

# 必趣全家桶 M8P+CB1+CAN 教程

## 目录

|   |    |
|---|----|
| 1、 系统准备 .....                                 | 2  |
| 1.1 镜像系统烧录 .....                              | 2  |
| 1.2 ssh 连接上位机 .....                           | 4  |
| 2.主板固件烧录（不使用 Can 板） .....                     | 6  |
| 2.1 编译主板 klipper 固件 .....                     | 6  |
| 2.2 刷入 M8P 主板的 klipper 固件 .....               | 7  |
| 2.3 读取 M8P 主板的 UUID .....                     | 7  |
| 3.主板固件烧录（使用 Can 板） .....                      | 8  |
| 3.1 编译主板 katapult 固件 .....                    | 8  |
| 3.2 编译主板的 klipper 固件 .....                    | 8  |
| 3.3 刷入 M8P 主板的 katapult 和 klipper 固件 .....    | 10 |
| 3.4 查询 Can0 通道 .....                          | 11 |
| 3.5 查询主板的 UUID .....                          | 12 |
| 4.刷写 Can 板固件 .....                            | 13 |
| 4.1 刷写 Can 板的 katapult 固件 .....               | 13 |
| 4.2 连接 CAN 板和上位机 .....                        | 14 |
| 4.3 给 Can 板烧录 katapult 固件 .....               | 14 |
| 4.4 给 Can 板编译 klipper 固件 .....                | 15 |
| 4.5 使用 katapult 固件来刷写 Can 板的 klipper 固件 ..... | 16 |
| 4.6 填写主板 uuid 和 can 板 uuid 进打印机配置 .....       | 17 |
| 4.7 FLY 的 SHT36V2 Can 板刷 klipper .....        | 17 |
| 5.0 katapult 使用指南（未完成） .....                  | 18 |
| 5.1 什么是 katapult? .....                       | 18 |
| 5.2 更新 Klipper 固件 .....                       | 18 |
| 6.调机器教程 .....                                 | 19 |
| 6.1 限位检查 .....                                | 19 |
| 6.2 电机检查 .....                                | 19 |
| 6.3 XY 移动方向检查 .....                           | 20 |
| 6.4 Z 轴复位传感器位置定义 .....                        | 21 |
| 6.5 加热器 PID 校准 .....                          | 22 |
| 6.6 probe 传感器精度 .....                         | 22 |
| 6.7 龙门调平（或 Z 倾斜） .....                        | 22 |
| 6.8 挤出机校准 .....                               | 22 |
| 6.9 Z 复位偏移调整 .....                            | 23 |
| 6.10 动态微调 Z 轴高度 .....                         | 23 |
| 6.11 加速度计的使用 .....                            | 24 |

# 1、系统准备

## 1.1 镜像系统烧录

烧录之前准备一张 8G 以上内存卡（推荐金士顿 32G）和读卡器。

进入 BIGTREETECH 的 GitHub 下载最新的 CB1 的系统镜像，推荐直接下载完整版。

地址：<https://github.com/bigtreetech/CB1/releases>

|  |          |        |    |    |    |
|--|----------|--------|----|----|----|
| ▼ Assets 6   |          |        |    |    |    |
| CB1_Debian11_Klipper_kernel5.16_202300712.img.sha256 | 65 Bytes | Jul 14 | 美国 | 美国 | 美国 |
| CB1_Debian11_Klipper_kernel5.16_202300712.img.xz     | 1.24 GB  | Jul 14 | 美国 | 美国 | 美国 |
| CB1_Debian11_minimal_kernel5.16_20230712.img.sha256  | 65 Bytes | Jul 14 | 美国 | 美国 | 美国 |
| CB1_Debian11_minimal_kernel5.16_20230712.img.xz      | 327 MB   | Jul 14 | 美国 | 美国 | 美国 |
| Source code (zip)                                    |          | May 4  | 美国 | 美国 | 美国 |
| Source code (tar.gz)                                 |          | May 4  | 美国 | 美国 | 美国 |

如果下载太慢就右键复制链接地址，再进入下面任意一个加速网站进行下载：

<https://ghproxy.com/>

<https://github.ur1.fun/>

下载完成后解压到自己找得到的地方

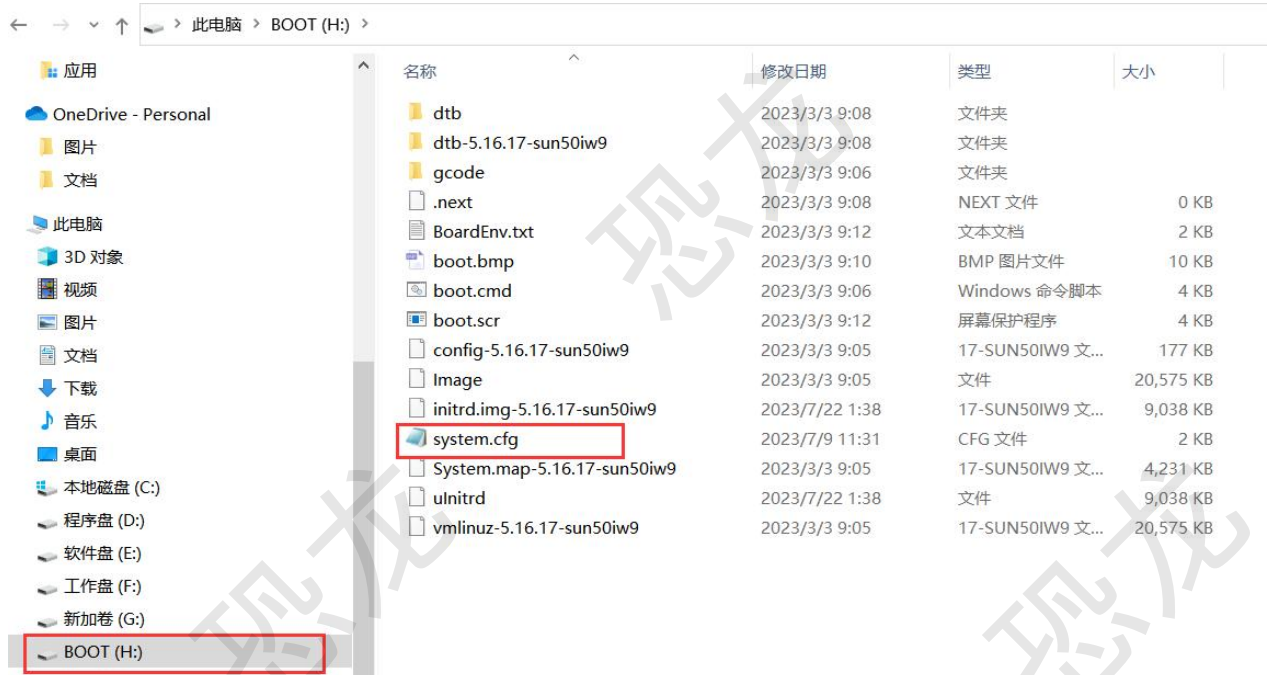
接着使用树莓派烧录软件进行镜像烧录

树莓派烧录器下载地址：<https://www.raspberrypi.com/software/>

烧录之前可以用烧录器自带的擦除功能格式化一下内存卡，格式化完毕从新拔插一次内存卡。



烧录完毕后重新拔插一次内存卡，用记事本（.txt）格式打开 boot 盘里面的 system.cfg 文件

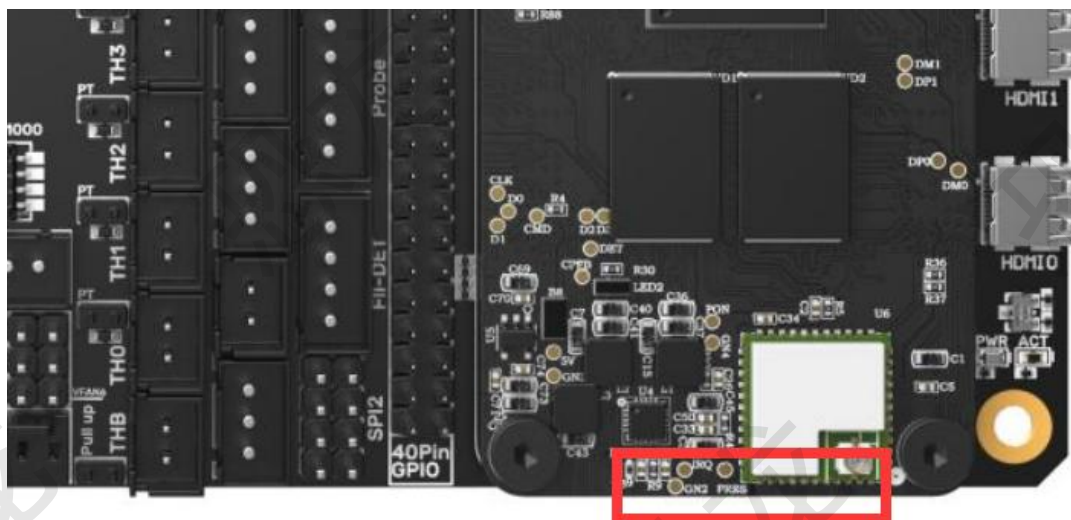


找到 wifi 相关的内容，修改引号内的内容为自己的 wifi 账号和密码，注意不要删除引号！

```
#####  
# wifi name  
WIFI_SSID="恐龙"  
# wifi password  
WIFI_PASSWD="konglong"
```

改完以后一定要保存再关闭！

拔下内存卡插到 M8P 的 SOC Card 插口，即 CB1 天线模块下方的那个。不要插错，M8P 有两个内存卡插口。  
另外，CB1 的天线需要插上，不插上搜不到 wifi！有条件的可以换个好点的 2.4G 天线，附带的天线有点拉跨。



