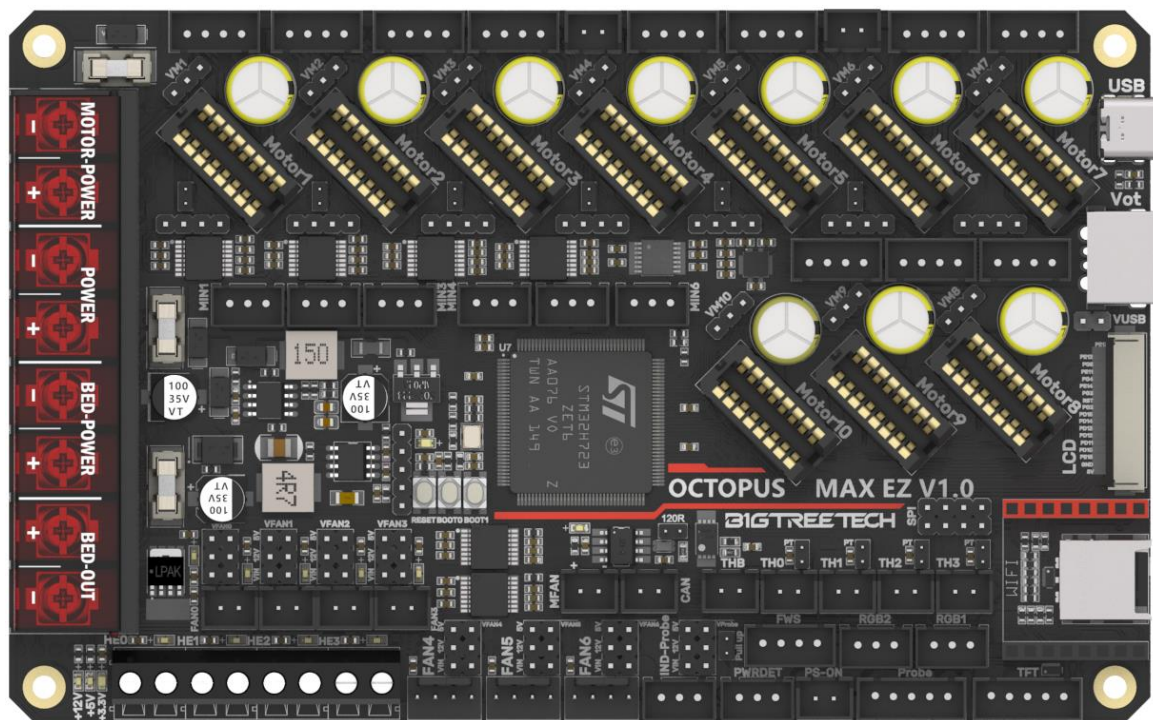


BIGTREETECH

Octopus MAX EZ

V1.0

User Manual



CONTENTS

Revision History	4
Product Profile	5
Feature Highlights	5
Specifications	6
Dimensions	7
Peripheral Port	7
Connector Diagram	7
Pinout Diagram	8
Connection Description	8
USB Power Supply	8
Stepper Motor Driver	9
UART/SPI Mode of EZ Driver	9
TMC Driver DIAG(Sensorless Homing)	9
Driver Voltage Selection	9
Voltage Selection for CNC Fan	9
100K NTC or PT1000 Setting	10
BLTouch Wiring	10
Auto Power Off (Relay V1.2) Wiring	11
Connecting with MINI12864/TFT Screen	11
RGB Wiring	12
Filament Sensor Wiring	12
Proximity Switch Wiring	13
Wiring of 4 pins CNC Fan and Water Cooling System	14
Marlin	15
Install Compiling Environment	15
Download Marlin Firmware	15
Configure Firmware	15
Open Marlin Project	15
Compiling Environment	15

Configure Motherboard and Serial Port	16
Configure Stepper Driver	17
Sensorless Homing	18
100K NTC or PT1000	19
BLTouch	19
Auto Power Off(Relay V1.2)	22
Power Loss Recovery	22
RGB	23
Filament Sensor	24
Smart Filament Sensor (SFS V1.0)	25
ESP3D	26
Compile Firmware	27
Klipper	28
Preparation	28
Download OS Image	28
Download and Install Raspberry Pi Imager	28
Write Image	29
WiFi Setting	31
SSH Connect to Raspberry Pi	31
Compile Firmware	33
Configure Klipper	34
Firmware Update	35
Cautions	35

Revision History

Version	Revisions	Date
01.00	Original	2022/10/06

Product Profile

BIGTREETECH Octopus MAX EZ, a 32 bit motherboard, is an upgraded version of Octopus Pro developed by the 3D printing team of Shenzhen Big Tree Technology Co., Ltd. Its self-developed stepper motor sockets enhance safety and user experience, and it adds a series of features that Octopus Pro does not have, greatly enhancing its DIY capabilities.

