

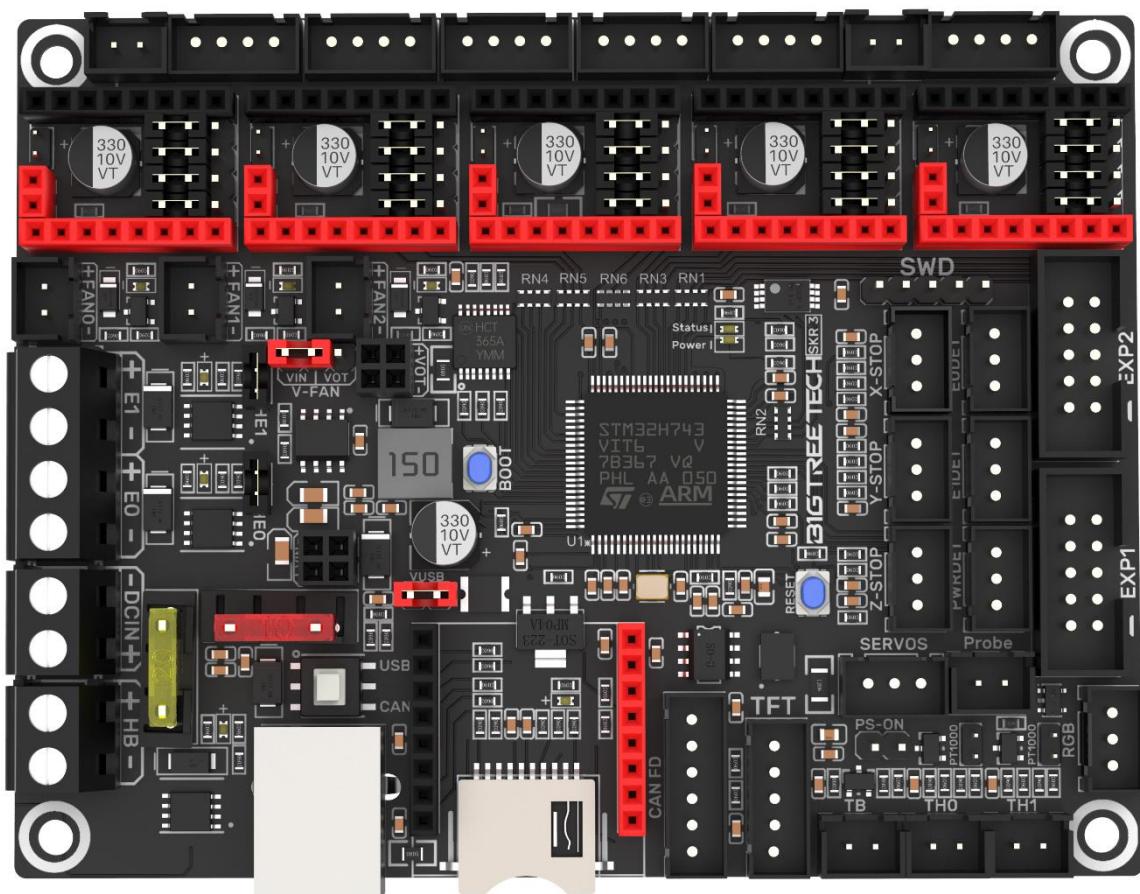
深圳市必趣科技有限公司

BIGTREETECH

BIGTREETECH

SKR 3

User manual



Content

Content	2
Revision History	5
1. Introduction	6
1. 1 Features	6
1. 2 Specification	7
1. 3 Firmware	8
1. 4 Dimensions	8
2. Peripheral Port	9
2. 1 Connector diagrams	9
2. 2 Pinout diagram	9
3. Connection description	10
3. 1 USB power supply	10
3. 2 Stepper driver	10
3. 2. 1 STEP/DIR(STANDALONE) mode	10
3. 2. 2 UART mode of TMC driver	12
3. 2. 3 TMC driver SPI mode	12
3. 2. 4 TMC driver DIAG(Sensorless Homing)	12
3. 3 USB and CAN mode	13
3. 4 Voltage selection for CNC Fan	13
3. 5 100K NTC or PT1000 setting	14
3. 6 BLTouch wiring	14
3. 7 Auto power off (Relay V1.2) wiring	15
3. 8 Power loss recovery (UPS 24V V1.0) wiring	15
3. 9 RGB wiring	16
3. 10 Filament sensor wiring	16
3. 11 Display wiring	17

Shenzhen Big Tree Technology CO.,LTD .

BIG TREE TECH

4. Marlin	18
4.1 install compiling environment	18
4.2 Download Marlin firmware	18
4.3 Configure firmware	18
4.3.1 Open Marlin project	18
4.3.2 Compiling environment	18
4.3.3 Configure motherboard and serial port	19
4.3.4 Configure stepper driver	20
4.3.5 Sensorless homing	21
4.3.6 100K NTC or PT1000	22
4.3.7 BLTouch	22
4.3.8 Auto power off(Relay V1.2)	25
4.3.9 Power loss recovery	25
4.3.10 RGB	26
4.3.11 Filament sensor	27
4.3.12 smart filament sensor(SFS V1.0 / V2.0)	28
4.3.13 ESP3D	29
4.4 Compile firmware	30
5. Klipper	31
5.1 Preparation	31
5.1.1 Download OS image	31
5.1.2 Download and install Raspberry Pi Imager	31
5.2 Write image	32
5.3 WIFI setting	34
5.4 ssh connect to raspberry pi	34
5.5 Compile firmware	36
5.6 Configure Klipper	37
6. Firmware update	38

Shenzhen Big Tree Technology CO.,LTD .

BIG TREE TECH

7. Precautions	38
8. FAQ	39

Revision History

version	Note	Date
01.00	Original	2022/03/08
01.01	Add support for RRF	2022/05/21

1. Introduction

The BIGTREETECH SKR 3 mother board is a brand new 32 bit motherboard developed by Shenzhen Big Tree Technology Co., Ltd that fixed all the problems of the V1.4/Turbo board with new features and improved functionality.

1.1 Features

1. Utilize 32bit 480MHz ARM Cortex-M7 series STM32H743VI MCU for massively improved performance
2. TPS5450-5A power supply chip, support DC12/24V power input, current output rated at 5A max continuous and 6A max instantaneous, sufficient power supply for Raspberry pi(3A requirement)
3. Onboard BOOT button to enable DFU mode to update bootloader
4. Thermistor circuit is protected to prevent MCU damage from shorted heatbed and heater cartridge connection.
5. Selectable voltage (24V, 12V, 5V by **SKR 3-DC MODE**) for CNC fan (**Note:** The voltages of the 3 * CNC fans are unified, different voltages cannot be set separately for different ports), no more need for external stepdown thus prevent board damage from user error
6. Thermistor connection supports Pull up resistance value setting using jumpers, No more extra module needed for PT1000
7. Supports all models of Serial TFT, SPI TFT, and LCD12864/2004 from BTT
8. Firmware installation from using MicroSD card, simple and efficient
9. Integrated SPI and UART mode of TMC driver and DIAG pin, easily configurable with jumpers
10. Supports power loss recovery, filament runout sensor, Auto poweroff, BLTouch, RGB led etc.
11. High efficiency MOSFET for less heat generation
12. Replaceable fuse for easy maintenance
13. WIFI module (ESP-12S、ESP-07、ESP32) supported
14. Onboard SDIO MicroSD slot for Higher transfer rate
15. Onboard EEPROM for setting storage

16. 2 method for CAN connection:USB & XH2.54 6Pin, USB port CAN/USB mode is selected by switches.

1.2 Specification

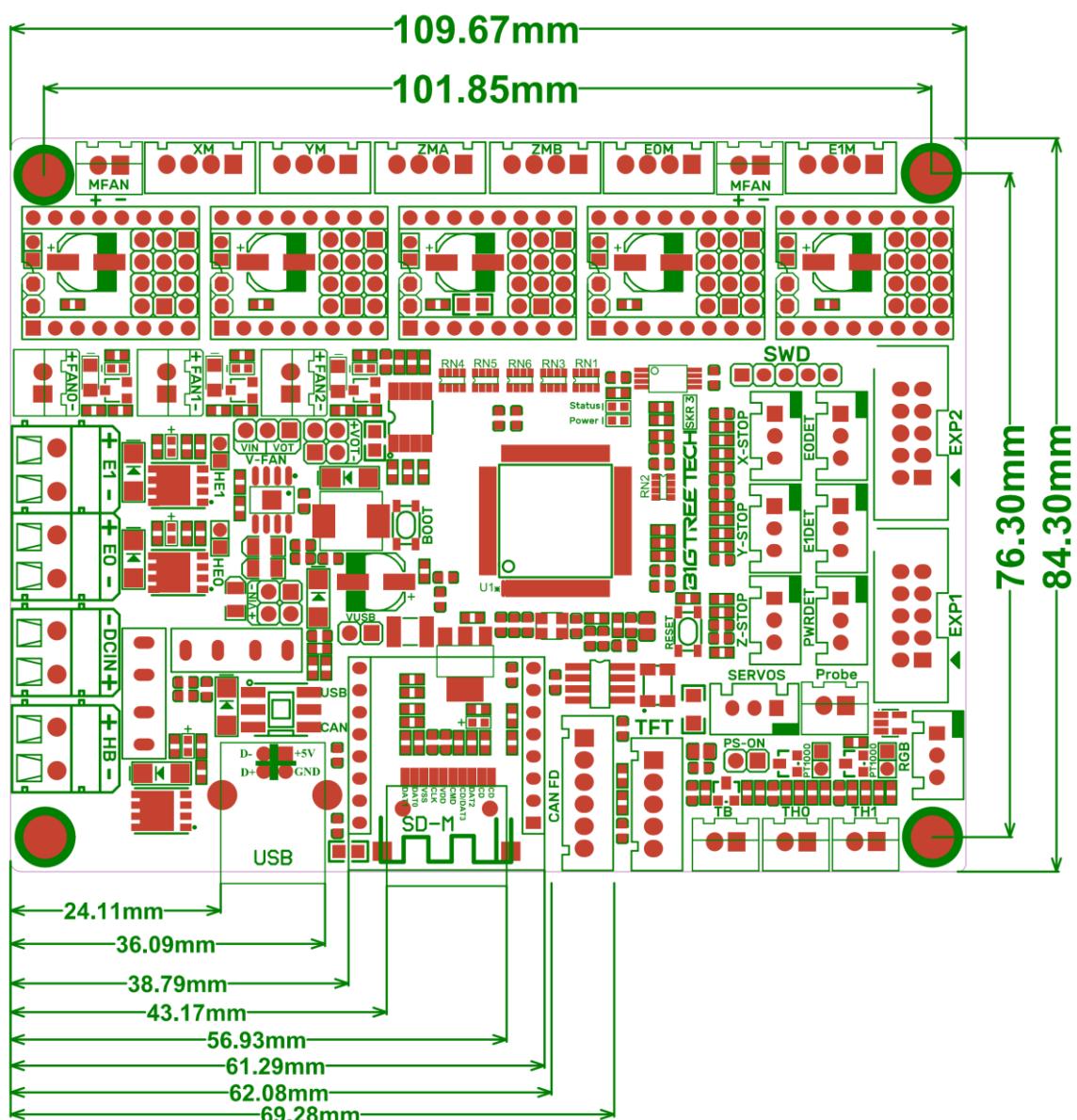
1. Dimention: 110*85mm, for details please refer to: **BIGTREETECH SKR 3-SIZE.pdf**
2. Mounting pattern: 102*76mm
3. MCU: ARM Cortex-M7 STM32H743VI
4. EEPROM: 24C32 32Kbit
5. Voltage in: DC12V-DC24V
6. Logic voltage: DC 3.3V
7. Heater connection: Heated bed (HB)、heater cartridge (E0、E1)
8. HB port max current: 10A continuous , 11A instantaneous
9. Heater cartridge max current: 5.5A continuous, 6A instantaneous
10. Fan port: 3 X CNC(selectable voltage) ,2 X always on(PSU voltage)
11. Fan port max current: 1A continuous, 1.5A instantaneous
12. Overall max current (HB+E0+E0+All fans):10A
13. WIFI connection: ESP-12S、ESP-07S、ESP32
14. Expansion port: BLTouch (Servos、Probe)、PS-ON、PWR-DET、Fil-DET、RGB、CAN FD
15. Steppers supported: TMC5160、TMC2209、TMC2225、TMC2226、TMC2208、TMC2130、ST820、LV8729、DRV8825、A4988 etc.
16. Stepper driver mode: SPI、UART、STEP/DIR
17. Stepper motor socket: X、Y、Z (双 Z 轴)、E0、E1 5 channels in total
18. Thermistor: 2 X thermistor port(NTC/PTC selectable)
19. Display: Serial、SPI、LCD
20. PC connection: USB A
21. Supported file format: G-code
22. Supported kinematics: Cartesian、Delta、Kossel、Ultimaker、CoreXY

23. Recommended Slicer/Console: Cura、Simplify3D、Pronterface、Repetier-host、Makerware

1.3 Firmware

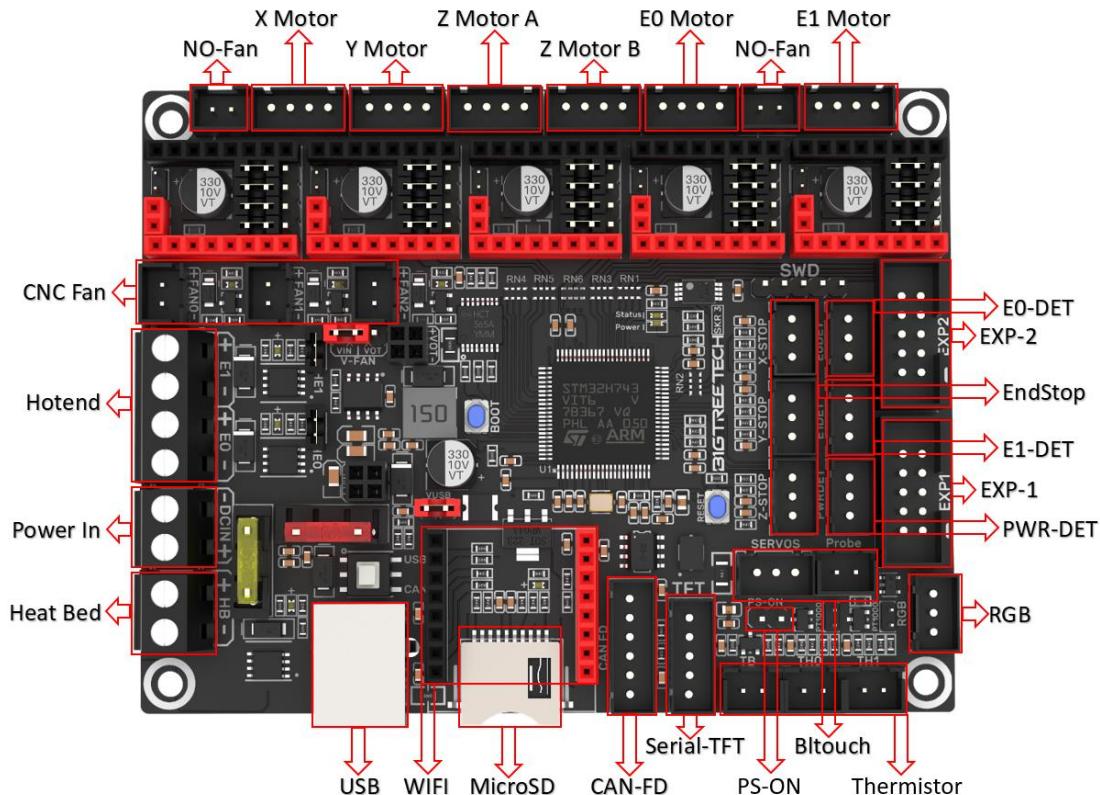
Supported Firmware: Marlin, Klipper, RRF

