MicroPython-LVGL 固件显示中文

1、系统版本

Windows 10 专业版 22H2
Thonny 4.0.2
Rubik Cube1 with ESP32（MicroPython v1.19）

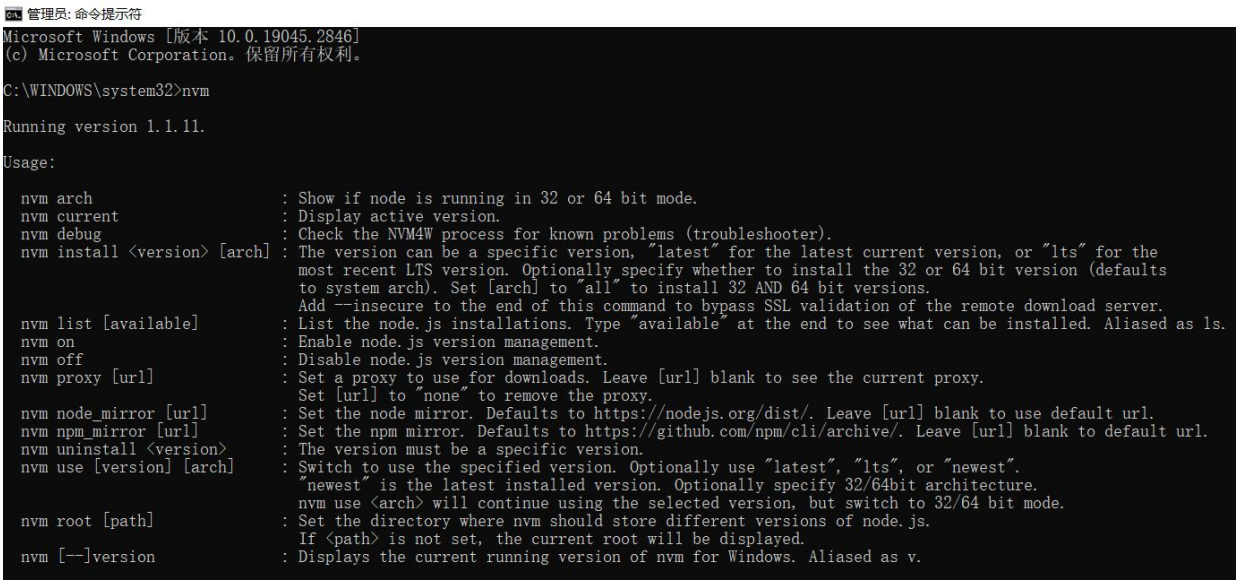
2、操作流程

2.1、 安装制作中文字体工具

2.1.1、 安装 NVM

从官方网站下载,使用默认选项进行安装。官方下载地址：
https://github.com/coreybutler/nvm-windows/releases

用管理员身份运行 Windows 命令提示符，输入 nvm，验证 NVM 是否安装成功，如果输出以下信息则代表成功。



2.1.2、 安装 Node.js

为了能够让 NVM 安装 Node 时更快，使用淘宝镜像。分别执行下面 2 个命令：

```
nvm node_mirror https://npm.taobao.org/mirrors/node/
nvm npm_mirror https://npm.taobao.org/mirrors/npm/
```

安装 Node.js，命令如下：

```
nvm install 16.18.0
```

```
C:\WINDOWS\system32>nvm install 16.18.0
Downloading node.js version 16.18.0 (64-bit)...
Extracting node and npm...
Complete
npm v8.19.2 installed successfully.


Installation complete. If you want to use this version, type

nvm use 16.18.0
```

