蓝牙 **5.2 BLE** 模块

BM7701-00-1
Arduino Library V1.0.2 说明

版本：V1.10　日期：2024-05-10

www.bestmodulescorp.com

目录

简介

BM7701-00-1 是倍创推出的蓝牙 5.2 BLE 模块，使用 UART 通信方式。本文档对 BM7701-00-1 的 Arduino Lib 函数、Arduino Lib 安装方式进行说明；范例使用 BMC77M001 模块，演示了数据透传功能。

适用型号：

型号	说明
BM7701-00-1	蓝牙 5.2 BLE 模块
BMC77M001	板载 BM7701-00-1 模块

Arduino Lib 函数

Arduino Lib 名称：BM7701-00-1		Lib 版本：V1.0.2
构造函数 & 初始化		
1	BM7701_00_1(HardwareSerial*theSeriall=&Serial)	
	描述	构造函数，使用硬件串行接口
	参数	*theSerial：选择硬件串行接口 (默认串行接口)
	返回值	—
	备注	—
2	BM7701_00_1(uint16_t rxPin, uint16_t txPin)	
	描述	构造函数，使用软件串行接口
	参数	rxPin：RX 引脚，连接 BM7701-00-1 的 UARTn_TX 引脚 (n=1/2) 或 BMC77M001 的 TX 引脚 txPin：TX 引脚，连接 BM7701-00-1 的 UARTn_RX 引脚 (n=1/2) 或 BMC77M001 的 RX 引脚
	返回值	—
	备注	—
3	void begin(uint32_t baud=115200)	
	描述	初始化模块，配置波特率
	参数	baud：波特率，范围 9600/14400/19200/38400/57600/115200/125000
	返回值	void
	备注	波特率默认 115200
功能函数		
4	bool setAdvCtrl(bool ctrl)	
	描述	设置广播开启 / 关闭
	参数	ctrl：广播状态 true：开启 false：关闭
	返回值	执行情况： true：成功 false：失败
	备注	开启广播后需要等待 600~800ms 校准时间

5	bool disconnect()	
	描述	断开蓝牙设备连接
	参数	—
	返回值	执行情况： true：成功 false：失败
	备注	断开后将等待 5ms 的缓冲时间
6	bool writeData(uint8_t dataBuf[], uint8_t length)	
	描述	发送数据给连接蓝牙模块的设备
	参数	dataBuf[]：需要发送的数据 length：需要发送的数据的长度
	返回值	执行情况： true：成功 false：失败
	备注	data[] 的字节长度需要 ≤ 57bytes
7	bool readData(uint8_t receiveBuff[], uint8_t &length)	
	描述	读取设备发送的数据
	参数	receiveBuff[]：读取到的数据 &length：读取到的数据的长度
	返回值	执行情况： true：成功 false：失败
	备注	receiveBuff[] 的长度建议设置 ≥ 57bytes
配置函数		
8	bool setAdvIntv(uint16_t min, uint16_t max, uint8_t advMap)	
	描述	设置广播参数
	参数	min：广播间隔最小时间参数，范围：0x0020~0x4000 max：广播间隔最大时间参数，范围：0x0020~0x4000 advMap：广播通道 (可同时开启多个通道) bit0=1：channel37 (2402MHz) 开关控制 bit0=1：开启 bit0=0：关闭 bit1=1：channel38 (2426MHz) 开关控制 bit1=1：开启 bit1=0：关闭 bit2=1：channel39 (2480MHz) 开关控制 bit2=1：开启 bit2=0：关闭
	返回值	执行情况： true：成功 false：失败
	备注	广播间隔最小时间计算公式：$t_{Adv\text{-}min}$=min×0.625ms 广播间隔最大时间计算公式：$t_{Adv\text{-}max}$=max×0.625ms