1.4 Dimensions

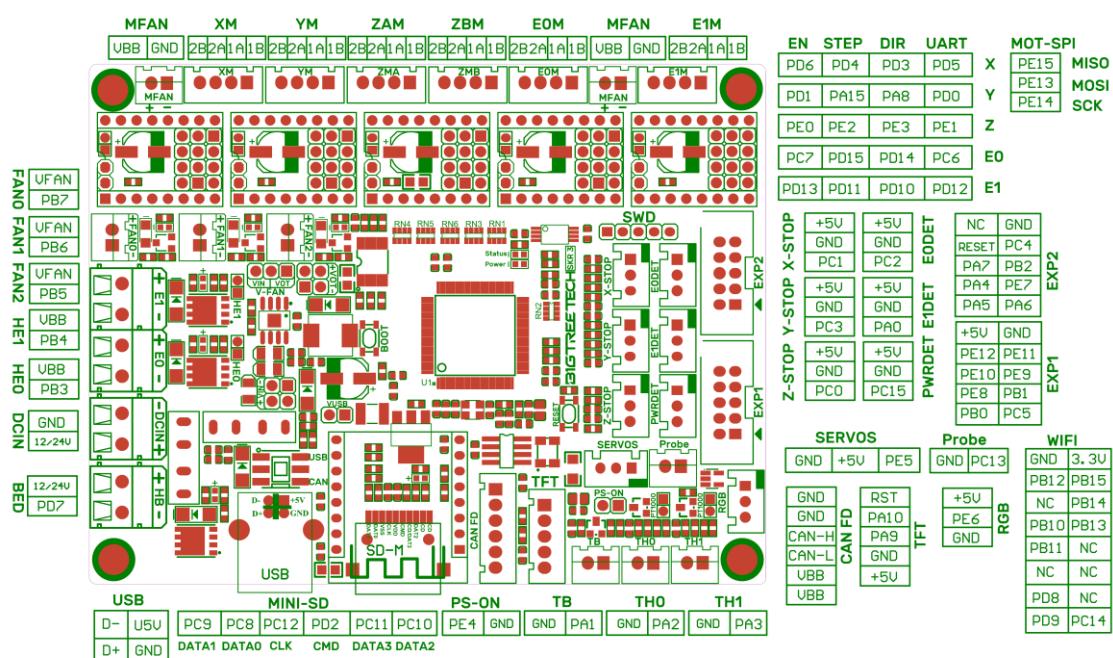


2. Peripheral Port

2.1 Connector diagrams



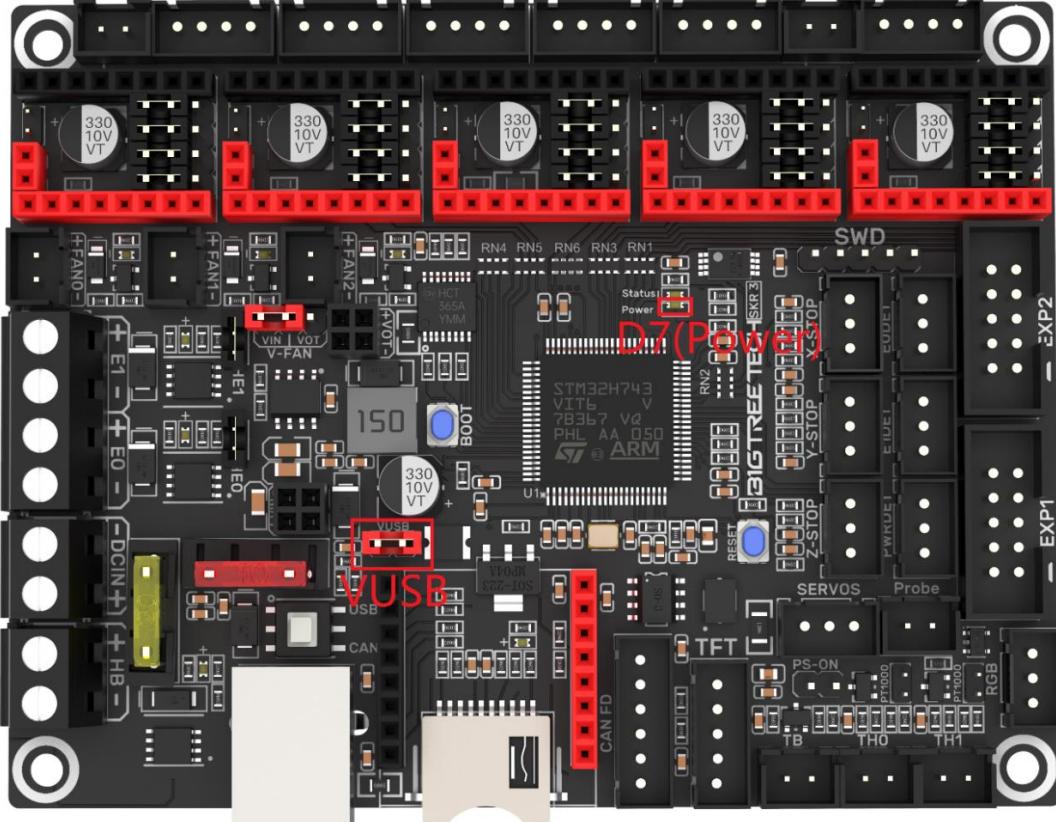
2.2 Pinout diagram



3. Connection description

3.1 USB power supply

After the SKR 3 board has been powered, the Red led D7(Power) to the upper right of the MCU will light up, indicating power on. When using only USB to power the board, Please insert the jumper cap onto the VUSB jumper.

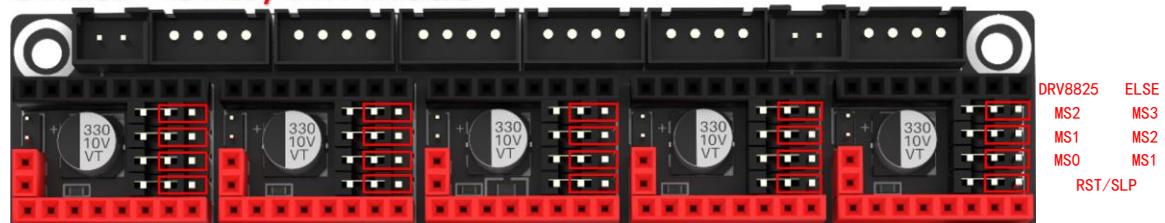


3.2 Stepper driver

3.2.1 STEP/DIR(STANDALONE) mode

i.e: A4988、DRV8825、LV8729、ST820 etc, connect jumpers(MS0~MS2) according to the microstep chart

Driver – STEP/DIR Mode



Note: RST and SLP must be shorted by jumpers for A4988 or DRV8825.

Shenzhen Big Tree Technology CO.,LTD .

BIG TREE TECH

Chips:	MS1	MS2	MS3	microstep	Excitation Mode
A4988 16 microstep max 35V 2A	L	L	L	Full Step	2 Phase
	H	L	L	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	H	H	L	1/8	2W1-2 Phase
	H	H	H	1/16	4W1-2 Phase
Current $R_S=0.1\Omega$	$I_{TripMAX} = \frac{V_{REF}}{8 * R_S}$				

Driver chips	MODE2	MODE1	MODE0	Microsteps	Excitation Mode
DRV8825 Maximum 32microsteps 8.2V-45V 2.5A at 24V T=25°C	L	L	L	Full Step	2 Phase
	L	L	H	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	L	H	H	1/8	
	H	L	L	1/16	
	H	L	H	1/32	
	H	H	L	1/32	
	H	H	H	1/32	
Current $R_{ISENSE}=0.1\Omega$	$I_{CHOP} = \frac{V_{(xREF)}}{5 * R_{ISENSE}}$				

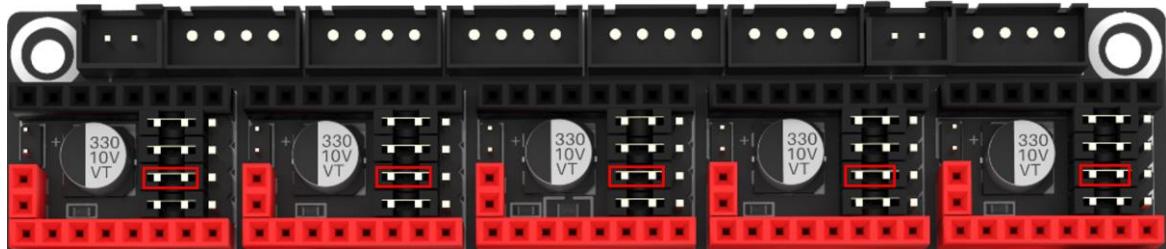
Driver chips	MD3	MD2	MD1	Microsteps	Excitation Mode
LV8729 Maximum 128microsteps 36V 1.8A	L	L	L	Full Step	2 Phase
	L	L	H	1/2	1-2 Phase
	L	H	L	1/4	W1-2 Phase
	L	H	H	1/8	2W1-2 Phase
	H	L	L	1/16	4W1-2 Phase
	H	L	H	1/32	8W1-2 Phase
	H	H	L	1/64	16W1-2 Phase
	H	H	H	1/128	32W1-2 Phase
Current $RF1=0.22\Omega$	$I_{OUT} = (V_{REF} / 5) / RF1$				

Driver chips	MS3	MS2	MS1	Microsteps
ST820 Maximum 256microsteps 45V 1.5A	L	L	L	Full Step
	L	L	H	1/2
	L	H	L	1/4
	L	H	H	1/8
	H	L	L	1/16
	H	L	H	1/32
	H	H	L	1/128
	H	H	H	1/256
Current $R_s=0.15\Omega$	$I_{peak} = \frac{V_{REF} * V_{DD}}{5 * R_s}$			

3. 2. 2 UART mode of TMC driver

i. e: TMC2208、TMC2209、TMC2225 etc. Place jumpers according to the diagram below, microstep and current can be configured in firmware.

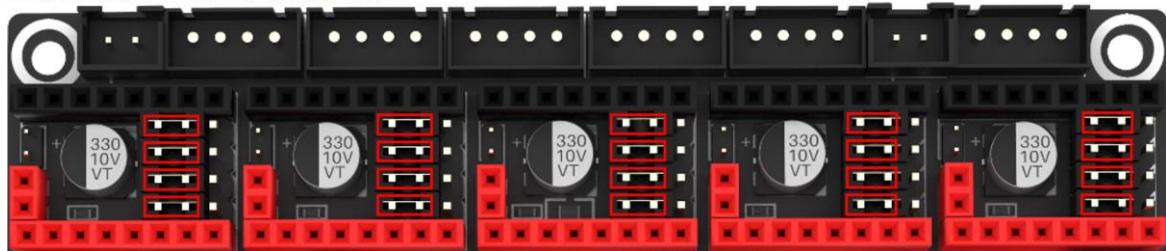
Driver – UART Mode



3. 2. 3 TMC driver SPI mode

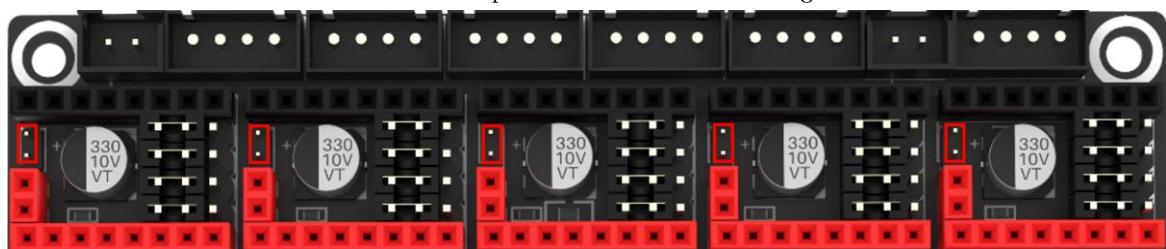
i. e: TMC2130、TMC5160、TMC5161 et.c, Place jumpers according to the diagram below, microstep and current can be configured in firmware.

Driver – SPI Mode



3. 2. 4 TMC driver DIAG(Sensorless Homing)

When using sensorless homing, place jumpers according to the diagram below, there is no need to cut the DIAG pin off when not being used.



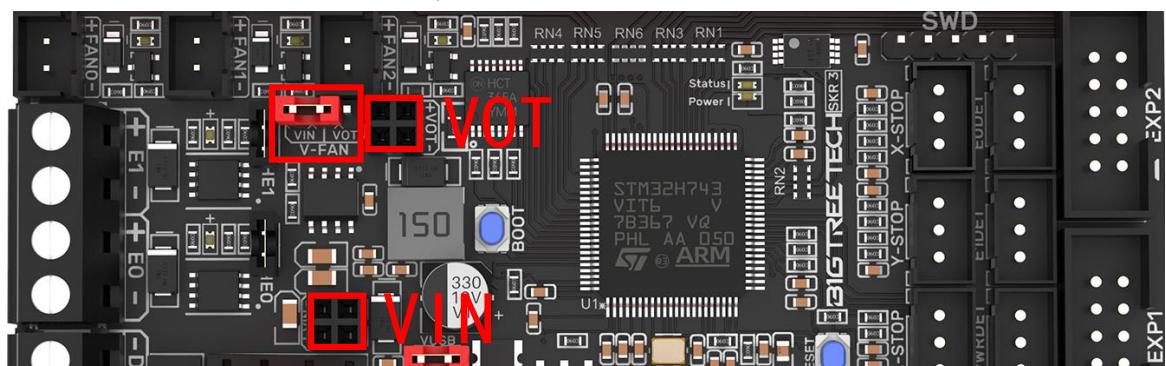
3.3 USB and CAN mode

When the button shown below is released, the board is in usb mode, when pressed down, the board is in CAN FD mode.

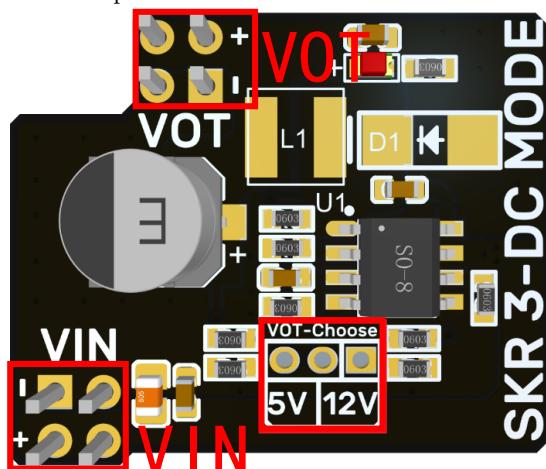


3.4 Voltage selection for CNC Fan

Connect jumper between the two pins of VIN if using DCIN as CNC fan voltage. Connect jumper between the two pins of VOT and insert SKR 3-DC MODE on to the 2*4 Pin sockets if 12V or 5V is desired (**Note:** The voltages of the 3*CNC fans are unified, different voltages cannot be set separately. i.e: the voltage of 3*CNC can be set to 24V, 12V or 5V at the same time, but it cannot be set to the combination of 24V+12V+5V) .

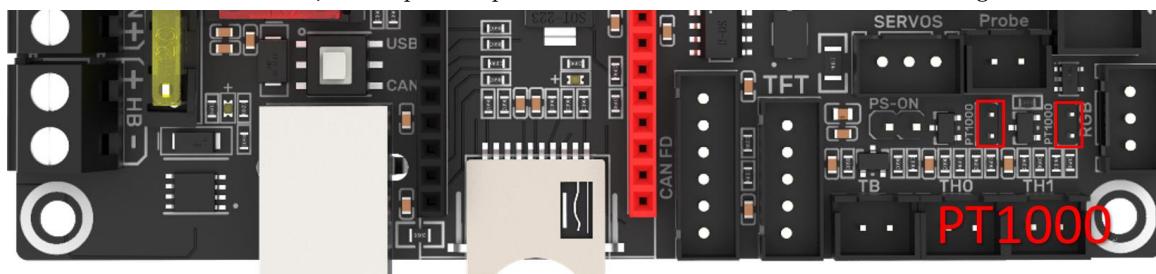


Voltage of the SKR 3-DC MODE is set by connecting jumpers on 5V or 12V on VOT- Choose pins.

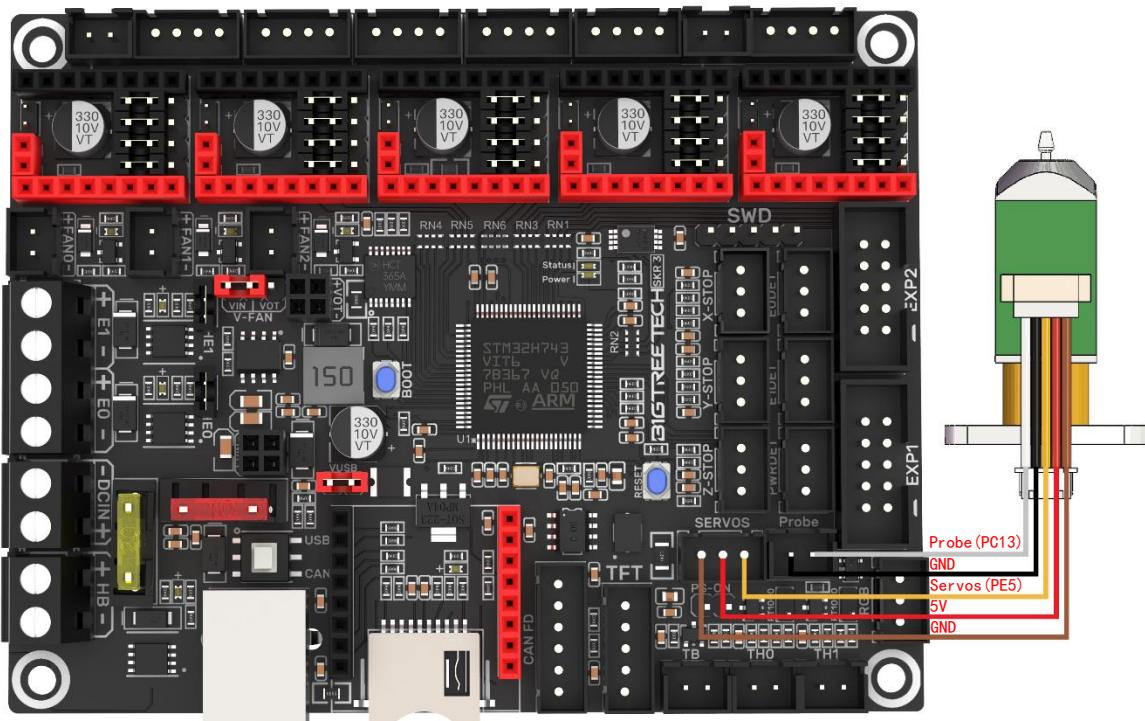


3.5 100K NTC or PT1000 setting

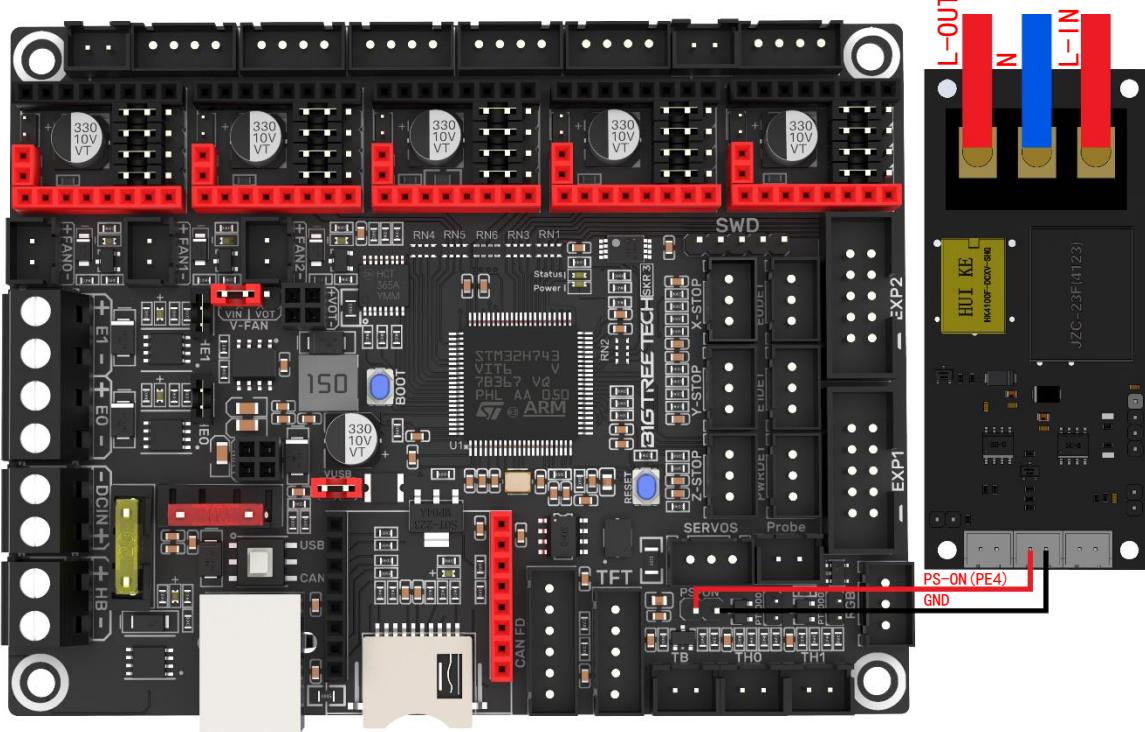
When using 100K NTC no jumpers need to be connected, the pull up resistance of TH0 & TH1 is 4.7K. When using PT1000 the jumpers indicated in the picture below needs to be connected, the pull up resistance of TH0 & TH1 is changed to 1K.