查看安装是否成功，执行下面命令

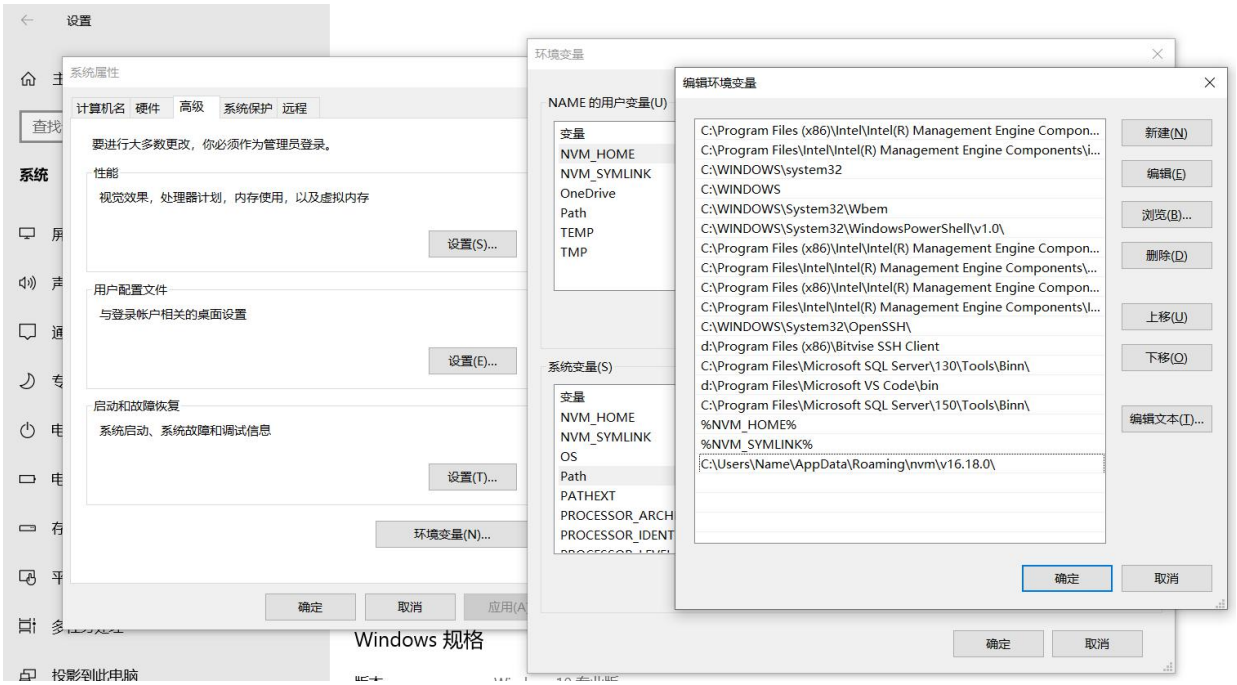
```
nvm list
```

```
C:\WINDOWS\system32>nvm list

  * 16.18.0 (Currently using 64-bit executable)
```

2.1.3、 设置系统变量 Path 路径

找到 v16.18.0 文件夹位置，在环境变量的系统变量下的 Path 中添加该文件夹位置



2.1.4、 安装 LVGL 字体生成模块

使用淘宝镜像，加速安装。在 Windows 命令提示符窗口输入以下命令，

```
npm config set registry https://registry.npm.taobao.org
```

```
C:\WINDOWS\system32>npm config set registry https://registry.npm.taobao.org
```

使用 npm 安装 LVGL 字体生成模块，输入以下命令：

```
npm i lv_font_conv -g
```

```
C:\WINDOWS\system32>npm i lv_font_conv -g

added 1 package in 2s
npm notice
npm notice New major version of npm available! 8.19.2 -> 9.6.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.4
npm notice Run npm install -g npm@9.6.4 to update!
npm notice
```

