CanBoot use

What is CanBoot?

CanBoot is a bootloader designed for ARM Cortex-M mcu. This bootloader was originally designed for CAN nodes for use with Klipper. The bootloader itself utilizes Klipper's hardware abstraction layer to minimize memory usage. In addition to CAN, CanBoot now also supports USB and UART interfaces. Currently, three types of mcu are supported: lpc176x, stm32 and rp2040. CAN support is currently limited to stm32 f-series and rp2040 devices.

Klipper already supports CanBoot and can directly burn firmware through CANBUS. After using CanBoot, there is no need to connect the USB cable to update the klipper firmware for the SHT36/42 board. The firmware can be burned directly while maintaining the existing CAN connection, making it more convenient and efficient to update the firmware of the CAN tool board.

1. Compile CanBoot boot firmware

1. Enter the SSH terminal
2. Execute the following command

```
git clone https://github.com/Arksine/CanBoot
cd CanBoot
```

```
fly@flygemini:~$ git clone https://github.com/Arksine/CanBoot
Cloning into 'CanBoot'...
remote: Enumerating objects: 534, done.
remote: Counting objects: 100% (135/135), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 534 (delta 100), reused 100 (delta 89), pack-reused 399
Receiving objects: 100% (534/534), 2.55 MiB | 1.61 MiB/s, done.
Resolving deltas: 100% (346/346), done.
Updating files: 100% (167/167), done.
fly@flygemini:~$
```

- Please pay attention to the motherboard model you are using

FLY-SHT36

```
(Top)
                                                CanBoot Configuration v0.0.1-
    Micro-controller Architecture (STMicroelectronics STM32)  --->
    Processor model (STM32F072)  --->
    Build CanBoot deployment application (Do not build)  --->
    Clock Reference (8 MHz crystal)  --->
    Communication interface (CAN bus (on PB8/PB9))  --->
    Application start offset (8KiB offset)  --->
(500000) 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
(!PC13) Status LED GPIO Pin
```

```
fly@flygemini:~/CanBoot$ make
  Building out/autoconf.h
  Compiling out/src/canboot_main.o
  Compiling out/src/led.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/flash.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/generic/armcm_boot.o
  Compiling out/src/generic/armcm_irq.o
  Compiling out/src/generic/crc16_ccitt.o
  Compiling out/src/../lib/stm32f0/system_stm32f0xx.o
  Compiling out/src/stm32/stm32f0.o
  Compiling out/src/stm32/stm32f0_timer.o
  Compiling out/src/stm32/gpioperiph.o
  Compiling out/src/stm32/can.o
  Compiling out/src/../lib/fast-hash/fasthash.o
  Compiling out/src/generic/canbus.o
  Building out/compile_time_request.o
  Preprocessing out/src/generic/armcm_link.ld
  Linking out/canboot.elf
  Creating hex file out/canboot.bin
fly@flygemini:~/CanBoot$
```

-

SB2040&ERCF

Tip

If you do not have RP2040 in your CanBoot options, please pull the latest CanBoot

```
(Top)
                                        CanBoot Configuration v0.0.1-33-g88e208a
    Micro-controller Architecture (Raspberry Pi RP2040)  --->
    Flash chip (W25Q080 with CLKDIV 2)  --->
    Build CanBoot deployment application (Do not build)  --->
    Communication interface (CAN bus)  --->
(4) CAN RX gpio number
(5) CAN TX gpio number
(500000) 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
(gpio24) Status LED GPIO Pin
```

After the configuration is completed, press the "Q" key, and then press the "Y" key to exit and save. Execute the following command to compile the firmware

```
make clean
make -j4
```

```
fly@flygemini:~/CanBoot$ make
  Building out/autoconf.h
  Compiling out/src/canboot_main.o
  Compiling out/src/led.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/flash.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/generic/armcm_boot.o
  Compiling out/src/generic/armcm_irq.o
  Compiling out/src/generic/crc16_ccitt.o
  Compiling out/src/../lib/stm32f0/system_stm32f0xx.o
  Compiling out/src/stm32/stm32f0.o
  Compiling out/src/stm32/stm32f0_timer.o
  Compiling out/src/stm32/gpioperiph.o
  Compiling out/src/stm32/can.o
  Compiling out/src/../lib/fast-hash/fasthash.o
  Compiling out/src/generic/canbus.o
  Building out/compile_time_request.o
  Preprocessing out/src/generic/armcm_link.ld
  Linking out/canboot.elf
  Creating hex file out/canboot.bin
fly@flygemini:~/CanBoot$
```

- `Creating hex file out/canboot.bin`If or appears like the picture above , `Creating uf2 file out/canboot.uf2`the compilation is successful.