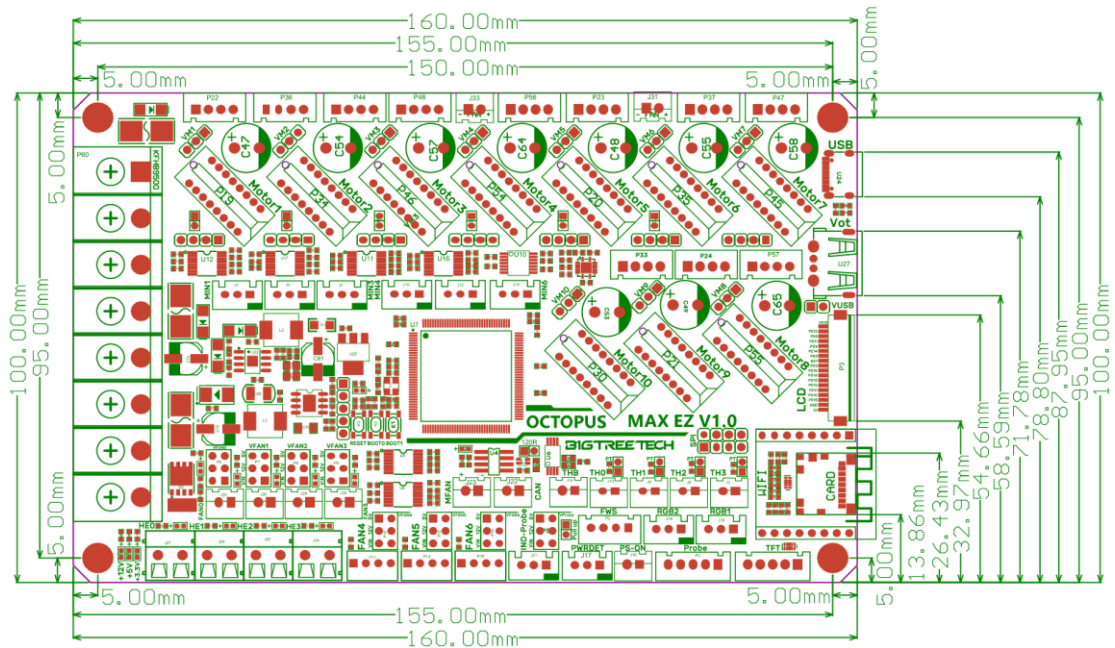
Feature Highlights

1. 32 bit 550 MHz ARM Cortex-M7 series STM32H723ZET6 MCU;
2. Onboard BOOT button to enable DFU mode to update bootloader;
3. The thermistor circuit is protected to prevent MCU damage from shorted heated bed and heater cartridge connections;
4. Selectable voltage (24V, 12V, 5V) for CNC fan, no more need for external stepdown thus preventing board damage from user error;
5. Upgraded with eFuse protection, which responses faster with strong protection, effectively protecting the motherboard from being damaged caused by short circuits, over-current, electric spark, etc.
6. MCU firmware can be upgraded via SD card, or use DFU via Klipper's make flash command;
7. 10 EZ driver sockets, working with pinless driver, safer to use; Onboard SPI and UART, can be used by simply setting in the firmware, no need for a jumper.
8. Support power loss recovery, filament runout sensor, CAN, auto power-off, BLTouch, RGB, etc;
9. Replaceable fuse for easy maintenance;
10. 3 x 4 pins fan ports, also for connecting water cooling system;
11. Onboard proximity switch port, supports NPN and PNP types, 24V, 12V, 5V voltage selectable;
12. Onboard SPI interface for connecting acceleration sensor to enable Klipper's input shaping.

Specifications

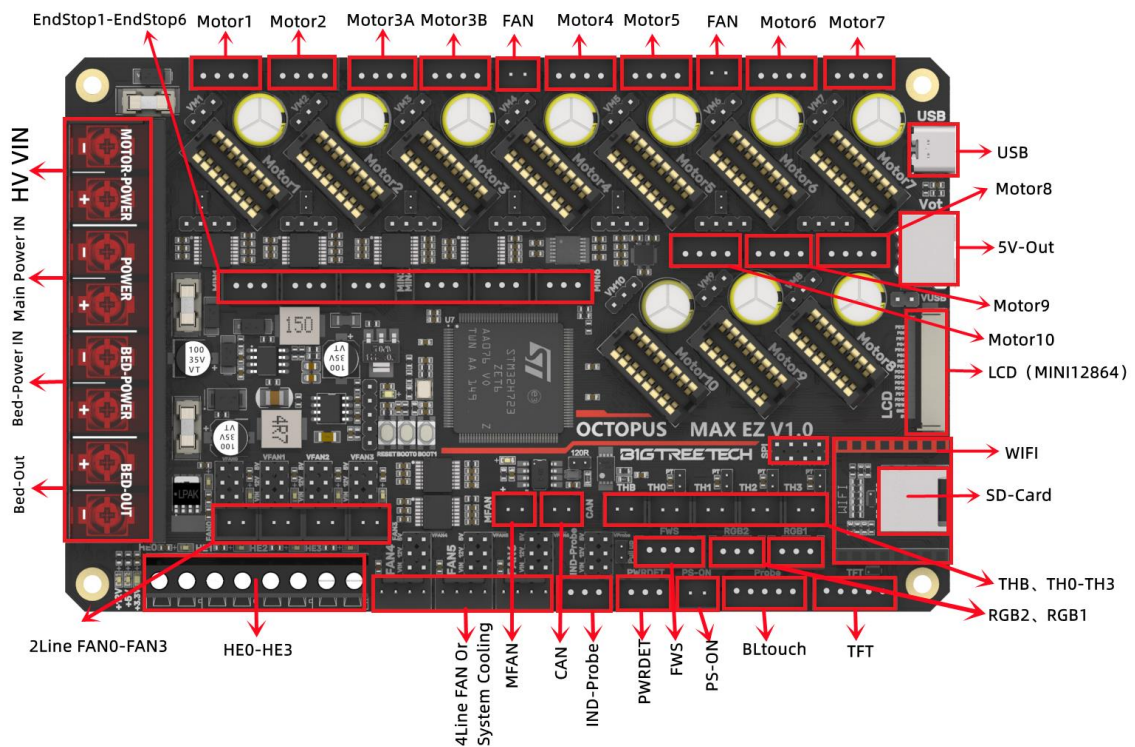
Dimensions	160mm x 100mm for details please refer to BIGTREETECH Octopus MAX EZ V1.0-SIZE.pdf
Mounting Size	Please refer to BIGTREETECH Octopus MAX EZ V1.0-SIZE.pdf
MCU	ARM Cortex-M7 STM32H723ZET6 550MHz
Driver	24V, HV($\leq 56V$) Selectable
Input Voltage	
Motherboard	VIN=DC12V or DC24V
Input Voltage	
Heated Bed	BED IN=DC12V or DC24V
Input Voltage	
Logic Voltage	DC 3.3V
Heater Connection	Heated Bed (HB), Heater Cartridge (HE0, HE1, HE2, HE3)
HB Port Max Current	10A Continuous, 12A Instantaneous
Heater Cartridge	5.5A Continuous, 6A Instantaneous
Max Current	
Fan Port	2 pins CNC Fan (FAN0, FAN1, FAN2, FAN3), 4 pins CNC Fan (FAN4, FAN5, FAN6), Always On (24V FAN x 2). CNC Fan and MFAN Voltage Selectable (5/12/24V)
Fan Port Max Current	1A Continuous, 1.5A Instantaneous
Overall Max Current (Heater Cartridge+ Driver+All Fans)	< 12A
Expansion Port	BLTouch (Servos, Probe), PS-ON, FWS, PWRDET, RGBx2, SPI, IND-Probe, CAN, WIFI, TFT
Motor Driver	Support EZ5160, EZ2209, EZ2225, EZ2226, EZ2208, EZ2130...
Driver Mode	SPI, UART
Motor Socket	Motor1, Motor2, Motor3 (Dual Motor Sockets), Motor4, Motor5, Motor6, Motor7, Motor8, Motor9, Motor10 10 Channels in Total
Thermistor	5 x 100K NTC, four of which are selectable for NTC and PT1000
Display	MINI12864 (FPC Connection), TFT Serial
PC Connection	Type-C
Supported Kinematics	Cartesian, Delta, Kossel, Ultimaker, CoreXY
Recommended Slicer/Console	Cura, Simplify3D, Pronterface, Repetier-host, Makerware