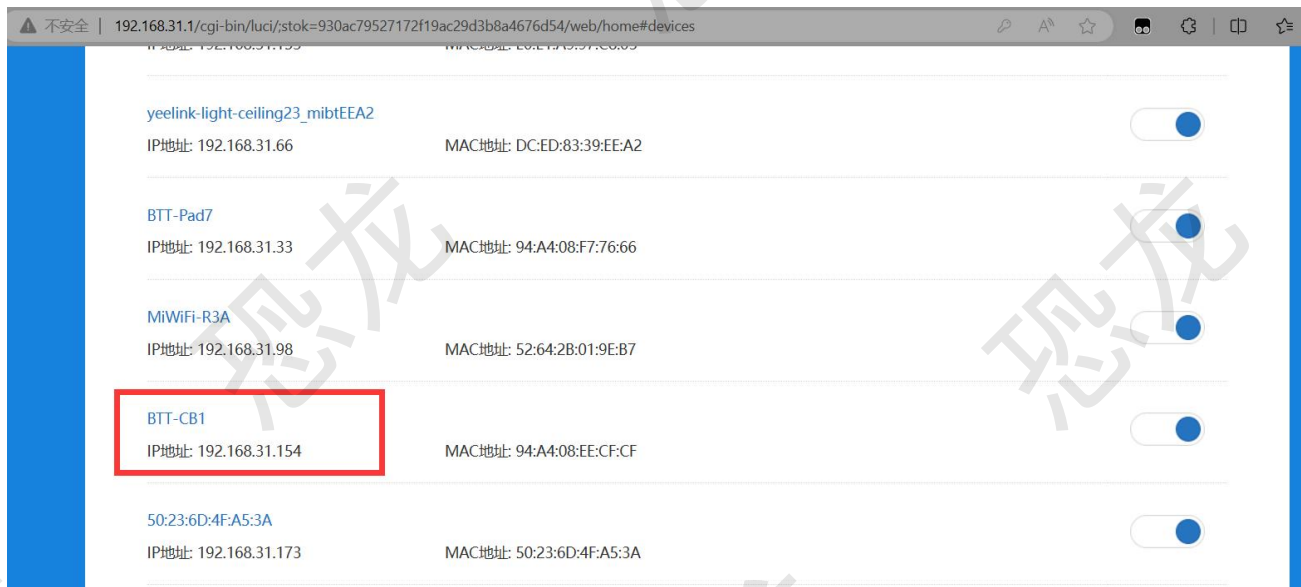
## 1.2 ssh 连接上位机

主板通电后使用 ssh 软件连接上位机（CB1）

安装 ssh 软件 MobaXterm: <https://mobaxterm.mobatek.net/download-home-edition.html>

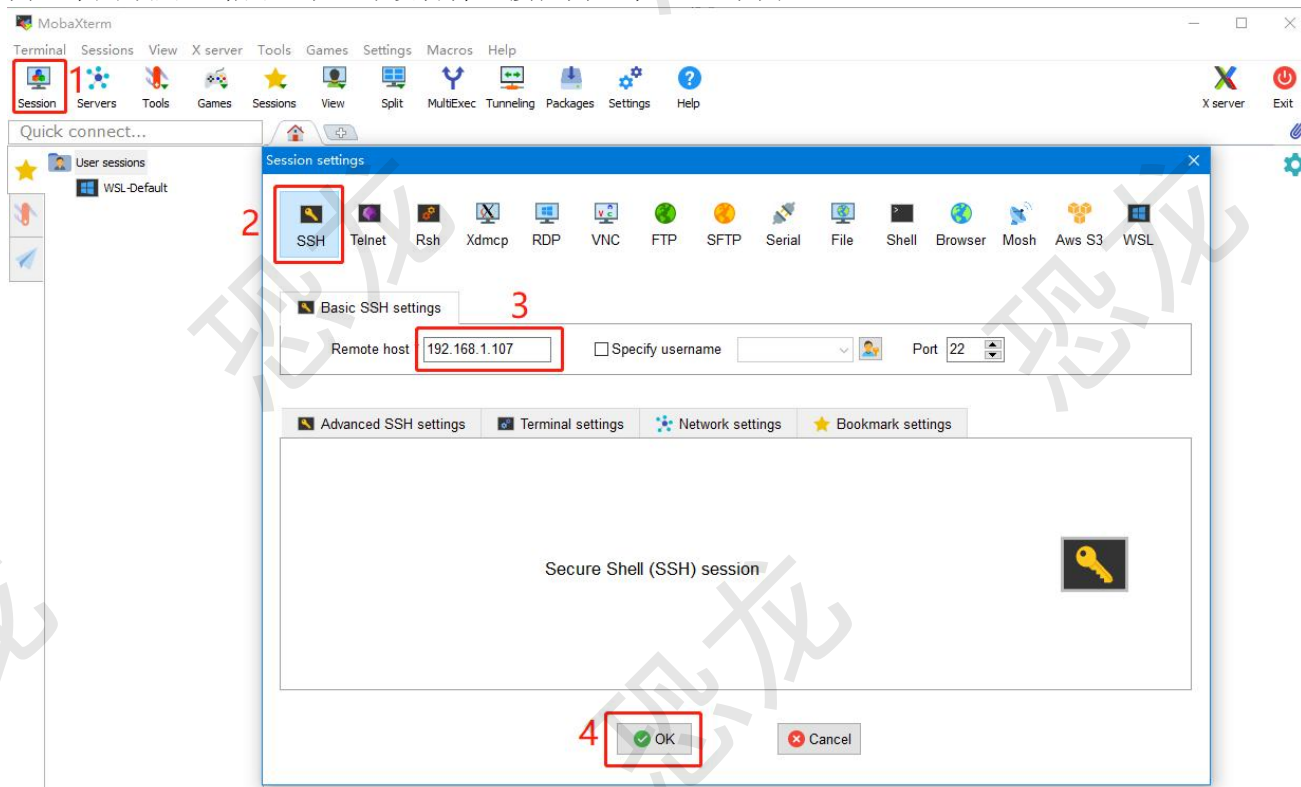
将 MicroSD 卡插到 MANTA M8P，通电后等待系统第一次启动，大概需要 2-5 分钟。

设备连上 WIFI 或者插上网线后，会被自动分配一个 IP，此时进入路由器管理界面找到设备分配的 IP



推荐使用路由器的固定 IP 功能固定一个 IP，防止下次 IP 变动导致进不了系统。

打开已经安装的 MobaXterm 软件，点击“Session”，在弹出的窗口中点击“SSH”，在 Remote host 一栏中输入设备的 IP 地址，点击“OK”（注意：电脑和设备必须要在同一个局域网（路由）下，即设备都连接在同一个 wifi 下面。）

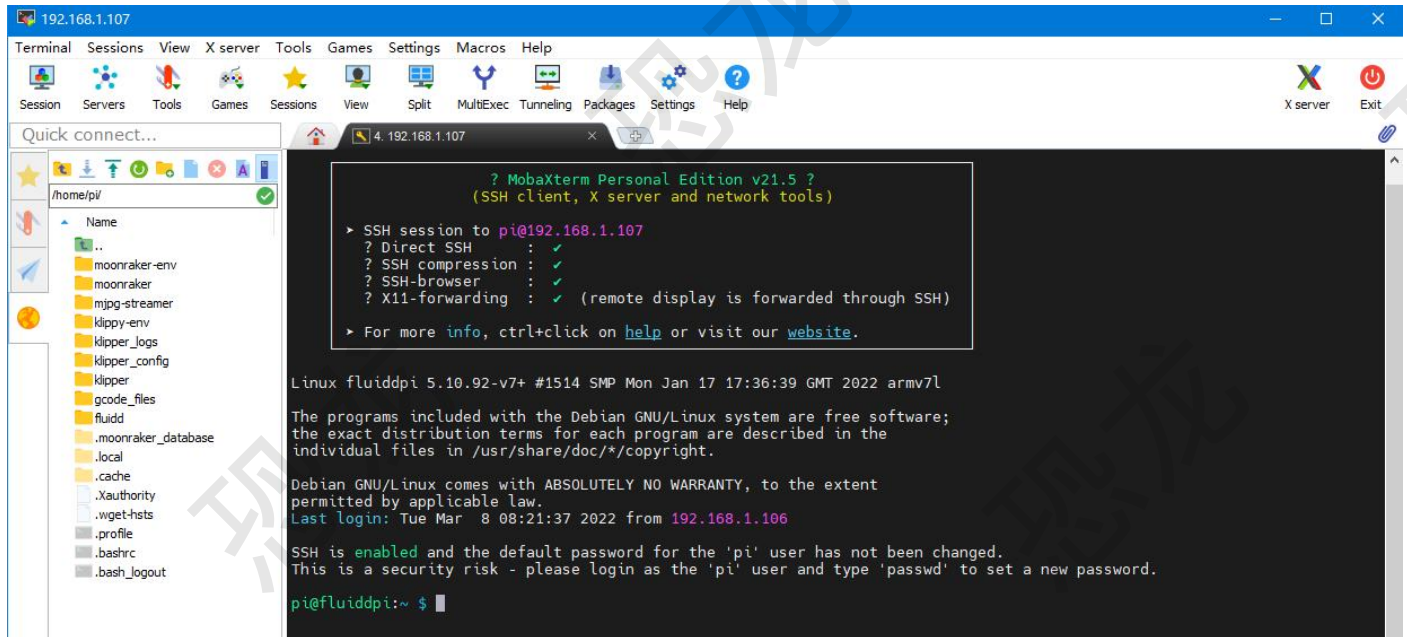


小恐龙交流群：786115036

输入登录名和登录密码进入 SSH 终端界面

登录名：biqu 密码：biqu（linux 系统输入密码不会显示内容，输完回车即可）

进入如下页面





## 2.主板固件烧录（不使用 Can 板）

### 2.1 编译主板 klipper 固件

进入 klipper 目录，输入：

```
cd ~/klipper
```

配置主板的 klipper 固件，输入：

```
make menuconfig
```

新手如果配置页面看到没东西，如下图，则在第一行的地方回车一下，就有更多内容可选了。

```
(Top)
Klipper Firmware Configuration
[ ] Enable extra low-level configuration options
Micro-controller Architecture (Atmega AVR) --->
Processor model (atmega2560) --->
```

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
Micro-controller Architecture (Atmega AVR) --->
Processor model (atmega2560) --->
Processor speed (16Mhz) --->
Communication interface (UART0) --->
(250000) Baud rate for serial port
() GPIO pins to set at micro-controller startup (NEW)
```

**M8P1.1** 配置内容如下图：

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32G0B1) --->
Bootloader offset (8KiB bootloader) --->
Clock Reference (8 MHz crystal) --->
Communication interface (USB (on PA11/PA12)) --->
USB ids --->
() GPIO pins to set at micro-controller startup
```

配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

**M8P2.0** 配置内容如下图：

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32H723) --->
Bootloader offset (128KiB bootloader (SKR SE BX v2.0)) --->
Clock Reference (25 MHz crystal) --->
Communication interface (USB (on PA11/PA12)) --->
USB ids --->
() GPIO pins to set at micro-controller startup
```

配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

## 2.2 刷入 M8P 主板的 klipper 固件

首先主板进入 DFU 模式：

**M8P1.1 主板**就按住主板上的 BOOT 键不松，再按下 RESET 键，然后同时松开。

**M8P2.0 主板**就按住主板上的 BOOT0 键不松，再按下 RESET 键，然后同时松开。

按完后在再 ssh 输入 `lsusb`，如果出现如下图的信息则主板成功进入 DFU：

