

## 目录

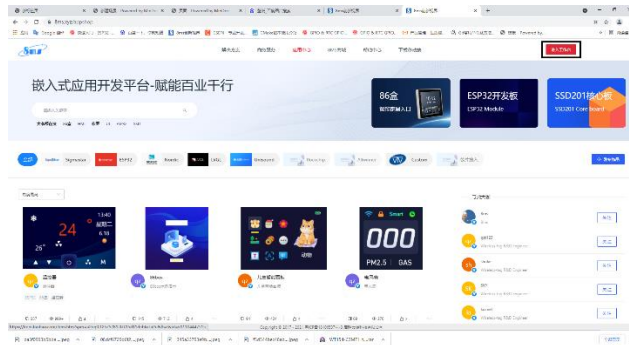
新建项目 .....	2
a) 登录到 8ms 平台 .....	2
b) 新建项目 : .....	2
c) 选择目标串口屏型号 .....	2
d) 创建场景 .....	2
编辑 UI .....	4
a) 准备好图片资源 .....	4
b) 将修改好的资源导入到 8ms 平台上 .....	4
c) 设置图片控件显示 .....	4
d) 可以使用在线编译预览 .....	5
e) 编辑其他场景的 UI .....	5
逻辑功能编辑 .....	7
a) 积木介绍 : .....	7
b) 设置初始化逻辑 .....	7
c) 设置获取串口数据并执行 相应代码,即从串口 输入 “abc” 或者 “a” 便执行显示 场景 “main_screen” .....	7
d) 设置逻辑 .....	8
保存下载 .....	9
a) 编译前需要保存 .....	9
b) 编译 .....	9
c) 下载好 bin 文件 .....	9
配置烧入工具 .....	11
a) 选择目标芯片 .....	11
b) 确定好下载的 bin 文件及应烧录的物理地址 .....	11
烧入及实际效果演示 .....	13
a) 接口图 .....	13
b) USB-TTL 与 C3SI 接线图 : .....	13
c) 打开设备管理器查看端口 .....	13
d) 上述操作无误后,下载 bin 文件到开发板 .....	14
c) 重启以运行 .....	14