Dimensions

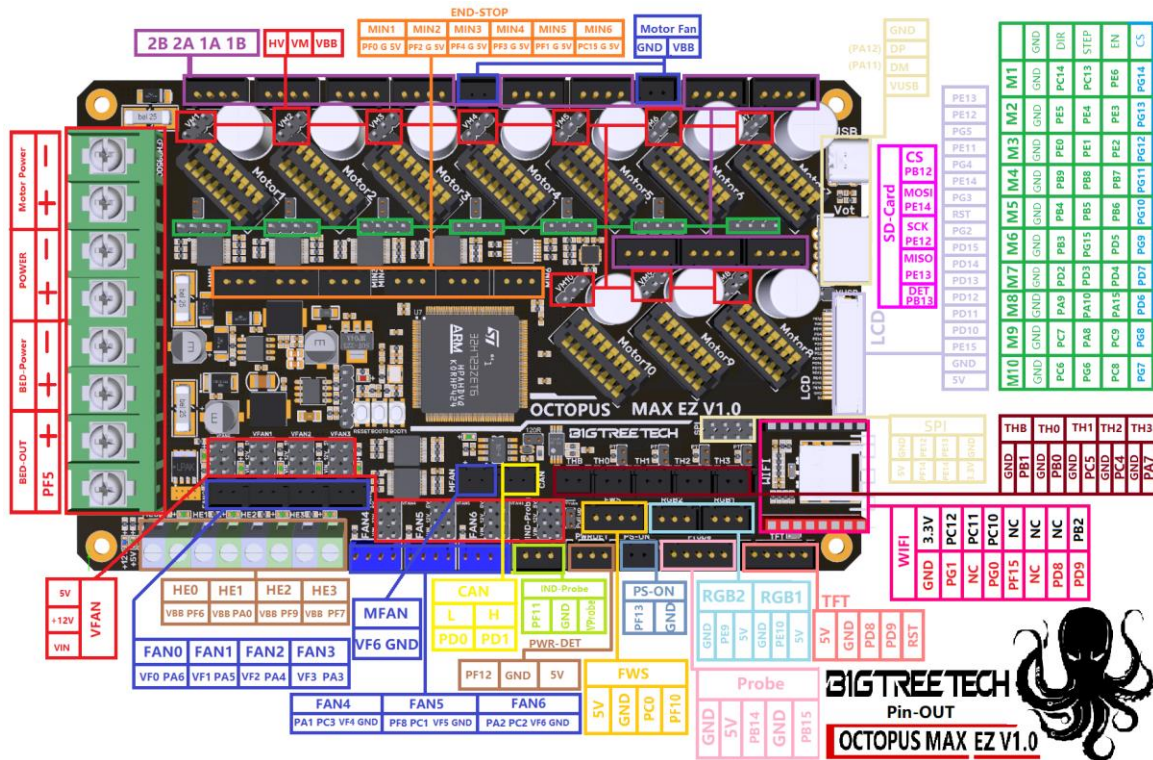


Peripheral Port

Connector Diagram



Pinout Diagram



Connection Description

USB Power Supply

After the Octopus MAX EZ has been powered, the Red light D32 on the left side of the MCU will light up, indicating power on. When using only USB to power the board or to supply power via USB, please insert the jumper cap onto the VUSB.



