

思兼的 Fly-Gemini 硬件简明使用手册（探索版 v1.2）

前言

原创文章，转载引用请务必注明链接，水平有限，如有疏漏，欢迎交流指正。

这篇为什么叫探索版呢，其实就是自己使用过程中记录的流水账，是 [发现问题](#) - [查阅探索](#) - [找到答案](#) 的过程，对于大多数人来说，可能看一下结论就可以了。

本文中我们约定，运行 Linux 的部分称为 MPU，运行 Klipper 固件的部分称为 MCU。关于主板方向，以驱动朝上为正。

本手册内容比较繁杂，尽量先简介开机上电运行 Klipper 的整个理想流程，然后再对于具体细节做介绍。

最重要的是请先阅读官方说明书，除了部分内容我觉得可以补充的，其余就不再赘述。

【2021年12月19日更新】

- 修正 adxl345 CS 引脚，优化关机按钮修复脚本路径
- 更正：MCU 部分引脚是 5V tolerant，在手册中 Pinouts and pin description 部分可以看到，FT = 5V tolerant, TT = 3.6V tolerant, [参考](#)。
- 补充：对于 MPU 部分，有说 AW 的 SoC UART 引脚支持 5v，有说仅 Rx 支持 5v，反正不要接 5v 输入脚最好。

内容介绍

- 使用 Gemini UART 接口自烧写 MCU BootLoader
- 介绍 MPU GPIO 资源的使用
- 介绍 MPU 的一些硬件资源
- 优化 Armbian 并修复部分问题

主板的一些情况与问题

1. 全部信号引脚均为 **3.3V 电平**，实际略低，输出电流较小，不适合大负载。
2. MPU 和 MCU 供电是一起的，无法独立使用。注意 MPU 运行 Linux，相当于一台电脑，**不要频繁强制断电**，会损坏系统及硬件。
3. USB-OTG 为以太网接口旁边上方的 USB0 接口，需要拉高 `USB ID` 引脚，并启用 `usbhost0` overlay。
4. 原理图 SPI 端子座引脚顺序（5v-GND-I2C）与主板丝印顺序（5v-IO-GND）冲突，原理图正确。
5. I2C 的数据脚和时钟脚分在两个端子座上，应该是 I2C1，如是则 overlay 启用错误。
6. `Core FAN` 引脚为 左(-)右(+)和有的风扇的线序相反（比如我的）。此外，此引脚输出为最大为 3.3v 不到，带不动 5v 风扇，建议接常开风扇
7. 如果使用 MKS 屏幕，其 EXP1 和 EXP2 开口方向与此主板相反，使用屏幕建议购买各主板的配套设备，否则注意线序进行必要修改。

0、Fly-Gemini 3D打印机控制主板介绍

- 产品链接 | [taobao.com](https://www.taobao.com)

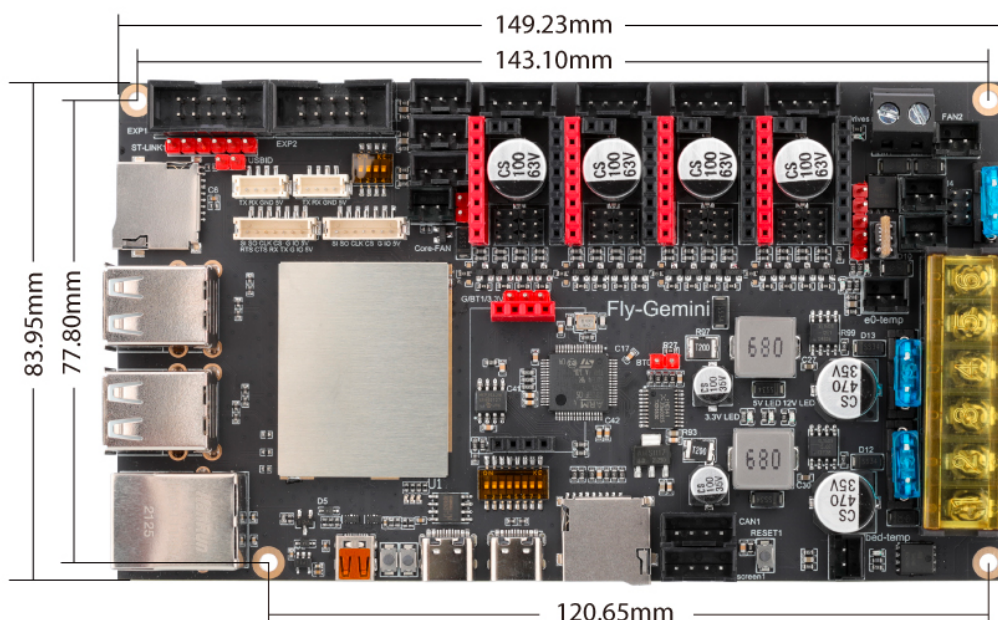
- 资料镜像地址 | github.com, 包括主板原理图与拨码开关使用说明
- 官方 QQ 群: 786561979

Fly-Gemini (以下简称 Gemini) 是 Fly3D (Mellow) 新推出的将 MPU 和 MCU 整合到一起的新 3D 打印机主板, 支持 Klipper 与 RRF 固件, 尤其是对于 Klipper 这种需要上位机进行计算的新固件意义重大, 由于比较新颖, 优点和缺点皆有之。

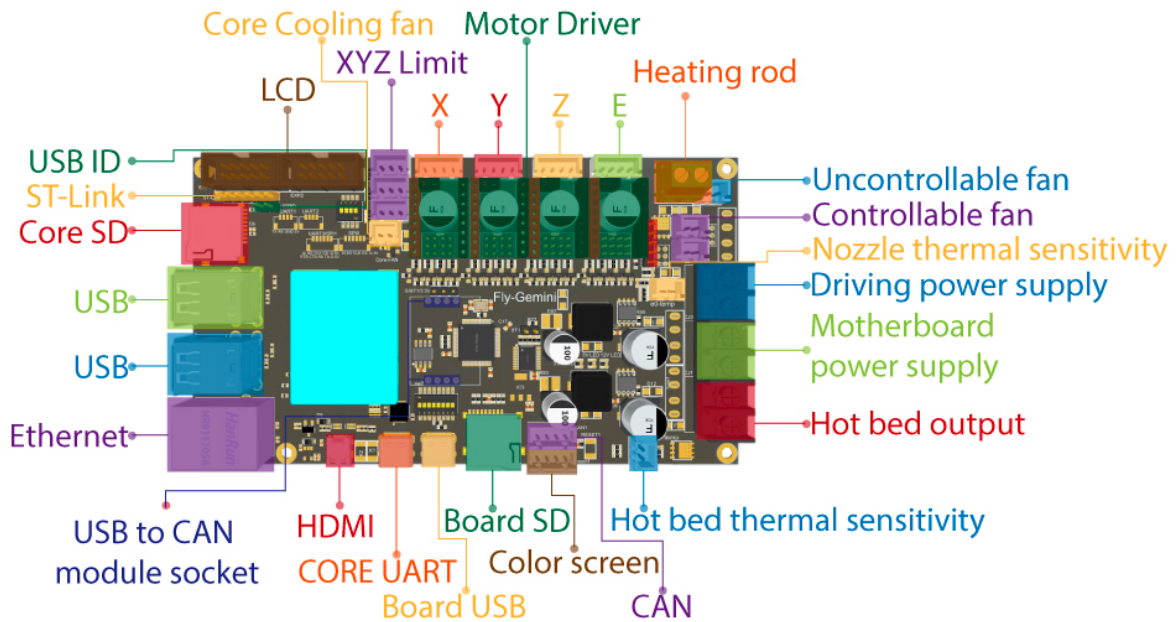
硬件规格:

- MPU: Allwinner H5 4 核 | linux-sunxi.org
- 内存: 512MB
- GPU: Mali450
- USB 2.0 × 3, USB 2.0 OTG × 1
- UART × 3, SPI × 2 (引脚复用)
- microHDMI × 1 (注意区分 miniHDMI)
- 100Mbps Ethernet
- MCU: STM32F405RGT6
- 其他特性:
 - 支持 CAN 模块
 - 支持 ST-Link 调试
 - 支持一路 MPU 可控风扇