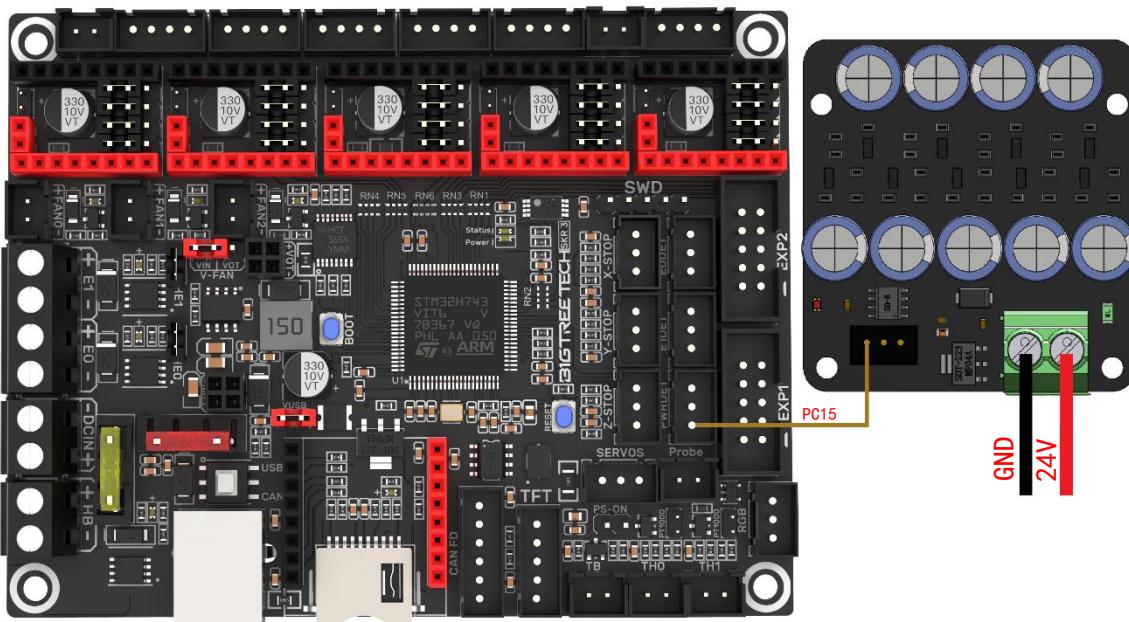
3.6 BLTouch wiring



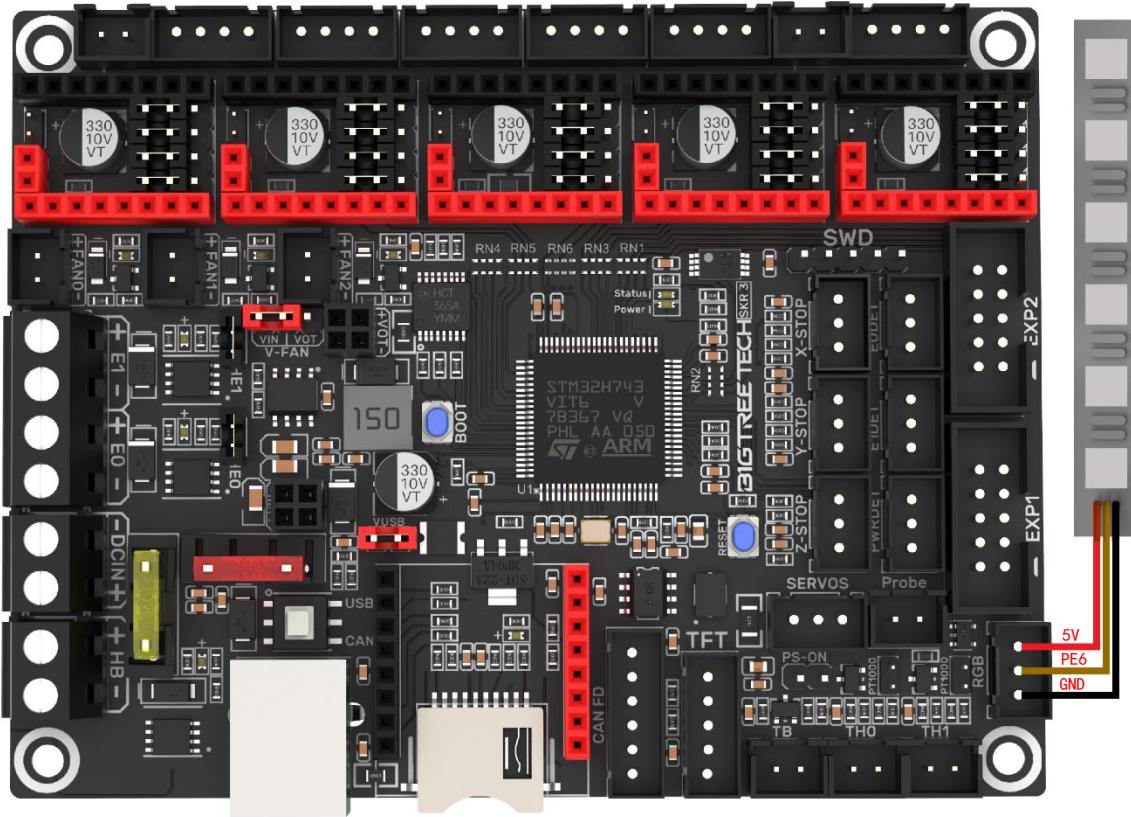
3.7 Auto power off (Relay V1.2) wiring



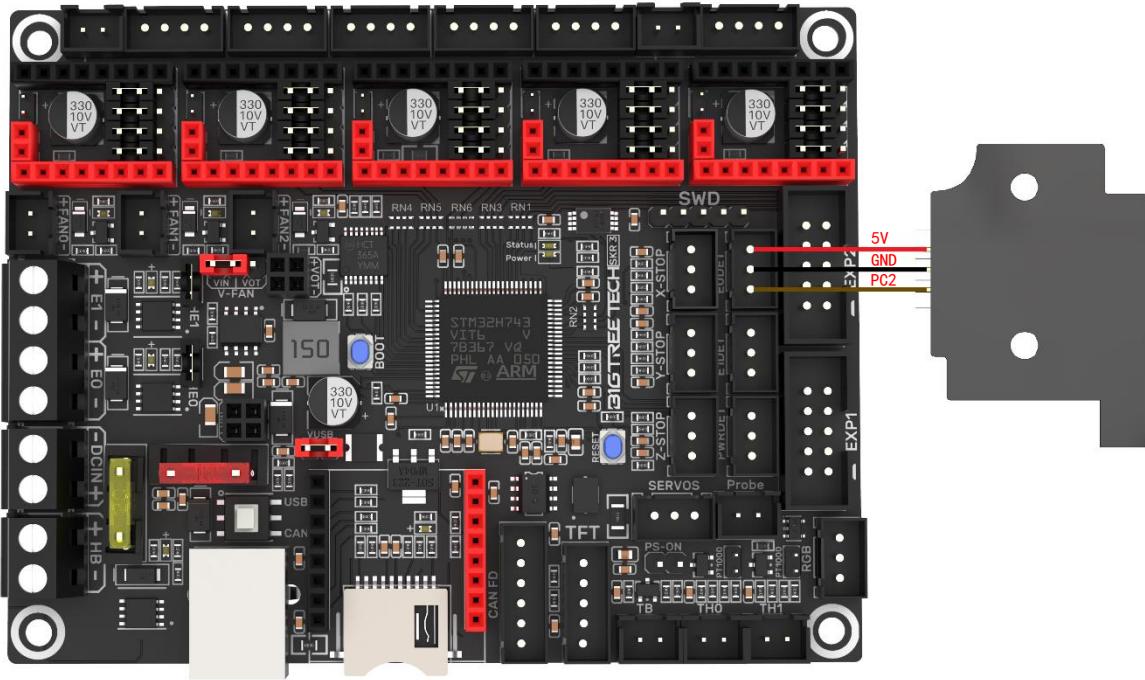
3.8 Power loss recovery (UPS 24V V1.0) wiring



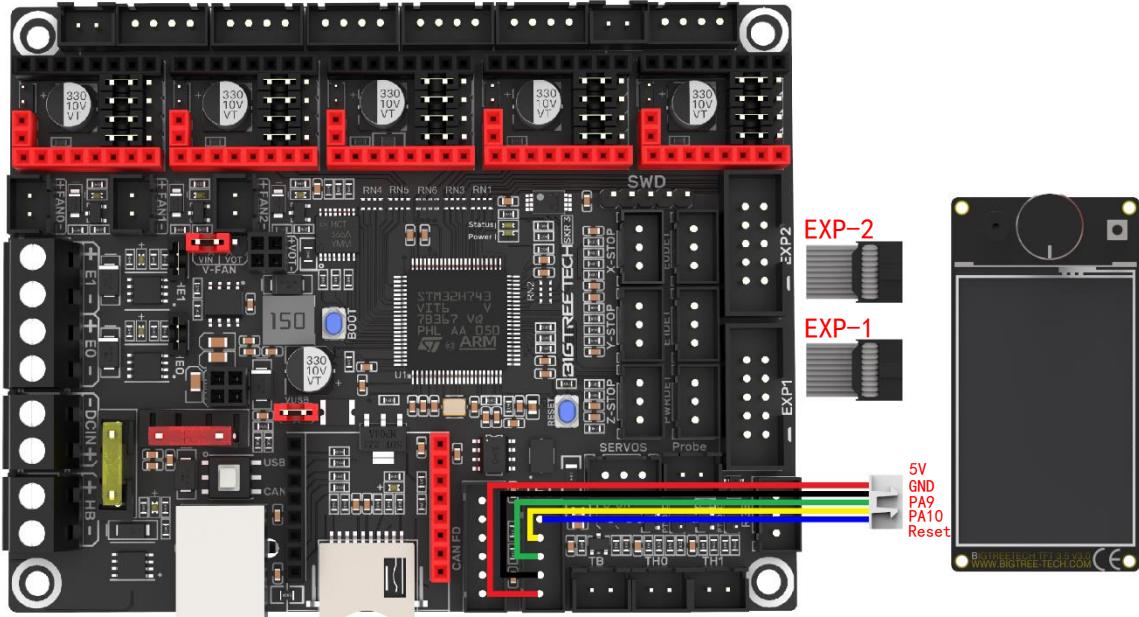
3.9 RGB wiring



3.10 Filament sensor wiring



3.11 Display wiring



4. Marlin

4.1 install compiling environment

<https://github.com/bigtreeTech/Document/blob/master/How%20to%20install%20VScode%2BPlatformio.md>
https://marlinfw.org/docs/basics/install_platformio_vscode.html

Refer to the link above for tutorial on installing VSCode and PlatformIO plugin

4.2 Download Marlin firmware

1. Download the newest bugfix version of Marlin from official website
<https://github.com/MarlinFirmware/Marlin/tree/bugfix-2.0.x>
2. Download Pre-configured firmware from our GitHub page
<https://github.com/bigtreeTech/SKR-3>

4.3 Configure firmware

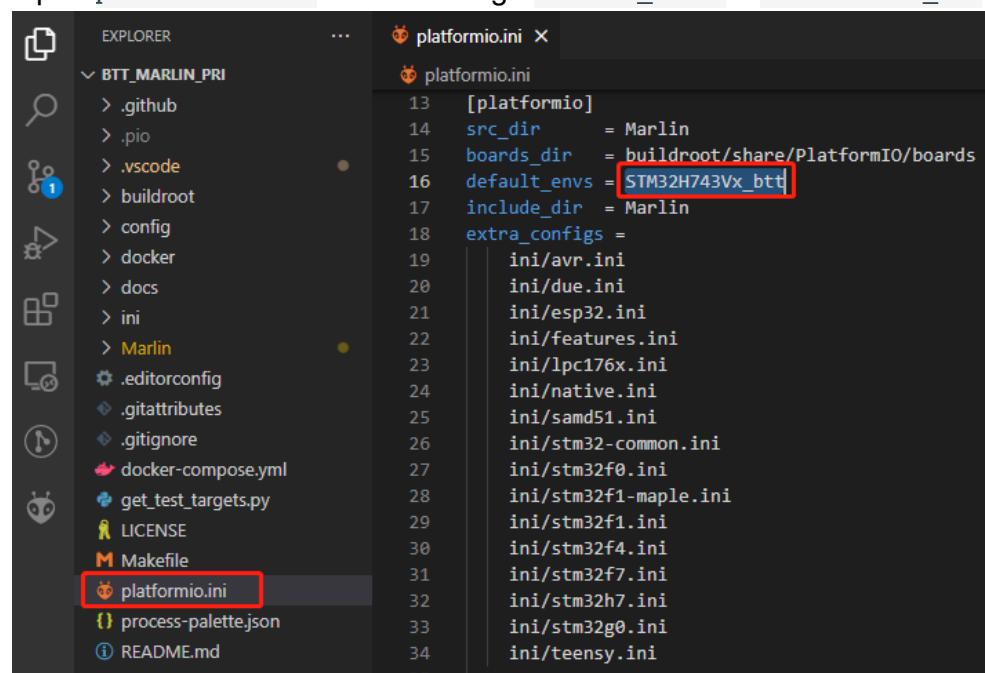
4.3.1 Open Marlin project

You can open Marlin in VS Code in one of several ways:

- Drag the downloaded Marlin Firmware folder onto the VScode application icon
- Use the **Open...** command in the VSCode **File** menu
- Open the PIO Home tab and click the “**Open Project**” button

4.3.2 Compiling environment

Open `platformio.ini` file and change `default_envs` to `STM32H743Vx_btt`



4. 3. 3 Configure motherboard and serial port

Set MOTHERBOARD to BOARD_BTT_SKR_3

Enable serial ports according to your needs

```
#define MOTHERBOARD BOARD_BTT_SKR_3
#define SERIAL_PORT 1      (enable TFT serial port)
#define BAUDRATE 115200    (set baudrate to the same as the communication
device)
#define SERIAL_PORT_2 -1   (enable USB serial port)
#define SERIAL_PORT_3 3     (enable WIFI serial port)
```

The screenshot shows a code editor interface with two files open: Configuration.h and Configuration_adv.h. The left sidebar displays a project structure with files like .github, .pio, .vscode, buildroot, config, docker, docs, ini, Marlin, lib, and src. The Configuration.h file is currently selected.

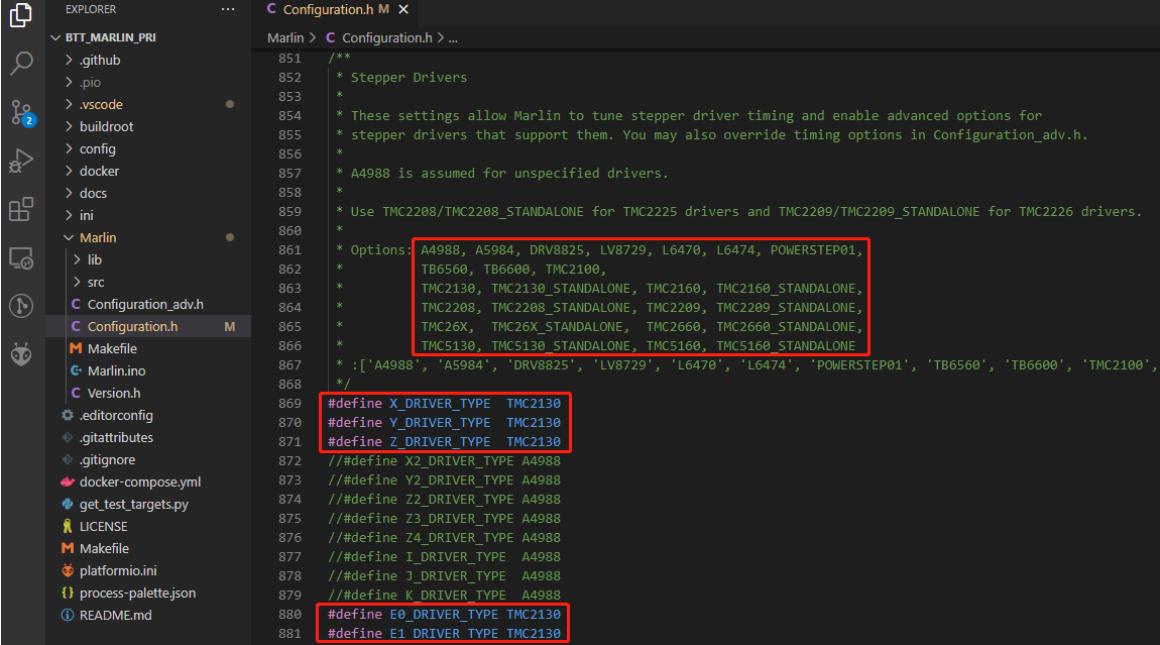
Configuration.h:

```
97 // Choose the name from boards.h that matches your setup
98 #ifndef MOTHERBOARD
99 | #define MOTHERBOARD BOARD_BTT_SKR_3
100#endif
101 /**
102 * Select the serial port on the board to use for communication with the host.
103 * This allows the connection of wireless adapters (for instance) to non-default port pins.
104 * Serial port -1 is the USB emulated serial port, if available.
105 * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
106 *
107 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
108 */
109 #define SERIAL_PORT 1
110
111 /**
112 * Serial Port Baud Rate
113 * This is the default communication speed for all serial ports.
114 * Set the baud rate defaults for additional serial ports below.
115 *
116 * 250000 works in most cases, but you might try a lower speed if
117 * you commonly experience drop-outs during host printing.
118 * You may try up to 1000000 to speed up SD file transfer.
119 *
120 * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
121 */
122 #define BAUDRATE 115200
123 //##define BAUD_RATE_GCODE // Enable G-code M575 to set the baud rate
124
125 /**
126 * Select a secondary serial port on the board to use for communication with the host.
127 * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
128 * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
129 */
130 #define SERIAL_PORT_2 -1
131 //##define BAUDRATE_2 250000 // Enable to override BAUDRATE
132
133 /**
134 * Select a third serial port on the board to use for communication with the host.
135 * Currently only supported for AVR, DUE, LPC1768/9 and STM32/STM32F1
136 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
137 */
138 #define SERIAL_PORT_3 3
139 //##define BAUDRATE_3 250000 // Enable to override BAUDRATE
140
141
```