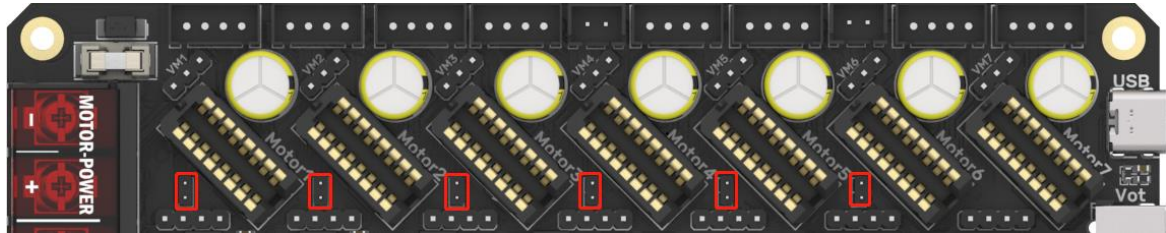
Stepper Motor Driver

UART/SPI Mode of Driver

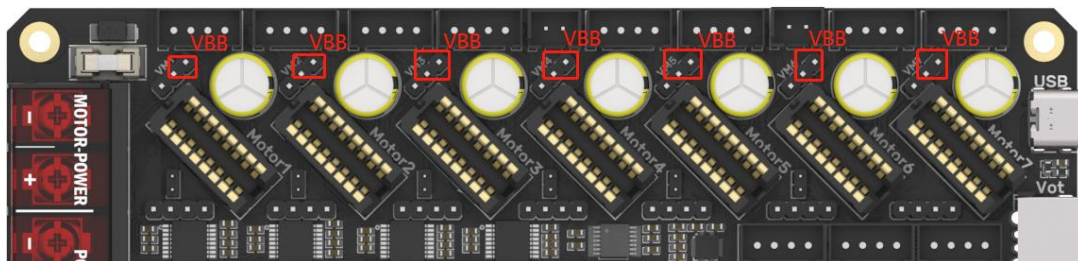
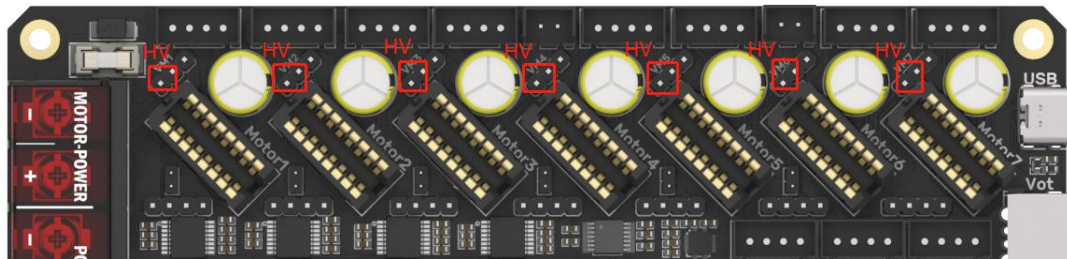
Set in the firmware, no need for a jumper.

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. (Motor1-Motor6).



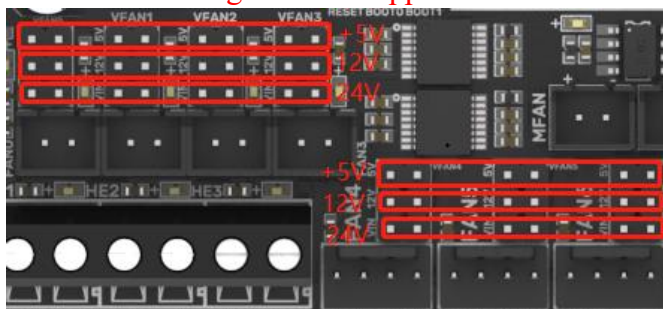
Driver Voltage Selection



Voltage Selection for CNC Fan

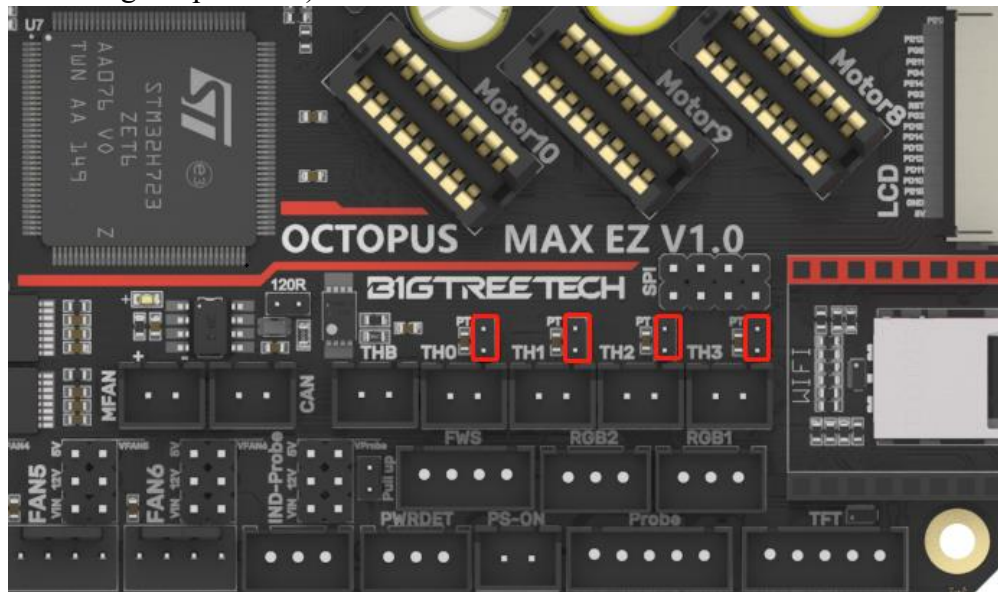
The output voltage can be set to 5V, 12V or 24V through a jumper cap. (MFAN and FAN6 share the power supply VFAN6).

Note: we are not responsible for fan burnout caused by incorrect voltage selection. Please confirm the voltage the fan supports before selecting the voltage.