9	bool setAdvData(uint8_t appendName, uint8_t length, uint8_t advData[])	
	描述	设置广播数据
	参数	appendName： 0x00 (APPEND_NAME)：附带数据 0x10 (NO_APPEND_NAME)：不附带数据 length：advdata[] 的字节长度 advData[]：数据，advData[] 的字节长度需要 ≤ 31bytes appendName=0x00 时，包含 advData + 其他属性信息 appendName=0x10 时，仅包含 advData
	返回值	执行情况： true：成功 false：失败
	备注	正常使用下无需修改 若要修改 advData 和其他信息详细参考 BLE_API 规格书 4.9API_AdvData
10	bool setScanData(uint8_t length, uint8_t scanData[])	
	描述	当广播数据被 app 扫描到时，模块要响应给手机 APP 的数据
	参数	length：scanData[] 的字节长度 scanData[]：格式为数据字节长度 + 数据类型 + 数据，需要 ≤ 31byte
	返回值	执行情况： true：成功 false：失败
	备注	正常使用下无需修改 数据类型若要修改参考 BLE_API 规格书 4.10API_ScanData
12	bool setConnIntv(uint16_t interval)	
	描述	连接成功后，设置蓝牙模块和手机 APP 的周期性通讯间隔时间
	参数	interval：通讯间隔时间参数，范围：0x0006~0x0c80
	返回值	执行情况： true：成功 false：失败
	备注	周期性通讯间隔时间 t_{inv}=interval×1.25ms，与函数 setConnIntv1 的设置相同
13	bool setConnIntv(uint16_t minIntv, uint16_t maxIntv, uint16_t latency, uint16_t timeout)	
	描述	连接成功后，设置蓝牙模块和手机 APP 的周期性通讯间隔时间最大值和最小值，设备延迟发送时间，连接超时时间
	参数	minIntv：最小间隔时间参数，范围：0x0006~0x0c80 maxIntv：最大间隔时间参数，范围：0x0006~0x0c80 latency：设备延迟发送的时间参数，范围：0x0000~0x01f3 timeout：连接超时时间参数，范围：0x000a~0x0c80
	返回值	执行情况： true：成功 false：失败
	备注	最小间隔时间 $t_{inv\text{-}min}$=minIntv×1.25ms 最大间隔时间 $t_{inv\text{-}max}$=maxIntv×1.25ms (最终连接间隔时间 t_{inv} 处于最大和最小区间) 设备延迟发送的时间 t_{del}=latency×t_{inv} 连接超时时间 t_{out}=timeout×10ms

14	bool setName(uint8_t length, uint8_t name[])	
	描述	设置蓝牙模块广播时被检测到显示的名称
	参数	length：蓝牙名称字节数 name[]：蓝牙名称
	返回值	执行情况： true：成功 false：失败
	备注	name[] 的字节长度需要 ≤ 31bytes
15	bool setAddress(uint8_t address[])	
	描述	设置蓝牙 mac 地址
	参数	address[]：蓝牙 mac 地址 (6byte)
	返回值	执行情况： true：成功 false：失败
	备注	—
16	bool setFeature(uint8_t cmdFlag, uint32_t codeFeature)	
	描述	设置模块数据更新方式
	参数	cmdFlag：数据写入方式 0x00 (FEATURE_DIR)：覆盖上一个值 0x10 (FEATURE_OR)：与上一个值以“或”的方式写入 0x20 (FEATURE_AND)：与上一个值以“与”的方式写入 codeFeature：4byte，具体参考 BLE_API 规格书 4.14API_Feature
	返回值	执行情况： true：成功 false：失败
	备注	—
17	bool setPowerSaving(uint8_t mode)	
	描述	设置模块节能模式
	参数	mode：节能模式 0x00：睡眠 (默认) 0x01：深度睡眠 0x15：关机，保持深度睡眠，唤醒将强制复位，RAM 中的所有设置复位为默认值
	返回值	执行情况： true：成功 false：失败
	备注	在睡眠和深度睡眠模式下唤醒，必须发送伪字节 (至少 100μs 长)；解除休眠使用唤醒函数 wakeUp()

