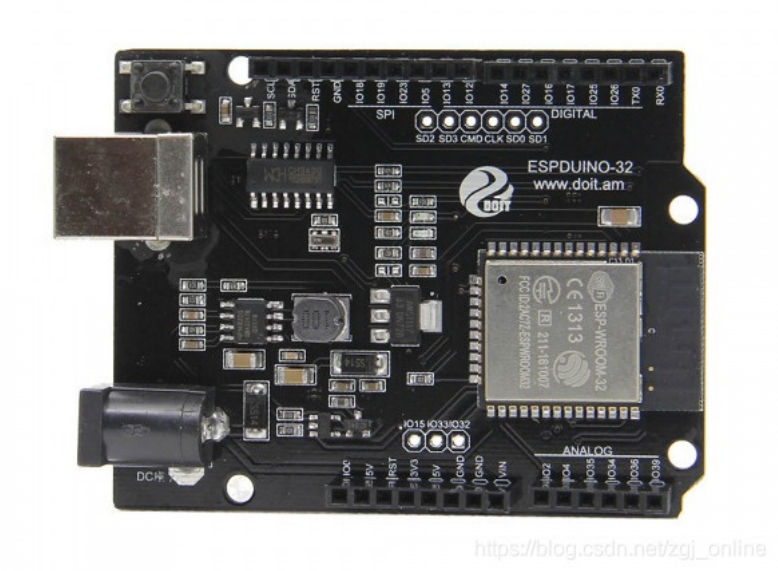
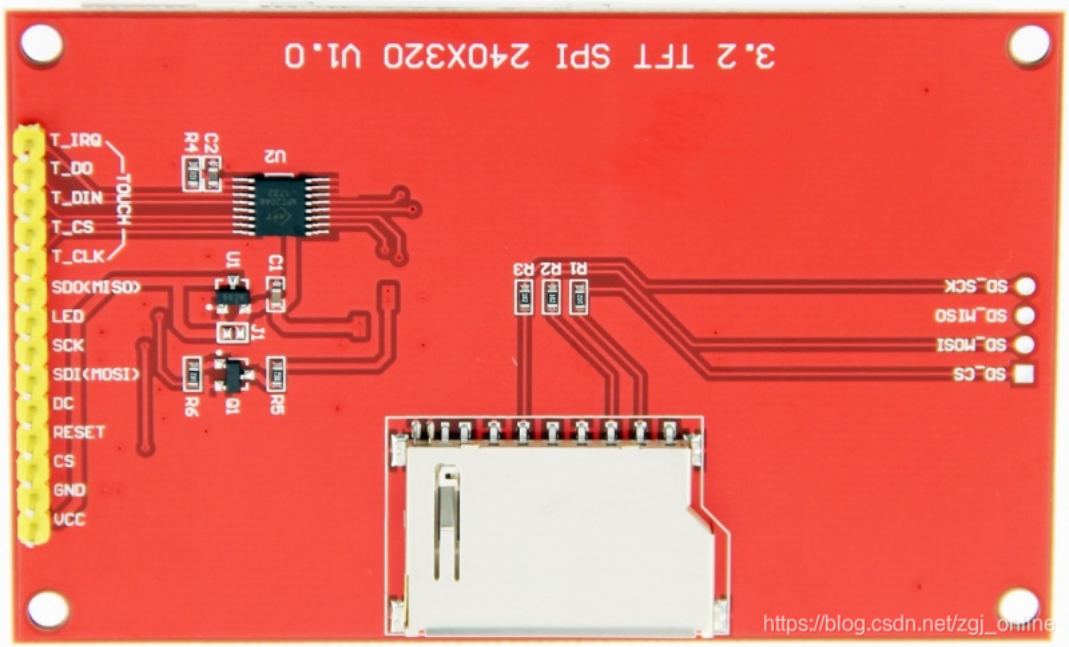
ESP32驱动3.2寸ILI9341显示屏+XPT2046触摸，GUIslice用户图形库

zgj_online 于 2020-03-20 16:22:24 发布

ESP32 的主板ESPDUINO-32如下：



屏用如下的：



显示驱动用TFT_eSPI，这个显示的速度比adafruit ILI9341快10倍。

一、配置TFT_eSPI:

arduino IDE 下载TFT_eSPI库，
TFT_eSPI库安装好后，进入C:\Users\xxx\Documents\Arduino\libraries\TFT_eSPI，可以从可以把User_Setup.h修改成ILI9341的配置文件，也可以用已经改好的，见下。如果文件名变了，或是用其它配置，要打开User_Setup_Select.h进行修改