```
biqu@BTT-CB1:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 004: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~/klipper$
```

确认主板进入 dfu 模式后，进入 klipper 文件夹：

```
cd ~/klipper
```

接着输入下面命令输入 klipper 固件：

```
make flash FLASH_DEVICE=0483:df11
```

完成后按一下主板的 RESET 按键重启主板

## 2.3 读取 M8P 主板的 UUID

输入 `ls /dev/serial/by-id/` 查询主板 uuid

```
pi@fluidapi:~/klipper $ ls /dev/serial/by-id/
usb-Klipper_stm32h723xx_41003D001751303232383230-if00
pi@fluidapi:~/klipper $
```

复制读取到的 uuid 并填写到 `printer.cfg` 里面即可完成上位机和主板的通讯

```
#####
#                               主板配置
#####
[mcu]      # 主板序列号
serial: /dev/serial/by-id/usb-Klipper_Klipper_firmware_12345-if00
```

```
#####
#                               主板配置
#####
[mcu]      # 主板序列号
serial: /dev/serial/by-id/Klipper_stm32h712xx_41003D001751303232383230-if00
```

## 3.主板固件烧录（使用 Can 板）

### 3.1 编译主板 katapult 固件即原 CanBoot 固件

首先下载 katapult（原 CanBoot），ssh 输入：

```
git clone https://github.com/Arksine/katapult
```

进入 katapult 目录，输入：

```
cd katapult
```

配置主板的 katapult 固件，输入：

```
make menuconfig
```

#### M8P1.1 配置内容如下图

注意：CAN bus speed 选项目前建议都设置为 1000000

```
(Top)
Katapult Configuration v0.0.1-57-gabd1545
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32G0B1) --->
Build Katapult deployment application (Do not build) --->
Clock Reference (8 MHz crystal) --->
Communication interface (CAN bus (on PD12/PD13)) --->
Application start offset (8KiB offset) --->
(1000000) CAN bus speed
() GPIO pins to set on bootloader entry
[*] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[ ] Enable Status LED
```

配置好以后按 Q 再按 Y 保存离开，然后输入 make 编译固件，等待固件编译完成即可。

#### M8P2.0 配置内容如下图

```
(Top)
Katapult Configuration v0.0.1-57-gabd1545
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32H723) --->
Build Katapult deployment application (128KiB bootloader (SKR SE BX v2.0))
Clock Reference (25 MHz crystal) --->
Communication interface (USB (on PA11/PA12)) --->
Application start offset (128KiB offset) --->
USB ids --->
() GPIO pins to set on bootloader entry
[*] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[ ] Enable Status LED
```

配置好以后按 Q 再按 Y 保存离开，然后输入 make 编译固件，等待固件编译完成即可。

### 3.2 编译主板的 klipper 固件

进入 klipper 目录，输入：

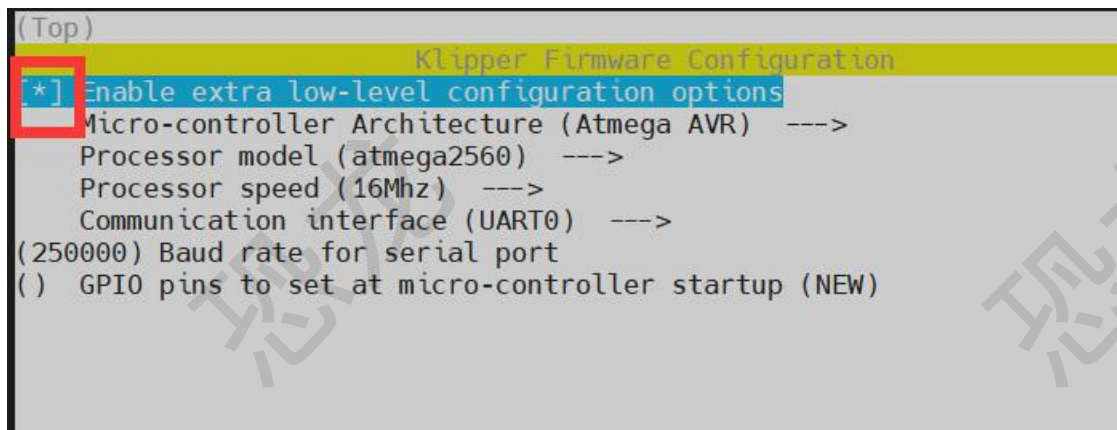
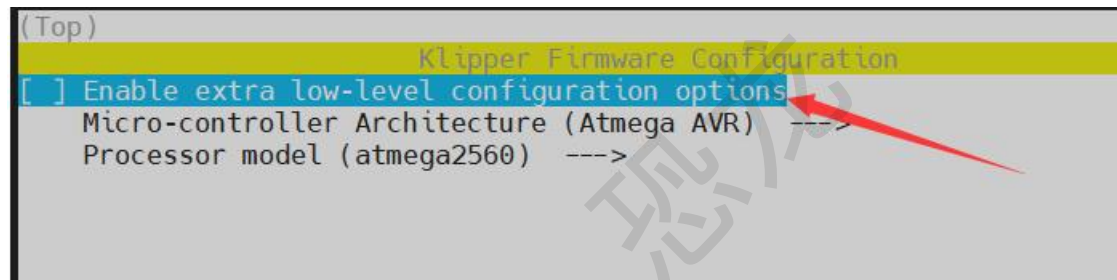
```
cd ~/klipper
```

配置主板的 klipper 固件，输入：

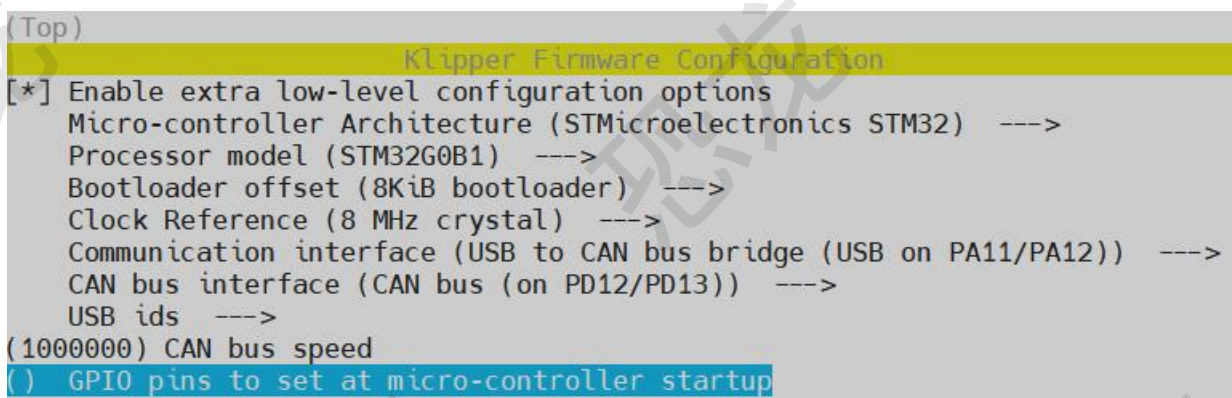
```
make menuconfig
```



新手如果配置页面看到没东西，在第一行的地方回车一下，勾选打星号，就有更多内容可选了。

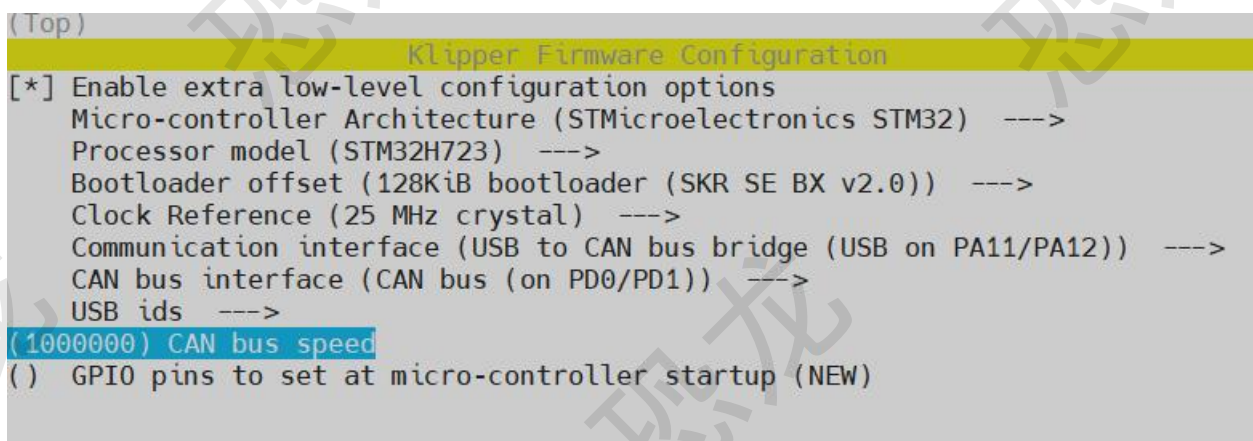


**M8P1.1 配置内容如下图：**



配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

**M8P2.0 配置内容如下图：**



配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

### 3.3 刷入 M8P 主板的 katapult 和 klipper 固件

首先主板进入 DFU 模式: **M8P1.1 主板**就按住主板上的 boot 键不松, 再按下 restart 键, 然后同时松开。

**M8P2.0 主板**就按住主板上的 boot0 键不松, 再按下 restart 键, 然后同时松开。

按完后再 ssh 输入 `lsusb`, 如果出现如下图的信息则主板成功进入 DFU:

```
biqu@BTT-CB1:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 004: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~/klipper$
```

确认主板进入 dfu 模式后:

#### M8P1.1 执行下面命令

首先输入下面命令刷写 katapult 固件:

```
sudo dfu-util -a 0 -D ~/katapult/out/katapult.bin --dfuse-address 0x08000000:force:mass-erase -d 0483:df11
```

上面命令刷写完成后, 再输入下面命令刷写 klipper 固件:

```
sudo dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08002000:force:leave -D ~/klipper/out/klipper.bin
```

#### M8P2.0 执行下面命令

首先输入下面命令刷写 katapult 固件:

```
sudo dfu-util -a 0 -D ~/katapult/out/katapult.bin --dfuse-address 0x08000000:force:mass-erase -d 0483:df11
```

刷入成功后, 按一下主板上的 **RESET** 按键重启主板

接着输入 `ls /dev/serial/by-id/`即可查询到主板的 UUID, 如下图所示:

```
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 1024
DfuSe interface name: "Internal Flash"
Performing mass erase, this can take a moment
Downloading to address = 0x08000000, size = 5668
Download [=====] 100% 5668 bytes
Download done.
File downloaded successfully
biqu@BTT-CB1:~/katapult$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 006: ID 1d50:6177 OpenMoko, Inc. stm32h723xx
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~/katapult$ ls /dev/serial/by-id/
usb-katapult_stm32h723xx_0A002A001851313434373135-1f00
```



小恐龙交流群：786115036

在烧录之前，对于 USB/UART 设备，请确保上位机安装了 pyserial 。此步骤只需要执行一次：

```
pip3 install pyserial
```

接着进入 katapult 的 scripts 目录：

```
cd ~/katapult/scripts
```

输入下面命令用 katapult 来刷入 klipper 固件，注意，红色部分替换成自己刚刚查询到的 uuid，

```
python3 flashtool.py -d /dev/serial/by-id/usb-katapult_stm32h723xx_0A002A001851313434373135-if00 -b 1000000
```

出现如下图所示即刷写成功！

```
biqu@BTT-CB1:~/katapult$ cd ~/katapult/scripts
biqu@BTT-CB1:~/katapult/scripts$ python3 flashtool.py -d /dev/serial/by-id/usb-katapult_stm32h723xx_0A002A001851313434373135-if00 -b 1000000
Attempting to connect to bootloader
Katapult Connected
Protocol Version: 1.0.0
Block Size: 64 bytes
Application Start: 0x8020000
MCU type: stm32h723xx
Flashing '/home/biqu/klipper/out/klipper.bin' ...

[#####]

Write complete: 1 pages
Verifying (block count = 492) ...

[#####]

Verification Complete: SHA = 5AD2DC9F0E213F554B9EBE3D21F535EB408D6473
Flash Success
biqu@BTT-CB1:~/katapult/scripts$
```

### 3.4 查询 Can0 通道

至此 M8P 的固件已经刷写完毕，但还需要编辑上位机的 Can0 通道。如果是 CB1、BTT Pi 则不需要下面的操作，系统已经写在里面了，但最好还是确定一下比特率是否一致。

输入：`sudo nano /etc/network/interfaces.d/can0`

复制并黏贴下面的四行命令，记得修改和固件一致的比特率。

```
allow-hotplug can0
```

```
iface can0 can static
```

```
    bitrate 1000000
```

```
up ifconfig $IFACE txqueuelen 1024
```

```
GNU nano 5.4 /etc/network/interfaces.d/can0
allow-hotplug can0
iface can0 can static
    bitrate 1000000
up ifconfig $IFACE txqueuelen 1024
```

确认无误后，Ctrl+X，按 Y，然后回车储存离开此页面。

重启上位机：`sudo reboot`

小恐龙交流群: 786115036

上位机启动后输入 `ifconfig` 查看 Can0 通道是否打开, 出现如下图的内容则 Can0 通道已经打开。

```
biqu@BTT-CB1:~$ ifconfig
can0: flags=193<UP,RUNNING,NOARP> mtu 16
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 1024 (UNSPEC)
    RX packets 1 bytes 8 (8.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1 bytes 1 (1.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 7819 bytes 4380371 (4.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7819 bytes 4380371 (4.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

### 3.5 查询主板的 UUID

接下来就可以查询到主板的 can bus UUID 了。输入以下指令:

```
~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0
```

```
biqu@BTT-CB1:~$ ~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0
Found canbus_uuid=20a342fca012, Application: Klipper
Total 1 uuids found
biqu@BTT-CB1:~$
```

复制一下你的 UUID, 后面需要在配置在打印机的里 `printer.cfg` 文件里面。



## 4.刷写 Can 板固件

### 4.1 刷写 Can 板的 katapult 固件（原 CanBoot 固件）

进入 katapult 目录，输入：

```
cd katapult
```

配置主板的 katapult 固件，输入：

```
make menuconfig
```

使用 **EBB SB2209/2240** 如下图配置

```
(top)
Katapult Configuration v0.0.1-33-g88e208a
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32G0B1) --->
Build CanBoot deployment application (Do not build) --->
Clock Reference (8 MHz crystal) --->
Communication interface (CAN bus (on PB0/PB1)) --->
Application start offset (8KiB offset) --->
(1000000) CAN bus speed
() GPIO pins to set on bootloader entry
[*] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[*] Enable Status LED
(PA13) Status LED GPIO Pin
```

配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

使用 **EBB SB RP2040** 如下图配置

```
(top)
Katapult Configuration v0.0.1-57-gabd1545
Micro-controller Architecture (Raspberry Pi RP2040) --->
Flash chip (W25Q080 with CLKDIV 2) --->
Build Katapult deployment application (16KiB bootloader) --->
Communication interface (CAN bus) --->
(4) CAN RX gpio number
(5) CAN TX gpio number
(1000000) CAN bus speed
() GPIO pins to set on bootloader entry
[*] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[*] Enable Status LED
(gpio26) Status LED GPIO Pin
```

配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

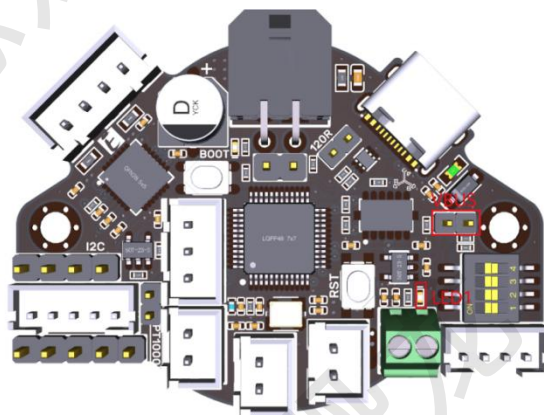
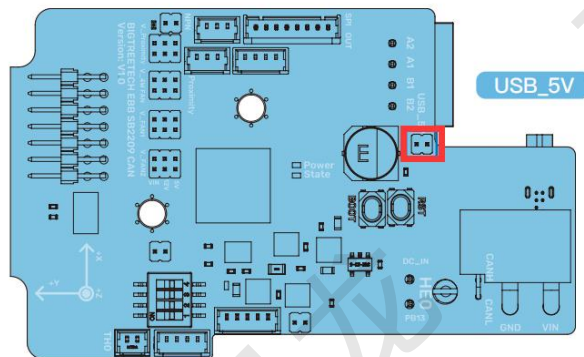
使用 **EBB36/42** 如下图配置

```
(Top)
Katapult Configuration v0.0.1-57-gabd1545
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32G0B1) --->
Build Katapult deployment application (Do not build) --->
Clock Reference (8 MHz crystal) --->
Communication interface (CAN bus (on PB0/PB1)) --->
Application start offset (8KiB offset) --->
(1000000) CAN bus speed
() GPIO pins to set on bootloader entry
[*] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[ ] Enable Status LED
```

配置好以后按 Q 再按 Y 保存离开，然后输入 **make** 编译固件，等待固件编译完成即可。

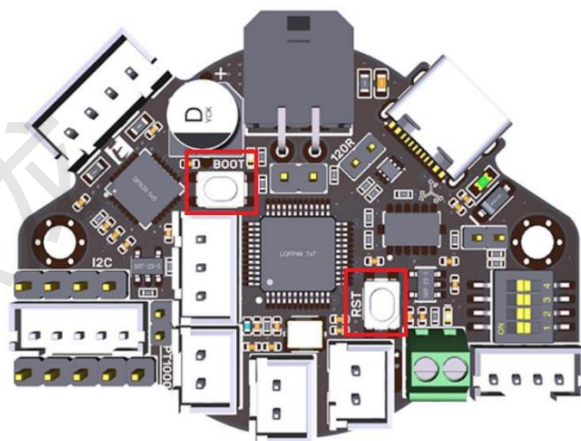
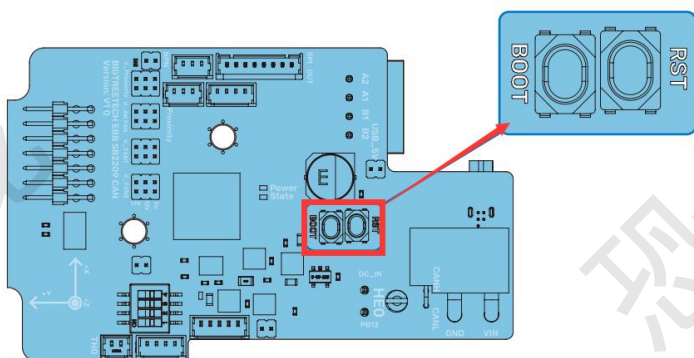
## 4.2 连接 CAN 板和上位机

如未使用 24V 电源，则使用 Type-C 连接线将 EBB SB2209 CAN V1.0(RP2040) 或 EBB36/42 CAN 板连接至主板 USB 口（上位机），并确保已连接 USB\_5V 跳线帽，以便通过 Type-C 为 EBB SB2209 CAN (RP2040)或 EBB36Can 板提供电源。



EBB SB2209 CAN V1.0(RP2040)如下操作进入 DFU 模式：  
按住 Boot 按钮，然后单击一下 Reset 按钮进入 DFU 模式。

使用 EBB36/42CAN 如下操作进入 DFU 模式：  
按住 Boot 按钮，然后单击一下 Reset 按钮进入 DFU 模式。



## 4.3 给 Can 板烧录 katapult 固件

输入：lsusb

EBB SB2209/2240 和 EBB36 如果出现如下图的信息则主板成功进入 DFU：

```
pi@fluidpi:~$ lsusb
Bus 001 Device 005: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 001 Device 004: ID 1d50:6061 OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 001 Device 003: ID 0424:0c00 Microchip Technology, Inc. (formerly SMSC) SMC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. (formerly SMSC) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@fluidpi:~$
```

接着输入以下命令以烧录 katapult 固件：

```
sudo dfu-util -a 0 -d 0483:df11 -s 0x08000000:mass-erase:force -D ~/katapult/out/katapult.bin
```

其中“0483:df11”需替换为查询到的实际设备 ID

**EBB SB Rp2040** 如果出现如下图的信息则主板成功进入 DFU:

```
pi@fluidpi:~$ lsusb
Bus 001 Device 005: ID 2e8a:0003 Raspberry Pi RP2 Boot
Bus 001 Device 004: ID 1d50:6061 OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 001 Device 003: ID 0424:0c00 Microchip Technology, Inc. (formerly SMSC) SMC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. (formerly SMSC) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@fluidpi:~$
```