## 新建项目

### a) 登录到 8ms 平台

<https://8ms.xyz/appshop>

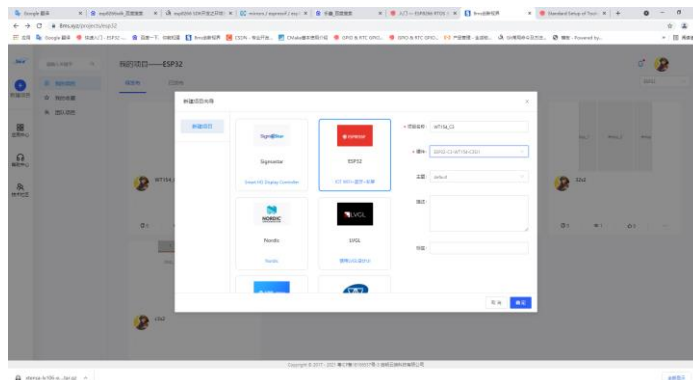
创建好账号并 登入, 进入工作台



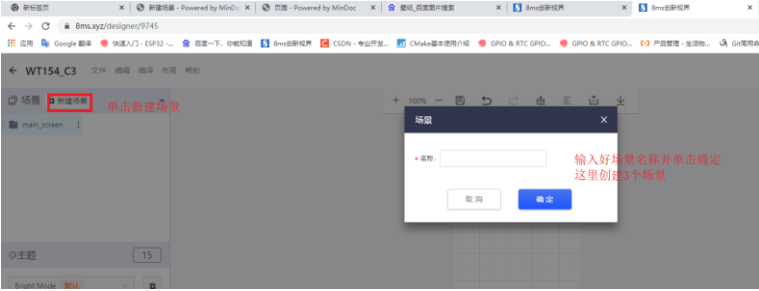
### b) 新建项目：



### c) 选择目标串口屏型号



d) 创建场景

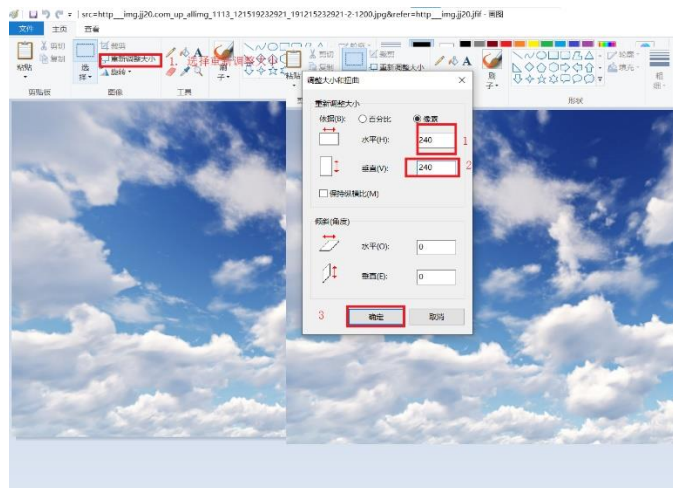


## 编辑 UI

### a) 准备好图片资源

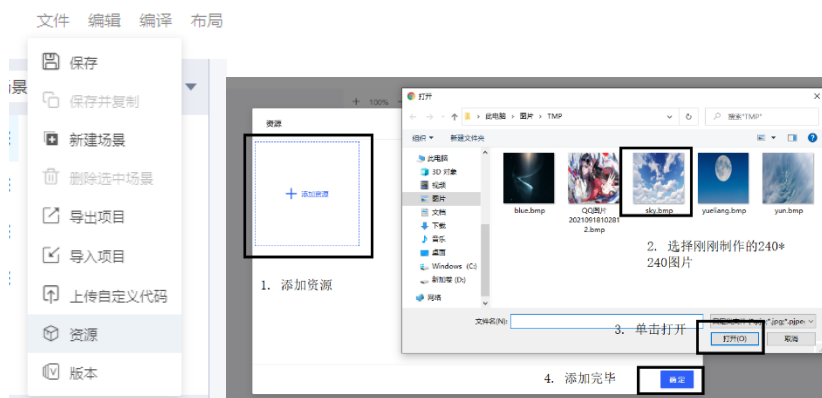


并用画图软件将下载好的图片资源整理成 240\*240 ( 因为该屏幕分辨率为 240x240)