Configuration_adv.h:

```
1 /**
2 * Select the secondary serial port on the board to use for communication with the host.
3 * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
4 * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
5 */
6 #define SERIAL_PORT_2 -1
7 //##define BAUDRATE_2 250000 // Enable to override BAUDRATE
8
9 /**
10 * Select a third serial port on the board to use for communication with the host.
11 * Currently only supported for AVR, DUE, LPC1768/9 and STM32/STM32F1
12 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
13 */
14 #define SERIAL_PORT_3 3
15 //##define BAUDRATE_3 250000 // Enable to override BAUDRATE
16
17
```

4. 3. 4 Configure stepper driver



The screenshot shows the VS Code interface with the 'Configuration.h' file open in the editor. The 'EXPLORER' sidebar on the left shows the project structure, including 'BTT_MARLIN_PRI' and 'Marlin' directories containing various configuration files like 'Configuration_adv.h', 'Configuration.h', and 'Version.h'. The 'Marlin' directory is currently selected. The 'Configuration.h' file in the editor has several lines of code highlighted with a red box:

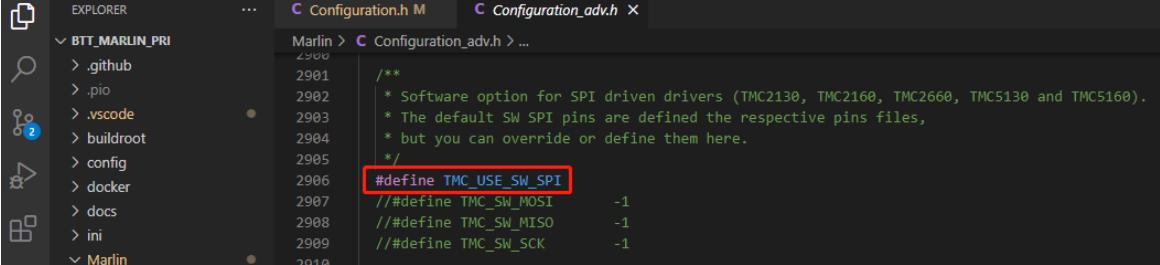
```

851 /**
852 * Stepper Drivers
853 *
854 * These settings allow Marlin to tune stepper driver timing and enable advanced options for
855 * stepper drivers that support them. You may also override timing options in Configuration_adv.h.
856 *
857 * A4988 is assumed for unspecified drivers.
858 *
859 * Use TMC2208/TMC2208_STANDALONE for TMC2225 drivers and TMC2209/TMC2209_STANDALONE for TMC2226 drivers.
860 *
861 * Options: A4988, A5984, DRV8825, LV8729, L6470, L6474, POWERSTEP01,
862 *           TB6560, TB6600, TMC2100,
863 *           TMC2130, TMC2130_STANDALONE, TMC2160, TMC2160_STANDALONE,
864 *           TMC2208, TMC2208_STANDALONE, TMC2209, TMC2209_STANDALONE,
865 *           TMC26X, TMC26X_STANDALONE, TMC2660, TMC2660_STANDALONE,
866 *           TMC5130, TMC5130_STANDALONE, TMC5160, TMC5160_STANDALONE
867 *           :['A4988', 'A5984', 'DRV8825', 'LV8729', 'L6470', 'L6474', 'POWERSTEP01', 'TB6560', 'TB6600', 'TMC2100',
868 */
869 #define X_DRIVER_TYPE TMC2130
870 #define Y_DRIVER_TYPE TMC2130
871 #define Z_DRIVER_TYPE TMC2130
872 // #define X2_DRIVER_TYPE A4988
873 // #define Y2_DRIVER_TYPE A4988
874 // #define Z2_DRIVER_TYPE A4988
875 // #define Z3_DRIVER_TYPE A4988
876 // #define Z4_DRIVER_TYPE A4988
877 // #define I_DRIVER_TYPE A4988
878 // #define J_DRIVER_TYPE A4988
879 // #define K_DRIVER_TYPE A4988
880 #define E0_DRIVER_TYPE TMC2130
881 #define E1_DRIVER_TYPE TMC2130

```

When using SPI mode, `TMC_USE_SW_SPI` needs to be Uncommented in Configuration_adv.h

```
#define TMC_USE_SW_SPI
```



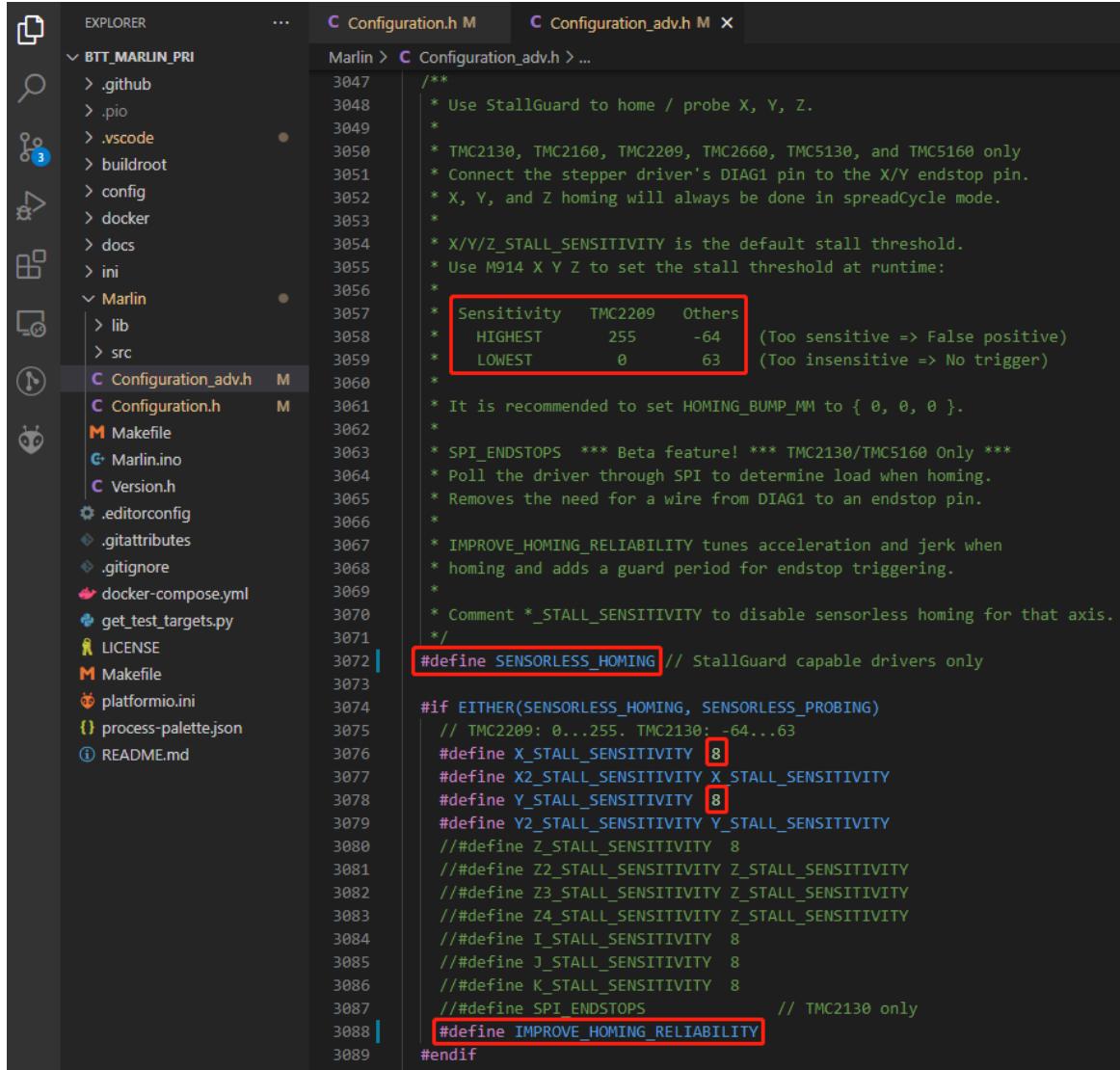
The screenshot shows the VS Code interface with the 'Configuration_adv.h' file open in the editor. The 'EXPLORER' sidebar on the left shows the project structure, including 'BTT_MARLIN_PRI' and 'Marlin' directories containing various configuration files like 'Configuration.h', 'Configuration_adv.h', and 'Version.h'. The 'Marlin' directory is currently selected. The 'Configuration_adv.h' file in the editor has one line of code highlighted with a red box:

```

2900 /**
2901 * Software option for SPI driven drivers (TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160).
2902 * The default SW SPI pins are defined the respective pins files,
2903 * but you can override or define them here.
2904 */
2905
2906 #define TMC_USE_SW_SPI
2907 // #define TMC_SW_MOSI -1
2908 // #define TMC_SW_MISO -1
2909 // #define TMC_SW_SCK -1

```

4. 3. 5 Sensorless homing



The screenshot shows the VS Code interface with the 'EXPLORER' view on the left and the 'Configuration_adv.h' file open in the main editor. The file contains C code for Marlin firmware. Several lines of code are highlighted with red boxes:

```

3047     /**
3048      * Use StallGuard to home / probe X, Y, Z.
3049      *
3050      * TMC2130, TMC2160, TMC2209, TMC2660, TMC5130, and TMC5160 only
3051      * Connect the stepper driver's DIAG1 pin to the X/Y endstop pin.
3052      * X, Y, and Z homing will always be done in spreadCycle mode.
3053      *
3054      * X/Y/Z_STALL_SENSITIVITY is the default stall threshold.
3055      * Use M914 X Y Z to set the stall threshold at runtime:
3056      *
3057      * Sensitivity    TMC2209    Others
3058      * HIGHEST       255       -64      (Too sensitive => False positive)
3059      * LOWEST        0        63      (Too insensitive => No trigger)
3060      *
3061      * It is recommended to set HOMING_BUMP_MM to { 0, 0, 0 }.
3062      *
3063      * SPI_ENDSTOPS *** Beta feature! *** TMC2130/TMC5160 Only ***
3064      * Poll the driver through SPI to determine load when homing.
3065      * Removes the need for a wire from DIAG1 to an endstop pin.
3066      *
3067      * IMPROVE_HOMING_RELIABILITY tunes acceleration and jerk when
3068      * homing and adds a guard period for endstop triggering.
3069      *
3070      * Comment *_STALL_SENSITIVITY to disable sensorless homing for that axis.
3071      */
3072 #define SENSORLESS_HOMING // StallGuard capable drivers only
3073
3074 #if EITHER(SENSORLESS_HOMING, SENSORLESS_PROBING)
3075 // TMC2209: 0...255, TMC2130: -64...63
3076 #define X_STALL_SENSITIVITY 8
3077 #define X2_STALL_SENSITIVITY X_STALL_SENSITIVITY
3078 #define Y_STALL_SENSITIVITY 8
3079 #define Y2_STALL_SENSITIVITY Y_STALL_SENSITIVITY
3080 //#define Z_STALL_SENSITIVITY 8
3081 //#define Z2_STALL_SENSITIVITY Z_STALL_SENSITIVITY
3082 //#define Z3_STALL_SENSITIVITY Z_STALL_SENSITIVITY
3083 //#define Z4_STALL_SENSITIVITY Z_STALL_SENSITIVITY
3084 //#define I_STALL_SENSITIVITY 8
3085 //#define J_STALL_SENSITIVITY 8
3086 //#define K_STALL_SENSITIVITY 8
3087 //#define SPI_ENDSTOPS // TMC2130 only
3088 #define IMPROVE_HOMING_RELIABILITY
3089#endif

```

```

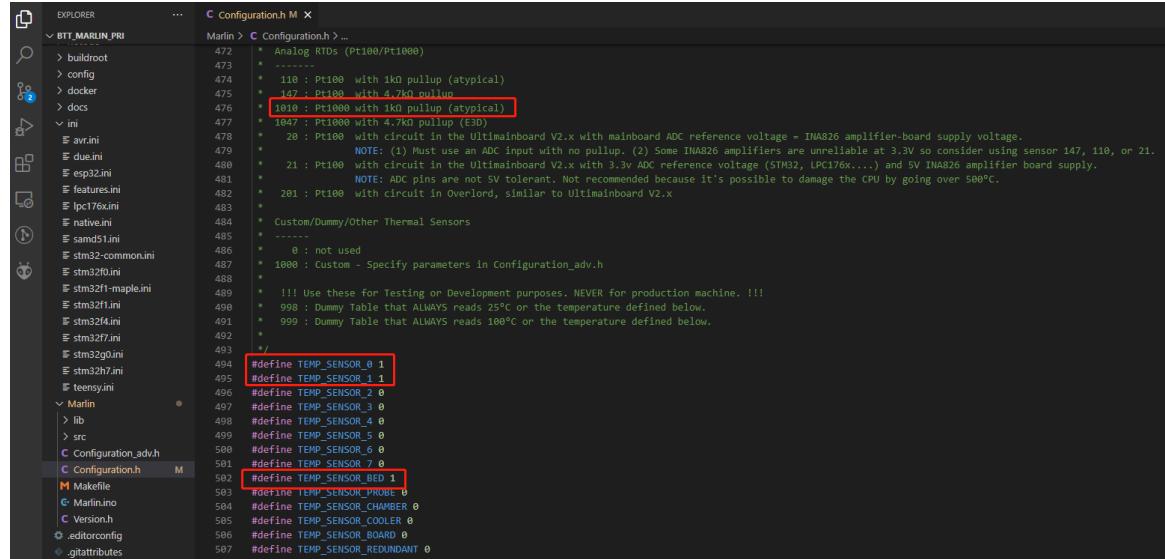
#define SENSORLESS_HOMING // enable sensorless homing
#define xx_STALL_SENSITIVITY 8 // sensitivity setting ,TMC2209 range from 0
to 255, higher number results in more sensitive trigger threshold, sensitivity
too high will cause endpoint to trigger before gantry actually move to the
end, lower number results in less sensitive trigger threshold,, too low of
sensitivity will cause endpoint to not trigger and gantrying continue. Other
drivers ranges from 63 to -64, lower numbers results in more sensitive trigger
threshold
#define IMPROVE_HOMING_RELIABILITY // can be used to set independent motor
current for homing moves(xx_CURRENT_HOME) to improve homing reliability.

```

4. 3. 6 100K NTC or PT1000

When using 100K NTC, pullup resistance is 4.7K, when using Pt1000, pullup resistance is 1K, set sensor type to 1 for 100K NTC +4.7K pullup resistance , 1010 for PT1000 + 1K pullup resistance.

