40Bit）组成。数据分小数部分和整数部分，一次完整的数据传输为 40bit，高位先出。DHT11 的数据格式为：8bit 湿度整数数据+8bit 湿度小数数据+8bit 温度整数数据+8bit 温度小数数据+8bit 校验和。其中校验和数据为前四个字节相加。传感器数据输出的是未编码的二进制数据。数据(湿度、温度、整数、小数)之间应该分开处理。因此某次从 DHT11 读到的数据应该如下表排列所示：

表 18.1 DHT11 数据格式

Byte4	Byte3	Byte2	Byte1	Byte0
00101101	00010010	00011100	00000000	01001001
整数部分	小数部分	整数部分	小数部分	校验和
湿度值		温度值		校验和

由以上表数据就可得到湿度和温度的值，计算方法很简单，把二进制数据转换为十进制数据，结果为：

湿度= byte4 . byte3=45.0 (%RH)

温度= byte2 . byte1=28.0 (℃)

校验= byte4+ byte3+ byte2+ byte1=73(=湿度+温度)(校验正确)

上面的演示可以看出，DHT11 的数据格式是十分简单的，DHT11 和 MCU 的一次通信最大为 3ms 左右，建议主机连续读取时间间隔不要小于 100ms。下面，我们介绍一下 DHT11 的传输时序。DHT11 的数据发送流程如图 18.2 所示：

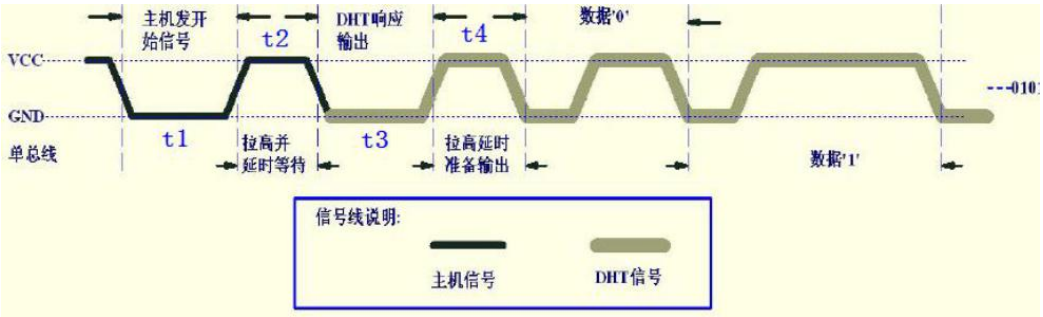

图 18.2：DHT11 的传输时序

首先主机发送开始信号，即：拉低数据线，保持 t1（至少 18ms）时间，然后拉高数据线 t2（20~40us）时间，然后读取 DHT11 的相应，正常的话，DHT11 会拉低数据线，保持 t3（40~50us）时间，作为响应信号，然后 DHT11 拉高数据线，保持 t4（40~50us）时间后，开始输出数据。代码配置如下：

```
u8 dht_Read_Bit(void) //读 bit 位的顺序
{
u8 time=0;
while(dht_data_IN&&time<100)
{
   time++;
   nrf_delay_us(1);
}
time=0;
while(!dht_data_IN&&time<100)
{
   time++;
   nrf_delay_us(1);
}
   nrf_delay_us(40);
   if(dht_data_IN)
     return 1;
     else return 0;
}
```

然后把 Bit 位连续的读取 8 个位，成为读取一个字节。编写代码如下所示：

```
u8 dht_Read_Byte(void) //读一个字节
{
u8 i,dat;
dat=0;
for (i=0;i<8;i++)
  {
  dat<<=1;
  dat|=dht_Read_Bit();
  }
  return dat;
}
```

开始读取传感器采集的温湿度参数。首先复位传感器，然后对传感器进行自检，自检通过表示可以开始读取数据了。根据 DHT11 传感器的数据格式，一次读取 5 个字节的数据，然后对读取的数据进行效验。效验正确就表明数据有效。此时再来区分温度数据和湿度数据。编写代码如下所示：

```
u8 dht_Read_Data(u8 *temp,u8 *humi) //读取整个温湿度数据
{
u8 dat[5];
u8 i;
dht_Rst();//复位
if(dht_Check()==0)//自检
```

```
{
for(i=0;i<5;i++)
{
dat[i]=dht_Read_Byte();//读取 5 个字节的数据
}
if((dat[0]+dat[1]+dat[2]+dat[3])==dat[4])//判断数据是否正确
{
 *humi=dat[0];//演示的时候不要小数部分。湿度值
 *temp=dat[2];//温度值
}
}else return 1;
return 0;
}
```

