矩阵式红外测温模块

BMS26M833
Arduino Library 说明

版本：V1.00　日期：2023-06-08

www.bestmodulescorp.com

目录

简介 3
Arduino Lib 函数 **3**
Arduino Lib 下载及安装 **7**
Arduino 范例 **8**
范例 1：readTemperature 8
范例 2：setInterrupt 10
范例 3：DisplayThermalImagingOnTheTFT 12

简介

BMS26M833 是倍创推出的 8×8 矩阵式红外测温模块，使用 I²C 通信方式。本文档对 BMS26M833 的 Arduino Lib 函数、Arduino Lib 安装方式进行说明；范例演示了读取温度等功能。

Arduino Lib 函数

Arduino Lib 名称：BMS26M833		Lib 版本：V1.0.1
构造函数 & 初始化		
1	BMS26M833(uint8_t intPin = 8, TwoWire * theWire = &Wire)	
	描述	构造函数，选择中断引脚以及 wire 的接口
	参数	intPin：INT 引脚默认为 D8，连接模块 INT 引脚 * theWire：I²C 接口默认为 wire
	返回值	—
	备注	—
2	void begin(uint8_t i2c_addr = BMS26M833_IICADDR)	
	描述	模块初始化
	参数	i2c_addr：I²C 地址，默认为 0x68
	返回值	void
	备注	—
功能函数		
3	void writeReg(uint8_t addr, uint8_t data)	
	描述	写入数值至特定的寄存器
	参数	addr：寄存器地址 data：要写入的数据
	返回值	void
	备注	寄存器地址请参考 AMG8833 规格书
4	uint8_t readReg(uint8_t addr)	
	描述	读取特定寄存器的数值
	参数	addr：寄存器地址
	返回值	寄存器里的数据
	备注	寄存器地址请参考 AMG8833 规格书
5	void readReg(uint8_t addr, uint8_t rBuf[], uint8_t rLen)	
	描述	读取寄存器来获取数据
	参数	addr：寄存器地址 rBuf[]：用于存储要获取的数据的变量 rLen：数据的字节
	返回值	void
	备注	—

6	void readPixels(float tempBuff[])	
	描述	获取 8×8 像素温度数据
	参数	tempBuff[]：存储来自传感器的温度数据，单位：℃
	返回值	void
	备注	tempBuff[] 建议长度设为 64-byte
7	void readPixelsAndMaximum(float tempBuff[], float &maxVlaue, float &minVlaue)	
	描述	获取 8×8 像素温度数据和其中的最值
	参数	tempBuff[]：存储来自传感器的温度数据，单位：℃ &maxVlaue：存储来自传感器的温度数据的最大值，单位：℃ &minVlaue：存储来自传感器的温度数据的最小值，单位：℃
	返回值	void
	备注	tempBuff[] 长度建议设为 64-byte
8	void getINTTable(uint8_t buf[], uint8_t size = 8)	
	描述	获取 8×8 中断表 (注)
	参数	buf[]：存储 8×8 中断表数据 size：要获取的中断寄存器的个数 (最多为 8，依次为 INT0~INT7)
	返回值	void
	备注	—
9	uint8_t getOperationMode()	
	描述	获取工作模式
	参数	—
	返回值	工作模式
	备注	—
10	void sleep()	
	描述	进入休眠模式
	参数	—
	返回值	void
	备注	—
11	void reset(uint8_t mode=INITIAL_RESET)	
	描述	设置复位模式
	参数	mode：复位模式选择 0x30(FLAG_RESET)：标志位复位 0x3f(INITIAL_RESET)：中断复位 (默认)
	返回值	void
	备注	—
12	uint8_t getFrameMode()	
	描述	获取帧模式
	参数	—
	返回值	帧模式 0x01(FPS_1)：1Hz 0x00(FPS_10)：10Hz
	备注	—