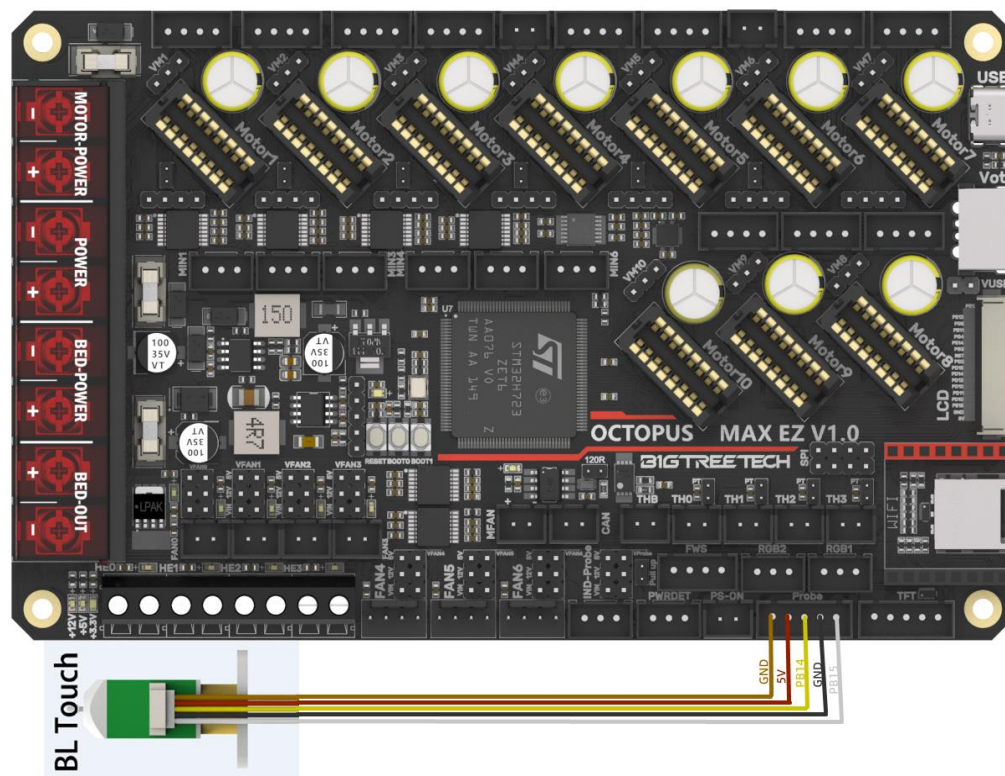


100K NTC or PT1000 Setting

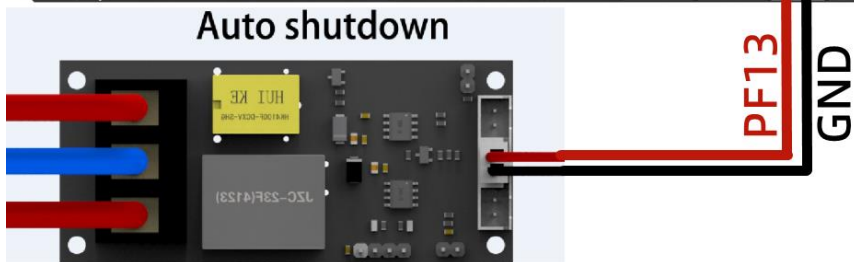
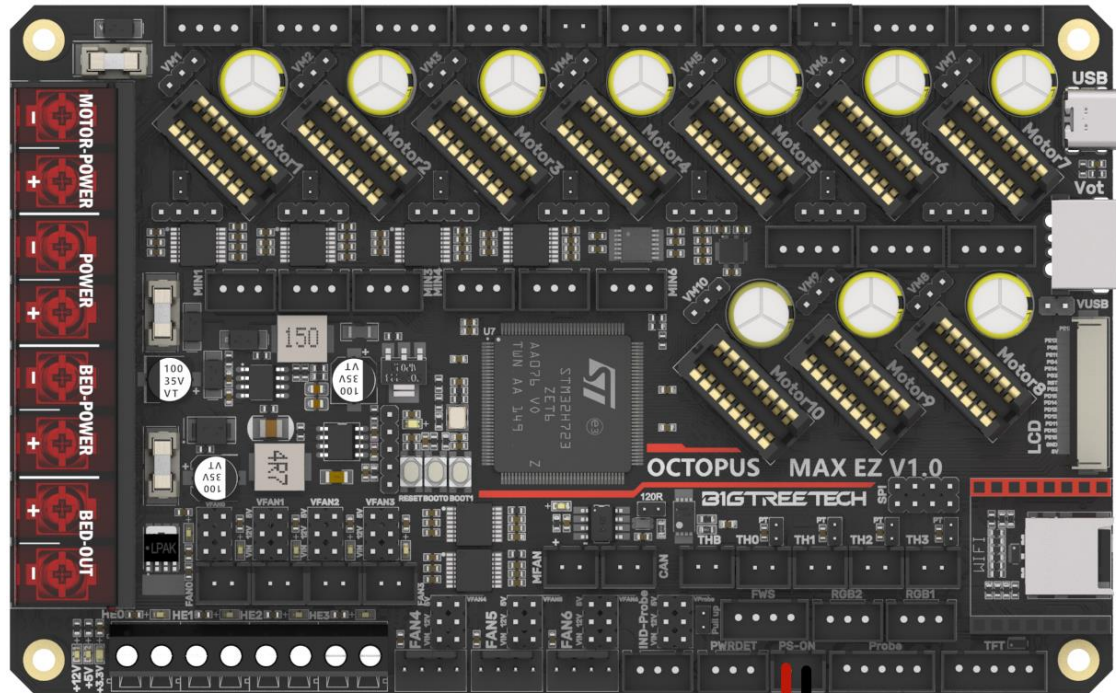
When using 100K NTC, no jumpers need to be connected, the pull-up resistance of TH0-TH3 is 4.7K 0.1%. When using PT1000, the pins indicated in the picture below need to be connected via jumpers, parallel connection of 4.12K 0.1% resistors, the pull-up resistance of TH0-TH1 is 2.2K. (**Note:** this method has a much lower accuracy than the MAX31865 in reading temperature.)