18	bool setWhiteList(bool erase, uint8_t whiteListAddr[], uint8_t mask[])	
	描述	修改白名单
	参数	erase：要对白名单进行的动作 true：添加 false：删除 whiteListAddr[]：要添加到白名单的设备地址 (6byte) mask[]：地址掩码，模块内部默认为 0xFFFFFFFF
	返回值	执行情况： true：成功 false：失败
	备注	具体参考 BLE_API 规格书 4.20API_WhiteList
19	bool setTXpower(uint8_t power)	
	描述	设置发射功率
	参数	power：发射功率档位，范围：0x00(min)~0x0F(max)
	返回值	执行情况： true：成功 false：失败
	备注	例如： power=0x00，TX 发射功率 = -32.5dbM power=0x05，TX 发射功率 = -11.5dbM power=0x0A，TX 发射功率 = +0.5dbM power=0x0F，TX 发射功率 = +3.5dbM
20	bool setCrystalOffset(uint8_t cload)	
	描述	设置外部 16MHz 晶振的射频输出频率偏移量
	参数	cload：射频输出频率偏移量，范围 0x00~0x0f
	返回值	执行情况： true：成功 false：失败
	备注	例如： cload=0x00，晶振频率 = 15.99985MHz cload=0x05，晶振频率 = 16.00004MHz cload=0x0A，晶振频率 = 16.00031MHz cload=0x0F，晶振频率 = 16.00074MHz
21	bool restoreDefault()	
	描述	软件复位，复位后恢复默认值
	参数	—
	返回值	执行情况： true：成功 false：失败
	备注	—
22	bool reset()	
	描述	软件复位，复位后维持之前的设定值
	参数	—
	返回值	执行情况： true：成功 false：失败
	备注	—

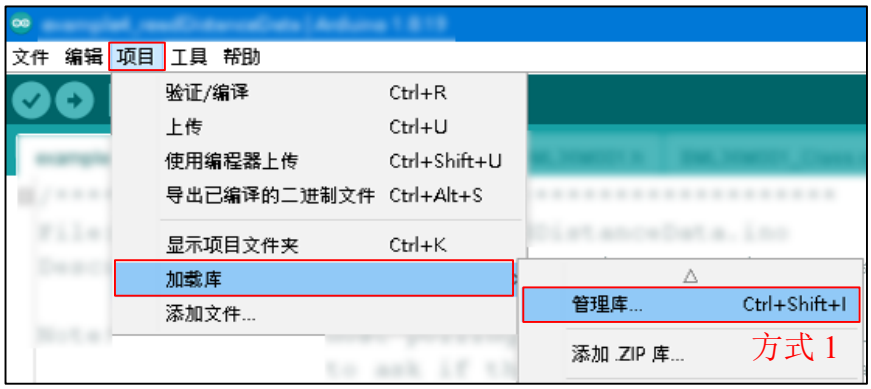
23	void wakeUp();	
	描述	唤醒处于休眠模式的蓝牙模块
	参数	—
	返回值	void
	备注	需要等待 30ms，以完全唤醒蓝牙模块
24	bool sendCMD(uint16_t type, uint8_t cmd_flag, uint8_t cmd_length, uint8_t value[])	
	描述	通用命令发送函数，向 BM7701-00-1 模块发送带数据的命令
	参数	type：类型 (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag ： API 命令的 flag cmd_length：type+value+cmd_flag 的字节长度 value[]：命令数据 Buffer
	返回值	执行情况： true：成功 false：失败
	备注	24~27 为 BLE API CmdType2 的通用型函数，参数以及对应的数据参考 BLE_API 规格书
25	void sendCMD_NoResponse(uint16_t type, uint8_t cmd_flag, uint8_t cmd_length, uint8_t value[])	
	描述	通用命令发送函数不等待响应
	参数	type：类型 (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag：API 命令的 flag cdm_length：type+value+cmd_flag 的字节长度 value[]：命令数据 Buffer
	返回值	void
	备注	—
26	bool readCMD(uint16_t type, uint8_t cmd_flag)	
	描述	通用命令读取函数
	参数	type：类型 (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag：API 命令的 flag
	返回值	执行情况： true：成功 false：失败
	备注	—
27	void readCMD_NoResponse(uint16_t type, uint8_t cmd_flag)	
	描述	通用命令读取函数不等待响应
	参数	type：类型 (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag：API 命令的 flag
	返回值	void
	备注	—

