Bluetooth 5.2 BLE module

BM7701-00-1
Arduino Library V1.0.2 Description

Revision: V1.10 Date: May 10, 2024

www.bestmodulescorp.com

Contents

Introduction

The Best Modules BM7701-00-1 is a Bluetooth 5.2 BLE module, which uses the UART communication method. This document provides the description of the BM7701-00-1 Arduino Lib functions and how to install the Arduino Lib. The example demonstrates the function of data transparent transmission with the BMC77M001 module.

Applicable model:

Model	Description
BM7701-00-1	Bluetooth 5.2 BLE module
BMC77M001	Integrated BM7701-00-1 module

Arduino Lib Functions

Arduino Lib Name: BM7701-00-1		Lib Version: V1.0.2
Constructors & Initialisation		
1	BM7701_00_1(HardwareSerial*theSeriall=&Serial)	
	Description	Constructor, use the hardware serial interface
	Parameter	*theSerial: Select hardware serial interface (default serial interface)
	Return Value	—
	Note	—
2	BM7701_00_1(uint16_t rxPin, uint16_t txPin)	
	Description	Constructor, use the software serial interface
	Parameter	rxPin: RX pin, connect to the BM7701-00-1 UARTn_TX pin(n=1/2) or the BMC77M001 TX pin txPin: TX pin, connect to the BM7701-00-1 UARTn_RX pin(n=1/2) or the BMC77M001 RX pin
	Return Value	—
	Note	—
3	void begin(uint32_t baud=115200)	
	Description	Module initialisation, baud rate configuration
	Parameter	Baud: baud rate, range: 9600/14400/19200/38400/57600/115200/125000
	Return Value	void
	Note	The default baud rate is 115200
Performance Functions		
4	bool setAdvCtrl(bool ctrl)	
	Description	Turn on/off the broadcast
	Parameter	ctrl: broadcast status true: Turn on false: Turn off
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Need to wait for 600~800ms calibration time after turning on the broadcast
5	bool disconnect()	
	Description	Disconnect the Bluetooth device
	Parameter	—
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Wait for 5ms buffer time after disconnecting

6	bool writeData(uint8_t dataBuf[], uint8_t length)	
	Description	Send data to the device connected to the Bluetooth module
	Parameter	dataBuf[]: data length: the data length
	Return Value	Execution result: true: Succeeded false: Failed
	Note	The data[] byte length need to be less than or equal to 57bytes
7	bool readData(uint8_t receiveBuff[], uint8_t &length)	
	Description	Read data sent by the device
	Parameter	receiveBuff[]: read data &length: the read data length
	Return Value	Execution result: true: Succeeded false: Failed
	Note	The receiveBuff[] length is recommended to be greater than or equal to 57bytes
Function Configuration		
8	bool setAdvIntv(uint16_t min, uint16_t max, uint8_t advMap)	
	Description	Set the broadcast parameters
	Parameter	min: minimum interval parameterbetween broadcasts, range: 0x0020~0x4000 max: maximum interval parameter between broadcasts, range: 0x0020~0x4000 advMap: broadcast channel (multiple channels can be opened at the same time) bit0=1: channel37 (2402MHz) on/off control bit0=1: turn on bit0=0: turn off bit1=1: channel38 (2426MHz) on/off control bit1=1: turn on bit1=0: turn off bit2=1: channel39 (2480MHz) on/off control bit2=1: turn on bit2=0: turn off
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Minimum interval computational formula between broadcasts $t_{Adv-min}$=min×0.625ms Maximum interval computational formula between broadcasts $t_{Adv-max}$=max×0.625ms
9	bool setAdvData(uint8_t appendName, uint8_t length, uint8_t advData[])	
	Description	Set broadcast data
	Parameter	appendName: 0x00 (APPEND_NAME): data attached 0x10 (NO_APPEND_NAME): no data attached length: the advData[] byte length advData[]: data, the advData[] byte length needs to be less than or equal to 31bytes appendName=0x00, contain advData+ other attribute information appendName=0x10, only contain advData
	Return Value	Execution result: true: Succeeded false: Failed
	Note	No need to modify in normal use Refer to the BLE_API datasheet 4.9API_AdvData for details to modify advData and other information