温湿度传感器的驱动，实际上是简单的读取输出 I0 端口的电平值，这些电平值组成的二进制数据就是当前测试的温湿度数据，然后把数据保存到数据指针内，为了演示方便，暂时没加入小数部分，如果需要加入小数部分，可以加入 2 个字节数组表示小数部分。

18.2：温湿度采集方法一

18.2.1 应用层初始化传感器

本节的方法以蓝牙串口工程为框架，在已经搭建好的数据通道内进行温湿度数据的上传服务。首先，需要在蓝牙串口的工程中，加入上一节编写的温湿度传感器的驱动函数文件 dht11.c，如下图 18.3 所示：

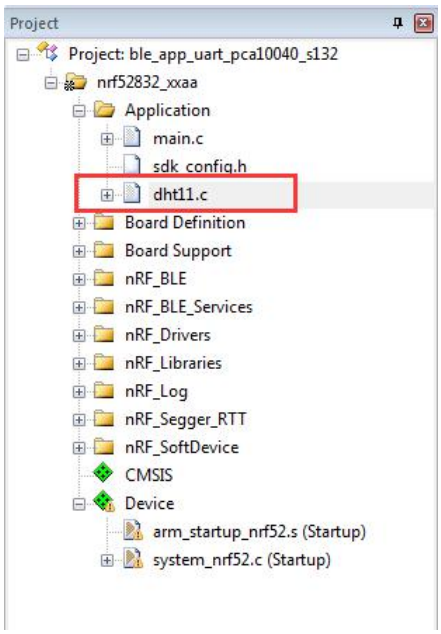

图 18.3：工程中加入 DHT11 驱动

添加完库文件后，需要同时添加该函数文件的路径，把 dht11.c 文件和 dht11.h 文件放到工程主

目录下新建的 drive 文件夹下，因此文件路径如下：\drive。在工程配置的 Options for Target 选项卡中的 C/C++选项下：点击 Include Paths 中添加路径，如下图 18.4 所示：

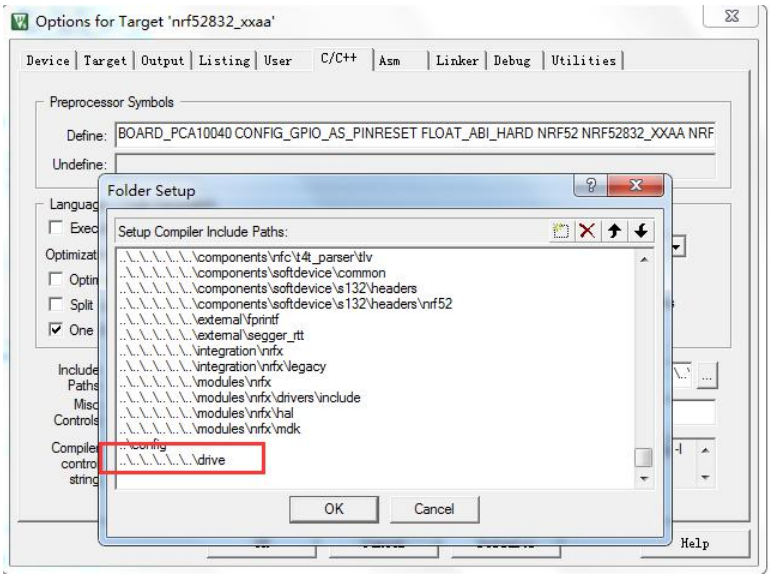

图 18.4：添加 DHT11 文件路径

文件路径添加完成后，首先在主函数文件 main.c 中添加头文件 dht11.h，主函数中只需要修改一个位置，也就是加入传感器初始化。代码如下所示，在主函数加入温湿度初始化函数，检查是否传感器加入成功，如果初始化温湿度传感器不成功，则设备不会开始广播：

```
1.  int main(void)
2.  {
3.  uint32_t err_code;
4.  bool erase_bonds;
5.  uint8_t   start_string[] = START_STRING;
6.  while(dht_Init())   //添加温湿度传感器初始化成功
7.  {
8.    nrf_delay_ms(1000);//延迟等待
9.  }
10. // Initialize.
11. APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_OP_QUEUE_SIZE, false);
12. uart_init();
13. buttons_leds_init(&erase_bonds);
14. ble_stack_init();
15. .....
16. .....
17. // Enter main loop.
18. for (;;)
19. {
20. power_manage();
21. }
22. }
```