13	uint8_t getINT()	
	描述	获取 INT 引脚电平
	参数	—
	返回值	INT 引脚电平 1：HIGH，温度在所设范围内 0：LOW，温度超过所设定的温度上限值或低于温度的下限值
	备注	—
14	uint8_t getStatus()	
	描述	获取状态寄存器
	参数	—
	返回值	状态寄存器数据
	备注	—
15	uint8_t getAverageOutputMode()	
	描述	获取平均输出模式
	参数	—
	返回值	输出模式 0x20(TWICE_MOVE_AVE_OUTPUT)：两次输出模式 0x00(ONE_MOVE_OUTPUT)：一次输出模式
	备注	—
16	float readThermistorTemp()	
	描述	获取热敏电阻温度值
	参数	—
	返回值	温度值，单位：℃
	备注	此温度值一般用于测量环境温度
参数配置		
17	void beginMeasure(uint8_t mode= NORMAL_MODE)	
	描述	开始测量
	参数	mode：测量模式选择 0x00(NORMAL_MODE)：正常模式 (默认) 0x20(STAND_BY_MODE_60SEC)：60s 测量一次 0x21(STAND_BY_MODE_10SEC)：10s 测量一次
	返回值	void
	备注	—
18	void setFrameMode(uint8_t mode =FPS_10)	
	描述	设置帧模式
	参数	mode：帧模式选择 0x01(FPS_1)：1Hz 0x00(FPS_10)：10Hz (默认)
	返回值	void
	备注	—

19	void setINT(uint8_t isEnable = false)	
	描述	设置中断状态
	参数	isEnable：中断模式选择 true：使能 false：除能
	返回值	void
	备注	—
20	void setInterruptLevels(float high, float low)	
	描述	设置中断的温度阈值
	参数	high：温度上限阈值，单位：℃ low：温度下限阈值，单位：℃
	返回值	void
	备注	默认滞后值为 high 的 95%
21	void setInterruptLevels(float high, float low, float hysteresis)	
	描述	设置中断的温度阈值
	参数	high：温度上限阈值，单位：℃ low：温度下限阈值，单位：℃ hysteresis：中断检测的滞后值，单位：℃
	返回值	void
	备注	分辨率为 0.25℃
22	void setStatusClear()	
	描述	清除状态寄存器
	参数	—
	返回值	void
	备注	同时清除以下标志位： OVT：热敏电阻温度溢出标志位 (当输出值 (REG：0x0E，0x0F) 超过 0xFFF) OVS：像素温度溢出标志位 (当输出值 (REG：0x80~0xFF) 超过 0xFFF) INT：中断标志位
23	void setAverageOutputMode(uint8_t mode= ONE_MOVE_OUTPUT)	
	描述	设置平均输出模式
	参数	mode：输出模式选择 0x20(TWICE_MOVE_AVE_OUTPUT)：两次输出模式 0x00(ONE_MOVE_OUTPUT)：一次输出模式
	返回值	void
	备注	—
24	void setOperationMode(uint8_t mode)	
	描述	设置工作模式
	参数	mode：输出模式选择 0x00(NORMAL_MODE)：正常工作模式 0x10(SLEEP_MODE)：睡眠模式 0x20(STAND_BY_MODE_60SEC)：待机模式 _60 秒 0x21(STAND_BY_MODE_10SEC)：待机模式 _10 秒
	返回值	void
	备注	—