接着输入以下命令以烧录 katapult 固件:

```
make flash FLASH_DEVICE=2e8a:0003
```

其中“2e8a:0003”需替换为查询到的实际设备 ID。

烧录完成后, 请拔下 USB\_5V 跳线帽和 Type-C 数据线, 如果用它继续供电则不需要拔除。

## 4.4 给 Can 板编译 klipper 固件

使用附带的 CAN 通讯线缆, 连接 CAN 板和主板上的 CAN 通讯接口, 确保供电正常, 将 CAN 板和主板的 120 欧姆的跳线帽均插上。

进入 klipper 目录, 输入:

```
cd ~/klipper
```

配置主板的 klipper 固件, 输入:

```
make menuconfig
```

**EBB SB2209/2240** 入如下图配置

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
  Micro-controller Architecture (STMicroelectronics STM32) --->
  Processor model (STM32G0B1) --->
  Bootloader offset (8KiB bootloader) --->
  Clock Reference (8 MHz crystal) --->
  Communication interface (CAN bus (on PB0/PB1)) --->
(1000000) CAN bus speed
( ) GPIO pins to set at micro-controller startup
```

配置好以后按 Q 再按 Y 保存离开, 然后输入 **make** 编译固件, 等待固件编译完成即可。

**EBB SB Rp2040** 如下图配置

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
  Micro-controller Architecture (Raspberry Pi RP2040) --->
  Bootloader offset (16KiB bootloader) --->
  Communication interface (CAN bus) --->
(4) CAN RX gpio number (NEW)
(5) CAN TX gpio number (NEW)
(1000000) CAN bus speed
( ) GPIO pins to set at micro-controller startup (NEW)
```

配置好以后按 Q 再按 Y 保存离开, 然后输入 **make** 编译固件, 等待固件编译完成即可。



## EBB36/42 如下图配置

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
  Micro-controller Architecture (STMicroelectronics STM32) --->
  Processor model (STM32G0B1) --->
  Bootloader offset (8KiB bootloader) --->
  Clock Reference (8 MHz crystal) --->
  Communication interface (CAN bus (on PB0/PB1)) --->
(1000000) CAN bus speed
() GPIO pins to set at micro-controller startup
```

配置好以后按 Q 再按 Y 保存离开，然后输入 `make` 编译固件，等待固件编译完成即可。

## 4.5 使用 katapult 固件来刷写 Can 板的 klipper 固件

先输入命令: `cd ~/katapult/scripts`

再输入查看设备 uuid 命令: `python3 flash_can.py -i can0 -q`

```
biqu@BTT-CB1:~$ cd ~/katapult/scripts
biqu@BTT-CB1:~/katapult/scripts$ python3 flash_can.py -i can0 -q
Resetting all bootloader node IDs ...
Checking for Katapult nodes ...
Detected UUID: 8030d70005ec Application: Klipper
Detected UUID: f0abf5d02406, Application: Katapult
Query Complete
```

查询到 UUID 后，输入下面命令来烧录 klipper 固件，注意后面的 UUID 替换为实际的 UUID。

`python3 flash_can.py -i can0 -f ~/klipper/out/klipper.bin -u be69315a613c`

等待烧录完成后，再次输入下面命令查询固件状态，

`python3 flash_can.py -i can0 -q`

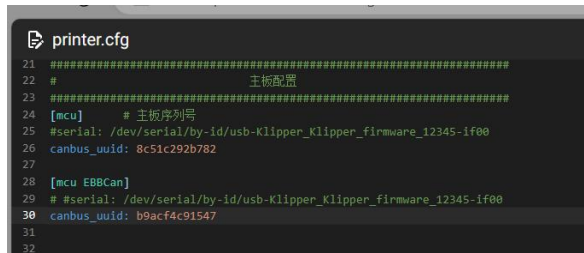
```
Verification Complete: SHA = A1DF04005577E38A8081F2E88F042A949B977D5E
Flash Success
biqu@BTT-CB1:~/katapult/scripts$ python3 flash_can.py -i can0 -q
Resetting all bootloader node IDs ...
Checking for Katapult nodes ...
Detected UUID: 8030d70005ec Application: Klipper
Detected UUID: f0abf5d02406, Application: Klipper
Query Complete
```

此时 Application 后面的 katapult 变为 Klipper，代表 Klipper 已经正常运行。



## 4.6 填写主板 uuid 和 can 板 uuid 进打印机配置

参考，记得替换自己的 uuid。



```
printer.cfg
21 #####
22 #
23 # 主板配置
24 #####
25 [mcu] # 主板序列号
26 #serial: /dev/serial/by-id/usb-Klipper_Klipper_firmware_12345-if00
27 canbus_uuid: 8c51c292b782
28
29 [mcu EBBCan]
30 #serial: /dev/serial/by-id/usb-Klipper_Klipper_firmware_12345-if00
31 canbus_uuid: b9acf4c91547
32
```

```
#####
#
# 主板配置
#####
[mcu] # 主板序列号
#serial: /dev/serial/by-id/usb-Klipper_Klipper_firmware_12345-if00
canbus_uuid: 8c51c292b782

[mcu EBBCan]
#serial: /dev/serial/by-id/usb-Klipper_Klipper_firmware_12345-if00
canbus_uuid: b9acf4c91547
```

填写完毕后保存并重启即可完成上位机和主板、CAN 板的通讯！

## 4.7 FLY 的 SHT36V2 Can 板刷 klipper

如使用 FLY 家的 SHT36 V2 CAN 板，则它已经自带 canboot 固件，只需要配置 klipper 固件即可。

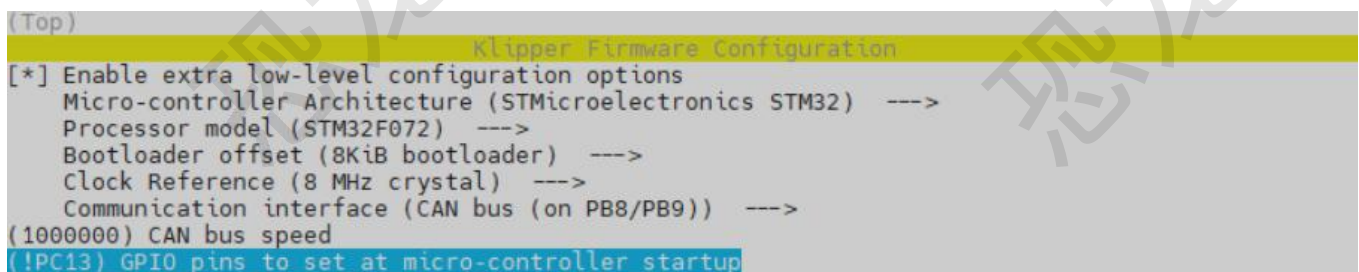
将 36Can 通过通讯线连接到打印机主板并供电，确保接线正确！

进入 ssh 直接配置 SHT 36 V2 CAN 板的 klipper 固件即可

```
cd ~/klipper
```

```
make menuconfig
```

如下图配置即可：



```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
  Micro-controller Architecture (STMicroelectronics STM32) --->
  Processor model (STM32F072) --->
  Bootloader offset (8KiB bootloader) --->
  Clock Reference (8 MHz crystal) --->
  Communication interface (CAN bus (on PB8/PB9)) --->
  (1000000) CAN bus speed
  (!PC13) GPIO pins to set at micro-controller startup
```

配置好以后按 Q 再按 Y 保存离开，然后输入 make 编译固件，等待固件编译完成。

接着烧录固件

```
cd ~/klipper
```

输入查询 Can 板 UUID 命令：

```
python3 ~/klipper/lib/canboot/flash_can.py -q
```

接着输入下面的命令烧录 klipper 固件，注意后面的 UUID 替换为实际的 UUID。

```
python3 lib/canboot/flash_can.py -i can0 -f ./out/klipper.bin -u fea6a45462e9
```

## 5.0 katapult（原 CanBoot）使用指南（未完成）

### 5.1 什么是 katapult？

Katapult 是一种为 ARM Cortex-M mcu 设计的引导加载程序。这个引导加载程序最初是为 CAN 节点设计的，以便与 Klipper 一起使用。引导加载程序本身利用了 Klipper 的硬件抽象层，将内存占用最小化。除了 CAN, katapult 现在还支持 USB 和 UART 接口。目前支持 lpc176x、stm32 和 rp2040 三种 mcu。CAN 支持目前仅限于 stm32 f 系列和 rp2040 设备。

Klipper 已支持 katapult，通过 CANBUS 直接烧录固件。使用 katapult 后为主板和 Can 通讯设备更新 klipper 固件就无需再连接 USB 线，保持现有的 CAN 连接的情况下可直接烧录固件，能够更便捷、高效的更新 CAN 工具板的固件。

### 5.2 更新 Klipper 固件

已经烧录过 Canboot 引导固件和带 Canboot 的固件，以后需要更新 klipper 固件看下面操作即可：

进入 klipper 目录，输入：