18.2.2 采集指令发送

蓝牙串口发送采集指令，通过命令判断是否开始采集，类似于串口的 AT 指令，然后再通过 BLE 串口透传通道温湿度模块采集到的数据上传给你的主机。首先编写温湿度上传函数 ble_send（），这个函数调用了串口上传函数代码 ble_nus_data_send（），具体编写如下代码所示：

```
1.  void ble_send(uint8_t * string,uint16_t length)
2.  {
3.  uint32_t err_code;
4.  /把采集的数据通过 BLE 串口透传通道上传给主机
5.  err_code = ble_nus_data_send(&m_nus,string,&length,m_conn_handle);
6.  if (err_code != NRF_ERROR_INVALID_STATE)
7.  {
8.  APP_ERROR_CHECK(err_code);
9.  }
10. }
```

在串口蓝牙处理回到函数中，判断手机发过来的指令。当手机发送 ON 指令控制温湿度模块开始采集，开发板作为从机会触发一个串口蓝牙接收事件 BLE_NUS_EVT_RX_DATA，在该事件下判断主机发送的指令是不是开始采集的命令，如果是开始采集的指令，则启动温湿度传感器进行数据采集，然后把采集后的数据通过温湿度上传函数发送给手机，关于蓝牙部分的功能设置请具体参考《第十六章：蓝牙串口详解》。具体编写的代码如下：

```
1.  static void nus_data_handler(ble_nus_t * p_nus, uint8_t * p_data, uint16_t length)
2.  {
3.  uint8_t data[4],w,s;//声明一个数组和温湿度字节
4.  if (p_evt->type == BLE_NUS_EVT_RX_DATA)
5.  {
6.    NRF_LOG_HEXDUMP_DEBUG(p_evt->params.rx_data.p_data, p_evt->params.rx_data.length);
7.     if(((p_evt->params.rx_data.p_data[0]=='O')||(p_evt->params.rx_data.p_data[0]=='o'))
8.        &&((p_evt->params.rx_data.p_data[1]=='N')||(p_evt->params.rx_data.p_data[1]=='n')))
9.  //如果收到的数据为“ON”或者"on"
10. {
11.   data[0]=0x33; //首先给一个头标志
12. if(dht_Read_Data(&w,&s)) //如果没有读取到数据则发送 0
13. {
14.   data[1]=0x00;
15.   data[2]=0x00;
16.   ble_send(data,3);
17. }
18. else //如果读取了数据
19. {
20. data[1]=w;
21. data[2]=s;//读取到了数据后，把读取的数据上传主机
22. ble_send(data,3);//上传实际采集的数据
```

```
23. }
24.
25. }
26. ..............................
27. ..............................
28. }
29. }
```

18.2.3 下载与测试

首先需要把 DHT11 传感器插上开发板接。DHT11 模块如下图 18.5 所示连接开发板的 P5 接口。传感器的三个接口，分别对应 VCC DATA GND:

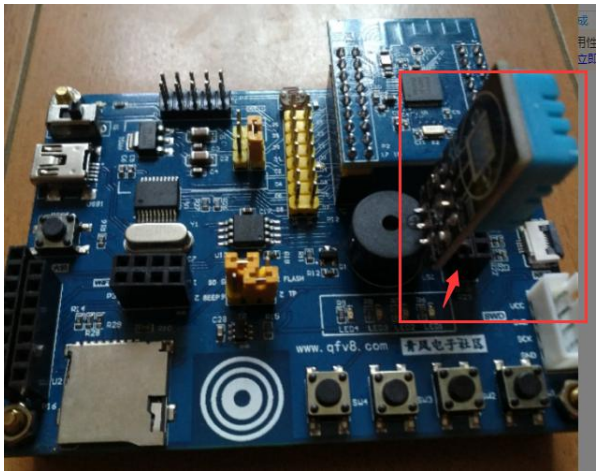

图 18.5：温湿度模块接法

然后打开 nRFgo studio 软件进行协议栈的下载，下载完后可以下载工程，首先把工程编译一下，编译通过后点击 KEIL 上的下载按键。下载成功后程序开始运行，同时开发板上广播 LED 开始广播。打开手机 APP 软件 TOOLbox,打开其中的串口 uart 功能，如下图 18.6 所示：