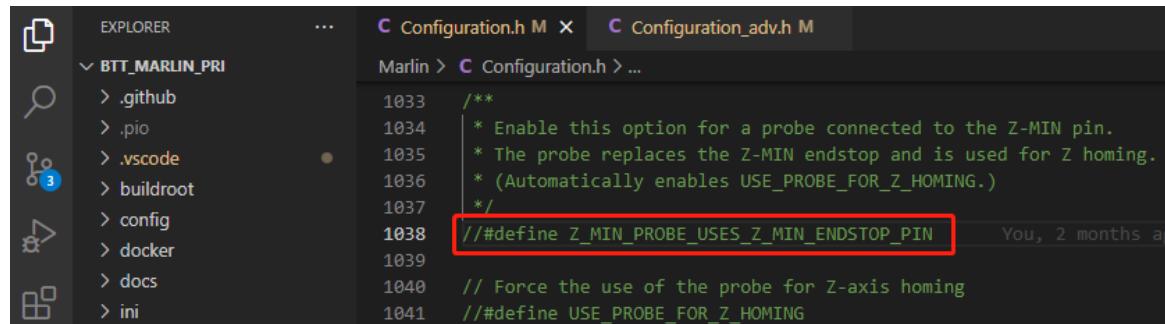
```
#define TEMP_SENSOR_0 1
#define TEMP_SENSOR_1 1
#define TEMP_SENSOR_BED 1
```



```
Configuration.h M X
Marlin > Configuration.h > ...
472 * Analog RTDs (Pt100/Pt1000)
473 *
474 * 110 : Pt100 with 1kΩ pullup (atypical)
475 147 : Pt100 with 4.7kΩ pullup
476 1010 : Pt100 with 1kΩ pullup (atypical)
477 1847 : Pt1000 with 4.7kΩ pullup (E3D)
478 *
479 * NOTE: (1) Must use an ADC input with no pullup. (2) Some INA826 amplifiers are unreliable at 3.3V so consider using sensor 147, 110, or 21.
480 *
481 * 21 : Pt100 with circuit in the Ultimainboard V2.x with 3.3V ADC reference voltage (STM32, LPC176x,...) and 5V INA826 amplifier board supply.
482 *
483 * NOTE: ADC pins are not 5V tolerant. Not recommended because it's possible to damage the CPU by going over 500°C.
484 *
485 * 201 : Pt100 with circuit in Overlord, similar to Ultimainboard V2.x
486 *
487 * Custom/Dummy/Other Thermal Sensors
488 *
489 * 0 : not used
490 * 1000 : Custom - Specify parameters in Configuration_adv.h
491 *
492 * !!! Use these for Testing or Development purposes. NEVER for production machine. !!!
493 *
494 #define TEMP_SENSOR_0 1
495 #define TEMP_SENSOR_1 1
496 #define TEMP_SENSOR_2 0
497 #define TEMP_SENSOR_3 0
498 #define TEMP_SENSOR_4 0
499 #define TEMP_SENSOR_5 0
500 #define TEMP_SENSOR_6 0
501 #define TEMP_SENSOR_7 0
502 #define TEMP_SENSOR_BED 1
503 #define TEMP_SENSOR_PROBE 0
504 #define TEMP_SENSOR_CHAMBER 0
505 #define TEMP_SENSOR_COOLER 0
506 #define TEMP_SENSOR_BOARD 0
507 #define TEMP_SENSOR_REDUNDANT 0

Configuration_adv.h M
Marlin > Configuration.h > ...
1033 /**
1034 * Enable this option for a probe connected to the Z-MIN pin.
1035 * The probe replaces the Z-MIN endstop and is used for Z homing.
1036 * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
1037 */
1038 //##define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
1039
1040 // Force the use of the probe for Z-axis homing
1041 //##define USE_PROBE_FOR_Z_HOMING
```

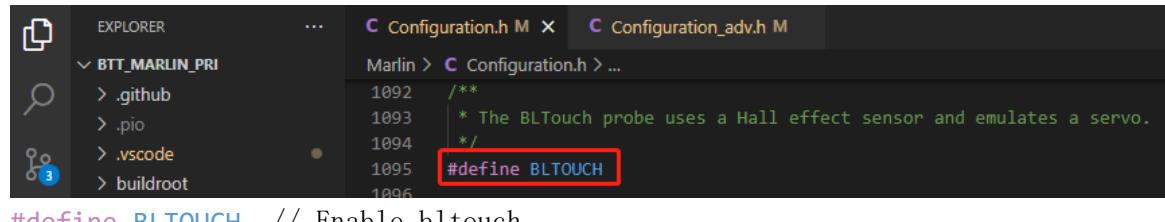
4. 3. 7 BLTouch



```
Configuration.h M X Configuration_adv.h M
Marlin > Configuration.h > ...
1033 /**
1034 * Enable this option for a probe connected to the Z-MIN pin.
1035 * The probe replaces the Z-MIN endstop and is used for Z homing.
1036 * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
1037 */
1038 //##define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
1039
1040 // Force the use of the probe for Z-axis homing
1041 //##define USE_PROBE_FOR_Z_HOMING

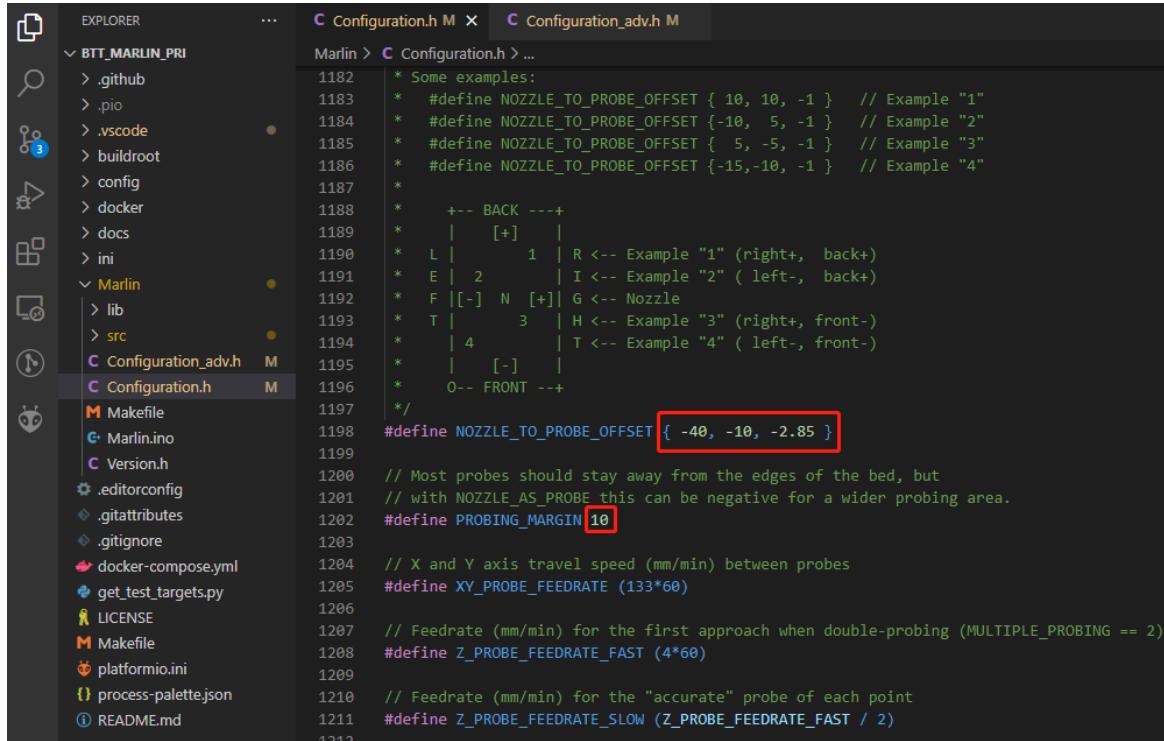
Configuration.h M X Configuration_adv.h M
Marlin > Configuration.h > ...
1092 /**
1093 * The BLTouch probe uses a Hall effect sensor and emulates a servo.
1094 */
1095 #define BLTOUCH
1096
```

```
//##define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN //
```



```
Configuration.h M X Configuration_adv.h M
Marlin > Configuration.h > ...
1092 /**
1093 * The BLTouch probe uses a Hall effect sensor and emulates a servo.
1094 */
1095 #define BLTOUCH
1096

Configuration.h M X Configuration_adv.h M
Marlin > Configuration.h > ...
#define BLTOUCH // Enable bltouch
```



The screenshot shows the VS Code interface with two tabs open: Configuration.h and Configuration_adv.h. The Configuration.h tab is active. The code editor displays configuration settings for a printer. A specific line of code is highlighted with a red box:

```

1182 * Some examples:
1183 * #define NOZZLE_TO_PROBE_OFFSET { 10, 10, -1 } // Example "1"
1184 * #define NOZZLE_TO_PROBE_OFFSET {-10, 5, -1 } // Example "2"
1185 * #define NOZZLE_TO_PROBE_OFFSET { 5, -5, -1 } // Example "3"
1186 * #define NOZZLE_TO_PROBE_OFFSET {-15,-10, -1 } // Example "4"
1187 *
1188 *     +-+ BACK +-+
1189 *     |   [+]   |
1190 *     L | 1      | R <- Example "1" (right+, back+)
1191 *     E | 2      | I <- Example "2" (left-, back+)
1192 *     F |[-] N [+]| G <- Nozzle
1193 *     T | 3      | H <- Example "3" (right+, front-)
1194 *     | 4      | T <- Example "4" (left-, front-)
1195 *     | [-]   |
1196 *     O-- FRONT --+
1197 */
1198 #define NOZZLE_TO_PROBE_OFFSET { -40, -10, -2.85 }
1199
1200 // Most probes should stay away from the edges of the bed, but
1201 // with NOZZLE_AS_PROBE this can be negative for a wider probing area.
1202 #define PROBING_MARGIN 10
1203
1204 // X and Y axis travel speed (mm/min) between probes
1205 #define XY_PROBE_FEEDRATE (133*60)
1206
1207 // Feedrate (mm/min) for the first approach when double-probing (MULTIPLE_PROBING == 2)
1208 #define Z_PROBE_FEEDRATE_FAST (4*60)
1209
1210 // Feedrate (mm/min) for the "accurate" probe of each point
1211 #define Z_PROBE_FEEDRATE_SLOW (Z_PROBE_FEEDRATE_FAST / 2)
1212

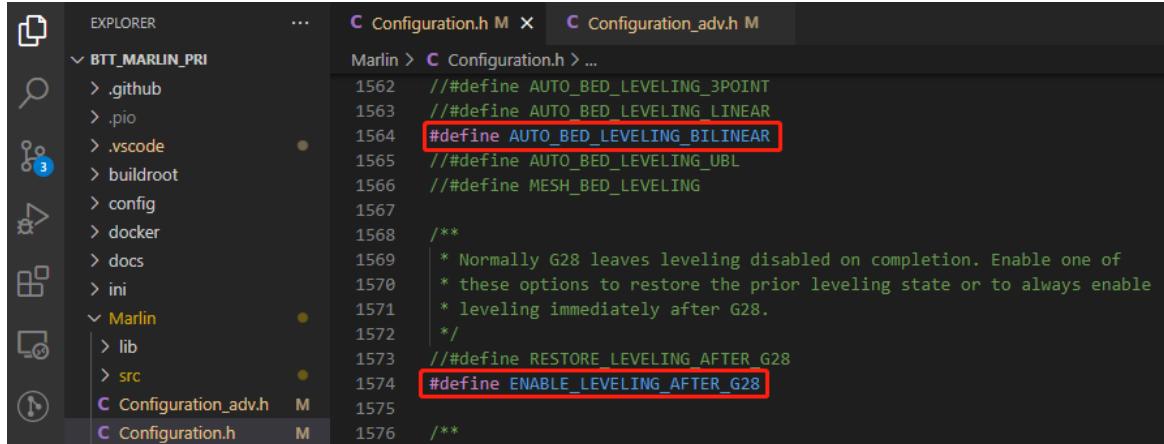
```

```

#define NOZZLE_TO_PROBE_OFFSET { -40, -10, -2.85 } // set BLtouch probe
offset

#define PROBING_MARGIN 10 // set distance between probe area and print area
perimeter

```



The screenshot shows the VS Code interface with two tabs open: Configuration.h and Configuration_adv.h. The Configuration.h tab is active. The code editor displays configuration settings for a printer. A specific line of code is highlighted with a red box:

```

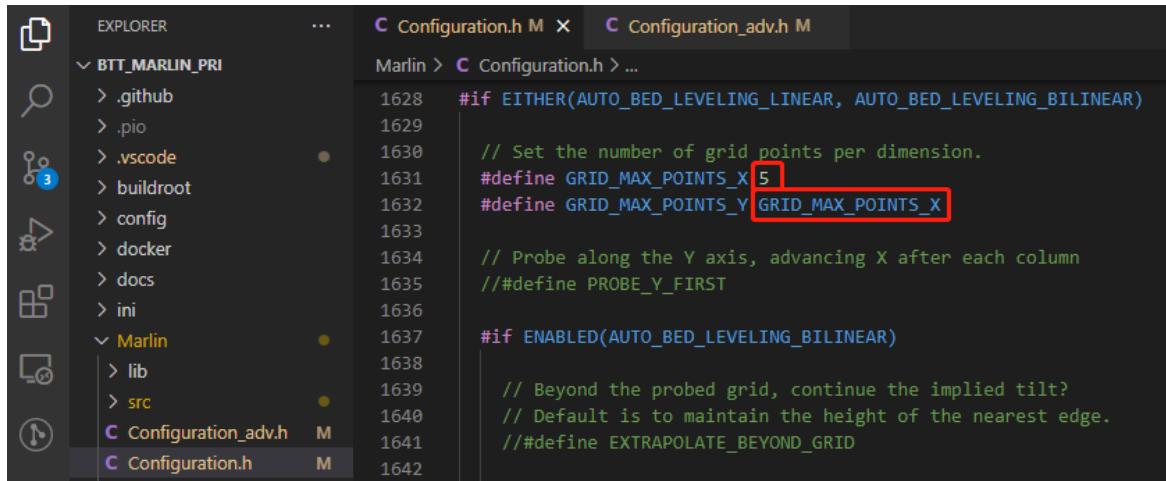
1562 //##define AUTO_BED_LEVELING_3POINT
1563 //##define AUTO_BED_LEVELING_LINEAR
1564 #define AUTO_BED_LEVELING_BILINEAR
1565 //##define AUTO_BED_LEVELING_UBL
1566 //##define MESH_BED_LEVELING
1567
1568 /**
1569 * Normally G28 leaves leveling disabled on completion. Enable one of
1570 * these options to restore the prior leveling state or to always enable
1571 * leveling immediately after G28.
1572 */
1573 //##define RESTORE_LEVELING_AFTER_G28
1574 #define ENABLE_LEVELING_AFTER_G28
1575
1576 /**

```

```

#define AUTO_BED_LEVELING_BILINEAR // set probe pattern
#define RESTORE_LEVELING_AFTER_G28 // apply leveling after G28 homing command

```

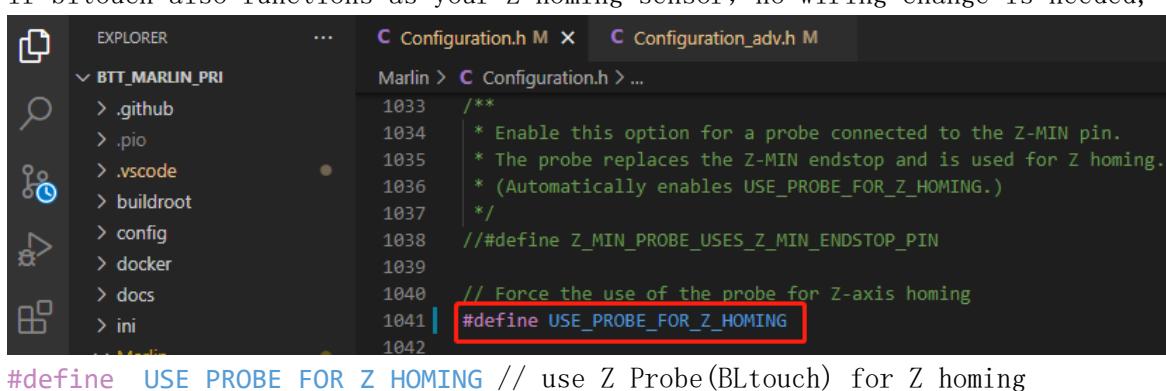


```

#define GRID_MAX_POINTS_X 5 // set number of probe points for x axis, usually
5 point is sufficient
#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X // set number of probe points
for Y axis to the same as X axis

```

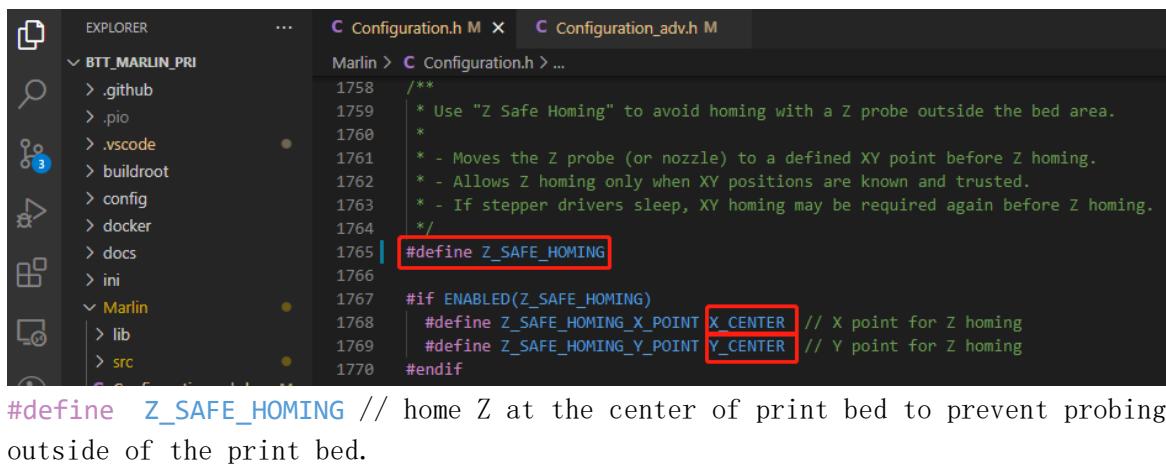
If bltouch also functions as your Z homing sensor, no wiring change is needed,



```

#define USE_PROBE_FOR_Z_HOMING // use Z Probe(BLtouch) for Z homing

```



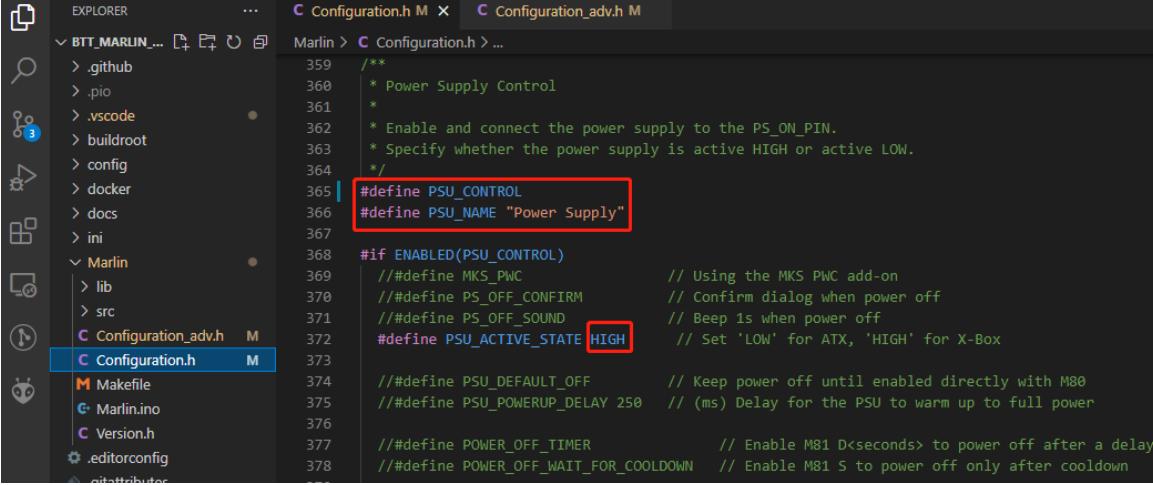
```

#define Z_SAFE_HOMEING
#if ENABLED(Z_SAFE_HOMEING)
#define Z_SAFE_HOMEING_X_POINT X_CENTER // X point for Z homing
#define Z_SAFE_HOMEING_Y_POINT Y_CENTER // Y point for Z homing
#endif

```

#define Z_SAFE_HOMEING // home Z at the center of print bed to prevent probing outside of the print bed.