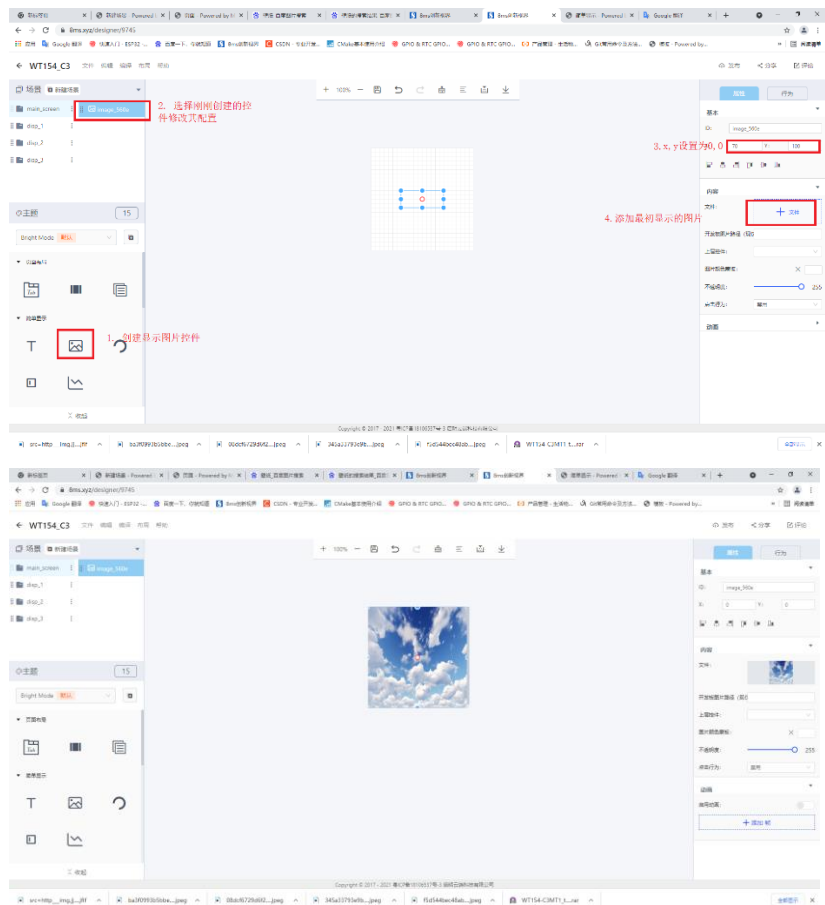


将文件另存为或者保存到自己的路径

### b) 将修改好的资源导入到 8ms 平台上



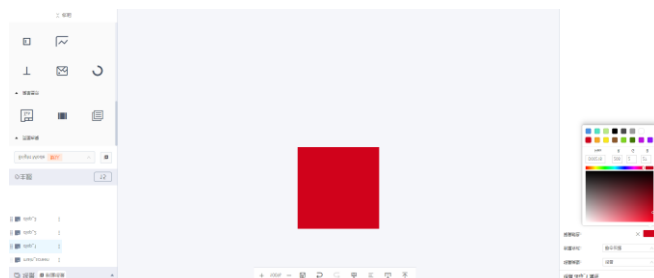
### c) 设置图片控件显示



d) 可以使用在线编译预览



e) 编辑其他场景的 UI



可将 disp\_1 背景颜色设置为 红色

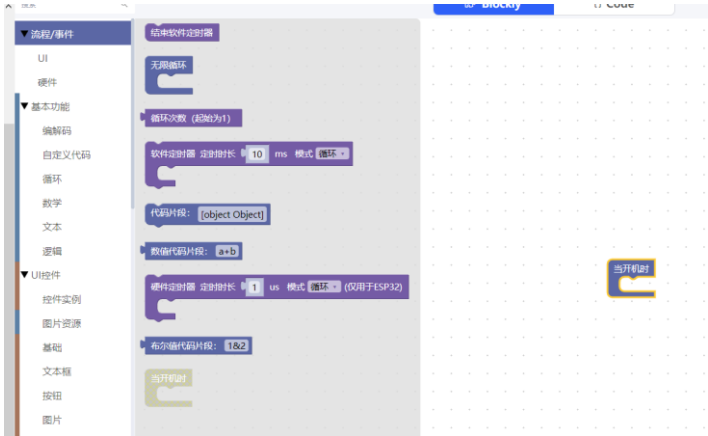


逻辑功能编辑

- a) 积木介绍：
- 布局中选择 积木



自定义代码 中 段外代码 可以头文件和 全局变量及宏定义

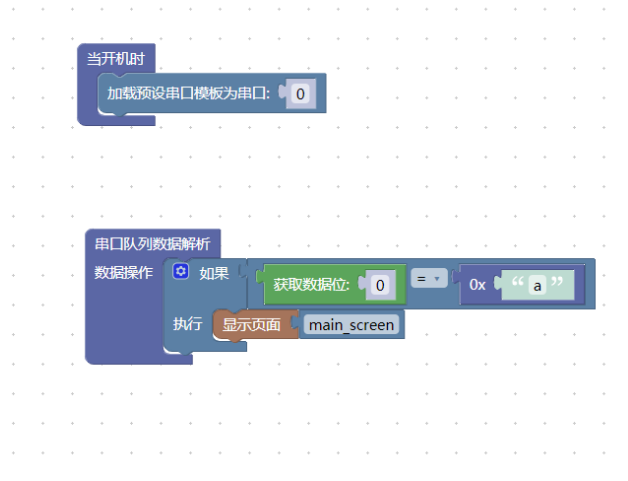


当开机时 逻辑 积木，此积木在线程开始之前便调用,因此此时未启动 UI  
所以此积木内代码执行时并不能显示  
一般执行 初始化，及软件定时器代码

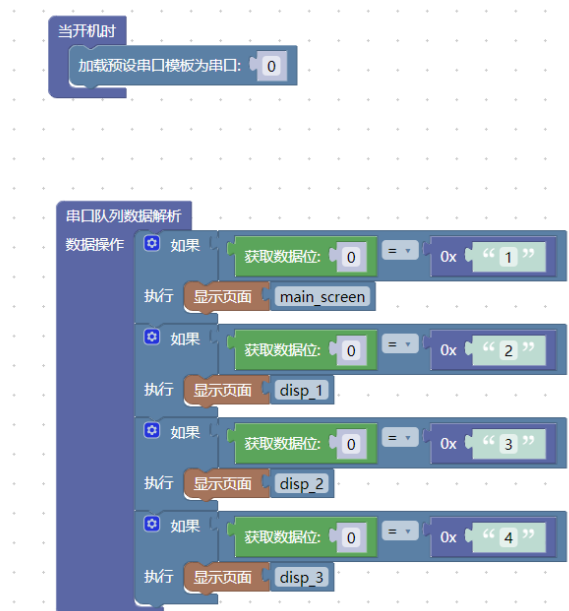
- b) 设置初始化逻辑
- 找到基础硬件中的 Preset 积木中的 “加载预设串口模板为串口”
- 设置 加载预设串口模板为串口 0, 即初始化串口 0 为与外界通信串口  
(串口 0 为烧入串口)



- c) 设置获取串口数据并执行 相应代码,即从串口 输入 “abc” 或者 “a” 便执行显示场景  
“main\_screen”



d) 设置逻辑





## 保存下载

### a) 编译前需要保存



### b) 编译



这里编译只能选择编译 需要生成 bin 文件和源代码 时间相较于在线编译长很多

```
Running make in directory /mnt/9832/build
Executing "make -j 6 all"...
Project build complete. To flash, run this command:
/root/.espressif/python_env/idf4.3_py3.8_env/bin/python ./../root/esp/esp-
idf/components/esptool_py/esptool/esptool.py -p (PORT) -b 460800 --before default_reset --after hard_reset --chip
esp32c3 write_flash --flash_mode dio --flash_size detect --flash_freq 80m 0x0 build/bootloader/bootloader.bin 0x8000
build/partition_table/partition-table.bin 0x10000 build/lvgl-demo.bin
or run 'idf.py -p (PORT) flash'
编译进程退出, 返回值:0
时间花费: 69993
没有设置远程git仓库跳过
编译成功 !!!
Done !!!
```

记下每个 bin 文件硬件地址


例如 : build/lvgl-demo.bin 0x10000

为下面的烧入做准备

### c) 下载好 bin 文件

编译 布局 帮助

 编译

 编译(在线预览)