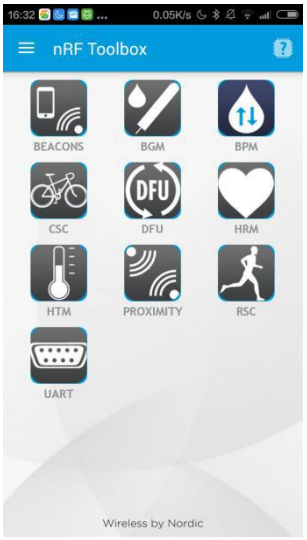

图 18.6：手机 APP 软件 TOOLbox 界面

点击 uart 选项后，发现广播信号并且进行连接后。如下图 18.7 左图所示，点击连接设备的 show log 选项,如下图 18.7 所示：

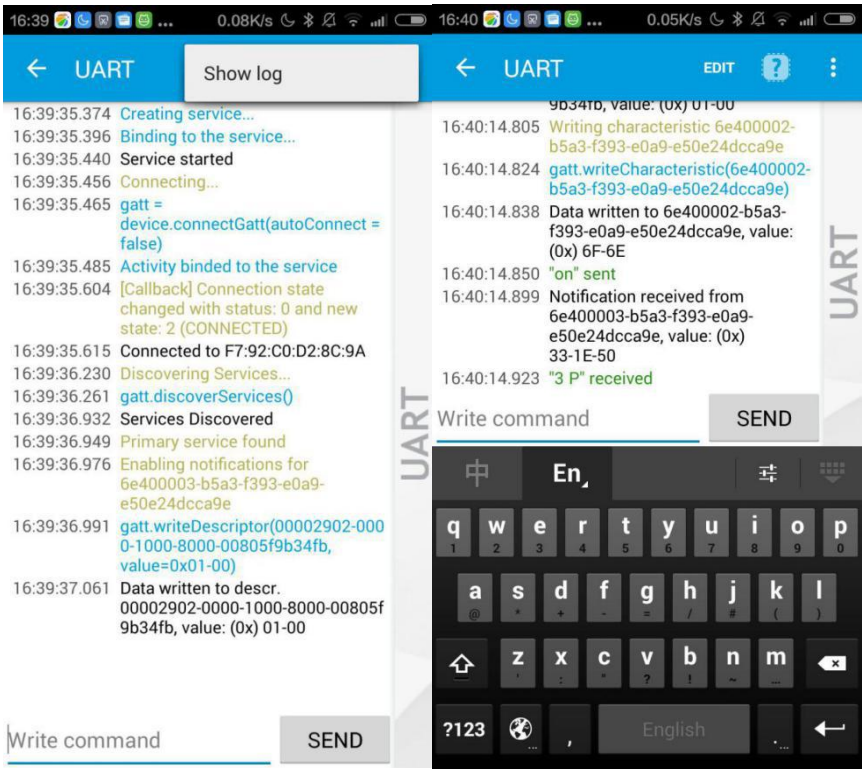

图 18.7：连接后发生采集

show log 选项可以观察到我们发送的指令和接收的数据。再 Write command 命令行填入指令 on，并且点击 SEND。开发板接收到采集开始命令。温湿度传感器就开始采样温度和湿度：如上图 18.7 右图所示，采集的 VALE 值显示如图：33-1E-50，依顺序分别对应其中标志码 data[0]=0x33，温度 data[1]=w;湿度 data[2]=s;

18.3：温湿度采集方法二

本节接上一节的教程来讲的，讲述温湿度采集的方法二。这里我们通过一个简单的例子：蓝牙 BLE 温湿度采集服务，来进行一个简单的思路验证。通过建立私有任务来实现数据信息的采集。程序框架在 BLE 实验按键通知的基础上进行修改，加入传感器驱动和私有蓝牙任务的建立，从而实现我们的设计目标。

18.3.1 初始化传感器

首先在主函数文件 main.c 中添加头文件 dht11.h，主函数中只需要修改一个位置，也就是加入传感器初始化。代码如下所示，在主函数加入温湿度初始化函数，检查是否传感器加入成功，如果初始化温湿度传感器不成功，则设备不会开始广播，同时本例中需要开启定时器，因此主函数中，还需要调用 application_timers_start()函数来启动定时器。具体代码如下所示：

```
#include "dht11.h"//加入头文件

int main(void)
{
    // 初始化
    log_init();
    leds_init();
    timers_init();
    buttons_init();
    while(dht_Init())
     {
     nrf_delay_ms(100);
     }

    .................
    .................
    application_timers_start();//启动定时器
    NRF_LOG_INFO("Blinky example started.");
    advertising_start();

    //进入待机循环
    for (;;)
    {
        idle_state_handle();
    }
}
```

本节的方法以蓝牙按键通知的工程为框架，因此需要在已经搭建好的工程服务中，加入温湿度数据的上传服务特性。 首先在蓝牙按键通知的工程中，加入编写好温湿度传感器的驱动函数文件，如下图 18.8 所示：