```
cd ~/klipper
```

拉取最新的 klipper 文件

```
git pull
```

编译最新的 klipper 固件

```
make clean
```

```
make menuconfig
```

继续按照文档中的 3.2 和 4.4 配置完后，输入 `make` 编译固件，等待固件编译完成

```
make -j4
```

#### 对于 CAN 设备：

先输入下面命令查询 UUID

```
cd ~/katapult/scripts
```

```
python3 flash_can.py -i can0 -q
```

下面命令中的 fea6a45462e9 需要替换为你查询到的 UUID

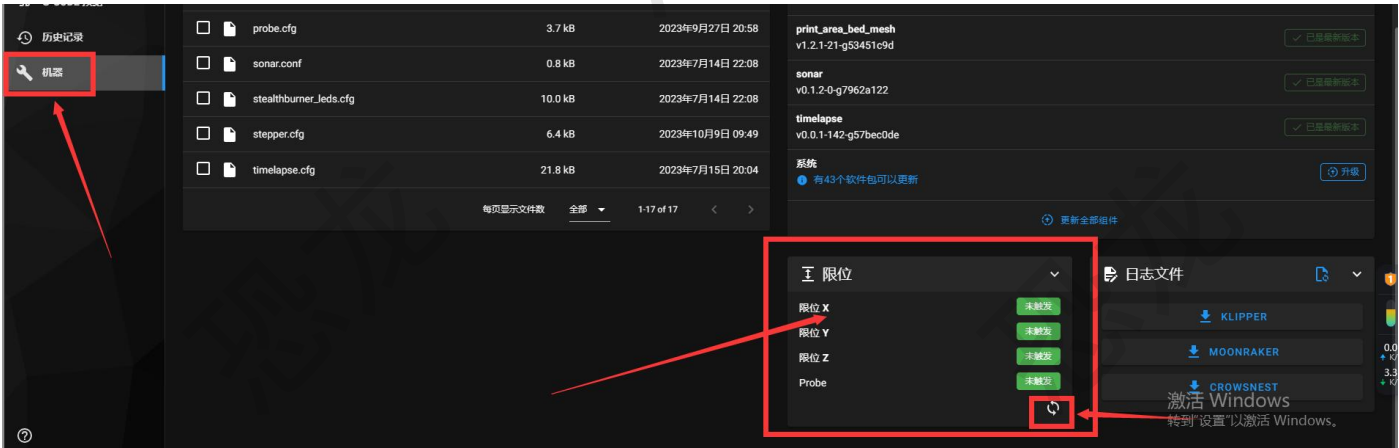
```
python3 flash_can.py -i can0 -f ~/klipper/out/klipper.bin -u be69315a613c
```

#### 对于 USB/UART 设备：阿巴阿巴阿巴阿吧

## 6.调机器教程

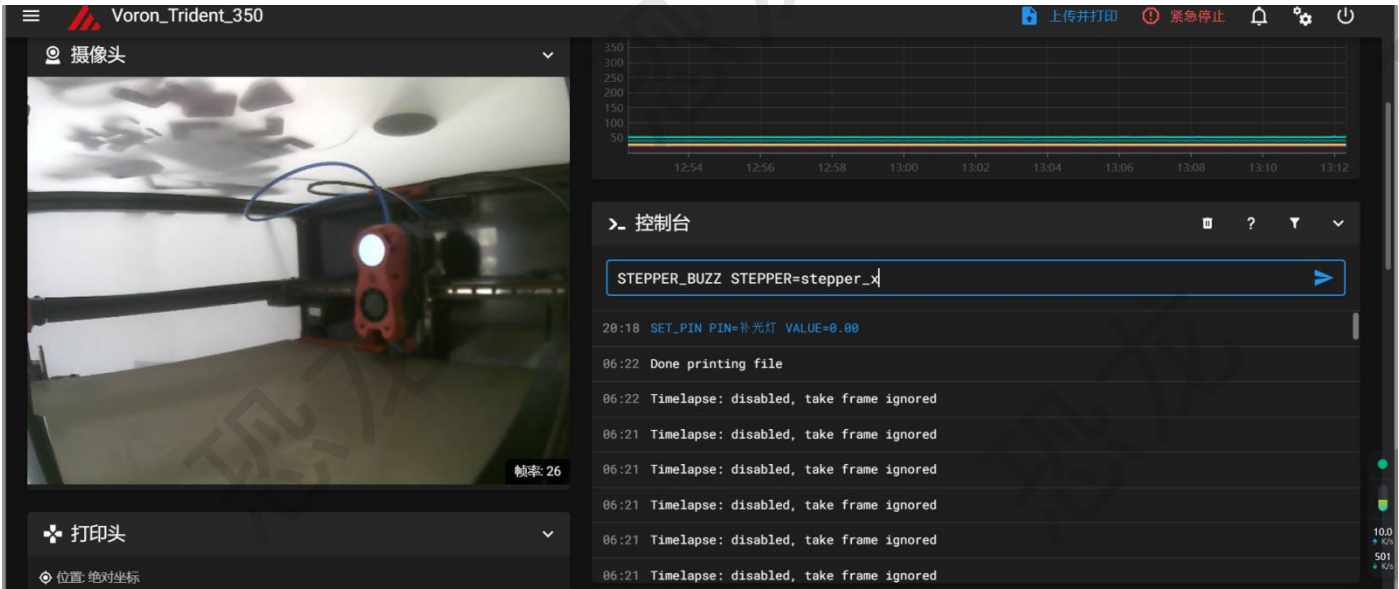
### 6.1 限位检查

点击“机器”，右下角限位模块里面点击刷新按钮，未触发提示已触发就修改限位引脚，前面增加或者删除“！”，然后挨个手动触发再点击刷新，查看是否正常！



### 6.2 电机检查

如下图在打印机的控制台依次输入



STEPPER\_BUZZ STEPPER=stepper\_x  
STEPPER\_BUZZ STEPPER=stepper\_y  
STEPPER\_BUZZ STEPPER=stepper\_z  
STEPPER\_BUZZ STEPPER=stepper\_z1  
STEPPER\_BUZZ STEPPER=stepper\_z2  
STEPPER\_BUZZ STEPPER=stepper\_z3

X 电机应左右，左右的轻微转动  
Y 电机应左右，左右的轻微转动  
Z 电机(2.4 左前，三叉戟左前)应轻微上下转动  
Z1 电机（2.4 左后，三叉戟后中）应轻微上下转动  
Z2 电机(2.4 右后，三叉戟右前)应轻微上下转动  
Z3 电机(2.4 右前)应轻微上下转动

2.4 的皮带带动龙门上下，上下的微动。  
三叉戟的丝杆带动热床下上，下上的微动。

小恐龙交流群：786115036

如果方向不对，在电机配置的方向引脚前面增加或删除 “！”

如下图位置：

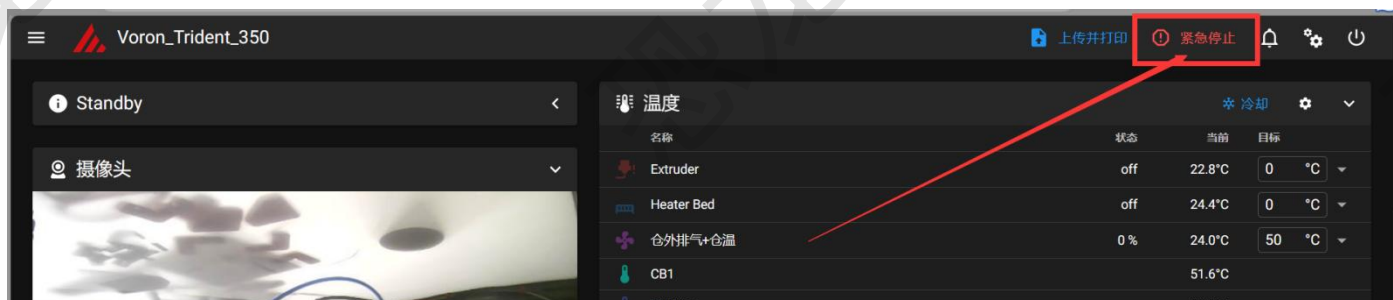
```
#####
#                               X/Y轴电机
#####
## X 轴步进电机 on MOTOR0(B Motor)
[stepper_x]
step_pin: PF12                # Y轴电机脉冲引脚
dir_pin: PF11                 # 方向设置
enable_pin: !PB5              # 使能引脚
rotation_distance: 40          # 主动轮周长mm (2GT-20T为 40mm 16T为 32mm)
microsteps: 32                 # 细分
full_steps_per_rotation:200    # 单圈脉冲数-对于0.9度步进设置为400
endstop_pin: EBBCan: PB6       # 限位开关接口
#endstop_pin: tmc5160_stepper_x:virtual_endstop
position_min: 0                # 软限位最小行程
position_endstop: 350          # 软限位最大行程 (250mm-300mm-350mm)
position_max: 350              # 机械限位最大行程 (250mm-300mm-350mm)
homing_speed: 100              # 复位速度-最大 100
homing_retract_dist: 5         # 后撤距离
homing_positive_dir: true      # 复位方向
```

## 6.3XY 移动方向检查

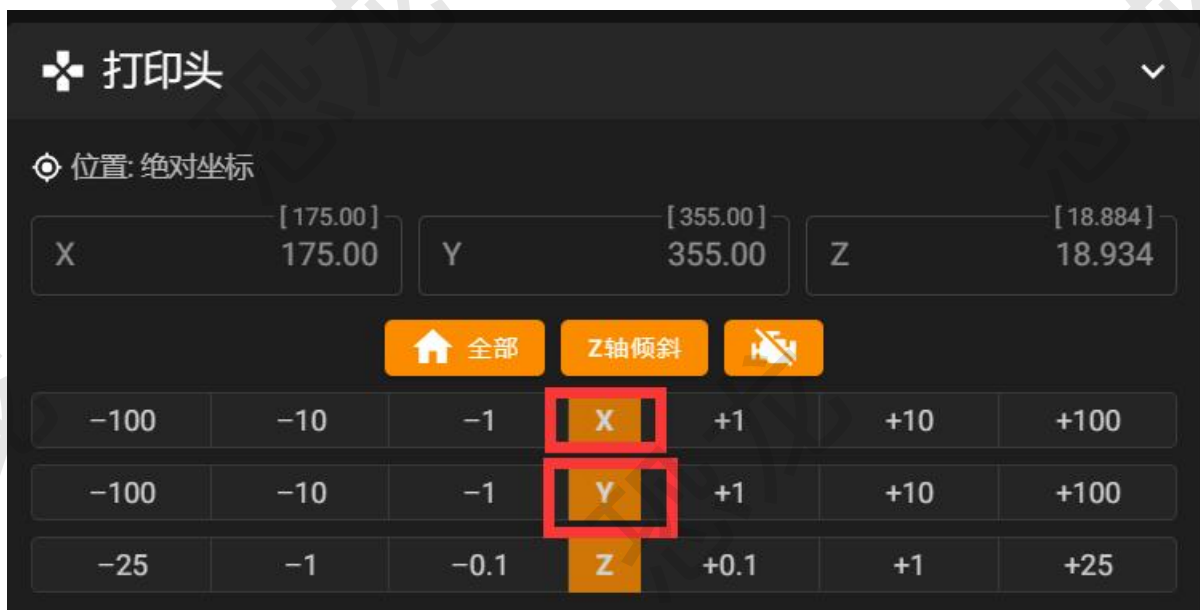
Cxy 结构的打印机左后方为 B（Y）电机，右后方的电机为 A（X）电机

Voron 配置里的归为方向为 **X 轴**向右移动，**Y 轴**向后移动，切记！切记！！切记！！

在 xy 方向检查开始之前，为了防止发生不必要的碰撞造成机器损坏，**请随时准备点击急停按钮！**



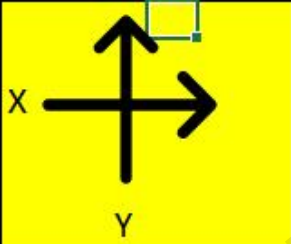
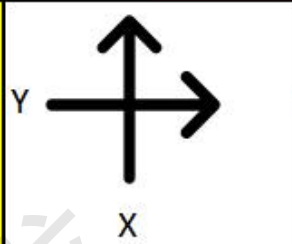
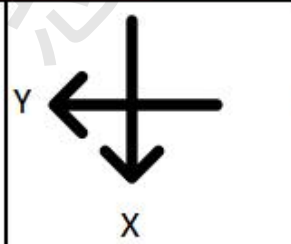
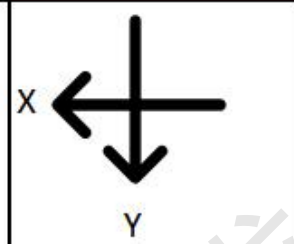
首先将打印头移动到打印机热床的最中间位置，然后依次点击下面的图标



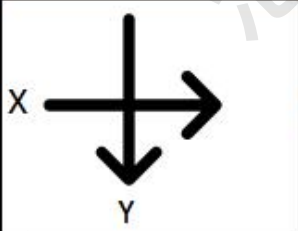
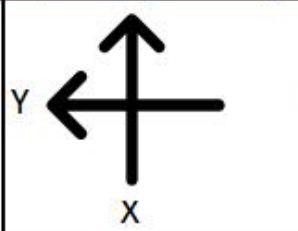
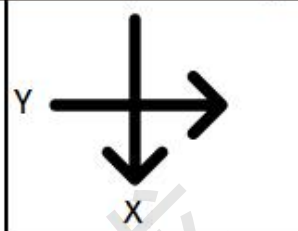
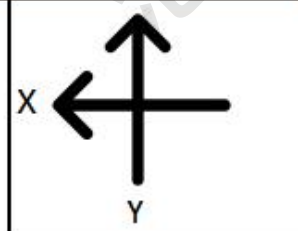


小恐龙交流群：786115036

先点击 X 图标，X 轴会进行归位，如果打印头向右归位则正常，如果向其他方向则不正常，请立即急停！记住他的方向，再次将打印头移动到打印机热床的最中间位置  
再点击 Y 图标，Y 轴归位，如果打印头向后归位则正常，如果向其他方向则不正常，请立即急停！  
然后参考下图，如果出现下图情况的其中一种，则在对应轴电机配置的方向引脚前面增加或删除 “！”

| Motor B   | Motor A | Motor B   | Motor A  | Motor B  | Motor A | Motor B   | Motor A  |
|---|---------|---|----------|--|---------|---|----------|
| OK  | OK      | OK  | Inverted | Inverted   | OK      | Inverted  | Inverted |
|  |         |  |          |  |         |  |          |
| Y   |         | X   |          | X  |         | Y   |          |

如果出现下图情况的其中一种，则对调 XY 电机线。

| Motors are swapped, swap X and Y connectors<br>There is no possible good configuration here just inverting directions |  |   |  |
|---|--|---|--|
|                                     |  |  |  |
| Y   | X  | X   | Y  |

接着继续重复上述步骤，直到方向完全正确即可。

## 6.4 Z 轴复位传感器位置定义

执行一次 G28 XY，然后利用控制台点动移动喷嘴的 X 或 Y 位置，直到喷嘴位于 Z 轴限位开关的正上方。发送 M114 命令并记录 X 轴和 Y 轴的坐标值。打开 printer.cfg 文件，更新回原点宏例程（[homing\_override]）如下图，然后保存文件并重启 klipper。

如果您是使用[safe\_z\_home]，那么可采用上述方法进行参数的更新替换。

修改完毕后，可尝试使用 G28 命令完成完整的复位动作。如果在 G28 结束时，您的喷嘴没有超过热床的平面，请调整 Z 抬升高度。