10	bool setScanData(uint8_t length, uint8_t scanData[])	
	Description	When the broadcast data is scanned by the app, the module need to respond the data to the phone APP.
	Parameter	length: the scanData[] byte length scanData[]:the format is data byte length + data type + data, need less than and equal to 31bytes
	Return Value	Execution result: true: Succeeded false: Failed
	Note	No need to modify in normal use Refer to the BLE_API datasheet 4.10API_ScanData to modify data types
12	bool setConnIntv(uint16_t interval)	
	Description	After successful connection, set the periodic communication interval time between the Bluetooth module and the mobile phone APP
	Parameter	interval: communication interval time parameter, range: 0x0006~0x0c80
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Periodic communication interval time: t_{inv}=interval×1.25ms. Same as the function setConnIntv1 setup
13	bool setConnIntv(uint16_t minIntv, uint16_t maxIntv, uint16_t latency, uint16_t timeout)	
	Description	After the connection is successful, set the maximum and minimum periodic communication interval time between the Bluetooth module and the mobile phone APP, set the device delay sending time and connection timeout time
	Parameter	minIntv: minimum interval time parameter, range: 0x0006~0x0c80 maxIntv: maximum interval time parameter, range: 0x0006~0x0c80 latency: device delay sending time parameter, range: 0x0000~0x01f3 timeout: connection timeout time parameter, range: 0x000a~0x0c80
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Minimum interval time: $t_{inv-min}$=minIntv×1.25ms Maximum interval time: $t_{inv-max}$=maxIntv×1.25ms (Final connection interval time in the maximum and minimum range) Device delay sending time: t_{del}=latency×t_{inv} Connection timeout time: t_{out}=timeout×10ms
14	bool setName(uint8_t length, uint8_t name[])	
	Description	Set the display name that the Bluetooth module is detected when broadcasting.
	Parameter	length: the Bluetooth name byte number name[]: Bluetooth name
	Return Value	Execution result: true: Succeeded false: Failed
	Note	The name[] byte length need to be less than or equal to 31bytes
15	bool setAddress(uint8_t address[])	
	Description	Set the Bluetooth mac address
	Parameter	address[]: Bluetooth mac address (6bytes)
	Return Value	Execution result: true: Succeeded false: Failed
	Note	—

16	bool setFeature(uint8_t cmdFlag, uint32_t codeFeature)	
	Description	Set the module data update method
	Parameter	cmdFlag: data write method 0x00 (FEATURE_DIR): cover the previous value 0x10 (FEATURE_OR): write in the “or” mode with the previous value 0x20 (FEATURE_AND): write in the “and” mode with the previous value codeFeature: 4byte, refer to BLE_API datasheet 4.14API_Feature for details
	Return Value	Execution result: true: Succeeded false: Failed
	Note	—
17	bool setPowerSaving(uint8_t mode)	
	Description	Set the module energy saving mode
	Parameter	mode: energy saving mode 0x00: sleep (default) 0x01: deep sleep 0x15: power down, keep in deep sleep, wake up will force it to reset, all settings in RAM are reset to default values
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Wake up in sleep and deep sleep mode, must send dummy bytes (at least 100μs); use wakeUp() function to get out of sleep.
18	bool setWhiteList(bool erase, uint8_t whiteListAddr[], uint8_t mask[])	
	Description	Set the whitelist
	Parameter	erase: action to the whitelist true: Add false: Delete whiteListAddr[]:the device address to be added to the whitelist (6 bytes) mask[]: address mask, the module internal default value is 0xFFFFFFFF
	Return Value	Execution result: true: Succeeded false: Failed
	Note	Refer to BLE_API datasheet 4.20API_WhiteList for details
19	bool setTXpower(uint8_t power)	
	Description	Set transmit power
	Parameter	power: transmit power gears, range: 0x00 (min)~0x0F (max)
	Return Value	Execution result: true: Succeeded false: Failed
	Note	For example: power=0x00, TX transmit power=-32.5dbM power=0x05, TX transmit power=-11.5dbM power=0x0A, TX transmit power=+0.5dbM power=0x0F, TX transmit power=+3.5dbM
20	bool setCrystalOffset(uint8_t cload)	
	Description	Set the RF output frequency offset of the external 16MHz crystal oscillator
	Parameter	cload: RF output frequency offset, range: 0x00~0x0f
	Return Value	Execution result: true: Succeeded false: Failed
	Note	For example: cload=0x00, crystal frequency=15.99985MHz cload=0x05, crystal frequency=16.00004MHz cload=0x0A, crystal frequency=16.00031MHz cload=0x0F, crystal frequency=16.00074MHz