Arduino Lib 下载及安装

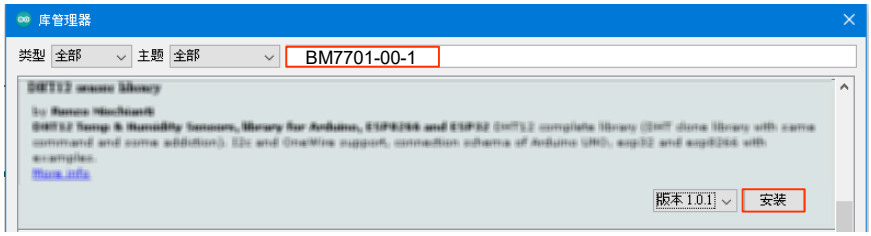
BM7701-00-1 Library：可参考下面两种方法安装 BM7701-00-1 的 Arduino Library

方式 1：搜索安装

搜索安装：Arduino IDE → 项目 → 加载库 → 管理库 → 搜索 BM7701-00-1 → 安装



搜索安装流程 **1**

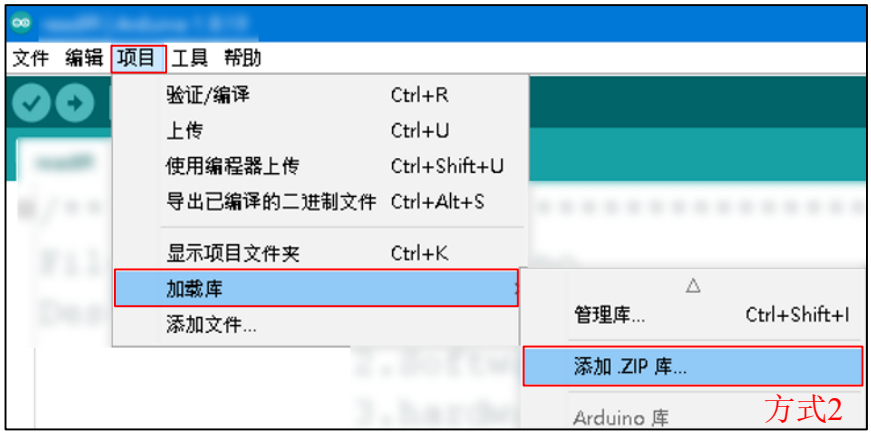


搜索安装流程 **2**

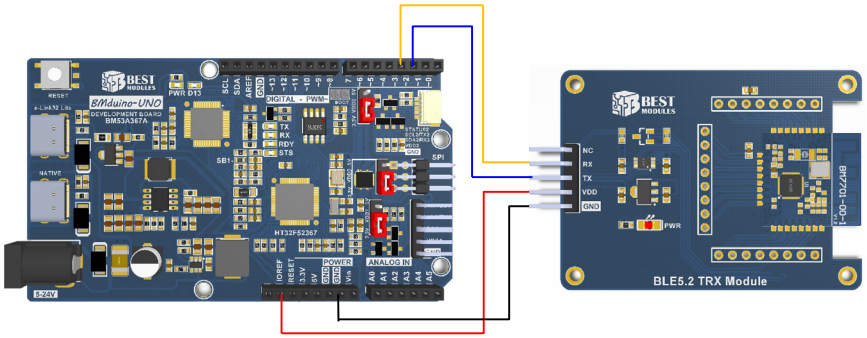
方式 2：添加 .ZIP 库，需提前下载 .ZIP 库

下载方法：打开倍创官方网站 (https://www.bestmodulescorp.com/BM7701-00-1.html)“文件”目录下的 Arduino 范例程序 (BM7701-00-1 Library)。

添加 .ZIP 库：Arduino IDE → 项目 → 加载库 → 添加 .ZIP 库 ...



Arduino 范例



实物连接示意图

范例：writeAndRead

范例实现功能：模块连接手机 APP“BLEDemo”或“BCBLEDemo”，APP 上按下按键后，模块接收到对应数据，打印至串口监视器，再发送对应数据给 APP，使 APP 上的 LED 亮。APP 上再次按下此按键则 LED 灭。

