电容式指纹识别模块

BM92S2231-1
Arduino Library V1.0.1 说明

版本：V1.00　日期：2024-05-28

www.bestmodulescorp.com

目录

简介

BM92S2231-1 是倍创推出的电容式指纹识别模块，使用 UART 通信方式。本文档对 BM92S2231-1 的 Arduino Lib 函数、Arduino Lib 安装方式进行说明；范例使用 BMA92K223 模块，演示了指纹注册、指纹识别功能。

适用型号：

型号	说明
BM92S2231-1	电容式指纹识别模块
BMA92K223	包括转接板、连接线、BM92S2231-1

Arduino Library 函数

Arduino Lib 名称：BM92S2231-1		Lib 版本：V1.0.1
构造函数 & 初始化		
1	BM92S2231_1(uint8_t keyoutPin=2,HardwareSerial *theSerial = &Serial)	
	描述	构造函数，选择硬件 UART 通信
	参数	keyoutPin：触控引脚 (默认为 D2)，连接 BM92S2231-1 的 Keyout 引脚或 BMA92K223 的 I/O (Keyout) 引脚 *theSerial：选择硬件 UART 接口 (默认为 Serial1 接口)
	返回值	—
	备注	—
2	BM92S2231_1(uint8_t keyoutPin,uint8_t rxPin, uint8_t txPin)	
	描述	构造函数，选择软件 UART 通信
	参数	keyout：触控引脚，连接 BM92S2231-1 的 Keyout 引脚或 BMA92K223 的 I/O (Keyout) 引脚 rxPin：RX 引脚，连接 BM92S2231-1 或 BMA92K223 的 TX 引脚 txPin：TX 引脚，连接 BM92S2231-1 或 BMA92K223 的 RX 引脚
	返回值	—
	备注	—
3	void begin(unsigned long baud = 115200)	
	描述	模块初始化
	参数	baud：通信波特率选择 (默认 115200 bps)
	返回值	void
	备注	—
功能函数 – 注册		
4	int8_t InputEnrollID(int16_t ID)	
	描述	输入注册的指纹 ID
	参数	ID：指纹 ID，范围：1~1000
	返回值	执行情况 0：ID 在范围内且未被使用 4：ID 已使用 8：ID 超出范围
	备注	—

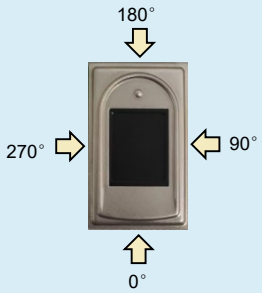
5	uint8_t enroll()	
	描述	注册指纹
	参数	—
	返回值	执行情况： 0x01：再次按压手指 0x02：注册成功 0x03：注册失败 0x05：用手指其他区域重试 0x06：请移动手指
	备注	需在“if(InputEnrollID ()==0)” 后使用
6	bool stopEnroll()	
	描述	停止注册
	参数	—
	返回值	执行情况 true：成功 false：失败
	备注	—
功能函数 – 识别指纹		
7	uint8_t getKeyout()	
	描述	获取 I/O (Keyout) 引脚当前状态
	参数	—
	返回值	I/O (Keyout) 引脚电平 0：Low，无触摸按下 1：High，有触摸按下
	备注	—
8	int16_t identify()	
	描述	根据放置的手指指纹 (已注册)，识别出其 ID
	参数	—
	返回值	执行情况或 ID： 0：无指纹按压信息 0xffff：检测指纹失败 其他：检测成功，对应放置的手指指纹的 ID
	备注	—
9	bool stopIdentify()	
	描述	停止检测
	参数	—
	返回值	执行情况： true：成功 false：失败
	备注	—

功能函数 – 删除指纹		
10	uint8_t deleteID(uint16_t id)	
	描述	删除指定 ID 的指纹
	参数	id：指纹 ID，范围：1~1000
	返回值	执行情况： 0x00：数据接收有误 0x01：删除成功 0x02：删除失败 0x03：删除的 ID 不存在
	备注	—
11	bool deleteAll()	
	描述	删除全部 ID 的指纹
	参数	—
	返回值	执行情况 true：成功 false：失败
	备注	—
功能函数 – 通用		
12	bool StandbyMode()	
	描述	进入待机模式
	参数	—
	返回值	进入待机模式情况： true：成功 false：失败
	备注	可以通过手指触摸指纹检测区域来唤醒
13	bool changeBaud(unsigned long baud)	
	描述	修改模块的波特率
	参数	baud：通信波特率选择 9600：9600bps 19200：19200bps 38400：38400bps 57600：57600bps 115200：115200bps (默认) 128000：128000bps 230400：230400bps 256000：256000bps 460800：460800bps
	返回值	执行情况： true：成功 false：失败
	备注	模块断电后重新上电或进入待机模式再次唤醒时波特率将变为默认波特率

