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

2.30 蓝牙 BLE 温湿度检测方法一

很多朋友和客户希望能够把通过蓝牙检测环境温湿度，并且通过手机接收温湿度参数。为智能家居做必要的准备。本章将来讨论蓝牙温湿度采集，使用温湿度模块 DHT11.

那么大体思路有两种方法，方法一：使用蓝牙串口工程，发送指令（类似 AT 指令）后开始采集温湿度，温湿度通过蓝牙串口 APP 接收，本章主要介绍这种方法。第二种方法，采用类似蓝牙按键通知这章的内容，建立一个私有任务下，通过通知的内容显示温湿度。

本例接上一章的原创教程来讲的，作为方法二。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。通过建立私有任务来信息数据信息的采集。

1：协议栈下实现流程：

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。实现我们的设计目标。

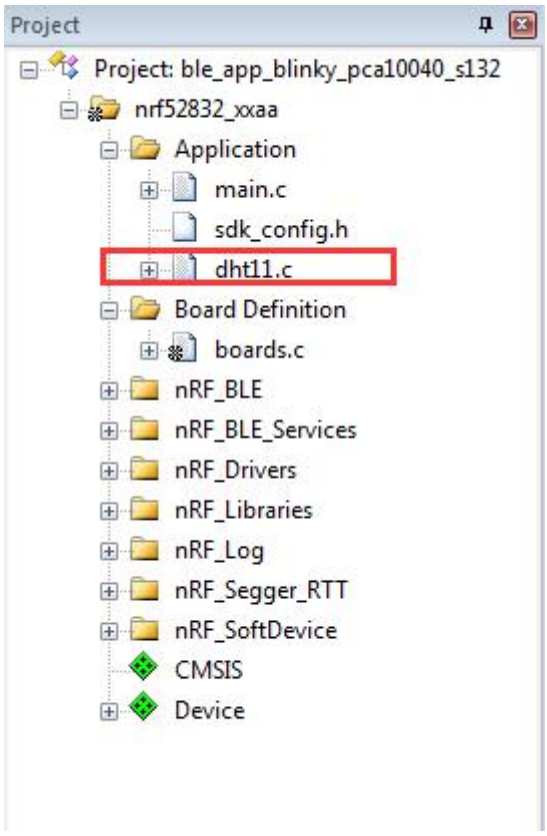
1.1 初始化传感器

本例在主函数中，只需要修改一个位置，也就是加入传感器初始化，代码如下，检查是否传感器加入成功，温湿度初始化的代码上一章已经详细讲解过，这里不再重复：

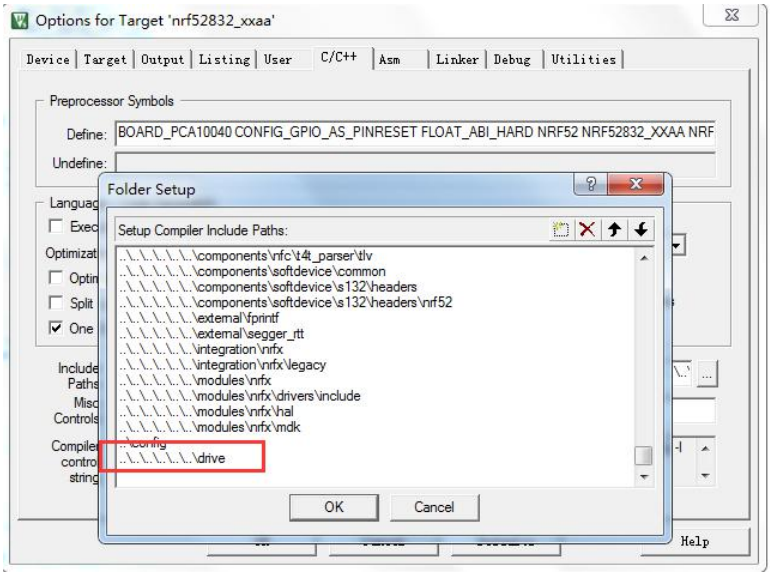
```
int main(void)
{
    uint32_t err_code;
    bool erase_bonds;
    uint8_t  start_string[] = START_STRING;
      while(dht_Init())
    {
```

```
    delay_ms(1000);
    }
    // Initialize.
    APP_TIMER_INIT(APP_TIMER_PRESCALER,     APP_TIMER_OP_QUEUE_SIZE,
false);
    uart_init();
    buttons_leds_init(&erase_bonds);
    ble_stack_init();
    gap_params_init();
    services_init();
    advertising_init();
    conn_params_init();
    err_code = ble_advertising_start(BLE_ADV_MODE_FAST);
    APP_ERROR_CHECK(err_code);

    // Enter main loop.
    for (;;)
    {
        power_manage();
    }
}
```