4. 3. 8 Auto power off(Relay V1. 2)



```

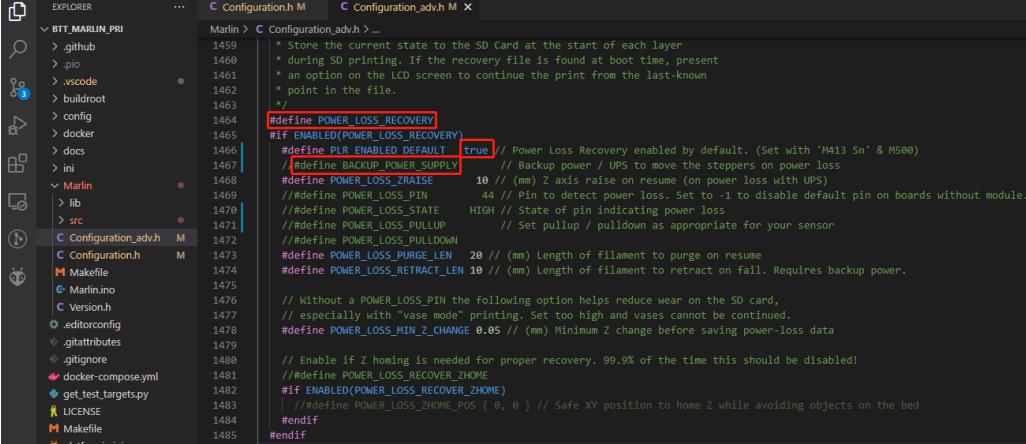
#define PSU_CONTROL // enable PSU control to turn on and off using M80 and M81
#define PSU_ACTIVE_STATE HIGH // set turn on level, Relay V1.2 is turned on with high level and turned off with low level ,so this setting needs to be HIGH.

```

4. 3. 9 Power loss recovery

There are two methods for power lost recovery

- No extra module needed, the motherboard will write current print status to the SD card after every layer is printed, which shortens the life of the SD card severely.



```

#define POWER LOSS RECOVERY // enable power loss recovery#define PLR_ENABLED_DEFAULT true // true default to power loss recovery enabled

```

- external UPS 24V V1.0 module, when power is cut, the module will provide power to the board and signal the board to save current print status to SD card. This method has virtually no effect on the life of the SD card.

```

#define POWER LOSS RECOVERY // enable power loss recovery
#define PLR_ENABLED_DEFAULT true // true default to power loss recovery
enabned
#define POWER LOSS ZRAISE 10 // raise the print head by 10mm after
power loss to prevent the nozzle from touching the printed part.
#define POWER LOSS STATE HIGH // set signal level, UPS 24V V1.0 returns
low level when not triggered and HIGH level when power is cut, thus this
setting needs to be HIGH.

```

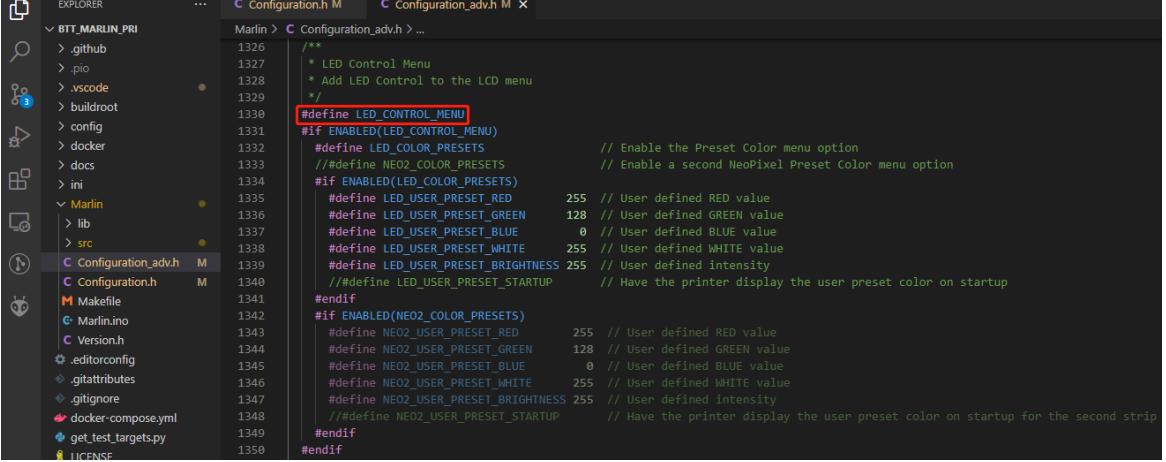
4. 3. 10 RGB

```

#define NEOPIXEL LED // enable Neopixel
#define NEOPIXEL_TYPE NEO_GRB // set Neopixel type
//#define NEOPIXEL_PIN 4 // disable PIN setting, use the correct signal pin
in the pin file of the motherboard
#define NEOPIXEL_PIXELS 30 // number of leds
#define NEOPIXEL_STARTUP_TEST // the light will show red green and blue
sequentially to self-test.

```

If you are using displays like LCD2004, 12864, mini12864 etc, you can also control RGB from your display directly.



```

EXPLORER ... C Configuration.h M C Configuration_adv.h M
BTT_MARLIN_PRI Marlin > C Configuration.h ...
>.github 1326 /**
>.pio 1327 * LED Control Menu
>.vscode 1328 * Add LED Control to the LCD menu
>buildroot 1329 */
>config 1330 */
>docker 1331 */
>docs 1332 */
>ini 1333 */
Marlin 1334 */
>lib 1335 */
>src 1337 */
Configuration_adv.h M 1338 */
Configuration.h M 1339 */
Makefile 1340 */
Marlin.ino 1342 */
Version.h 1343 */
.editorconfig 1344 */
.gittarributes 1345 */
.gitignore 1347 */
.docker-compose.yml 1348 */
.get_test_targets.py 1349 */
LICENSE 1350 */

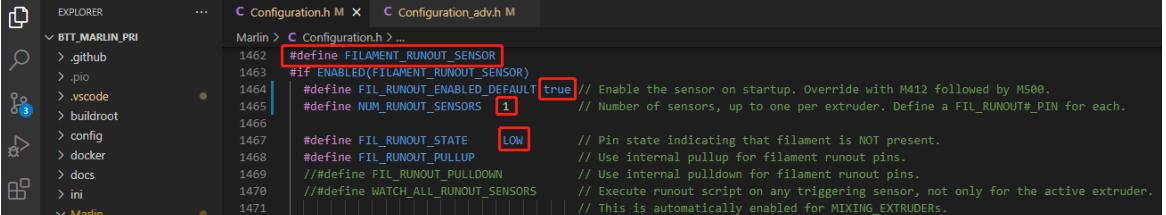
#define LED_CONTROL_MENU // And led control to your menu
#endif
#if ENABLED(LED_CONTROL_MENU)
#define LED_COLOR_PRESETS // Enable the Preset Color menu option
//define NEO2_COLOR_PRESETS // Enable a second NeoPixel Preset Color menu option
#if ENABLED(LED_COLOR_PRESETS)
#define LED_USER_PRESET_RED 255 // User defined RED value
#define LED_USER_PRESET_GREEN 128 // User defined GREEN value
#define LED_USER_PRESET_BLUE 0 // User defined BLUE value
#define LED_USER_PRESET_WHITE 255 // User defined WHITE value
#define LED_USER_PRESET_BRIGHTNESS 255 // User defined intensity
//define LED_USER_PRESET_STARTUP // Have the printer display the user preset color on startup
#endif
#if ENABLED(NEO2_COLOR_PRESETS)
#define NEO2_USER_PRESET_RED 255 // User defined RED value
#define NEO2_USER_PRESET_GREEN 128 // User defined GREEN value
#define NEO2_USER_PRESET_BLUE 0 // User defined BLUE value
#define NEO2_USER_PRESET_WHITE 255 // User defined WHITE value
#define NEO2_USER_PRESET_BRIGHTNESS 255 // User defined intensity
//define NEO2_USER_PRESET_STARTUP // Have the printer display the user preset color on startup for the second strip
#endif
#endif

```

`#define LED_CONTROL_MENU // And led control to your menu`

4.3.11 Filament sensor

Standard filament run out sensors are usually comprised of a microswitch which signals the mainboard of filament status with High or Low level signal.



```

EXPLORER ... C Configuration.h M X C Configuration_adv.h M
BTT_MARLIN_PRI Marlin > C Configuration.h ...
>.github 1462 #define FILAMENT_RUNOUT_SENSOR
>.pio 1463 #if ENABLED(FILAMENT_RUNOUT_SENSOR)
>.vscode 1464 #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M500.
>buildroot 1465 #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT_PIN for each.
>config 1466 */
>docker 1467 */
>docs 1468 */
>ini 1469 */
Marlin 1470 */
1471 */

#define FILAMENT_RUNOUT_SENSOR // enable filament run out sensor
#define FIL_RUNOUT_ENABLED_DEFAULT true // true default to filament run out sensor enabled
#define NUM_RUNOUT_SENSORS 1 // number of filament run out sensor
#define FIL_RUNOUT_STATE LOW // voltage level of the filament runout sensor trigger signal.
//define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins.
//define FIL_RUNOUT_PULLDOWN // Use internal pulldown for filament runout pins.
//define WATCH_ALL_RUNOUT_SENSORS // Execute runout script on any triggering sensor, not only for the active extruder.
// This is automatically enabled for MIXING_EXTRUDERS.

```

`#define FILAMENT_RUNOUT_SENSOR // enable filament run out sensor`
`#define FIL_RUNOUT_ENABLED_DEFAULT true // true default to filament run out sensor enabled`
`#define NUM_RUNOUT_SENSORS 1 // number of filament run out sensor`
`#define FIL_RUNOUT_STATE LOW // voltage level of the filament runout sensor trigger signal.`

4. 3. 12 smart filament sensor (SFS V1. 0 / V2. 0)

The Smart filament sensor works by continuously sending signal to the mainboard to communicate filament status

```

#define FILAMENT_RUNOUT_SENSOR
#if ENABLED(FILAMENT_RUNOUT_SENSOR)
#define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M500.
#define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT#_PIN for each.
#define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT present.
#define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins...
// Override individually if the runout sensors vary ...
//##define FIL_RUNOUT2_STATE LOW...
//##define FIL_RUNOUT3_STATE LOW...
//##define FIL_RUNOUT4_STATE LOW...
//##define FIL_RUNOUT6_STATE LOW...
//##define FIL_RUNOUT7_STATE LOW...
//##define FIL_RUNOUT8_STATE LOW...
// Commands to execute on filament runout.
// With multiple runout sensors use the %c placeholder for the current tool in commands (e.g., "M600 T%c")
// NOTE: After "M412 H1" the host handles filament runout and this script does not apply.
#define FILAMENT_RUNOUT_SCRIPT "M600"

// After a runout is detected, continue printing this length of filament
// before executing the runout script. Useful for a sensor at the end of
// a feed tube. Requires 4 bytes SRAM per sensor, plus 4 bytes overhead.
#define FILAMENT_RUNOUT_DISTANCE_MM 3

#ifdef FILAMENT_RUNOUT_DISTANCE_MM
// Enable this option to use an encoder disc that toggles the runout pin
// as the filament moves. (Be sure to set FILAMENT_RUNOUT_DISTANCE_MM
// large enough to avoid false positives.)
#define FILAMENT_MOTION_SENSOR
#endif
#endif

```

```

#define FILAMENT_MOTION_SENSOR // set encoder type
#define FILAMENT_RUNOUT_DISTANCE_MM 3 // set sensitivity, SFS V1.0 nominal
setting should be 7mm, which means if no signal of filament movement is
detected after 7mm of filament travel command, filament error will be
triggered , SFS V2.0 nominal setting should be 3mm

```

The settings below also need to be set to instruct the printer to park the nozzle after filament error is detected.

```

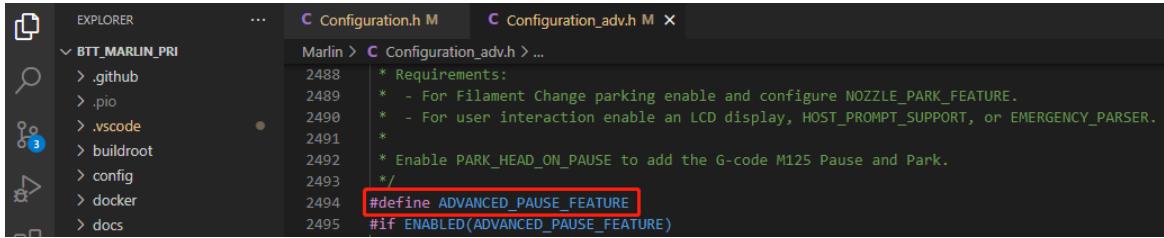
#define NOZZLE_PARK_FEATURE
#if ENABLED(NOZZLE_PARK_FEATURE)
// Specify a park position as { X, Y, Z raise }
#define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 }
//##define NOZZLE_PARK_X_ONLY // X move only is required to park
//##define NOZZLE_PARK_Y_ONLY // Y move only is required to park
#define NOZZLE_PARK_Z_RAISE_MIN 2 // (mm) Always raise Z by at least this distance
#define NOZZLE_PARK_XY_FEEDRATE 100 // (mm/s) X and Y axes feedrate (also used for delta Z axis)
#define NOZZLE_PARK_Z_FEEDRATE 5 // (mm/s) Z axis feedrate (not used for delta printers)
#endif

```

```

#define NOZZLE_PARK_FEATURE // park nozzle
#define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 } //set
the X, Y, and Z offset coordinate of the nozzle

```



```

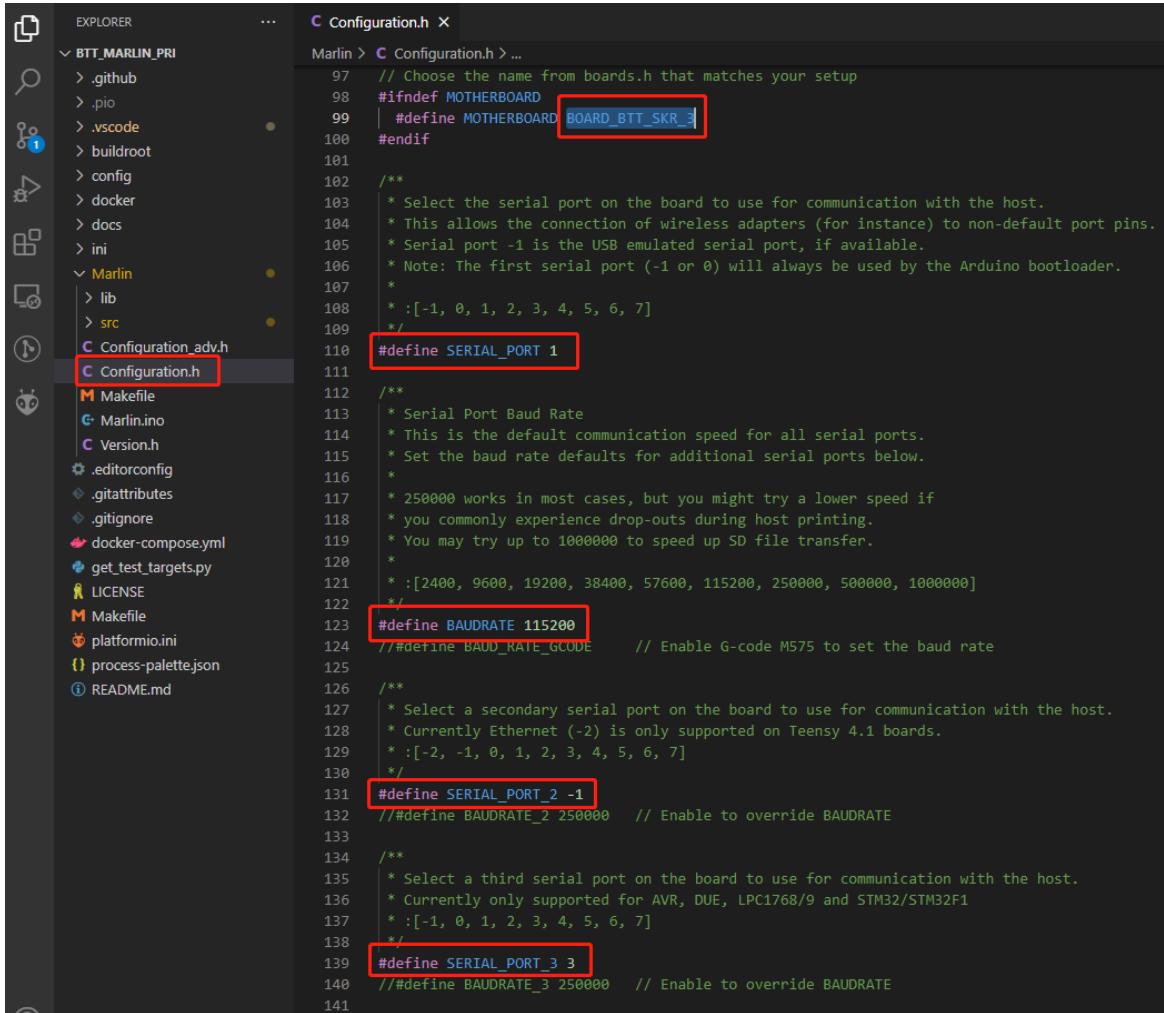
C Configuration.h M C Configuration_adv.h M
Marlin > C Configuration_adv.h > ...
2488 * Requirements:
2489 * - For Filament Change parking enable and configure NOZZLE_PARK_FEATURE.
2490 * - For user interaction enable an LCD display, HOST_PROMPT_SUPPORT, or EMERGENCY_PARSER.
2491 *
2492 * Enable PARK_HEAD_ON_PAUSE to add the G-code M125 Pause and Park.
2493 */
2494 #define ADVANCED_PAUSE_FEATURE
2495 #if ENABLED(ADVANCED_PAUSE_FEATURE)

```

`#define ADVANCED_PAUSE_FEATURE` // retraction setting of nozzle park movement and filament purge distance after print is resumed.