14	uint8_t getDeviceInformation()	
	描述	返回获取的模块设备信息
	参数	—
	返回值	执行情况： 0：获取失败 1：BM92S2131_1 2：BM92S2231_1 3：BM92S2331_1
	备注	
15	bool getModuleSettingsInformation(uint8_t inforArray[])	
	描述	获取模块目前配置的分数阈值、检测角度和模板数量。
	参数	inforArray[]：获取到的参数信息 inforArray[0]：分数阈值低位 inforArray[1]：分数阈值高位 inforArray[2]：检测角度 inforArray[3]：模板数量
	返回值	执行情况： true：成功 false：失败
	备注	—
16	bool getImageSettingInformation(uint8_t inforArray[])	
	描述	获取模块目前配置的图像阈值和图像百分比
	参数	inforArray[]：获取到的参数信息 inforArray[0]：图像阈值低位 inforArray[1]：图像阈值高位 inforArray[2]：图像百分比低位 inforArray[3]：图像百分比高位
	返回值	执行情况： true：成功 false：失败
	备注	—

17	bool userSet(uint16_t score,uint8_t checkAngle,uint8_t numberTemplates)	
	描述	用户设置
	参数	score：分数阈值，指纹检测分数大于分数阈值，则检测成功 范围是 0~100，默认 90 checkAngle：检测角度（注） 0x00：全角度检测方式 1，手指检测的按压方向可以是 0°、90°、180°、270° 0x01：单角度检测，手指检测的按压方向需和注册的方向一致 (0°) 0x03：3 个角度检测，手指检测的按压方向可以是 0°、90°、270° 0x04(default)：全角度检测方式 2，手指检测的按压方向可以是 0°、90°、180°、270° numberTemplates：模板数量 范围是 1~3，默认 3
	返回值	执行情况： true：成功 false：失败
	备注	① 模板数量和分数阈值的设置共同决定 FAR (误识率)，FRR (拒识率) 模板数量 1：分数阈值 79 (FAR≤0.001%，FRR≤9%) 分数阈值 73 (FAR≤0.01%，FRR≤7%) 模板数量 2：分数阈值 89 (FAR≤0.001%，FRR≤5%) 分数阈值 78 (FAR≤0.01%，FRR≤3%) 模板数量 3：分数阈值 87 (FAR≤0.001%，FRR≤3%) 分数阈值 80 (FAR≤0.01%，FRR≤2%) ② 全角度检测方式 1、2 的区别：1 的耗时比 2 少，但是识别效果比 2 略差。
18	bool imageSet(uint16_t imageThreshold,uint16_t imagePercentage)	
	描述	图像设置
	参数	imageThreshold：图像阈值，范围是 50~1000，默认 128 imagePercentage：图像百分比，范围是 0~100，默认 60
	返回值	执行情况： true：成功 false：失败
	备注	—