FLY-SHT36 v2

```
(Top)
                                                CanBoot Configuration v0.0.1-
    Micro-controller Architecture (STMicroelectronics STM32)  --->
    Processor model (STM32F072)  --->
    Build CanBoot deployment application (Do not build)  --->
    Clock Reference (8 MHz crystal)  --->
    Communication interface (CAN bus (on PB8/PB9))  --->
    Application start offset (8KiB offset)  --->
(500000) 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
(!PC13) Status LED GPIO Pin
```

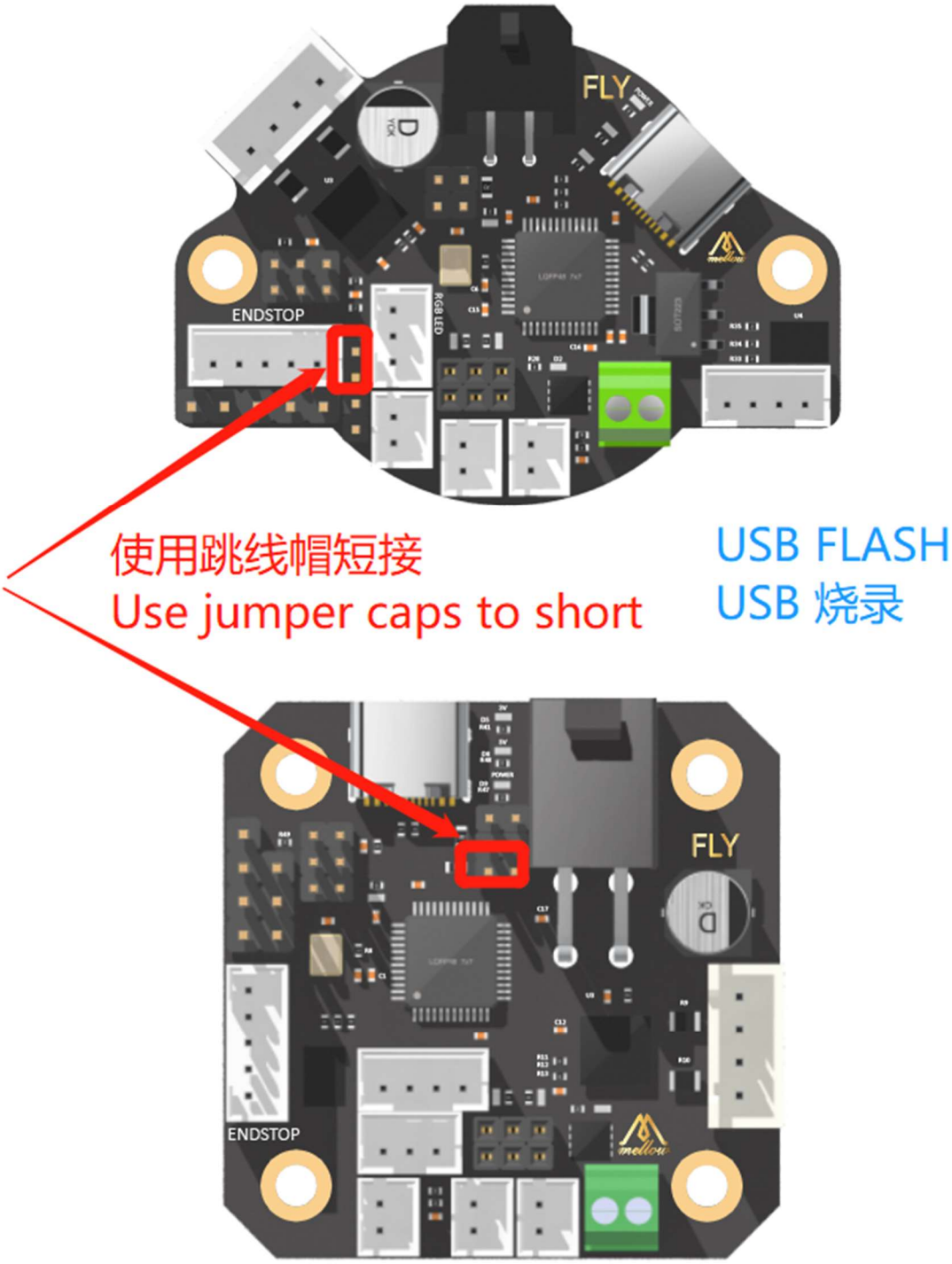
```
fly@flygemini:~/CanBoot$ make
  Building out/autoconf.h
  Compiling out/src/canboot_main.o
  Compiling out/src/led.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/flash.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/generic/armcm_boot.o
  Compiling out/src/generic/armcm_irq.o
  Compiling out/src/generic/crc16_ccitt.o
  Compiling out/src/../lib/stm32f0/system_stm32f0xx.o
  Compiling out/src/stm32/stm32f0.o
  Compiling out/src/stm32/stm32f0_timer.o
  Compiling out/src/stm32/gpioperiph.o
  Compiling out/src/stm32/can.o
  Compiling out/src/../lib/fast-hash/fasthash.o
  Compiling out/src/generic/canbus.o
  Building out/compile_time_request.o
  Preprocessing out/src/generic/armcm_link.ld
  Linking out/canboot.elf
  Creating hex file out/canboot.bin
fly@flygemini:~/CanBoot$
```

-

2. Burn CanBoot boot firmware

FLY-SHT

1.


FLY
ENDSTOP
使用跳线帽短接
Use jumper caps to short
USB FLASH
USB 烧录
FLY
ENDSTOP

2.
3.
4.

```
fly@flygemini:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 002: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 007 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 006 Device 002: ID 1d50:606f OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 006 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 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
fly@flygemini:~/klipper$
```

5.
6.

```
fly@flygemini:~/klipper$ dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08000000 -D ~/klipper/out/klipper.bin
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
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 2048
DfuSe interface name: "Internal Flash  "
Downloading to address = 0x08000000, size = 19620
Download        [=========================] 100%        19620 bytes
Download done.
File downloaded successfully
fly@flygemini:~/klipper$
```