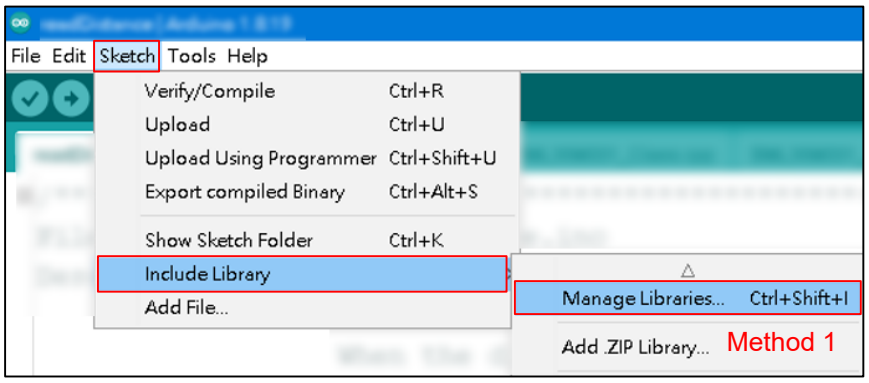
21	bool restoreDefault()	
	Description	Software reset, restore the default value after reset
	Parameter	—
	Return Value	Execution result: true: Succeeded false: Failed
	Note	—
22	bool reset()	
	Description	Software reset, maintain the previous set value after reset
	Parameter	—
	Return Value	Execution result: true: Succeeded false: Failed
	Note	—
23	void wakeUp();	
	Description	Wake up the Bluetooth module from sleep mode
	Parameter	—
	Return Value	void
	Note	Need to wait for 30ms to wake up Bluetooth module completely
24	bool sendCMD(uint16_t type, uint8_t cmd_flag, uint8_t cmd_length, uint8_t value[])	
	Description	General command send function, send commands with data to the BM7701-00-1 module
	Parameter	type: type (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag: the API command flag cmd_length: type+value+cmd_flag byte length value[]: the command data Buffer
	Return Value	Execution result: true: Succeeded false: Failed
	Note	24~27 are general functions of BLE API CmdType2, refer to the BLE_API datasheet for parameters and corresponding data
25	void sendCMD_NoResponse(uint16_t type, uint8_t cmd_flag, uint8_t cmd_length, uint8_t value[])	
	Description	General command send function without response
	Parameter	type: type (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag: the API command flag cdm_length: type+value+cmd_flag byte length value[]: the command data Buffer
	Return Value	void
	Note	—
26	bool readCMD(uint16_t type, uint8_t cmd_flag)	
	Description	General command read function
	Parameter	type: type (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag: the API command flag
	Return Value	Execution result: true: Succeeded false: Failed
	Note	—
27	void readCMD_NoResponse(uint16_t type, uint8_t cmd_flag)	
	Description	General command read function without response
	Parameter	type: type (UUID/Profile/Service/Characteristics/Descriptor) cmd_flag: the API command flag
	Return Value	void
	Note	—

Arduino Lib Download and Installation

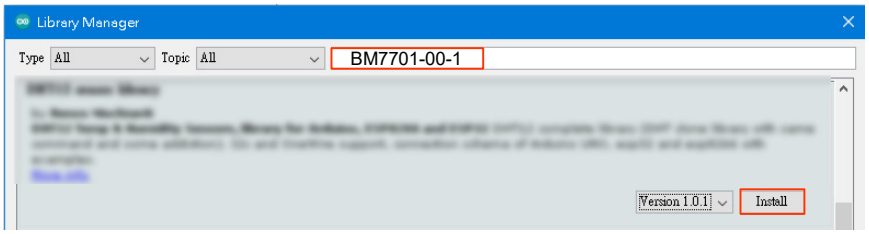
BM7701-00-1 Library: Refer to the following two methods to install the BM7701-00-1 Arduino Library.