 预览

 刷新

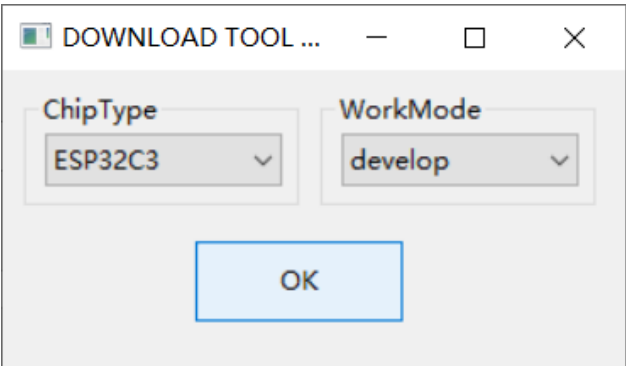
 下载源码

 开发板运行

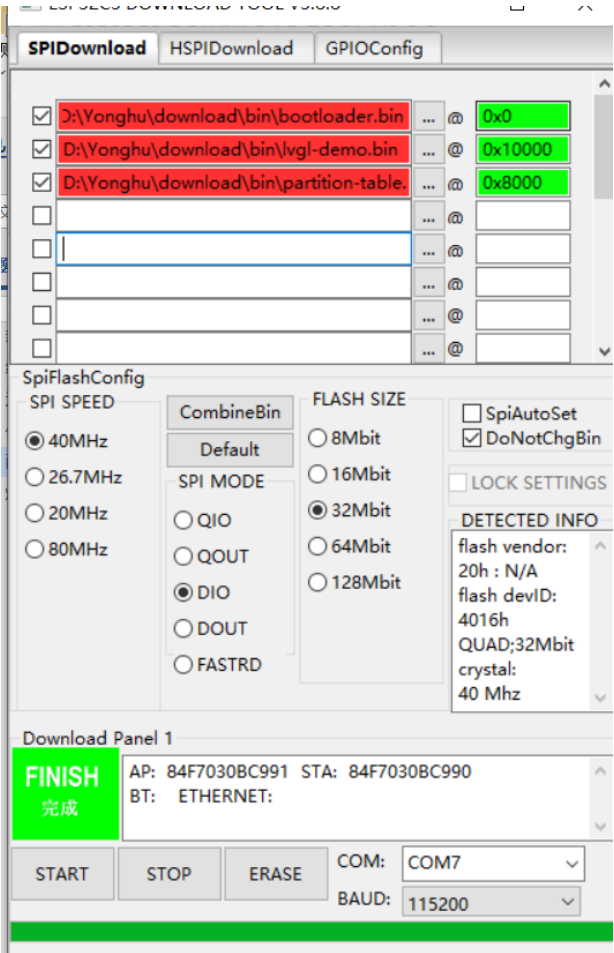
 下载bin

配置烧入工具

a) 选择目标芯片



b) 确定好下载的 bin 文件及应烧录的物理地址



当 bin 文件不存在时如图文件目录标红  
配置为如图：

SPIDownloadHSPIDownloadGPIOConfig

☒D:\Yonghu\download\bin\bootloader.bin...@0x0

☒D:\Yonghu\download\bin\lvgl-demo.bin...@0x10000

☒D:\Yonghu\download\bin\partition-table...@0x8000

☐

☐

☐

☐

☐

SpiFlashConfig

SPI SPEED

☒40MHz

☐26.7MHz

☐20MHz

☐80MHz

CombineBin

Default

SPI MODE

☐QIO

☐QOUT

☒DIO

☐DOUT

☐FASTRD

FLASH SIZE

☐8Mbit

☐16Mbit

☒32Mbit

☐64Mbit

☐128Mbit

☐SpiAutoSet

☒DoNotChgBin

☐LOCK SETTINGS

DETECTED INFO

flash vendor: ^

20h : N/A

flash devID: 4016h

QUAD;32Mbit

crystal: 40 Mhz

▼

Download Panel 1

FINISH

完成

AP: 84F7030BC991 STA: 84F7030BC990

BT: ETHERNET:

START

STOP

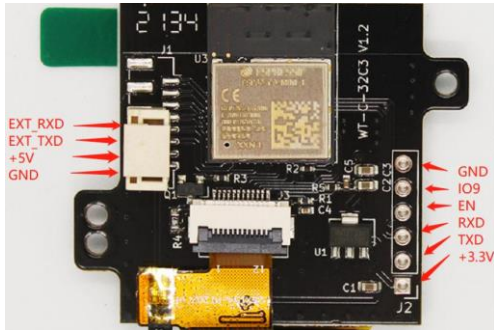
ERASE

COM: COM7

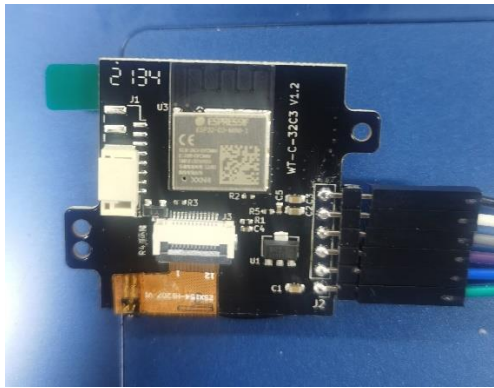
BAUD: 115200

烧入及实际效果演示

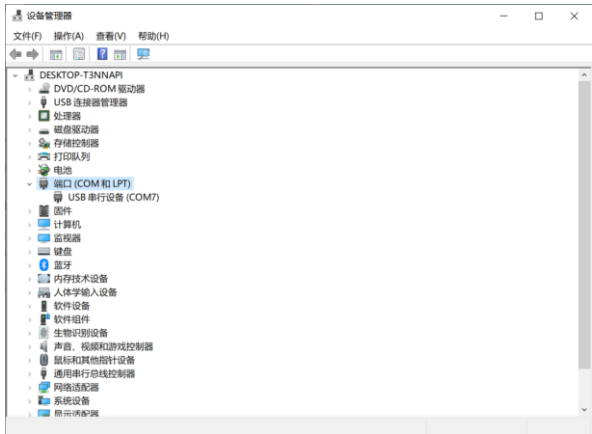
a) 接口图



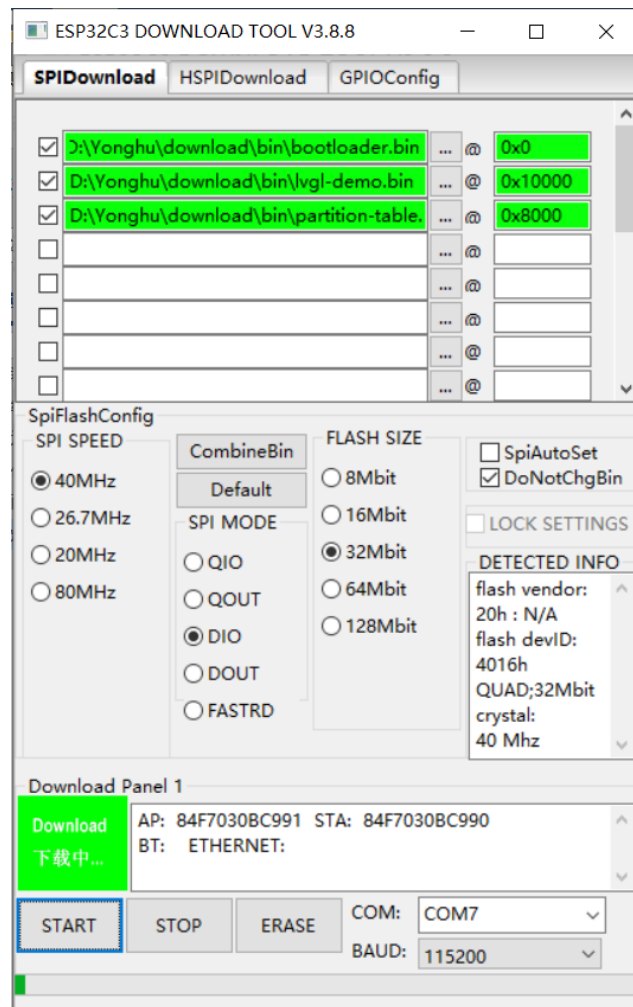
b) USB-TTL 与 C3SI 接线图：



c) 打开设备管理器查看端口



d) 上述操作无误后,下载 bin 文件到开发板

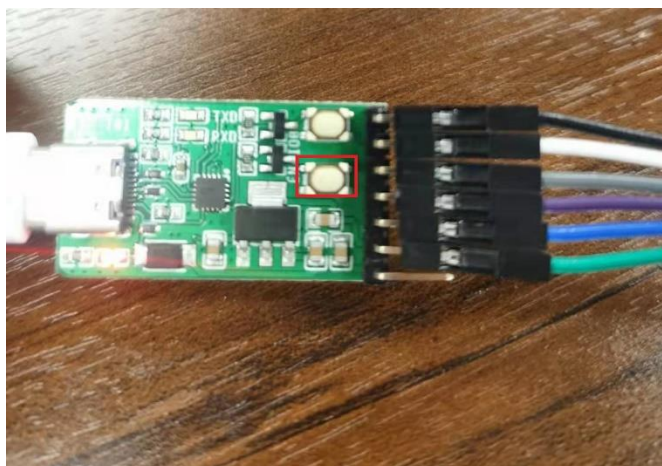


c) 重启以运行

等待下载完毕后

按下 USB-TTL 的 EN

或者重新上电