注释下面的行

```
#include <User_Setup.h>           // Default setup is root library folder
```

然后添加自己的配置文件

```
#include <ILI9341.h> // ILI9341 3.2寸屏
```

ILI9341的配置文件及翻译如下：

```
//                            用户定义设置
//   设置驱动程序类型、要加载的字体、使用的引脚和SPI控制方法等
//
//   如果希望能够定义多个设置，然后轻松选择编译器使用的安装文件。
//   看文件User_Setup_Select.h
//
//   如果此文件编辑正确，则所有库示例程序都能运行，而无需对特定硬件设置进行任何更改！
//   注意，有些程序是为特定的TFT像素宽度/高度设计的

// ##################################################################################
//
// 第一节.调出正确的驱动程序文件及其选项
//
// ##################################################################################

// 定义STM32以调用优化的处理器支持（仅适用于STM32）
//#define STM32

// 定义STM32板允许库优化性能
// 对于UNO兼容的“MCUfriend”式防护罩
//#define NUCLEO_64_TFT
//#define NUCLEO_144_TFT

// 告诉库使用8位并行模式（否则假定为SPI）
//#define TFT_PARALLEL_8_BIT

// 显示类型-仅定义是否RPi显示
//#define RPI_DISPLAY_TYPE // 20MHz maximum SPI