1. 范例打开：文件→示例→Lib 选择(BM7701-00-1)→选择范例(writeAndRead)
2. 示例说明：
   a. 调用库函数，初始化模块，参数设定包括按键、LED 和部分蓝牙参数设定

```
#include <BM7701-00-1.h>
BM7701_00_1          BC7701(2, 3);          // 使用软件 UART
#define TX_POWER     0x0F                   // TX Power
#define XTAL_CLOAD   0x04                   // 16MHz 晶振负载
#define ADV_MIN      100                    // 广播间隔最小值
#define ADV_MAX      100                    // 广播间隔最大值
#define CON_MIN      30                     // 连接间隔最小值
#define CON_MAX      30                     // 连接间隔最大值
#define CON_LATENCY  00                     // 连接延时
#define CON_TIMEOUT  300                    // 连接超时
uint8_t BDAddress[6]={0x11, 0x22, 0x33, 0x44, 0x55, 0x66}; // 设备地址
uint8_t BDName[]={'B', 'M', 'C', '7', '7', 'M', '0', '0', '1'};
                                            // 设备名称
uint8_t Adata[]={0x02, 0x01, 0x06};         // 广播数据
uint8_t Sdata[]={0x03, 0x02, 0x0f, 0x18};   // 被扫描后响应数据
#define BUTTON_CONSISTENCY_DURATION 6
#define BUTTON_REPEAT1_DURATION    (600/BUTTON_CONSISTENCY_DURATION)
#define BUTTON_REPEAT2_DURATION    (150/BUTTON_CONSISTENCY_DURATION)
#define INVERT_TIME                 500
bool board_connect=false;
bool board_receive=false;
bool button_change=false;
bool board_conIntv=false;
bool txf=false;
uint8_t Status;
uint8_t flag=0;
```

```
uint8_t count=0;
uint8_t sel=1;
uint8_t receiveBuf[256]={0};
KEY_MESSAGE Keymessage;
```

b. 上电复位后，先延时 60ms 等待上电完成才能执行 begin( ) 函数。

```
void setup() {
  delay(60);                        // 等待上电完成
  Serial.begin(9600);               // 配置串口监视器
  BC7701.begin(BAUD_115200);        // 模块初始化
  setup()
  while (sel != 10)
  {
    switch (sel)
    {
      case 1: if (BC7701.setAddress(BDAddress) == true) sel++;
                                    // 设置蓝牙地址
        else sel=0xFF; break;
      case 2: if (BC7701.setName(sizeof(BDName), BDName) == true)
      sel++;
        else sel=0xFF; break;
      case 3: if (BC7701.setAdvIntv(ADV_MIN/0.625, ADV_MAX/0.625, 7)
      == true) sel++;
        else sel=0xFF; break;  // 设置广播间隔
      case 4: if (BC7701.setAdvData(APPEND_NAME, sizeof(Adata),
      Adata) == true) sel++;
        else sel=0xFF; break;  // 设置广播数据
      case 5: if (BC7701.setScanData(sizeof(Sdata),
      Sdata) == true) sel++;
        else sel=0xFF; break;  // 设置广播被扫描后的响应数据
      case 6: if (BC7701.setTXpower(TX_POWER) == true) sel++;
        else sel=0xFF; break;  // 设置 TX 发射功率
      case 7: if (BC7701.setCrystalOffset(XTAL_CLOAD) == true) sel++;
        else sel=0xFF; break;  // 设置外部晶振偏移值
      case 8: if (BC7701.setFeature(FEATURE_DIR, AUTO_SEND_SATUS)
      == true) sel++;
        else sel=0xFF; break;  // 设置数据更新方式
      case 9: if (BC7701.setAdvCtrl(ENABLE) == true) sel++;
        else sel=0xFF; break;  // 使能广播功能
      case 0xFF: digitalWrite(13,HIGH);break; // 配置失败，LED13 亮
    }
  }
  delay(650);                                 // 等待广播完全开启
}
```