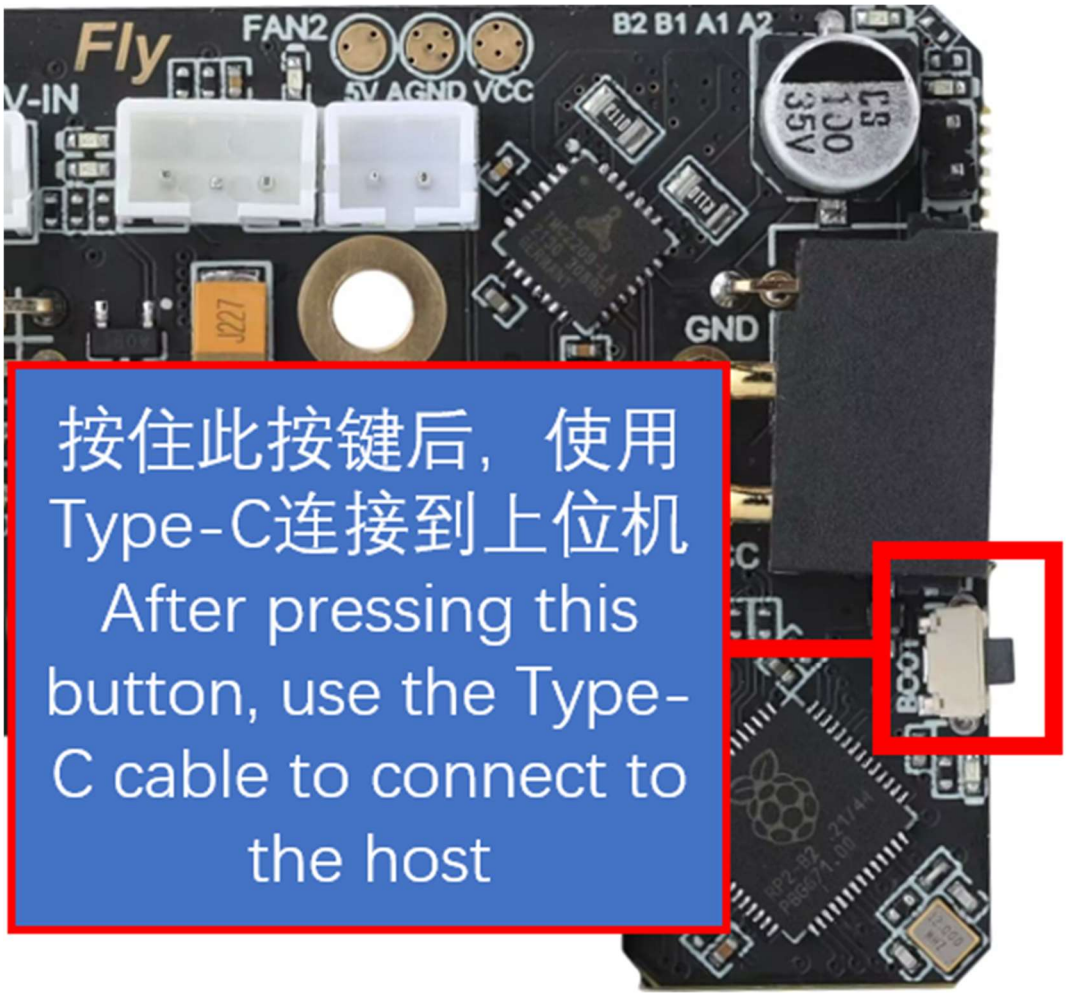
7.

SB2040&ERCF

Tip

If you do not have RP2040 in your CanBoot options, please pull the latest CanBoot (enter in the CanBoot directory `git pull`to pull the latest CanBoot)

1. Check whether it is connected to the BOOT burning mode of SB2040
   Press and hold the BOOT button of the SB2040 board, and then connect the USB to the host computer



lsusb

Execute the above command to see if ID 2e8a:0003 Raspberry Pi RP2 Bootthis line exists. If not, please check the USB cable (remember to hold down the BOOT key before connecting)

```
fly@flygemini:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 002: ID 2e8a:0003 Raspberry Pi RP2 Boot
Bus 007 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 006 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 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 1b3f:1167 Generalplus Technology Inc. WEB CAM
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

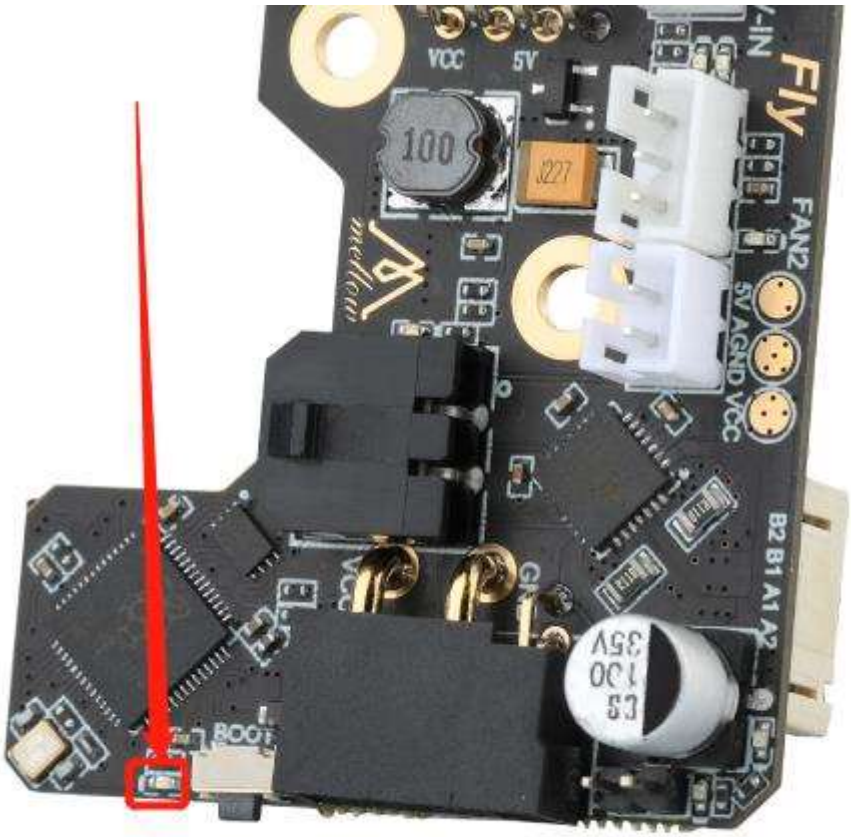
2. Burn
3. cd ~/CanBoot/
   make flash FLASH_DEVICE=2e8a:0003
   Executing the above command may prompt you to enter a password. Just enter the password of the current user. The password will not be visible when you enter it. After typing, press Enter

If the following picture appears, the burning is successful.