```

```
// 只定义一个驱动程序，其他的必须注释掉
#define ILI9341_DRIVER
//#define ST7735_DRIVER      // Define additional parameters below for this display
//#define ILI9163_DRIVER     // Define additional parameters below for this display
//#define S6D02A1_DRIVER
//#define RPI_ILI9486_DRIVER // 20MHz maximum SPI
//#define HX8357D_DRIVER
//#define ILI9481_DRIVER
//#define ILI9486_DRIVER
//#define ILI9488_DRIVER     // WARNING: Do not connect ILI9488 display SDO to MISO if other devices share the SPI bus (TFT SDO doe
//#define ST7789_DRIVER      // Full configuration option, define additional parameters below for this display
//#define ST7789_2_DRIVER    // Minimal configuration option, define additional parameters below for this display
//#define R61581_DRIVER
//#define RM68140_DRIVER
//#define ST7796_DRIVER

// 一些显示屏支持通过MISO引脚读取SPI，其他显示屏有一个双向SDA引脚，库将尝试通过MOSI线读取。
// 要使用SDA行从TFT读取数据，请取消注释以下行：

// #define TFT_SDA_READ      // 此选项仅适用于ESP32，仅用ST7789显示屏测试

// 仅限ST7789和ILI9341，如果显示屏上的蓝色和红色互换，请定义颜色顺序
// 一次尝试一个选项，为您的显示屏找到正确的颜色顺序

//  #define TFT_RGB_ORDER TFT_RGB  // Colour order Red-Green-Blue
//  #define TFT_RGB_ORDER TFT_BGR  // Colour order Blue-Green-Red

// 对于仅集成ILI9341显示屏的M5Stack ESP32模块，删除下面行中的//

// #define M5STACK

// 仅限ST7789、ST7735和ILI9163，在纵向方向上定义像素宽度和高度
// #define TFT_WIDTH  80
// #define TFT_WIDTH  128
// #define TFT_WIDTH  240 // ST7789 240 x 240 and 240 x 320
// #define TFT_HEIGHT 160
// #define TFT_HEIGHT 128
// #define TFT_HEIGHT 240 // ST7789 240 x 240
// #define TFT_HEIGHT 320 // ST7789 240 x 320

// 仅限ST7735，定义显示类型，最初这是基于屏幕保护膜上标签的颜色，
// 但这并不总是正确的，因此如果屏幕不能正确显示图形，请尝试下面的不同选项，
// 例如颜色错误、镜像或边缘的托盘像素。注释掉ST7735显示驱动程序的所有选项
//  （除了其中一个选项），保存此用户设置文件，然后重新生成草图并再次将其上载到板：

// #define ST7735_INITB
// #define ST7735_GREENTAB
// #define ST7735_GREENTAB2
// #define ST7735_GREENTAB3
// #define ST7735_GREENTAB128    // For 128 x 128 display
// #define ST7735_GREENTAB160x80 // For 160 x 80 display (BGR, inverted, 26 offset)
// #define ST7735_REDTAB
// #define ST7735_BLACKTAB
// #define ST7735_REDTAB160x80   // For 160 x 80 display with 24 pixel offset

// 如果颜色是反转的（白色显示为黑色），则取消注释后的两行中的一行尝试两个选项，其中一个选项应更正反转。

// #define TFT_INVERSION_ON
// #define TFT_INVERSION_OFF

// 如果背光控制信号可用，则在下面第2节中定义TFT-BL引脚。
// 调用tft.begin()时，背光将打开，但库需要知道LED是否打开，
// 引脚是高还是低。如果led是用PWM信号驱动或关闭/打开的，
// 则必须由用户代码处理。例如.使用数字写入（TFT-BL，低）；

// #define TFT_BACKLIGHT_ON HIGH  // HIGH or LOW are options

// ##############################################################################
//
// 第二节.在此处定义用于与显示屏接口的引脚
//
```

```
// ##############################################################################

// 我们必须使用硬件SPI，至少需要3个GPIO引脚。
// ESP8266 NodeMCU ESP-12的典型设置为：
//
// Display SDO/MISO  to NodeMCU pin D6 (or leave disconnected if not reading TFT)
// Display LED       to NodeMCU pin VIN (or 5V, see below)
// Display SCK       to NodeMCU pin D5
// Display SDI/MOSI  to NodeMCU pin D7
// Display DC (RS/AO)to NodeMCU pin D3