c. 判断蓝牙模块状态，如果连接成功就可以接收数据，判断数据和连接状态。

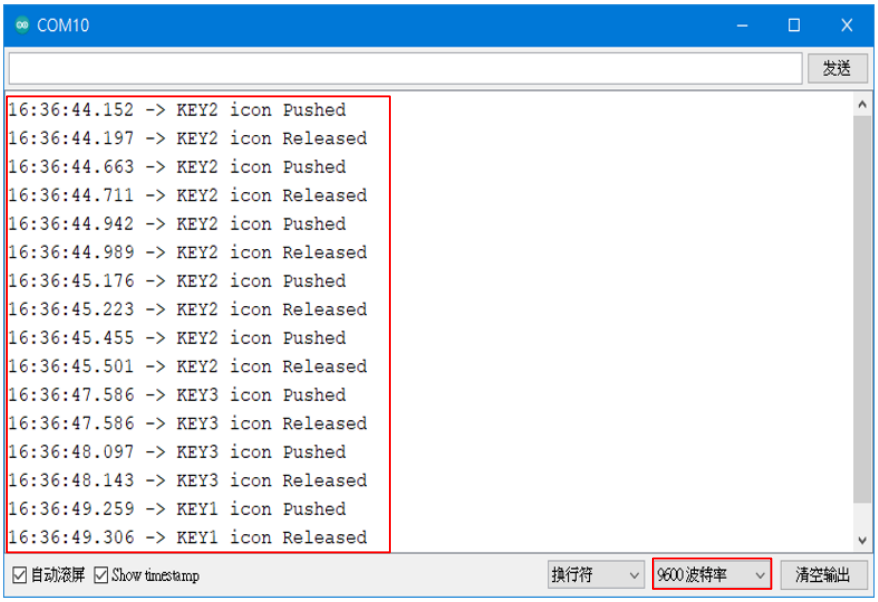
```
void loop()
{
  Status=bleProcess();                          // 读取 BLE 当前状态
  if (Status)
  {
    switch (Status)
    {
      case API_CONNECTED:                       // 设备首次连接
        if (board_connect == false)
        {
        board_connect=true;
        board_receive=false;
        }
        break;
      case API_DISCONNECTED:                    // 设备连接失败
        board_connect=false;
        board_receive=false;
        board_conIntv=false;
        break;
      case DATA_RECEIVED:                       // 设备接收到数据
        if (board_connect == true)
        {
        digitalWrite(13,LOW);
        board_receive=true;
        }
        break;
      case API_ERROR:                           // 错误
        digitalWrite(13,HIGH);
        break;
    }
  }
```

d. 连接成功后，设置连接间隔，接收数据来做对应动作

```
if (board_connect == true)                      // 连接状态正确
{
  if (board_conIntv == false)                   // 判断是否配置过连接间隔
  {
    BC7701.wakeUp();
    if (BC7701.setConnIntv(CON_MIN/1.25, CON_MAX/1.25,
    CON_LATENCY, CON_TIMEOUT) == true)          // 配置连接间隔
    board_conIntv = true;
  }
  if (board_receive == true)                    // 接收到数据
  {
    board_receive = false;                      // 清空标志
    if (receiveBuf[3] == 0xB0)                  // 数据包帧头
    {
      switch (receiveBuf[4])                    // 按键数据位
      {
        case 0x11:
          count=1;
          Serial.println(“KEY1 icon Pushed”);// 按键 1 按下
          break;
```

```
          case 0x10:
            count=2;
            Serial.println(“KEY1 icon Released”);  // 按键 1 松开
            break;
          case 0x22:
            count=1;
            Serial.println(“KEY2 icon Pushed”);    // 按键 2 按下
            break;
          case 0x20:
            count=2;
            Serial.println(“KEY2 icon Released”);  // 按键 2 松开
            break;
          case 0x44:
            count=1;
            Serial.println(“KEY3 icon Pushed”);    // 按键 3 按下
            break;
          case 0x40:
            count=2;
            Serial.println(“KEY3 icon Released”);  // 按键 3 松开
            break;
        }
        if (receiveBuf[4]!=0 && count==2&& flag==0)  // 接收到 2 笔数据，
                                                     // 发送亮灯数据
        {
          Keymessage.key = receiveBuf[4]>>4;
          Keymessage.key += receiveBuf[4];
          Keymessage.serial ++;
          Keymessage.checksum=0xB1 ^ Keymessage.key ^ Keymessage.
          serial;
          BC7701.writeData((uint8_t*)&Keymessage,3);
          flag=1;
          receiveBuf[4] = 0;
        }
        if (receiveBuf[4]!=0&&count==2&&flag==1)     // 再次接收到 2 笔数据，
                                                     // 已亮灯则灭灯
        {
          Keymessage.key = receiveBuf[4];
          Keymessage.serial ++;
          Keymessage.checksum=0xB1 ^ Keymessage.key ^ Keymessage.
          serial;
          BC7701.writeData((uint8_t*)&Keymessage,3);
          flag=0;
          receiveBuf[4]=0;
        }
      }
    }
  }
}
```