```
fly@flygemini:~/klipper$ make flash FLASH_DEVICE=2e8a:0003
  Building rp2040_flash
gcc -c -Wall -ggdb -I../rp2040/ `pkg-config libusb-1.0 --cflags` main.c -o main.o
gcc -c -Wall -ggdb -I../rp2040/ `pkg-config libusb-1.0 --cflags` picoboot_connection.c
gcc  main.o picoboot_connection.o `pkg-config libusb-1.0 --libs` -o rp2040_flash
  Flashing out/klipper.uf2 to 2e8a:0003
sudo lib/rp2040_flash/rp2040_flash out/klipper.uf2

[sudo] password for fly:
Loaded UF2 image with 114 pages
Found rp2040 device on USB bus 7 address 2
Flashing...
Resetting interface
Locking
Exiting XIP mode
Erasing
Flashing
Rebooting device
fly@flygemini:~/klipper$
```

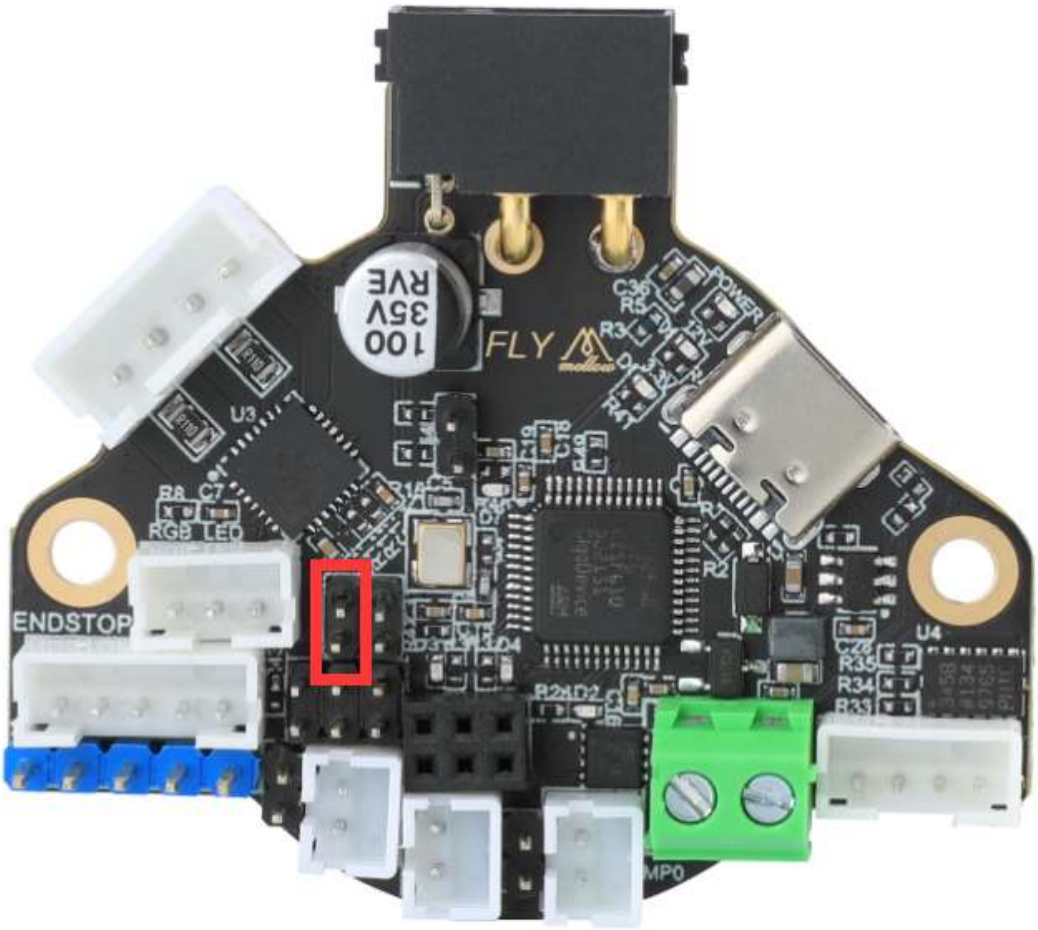
3. examine

If the configuration is compiled correctly and the programming is successful, the light on the SB2040 board will flash at a certain frequency! ! !



SHT36 v2

1.



2.
3.
4.