注：传感器的方向

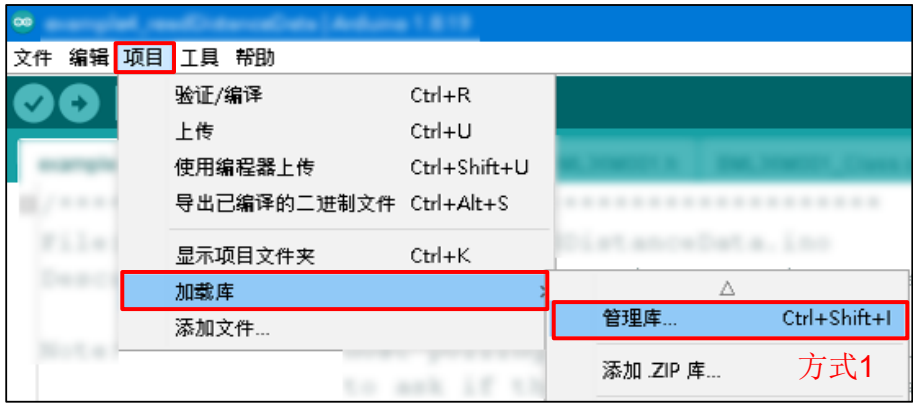


Arduino Lib 下载及安装

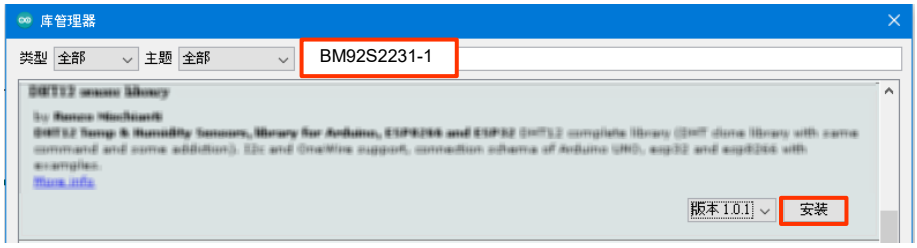
BM92S2231-1 A Library：可参考下面两种方法安装 BM92S2231-1 的 Arduino Library

方式 1：搜索安装

搜索安装：Arduino IDE → 项目 → 加载库 → 管理库 → 搜索 BM92S2231-1 → 安装



搜索安装流程 **1**

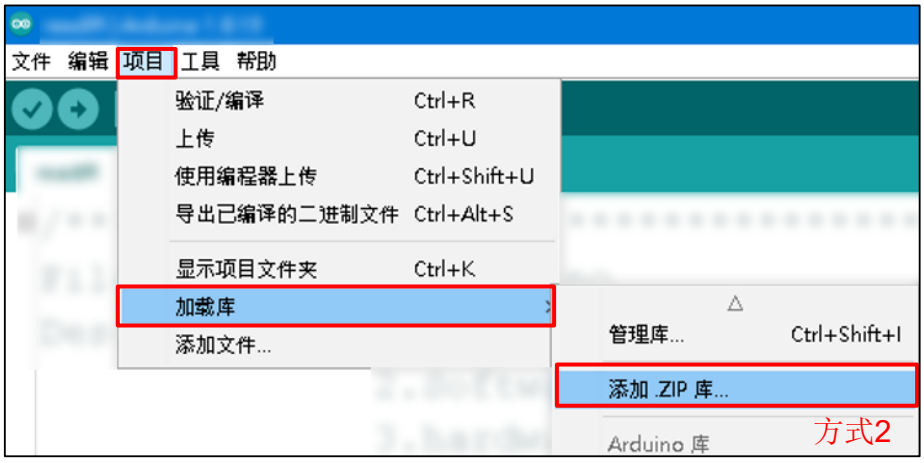


搜索安装流程 **2**

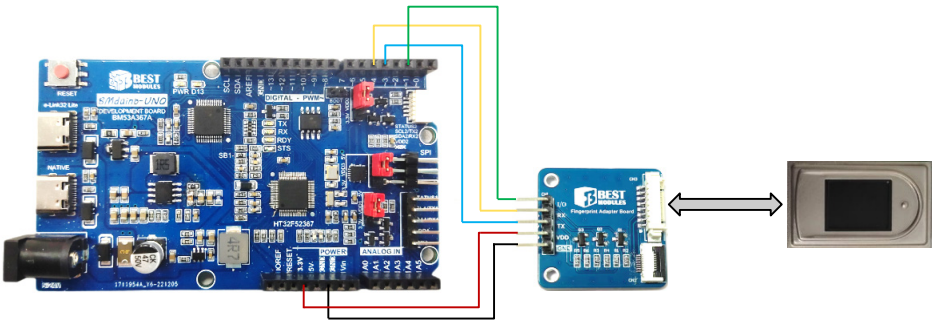
方式 2：添加 .ZIP 库，需提前下载 .ZIP 库

下载方法：打开倍创官方网站 (https://www.bestmodulescorp.com/bm92s2231-1.html) 文件目录下的 Arduino 范例程序 (BM92S2231-1.zip)。