DIMENSIONS

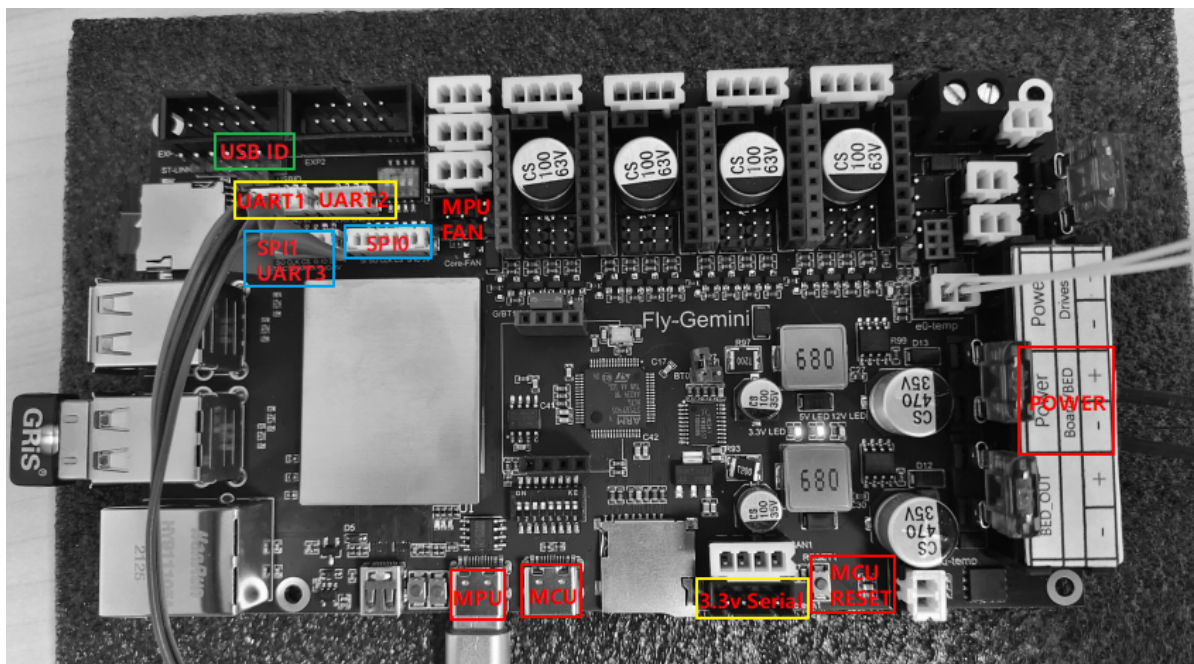


INTERFACE DIAGRAM



1、理想使用流程

为了防止劝退新手，先把基本剧本顺一遍，走一遍流程。



【部分接口分布图】有些是我参考主板原理图和测试出来的。

1.1 供电

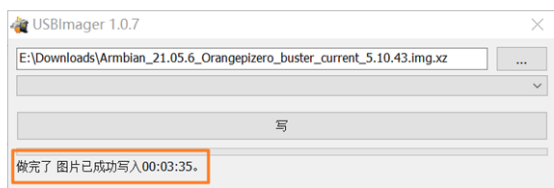
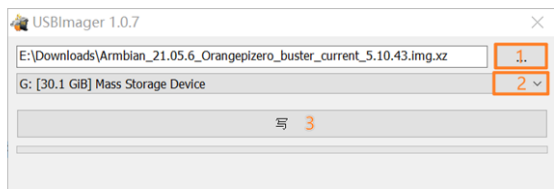
主板 MPU 和 MCU 是同时供电的，无法独立供电，方式包括 MPU Type-C、MCU Type-C、端子 三种方式。和树莓派一样，由于 GPIO 区域的引脚一般不存在保护电路，我不推荐通过引脚反向供电的方式（Fysetc Spider 支持该选项），除非你能保证反向供电的电压电流总是稳定可靠。



1. 主板启动过程最大电流约 0.5A (MPU Type-C 供电, 无线网连接状态)
2. MPU 信号电平为 3.3v

1.2 烧录系统镜像

1. 从官方QQ群或者 GoogleDrive 下载系统镜像, 文件名: `FLY-Gemini_Ambian_21_12_8_server.img.xz`。
2. 下载并打开系统烧录工具: [USBImager \(100+ KB\)](#) 或者 [balenaEtcher \(100+ MB\)](#), 这里以更轻巧的 USBImager 为例。
3. 插入 microSD 卡 (8GB 以上即可), 如果烧录失败可以先使用 `SDFormatter` 格式化



4. 将 SD 卡插入 MPU 的 SD 卡插槽，MPU Type-C 另一头连接到电脑，根据官方文档操作即可。远程管理工具以免费强大的 [MobaXterm](#) 为例。

1.3 Armbian 优化设置与常见问题解决（重要）

粗看了一眼官方系统，有些地方需要优化，同时还有一些常见问题没有解决。

```
1  #!/bin/bash
2  #////////////////////////////////////
3  # Armbian Setting Up for Klipper Script
4  #
5  #////////////////////////////////////
6  # Created by 思兼 / sjqlwy@gmail.com
7
8  # 1. 根据个人习惯，添加名为 pi 的用户
9  # adduser --gecos GECOS --add_extra_groups pi
10 # su pi
11
12 # 2. armbian-config 会访问 github.com 测试网络连接，改为 baidu.com
13 sudo sed -i 's/http://github.com/https://baidu.com/g' /usr/bin/armbian-
    config
14
15 # 3. 加速 Python Package Index(Pypi)，启用清华大学镜像
16 mkdir -p $HOME/.config/pip
17 cat << _EOF_ > $HOME/.config/pip/pip.conf
18 [global]
19 index-url=https://pypi.tuna.tsinghua.edu.cn/simple
20 extra-index-url=https://www.piwheels.org/simple
21 timeout = 600
22 _EOF_
23
24 # 4. 加速 git，使用 fastgit.org 镜像
25 git config --global url."https://hub.fastgit.org/".insteadOf
    "https://github.com/"
26
27 # 5. 加速 APT 源与 Armbian 源（重要）
28 sudo sed -i 's/deb.debian.org/mirrors.ustc.edu.cn/g' /etc/apt/sources.list
29 sudo sed -i 's|security.debian.org|mirrors.ustc.edu.cn/debian-security|g'
    /etc/apt/sources.list
30 sudo sed -i 's|apt.armbian.com|mirrors.ustc.edu.cn/armbian|g'
    /etc/apt/sources.list.d/armbian.list
31
32 # 6. 更新系统软件包
33 sudo apt update && sudo apt upgrade -y
34
35 # 7. 安装必要组件，注意如果使用 Klipper Python3，可以不安装 python-pip
36 sudo apt-get install -y avahi-daemon python-pip python3-pip fonts-wqy-zenhei
37
38 # 8. 添加常用命令别名
39 echo 'alias lid="ls /dev/serial/by-id/*"' >> ~/.bashrc && source ~/.bashrc
40 echo 'alias k=~/.kiauh/kiauh.sh' >> ~/.bashrc && source ~/.bashrc
41
42 # 9. 添加系统环境变量，修复 Fluidt 网页关机/重启功能，修复可能的串口操作权限问题
43 echo 'export PATH="/sbin:/usr/sbin:$PATH"' >> ~/.bashrc && source ~/.bashrc
44 ~/moonraker/scripts/sudo_fix.sh
45 sudo usermod -a -G tty $USER
46 sudo usermod -a -G dialout $USER
```

```

47
48 # 10. 设置系统时区
49 timedatectl set-timezone Asia/Shanghai
50
51 # 11. 优化 Fluidd/MJPEG-Streamer 配置文件下载链接
52 sed -i '0,/FLUIDD_DL_URL/{s/FLUIDD_DL_URL/FLUIDD_DL_URL_ORI/}'
kiauh/scripts/install_klipper_webui.sh
53 sed -i '/FLUIDD_DL_URL_ORI/a\\
FLUIDD_DL_URL=${FLUIDD_DL_URL_ORI}\\github.com\\download.fastgit.org}'
kiauh/scripts/install_klipper_webui.sh
54 sed -i 's/githubusercontent.com/fastgit.org/g' kiauh/scripts/install_mjpg-
streamer.sh
55
56 # 12. 【可选】切换 Klipper Python3
57 sed -i 's|python2|python3|g' kiauh/scripts/install_klipper.sh

```

注：【11】可以参考 [该方法](#)： `FLUIDD_DL_URL=$(curl -s`

`https://api.github.com/repositories/295836951/releases/latest | grep`

`browser_download_url | cut -d'"' -f4 | sed 's/github.com/download.fastgit.org/g')`

优化后效果：

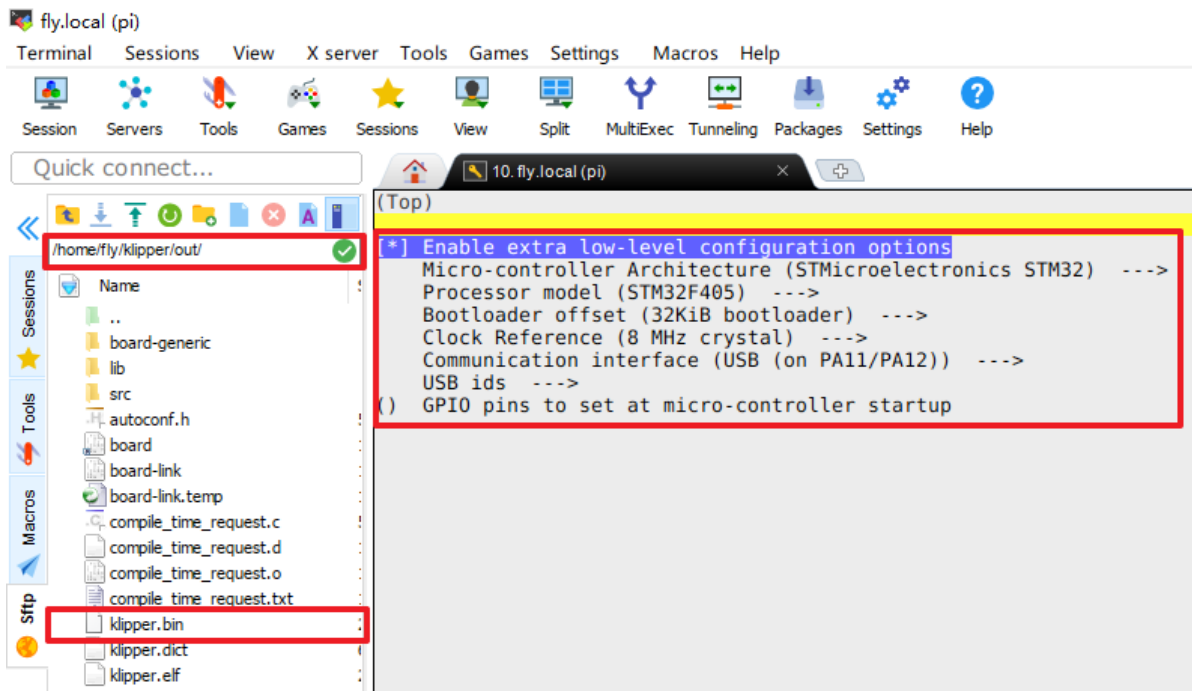
1. 支持 Fly-Gemini.local 访问设备
2. 支持输入 `l`id 命令查看串口设备，`k` 启动 `kiauh`
3. 【重要】修复 Fluidd 网页无法使用右上角命令关机/重启 MPU 的问题
4. 加速 git/pypi/apt 下载加速，减少因网络超时导致安装失败等问题
5. 优化 Fluidd 以及摄像头支持包 MJPG-Streamer 配置文件的链接地址，防止下载超时失败
6. 修复环境变量导致的部分命令无法使用的问题
7. 修复可能的权限导致串口设备访问失败的问题
8. 支持切换到 Klipper Python3 运行环境，解决新版本不支持中文名称 Gode 文件的问题，具体请移步 [此链接](#)。