注：8×8 中断表如下表所示 (与传感器的像素点一致)，当某个像素温度超过温度上限值或低于下限值时，对应的中断置 1，否则为 0。

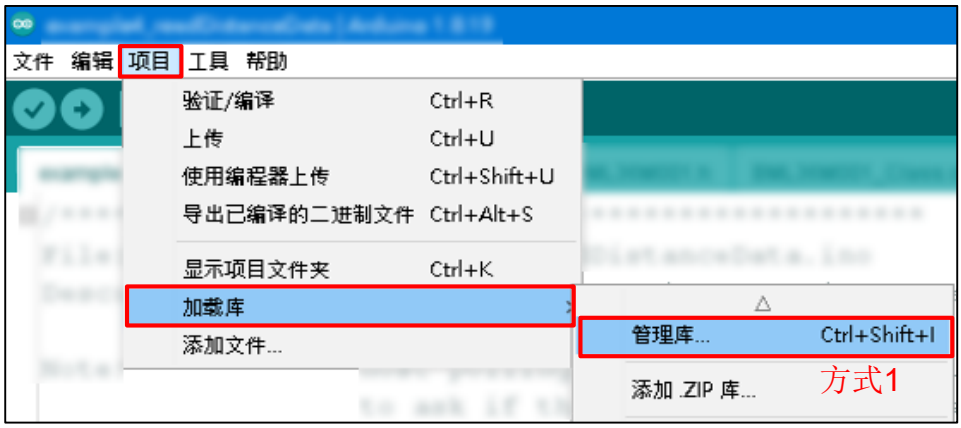
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
byte 0	0	1	0	0	0	0	0	1
byte 1	0	0	0	0	0	0	0	0
byte 2	0	0	0	0	0	0	0	0
byte 3	0	0	0	1	0	0	0	0
byte 4	0	0	0	0	0	0	0	0
byte 5	0	0	0	0	0	0	0	0
byte 6	0	0	0	0	0	1	0	0
byte 7	0	0	0	0	0	0	0	0

Arduino Lib 下载及安装

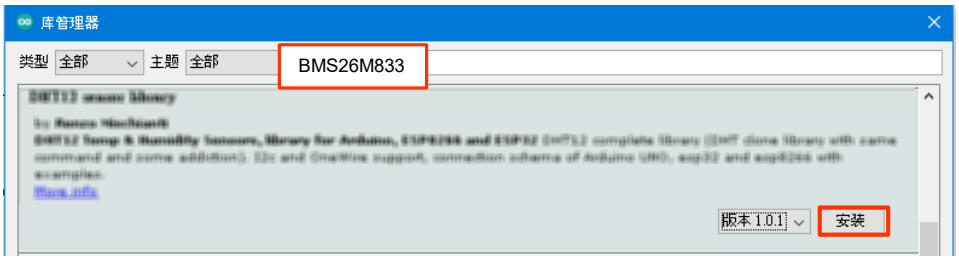
BMS26M833 Arduino Library：可参考下面两种方法安装 BMS26M833 的 Arduino Library