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

注意：模块 touch 灵敏度高，不检测指纹时，手指请勿触碰到模块

范例 1：enroll

范例 1 实现功能：通过一个手指进行注册指纹，并在 Serial 上显示。

1. 范例打开方式：Arduino IDE → 文件 → 示例 → Lib 选择 (BM92S2231-1) → 选择范例 (enroll)
2. 示例说明：
   a. 构建对象 & 模块初始化及配置

```
#include "BM92S2231-1.h"
BM92S2231_1 fps(2,4,5); // 使用软件 UART 通信。keyout 引脚连接 D2，TX 引脚连
                        // 接 D4，RX 引脚连接 D5
uint16_t ID=6; // 创建变量，用于存储注册 ID
uint8_t idCheck = 1,enrollEnable=0; // 创建变量，idCheck 用于记录是否开启
                                    // ID 检测（开启检测 idCheck=1；
                                    // 关闭检测：idCheck=0 ），
                                    // enrollEnable 用于记录是否开启注册
                                    // （开启注册 enrollEnable=1；关闭注
                                    // 册：enrollEnable=0 ）
uint16_t enrollState; // 创建变量，用于记录注册的执行情况
uint8_t idState=0xff;  // 创建变量，用于记录 ID 使用情况
uint16_t timeCnt=0;    // 创建变量，用于记录未检测到手指的时间
void setup() {
  Serial.begin(115200); // 串口监视器初始化
  delay(100);  // 等待模块稳定
  fps.begin(); // 模块初始化
}
```

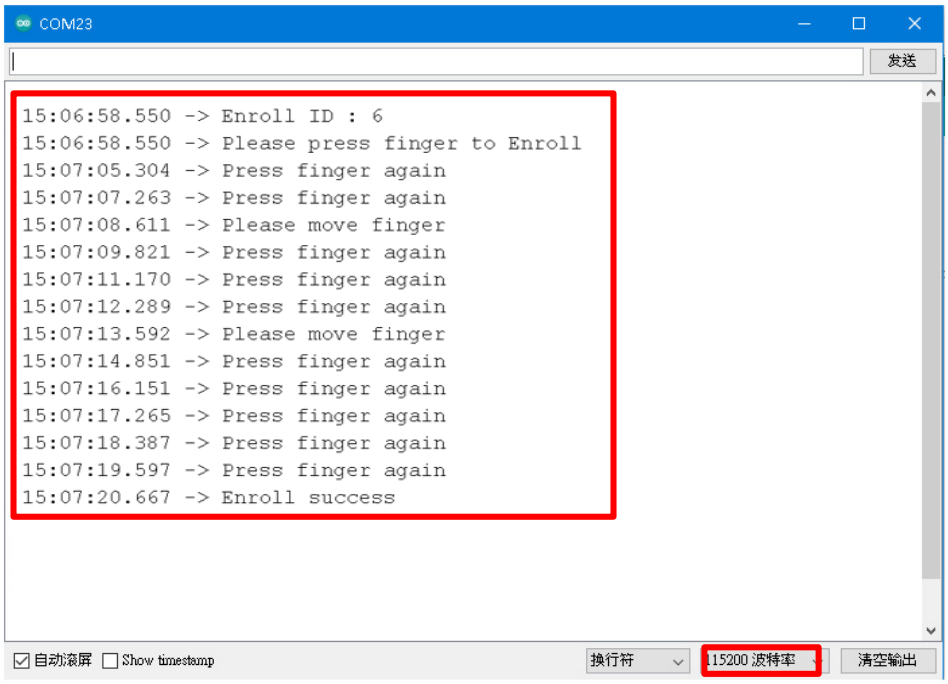
b. 注册指纹并显示在串口监视器上

```
void loop() {

     if(idCheck==1)
    {
      idState=fps.InputEnrollID(ID); // 检测输入的 ID，记录当前 ID 使用情况
      if(idState==0) // ID 在范围内且未被使用
      {
        idCheck=0; // 退出 ID 检测
        enrollEnable=1; // 开启注册
        timeCnt=0; // 手指检测时间清 0
        Serial.print(F("Enroll ID : ")); // 打印要录入指纹的 ID
        Serial.println(ID);
        Serial.println(F("Please press finger to Enroll"));
        // 打印开始注册的提示
      }
      else if(idState==4) // ID 已被使用，加 1 再进行 ID 检测
      {
       ID ++;
       //Serial.println(F("ID already used"));
      }
      else if(idState==8) // ID 超出范围
      {
        idCheck=0;
       Serial.println(F("ID out of range"));
      }
      else // 通信有误
      {
        idCheck=0;
       Serial.println(F("Communication error"));
      }
    }
    if(enrollEnable==1) // 开始注册
    {

        enrollState=fps.enroll(); // 开始指纹注册
        if(enrollState!=0) // enrollState 不为 0，时间计数清 0
          timeCnt=0;

        if(enrollState==1) // enrollState=1，提示继续按压手指
         Serial.println(F("Press finger again"));

        else if(enrollState==2) // enrollState=2，注册成功
        {
         enrollEnable=0; // 关闭注册
         Serial.println(F("Enroll success")); // 提示注册成功

         if(fps.stopEnroll()==false) // 停止注册
         Serial.println(F("Communication error"));
         else
         idCheck=1; // 开启 ID 检测
         Serial.println(F(""));
        }
```

```
        else if(enrollState==3) // enrollState=3，注册失败
        {
          enrollEnable=0; // 关闭注册
          Serial.println(F("Enroll fail")); // 提示注册失败

          if(fps.stopEnroll()==false) // 停止注册
          Serial.println(F("Communication error"));
          else
          idCheck=1; // 开启 ID 检测

          Serial.println(F(""));
        }

        else if(enrollState==5) // enrollState=5，提示用手指表面其他区域
                                // 重试
        Serial.println(F("Try again with other areas on the surface
of your fingers"));

        else if(enrollState==6) // enrollState=6，提示移动手指
         Serial.println(F("Please move finger"));

        else if(enrollState==0) // enrollState=0，手指没有按压
        {
          timeCnt++; // 时间计数加 1
          if(timeCnt>60) // 时间计数超过 60
          {
            Serial.println(F("No fingerprint detected")); // 提示没有手
                                                          // 指按下
            enrollEnable=0; // 关闭注册

            if(fps.stopEnroll()==false) // 停止注册
            Serial.println(F("Communication error"));
          }
        }

    }

}         else if(enrollState==5) // 需要更多的指纹信息
        Serial.println(F("Try again with other areas on the surface
of your fingers"));

        else if(enrollState==6||enrollState==7) // 需要移动手指
         Serial.println(F("Please move finger"));

        else if(enrollState==0)
        {
          timeCnt++;
          if(timeCnt>60) // 超时检测，30 秒无操作
          {
            Serial.println(F("No fingerprint detected"));
            enrollEnable=0;
```

```
            if(fps.stopEnroll()==false)
            Serial.println(F("Communication error"));
          }
        }

    }

}
```