3. 打开串口监视器，波特率选择 9600；串口监视器收到 APP 的按键状态如下：

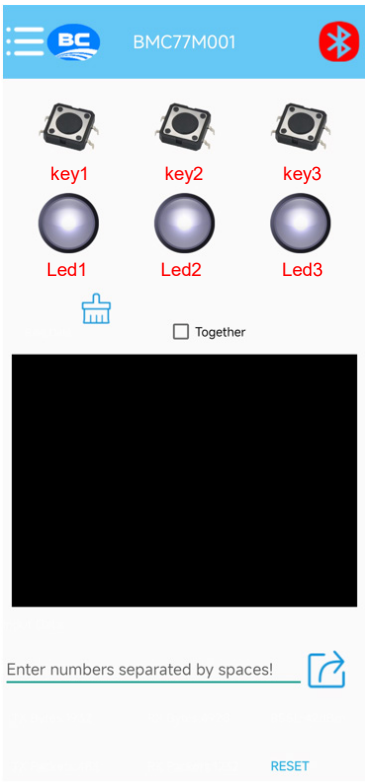


附录

1. 手机 APP 安装

Android

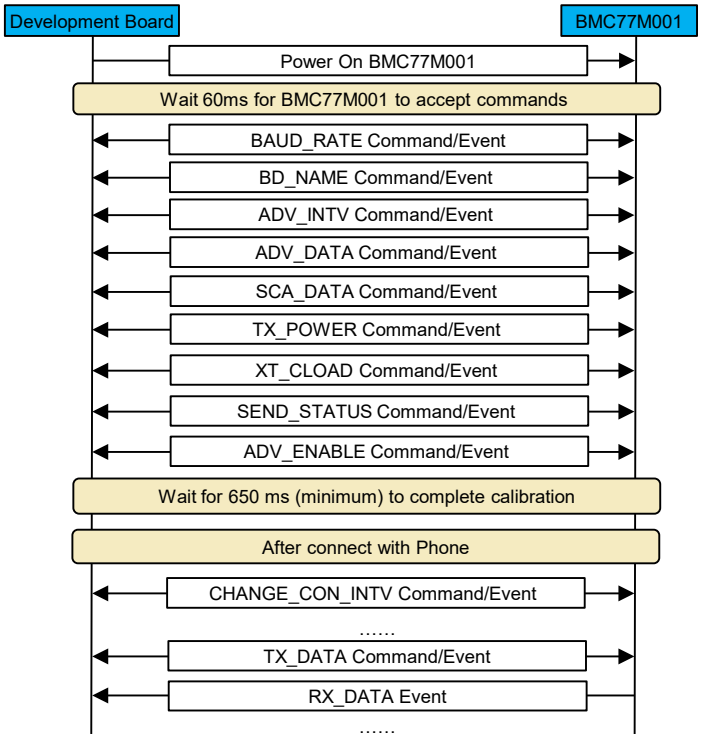
IOS



安装后 UI 显示

注：有部分手机型号，若出现 APP 打开后蓝牙列表以及主界面存在文字会遗失的现象，请尝试切换手机的显示模式 (日间模式 / 夜间模式 / 浅色模式 / 深色模式)，至可看到文字即可。

2. 命令流程


Development Board
BMC77M001
Power On BMC77M001
Wait 60ms for BMC77M001 to accept commands
BAUD_RATE Command/Event
BD_NAME Command/Event
ADV_INTV Command/Event
ADV_DATA Command/Event
SCA_DATA Command/Event
TX_POWER Command/Event
XT_CLOAD Command/Event
SEND_STATUS Command/Event
ADV_ENABLE Command/Event
Wait for 650 ms (minimum) to complete calibration
After connect with Phone
CHANGE_CON_INTV Command/Event
.....
TX_DATA Command/Event
RX_DATA Event
.....