为验证安装工具是否成功，在命令提示符窗口输入以下命令：

`lv_font_conv`

如果出现以下信息，则代表安装成功：

```
C:\WINDOWS\system32>lv_font_conv
usage: lv_font_conv.js [-h] [-v] --size PIXELS [-o <path>] --bpp {1,2,3,4,8} [--lcd | --lcd-v] [--use-color-info] --format {dump,bin,lvgl} --font <path> [-r RANGE] [--symbols SYMBOLS]
                       [--autohint-off] [--autohint-strong] [--force-fast-kern-format] [--no-compress] [--no-prefilter] [--no-kerning] [--lv-include <path>] [--full-info]

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  --size PIXELS         Output font size, pixels.
  -o <path>, --output <path>
                        Output path.
  --bpp {1,2,3,4,8}     Bits per pixel, for antialiasing.
  --lcd                 Enable subpixel rendering (horizontal pixel layout).
  --lcd-v               Enable subpixel rendering (vertical pixel layout).
  --use-color-info      Try to use glyph color info from font to create grayscale icons. Since gray tones are emulated via transparency, result will be good on contrast background only.
  --format {dump,bin,lvgl}
                        Output format.
  --font <path>         Source font path. Can be used multiple times to merge glyphs from different fonts.
  -r RANGE, --range RANGE
                        Range of glyphs to copy. Can be used multiple times, belongs to previously declared "--font". Examples:
                        -r 0x1F450
                        -r 0x20-0x7F
                        -r 32-127
                        -r 32-127,0x1F450
                        -r '0x1F450=>0xF005'
                        -r '0x1F450-0x1F470=>0xF005'
  --symbols SYMBOLS     List of characters to copy, belongs to previously declared "--font". Examples:
                        --symbols ,.0123456789
                        --symbols abcdefghigklmnopqrstuvwxyz
  --autohint-off        Disable autohinting for previously declared "--font"
  --autohint-strong     Use more strong autohinting for previously declared "--font" (will break kerning)
  --force-fast-kern-format
                        Always use kern classes instead of pairs (might be larger but faster).
  --no-compress         Disable built-in RLE compression.
  --no-prefilter        Disable bitmap lines filter (XOR), used to improve compression ratio.
  --no-kerning          Drop kerning info to reduce size (not recommended).
  --lv-include <path>   Set alternate "lvgl.h" path (for --format lvgl).
  --full-info           Don't shorten "font_info.json" (include pixels data).
```

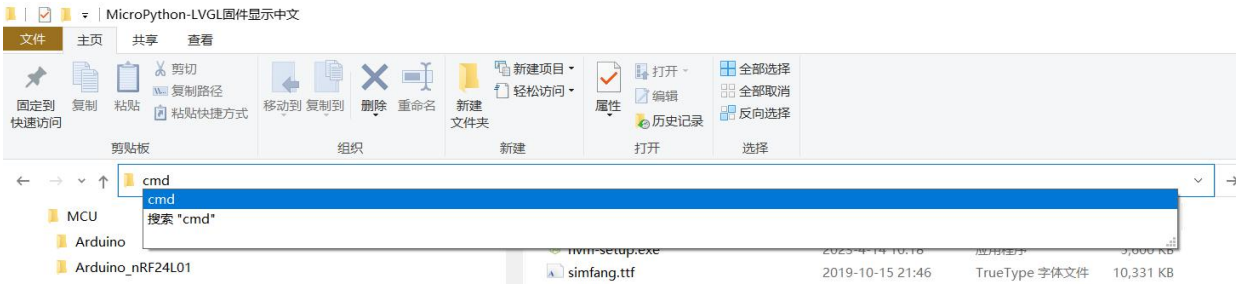
2.2、 制作字体文件

2.2.1、 选择字体

制作 LVGL 字体文件，需要 ttf 字体。这里使用 Windows 自带的仿宋字体文件 simfang.ttf（位置：C:\WINDOWS\Fonts），将该文件拷贝到要生成字体文件的文件夹中。