// Display RESET     to NodeMCU pin D4 (or RST, see below)
// Display CS        to NodeMCU pin D8 (or GND, see below)
// Display GND       to NodeMCU pin GND (0V)
// Display VCC       to NodeMCU 5V or 3.3V
//
// TFT复位引脚可以连接到NodeMCU RST引脚或3.3V以释放控制引脚
//
// DC（Data Command数据命令）引脚可以标记为AO或RS（寄存器选择）
//
// 对于某些显示屏，如ILI9341，如果没有更多的SPI设备（如SD卡）连接，
// TFT CS引脚可以连接到GND，在这种情况下，请注释下面的“定义TFT CS”行，
// 以便不定义它。在ST7735上的其他显示屏需要在设置期间切换TFT-CS引脚，
// 因此在这些情况下，必须定义和连接TFT-CS线。
//
// NodeMCU D0 pin可用于RST
//
//
// 注意：只有部分版本的NodeMCU在VIN引脚上提供了USB 5V，
// 如果引脚上没有5V，则可以使用3.3V，但背光亮度会更低。

// ###### 编辑以下行中的引脚号以适合您的ESP8266设置 ######

// 对于NodeMCU-使用pin_Dx形式的引脚号，其中Dx是NodeMCU引脚名称
//#define TFT_CS   PIN_D8  // Chip select control pin D8
//#define TFT_DC   PIN_D3  // Data Command control pin
//#define TFT_RST  PIN_D4  // Reset pin (could connect to NodeMCU RST, see next line)
//#define TFT_RST  -1    // Set TFT_RST to -1 if the display RESET is connected to NodeMCU RST or 3.3V

//#define TFT_BL PIN_D1  // LED back-light (only for ST7789 with backlight control pin)

//#define TOUCH_CS PIN_D2     // Chip select pin (T_CS) of touch screen

//#define TFT_WR PIN_D2       // Write strobe for modified Raspberry Pi TFT only

// ######  对于ESP8266重叠模式，编辑以下行中的引脚号  ######

// 重叠模式与TFT共享ESP8266闪存SPI总线，因此对性能有影响，但为其他功能保存引脚。
// 最好不要连接MISO，因为当芯片选择引脚为高电平时，某些显示屏不会三态显示该行！
// 在NodeMCU 1.0上，SD0=MISO，SD1=MOSI，CLK=SCLK 以重叠模式连接到TFT
// 在NodeMCU V3上，  S0=MISO，S1=MOSI，S2=SCLK
// 在ESP8266重叠模式下，必须定义以下内容

//#define TFT_SPI_OVERLAP

// 在ESP8266重叠模式下，TFT芯片选择必须连接到引脚D3
//#define TFT_CS   PIN_D3
//#define TFT_DC   PIN_D5  // Data Command control pin
//#define TFT_RST  PIN_D4  // Reset pin (could connect to NodeMCU RST, see next line)
//#define TFT_RST  -1  // Set TFT_RST to -1 if the display RESET is connected to NodeMCU RST or 3.3V

// ###### 编辑以下行中的引脚号以适合您的ESP32设置   ######

// 用于ESP32开发板（仅用ILI9341显示屏测试）
// 硬件SPI可以映射到任何引脚
// 我把背光引脚LED接到了3.3V上，常亮了

#define TFT_CS   32  // 芯片选择控制引脚
//#define TFT_RST   4  // 复位引脚（可以连接到RST引脚）
#define TFT_RST  -1  // 如果显示屏复位连接到ESP32板RST，则将TFT_RST设置为-1
#define TFT_DC   33  // 数据命令控制引脚
#define TFT_MOSI 15
```

```
#define TFT_SCLK 4
#define TFT_MISO 2

//#define TFT_BL   32  // LED背光 (仅适用于带背光控制引脚的ST7789)

//#define TOUCH_CS 14     // 触摸屏的芯片选择引脚 (T_CS),

//#define TFT_WR 22    // 仅适用于改性树莓派TFT的写入选通 Write strobe for modified Raspberry Pi TFT only

// 对于M5Stack模块，使用以下定义行
//#define TFT_MISO 19
//#define TFT_MOSI 23
//#define TFT_SCLK 18
//#define TFT_CS   14  // Chip select control pin
//#define TFT_DC   27  // Data Command control pin
//#define TFT_RST  33  // Reset pin (could connect to Arduino RESET pin)
//#define TFT_BL   32  // LED back-light (required for M5Stack)

// ######       编辑下面的引脚以适合您的ESP32并行TFT设置        ######

// 库支持ESP32的8位并行TFT，下面的引脚选择与UNO格式的ESP32板兼容。
// 需要修改Wemos D32板，请参阅“工具”文件夹中的图表。
// 只测试了基于ILI9481和ILI9341的显示器！

// 仅ESP32支持并行总线
// 取消下面的注释行以使用ESP32并行接口而不是SPI

//#define ESP32_PARALLEL