```
fly@flygemini:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 002: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 007 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 006 Device 002: ID 1d50:606f OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 006 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 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
fly@flygemini:~/klipper$
```

5.
6.

```
fly@flygemini:~/klipper$ dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08000000 -D ~/klipper/out/klipper.bin
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
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 2048
DfuSe interface name: "Internal Flash  "
Downloading to address = 0x08000000, size = 19620
Download        [=========================] 100%        19620 bytes
Download done.
File downloaded successfully
fly@flygemini:~/klipper$
```

7.

3. Burn Klipper firmware with CanBoot for the first time

- Make sure your SHT36/42 is properly connected to the UTOC or other CAN device

1. Pull the latest klipper

```
cd ~/klipper
git pull
```

```
fly@flygemini:~/CanBoot$ cd ~/klipper
fly@flygemini:~/klipper$ git pull
hint: Pulling without specifying how to reconcile divergent branches is
hint: discouraged. You can squelch this message by running one of the following
hint: commands sometime before your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
remote: Enumerating objects: 201, done.
remote: Counting objects: 100% (127/127), done.
remote: Total 201 (delta 127), reused 127 (delta 127), pack-reused 74
Receiving objects: 100% (201/201), 576.37 KiB | 2.08 MiB/s, done.
Resolving deltas: 100% (156/156), completed with 46 local objects.
From https://github.com/Klipper3d/klipper
   7e76bd56..2c441b45  master                -> origin/master
   c3db3ec7..9c43c908  gh-pages              -> origin/gh-pages
 * [new branch]        work-install-20220510 -> origin/work-install-20220510
Updating 7e76bd56..2c441b45
Fast-forward
 docs/Bootloaders.md      |  38 +++++++++++
 docs/G-Codes.md          |   2 +-
 klippy/pins.py           |   4 +-
 lib/README               |   6 ++
 lib/canboot/flash_can.py | 553 +++++++++++++++++++++++++++++++++++++++++++++++++
 scripts/canbus_query.py  |  13 +++-
 src/generic/canbus.c     |  38 +++++++++--
 src/stm32/Kconfig        |   2 +-
 src/stm32/i2c.c          |   4 +-
 9 files changed, 649 insertions(+), 11 deletions(-)
 create mode 100755 lib/canboot/flash_can.py
fly@flygemini:~/klipper$
```

2. Configure the latest klipper firmware

make menuconfig

- Please pay attention to the motherboard model you are using

FLY-SHT36

1.

```
(Top)

[*] 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))  --->
(500000) CAN bus speed
()  GPIO pins to set at micro-controller startup
```

-
2.

```
  Compiling out/src/buttons.o
  Compiling out/src/tmcuart.o
  Compiling out/src/neopixel.o
  Compiling out/src/pulse_counter.o
  Compiling out/src/stm32/watchdog.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/generic/crc16_ccitt.o
  Compiling out/src/generic/armcm_boot.o
  Compiling out/src/generic/armcm_irq.o
  Compiling out/src/generic/armcm_reset.o
  Compiling out/src/../lib/stm32f0/system_stm32f0xx.o
  Compiling out/src/generic/timer_irq.o
  Compiling out/src/stm32/stm32f0_timer.o
  Compiling out/src/stm32/stm32f0.o
  Compiling out/src/stm32/gpioperiph.o
  Compiling out/src/stm32/stm32f0_adc.o
  Compiling out/src/stm32/stm32f0_i2c.o
  Compiling out/src/stm32/spi.o
  Compiling out/src/stm32/can.o
  Compiling out/src/../lib/fast-hash/fasthash.o
  Compiling out/src/generic/canbus.o
  Building out/compile_time_request.o
Version: v0.10.0-408-g2c441b45
  Preprocessing out/src/generic/armcm_link.ld
  Linking out/klipper.elf
  Creating hex file out/klipper.bin
fly@flygemini:~/klipper$
```

-
3.
4.
5.
6.

```
fly@flygemini:~$ cd klipper
fly@flygemini:~/klipper$ sudo dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08002000 -D out/klipper.bin
[sudo] password for fly:
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuERROR, status = 10
dfuERROR, clearing status
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
DfuSe interface name: "Internal Flash  "
Downloading to address = 0x08002000, size = 23052
Download        [=========================] 100%        23052 bytes
Download done.
File downloaded successfully
```

7.

8.
9.

-
-
-

```
fly@flygemini:~/klipper$ python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: fea6a45462e9, Application: Klipper
Query Complete
```

SB2040&ERCF

1. First compile the Klipper firmware and configure it as shown below

