谷雨物联网
专注无线联网通信

BLE系列Arduino测试板使用说明

使用Arduino开发的模组测试板

2021-07-22

无锡谷雨电子有限公司

Wuxi Ghostyu Electronics Co,.LTD

目录

Arduino是非常受欢迎的、便捷灵活、方便上手的开源硬件平台，包括多种多样的开源硬件和软件开发工具。即便没有任何硬件或软件开发经验的人都可以在短时间快速上手，这便是我们选择Arduino作为模组测试板主控器的原因。通过Arduino，可以快速搭建BLE系列模组的实测环境。

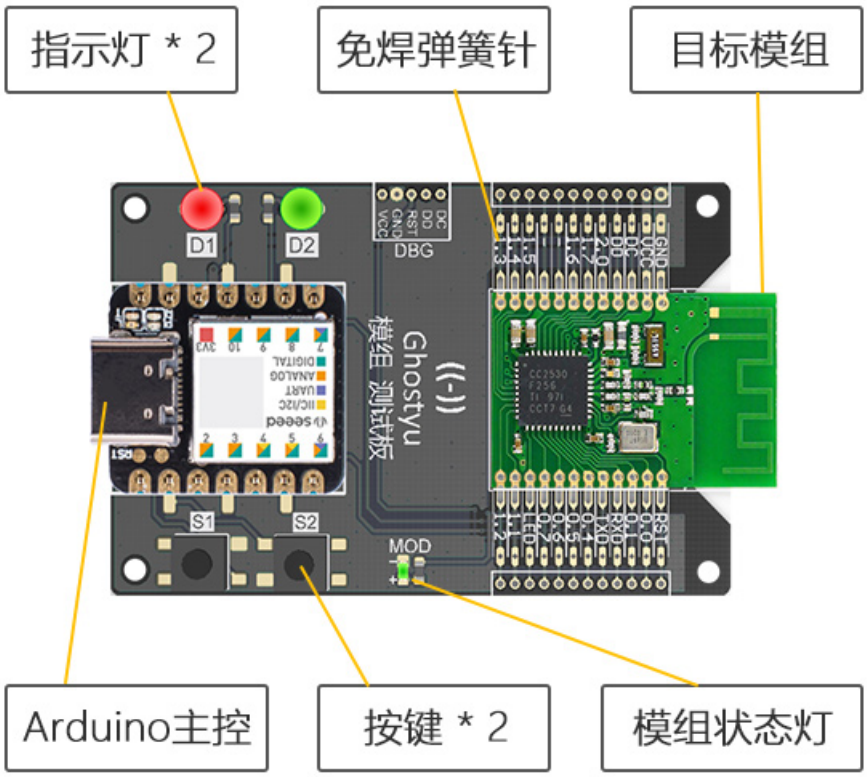
本文详细讲解如何使用Arduino模组测试板（测试BLE从机模块），以及在此基础上的二次开发。