同时在工程中，加入我们写的温湿度传感器的驱动函数，如下图所示：



同时需要添加工程文件路径：



1.2 私有任务建立

1.2.1 DHT11 服务特性的添加：

私有任务的建立实际上在之前《蓝牙 LED 任务读写详解》和《蓝牙按键反馈通知》这两篇远程教程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属于通知类型，和按键反馈的任务类似。

在蓝牙任务文件 ble_lbs.c 文件中首先是蓝牙 DHT11 的任务服务特征的添加，代码如下：

```
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;

    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;

    memset(&char_md, 0, sizeof(char_md));
```

```
    char_md.char_props.read = 1;
    char_md.char_props.notify = 1;
    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;

    ble_uuid.type = p_lbs->uuid_type;
    ble_uuid.uuid = LBS_UUID_DHT11_CHAR;

    memset(&attr_md, 0, sizeof(attr_md));

    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);
    BLE_GAP_CONN_SEC_MODE_SET_NO_ACCESS(&attr_md.write_perm);
    attr_md.vloc        = BLE_GATTS_VLOC_STACK;
    attr_md.rd_auth     = 0;
    attr_md.wr_auth     = 0;
    attr_md.vlen        = 0;

    memset(&attr_char_value, 0, sizeof(attr_char_value));

    attr_char_value.p_uuid      = &ble_uuid;
    attr_char_value.p_attr_md   = &attr_md;

    attr_char_value.init_len    = 2;
    attr_char_value.init_offs   = 0;
    attr_char_value.max_len     = 2;
    attr_char_value.p_value     = NULL;

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

以上代码分析如下，需要注意的几点：

1. 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。

```
BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.read_perm);
BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.write_perm);
```

```
...
memset(&attr_md, 0, sizeof(attr_md));
BLE_GAP_CONN_SEC_MODE_SET_OPEN(&attr_md.read_perm);
BLE_GAP_CONN_SEC_MODE_SET_NO_ACCESS(&attr_md.write_perm);
```

2.设置 DHT11 的 UUID 的类型和 UUID 的值：

```
ble_uuid.type = p_lbs->uuid_type;
ble_uuid.uuid = LBS_UUID_DHT11_CHAR;
```

3.初始化值不重要，你可以把 p_initial_value 设置为 NULL。确保把返回的特性句柄 DHT11_char_handles 保存在正确的地方，最后调用的函数如下：

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

1.2.2 增加特性：

创建了增加特性的函数之后，你可以在服务初始化的末尾调用它们，如下面的例子：

```
// Add characteristics
  err_code = dht_char_add(p_lbs, p_lbs_init);
if (err_code != NRF_SUCCESS)
{
return err_code;
}
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。

1.2.3 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：

```
uint16_t    ble_lbs_on_dht_change(ble_lbs_t    *    p_lbs,    uint8_t    dht11tempval,uint8_t
dht11humival)
{
      ble_gatts_hvx_params_t params;
         uint16_t len = 2;//数据长度
```

```
        uint32_t err_code;
        uint8_t    thi_val[2];

        thi_val[0] = (uint8_t)dht11tempval;   //文档采集数据
        thi_val[1] = (uint8_t)dht11humival;      //湿度采集数据


        if (p_lbs->conn_handle != BLE_CONN_HANDLE_INVALID)
           {
                memset(&params, 0, sizeof(params));
        params.type = BLE_GATT_HVX_NOTIFICATION;
        params.handle = p_lbs->dht11_char_handles.value_handle;
        params.p_data = thi_val;
        params.p_len = &len;

        return sd_ble_gatts_hvx(p_lbs->conn_handle, &params);//数据上传
           }
           else{
                err_code = NRF_ERROR_INVALID_STATE;
           }
      return err_code;

}
```

1.3 定时器的建立

数据更新是需要定时器进行定时采样的，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：

```
static void timers_init(void)
{
        ret_code_t err_code = app_timer_init();
        APP_ERROR_CHECK(err_code);

        // 创建定时器
      err_code = app_timer_create(&m_thi_timer_id,
                                  APP_TIMER_MODE_REPEATED,
                                  thi_monitor_timeout_handler);
        APP_ERROR_CHECK(err_code);
}
```