方式 1：搜索安装

搜索安装：Arduino IDE → 项目 → 加载库 → 管理库 → 搜索 BMS26M833 → 安装



搜索安装流程 **1**

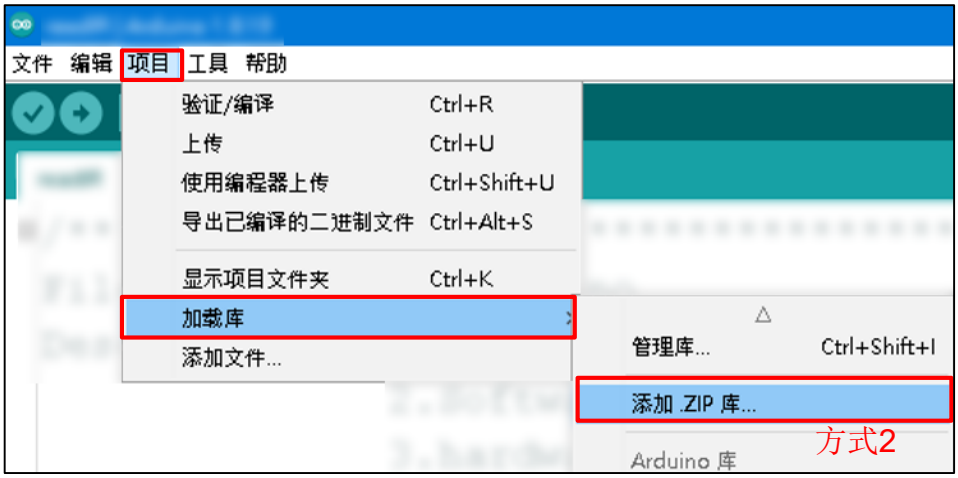


搜索安装流程 **2**

方式 2：添加 .ZIP 库，需提前下载 .ZIP 库

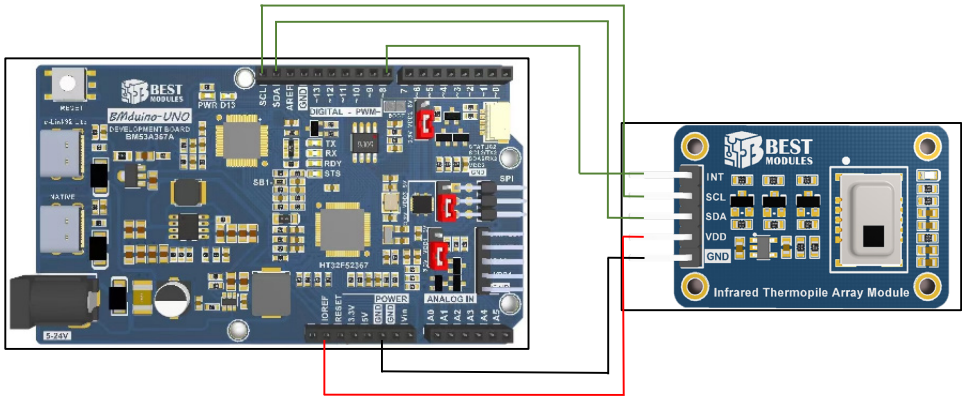
下载方法：打开倍创官方网站 (https://www.bestmodulescorp.com/bms26m833.html#tab-product2)“文件”目录下的 Arduino 范例程序 (BMS26M833 Library)。

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



Arduino 范例

范例 1：readTemperature



实物连接示意图

范例 1 实现功能：获取 8×8 温度值，并在串口监视器上显示

1. 范例打开：Arduino IDE → 文件 → 示例 → Lib 选择 (BMS26M833) → 选择对应范例 (readTemperature)

2．示例说明：

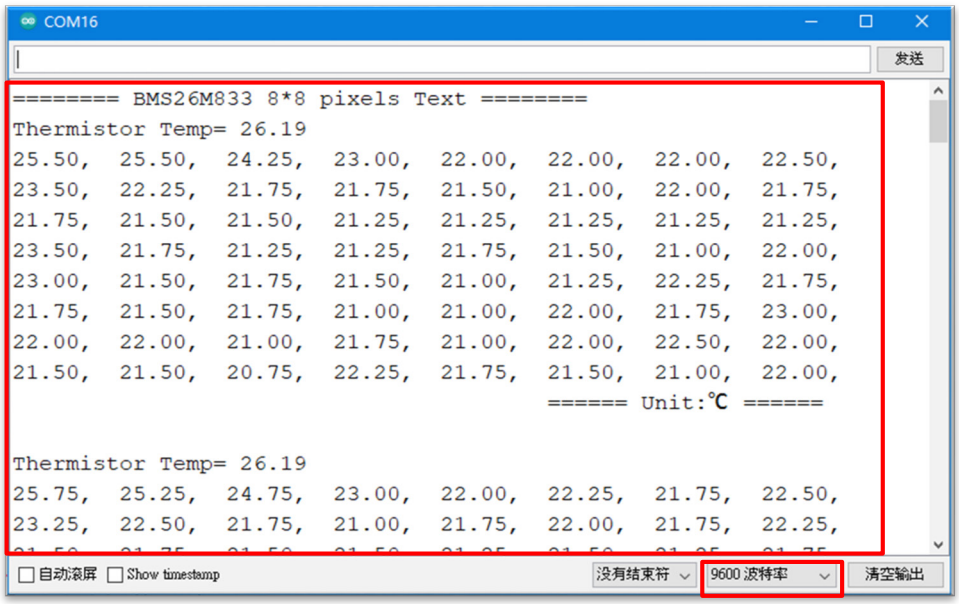
a. 创建对象 & 初始化模块

```
#include "BMS26M833.h"
BMS26M833 Amg; // 创建对象
float temp;
float TempMat[64];
void setup()
{
    Amg.begin();                  // 模块初始化
    Serial.begin(9600);           // 串口监视器初始化
    Serial.println("======== BMS26M833 8*8 pixels Text ========");
}
```