Method 1: Search for installation

Arduino IDE → Sketch → Include Library → Manage Libraries... → Search BM7701-00-1 → Install



Search for Installation Step 1

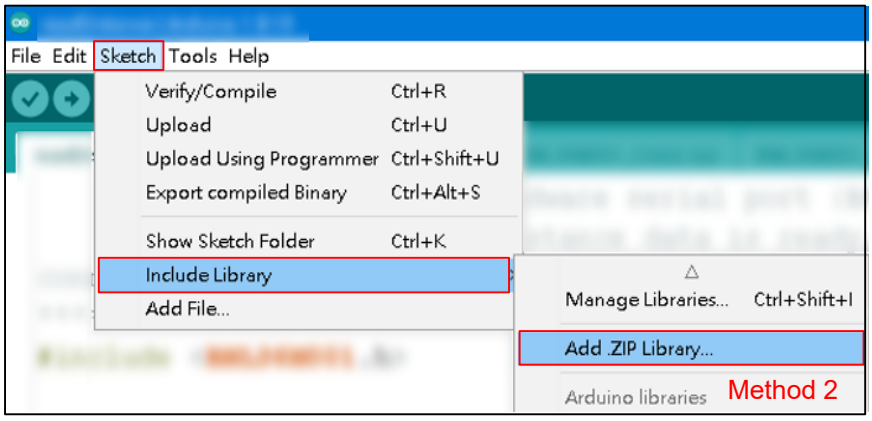


Search for Installation Step 2

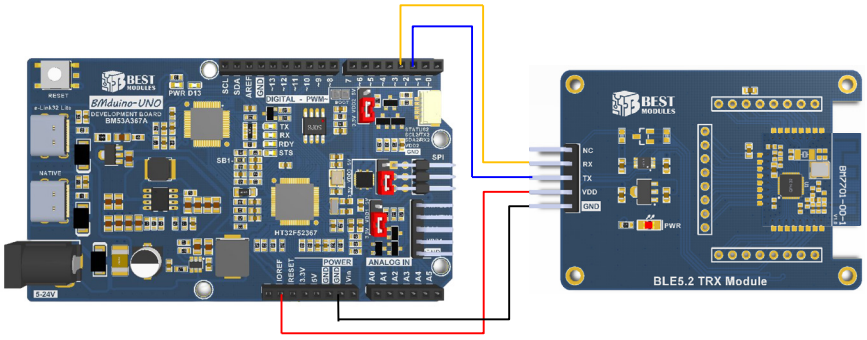
Method 2: Download the .ZIP library before adding it

Download the Arduino example (BM7701-00-1 Library) under the DOCUMENTS menu from the Best Modules website (https://www.bestmodulescorp.com/BM7701-00-1.html).

Add .ZIP library: Arduino IDE → Sketch → Include Library → Add .ZIP Library...



Arduino Examples



Physical Connection Diagram

Example: writeAndRead

Example function: The module connects to the mobile phone APP "BLEDemo"or "BCBLEDemo". After pressing the APP key, the module receives the corresponding data and prints it to the serial monitor, then sends the corresponding data to the APP to turn on the APP LED. Press this key once again on the APP, the light will turn off.