// 用于测试的ESP32和TFT引脚为：
//#define TFT_CS   33  // Chip select control pin (library pulls permanently low
//#define TFT_DC   15  // Data Command control pin - must use a pin in the range 0-31
//#define TFT_RST  32  // Reset pin, toggles on startup

//#define TFT_WR    4  // Write strobe control pin - must use a pin in the range 0-31
//#define TFT_RD    2  // Read strobe control pin

//#define TFT_D0   12  // Must use pins in the range 0-31 for the data bus
//#define TFT_D1   13  // so a single register write sets/clears all bits.
//#define TFT_D2   26  // Pins can be randomly assigned, this does not affect
//#define TFT_D3   25  // TFT screen update performance.
//#define TFT_D4   17
//#define TFT_D5   16
//#define TFT_D6   27
//#define TFT_D7   14

// ##################################################################################
//
// 第三节.定义此处使用的字体
//
// ##################################################################################

// 用//注释掉下面的定义，以停止加载该字体
// ESP8366和ESP32有足够的内存，因此通常不需要注释字体。
// 如果加载了所有字体，则所需的额外闪存空间约为17Kbytes。
// 为了节省内存空间，只启用您需要的字体！

#define LOAD_GLCD   // 字体 1. 原来的Adafruit 8像素字体需要约1820字节的FLASH
#define LOAD_FONT2  // 字体 2. 16像素高的小字体，需要大约3534字节的FLASH，96个字符
#define LOAD_FONT4  // 字体 4. 中等26像素高字体，FLASH需要5848字节，96个字符
#define LOAD_FONT6  // 字体 6. 48像素的大字体，FLASH需要2666字节，只有字符1234567890:-.
#define LOAD_FONT7  // 字体 7. 7段48像素字体，FLASH需要约2438字节，仅字符1234567890:-.
#define LOAD_FONT8  // 字体 8. 75像素的大字体在FLASH中需要3256字节，只有1234567890个字符:-.
//#define LOAD_FONT8N // 字体 8. 上面字体8的替代品，稍微窄一些，因此3位数字适合160像素的TFT
#define LOAD_GFXFF  // 自由字体. 包括访问48个Adafruit_GFX免费字体FF1到FF48和自定义字体

// 注释掉下面的定义，以停止SPIFFS文件系统并平滑加载字体代码
// 这将节约 ~20kbytes of flash
#define SMOOTH_FONT

// ##################################################################################
```

```
//
// 第四节.其他选项
//
// ###########################################################################

// 定义SPI时钟频率，这会影响图形渲染速度。速度太快，TFT驱动程序无法跟上，显示不正确。
// 对于ILI9341显示屏，40MHz工作正常，80MHz有时出现故障
// 对于ST7735显示屏，超过27MHz可能无法工作（杂散像素和线）
// 对于ILI9163显示屏，27MHz工作正常。

// #define SPI_FREQUENCY   1000000
// #define SPI_FREQUENCY   5000000
// #define SPI_FREQUENCY  10000000
// #define SPI_FREQUENCY  20000000
#define SPI_FREQUENCY  27000000 // 实际设置为 26.67MHz = 80/3
// #define SPI_FREQUENCY  40000000
// #define SPI_FREQUENCY  80000000

// 用于读取TFT的可选降低SPI频率
#define SPI_READ_FREQUENCY  20000000

// XPT2046（触摸屏驱动库）需要2.5MHz的较低SPI时钟速率，因此我们在此定义：
#define SPI_TOUCH_FREQUENCY  2500000

// ESP32有两个空闲的SPI端口，即VSPI和HSPI，VSPI是默认端口。
// 如果VSPI端口正在使用中，并且无法访问管脚（例如TTGO T-Beam）
// VSPI: CS->5  SCLK->18 MISO->19 MOSI->23
// HSPI: CS->15 SCLK->14 MISO->12 MOSI->13
// 则取消注释以下行:
// 重要：如果触摸屏要独立使用XPT2046_TouchScreen驱动，下面这行一定要取消注释，2天的工夫找到这行
#define USE_HSPI_PORT

// 如果不需要支持“SPI事务”，请注释掉以下定义。
// 当被注释掉，代码大小会变小，程序运行得稍微快一点，
// 所以除非你需要，否则就不要注释

// 使用SD库需要事务支持，但不需要TFT-SdFat
// 如果连接了其他SPI设备，则需要事务支持。

// 库为ESP32自动启用事务（使用HAL互斥）
// 所以在这里改变它没有效果

// #define SUPPORT_TRANSACTIONS
```