3. 打开串口监视器，波特率选择 115200。串口监视器显示如下：



范例 2：identify

范例 2 实现功能：识别指纹是否为注册过的指纹，并在串口监视器上显示注册的指纹 ID。

1. 范例打开方式：Arduino IDE → 文件 → 示例 → Lib 选择 (BM92S2231-1) → 选择范例 (identify)
2. 示例说明：
   a. 构建对象 & 模块初始化及配置

```
#include "BM92S2231-1.h" // 包含头文件

BM92S2231_1 fps(2,4,5); // 使用软件 UART 通信。Keyout 引脚连接 D2，TX 引脚连
                        // 接 D4，RX 引脚连接 D5
uint16_t result;
void setup() {
  Serial.begin(115200); // 初始化串口
  delay(100);  // 上电等待初始化完成
  fps.begin(); // 初始化模块
  Serial.println(F("Please press your finger to start detection"));
}
```

b. 识别指纹是否为注册过的指纹并显示在串口监视器上

```
void loop() {
  if(fps.getKeyout()!=BM92S2231_1_NO_KEY) // 如果检测到触控按下
  {
    delay(100); // 深度休眠唤醒时间
    result=fps.identify(); // 识别指纹
    if(result>0) // 识别完成
    {
      if(result==0xffff) // 识别失败
      {
      Serial.println(F("Detection failed"));
      }
      else // 识别成功
      {
      Serial.print(F("Verified ID: "));
      Serial.println(result);
      }
    }
  }
```

3. 打开串口监视器，波特率选择 115200。串口监视器显示如下：