```
(Top)
                                                    Klipper Firmware
[*] 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)
(500000) CAN bus speed
(gpio24) GPIO pins to set at micro-controller startup
```

- After configuring as shown in the picture above, press the "Q" key, and then press the "Y" key to exit and save.

2. Compile klipper firmware

make clean
make -J4

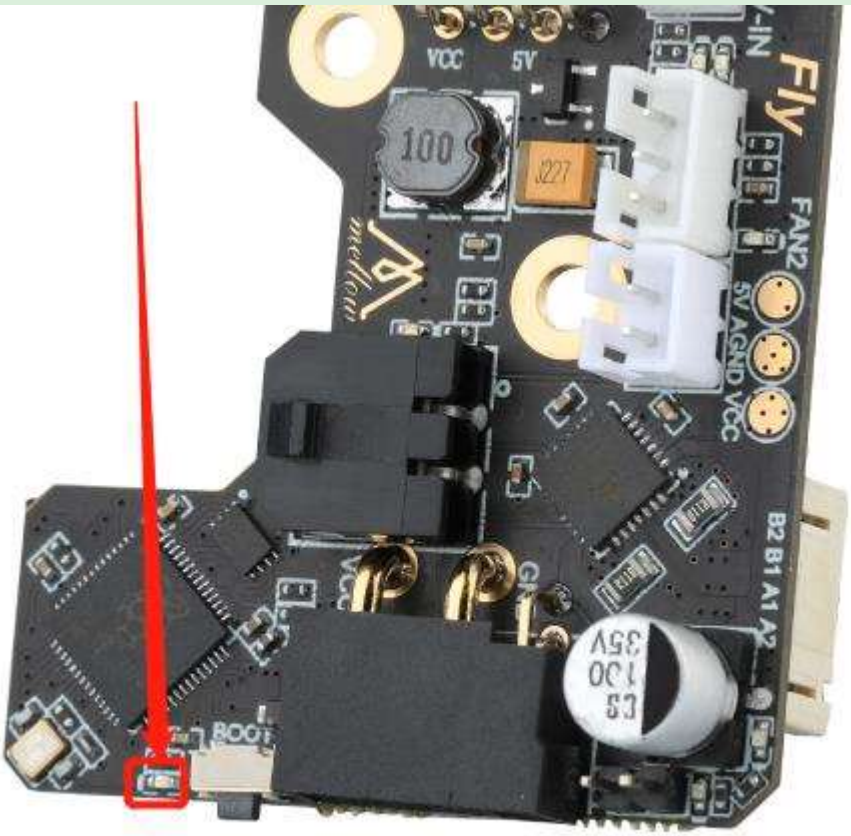
```
  Compiling out/src/buttons.o
  Compiling out/src/tmcuart.o
  Compiling out/src/neopixel.o
  Compiling out/src/pulse_counter.o
  Compiling out/src/stm32/watchdog.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/generic/crc16_ccitt.o
  Compiling out/src/generic/armcm_boot.o
  Compiling out/src/generic/armcm_irq.o
  Compiling out/src/generic/armcm_reset.o
  Compiling out/src/../lib/stm32f0/system_stm32f0xx.o
  Compiling out/src/generic/timer_irq.o
  Compiling out/src/stm32/stm32f0_timer.o
  Compiling out/src/stm32/stm32f0.o
  Compiling out/src/stm32/gpioperiph.o
  Compiling out/src/stm32/stm32f0_adc.o
  Compiling out/src/stm32/stm32f0_i2c.o
  Compiling out/src/stm32/spi.o
  Compiling out/src/stm32/can.o
  Compiling out/src/../lib/fast-hash/fasthash.o
  Compiling out/src/generic/canbus.o
  Building out/compile_time_request.o
Version: v0.10.0-408-g2c441b45
  Preprocessing out/src/generic/armcm_link.ld
  Linking out/klipper.elf
  Creating hex file out/klipper.bin
fly@flygemini:~/klipper$
```

- If it appears like the picture above, `Creating hex file out/klipper.bin`the compilation is successful.

3. After connecting the wires, it is recommended to power off the entire machine and power it on again. Then you can use CanBoot to program the firmware. First enter the following command

Tip

After power on, this status light should flash at a certain frequency!!! If it does not flash, please re-burn the CanBoot boot firmware! !



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

The highlighted part in the picture below `365f54003b9d`is the uuid of this SHT. This uuid is different for each board. The uuid will not change after burning firmware on the same SB2040 board.