此外可以切换使用我 [自编译的 Fluidd 开发版本](#)，补全了中文本地化翻译，支持更流畅的摄像头视频流方式。

至此，可以愉快使用 Klipper 了。

1.4 Klipper 配置与启动

参考官方文档，为 MCU 刷入 Klipper 固件，参数设置如下图所示。编译成功后，下载 `klipper.bin` 重命名为 `firmware.bin`（可选）并拷贝到格式化为 **FAT32文件系统** 的 microSD 中，插入 MCU SD卡槽后按下 MCU Reset键 5s 左右，重新把 microSD 卡插入电脑，看到文件重命名为 `fly.cur` 则代表刷入新固件成功。查看说明书，将拨码开关 3/4 调到 ON。



如果想对 Fly-Gemini 进行脱机测试，请至少接入挤出头的温敏电阻。这里附上我写的 Gemini MCU 引脚文件，配合我的 [Klipper-Box](#) 项目使用。

```

1 #####
2 # FLY flyboard FLY-Gemini 引脚别名，用于思兼的 Klipper-Box 项目
3 # - 整合全志H5+512MB MPU, STM32F405RGT6 32bit MCU
4 # - 编译选项 8MHz、32KB BL，默认SD卡刷，firmware.bin -> fly.cur
5 # - 支持 DC12-24v 输入
6 # - 默认支持4轴驱动
7 # - 1路热床
8 # - 1路加热头
9 # - 支持2个温度传感器
10 # - 支持 TMC UART 和 SPI模式，12-48v，支持TMC5160
11 # - 1路可控风扇(并联2路?)，1路不可控风扇，1个 MPU 风扇
12 # - 3个限位开关+1个BLTouch
13 # - EXP1、EXP2 支持 LCD 显示
14 # 资料镜像地址: https://hub.fastgit.org/Mellow-3D/Fly-Gemini
15 # Klipper CAN:
16 # https://github.com/Klipper3d/klipper/blob/master/src/stm32/can.c
17 # MKS 3D TOUCH: https://www.bilibili.com/video/BV1gT4y1g78z
18 #####
19 [mcu] # 【重要】插拔主板前后通过 "ls -l /dev/serial/by-id/*" 命令，确认正确的通讯地址
20 serial: /dev/serial/by-id/usb-klipper_stm32f405xx_29002A000450314335393220-if00
21 restart_method: command
22
23 [temperature_sensor mcu]
24 sensor_type: temperature_mcu
25 sensor_mcu: mcu
26
27 [board_pins]
28 aliases:
29 # Stepper drivers
30 MOTO_EN=PB2, MOTO_STEP=PC13, MOTO_DIR=PC1, MOTO_UART=PB11, # MOTOR_X
31 MOT1_EN=PB6, MOT1_STEP=PC14, MOT1_DIR=PC4, MOT1_UART=PB9, # MOTOR_Y

```

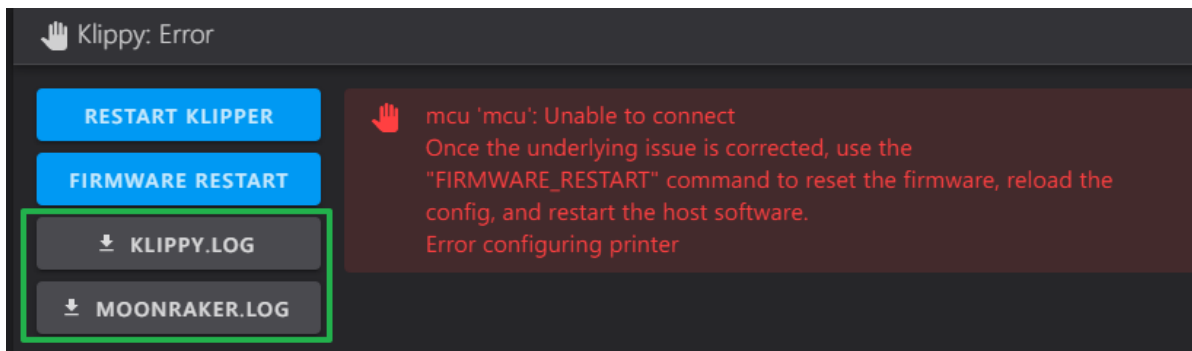


```

32     MOT2_EN=PB5,  MOT2_STEP=PC15,  MOT2_DIR=PC5,  MOT2_UART=PB8,  # MOTOR_Z
33     MOT3_EN=PB4,  MOT3_STEP=PC3,   MOT3_DIR=PC7,  MOT3_UART=PB7,  #
MOTOR_E0
34
35     # Heaters
36     BED_OUT=PA2,
37     HE0=PA0,
38
39     # Thermisors
40     TB=PC2, T0=PC0,
41
42     # Fans
43     FAN0=PC6,
44
45     # End stops
46     ESTOP0=PA3,    # X_Min
47     ESTOP2=PB1,    # Y_Min
48     ESTOP4=PB10,   # Z_Min
49
50     # EXP1
51     EXP1_1=<NC>,    EXP1_2=PA4,
52     EXP1_3=PA13,    EXP1_4=PA10,
53     EXP1_5=PA9,     EXP1_6=PA8,    # ! 注意! 开口在反面
54     EXP1_7=<NC>,    EXP1_8=<NC>,
55     EXP1_9=<GND>,   EXP1_10=<5V>,
56
57     # EXP2, spi2, connect with core spi0 (optional)
58     EXP2_1=PB14,    EXP2_2=PB13,
59     EXP2_3=PA15,    EXP2_4=PB12,
60     EXP2_5=PA14,    EXP2_6=PB15,    # ! 注意! 开口在反面
61     EXP2_7=PB3,     EXP2_8=<RST>,
62     EXP2_9=<GND>,   EXP2_10=<NC>,
63
64     # BL Touch/3D Touch
65     BLT_5=<GND>,
66     BLT_4=<5V>,
67     BLT_CTL=PB0,    # BL Touch servo pin
68     BLT_2=<GND>,
69     BLT_SNSR=PA1,   # BL Touch end stop pin
70
71     # SWD-ST Link
72     SWDIO=PA13,
73     SWCLK=PA14,
74
75     # 3.3v Serial UART, Screen1
76     TX1=PA9,
77     RX1=PA10,
78     BOOT1=PB2,
79
80     # CAN
81     USB_CAN_TX=PA12,
82     USB_CAN_RX=PA11