1. Open the example: File → Examples → Select Lib (BM7701-00-1) → Select example (writeAndRead)

2. Example Description:

   a. Call library function, module initialisation, parameter settings includes key, LED and some Bluetooth parameter settings

```
#include <BM7701-00-1.h>
BM7701_00_1       BC7701(2, 3);// Use software UART
#define TX_POWER     0x0F       // TX Power
#define XTAL_CLOAD   0x04       // 16MHz crystal load
#define ADV_MIN      100        // Minimum broadcast interval
#define ADV_MAX      100        // Maximum broadcast interval
#define CON_MIN      30         // Minimum connection interval
#define CON_MAX      30         // Maximum connection interval
#define CON_LATENCY  00         // Connection latency
#define CON_TIMEOUT  300        // Connection timeout
uint8_t BDAddress[6]={0x11, 0x22, 0x33, 0x44, 0x55, 0x66};
                                // Device address
uint8_t BDName[]={'B', 'M', 'C', '7', '7', 'M', '0', '0', '1'};
                                // Device name
uint8_t Adata[]={0x02, 0x01, 0x06};      // Broadcast data
uint8_t Sdata[]={0x03, 0x02, 0x0f, 0x18};// The response data
                                         // after scanning
#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. After power on reset, delay 60ms for power on to complete before executing the begin() function

```
void setup() {
  delay(60);                 // Wait for power on complete
  Serial.begin(9600);        // Configure serial monitor
  BC7701.begin(BAUD_115200); // Module initialisation
  setup()
  while (sel != 10)
  {
    switch (sel)
    {
      case 1: if (BC7701.setAddress(BDAddress) == true) sel++;
                             // Set the Bluetooth address
        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;// Set broadcast interval
      case 4: if (BC7701.setAdvData(APPEND_NAME, sizeof(Adata),
      Adata) == true) sel++;
        else sel=0xFF; break;// Set broadcast data
      case 5: if (BC7701.setScanData(sizeof(Sdata), Sdata) == true)
      sel++;
        else sel=0xFF; break;// Set the response data after
                             // the broadcast is scanned
      case 6: if (BC7701.setTXpower(TX_POWER) == true) sel++;
        else sel=0xFF; break;// Set the TX transmit power
      case 7: if (BC7701.setCrystalOffset(XTAL_CLOAD) == true) sel++;
        else sel=0xFF; break;// Set the external crystal offset
      case 8: if (BC7701.setFeature(FEATURE_DIR, AUTO_SEND_SATUS)
      == true) sel++;
        else sel=0xFF; break;// Set the data update method
      case 9: if (BC7701.setAdvCtrl(ENABLE) == true) sel++;
        else sel=0xFF; break;// Enable broadcast function
      case 0xFF: digitalWrite(13,HIGH);break;// Failed configuration,
                                             // turn on the LED13
    }
  }
  delay(650); // Wait for the broadcast to turn on completely
}
```

c. Determine the Bluetooth module status, if the connection is successful, it can receive data to determine the data and connection status.

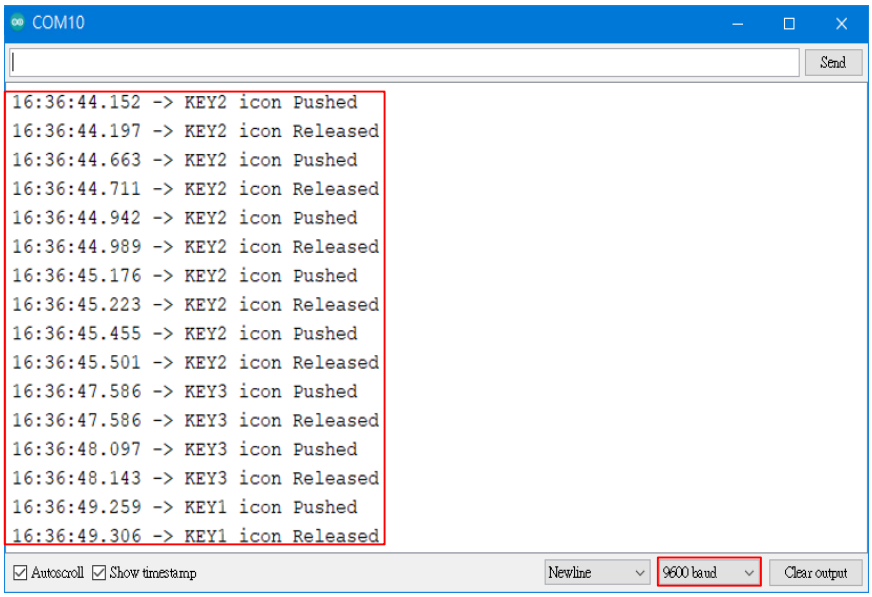
```
void loop()
{
  Status=bleProcess();          // Read the BLE current state
  if (Status)
  {
    switch (Status)
    {
      case API_CONNECTED:       // Device first connection
        if (board_connect == false)
        {
          board_connect=true;
          board_receive=false;
        }
        break;
      case API_DISCONNECTED:    // Device failed connection
        board_connect=false;
        board_receive=false;
        board_conIntv=false;
        break;
      case DATA_RECEIVED:       // Device has received the data
        if (board_connect == true)
        {
          digitalWrite(13,LOW);
          board_receive=true;
        }
        break;
      case API_ERROR:           // Error
        digitalWrite(13,HIGH);
        break;
    }
  }
```

d. Set the connection interval after successful connection, receive data to perform corresponding actions