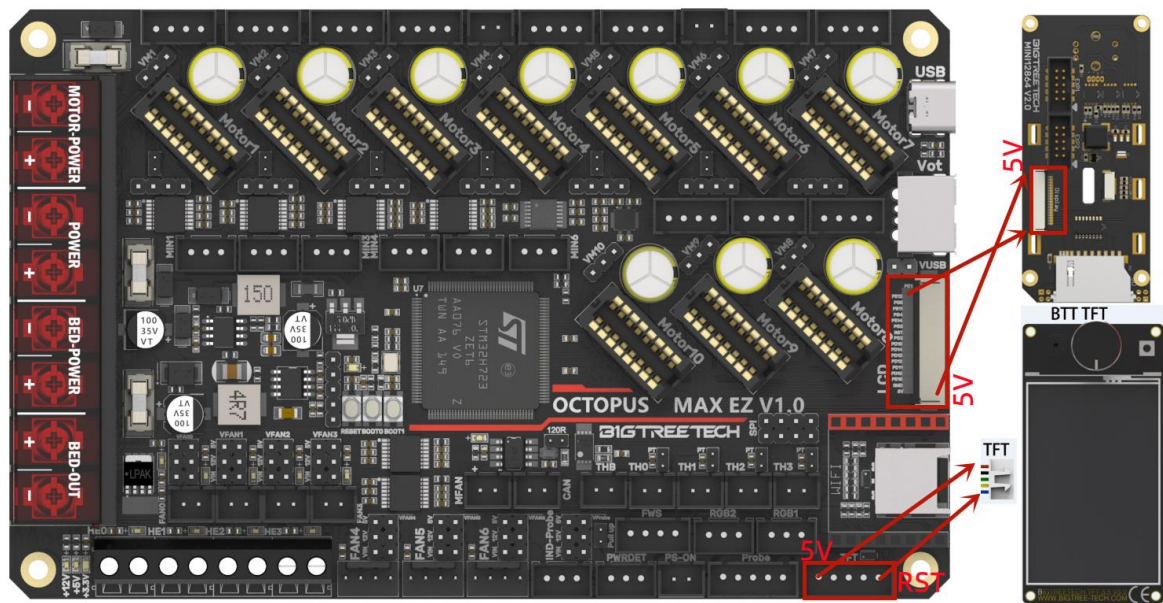
BLTouch Wiring



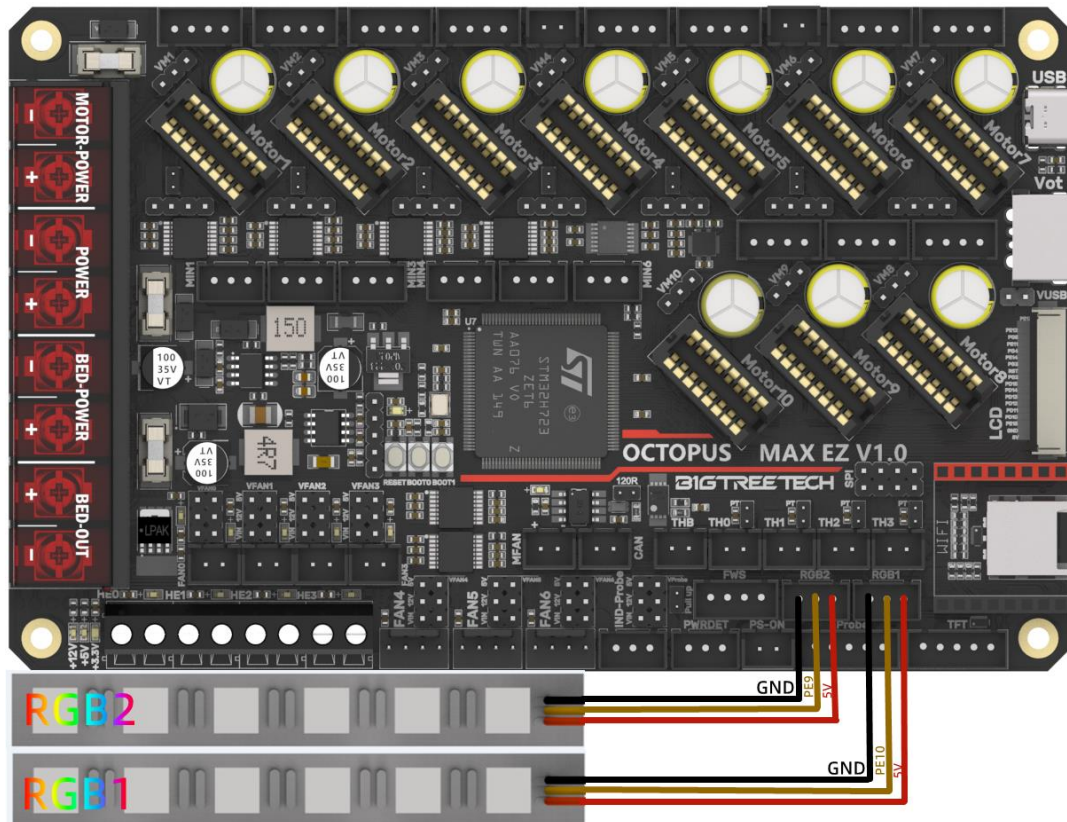
Auto Power Off (Relay V1.2) Wiring



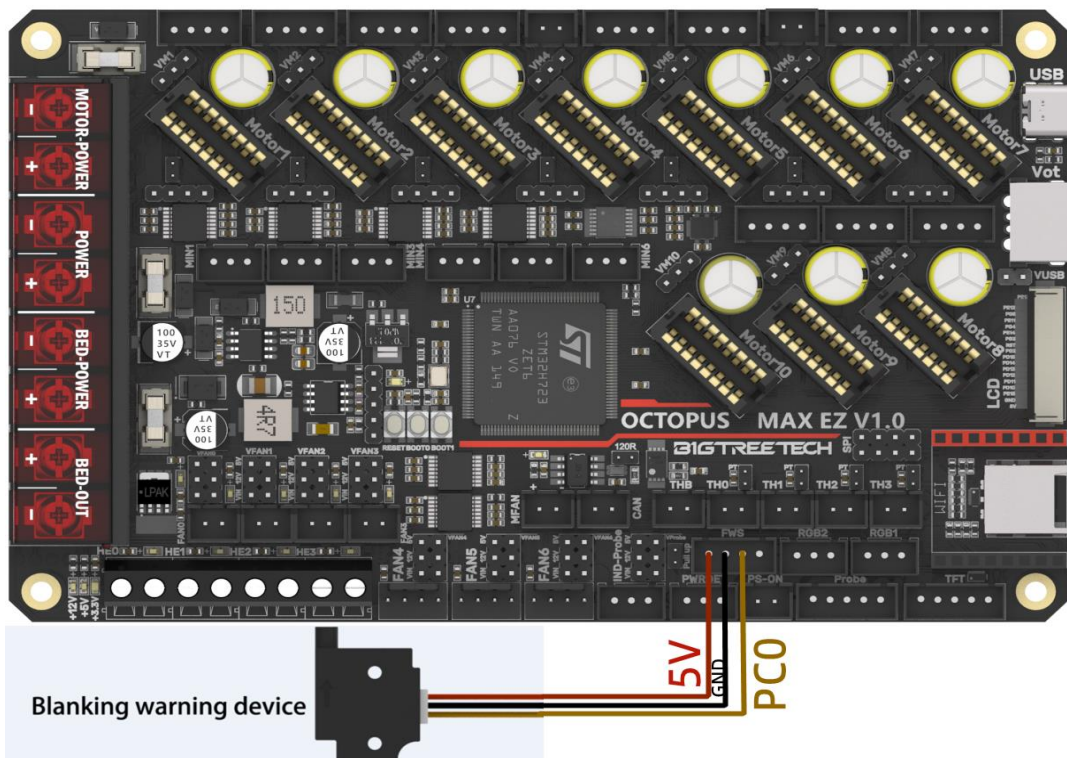
Connecting with MINI12864/TFT Screen



RGB Wiring