```

另外附上 Klipper 和 Moonraker 常见问题的解决步骤。



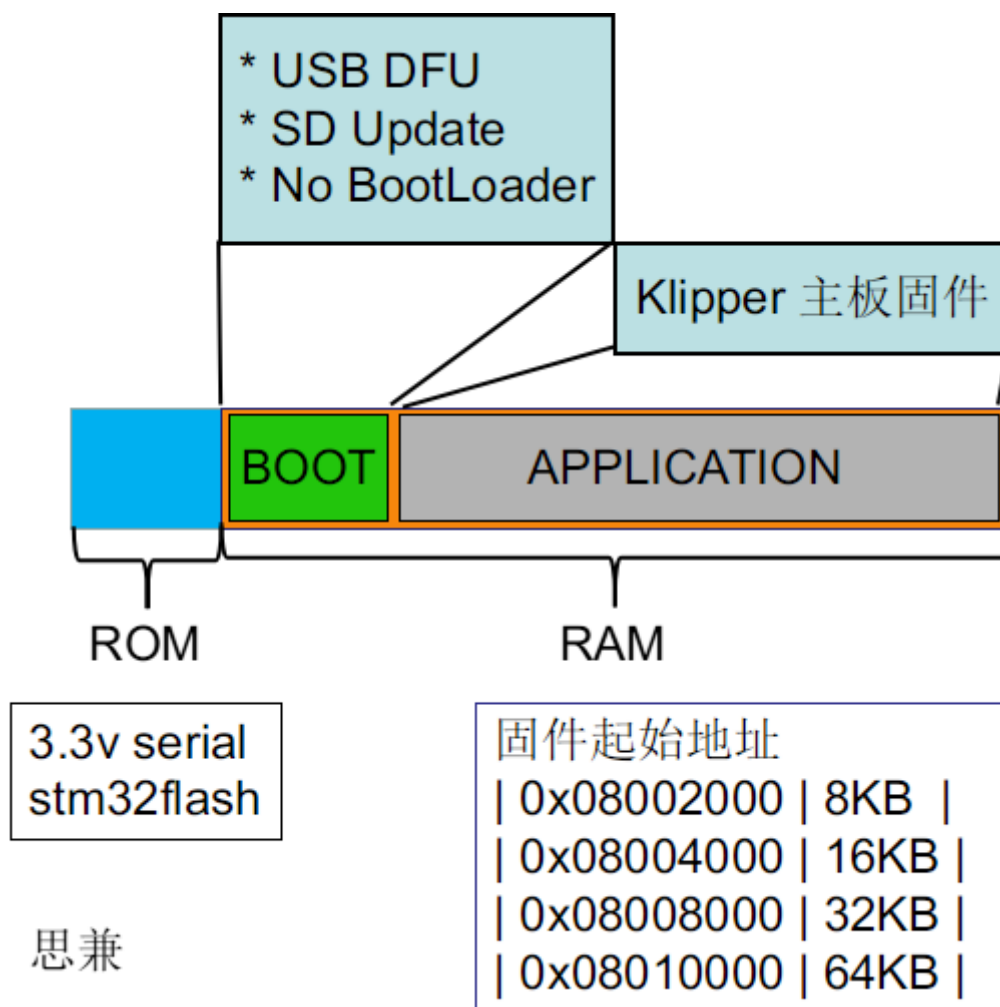
```
1 # 遇到问题请附上 klippy.log 和 moonraker.log 文件，点击上图中的按钮即可下载。
2 # 手动执行以下命令，并附上输出截图
3 systemctl status klipper
4 systemctl status moonraker
5 /home/fly/moonraker-env/bin/python /home/fly/moonraker/moonraker/moonraker.py
  -l /home/fly/klipper_logs/moonraker.log -c
  /home/fly/klipper_config/moonraker.conf
```

2、正文

好，流程走完，我们开始去深入讲讲 Gemini 的硬件。主要分为三部分：MPU、MCU 和两者的通讯手段。

2.1 刷写 MCU BootLoader

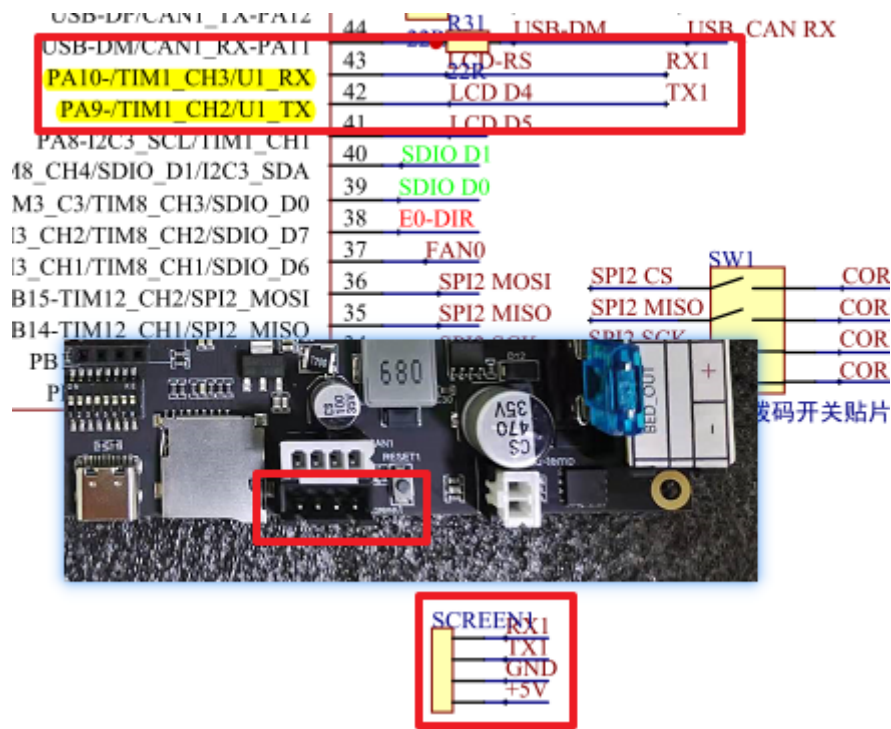
有些 Gemini 的 MCU 出厂没有刷入 Bootloader（以下简称 BL），导致无法使用 SD 卡刷入固件。我们首先尝试进入 STM32F4 的 USB DFU 模式，发现 MPU Linux 系统使用 `lsusb` 和 `dmesg | tail` 命令都无法识别设备。后来发现，不仅没有刷入 Fly BL，也没有 ST 官方 DFU BL。如此想要刷入 BL 或者 Klipper 固件，只能使用 3.3v Serial。具体参考：[加速度计与输入整形器|1](#)、[番外23-UART通讯设置与stm32线刷方法|3](#)、[USB DFU培训](#)。画张图示意如下：



如图所示，STM32的存储包括只读存储器（ROM）和存取存储器（RAM），后者起始地址为 0x08000000，这是基本固定的，可以将 0x08000000 ~ 0x08002000 这 8KB 的空间存放Bootloader 代码。前者可以使用 3.3v serial + stm32flash 工具烧写 BL 和 APP。

0x08002000	8KB
0x08004000	16KB
0x08008000	32KB
0x08010000	64KB

这里我们只能使用 3.3v serial 进行烧录，STM32 默认使用 PA10（Rx）和 PA9（Tx）引脚，查看原理图发现位于主板下方的 Screen1 接口。



2.1.1 使用 USB-TTL 模块刷写 Bootloader

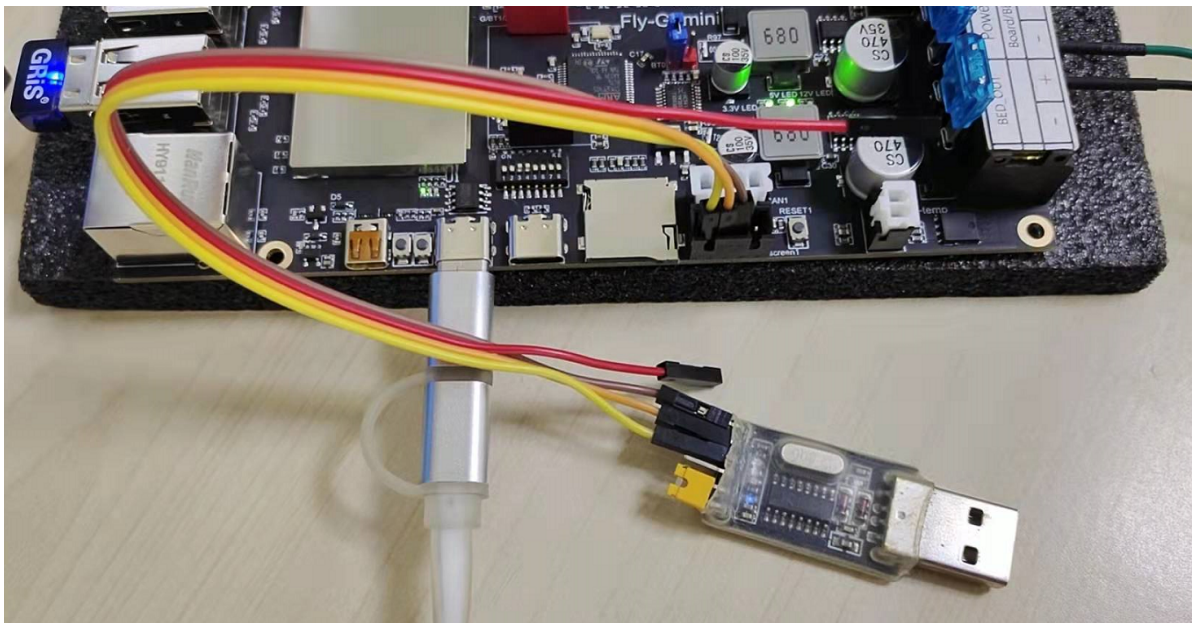
我们先尝试群主推荐的 USB-TTL 模块烧写 BL，注意主板信号电平为 3.3v，且不建议连接 VCC 引脚。

1. 主板上 BOOT0-置高(插上跳帽)，BOOT1-置低(跳帽短接GND)
2. 按住 MCU-RESET 5s左右，重启 MCU 进入 3.3v Serial 烧写状态
3. 从 QQ 群下载 bootloader.bin 放入 stm32flash 文件夹
4. 使用以下命令烧录 BL:

```

1 # 查看设备信息
2 sudo stm32flash /dev/ttyUSB0
3 # 烧录 Fly BL
4 sudo stm32flash -w bootloader.bin -v -g 0 [/dev/ttyUSB0]
5 # .\stm32flash.exe -w bootloader.bin -v -g 0 [COM11]
6 # 也可以使用 3.3v serial 直接烧录 klipper 固件，起始地址 0x08000000，编译固件时选
  择 Bootloader offset (No bootloader)
7 # sudo stm32flash -w ~/klipper/out/klipper.bin -v -g 0 /dev/ttyUSB0
  