4.3.13 ESP3D

The serial port between ESP8266 and Marlin on the motherboard is UART3



```

C Configuration.h X
Marlin > C Configuration.h > ...
97 // Choose the name from boards.h that matches your setup
98 #ifndef MOTHERBOARD
99 | #define MOTHERBOARD BOARD_BTT_SKR_3
100 #endif
101 /**
102 * Select the serial port on the board to use for communication with the host.
103 * This allows the connection of wireless adapters (for instance) to non-default port pins.
104 * Serial port -1 is the USB emulated serial port, if available.
105 * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
106 *
107 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
108 */
109 #define SERIAL_PORT_1
110 /**
111 * Serial Port Baud Rate
112 * This is the default communication speed for all serial ports.
113 * Set the baud rate defaults for additional serial ports below.
114 *
115 * 250000 works in most cases, but you might try a lower speed if
116 * you commonly experience drop-outs during host printing.
117 * You may try up to 1000000 to speed up SD file transfer.
118 *
119 * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
120 */
121 #define BAUDRATE_115200
122 //##define BAUD_RATE_GCODE // Enable G-code M575 to set the baud rate
123 /**
124 * Select a secondary serial port on the board to use for communication with the host.
125 * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
126 * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
127 */
128 #define SERIAL_PORT_2 -1
129 //##define BAUDRATE_2 250000 // Enable to override BAUDRATE
130 /**
131 * Select a third serial port on the board to use for communication with the host.
132 * Currently only supported for AVR, DUE, LPC1768/9 and STM32/STM32F1
133 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
134 */
135 #define SERIAL_PORT_3 3
136 //##define BAUDRATE_3 250000 // Enable to override BAUDRATE
137
138
139
140
141

```

the newest ESP3D firmware can be found at <https://github.com/luc-github/ESP3D> , compile your own binary file and rename to “esp3d.bin” , copy to the root directory of the sd card, insert into the motherboard and press reset button. The bootloader will update the firmware to ESP8266 automatically. If updated successfully, the file will be renamed to ESP3D.CUR

4.4 Compile firmware

1. Click “√” to compile firmware



2. Copy the compiled “firmware.bin” to SD card and insert to motherboard to update firmware

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL GITLENS

Indexing .pio\build\STM32H743Vx_btt\libFrameworkArduino.a
Linking .pio\build\STM32H743Vx_btt\firmware.elf
Checking size .pio\build\STM32H743Vx_btt\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [ ] 1.7% (used 18088 bytes from 1048576 bytes)
Flash: [= ] 10.9% (used 228120 bytes from 2097152 bytes)
Building .pio\build\STM32H743Vx_btt\firmware.bin
===== [SUCCESS] Took 95.65 seconds =====

Environment Status Duration
STM32H743Vx_btt SUCCESS 00:01:35.650
===== 1 succeeded in 00:01:35.650 =====

Terminal will be reused by tasks, press any key to close it.
```

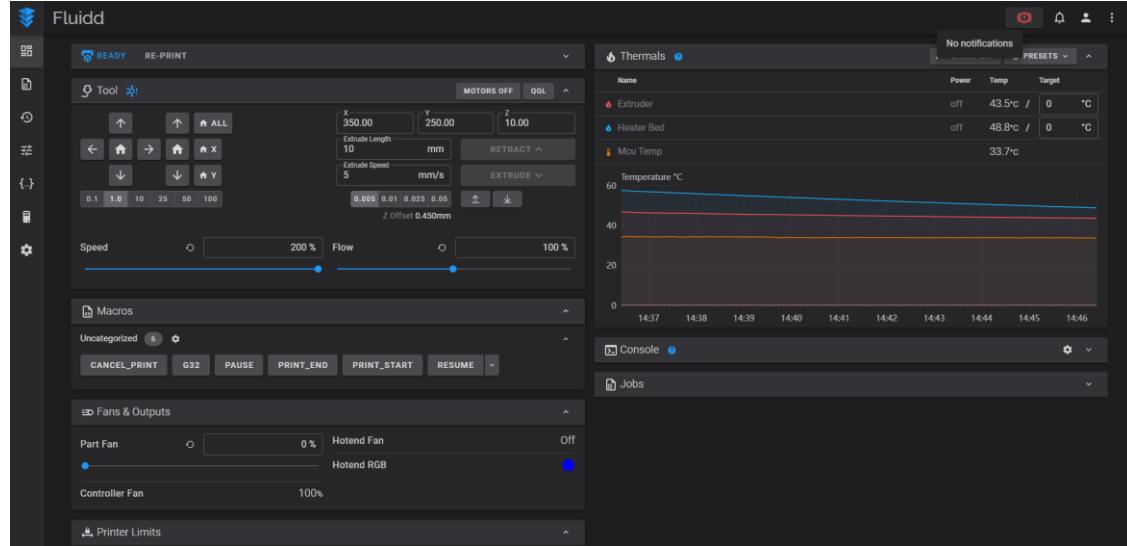
5. Klipper

5.1 Preparation

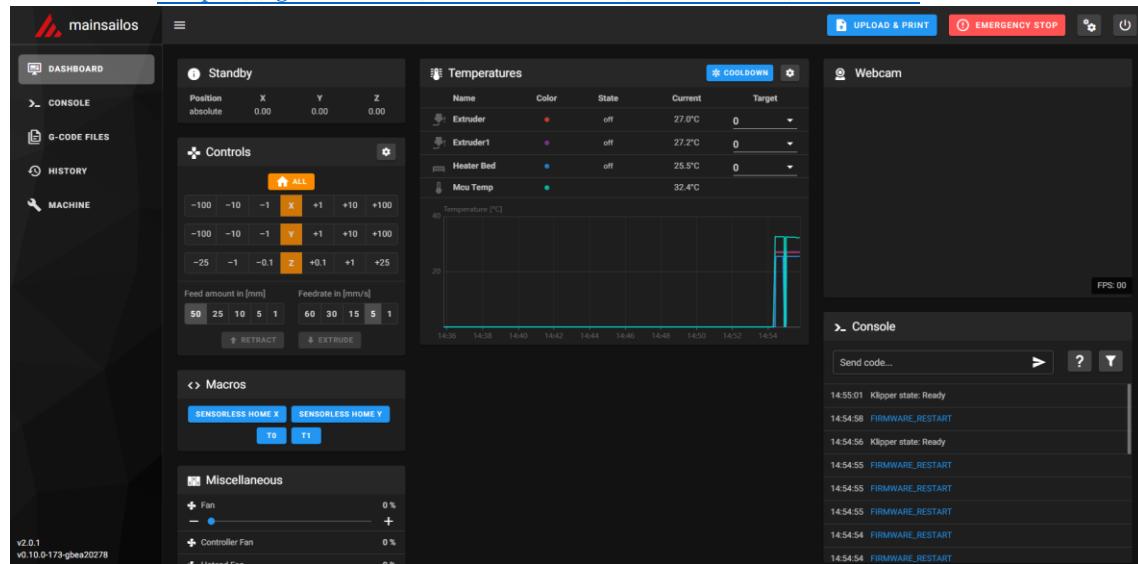
5.1.1 Download OS image

Download your preferred OS image with build in WebUI, popular choices are Fluidd, Mainsail etc.

Fluidd: <https://github.com/fluidd-core/FluiddPI/releases>



Mainsail: <https://github.com/mainsail-crew/MainsailOS/releases>



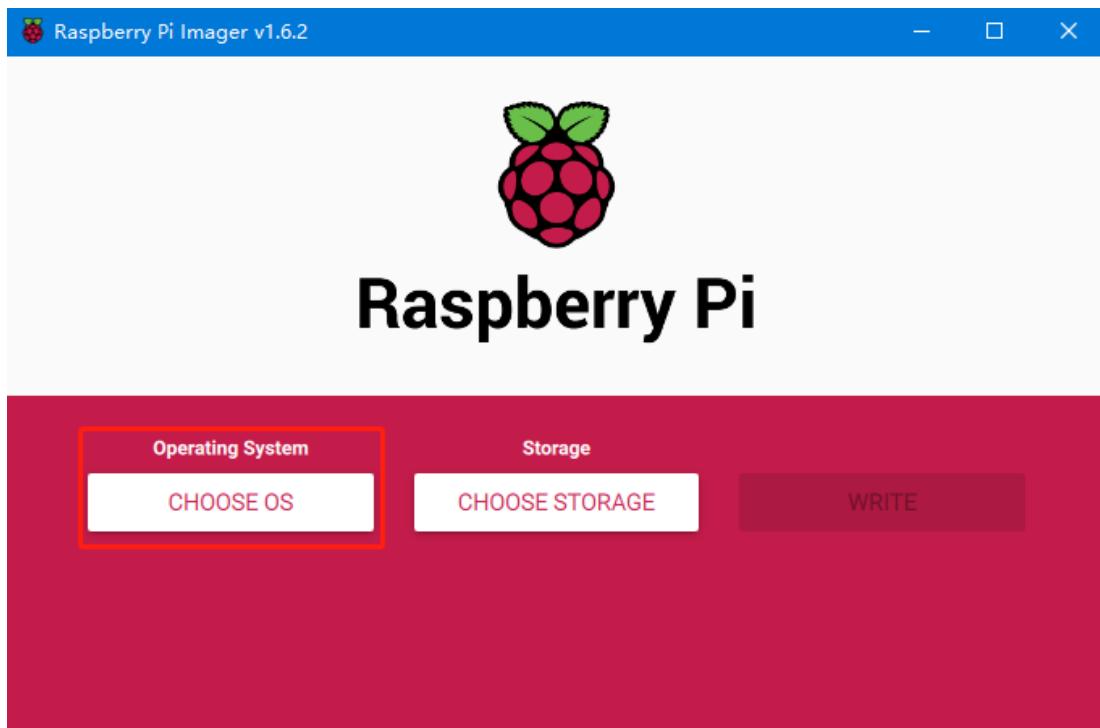
Or refer to [Klipper official installation guide](#) using Octoprint

5.1.2 Download and install Raspberry Pi Imager

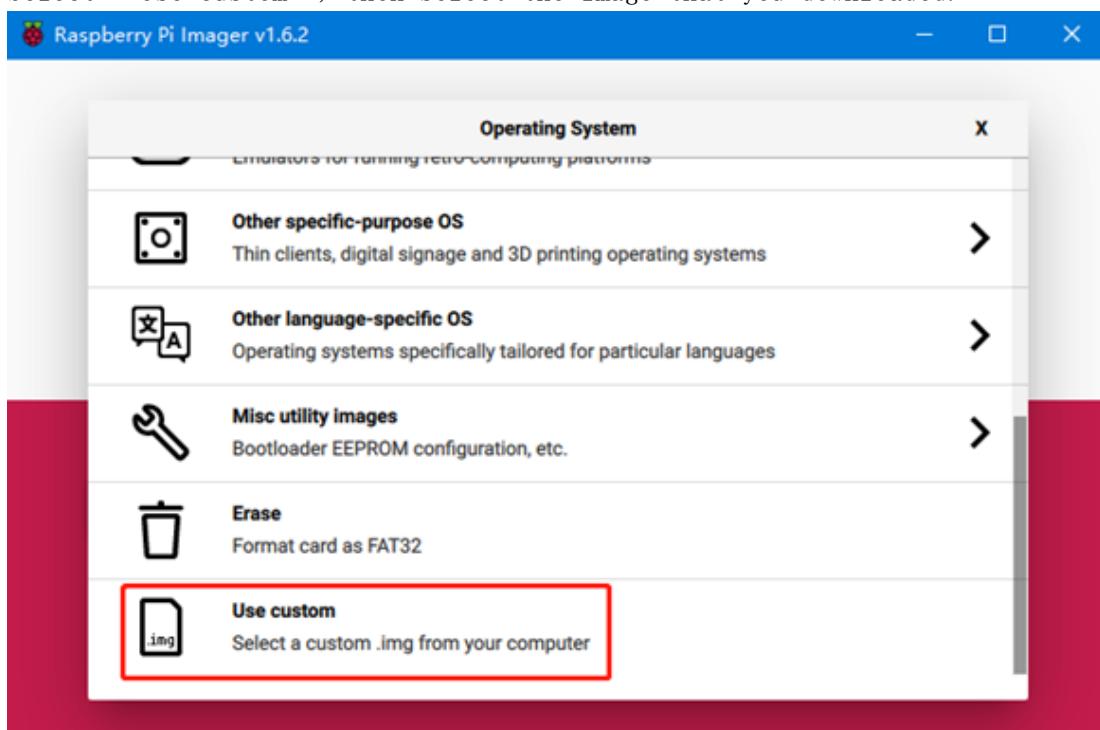
Install the official raspberry pi imager <https://www.raspberrypi.com/software/>

5.2 Write image

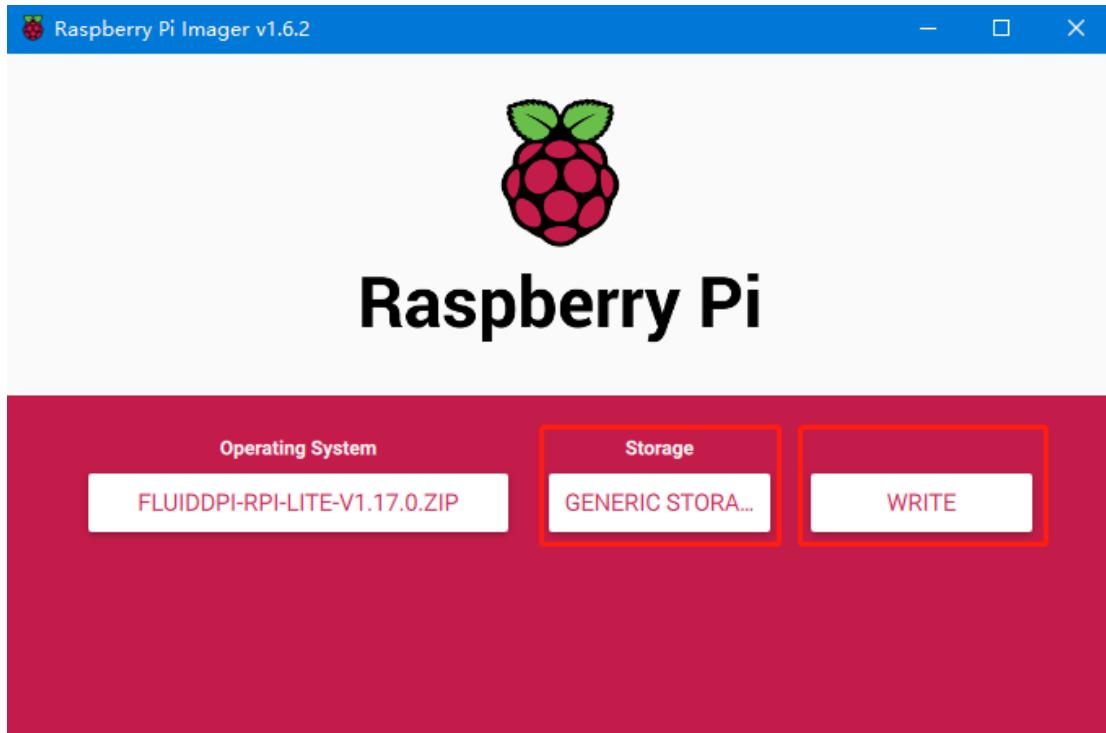
1. Insert Micro SD to your computer
2. Choose OS



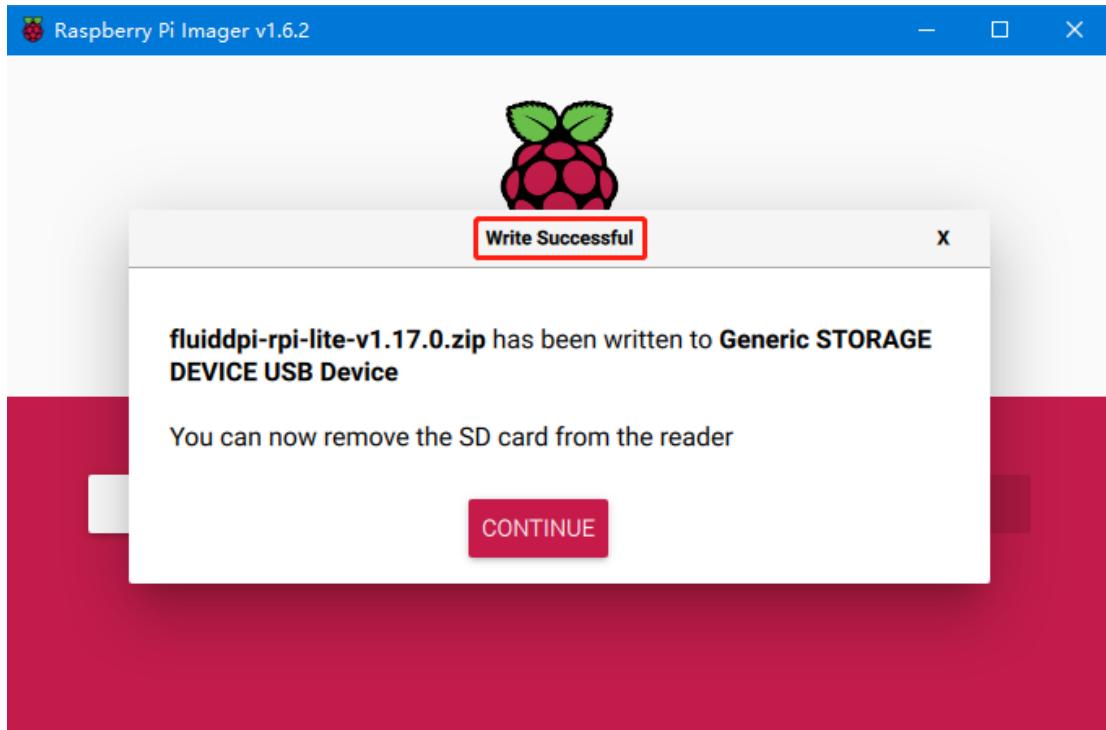
3. Select “Use custom” , then select the image that you downloaded.



4. Select the SD card and click “WRITE” (WRITE the image will format the SD card. Be careful not to select the wrong storage device, otherwise the data will be formatted)



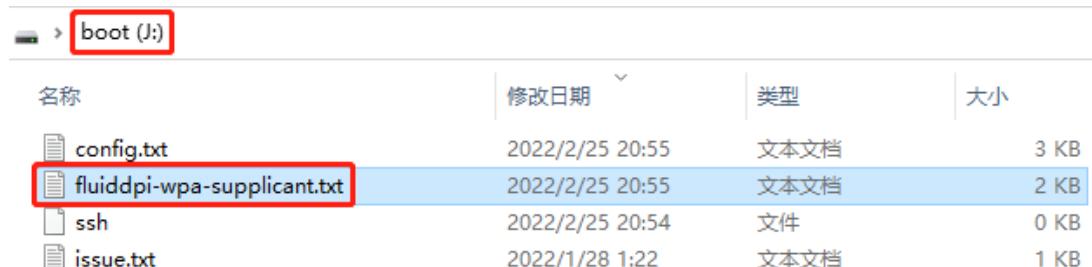
5. Wait for writing to finish



5.3 WIFI setting

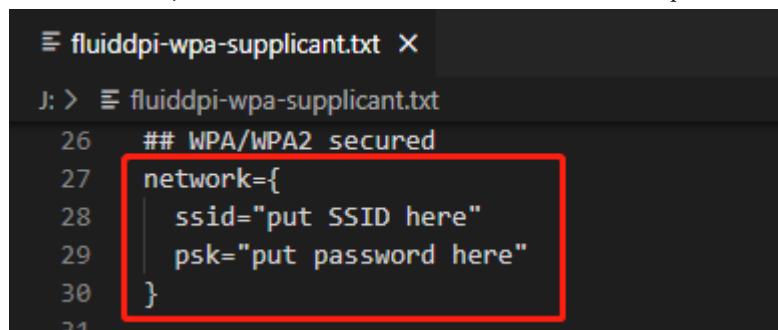
note: skip this step if you are using ethernet port not using WIFI

1. Reinsert the SD card
2. Find “fluiddpi-wpa-supplicant.txt” or “mainsail-wpa-supplicant.txt” in the SD card root directory, open it with VSCode (do not open it with windows notepad)



名称	修改日期	类型	大小
config.txt	2022/2/25 20:55	文本文档	3 KB
fluiddpi-wpa-supplicant.txt	2022/2/25 20:55	文本文档	2 KB
ssh	2022/2/25 20:54	文件	0 KB
issue.txt	2022/1/28 1:22	文本文档	1 KB

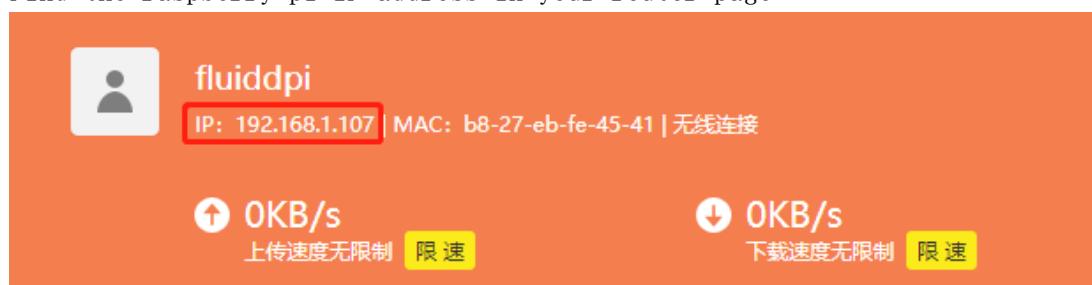
3. delete ‘#’ , insert the correct wifi SSID and password then save the file,