下文一律将Arduino测试板简称为：ARD测试板。

1 测试板简介

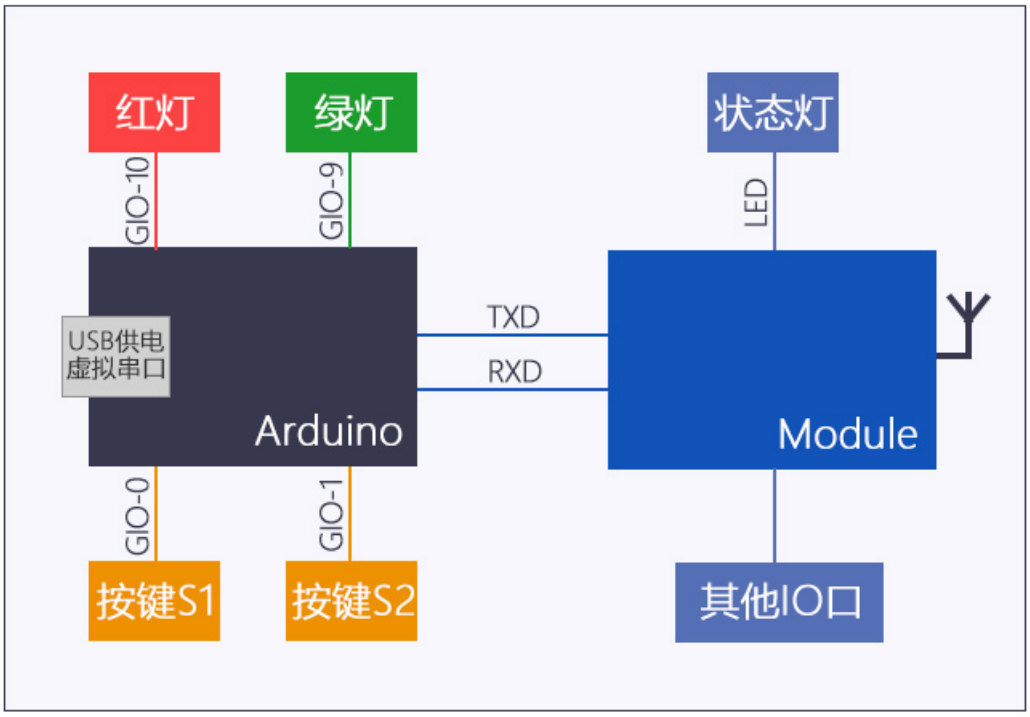
1.1 测试板实物图

测试板图片如下（以MO测试板为例，其他型号的模组的测试板与此类似）。包括Ardduino控制的两个LED，两个按键，以及使用弹簧顶针免焊连接的无线模组。



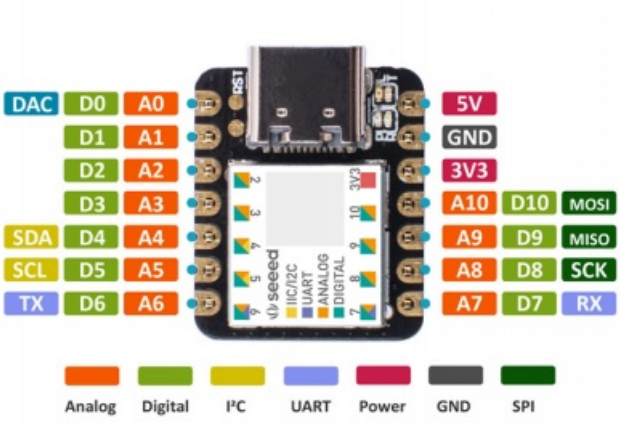
1.2 功能框图

功能框图如下，Arduino的串口1与无线模块连接，负责解析模组转发的来自手机的无线数据，然后转化成灯的控制，同时采集按键状态并通过无线蓝牙发送给手机。



1.3 Arduino选型

BLE系列Arduino测试板采用的是 Seeed Studio 公司的Seeed XIAO硬件。硬件如下图所示，它具有一路串口，共10路GPIO口，其小巧的尺寸很适合我们的测试应用。



大家也可以根据自身情况，选择其他开源硬件，因为Arduino有一套标准的硬件接口和软件代码接口，理论上任意的Arduino硬件均可以实现本文描述的功能。

2 测试流程

测试全程不需要修改代码，全部按键和指示灯完成。

2.1 上电广播

ARD测试板上电后（通过USB供电），模组也将上电工作，上电后立即开始广播，此时可以通过手机App或者微信小程序来连接改从机。

ARD测试板上电后红灯和绿灯会同时闪烁一次，此时表示程序已经启动完成。

2.2 建立连接

在App或者微信小程序里找到从机对应的设备，然后点击连接，连接成功后，ARD测试板红灯亮起，表示模块已经收到了已连接的事件通知（+CONN指令通知）。

2.3 点灯测试

连接后在App里找到UUID为0xFFE0的服务，其中0xFFE1是写通道，用于手机向蓝牙模块发送数据，0xFFE2是数据通知通道，用于模块向手机发送数据。

向0xFFE1通道发送自定义的灯控命令：`ON`，表示开启绿灯；`OFF`，表示关闭绿灯。此时观察ARD测试板，可以看到对应的灯控状态。

2.4 按键测试

首先在手机App里找到0xFFE2，点击打开通知的按钮，否则无法收到模块上报的数据，有些App会自动打开Notify属性的通知，有些则需要手动打开。

按下按键S1，手机上将收到：`#KEY1` 的数据通知。

按下按键S2，手机上将收到：`##KEY2` 的数据通知。

2.5 断开连接

在App里断开蓝牙连接后，ARD测试板红灯熄灭，表示收到了断开连接的事件通知（+DISCONN指令通知）

3 代码解释

下面对主要代码进行讲解，用户可以在此代码基础上增加更多功能。

3.1 Arduino代码框架介绍

arduino源码由一个或者多个后缀是ino的文件组成，C++的编程语法。本测试例子由三个文件组成，如下图所示。

main.ino 是项目主文件，key.ino 和 key.h 是按键服务的独立代码。下面主要介绍 main.ino

在arduino程序中，由两个主要的函数，一个是setup()，另外一个是 loop()，setup是硬件启动后只调用一次的函数，在这里可以放置硬件相关的初始化函数，loop循环调用的一个函数，相当于普通单片机中main函数里的 while(1){}中的内容。结构虽然简单，但可以完成我们的所有测试动作。下面将