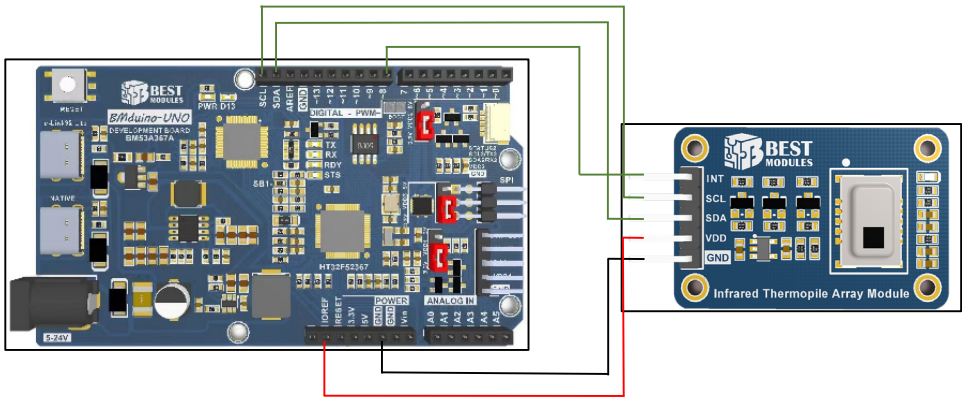
b. 获取温度值并在串口监视器中显示

```
void loop()
{
    Amg.readPixels(TempMat);          // 获取 8×8 温度矩阵
    temp = Amg.readThermistorTemp();// 获取环境温度
    Serial.println("\t\t====== Unit:°C ======");
    Serial.print("Thermistor Temp= ");
    Serial.println(temp);
    for(int x = 1; x <= 64 ; x++)     // 输出 8×8 温度矩阵
    {
      Serial.print(TempMat[x-1],1);
      Serial.print(",  ");
      if(x%8 == 0)  Serial.println();
    }
    Serial.println();
    delay(500);
}
```

3．打开串口监视器，波特率选择 9600；串口监视器显示如下：



范例 2：setInterrupt



实物连接示意图

范例 2 实现功能：使能模块的中断功能，获取 8×8 中断表，并在串口监视器上显示

1. 范例打开：Arduino IDE → 文件 → 示例 → Lib 选择 (BMS26M833) → 选择对应范例 (setInterrupt)
2. 示例说明：