```
if (board_connect == true)      // Connection status is correct
{
  if (board_conIntv == false)   // Determine whether the overconnection
                                // interval has been configured
  {
    BC7701.wakeUp();
    if (BC7701.setConnIntv(CON_MIN/1.25, CON_MAX/1.25, CON_LATENCY,
    CON_TIMEOUT) == true)       // Configure connection interval
    board_conIntv = true;
  }
  if (board_receive == true)    // Received data
  {
    board_receive = false;      // Clear the flag
    if (receiveBuf[3] == 0xB0) // Data packet frame header
    {
      switch (receiveBuf[4])    // Key data bit
      {
        case 0x11:
          count=1;
```

```
            Serial.println("KEY1 icon Pushed");      // Press Key 1
            break;
          case 0x10:
            count=2;
            Serial.println("KEY1 icon Released");    // Release Key 1
            break;
          case 0x22:
            count=1;
            Serial.println("KEY2 icon Pushed");      // Press Key 2
            break;
          case 0x20:
            count=2;
            Serial.println("KEY2 icon Released");    // Release Key 2
            break;
          case 0x44:
            count=1;
            Serial.println("KEY3 icon Pushed");      // Press Key 3
            break;
          case 0x40:
            count=2;
            Serial.println("KEY3 icon Released");    // Release Key 2
            break;
        }
        if (receiveBuf[4]!=0 && count==2&& flag==0) // Received 2 pieces
                                                    // of data, send light data
        {
          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) // Received 2 pieces of
                         // data again, turn off the light when it is on
        {
          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. Open the serial monitor and set the baud rate to be 9600. The serial monitor will receive the key status of APP as follows.

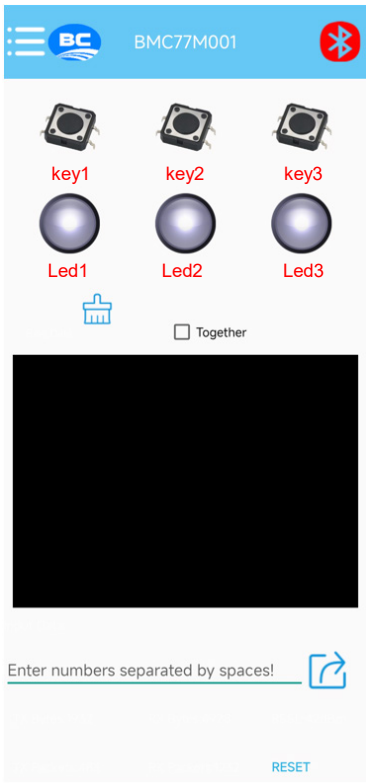


Appendix

1. Mobile APP installation

Android

IOS



UI display after installation

Note: For some mobile phone models, if the text in Bluetooth list and main interface is lost after the APP is opened, try to switch the display mode of the phone (day mode/night mode/ light mode/dark mode) until you can see the text.

2. Command Process