```
fly@um:~/klipper$ python3 ~/klipper/lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 365f54003b9d, Application: CanBoot
Query Complete
```

Tip

If you cannot find the CAN ID, please check:

- Whether the wiring is correct, for example, whether CANH and CANL are connected reversely or have poor contact
- Is the 120Ω jumper cap on the SB2040 board plugged in?
- Does your image kernel support CAN?
- Check whether CanBoot is compiled correctly. If there are no errors, you can try to flash CanBoot again. Please refer to the flashing steps: SB2040 Firmware Flashing
- If the ID still cannot be found, you can use the host computer to program the firmware again.

4. Burn Klipper firmware via CANBUS

- The following commands fea6a45462e9need to be replaced with the UUID you got in the previous step.

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

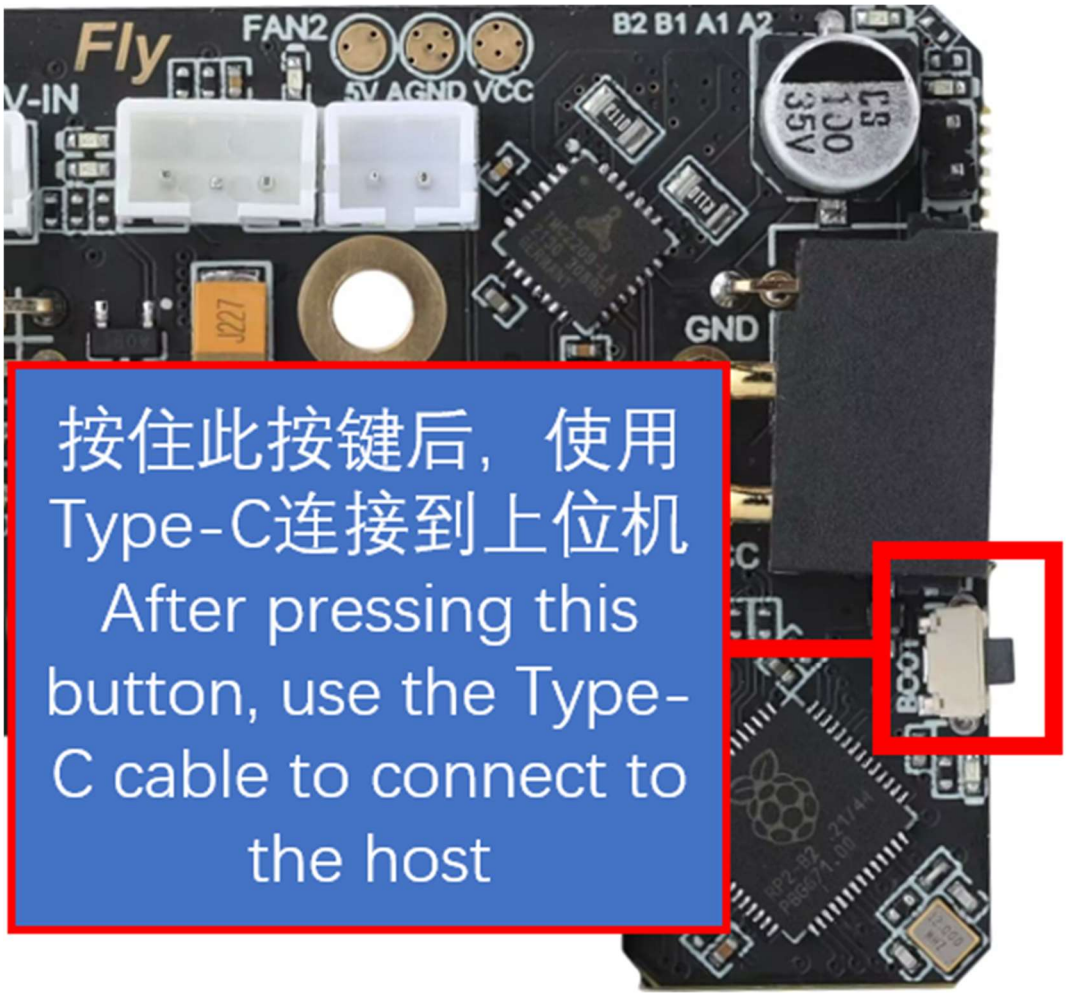
```
fly@flygemini:~/klipper$ python3 lib/canboot/flash_can.py -i can0 -f ./out/klipper.bin -u fea6a45462e9
Sending bootloader jump command...
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: fea6a45462e9, Application: CanBoot
Attempting to connect to bootloader
CanBoot Connected
Protocol Version: 1.0.0
Block Size: 64 bytes
Application Start: 0x8002000
MCU type: stm32f072xb
Verifying canbus connection
Flashing '/home/fly/klipper/out/klipper.bin'...

[##################################################]

Write complete: 12 pages
Verifying (block count = 361)...

[##################################################]

Verification Complete: SHA = E8B616ABAB17779CBAC3177A4F6DAEA36FBF826D
CAN Flash Success
fly@flygemini:~/klipper$
```

- If it appears in the picture above CAN Flash Success, it means the burning is successful.

Tip

If you have burned CanBoot multiple times and still cannot find the CanBoot ID, you can use the following method to burn the firmware:

1. Check whether it is connected to the BOOT burning mode of SB2040
   Press and hold the BOOT button of the SB2040 board, and then connect the USB to the host computer



lsusb

Execute the above command to see if ID 2e8a:0003 Raspberry Pi RP2 Bootthis line exists. If not, please check the USB cable (remember to hold down the BOOT key before connecting)

```
fly@flygemini:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 002: ID 2e8a:0003 Raspberry Pi RP2 Boot
Bus 007 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 006 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 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 1b3f:1167 Generalplus Technology Inc. WEB CAM
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

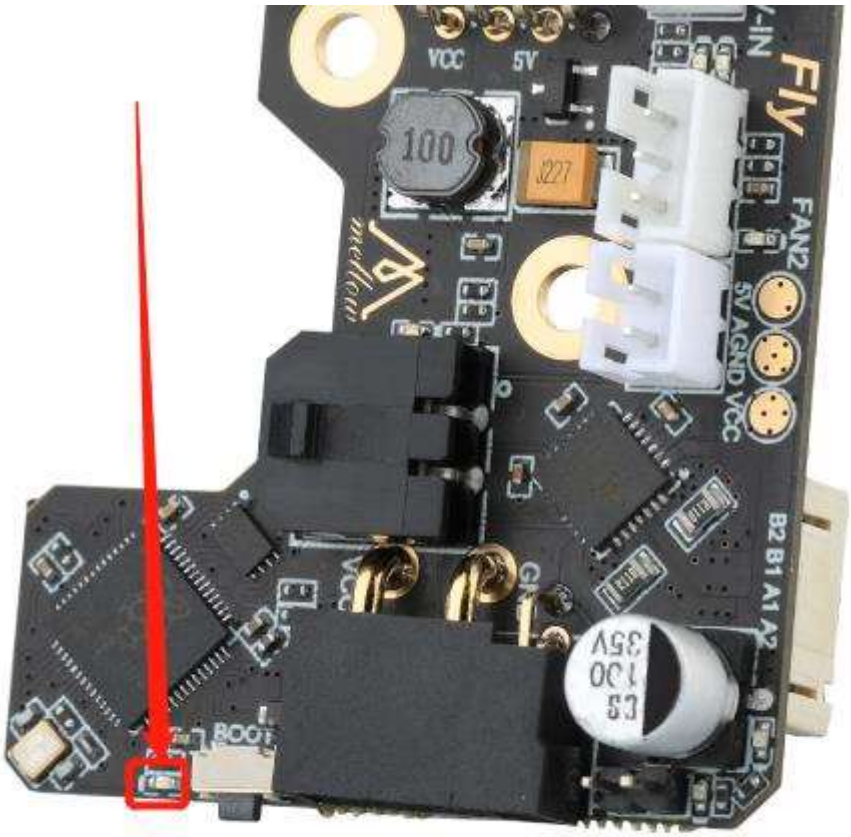
2. Burn
3. cd ~/klipper/
   make flash FLASH_DEVICE=2e8a:0003
   Executing the above command may prompt you to enter a password. Just enter the password of the current user. The password will not be visible when you enter it. After typing, press Enter