定时超时中断中就应该执行数据更新程序，那么我们下面编写一下中断程序，程序中更新采样数据，具体代码如下：

```
static void thi_monitor_handler(void)
{
   uint32_t err_code;
      uint8_t dht11_temp_val;
   uint8_t    dht11_humi_val;
    uint8_t w,s;
    if(dht_Read_Data(&w,&s)) //判断数据读取是否成功
                              {
                                    dht11_temp_val=0x00;
                                    dht11_humi_val=0x00;

                              }
                              else
                              {
                                   dht11_temp_val=w;
                                   dht11_humi_val=s;
                              }

          err_code = ble_lbs_on_dht_change(&m_lbs,dht11_temp_val,dht11_humi_val);// 成功后更新数据到手机上
      if ((err_code != NRF_SUCCESS) &&
          (err_code != NRF_ERROR_INVALID_STATE) &&
          (err_code != BLE_ERROR_INVALID_CONN_HANDLE)&&
          (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
          )
      {
                  APP_ERROR_HANDLER(err_code);
      }

}

//超时中断操作
static void thi_monitor_timeout_handler(void * p_context)
{
      UNUSED_PARAMETER(p_context);
      thi_monitor_handler();
}
```

最后，开启定时器，开始定时，等待定时器中断发生：

```
static void application_timers_start(void)
{
      uint32_t err_code;
      //启动定时器
```

```
    err_code = app_timer_start(m_thi_timer_id, THI_MONITOR_INTERVAL, NULL);
    APP_ERROR_CHECK(err_code);
}
```

1.4 主函数的实现：

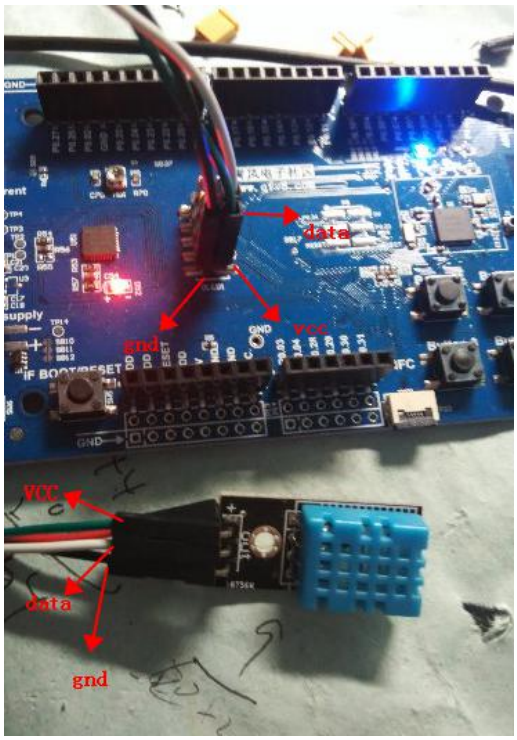
主函数需要调用几个函数，比如 dht 的初始化，定时器的初始化，开启定时器定时等，这几个函数上面都已经编写成功，代码如下：

```
int main(void)
{
    // Initialize.
    leds_init();
    timers_init();
    while(dht_Init())
    {
    nrf_delay_ms(1000);
    }

//    buttons_init();
    ble_stack_init();
    gap_params_init();
    services_init();
    advertising_init();
    conn_params_init();
    application_timers_start();//开启定时器
    // Start execution.
    advertising_start();

    // Enter main loop.
    for (;;)
    {
        power_manage();
    }
}
```