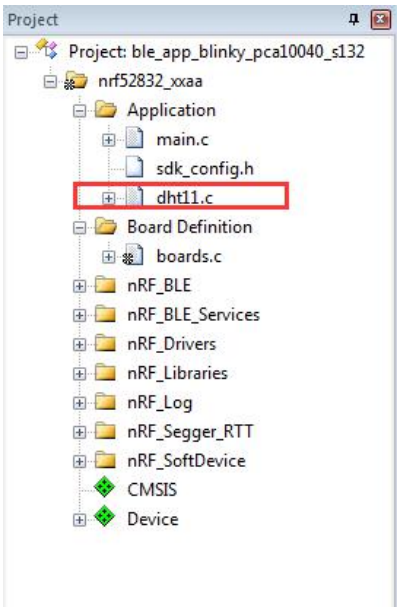

图 18.8：工程中加入 DHT11 驱动

添加完库文件后，需要同时添加该函数文件的路径，把 dht11.c 文件和 dht11.h 文件放到工程主目录下新建的 drive 文件夹下，因此文件路径如下：\drive。在工程配置的 Options for Target 选项卡中的 C/C++选项下：点击 Include Paths 中添加路径，如下图 18.4 所示：

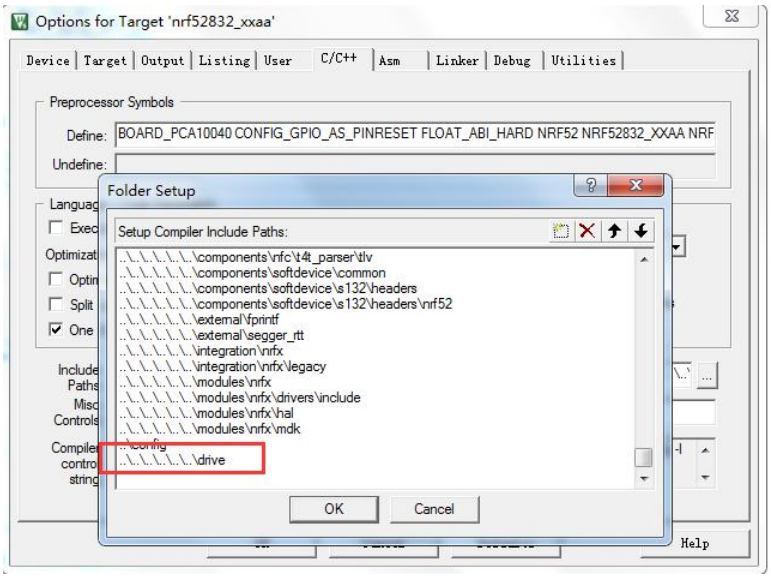

图 18.9：加入 DHT11 驱动文件路径

18.3.2 私有任务建立

1 DHT11 服务特性的添加：

对于私有任务的建立在前面的《蓝牙 LED 任务读写》和《蓝牙按键通知》这两章里已经详细的进行过分析讲解。本例要建立的蓝牙私有任务的空中属于属性通知类型，和按键反馈的任务类似，

把采集到的温湿度数据上传给主机。在配置温湿度传感特性之前，需要首先在服务文件的头文件 ble_lbs.h 中添加温湿度采集特性的 UUID 和温湿度采集特性的操作句柄。

温湿度采集特性的 UUID 为一个 16bit 的 UUID，可以自己任意声明，这个 UUID 用来表示温湿度采集的任务。同时再添加服务特性的时候，服务操作的句柄也需要自己进行声明，服务句柄的类型为 ble_gatts_char_handles_t 结构体类型，具体代码如下所示：

```
#define LBS_UUID_SERVICE              0x1523
#define LBS_UUID_BUTTON_CHAR     0x1524
#define LBS_UUID_LED_CHAR           0x1525
#define LBS_UUID_DHT11_CHAR       0x1526//自己声明一个温湿度采集特性的 UUID

struct ble_lbs_s//服务参数结构体
{
    uint16_t                        service_handle;
    ble_gatts_char_handles_t    led_char_handles;
    ble_gatts_char_handles_t    button_char_handles;
   ble_gatts_char_handles_t    dht11_char_handles;//加入温湿度采集的句柄
    uint8_t                         uuid_type;
    ble_lbs_led_write_handler_t led_write_handler;
};
```