重要：如果触摸屏要独立使用XPT2046_TouchScreen驱动，#define USE_HSPI_PORT这行一定要取消注释，2天的工夫找到这行

二、配置XPT2046

arduino IDE下载：XPT2046_TouchScreen库。
触摸搞了好久，XPT2046库下了，但没反应。后来下了adafruit touchscreen驱动，发现不支持ESP32，国外有人改写了驱动，但引脚要重新连线，看着就不放心。
XPT2046之前之所以没反应，是配置只有两个脚：

```
#define TOUCH_CS_PIN       14
#define TOUCH_IRQ_PIN      27
```

可是屏后面触摸的引脚有5个T_IRQ、T_DO、T_DIN、T_CS、T_CLK，咋整？查了很多资料才悟道，该驱动使用ESP32的硬件SPI，不进行引脚映射。
接线如下

触摸屏引脚	触摸屏引脚说明	ESP32引脚	ESP32引脚说明
T_IRQ	触摸屏中断信息，检测到触摸时为低电平	27	可自定义引脚
T_DO	触摸SPI总线输出，不可自定义引脚	19（MISO）	VSPI引脚MISO
T_DIN	触摸SPI总线输入，不可自定义引脚	23（MOSI）	VSPI引脚MOSI
T_CS	触摸片选信号，低电平使能，可自定义引脚	14	可自定义引脚
T_CLK	触摸SPI总线时钟信号，不可自定义引脚	18（SCLK）	SCLK

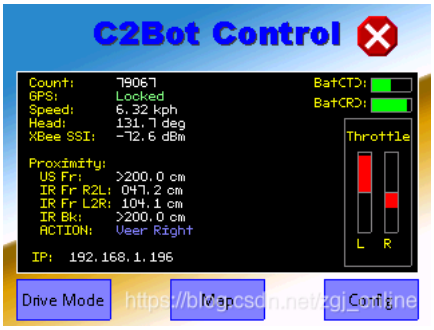
触摸屏引脚	触摸屏引脚说明	ESP32引脚	ESP32引脚说明

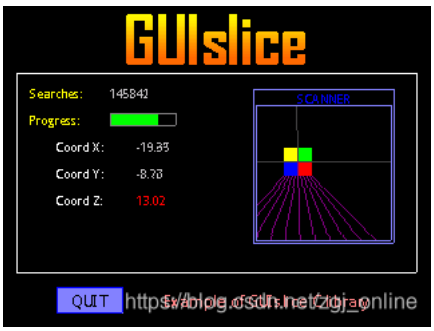
触摸屏测试代码

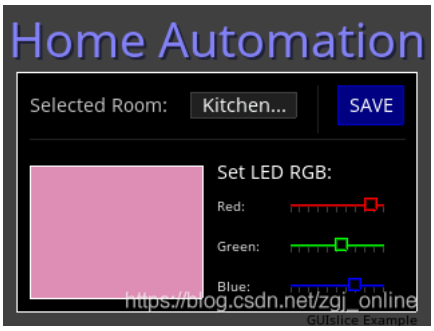
```
/* Map XPT2046 input to ILI9341 320x240 raster */
#include "SPI.h"
#include <XPT2046_Touchscreen.h> /* https://github.com/PaulStoffregen/XPT2046_Touchscreen */

#define TOUCH_CS_PIN       14
#define TOUCH_IRQ_PIN      27

XPT2046_Touchscreen touch( TOUCH_CS_PIN, TOUCH_IRQ_PIN );

void setup() {
  Serial.begin( 115200 );
  touch.begin();
  Serial.println( "Touch screen ready." );
}

TS_Point rawLocation;

void loop() {
    while ( touch.touched() )
    {
      rawLocation = touch.getPoint();
        Serial.print("x = ");
        Serial.print(rawLocation.x);
        Serial.print(", y = ");
        Serial.print(rawLocation.y);
        Serial.print(", z = ");
        Serial.println(rawLocation.z);
    }
}
```