2.2.2、 生成 LVGL 二进制字体文件

打开字体文件所在的文件夹，在地址栏中输入 `cmd` 后回车，进入命令提示符窗口



在命令提示符窗口输入以下命令：

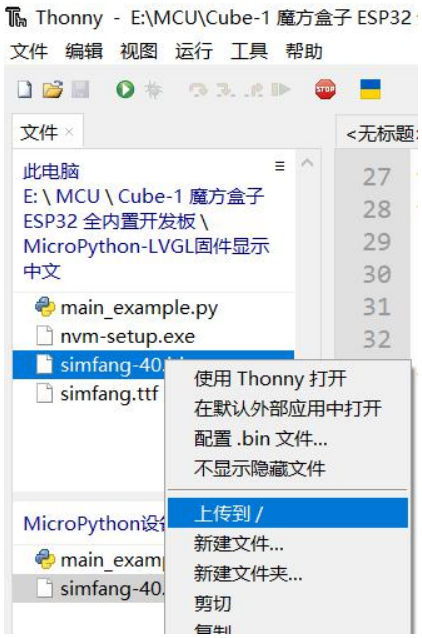
```
lv_font_conv --size 40 --format bin --bpp 1 --font simfang.ttf -r 32-127 --symbols 欢迎你 --no-compress -o simfang-40.bin
```

在字体文件所在的文件夹下，生成了 LVGL 二进制字体文件 simfang-40.bin

2.3、 使用字体文件，显示中文

2.3.1、 上传二进制字体文件

打开 Thonny，将字体文件 simfang-40.bin 上传到设备根目录下



2.3.2、 输入 MicroPython 代码并运行

在 Thonny 代码窗口中输入以下代码：

```
import lvgl as lv
import sys
from machine import I2C,Pin,SPI
from ili9XXX import ili9341
from ft6x36 import ft6x36
import utime as time
import fs_driver
import machine
import utime


# ------------------ 屏幕初始化 --start---------------------------
WIDTH=320
HEIGHT=240

# 创建显示屏对象
disp=ili9341(miso=19,mosi=23,clk=18,cs=14,dc=27,rst=33,power=32,backlight_on=0,power_on=1,
mhz=40, factor=4,
rot=0,hybrid=False,width=WIDTH,height=HEIGHT,start_x=0,start_y=0,invert=True,double_buffer=True,
half_duplex=False,initialize=True)
i2c_bus = I2C(1,sda=Pin(21), scl=Pin(22))

# 创建触控对象
touch=ft6x36()
# ------------------------------ 屏幕初始化操作 --stop-----------------------


# 1. 创建显示 screen 对象。将需要显示的组件添加到这个 screen 对象才能显示
scr = lv.obj()  # scr====> screen 屏幕
fs_drv = lv.fs_drv_t()
fs_driver.fs_register(fs_drv, 'S')
scr = lv.scr_act()
scr.clean()


# 2. 封装需要显示的标签
class InforLbl():
    def __init__(self, scr):
        self.cnt = 0
        lbl = lv.label(scr)  # 将当前标签与 screen 对象进行关联
```

```
        #lbl.set_pos(0, 10)  # 标签定位，相对于屏幕左上角，x 为 0，y 为 10
        #lbl.align(lv.ALIGN.CENTER, -20, 50)  # 标签对齐，居中偏移（x 偏移-20，y 偏移 50）
        lbl.center()  # 相对于父对象居中
        lbl.set_size(320, 40)  # 设置标签的宽度为 320，高度为 40
        self.myfont_cn = lv.font_load("S:./simfang-40.bin")  # 装载字体文件
        lbl.set_style_text_font(self.myfont_cn, 0)  # 设置标签字体
        lbl.set_text("RubikCube1 欢迎你")  # 设置标签文字内容

# 3. 创建标签
inforLbl = InforLbl(scr)

# 4. 显示 screen 对象中的内容
lv.scr_load(scr)


# ------------------------------ 看门狗，用来重启 ESP32 设备
--start-----------------------
try:
    from machine import WDT
    wdt = WDT(timeout = 1000)  # ESP32 上，最小超时时间为 1 秒。1 秒内未喂狗（程序跑飞），
ESP 复位
    print("提示：Ctrl+C 结束")
    while True:
        wdt.feed()
        time.sleep(0.9)
except KeyboardInterrupt as ret:
    print("程序停止运行，ESP32 已经重启...")
# ------------------------------ 看门狗，用来重启 ESP32 设备
--stop------------------------
```

运行代码，设备屏幕上正常显示中文


RubikCube1欢迎你