声明完成后，在蓝牙服务文件 ble_lbs.c 文件中，进行的是蓝牙 DHT11 的任务服务特征的添加。具体代码如下所示：

```
static uint32_t dht_char_add(ble_lbs_t * p_lbs, const ble_lbs_init_t * p_lbs_init)
{
ble_gatts_char_md_t   char_md;
ble_gatts_attr_md_t   cccd_md;
ble_gatts_attr_t          attr_char_value;
ble_uuid_t                ble_uuid;
ble_gatts_attr_md_t     attr_md;
//设置 CCCD 特征描述符的安全属性
memset(&cccd_md, 0, sizeof(cccd_md));
BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.read_perm);
BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.write_perm);
cccd_md.vloc = BLE_GATTS_VLOC_STACK;
//配置空中属性
memset(&char_md, 0, sizeof(char_md));
char_md.char_props.read = 1;//可读
char_md.char_props.notify = 1;//通知
char_md.p_char_user_desc    = NULL;
char_md.p_char_pf             = NULL;
char_md.p_user_desc_md      = NULL;
char_md.p_cccd_md             = NULL;
char_md.p_cccd_md             = &cccd_md;
//设置特性的 UUID
```

```
23. ble_uuid.type = p_lbs->uuid_type;
24. ble_uuid.uuid = LBS_UUID_DHT11_CHAR;
25.
26. memset(&attr_md, 0, sizeof(attr_md));
27. //设置安全级别
28. BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);
29. BLE_GAP_CONN_SEC_MODE_SET_NO_ACCESS(&attr_md.write_perm);
30. attr_md.vloc            = BLE_GATTS_VLOC_STACK;
31. attr_md.rd_auth        = 0;
32. attr_md.wr_auth        = 0;
33. attr_md.vlen            = 0;
34.
35. memset(&attr_char_value, 0, sizeof(attr_char_value));
36. //配置 GATT 特征值参数
37. attr_char_value.p_uuid            = &ble_uuid;
38. attr_char_value.p_attr_md        = &attr_md;
39. attr_char_value.init_len         = 2;
40. attr_char_value.init_offs        = 0;
41. attr_char_value.max_len          = 2;
42. attr_char_value.p_value          = NULL;
43. //添加此特性到 GATT 协议层
44. return sd_ble_gatts_characteristic_add(p_lbs->service_handle,
45.                                         &char_md,
46.                                         &attr_char_value,
47.                                         &p_lbs->dht11_char_handles);
48. }
```

以上代码分析如下，需要注意的几点：

1.对应通知的空中属性，CCCD 客户端特征描述符需要配置安全级别，使用宏 BLE_GAP_CONN_SEC_MODE_SET_OPEN 把 CCCD 设置成对任何连接和加密都是可读可写的模式。如果对于按键状态通知的特性，我们想让每一个连接都可读但不能写，可以使用 BLE_GAP_CONN_SEC_MODE_SET_NO_ACCESS 代替 BLE_GAP_CONN_SEC_MODE_SET_OPEN。

```
1.  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.read_perm);
2.  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.write_perm);//cccd 的安全特性
3.  .....................
4.  .....................
5.  BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);//通知属性的安全特性
6.  BLE_GAP_CONN_SEC_MODE_SET_NO_ACCESS(&attr_md.write_perm);//不允许写
```

2 设置 DHT11 的 UUID 的类型和 UUID 的值，关于这一点内容，参考第十四章《蓝牙任务的 UUID 设置和总结》内容。

3.对应温湿度传感器 DHT11 的数据来说，我们只需要上传给主机两个字节的长度，一个字节表示温度，一个字节表示湿度，因此，对应 GATT 访问的数据长度初始化设置 2，最大的数据长度也设置为 2，来表示 2 个字节的数据上传长度，如果需要加入小数部分，则这个字节长度就应该为 4。

```
7.  attr_char_value.init_len    = 2;
8.  attr_char_value.init_offs = 0;
```

```
9.  attr_char_value.max_len = 2;
```

4.可以不需要设置存储数据的结构体，你可以把 p_initial_value 设置为 NULL，只需要注意确保把返回的特性句柄 DHT11_char_handles 中的参数保存在正确的地方，最后调用的特性添加函数如下：

```
return ble_gatts_characteristic_add( p_lbs->service_handle,&char_md,&attr_char_value,
                                     &p_lbs->dht11_char_handles );
```

2 主服务中增加特性：

创建了增加的特性的函数之后，你可以在服务初始化函数 ble_lbs_init（）的末尾调用它们，如下面的代码所示：

```
1.  // 添加特性
2.  err_code = dht_char_add(p_lbs, p_lbs_init);
3.  if (err_code != NRF_SUCCESS)
4.  {
5.  return err_code;
6.  }
7.  return   NRF_SUCCESS;
```