```
J: > fludpi-wpa-supplicant.txt
26 ## WPA/WPA2 secured
27 network={
28   ssid="put SSID here"
29   psk="put password here"
30 }
```

5.4 ssh connect to raspberry pi

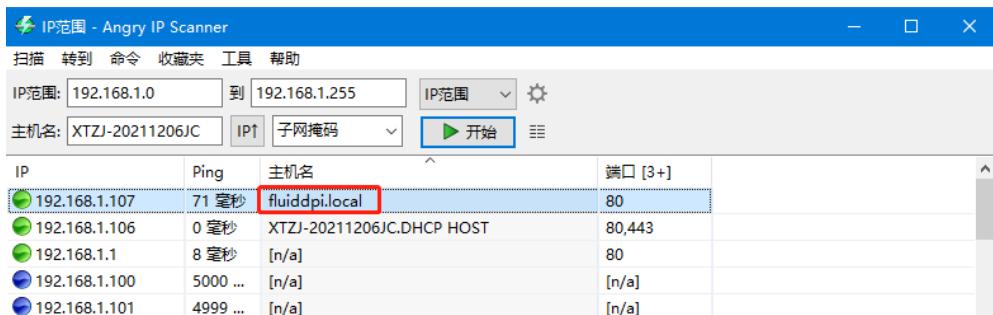
1. Install the ssh application Mobaxterm: <https://mobaxterm.mobatek.net/download-home-edition.html>
2. Insert SD card to raspberry pi, wait for system to load after power one, approx. 1-2min
3. The raspberry pi will automatically be assigned a IP address after successfully connected to the network
4. Find the raspberry pi IP address in your router page



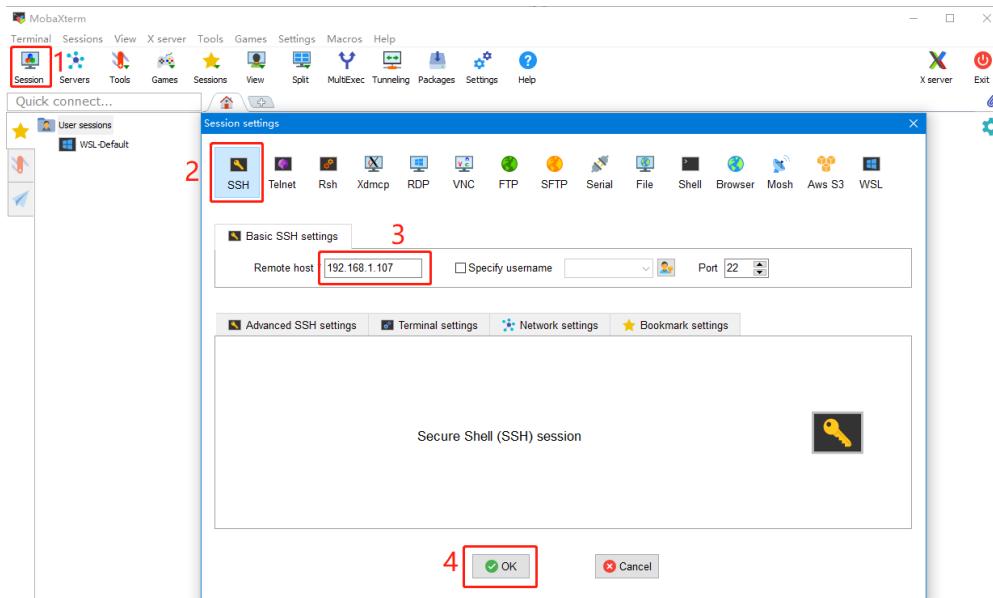
Shenzhen Big Tree Technology CO.,LTD .

BIG TREE TECH

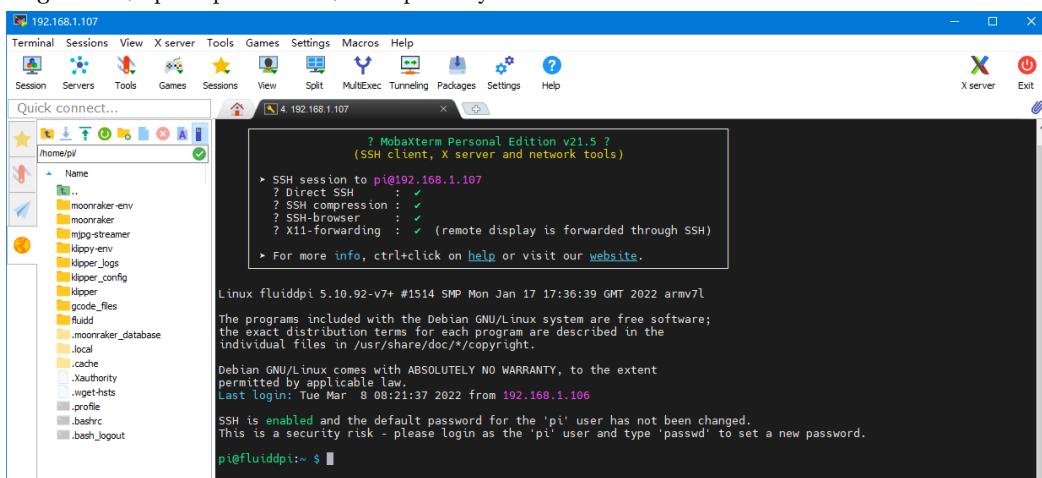
5. Or use the <https://angryip.org/> tool, scan all IP address in the current network organize by names, find the IP named Fluidd or Mailsail like shown below



6. Open Mobaxterm and click “Session”, and click “SSH”, inset the raspberry pi IP into Remote host and click “OK” (note: your computer and the raspberry pi needs to be in the same network)



7. login as: pi password: raspberry



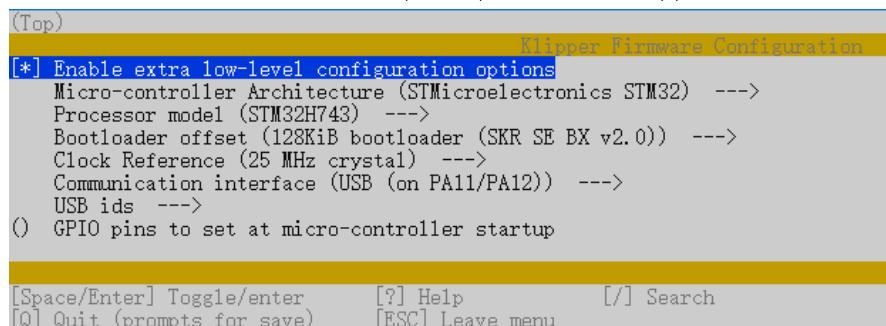
5.5 Compile firmware

1. After ssh successfully connected to the raspberry pi, enter in terminal:

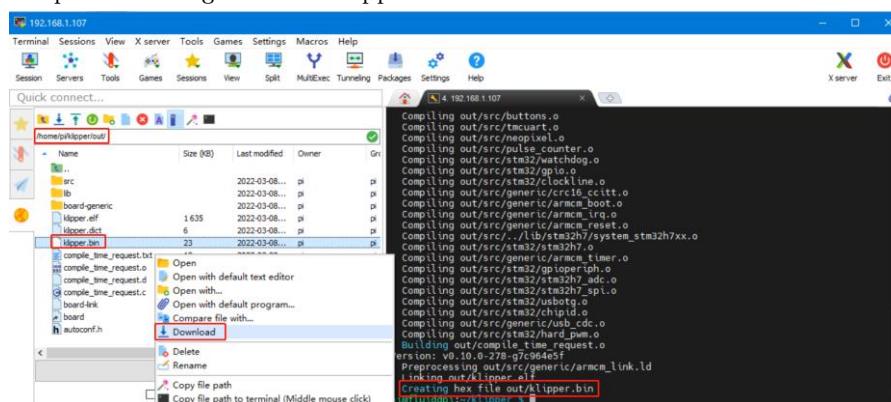
```
cd ~/klipper/
make menuconfig
```

Compile with the configuration shown below(if the options below is not available, please update you Klipper source code to the newest version)

```
* [*] Enable extra low-level configuration options
* Micro-controller Architecture (STMicroelectronics STM32) -->
* Processor model (STM32H743) -->
* Bootloader offset (128KiB bootloader (SKR SE BX v2.0)) -->
* Clock Reference (25 MHz crystal) -->
* Communication interface (USB (on PA11/PA12)) -->
```



2. Press **q** to exit, and **Yes** when asked to save the configuration
3. Run **make** to compile firmware, “klipper.bin” file will be generated in **home/pi/klipper/out** folder when **make** is finished, download it onto your computer using the ssh application.



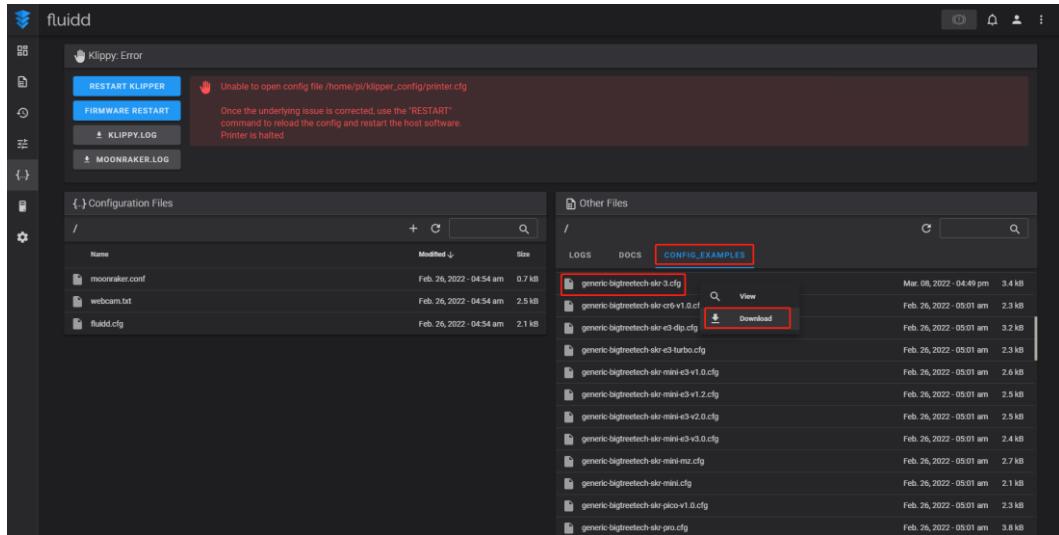
4. Rename klipper.bin to “firmware.bin”, copy to SD card to update firmware
5. Enter: **ls /dev/serial/by-id/** in command line to check motherboard ID to confirm whether firmware is updated successfully like showm below.

```
pi@fluidpi:/klipper $ ls /dev/serial/by-id/
usb-Klipper_stm32h743xx_41003D001751303232383230-if00
pi@fluidpi:~/klipper $
```

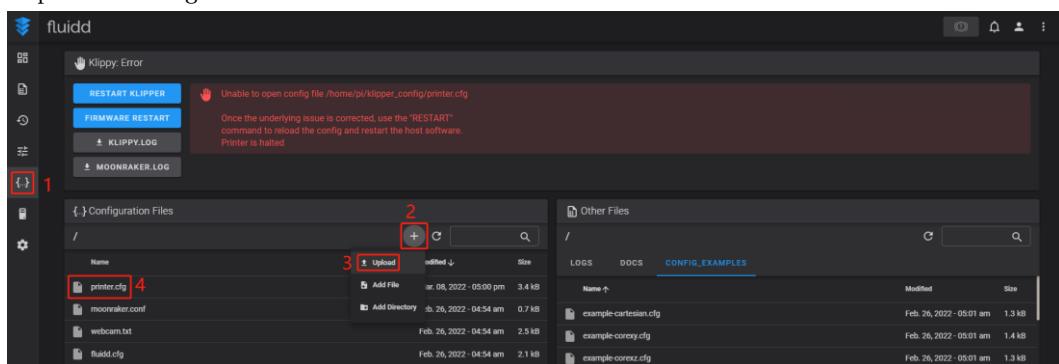
copy and save this ID, it is needed when modifying klipper config

5.6 Configure Klipper

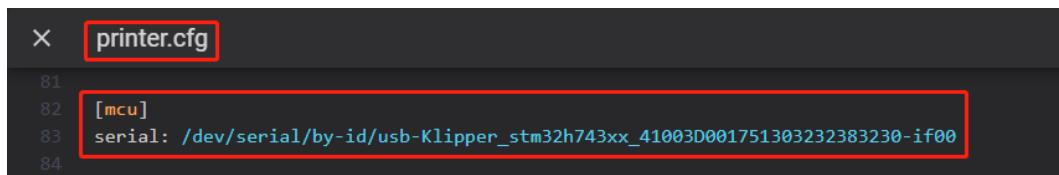
1. Enter your raspberry pi IP address into your browser to open the webUI, find the reference config for motherboard in the directory shown below, if there is no such config available, update your klipper source code to the newest version or download from github:<https://github.com/bigtreeTech/SKR-3>



2. Upload your finished config file into Configuration Files, and rename to "printer.cfg"



3. Insert the correct motherboard ID



4. Refer to <https://www.klipper3d.org/Overview.html> for detailed configuration guide according to your machine type.

6. Firmware update

Update using Micro SD

1. Make sure Micro SD is formatted to FAT32
2. Rename your firmware file to “firmware.bin” (**note:** make sure your system is showing file suffix, if suffix is hided, “firmware.bin” will be shown as “firmware”)
3. Copy “firmware.bin” to the root directory of your SD card.
4. Insert Micro SD to the motherboard and power on, the bootloader will automatically update the firmware
5. The status indicator led will flash during the update process
6. When the led stops flashing and the firmware.bin file has been renamed to firmware.cur, the firmware has been successfully updated.

7. Precautions

1. When not using PT1000, do not connect any jumper in the PT1000 pins, otherwise 100K NTC will not work.
2. Maximum heated bed current is 10A, if high power heated bed is preferred, please use 24v to power the system and use a 24v heated bed.
3. CNC fan voltage selecting jumper must be inserted for CNC fan port to work correctly
4. If the board is plugged into your computer and not responding, make sure the USB/CAN selecting button is released and set to USB mode.
5. The Micro SD card slot is not spring loaded, please be careful when inserting the Micro SD card to prevent damage to the card slot. BTT is not responsible for any damage caused by forcefully inserting the Micro SD card

8. FAQ

Q: Max current of heated bed, heater cartridge, fan port?

A: heated bed: 10A continuous, 11A instantaneous

heater cartridge: 5.5A continuous, 6A instantaneous

Fan port: 1A continuous, 1.5A instantaneous

combined current of heated bed, heater cartridge and fan port should not exceed 10A

Q: Can not update firmware with SD card

A: make sure your sd card is formatted to FAT32, firmware file name is “firmware.bin”, make sure your system is showing file suffix, if suffix is hided, “firmware.bin” will be shown as “firmware”