三、GUIslice用户界面图形库

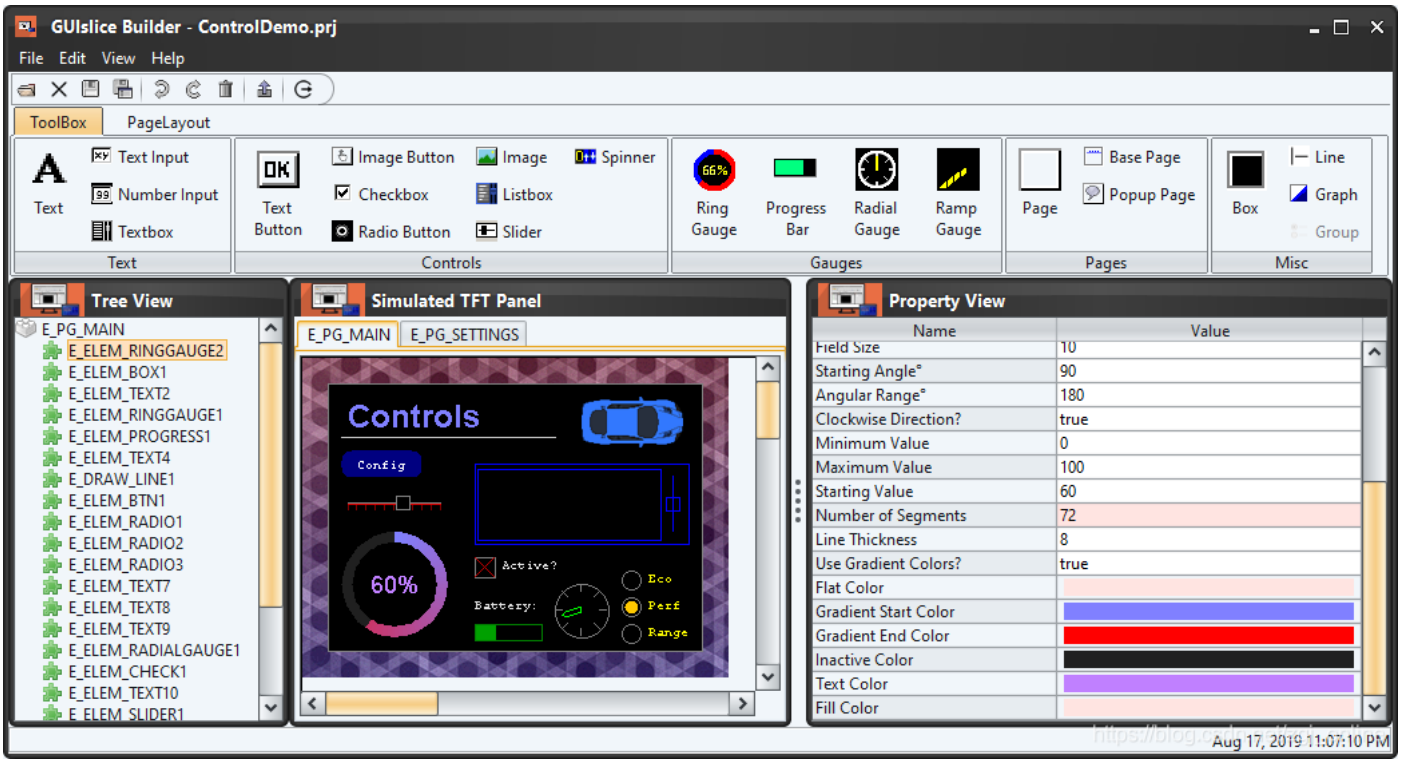
发现一个专用于ESP32+TFT_eSPI+XPT2046的嵌入式GUI库 GUIslice， https://github.com/ImpulseAdventure/GUIslice
有这个库，操作界面省事不少啊，否则自己去画有事干了。
简单介绍一下，在单片机端的显示界面如：







这也太漂亮了。还有一个工具，GUIslice BUilder，把需要的控件拖拉进去就可以生成图形操作界面了，如下图：



然后点菜单 File->Generate Code就可以生成代码了，用arduino IDE打开就能编译上传了。

GUIslice配置：

1.arduino IDE要下载相应的GUIslice，菜单 工具->管理库， 搜索GUIslice就能下载。

2.打开：C:\Users\xxx\Documents\Arduino\libraries\GUIslice\src\GUIslice_config.h

取消注释下面行，同学们根据自己的板类型自行选择啊，我是这一行。

```
1 //#include "../configs/esp-tftespi-default-xpt2046.h"
```

3.打开上面对应的文件：C:\Users\zheng\Documents\Arduino\libraries\GUIslice\configs\esp-tftespi-default-xpt2046.h
修改触摸屏的CS引脚号：(貌似不用设置中断引脚)

```
#define XPT2046_CS     14
```

打开示例代码愉快的玩玩吧。
官方配置说明： https://github.com/ImpulseAdventure/GUIslice/wiki/Configure-GUIslice-for-ESP8266-&-ESP32

触摸参数需要调校：
打开示例：GUIslice->arduino->diag_ard_touch_calib
运行后，点击屏幕四角的点，最后会出现参数

- ADATOUCH_X_MIN
- ADATOUCH_X_MAX
- ADATOUCH_Y_MIN
- ADATOUCH_Y_MAX
  把这些数值修改到esp-tftespi-default-xpt2046.h中。