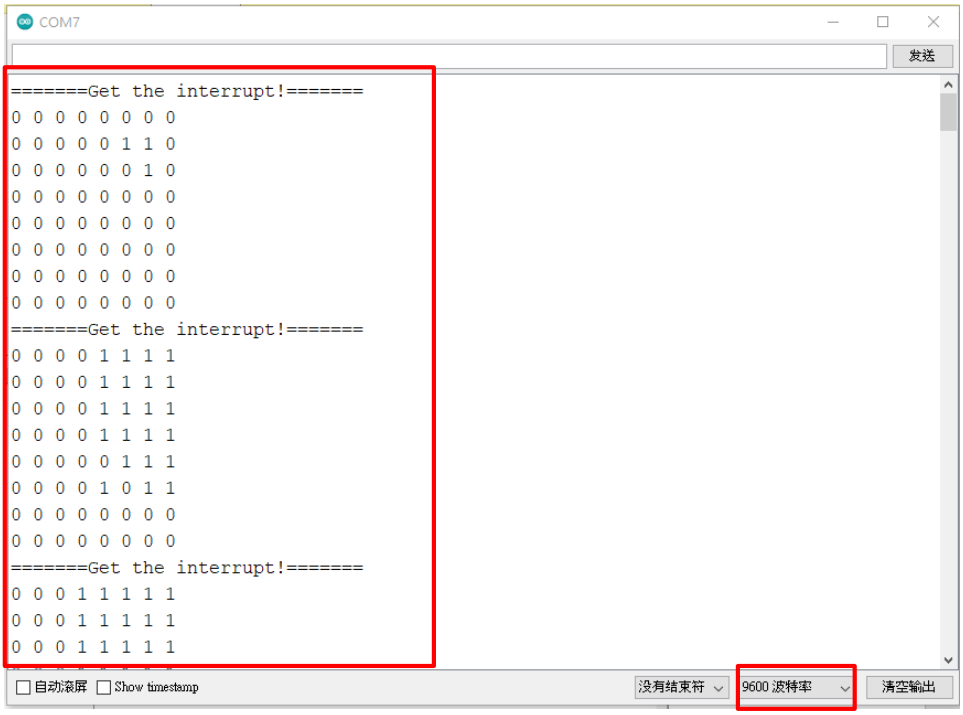
   a. 创建对象 & 初始化模块

```
#include <BMS26M833.h>
// 设定超温报警的上限值和下限值，需注意分辨率为 0.25°C
#define TEMP_INT_HIGH 30
#define TEMP_INT_LOW 15
BMS26M833 Amg;  // 创建对象
uint8_t interruptTable[8];
void setup()
{
  Serial.begin(9600);
  Amg.begin();// 模块初始化
  Amg.setInterruptLevels(TEMP_INT_HIGH, TEMP_INT_LOW);// 设定上限值与下
                                                      // 限值
  Amg.setINT(true);// 配置中断使能
}
```

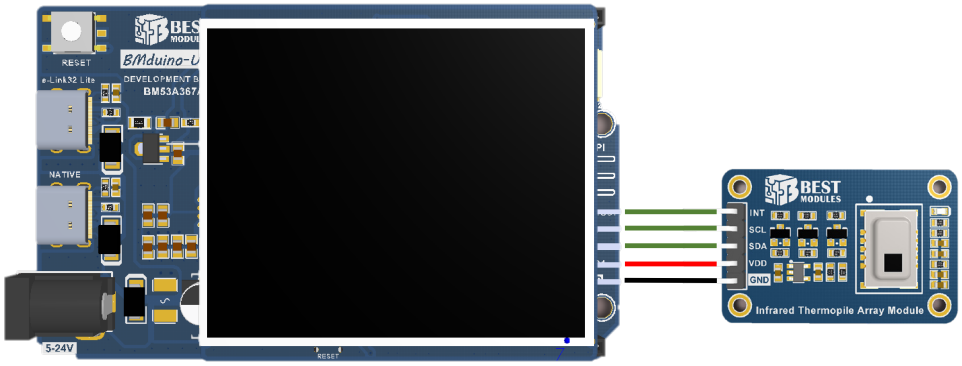
b. 获取中断表并在串口监视器中显示

```
void loop()
{
  if(Amg.getINT() == 0)// 当 INT 脚为低电平时，说明有中断发生
  {
    Amg.getINTTable(interruptTable);// 获取中断表
    Serial.println("=======Get the interrupt!=======");
    for(int i=0;i<8;i++)
    {
      for(int j=7;j>=0;j--)
      {
        bool intBit = interruptTable[i] & (1<<j);
        Serial.print(intBit);
        Serial.print(" ");
      }
    Serial.println();
    }
    Amg.setStatusClear();// 清除中断标志位以便于接收下一次中断
  }
}
```