分解介绍各个硬件功能。

3.2 setup函数

setup函数代码如下，主要执行硬件初始化，以及两个按键的特殊功能处理。

```
void setup() {
  // 串口初始化
  UartInit();

  // 两个用户LED灯。
  LedInit();

  // 按键初始化，参数是按键处理的回调函数。
  HalKeyInit((void*)handleKeyEvent);

  // 打印一条启动信息
  log("System Running...");

  // 两个LED等闪烁一下，指示程序已经运行，同时可以检查LED灯是否有异常。
  LedRedCtrl(true);
  LedGreenCtrl(true);
  delay(50);
  LedRedCtrl(false);
  LedGreenCtrl(false);

  // 等待连接的模组正常运行。
  delay(500);
}
```

3.3 loop函数

```
// 注意，所有轮训不能阻塞，需要使用状态机。
void loop() {
  // 1-USB串口轮询
  handleSerialLoop();
  // 2-模块串口轮询
  handleSerial1Loop();

  // 3-按键轮训
  HalKeyPoll();

  // 4-其他任务轮询
  // code here
}
```

op函数中执行各个轮询任务，建议使用状态机的方式开发程序。

3.4 红灯绿灯

代码分为两个部分，端口初始化和端口高低电平的控制。

```
// 红灯使用 GPIO-10，绿灯使用 GPIO-9
#define LED_RED 10
#define LED_GREEN 9

// 端口初始化，此函数在setup函数中调用。
void LedInit()
{
  // 红灯，
  pinMode(LED_RED, OUTPUT);
  digitalWrite(LED_RED, LOW);

  // 绿灯
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_GREEN, LOW);
}

// 端口高低电平控制，在需要的时候调用
// true 亮灯; false 关灯
void LedRedCtrl(bool on)
{
  if(on){
    digitalWrite(LED_RED, HIGH);
  }else{
    digitalWrite(LED_RED, LOW);
  }
}

// true 亮灯; false 关灯
void LedGreenCtrl(bool on)
{
  if(on){
    digitalWrite(LED_GREEN, HIGH);
  }else{
    digitalWrite(LED_GREEN, LOW);
  }
}
```

3.5 按键功能

按键部分为三个部分，首先是端口初始化，其次是轮询函数，最后是按键动作处理。

1、端口初始化

```
// 按键初始化，此函数在 setup中调用。
void HalKeyInit(void *cback){
  // 输入上拉，低电平触发按键。
  pinMode(PIN_BTN_S1, INPUT_PULLUP);
  pinMode(PIN_BTN_S2, INPUT_PULLUP);

  // 设置回掉函数，用来处理按键动作。
  pHalKeyProcessFunction = (halKeyCBack_t)cback;
}
```

2、轮询函数

```
// 按键轮询，此函数在loop中循环调用
void HalKeyPoll (void){
  uint8_t keys = 0;
  uint8_t notify = 0;
  static bool delayRead = false;
  static unsigned long time = 0;
  keys = HalKeyRead();
  if (keys == halKeySavedKeys && delayRead == false) {
    // Exit - since no keys have changed
    return;
  } else {
    // notify = 1;
  }

  // 第一次触发。保存时间。若是第二次，则delayRead=true
  if(delayRead == false){
    // 记录这一次的按键。
    halKeySavedKeys = keys;
    time = millis();
    delayRead = true;
    return ;
  }
  // 去抖动，准备第二次读取数据。
  if((millis() - time) > 80){
    if(halKeySavedKeys == keys){
      // 延时后，重新检测和上次的结果相同。则
      notify = 1;
    }else {
      // 重新开始
      delayRead = false;
      halKeySavedKeys = 0;
    }
  }
  // Invoke Callback if new keys were depressed
  if (notify && (pHalKeyProcessFunction))
  {
    static uint8_t lastKeys = 0;
    delayRead = false;
    // 按键按下时，触发 keys & KEY1 == true && lastKeys & KEY1 == false 事件
    // 按键释放时，触发 keys & KEY1 == false && lastKeys & KEY1 == true 事件
    (pHalKeyProcessFunction) (keys, lastKeys);
    lastKeys = keys;
  }
}
```