If the following picture appears, the burning is successful.

```
fly@flygemini:~/klipper$ make flash FLASH_DEVICE=2e8a:0003
  Building rp2040_flash
gcc -c -Wall -ggdb -I../rp2040/ `pkg-config libusb-1.0 --cflags` main.c -o main.o
gcc -c -Wall -ggdb -I../rp2040/ `pkg-config libusb-1.0 --cflags` picoboot_connection.c
gcc  main.o picoboot_connection.o `pkg-config libusb-1.0 --libs` -o rp2040_flash
  Flashing out/klipper.uf2 to 2e8a:0003
sudo lib/rp2040_flash/rp2040_flash out/klipper.uf2

[sudo] password for fly:
Loaded UF2 image with 114 pages
Found rp2040 device on USB bus 7 address 2
Flashing...
Resetting interface
Locking
Exiting XIP mode
Erasing
Flashing
Rebooting device
fly@flygemini:~/klipper$
```

3. examine

If the configuration is compiled correctly and the programming is successful, the light on the SB2040 board will be always on! ! !



FLY-SHT36 V2

1.

•

```
(Top)
                                                Klipper Firmware Configuration
[*] Enable extra low-level configuration options
    Micro-controller Architecture (STMicroelectronics STM32)  --->
    Processor model (STM32F103)  --->
[ ] Only 10KiB of RAM (for rare stm32f103x6 variant) (NEW)
[ ] Disable SWD at startup (for GigaDevice stm32f103 clones) (NEW)
    Bootloader offset (8KiB bootloader)  --->
    Clock Reference (8 MHz crystal)  --->
    Communication interface (CAN bus (on PB8/PB9))  --->
(500000) CAN bus speed
(!PC13) GPIO pins to set at micro-controller startup
```

•

```
(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

[Space/Enter] Toggle/enter      [?] Help            [/] Search
[Q] Quit (prompts for save)     [ESC] Leave menu
```

•

2.

```
  Compiling out/src/buttons.o
  Compiling out/src/tmcuart.o
  Compiling out/src/neopixel.o
  Compiling out/src/pulse_counter.o
  Compiling out/src/stm32/watchdog.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/generic/crc16_ccitt.o
  Compiling out/src/generic/armcm_boot.o
  Compiling out/src/generic/armcm_irq.o
  Compiling out/src/generic/armcm_reset.o
  Compiling out/src/../lib/stm32f0/system_stm32f0xx.o
  Compiling out/src/generic/timer_irq.o
  Compiling out/src/stm32/stm32f0_timer.o
  Compiling out/src/stm32/stm32f0.o
  Compiling out/src/stm32/gpioperiph.o
  Compiling out/src/stm32/stm32f0_adc.o
  Compiling out/src/stm32/stm32f0_i2c.o
  Compiling out/src/stm32/spi.o
  Compiling out/src/stm32/can.o
  Compiling out/src/../lib/fast-hash/fasthash.o
  Compiling out/src/generic/canbus.o
  Building out/compile_time_request.o
Version: v0.10.0-408-g2c441b45
  Preprocessing out/src/generic/armcm_link.ld
  Linking out/klipper.elf
  Creating hex file out/klipper.bin
fly@flygemini:~/klipper$
```

-
3.

```
fly@um:~/klipper$ python3 ~/klipper/lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 365f54003b9d, Application: CanBoot
Query Complete
```

-
-
-
-

```
fly@flygemini:~/klipper$ python3 lib/canboot/flash_can.py -i can0 -f ./out/klipper.bin -u fea6a45462e9
Sending bootloader jump command...
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: fea6a45462e9, Application: CanBoot
Attempting to connect to bootloader
CanBoot Connected
Protocol Version: 1.0.0
Block Size: 64 bytes
Application Start: 0x8002000
MCU type: stm32f072xb
Verifying canbus connection
Flashing '/home/fly/klipper/out/klipper.bin'...

[##################################################]

Write complete: 12 pages
Verifying (block count = 361)...

[##################################################]

Verification Complete: SHA = E8B616ABAB17779CBAC3177A4F6DAEA36FBF826D
CAN Flash Success
fly@flygemini:~/klipper$
```

-

4. The firmware with CanBoot has been burned

If you have already burned **Canboot boot firmware** and **firmware with Canboot** , you need to update the klipper firmware in the future. Just follow the steps below.

- Pull the latest klipper

```
cd ~/klipper
git pull
```

- Compile the latest klipper

```
make menuconfig
make clean
make -J4
```

- Burn klipper for SHT36/42
- In the following command, fea6a45462e9you need to replace it with the UUID you queried.

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