3．打开串口监视器，波特率选择 9600；串口监视器显示如下：



范例 3：DisplayThermalImagingOnTheTFT



实物连接示意图

范例 3 实现功能：获取温度值，通过相关算法处理，生成热成像并配合使用 BMD58T280 TFT 模块显示图像

(需要下载 BMD58T280 相关的库文件，具体安装方法：Arduino IDE → 项目 → 加载库 → 管理库 → 搜索 BMD58T280 → 安装)

1. 范例打开：Arduino IDE → 文件 → 示例 → Lib 选择 (BMS26M833) → 选择对应范例 (DisplayThermalImagingOnTheTFT)
2. 示例说明：

   a. 构建 & 初始化模块

```
#include <SPI.h>
#include <BMD58T280.h>
#include “lcdfont.h”
#include <BMS26M833.h>

/* 热成像算法采用 c 编译 */
extern “C” {
  #include “InfraredThermalImaging.h”       // 算法
}
#if !defined(ARDUINO_AVR_UNO)// 选择 BMduino 开发板
BMD58T280 TFTscreen;          // 选择 EBI 通信
#else                         // 选择 UNO 开发板
BMD58T280 TFTscreen(&SPI);    // 选择 SPI 通信
#endif
/* 宏定义方便修改调试 */
#define OutMatWidth  55       // 输出热成像宽度
#define OutMatHeight 55       // 输出热成像高度
#define Magnification 3       // 输出热成像的放大倍数
#define TempDiffConfig  4     // 当温差小于此值时，认作无效热成像，不予显示，
                              // 单位：°C
#define BackgroundConfig 20   // 背景去除程度，0 为 DISABLE，值越大，目标与背
                             // 景之间的过度层越小，经验值为 20，范围为 0-255
#define ColorBarConfig Rainbow2_65K // 提前将调色板 RGB 转为 65K 值，显示过
                                    // 程中采用查表方式可以提升前端显示速度
```

```
ThermalImagingConfig ThermImaConfig;
BMS26M833 amg(22,&Wire1); //AM8833 Sencor 实例化

float TempMat[8 * 8];
uint8_t  DataBuff[OutMatWidth * OutMatHeight];
uint16_t timecnt=0;
float TempMax,TempMin;
char AxisPrintout[20];
uint16_t maging_xStart,maging_yStart;
uint32_t Systime = 0;

void setup() {
  Serial.begin(9600);          // 串口初始化
  TFTscreen.begin();           //TFT 初始化
  TFTscreen.setRotation(1);    // 设置屏幕显示方向
  amg.begin();                 // 模块初始
  LCDShowColorbar();
  maging_xStart = 100-Magnification*25;      // 定位热成像画面 x 坐标位置
  maging_yStart = 130-Magnification*25;      // 定位热成像画面 y 坐标位置
  ThermImaConfig.InWidth    = 8;
  ThermImaConfig.InHeight   = 8;
  ThermImaConfig.InMat      = TempMat;
  ThermImaConfig.OutMat     = DataBuff;
  ThermImaConfig.OutWidth   = OutMatWidth;
  ThermImaConfig.OutHeight  = OutMatHeight;
  ThermImaConfig.Background = BackgroundConfig;
  ThermImaConfig.TempDiff   = TempDiffConfig;
}
```