```
##### 归位和龙门调整程序 #####
#
#####
# 以下两种设置z轴限位位置方式选择一种，把另一种注释掉即可，
# [homing_override]
# axes: z
# set_position_z: 0
# gcode:
#   G90
#   G0 Z15 F600
#   G28 X Y
#   ## z 限位开关的 xy 位置
#   ##通过后将x0和y0更新为你的值（如X157、Y305）
#   ## z 轴限位位置定义
#   G0 X175 Y175 F3600 #250mm
#   G28 Z
#   G0 Z10 F1800
#
#-----
[safe_z_home]          # z轴限位坐标
home_xy_position:175,175 # z轴限位位置定义（重要！！！自行进行调整）
speed:100              # 归位速度
z_hop:10               # 归位之后抬升高度
#-----
```

## 6.5 加热器 PID 校准

1、热床 PID 校准命令如下

```
PID_CALIBRATE HEATER=heater_bed TARGET=100
```

2、加热棒 PID 校准命令如下，记得先打开模型散热风扇，风力调整到 25%。

```
PID_CALIBRATE HEATER=extruder TARGET=245
```

## 6.6 probe 传感器精度

首次测试 probe 传感器，不需要加热喷头和热床，G28 复位后将打印头移到热床的中心，然后发送命令：

```
PROBE_ACCURACY
```

此时喷头会自动下降直至 probe 传感器被触发后反向抬升，如此连续探测热床 10 次，并在最后输出一个标准偏差值，标准偏差应小于 0.003mm。如果每次的探测值不稳定或者是呈趋势性的变化，需要检查 Z 轴皮带的松紧度。劣质的传感器也会影响重复探测的精度。

## 6.7 龙门调平（或 Z 倾斜）

在进行 4Z 调平操作之前，先打开 printer.cfg 配置文件，找到 4Z 调平的宏[quad\_gantry\_level]，根据热床大小调整 4 个探测点的坐标位置，确保探测点都在热床打印区域以内。

第一次运行 4Z 调平前，请先用手将龙门的 4 个角高度调至差不多一致，先运行一次 G28 归位，然后发送命令：

```
Voron2.4 发送: Quad_Gantry_Level
```

```
三叉戟发送: Z_Tilt
```

打印机开始执行自动调平动作，喷头从左前角开始采集高度数据，并按逆时针依次探测完 4 个点。每次探测完，系统会自动计算出一个平均的公差值，并自动调整 4 个角的高度。重复探测 2-3 次，直至公差值小于 retry\_tolerance: 的设定值，即完成了 4Z 调平的动作。

注意：果偏差越来越大，请检查是电机的顺序是否正确。如果探测次数超过 5 圈仍未达到预设公差，系统将会报错。Z 轴皮带的松紧程度以及劣质的传感器都会影响重复探测的精度

## 6.8 挤出机校准

在第一次打印之前，需要确保挤出机能挤出正确长度的材料。根据打印材料所需的温度，加热挤出头，从挤出机进料口的铁氟龙管口处量 150mm 料丝，并在 100mm 处用美工刀做一个标记。在网页中，手动操作挤出 50mm<sup>2</sup> 次，共 100mm（Klipper 的单个挤出量不允许超过 50mm）。待挤出机停止后，测量从挤出机进料口到标记处的长度 x。如果挤出量正确的时候，x 应该为 50mm（150mm - 100mm = 50mm）。但实际是会有偏差的，找到配置文件中的现有挤出值，并使用以下方法更新它。

```
#####
#                               挤出机配置
#####
[extruder]
step_pin: EBBCan: PD0
dir_pin: EBBCan: PD1
enable_pin: !EBBCan: PD2
microsteps: 16
rotation_distance: 4.27718718
nozzle_diameter: 0.400
filament_diameter: 1.750
heater_pin: EBBCan: PB13
sensor_type: Generic 3950
sensor_pin: EBBCan: PA3
control: pid
```

### 新收缩值=旧收缩值×（实际挤出量÷目标挤出量）

将新值替换配置文件中，保存并重启 klipper。然后按上述方法重新再验证一遍，如果挤出量在目标值的±0.5%范围内（即，目标值为 99.5-100.5mm，目标挤出长度为 100mm），挤出机就校准好了。

## 6.9 Z 复位偏移调整

对 Z 复位偏移的调整，其实就是调整 Z 轴在高度为 0 时，喷嘴距热床的高度，他关系到打印时，首层的高度是否正确，料丝是否能良好的粘附在热床上。在做此项操作前，请将挤出机设置为 240℃，热床加热至 100℃，预热 15 分钟后开

始以下操作。

- 1、先执行一次 G32 。
- 2、然后将打印头移至热床中心，运行命令

#### Z\_ENDSTOP\_CALIBRATE

然后使用命令 TESTZ Z=-1 缓慢地将喷头移向热床表面，直到喷头离热床表面约 1mm 时，在热床与喷嘴之间垫一张 A4 白纸，然后使用微调指令 TESTZ Z=-0.1 使喷嘴慢慢接近热床表面，直至喷嘴接触到 A4 纸，用手抽拉白纸有轻微阻力，此时可执行确认命令 ACCEPT，然后执行保存命令 SAVE\_CONFIG，将自动把参数保存到配置文件中。

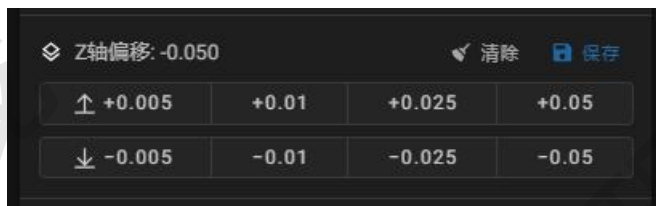
在上述调试过程中，如果喷头下降的太多，导致 A4 纸抽拉不动，可使用正数 TESTZ Z=0.1，使喷嘴抬高。

重要：在保存完参数后，马上执行 G28 复位，使喷头离开热床表面，以免烫坏 PEI 膜。

如果在执行过程中出现报错(越界)，一般是由于 Z 限位传感器的轴太长，可能会在打印过程中卡住打印头。最好切割轴，使其与 PEI 的表面平齐。

## 6.10 动态微调 Z 轴高度

由于 A4 纸大法并不能完美首层高度，所以打印过程中，可以使用工具面板上的 Z 偏移微调按钮来动态调整喷嘴的高度。



调整完成如果希望系统一直使用当前高度，直接点击保存即可。

## 6.11 加速度计的使用

如果使用了带加速度计 CAN 板，则可以直接使用，参考 Can 板引脚配置好加速度计后，如果是 CB1 上位机直接输入下面的命令即可。

SHAPER CALIBRATE

测试完成后直接点击保存即可。

如果不是 CB1 上位机则需要安装科学依赖库

## 1.安装 Python 的科学计算库