```

5. 主板上 BOOT0-置低(拔除跳帽)，BOOT1-置低(跳帽短接GND)，会自动重启 MCU
6. 此时可通过 SD 卡更新 Klipper 固件了



【使用 USB 串口模块刷入 BootLoader】注意 Rx-Tx, Tx-Rx, GND-GND, 信号电平为 3.3v

2.1.2 使用 Gemini UART 自刷写 BootLoader

之前文章说过，可以通过 MPU 的板载 UART 接口与主板通讯，当然也可以用来烧录 BL。H5 有四组 UART，TTL 电平3.3V，其中UART0 可用于调试，UART3 可以复用为SPI 接口。

UART0

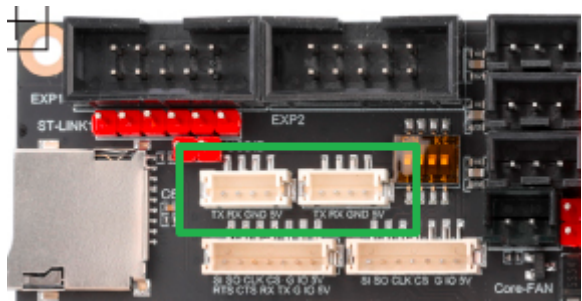
Gemini 板载 CH340N 串口芯片，一端连接 MPU TypeC，一端与 UART0 (ttyS0, CP-Tx, CP-Rx) 连接，用于内核调试 (Kernel Console)。

UART1 - UART2

分别对应 /dev/ttyS1 和 /dev/ttyS2，查看 ARMBIAN 配置文件可知默认启用 `uart1`。

```
1 fly@Fly-Gemini:~$ cat /boot/armbianEnv.txt
2 verbosity=1
3 bootlogo=true
4 console=both
5 disp_mode=1920x1080p60
6 overlay_prefix=sun50i-h5
7 overlays=i2c0 spi-spidev uart1
8 rootdev=UUID=7c4fee6e-3d75-4497-9764-fe4a130f862a
9 rootfstype=ext4
10 param_uart1_rtscts=1
11 param_spidev_spi_bus=0
12 user_overlays=spi
13 usbstoragequirks=0x2537:0x1066:u,0x2537:0x1068:u
14
```

查看原理图，uart1 和 uart2 位于左上角，最后测试发现左侧那个为 uart1，原来我参考原理图猜测右侧那个为 uart1，终究是错付了。如果想启用 uart2，添加相应的 overlay 即可。



UART3 与 SPI1

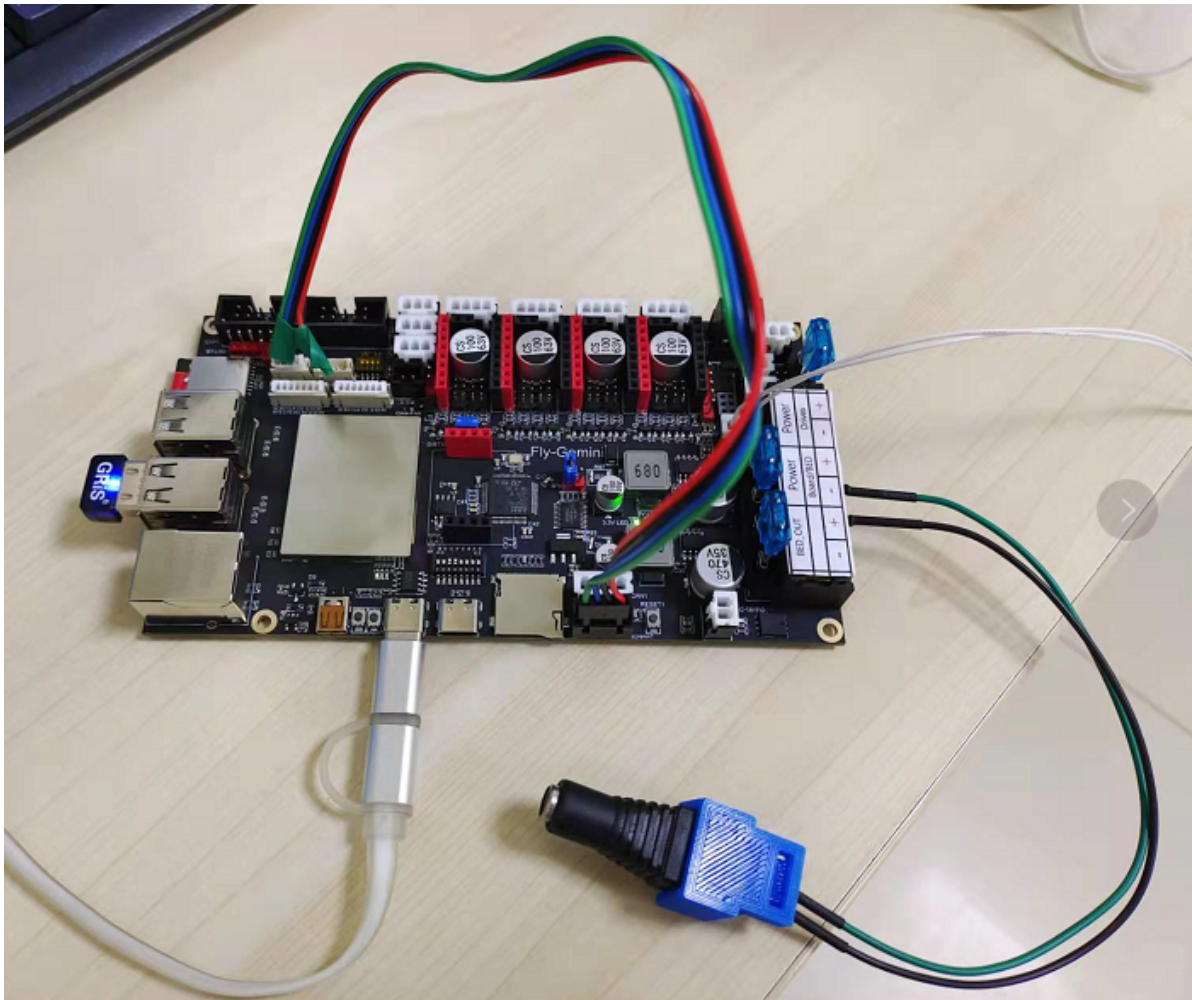
参考上方很奇怪的一点，UART3 和 SPI1 是共用引脚的，SPI1 缺少 CS 和 CLK 引脚。查阅 [H5手册](#) P57 可知 SPI1 和 UART3 引脚复用，通过 overlay 切换。

E15	PA13	Output	1	Function7	O	Z	PU/PD	20	VCC-IO
		SPI1_CS	2		I/O				
		UART3_TX	3		O				
		Reserved	4		NA				
		Reserved	5		NA				
		PA_EINT13	6		I				
		IO Disable	7		OFF				
G12	PA14	Input	0	Function7	I	Z	PU/PD	20	VCC-IO
		Output	1		O				
		SPI1_CLK	2		I/O				
		UART3_RX	3		I				
		Reserved	4		NA				
		Reserved	5		NA				
		PA_EINT14	6		I				
F14	PA15	IO Disable	7	Function7	OFF	Z	PU/PD	20	VCC-IO
		Input	0		I				
		Output	1		O				
		SPI1_MOSI	2		I/O				
		UART3_RTS	3		O				
		Reserved	4		NA				
		Reserved	5		NA				
D15	PA16	PA_EINT15	6	Function7	I	Z	PU/PD	20	VCC-IO
		IO Disable	7		OFF				
		Input	0		I				
		Output	1		O				
		SPI1_MISO	2		I/O				
		UART3_CTS	3		I				
		Reserved	4		NA				
		Reserved	5		NA				
		PA_EINT16	6		I				

6 引脚复用(Pin MUX)
7 SPI1 和 UART3 功能通过
0 overlay切换

使用 UART1 烧录固件

我们制作连接线，我端子用镊子压的，效果不好，凑活使用吧。



如上图所示进行接线，不连接 VCC 引脚，方法同上。当然也可以使用 UART1 与主板 Klipper 固件通讯。

```
1 # 查看设备信息
2 sudo stm32flash /dev/ttyS1
3 # 烧录 Fly BL
4 sudo stm32flash -w bootloader.bin -v -g 0 /dev/ttyS1
```

2.1.3 使用 USB 刷写固件

因为我测试的时候需要频繁刷写固件，使用 SD 卡频繁插拔太麻烦，所以 USB 线刷是更方便的方法，具体参考我之前的文章：[UART通讯设置与stm32线刷方法|3](#)，使用 HID-BootLoader，需要对源码修改后编译。但是该方法会覆盖 Fly BootLoader，使 SD 卡刷方法失效，需要的时候重刷官方 FLY BL 即可。

2.1.4 SD-Updates 伪线刷（不可行）

Klipper 对一些支持 SD 卡刷的主板，提供了伪线刷的方法，需要将 SD 卡插在主板上，然后上位机把编译好的固件通过 SPI 协议传到 SD 卡上，重启主板刷机。但是这种方法在 Gemini 上不可行，原因是其 SD 卡通讯采用的是 SDIO 接口。

2.2 Allwinner H5 MPU overlays

参考文档：<https://docs.armbian.com/User-Guide Allwinner overlays/>

H5 的 overlays 相关信息可以在此文件内查看：`/boot/dtb/allwinner/overlay/README.sun50i-h5-overlays`