b. 获取温度值并通过算法处理，显示在 LCD 屏幕上

```
void loop()
{
  amg.readPixelsAndMaximum(TempMat, TempMax, TempMin); // 获取温度以及
                                                       // 最值
  if(Temp_Min>0 && Temp_Max<80)// 测温范围 0~80°C
  {
    InfraredThermalImaging(&ThermImaConfig);  // 算法处理
    LCDShow(DataBuff,Temp_Max,Temp_Min,OutMatWidth,OutMatHeight,
    TempDiffConfig,Magnification,maging_xStart,maging_yStart); // 在屏
                                                       // 幕上显示
  }
  else
  {
    LCDShow(DataBuff,0,0,OutMatWidth,OutMatHeight,TempDiffConfig,
    Magnification,maging_xStart,maging_yStart );
    delay(30);
  }
}
```

```
/* 调用此函数：更新温度值与热成像画面在 TFT 屏幕上 */
void LCDShow(uint8_t *Buff,float Max,float Min,uint8_t Width,uint8_t
Height,uint8_t TempDiff,uint8_t Mul,uint16_t xStart,uint16_t yStart)
{
  uint16_t temp;
  uint16_t color;
  uint16_t i,j,k,l,xLeng,yLeng;
  xLeng = Mul * Width;
  yLeng = Mul * Height;
  LCD_ShowFloatNum1(210,110,Max,3,BM_ILI9341::WHITE,BM_ILI9341::
  BLACK,3);
  LCD_ShowFloatNum1(210,190,Min,3,BM_ILI9341::WHITE,BM_ILI9341::
  BLACK,3);
  if((Max-Min) <= TempDiff)
  {
    color=ColorBarConfig[0];
    TFTscreen.fill(color);
    TFTscreen.noStroke();
    TFTscreen.rect(xStart,yStart,xLeng,yLeng);
    return;
  }
  TFTscreen.setAddrWindow(xStart,yStart,xLeng,yLeng);
  for(i = 0;i < Height;i++)
  {
    for(k = 0;k < Mul;k++)
    {
      for(j = 0;j < Width;j++)
      {
        temp = i * Width + j;
        color=ColorBarConfig[Buff[temp]];
        TFTscreen.writeColor(color,Mul);
      }
    }
  }
}
/* 调用此函数：显示小数在 TFT 屏幕上 */
void LCD_ShowFloatNum1(uint16_t x,uint16_t y,float num,
uint8_t len,u32 fc,u32 bc,uint8_t sizey)
{
  String strResult = String(num);
  strResult.toCharArray(AxisPrintout, 5);
  TFTscreen.fill(bc);
  TFTscreen.noStroke();
  TFTscreen.rect(x-1, y-2, 70, 26);//background color
  TFTscreen.stroke(fc);//word color
  TFTscreen.setTextSize(sizey);
  TFTscreen.text(AxisPrintout, x, y);//display variable
}
```

```
/* 调用此函数：显示初始画面在 TFT 屏幕上 */
void LCDShowColorbar()
{
  TFTscreen.background(BM_ILI9341::BLACK);
  TFTscreen.stroke(BM_ILI9341::WHITE);
  TFTscreen.setTextSize(3);
  TFTscreen.text(“MAX:”, 205, 70);
  TFTscreen.text(“MIN:”, 205, 150);
  TFTscreen.stroke(BM_ILI9341::WHITE);
  TFTscreen.setTextSize(3);
  TFTscreen.text(“BMS26M833”, 25, 15);
  TFTscreen.drawImage(288,105,gImage_tempLogo,32,32,0XFFFF);//Display
                                                            //°C
  TFTscreen.drawImage(288,185,gImage_tempLogo,32,32,0XFFFF);//Display
                                                            // °C
  TFTscreen.drawImage(200,0,gImage_BMLogo,104,44,0XFFFF);//Display
                                                            // BMLOGO
  uint8_t a=0;
  uint8_t i;
  a=0;
  for(i = 245;i > 0;i--)
  {
    TFTscreen.stroke(ColorBarConfig[i]);
    TFTscreen.line(0,a,18,a);
    a++;
  }
  TFTscreen.noStroke();
}
```

3．按要求连接电路，LCD 屏幕显示效果如下：