3、按键动作处理

```
// 按键回调函数，在这里处理按键任务。
// 关于按键的实现机制请阅读 key.ino 中的代码
void handleKeyEvent(uint8_t keys, uint8_t lastKeys){
  unsigned long time = 0;
  uint8_t ret = 0;

  // 按键S1
  ret = KeyPressed(keys,lastKeys,HAL_KEY_BUTTON_S1, &time);
  if(ret == KEY_PRESSED){
    uint8_t msg[] = "#Key1";
    ModuleWrite(msg, sizeof(msg) / sizeof(msg[0]));
    log("Key1 Pressed.");
  }

  // 按键S2
  ret = KeyPressed(keys, lastKeys, HAL_KEY_BUTTON_S2, &time);
  if(ret == KEY_PRESSED){
    // 通过BLE发送出去。
    uint8_t msg[] = "##Key2";
    ModuleWrite(msg, sizeof(msg) / sizeof(msg[0]));
    log("Key2 Pressed.");
  }
}
```

3.6 USB虚拟串口和模块串口

USB虚拟串口接收函数轮询电脑端是否有数据待接收，如果有，则直接转发给模块串口连接的串口1，同时，与模块串口连接的串口1若有数据，也将转发给USB虚拟串口，因此，可以理解为Arduino的USB虚拟串口和模块串口间接直连，方便在电脑上使用BLE模组的上位机调试软件。

模块串口接收函数轮询是否由模块数据待接收，如果接自定义的灯控数据则解析并执行对应的动作，同时，模块收到的数据将转发给Arduino的USB虚拟串口，可以在电脑上监测收到的数据。

串口初始化函数：

```
// 串口初始化，该函数在setup中调用
void UartInit()
{
  // USB串口
  Serial.begin(38400);
  // 轮询一次的超时时间，单位为毫秒
  // 9600波特率不能低于4ms
  Serial.setTimeout(4);

  // 模块串口
  Serial1.begin(38400);
  // 9600波特率不能低于4ms
  Serial1.setTimeout(4);
}
```

串口轮询函数：

```
// 接收USB的串口数据
void handleSerialLoop(){
  if(Serial.available()){
    // 读一帧数据
    int len = Serial.readBytes(buf, 256);
    // 转发给模组
    Serial1.write(buf, len);
  }
}
// 接收模块的串口数据
void handleSerial1Loop(){
  if(Serial1.available()){
    int len = Serial1.readBytes(buf1, 256);
    // 转发给USB
    Serial.write(buf1, len);

    // 解析zigbee数据，查看是否是灯孔数据
    ZigBeeDataParse(buf1,len);
  }
}
```

3.7 蓝牙数数据解析

蓝牙数据在串口1的接收函数中处理，函数名称是：BLEDataParse

```
// 处理两种数据： 1自定义的点灯命令；2蓝牙连接/断开事件
// 1-自定义点灯命令，
//      ON 点亮绿灯
//      OFF 关闭绿灯
// 2-蓝牙连接/断开事件
//      +CONN=x,xxx 表示蓝牙已连接
//      +DISCONN=x,xxx,xxxx 表示蓝牙已断开
void BlueDataParse(char *data, int len)
{
  String msg = String(data);
  // 简单的判断指令头，不做详细的验证。
  if(msg.startsWith("\r\n+CONN=")){
    // 表示已连接，点亮红灯。
    LedRedCtrl(true);
  }else if(msg.startsWith("\r\n+DISCONN=")){
    // 表示已连接，点亮红灯。
    LedRedCtrl(false);
  }

  // 对比是否是期望的指令。
  if(msg.equalsIgnoreCase("ON")){
    LedGreenCtrl(true);
  }else if(msg.equalsIgnoreCase("OFF")){
    LedGreenCtrl(false);
  }
}
```

4 总结

以上是基于Arduino和我们BLE模块开发的一个简单灯控的应用，前后不过百行代码，耗时不超过10分钟。当然，我们的BLE模组还有更多灵活的应用，同时arduino也能够写出更加复杂的应用案例。

后面我们也将开发更多的实际案例，方便大家快速集成BLE模组。

5 联系方式

公司：无锡谷雨电子有限公司

电话：0510-83486610

网址：iotxx.com