- **[analog-codec]** | 音频相关，默认开启，可以关闭

Activates SoC analog codec driver that provides Line Out and Mic In functionality

- **[usbhost0]** | 启用 USB-OTG

Activates USB host controller 0, 所以我们猜测不能用的那个上面的就是 USB0-OTG。也就是我图中插无线网卡的接口。

配套的还有 **USB ID** 引脚，配合 overlay 启用 usb-otg 功能，需要测试验证。

USB_ID pin 为低电平时，则设备为host模式。比如PC和支持OTG设备做主设备时。USB_ID pin 为悬空（高电平）时，则设备为device模式。比如U盘和支持OTG设备做从设备时。

- **[spi-spidev]** | 启用硬件 SPI

可以用于连接加速度计和灯带等 SPI 通讯设备，可以在 `/dev/spidevX.Y` 看到。for userspace SPI access,

where X is the bus number and Y is the CS number。默认启用 **SPI0**，即右侧那个，可以通过拨码开关与 MCU 的 SPI 连接。SPI1 与 UART3 复用引脚。

此外，原理图引脚顺序（5v-GND-I2C）与主板丝印顺序（5v-IO-GND）冲突，是i0还是io？哪个正确？使用万用表测得原理图正确。

附上 **ASXL345** 加速度计设置：

```
1 [mcu host]
2 serial: /tmp/klipper_host_mcu
3
4 # Fly-Gemini, #待验证
5 [adx1345]
6 cs_pin: host:gpiochip1/gpio67
7 spi_bus: spidev0.0
8 axes_map: x,y,z
9
10 [resonance_tester]
11 # 探测点，建议放在热床中心上方，一行一个，注意安全位置
12 probe_points:
13     120,120,20
14 accel_chip: adx1345
15 min_freq: 30
16 max_freq: 100
```

- **[i2c]** | I2C接口，总共有3路

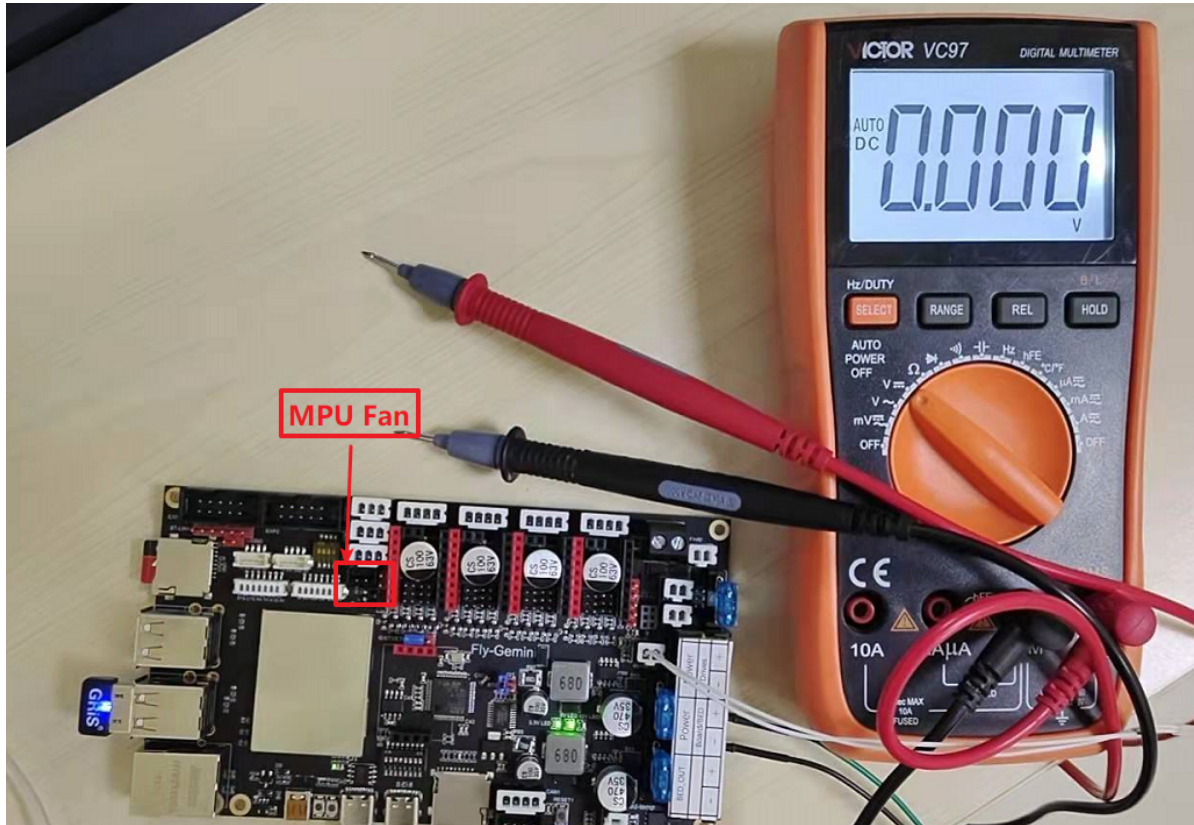
i2c2：多用于连接摄像头 CSI 接口。

原理图上引出 **I2C1**，引脚位置还分开的。。。另外默认启用 **I2C0**，但是我没找到引脚在哪。一个I2C总线只使用两条总线线路，一条双向串行数据线(SDA)，一条串行时钟线(SCL)。数据线即用来表示数据，时钟线用于数据收发同步。

2.3 MPU FAN 与 GPIO 控制

主板引出一路 MPU 风扇接口位于右上方，可以通过短接 **NC FAN** 引脚切换常开（Normal Close）和可控。不短接就是由 **FAN（GPIO3）** 引脚控制。是用了一个 **A03400场效应管**，属于电压控制元件，更多信息可以参考 [用MOS管还是三极管？](#)

这里介绍一下如何控制该引脚，更多内容可以参考 <https://linux-sunxi.org/GPIO>。



2.3.1 使用 sysfs 测试 MPU GPIO

我们查看原理图，发现 FAN 引脚为 GPIO13，需要计算其 GPIO 号。全志系列的计算公式为：

```
1 | (字母顺序 - 1) * 32 + 引脚号
```

以 GPIO13 为例，为 $(17-1) * 32 + 3 = 355$ 。我们验证一下：

```
1 | sudo su
2 | # 启用 GPIO13
3 | echo 355 > /sys/class/gpio/export
4 | # 设置引脚方向为输出
5 | echo "out" > /sys/class/gpio/gpio355/direction
6 | # 设置引脚输出值为1
7 | echo 1 > /sys/class/gpio/gpio355/value
8 | # 释放引脚
9 | echo 355 > /sys/class/gpio/unexport
```

可以使用万用表直流电压档验证输出，高电平为 ~3.3v 输出。说明测试成功。附一张方便记忆的图：
 $PL3 = 352 + 3$ 。


```

1 | #define SUNXI_PA_BASE    0
2 | #define SUNXI_PB_BASE    32
3 | #define SUNXI_PC_BASE    64
4 | #define SUNXI_PD_BASE    96
5 | #define SUNXI_PE_BASE   128
6 | #define SUNXI_PF_BASE   160
7 | #define SUNXI_PG_BASE   192
8 | #define SUNXI_PH_BASE   224
9 | #define SUNXI_PI_BASE   256
10 | #define SUNXI_PJ_BASE   288
11 | #define SUNXI_PK_BASE   320
12 | #define SUNXI_PL_BASE   352
13 | #define SUNXI_PM_BASE   384
14 | #define SUNXI_PN_BASE   416
15 | #define SUNXI_PO_BASE   448
16 | #define AXP_PIN_BASE    1024

```

cat /sys/kernel/debug/pinctrl/*/pinmux-pins

2.3.2 使用 libgpiod 测试 MPU GPIO

新版 Linux 内核更多使用 `libgpiod` 来操作 GPIO，更多信息可参考 [GPIO Programming: Exploring the libgpiod Library](#)。

```

1 | sudo apt-get install gpiod
2 | # 识别 gpiochip 芯片
3 | gpiodetect
4 | # 查看 gpio 引脚信息
5 | gpioinfo
6 | # 输出结果如下
7 | fly@Fly-Gemini:~$ gpiodetect
8 | gpiochip0 [1f02c00.pinctrl] (32 lines)
9 | gpiochip1 [1c20800.pinctrl] (224 lines)
10 | fly@Fly-Gemini:~$ gpioinfo
11 | gpiochip0 - 32 lines:
12 |     line 0:      unnamed      unused   input   active-high
13 |     line 1:      unnamed      unused   input   active-high
14 |     line 2:      unnamed      "power"  input   active-low [used]
15 |     line 3:      unnamed      "moonraker" output  active-high [used]
16 |     line 4:      unnamed      unused   input   active-high
17 |     line 5:      unnamed      "vcc-io" output  active-low [used]
18 |     line 6:      unnamed      unused   input   active-high
19 |     line 7:      unnamed      unused   input   active-high
20 |     ...
21 |
22 | # 读取 gpiochip0/gpio6 引脚值
23 | gpioget 0 6
24 | # 设置 gpiochip0/gpio24 输出高电平1s后恢复低电平
25 | gpioset --mode=time -s 1 0 24=1