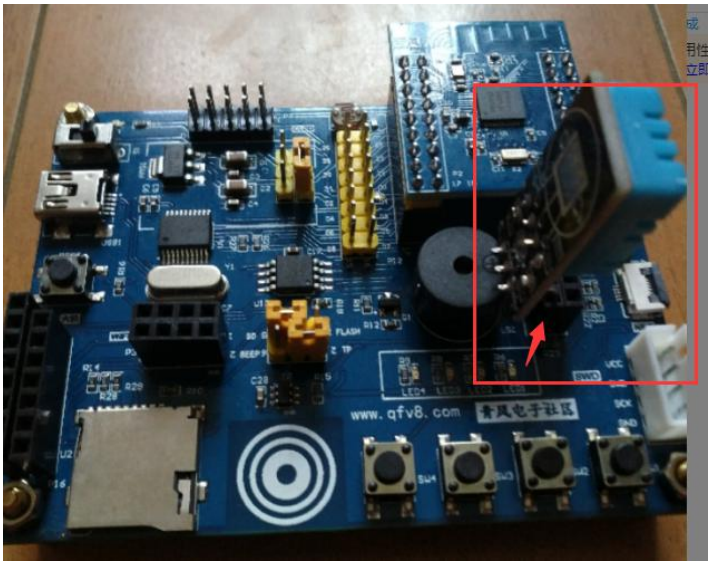
2 应用与调试

2.1 下载

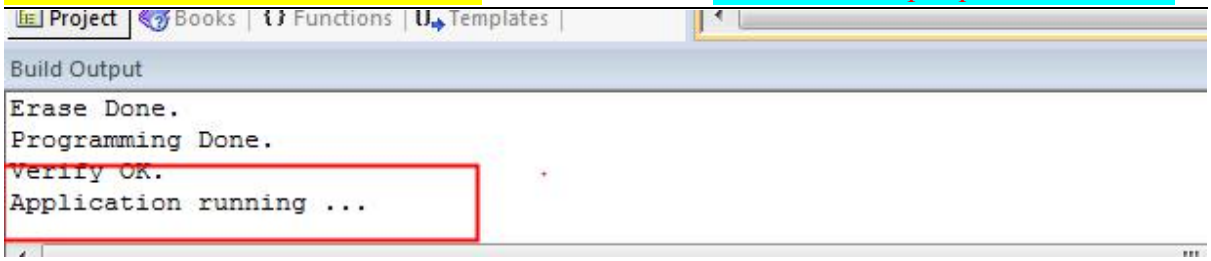
开发板接上传感器，如下图连接，连接传感器的三个接口：VDD,GND,data



EK 开发板如下所示接：



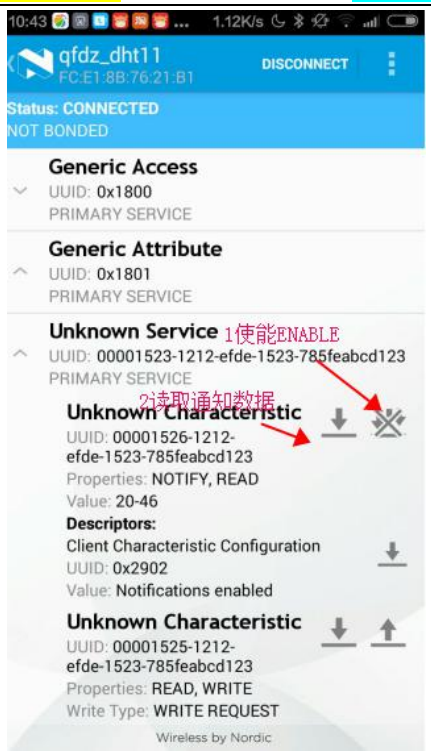
打开 NRFgo 进行下载，首先整片擦除，后下载协议栈:下载完后可以下载工程，首先把工程编译一下，通过后点击 KEIL 上的下载按键。下载成功后提示如图，程序开始运行，同时开发板上广播 LED 开始广播：



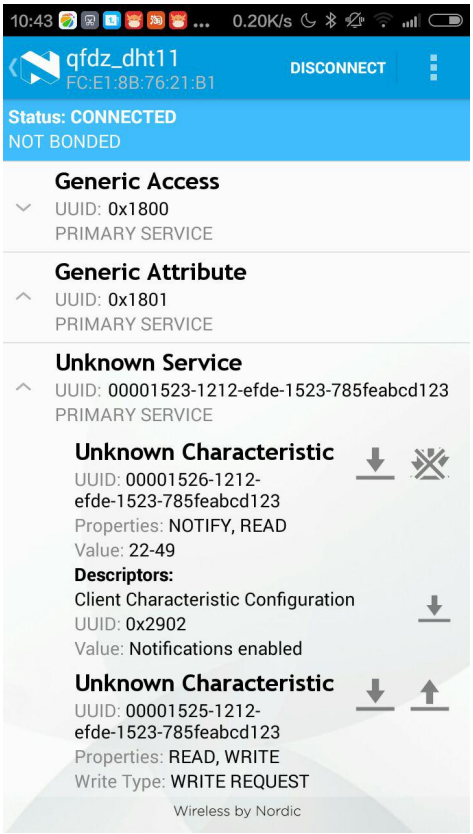
2.2 测试

打开手机 APP 软件 MCP,如下图所示,搜索到蓝牙信号 qfdz_dht11,如下图所示：

点击连接后如下图所示，找到私有服务，按下下图步骤，首先使能通知，然后读取通知值：



一段时间后数据发生自动更新：



如上图所示，采集的 VALE 值显示如图：20-46， data[1]为温度值;data[2]

湿度值；