因为任何错误都会导致函数提前退出，因此当你到达函数的末尾时可以认为初始化成功了。

18.3.3 采集温湿度数据上传

你已经添加了一个回调 API 函数让服务知道温湿度采集数据，但还没有全部实现，需要给对等设备发送一个通知以告知主机它新的采样数据变化。协议栈 SoftDevice API 函数 sd_ble_gatts_hvx 来完成这个事情，它需要连接句柄和结构体参数 ble_gatts_hvx_params_t 作为输入，它管理一个值被通知的整个过程，所以调用 sd_ble_gatts_hvx 函数成为关键，详细代码如下：

```
1.  uint16_t ble_lbs_on_dht_change(ble_lbs_t * p_lbs, uint8_t dht11tempval,uint8_t dht11humival)
2.  {
3.  ble_gatts_hvx_params_t params;
4.  uint16_t len = 2;//数据长度
5.  uint32_t err_code;
6.  uint8_t    thi_val[2];
7.
8.  thi_val[0] = (uint8_t)dht11tempval;   //温度采集数据
9.  thi_val[1] = (uint8_t)dht11humival;      //湿度采集数据
10.
11.
12. if (p_lbs->conn_handle != BLE_CONN_HANDLE_INVALID)
13. {
14. memset(&params, 0, sizeof(params));
15. params.type = BLE_GATT_HVX_NOTIFICATION;
16. params.handle = p_lbs->dht11_char_handles.value_handle;
```

```
17. params.p_data = thi_val;
18. params.p_len = &len;
19.
20. return    sd_ble_gatts_hvx(p_lbs->conn_handle, &params);//数据上传
21. }
22. else{
23. err_code = NRF_ERROR_INVALID_STATE;
24. }
25. return err_code;
26.
27. }
```

上面代码中，把温度采集数据和湿度采集数据作为形参，赋值给上传参数结构体中的数据 p_data 参数。同时本次数据上传的句柄为 DHT11 操作句柄中的参数句柄，当设备保存当初连接的时候，就可以调用 sd_ble_gatts_hvx（）这个协议栈的 API 函数进行数据上传了。

18.3.4 定时器的建立与启动

对应数据更新，这里需要定时器进行定时发起上传，这个蓝牙电池和蓝牙心电的应用类似。我们需要创建一个定时器，定一个时间进行数据的更新。代码如下：

```
1.  static void timers_init(void)
2.  {
3.    ret_code_t err_code = app_timer_init();//初始化定时器
4.    APP_ERROR_CHECK(err_code);
5.    // 创建定时器
6.    err_code = app_timer_create(&m_thi_timer_id,APP_TIMER_MODE_REPEATED,
7.                                thi_monitor_timeout_handler);
8.    APP_ERROR_CHECK(err_code);
9.  }
```

当定时的时间到了后，就会发生定时超时中断中，中断处理回调函数中就应该执行数据更新上传的操作，调用前面编写的温湿度数据上传函数 ble_lbs_on_dht_change（），那么我们下面编写一下中断处理回调程序，程序中更新采样数据，具体代码如下：

```
1.  static void thi_monitor_timeout_handlerr(void)
2.  {
3.  uint32_t err_code;
4.  uint8_t dht11_temp_val;
5.  uint8_t   dht11_humi_val;
6.  uint8_t w,s;
7.  if(dht_Read_Data(&w,&s)) //判断数据读取是否成功
8.  {
9.     dht11_temp_val=0x00;
10.    dht11_humi_val=0x00;
11. }
```

```
12.   else
13.  {
14.     dht11_temp_val=w;
15.     dht11_humi_val=s;
16.  }
17.  //成功后更新数据到手机上
18.  err_code = ble_lbs_on_dht_change(&m_lbs,dht11_temp_val,dht11_humi_val);
19.
20.  if ((err_code != NRF_SUCCESS) &&
21.  (err_code != NRF_ERROR_INVALID_STATE) &&
22.  (err_code != BLE_ERROR_INVALID_CONN_HANDLE)&&
23.  (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
24.  )
25.  {
26.  APP_ERROR_HANDLER(err_code);
27.  }
28.  }
29.
```

最后，需要在主函数中开启定时器，开始定时器后等待定时器中断发生，参考前面软件定时器章节的内容，声明本次的软件定时器 ID 为 m_thi_timer_id 后，直接调用 app_timer_start（）函数开启定时器：

```
30.  static void application_timers_start(void)
31.  {
32.  uint32_t err_code;
33.  //启动定时器
34.  err_code = app_timer_start(m_thi_timer_id, THI_MONITOR_INTERVAL, NULL);
35.  APP_ERROR_CHECK(err_code);
36.  }
```

18.3.5 下载与调试

首先需要把 DHT11 传感器插上开发板接。DHT11 模块如下图 18.10 所示连接开发板的 P5 接口。传感器的三个接口分别对应 VCC、 DATA、 GND:

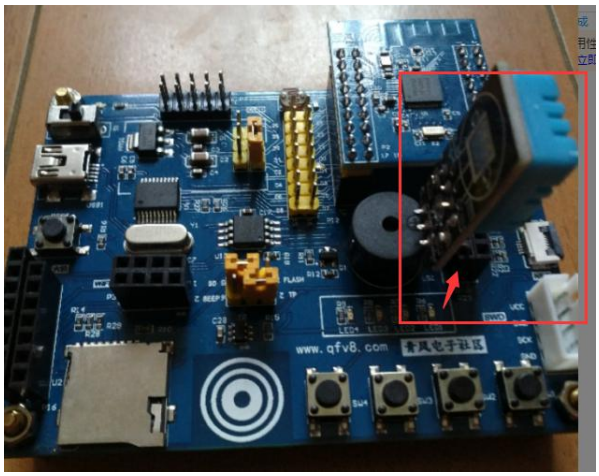

图 18.10：温湿度模块接法

然后打开 nRFgo studio 软件进行协议栈的下载，下载完后可以下载工程，首先把工程编译一下，编译通过后点击 KEIL 上的下载按键。下载成功后程序开始运行，同时开发板上广播 LED 开始广播。打开手机 APP 软件 nrf connect,，如下图 18.11 所示。搜索到蓝牙广播信号 qfdz_dht11：

图 18.11 搜索到蓝牙广播

点击连接后如下图 18.12 所示，找到私有服务，按下图步骤：首先使能通知，然后读取通知值，显示参数 VALE 值显示如左图：20-49：

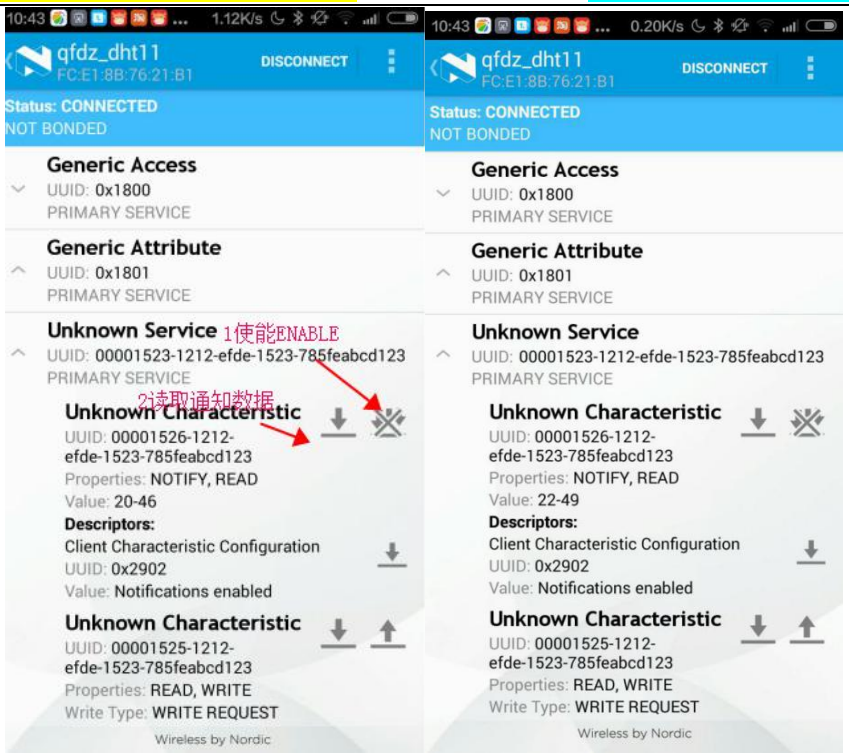

图 18.12：读取采集数据

一段时间后数据发生自动更新：如上图 18.12 右图所示，采集的 VALE 值显示如图：22-49， 数据表示两个字节，第一个字节 data[1]为温度值，第二个字节 data[2]湿度值； 显示的 VALE 值为十六进制数据，转换为十进制就是（℃）和（%RH）为单位。

18.4：本章总结

本章主要是对于从机上传主机这个数据通道的应用演示。演示了两种方式实现从机发送数据到主机的过程。第一种方法采用已经搭建好的传输通道框架蓝牙串口的方式，这种方式非常方便简单，数据接口灵活，编写 APP 容易，但是程序不简洁，占用资源较多；第二种方法则采用直接搭建服务的方式，分配一个通知或者指示的空中属性的特征值参数来实现数据通道的搭建，这种方法编程较为复杂，但是程序简洁，针对性很强；两种方法读者可以灵活选用，因此本章的方法可以应用与各种传感器的上传。