```

这里我们想操作 FAN 引脚，试图找到我们计算得到的 355 号引脚，但是并没有，有问题，属实有问题，那它肯定有其他名称，如何获悉呢？

```
1 | sudo cat /sys/kernel/debug/gpio
```

【图】如图所示，我们得知，gpio-355 属于 gpiochip0 的第 3 个，输入以下命令测试成功。参考：[IMX8 GPIO 编号计算和控制](#)

```
1 | gpioset 0 3=1
```

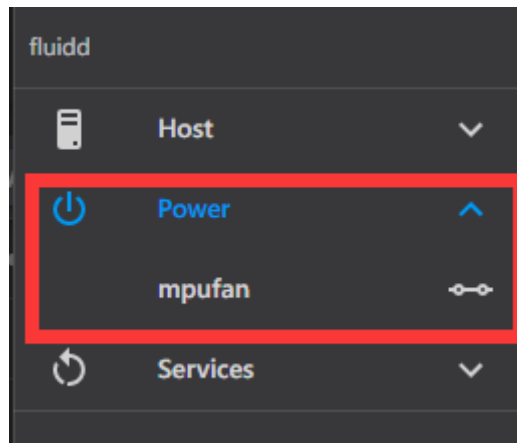
2.3.3 演示 moonraker 控制 MPU FAN

既往我会使用电源管理模块连接到树莓派 GPIO 引脚上，然后使用 Moonraker 控制打印机开/关，这里我们准备演示使用 moonraker 控制 FAN 引脚。

moonraker 的 [文档](#) 关于 power 部分格式是这么要求的：

```
1 | [power light_strip]
2 | type: gpio
3 | pin: gpiochip0/gpio17
4 | initial_state: on
```

于是我先后尝试了 gpiochip0/gpio355、gpiochip0/gpio3、gpiochip0/gpioL3，在 Fluidt 有上角都没有出现那个熟悉的电闸开关。



查看 moonraker.log 日志，报错信息如下：

```
2021-12-14 05:40:04,789 [gpio.py:setup_gpio_out()] - Unable to init 3. Make sure the gpio is not in use by another program or exported by sysfs.
Traceback (most recent call last):
  File "/home/fly/moonraker/moonraker/components/gpio.py", line 47, in setup_gpio_out
    chip = self._get_gpio_chip(chip_id)
  File "/home/fly/moonraker/moonraker/components/gpio.py", line 33, in _get_gpio_chip
    chip = self.gpiod.Chip(chip_name, self.gpiod.Chip.OPEN_BY_NAME)
PermissionError: [Errno 13] Permission denied
2021-12-14 05:40:04,790 [moonraker.py:add_warning()] - Failed to load power device [power mpu_fan]
Error parsing option (pin) from section [power mpu_fan]
2021-12-14 05:40:04,790 [app.py:register_local_handler()] - Registering HTTP Endpoint: (GET) /machine/device_power/devices
2021-12-14 05:40:04,792 [websockets.py:register_api_handler()] - Registering Websocket JSON-RPC methods: machine.device_power.devices
2021-12-14 05:40:04,792 [app.py:register_local_handler()] - Registering HTTP Endpoint: (GET) /machine/device_power/status
2021-12-14 05:40:04,793 [websockets.py:register_api_handler()] - Registering Websocket JSON-RPC methods: machine.device_power.status
2021-12-14 05:40:04,793 [app.py:register_local_handler()] - Registering HTTP Endpoint: (POST) /machine/device_power/on
2021-12-14 05:40:04,795 [websockets.py:register_api_handler()] - Registering Websocket JSON-RPC methods: machine.device_power.on
2021-12-14 05:40:04,795 [app.py:register_local_handler()] - Registering HTTP Endpoint: (POST) /machine/device_power/off
2021-12-14 05:40:04,797 [websockets.py:register_api_handler()] - Registering Websocket JSON-RPC methods: machine.device_power.off
2021-12-14 05:40:04,797 [app.py:register_local_handler()] - Registering HTTP Endpoint: (GET POST) /machine/device_power/device
2021-12-14 05:40:04,798 [websockets.py:register_api_handler()] - Registering Websocket JSON-RPC methods: machine.device_power.get_device, machine.device_power.post_device
2021-12-14 05:40:04,799 [moonraker.py:load_component()] - Component (power) loaded
2021-12-14 05:40:04,799 [moonraker.py:add_warning()] - Unparsed config option 'pin: !gpiochip0/gpio3' detected in section [power mpu_fan]. This may be an option no longer available or could be the result of a module that failed to load. In the future this will result in a startup error.
```

Stage 1、首先搜索关键词 `gpiochip0/gpio3`，提示无法解析。

报错 `Unparsed config option`，那我们看看源码对格式是怎么要求的，另外到底应该填入的 GPIO 号是什么，355、3 亦或是 L3？翻阅 [moonraker 源码](#) `moonraker/moonraker/components/gpio.py`，相关代码如下：

```
1  # Line 46
2      try:
3          chip = self._get_gpio_chip(chip_id)
4          line = chip.get_line(pin_id)
5          args: Dict[str, Any] = {
6              'consumer': "moonraker",
7              'type': self.gpiod.LINE_REQ_DIR_OUT
8          }
9
10 # Line 70
11 def _parse_pin(self, pin_name: str) -> Tuple[int, str, bool]:
12     pin = pin_name
13     invert = False
14     if pin[0] == "!":
15         pin = pin[1:]
16         invert = True
17     chip_id: str = "gpiochip0"
18     pin_parts = pin.split("/")
19     if len(pin_parts) == 2:
20         chip_id, pin = pin_parts
21     elif len(pin_parts) == 1:
22         pin = pin_parts[0]
23     # verify pin
24     if not chip_id.startswith("gpiochip") or \
25         not chip_id[-1].isdigit() or \
26         not pin.startswith("gpio") or \
27         not pin[4:].isdigit():
28         raise self.server.error(
29             f"Invalid Gpio Pin: {pin_name}")
30     pin_id = int(pin[4:])
31     return pin_id, chip_id,
```

- `pin_id = int(pin[4:])` 可知最后只返回 `gpio3` 的数字 3
- `line = chip.get_line(pin_id)` 可知最后读取的是 Line 号而非 GPIO 号

至此正式确认格式为 `gpiochip0/gpio3`，默认为 `gpiochip0` 可省略。我们写个小程序测试一下：

```
1  import gpiod
2  chip=gpiod.Chip('gpiochip0')
3  line = chip.get_line(3)
4  line.request(consumer='moonraker', type=gpiod.LINE_REQ_DIR_OUT)
```

提示 `Permission Denied`（权限不足），使用 `sudo` 执行 python 成功。此时再回头去看 `moonraker.log`，以 `gpio` 为关键词搜索，同样发现权限不足的问题，至此我们找到了新思路。

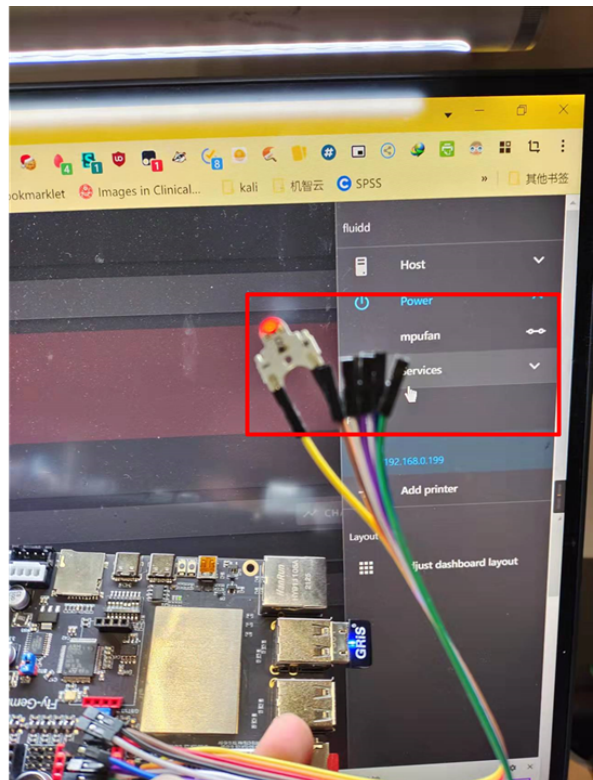
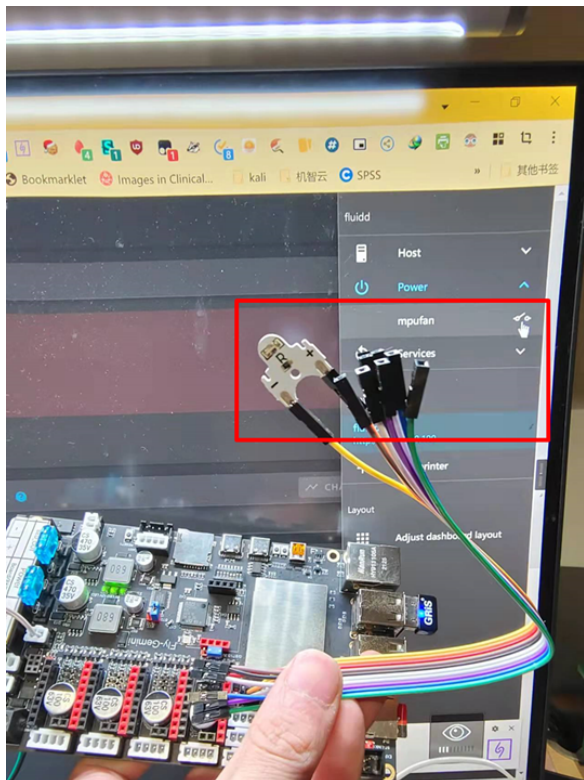
Stage 2、修复权限不足问题