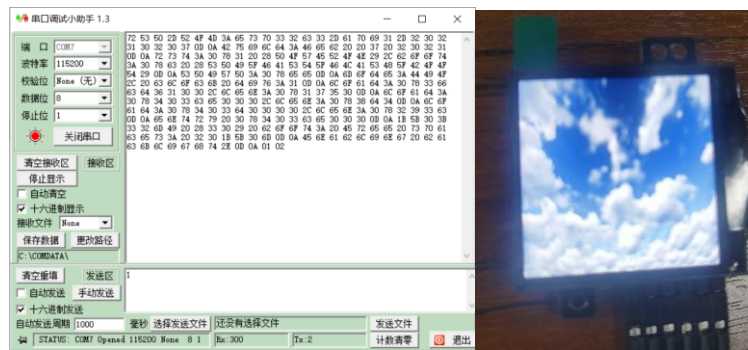
d) 使用

配置 串口助手

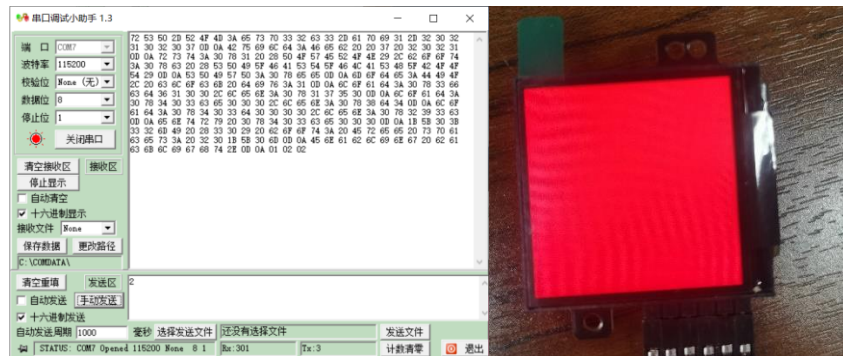


1. 端口设置为连接端口
2. 波特率配置为115200
3. 设置为16进制输入
4. 设置16进制输出, 该串口会将接收到的数据回发
5. 配置完毕, 打开串口

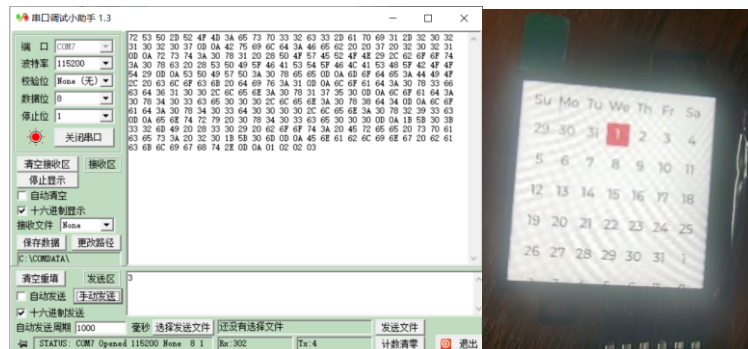
从串口输入 16 进制数据 1, 显示场景 main\_screen, 开始时默认显示场景 main\_screen



从串口输入 16 进制数据 2, 显示场景 disp\_1, 此时该场景中只有背景色便显示为背景色红



从串口输入 16 进制数据 2, 显示场景 disp\_2 日历控件



从串口输入 16 进制数据 2, 显示场景 disp\_3 文本控件