```
~/klippy-env/bin/pip install -v numpy
```

## 2.安装系统依赖库

```
sudo apt install python-numpy python-matplotlib
```

sudo 指令需要输入用户的密码

### 3.查看 klipper 是否会正常运行

ACCELEROMETER QUERY

#### 4.输入代码开始自动测试打印机震动

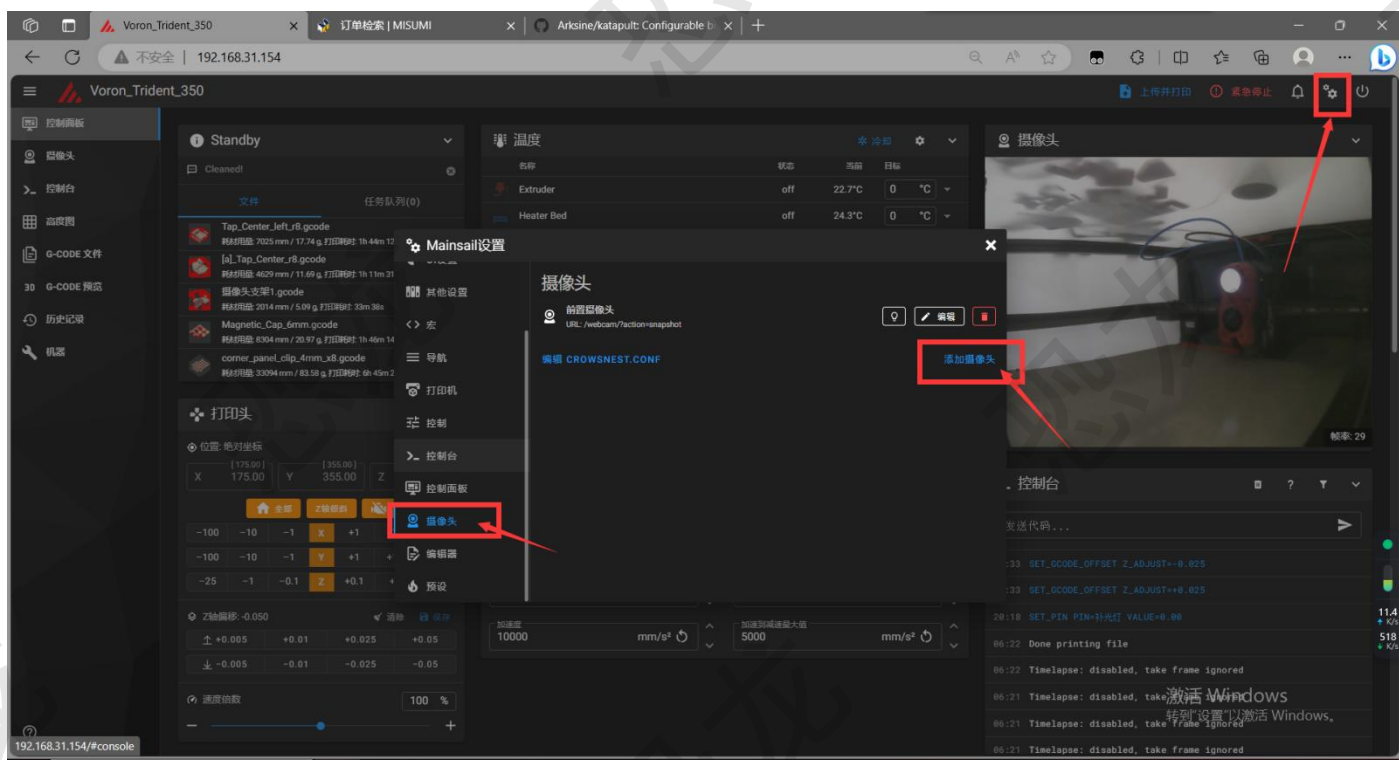
SHAPER CALIBRATE

## 5.保存

SAVE CONFIG

## 6.12 摄像头配置

CB1 系统镜像已经配置好摄像头，只需要插上 USB 免驱摄像头再按照下图添加即可，如图



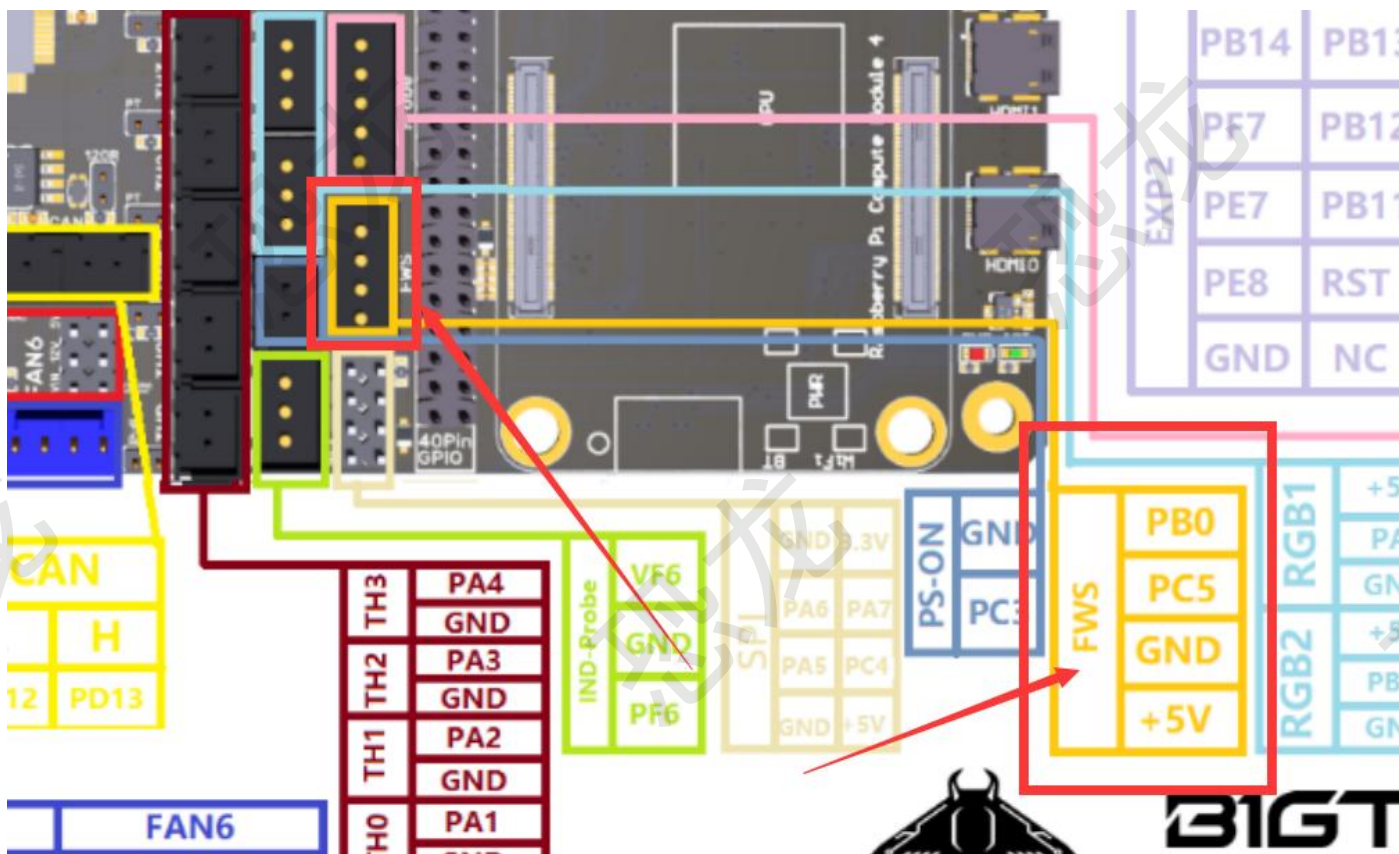


## 6.13 添加 KlipperScreen 屏幕

CB1 系统镜像已经配置好屏幕组件，只需要插上屏幕，即可点亮屏幕控制。

## 6.14 必趣 SFS2.0 智能断料传感器

如果使用 M8P 主板，则推荐将 SFS2.0 的两个 3pin 接口改成一个 4pin 接口，然后插在下图位置。



不要使用官方文档的配置，官方的配置莫名其妙的，推荐使用我这个配置，如下：

[filament\_motion\_sensor 转堵监测]

detection\_length: 2.88

# 触发传感器 switch\_pin 引脚状态变化的最小距离，默认为 2.88mm。

extruder:extruder

# 该传感器相关联的挤出机，必须提供此参数。

switch\_pin:PB0

# 连接到检测开关的引脚。必须提供此参数。#pause\_on\_runout:

runout\_gcode: PAUSE

# 在检测到耗材耗尽后会执行的 G 代码命令列表。

#insert\_gcode:

# 在检测到耗材插入后会执行的 G-Code 命令列表。默认不运行任何 G-Code 命令，这将禁用耗材插入检测。

#event\_delay:

# 事件之间的最小延迟时间（秒）。在这个时间段内触发的事件将被默许忽略。默认为 3 秒。

#pause\_delay:

小恐龙交流群: 786115036

# 暂停命令和执行 `runout_gcode` 之间的延迟时间, 单位是秒。如果在 `OctoPrint` 的情况下, 增加这个延迟可能改善暂停的可靠性。如果 `OctoPrint` 表现出奇怪的暂停行为, 考虑增加这个延迟。默认为 0.5 秒。

[`filament_switch_sensor` 断料监测]。

`pause_on_runout`: True

# 当设置为 "True" 时, 会在检测到耗尽后立即暂停打印机。请注意, 如果 `pause_on_runout` 为 False 并且没有定义。 `runout_gcode` 的话, 耗尽检测将被禁用。默认为 True。

`runout_gcode`: PAUSE

# 在检测到耗材耗尽后会执行的 G 代码命令列表。如果 `pause_on_runout` 被设置为 True, 这个 G-Code 将在暂停后执行。默认情况是不运行任何 G-Code 命令。

# `insert_gcode`:

# 在检测到耗材插入后会执行的 G-Code 命令列表。默认不运行任何 G-Code 命令, 这将禁用耗材插入检测。

`event_delay`: 3.0

# 事件之间的最小延迟时间 (秒)。在这个时间段内触发的事件将被默许忽略。默认为 3 秒。

`pause_delay`: 0.5

# 暂停命令和执行 `runout_gcode` 之间的延迟时间, 单位是秒。如果在 `OctoPrint` 的情况下, 增加这个延迟可能改善暂停的可靠性。如果 `OctoPrint` 表现出奇怪的暂停行为, 考虑增加这个延迟。默认为 0.5 秒。

`switch_pin`: PC5

# 连接到检测开关的引脚。必须提供此参数。