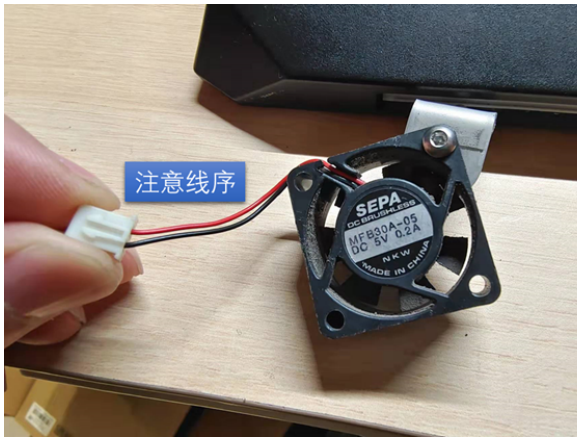
我们查看树莓派 FluidPi_OS 的是如何设置的：

```
1 ls -l /sys/class/gpio/  
2 # 查看用户组  
3 cat /etc/group  
4 # 查看当前用户所在组  
5 id $user  
6 # OR  
7 groups
```

可以看到树莓派中，gpiochip 所在 gpio 组，且 pi 用户在此用户组中。依此修复权限不足的问题

```
1 # 我们不建议把用户添加到root组，所以新建gpio组添加当前用户进去，重新登录（Relogin）生效  
2 sudo su  
3 groupadd gpio  
4 gpasswd -a fly gpio  
5 #####  
6 # 不建议使用usermod，会覆盖辅助用户组  
7 # sudo usermod -a -G gpio $user  
8 # usermod -a -G tty,disk,dialout,sudo,audio,gpio,fly,plugdev,users,systemd-  
journal,input,netdev,ssh,mnkrsudo $user  
9 # chown -R root:gpio /sys/class/gpio 似乎不需要  
10 #####  
11 # 添加 /etc/udev/rules.d/60-gpiod.rules  
12 cat << _EOF_ > /etc/udev/rules.d/60-gpiod.rules  
13 # udev rules for gpio port access through libgpiod  
14 SUBSYSTEM=="gpio", KERNEL=="gpiochip[0-4]", GROUP="gpio", MODE="0660"  
15 _EOF_  
16 # 重启 moonraker 并验证是否成功  
17 systemctl restart moonraker  
18 cat /sys/kernel/debug/gpio
```



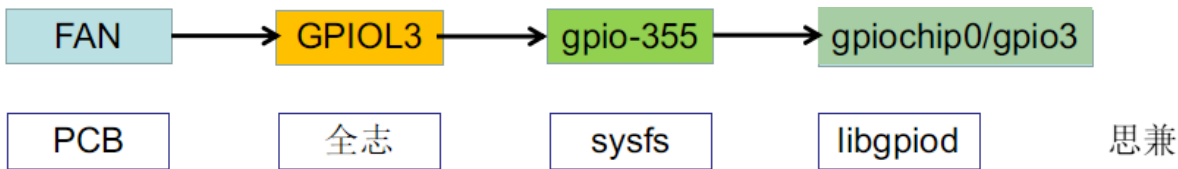


注意:

- MPU FAN 线序与我手里的风扇不一样，左(-)右(+), 如果你的风扇不转的话可以看一下线序。
- be之间等个二极管 会有0.3V压降 1k电阻上再降点 输出电压不高于3.3v, 带不动5v风扇, 只能常开使用。

参考:

- [GPIO not working correctly in Debian Bullseye / Python 3.9.2](#)
- [Power GPIO permission denied](#)



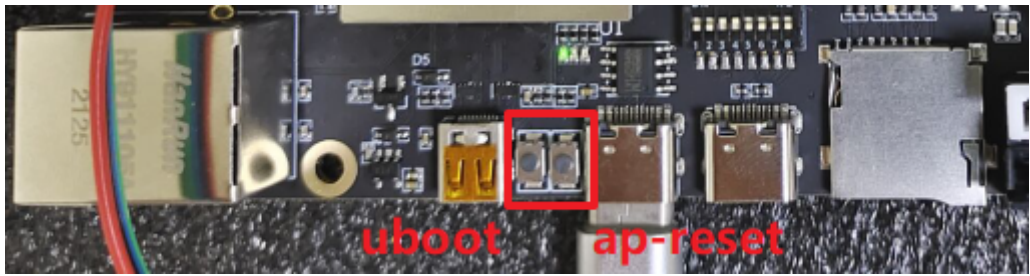
2.3.2 可用 MPU GPIO 汇总

当所有 overlay 关闭时，对应的引脚为 unused 状态，亦即 GPIO（General Purpose Input Output，通用输入输出），对应引脚可用作 GPIO 引脚使用。

功能	引脚			
I2C0	SCL	SDA		
	PA11	PA12		
I2C1	SCL	SDA		
	PA18	PA19		
SPI0	MOSI	MISO	SCK	CS
	PC0	PC1	PC2	PC3
SPI1	MOSI	MISO	SCK	CS
	PA15	PA16	PA14	PA13
UART1	TX	RX	RTS	CTS
	PG6	PG7	PG8	PG9
UART2	TX	RX	RTS	CTS
	PA0	PA1	PA2	PA3
UART3	TX	RX	RTS	CTS
	PA13	PA14	PA15	PA16
备注：SPI1和UART3共用引脚			I2C0我没找到引脚位置	
总计			18	

2.4 MPU 控制信号按钮

MPU 有两个按钮，功能分别是 **UBOOT**（左侧）和 **AP-RESET**（右侧）。其中 AP-RESET 相当于电脑的强制重启，一般情况下不建议使用。



UBOOT	烧录按键	刷机时使用此按键
AP-RESET	系统复位按键	当系统出现死机或者其他异常时，置低AP-RESET 信号大于300ms，系统会强制重启。

2.5 MCU EXP1 和 EXP2

兼容 Ramps 引脚顺序，但是我手中的 MKS 屏幕与 Gemini 主板的开口方向相反，建议大家购买主板配套屏幕。

EXP1 Define

Mini 12864	RAMPS PIN	RAMPS Def.	No.	No.	RAMPS Def.	RAMPS PIN	Mini 12864
VCC	VCC	VCC	1	2	GND	GND	GND
BLUE	D29	LCD_D7	3	4	LCD_D6	D27	GREEN
RED	D25	LCD_D5	5	6	LCD_D4	D23	LCD RST
LCD A0	D16	LCD_RS	7	8	LCD_EN	D17	LCD CS
BTN_ENC	D35	BTN_ENC	9	10	BEEP	D37	BEEP

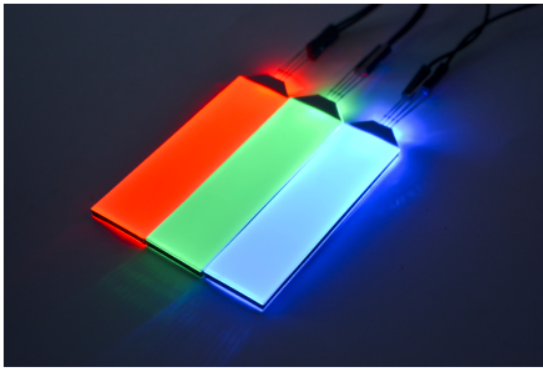
EXP2 Define

Mini 12864	RAMPS PIN	RAMPS Def.	No.	No.	RAMPS Def.	RAMPS PIN	Mini 12864
KILL	D41	KILL	1	2	GND	GND	GND
RST	RST	RST	3	4	CD	D49	CD
MOSI	D51	MOSI	5	6	BTN_EN2	D33	BTN_EN2
SS	D53	SS	7	8	BTN_EN1	D31	BTN_EN1
SCK	D52	SCK	9	10	MISO	D50	MISO

1. 去除KILL信号脚
2. 去除BEEP蜂鸣器脚
3. 去除老式LCD灯光脚

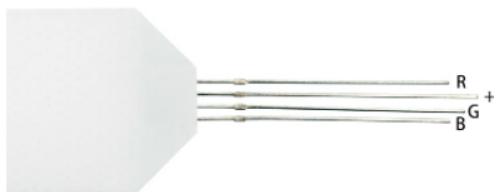
备注:

- 无法直接控制屏幕上的蜂鸣器
- EXP2 连接 MCU 硬件 SPI2
- ramps 中的 **LCD-R/G/B** 三个引脚应该是控制三色通道灯珠用的，目前常见的 miniRGB 屏采用 WS2812 灯带，使用SPI协议通讯，所以不影响使用



RGB LCD多彩发光板采用RGB色彩模式，这是工业界的一种颜色标准，通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色，总共1600万色或，几乎包括了人类视力所能感知的所有颜色。

内置共阳极二极管，以引脚处黑边朝上为正面，引脚从左到右依次为红，电源正极，绿，蓝。它既可以用于背光源，也可用于LED指示灯。



预告

- CAN模块的测试使用
- 为 MPU 添加硬件关机按钮以及屏幕关机命令
- 启用 USB-OTG 虚拟串口功能