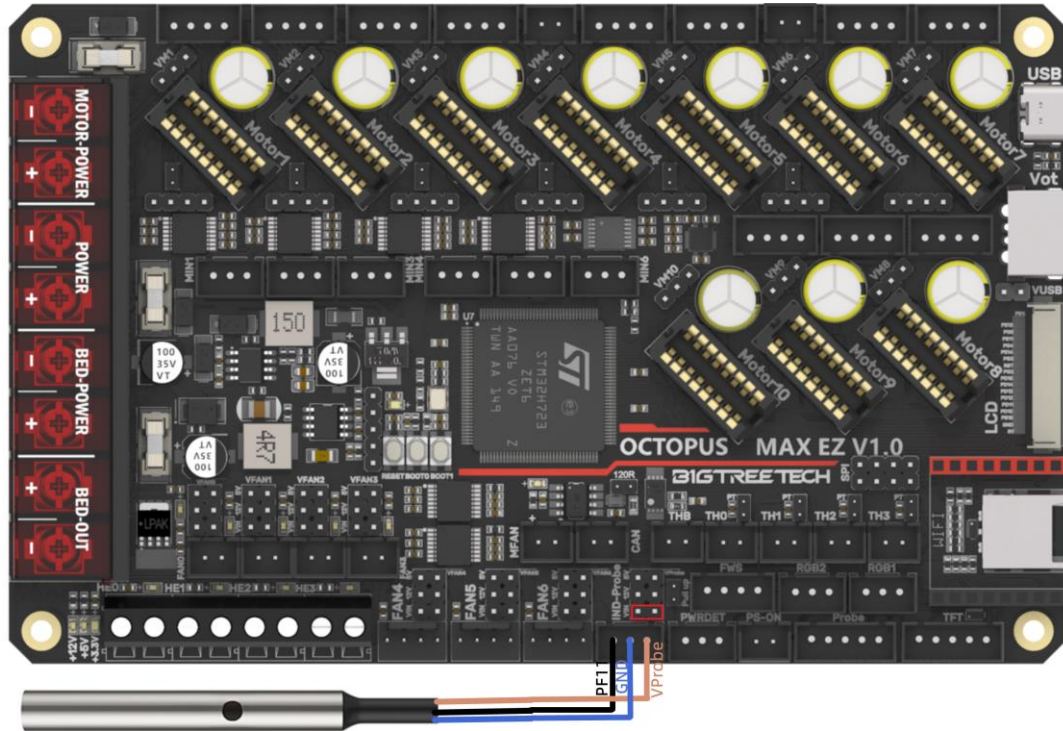


Filament Sensor Wiring

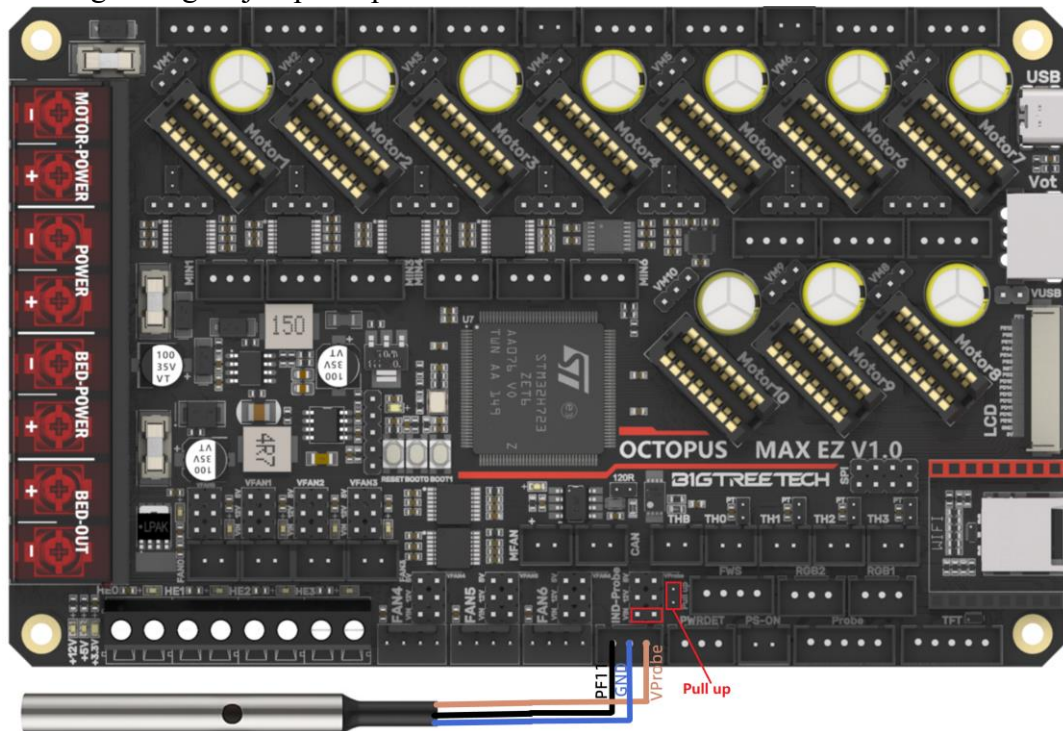


Proximity Switch Wiring

As shown in the figure below, 24V as an example, normally open (NPN type), no need for shorting through a jumper cap:

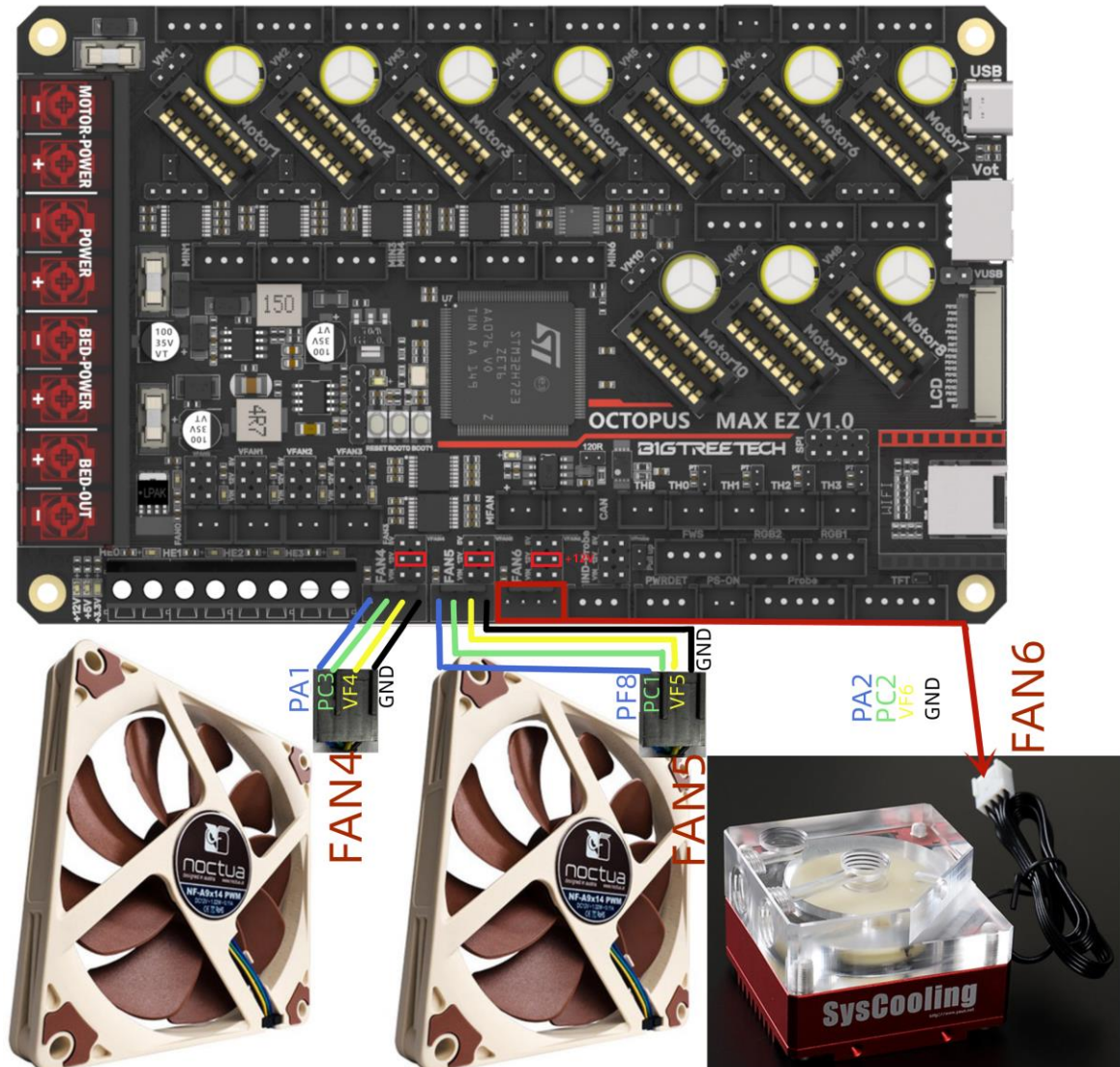


As shown in the figure below, 24V as an example, normally closed (PNP type), need for shorting through a jumper cap.



Wiring of 4 pins CNC Fan and Water Cooling System

(12V as an example:)



Marlin

Install Compiling Environment

<https://github.com/bigtreotech/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.

Download Marlin Firmware

1. Download the newest bugfix version of Marlin from the official website:
<https://github.com/MarlinFirmware/Marlin/tree/bugfix-2.0.x>
2. Download pre-configured firmware from our GitHub page:
<https://github.com/bigtreotech/BIGTREETECH-OCTOPUS-Max-EZ>

Configure Firmware

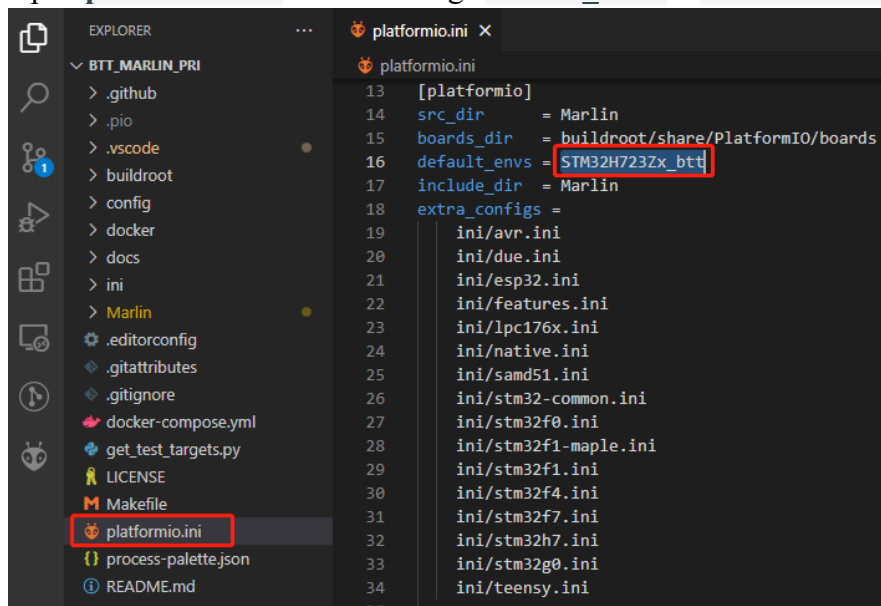
Open Marlin Project

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.

Compiling Environment

Open **platformio.ini** file and change **default_envs** to **STM32H723Zx_btt**.



```
platformio.ini
13 [platformio]
14 src_dir = Marlin
15 boards_dir = buildroot/share/PlatformIO/boards
16 default_envs = STM32H723Zx_btt
17 include_dir = Marlin
18 extra_configs =
19   ini/avr.ini
20   ini/duemilanove.ini
21   ini/esp32.ini
22   ini/features.ini
23   ini/lpc176x.ini
24   ini/native.ini
25   ini/samd51.ini
26   ini/stm32-common.ini
27   ini/stm32f0.ini
28   ini/stm32f1-maple.ini
29   ini/stm32f1.ini
30   ini/stm32f4.ini
31   ini/stm32f7.ini
32   ini/stm32h7.ini
33   ini/stm32g0.ini
34   ini/teensy.ini
```

Configure Motherboard and Serial Port

Set **MOTHERBOARD** to **BOARD_BTT_OCTOPUS_MAX_EZ**

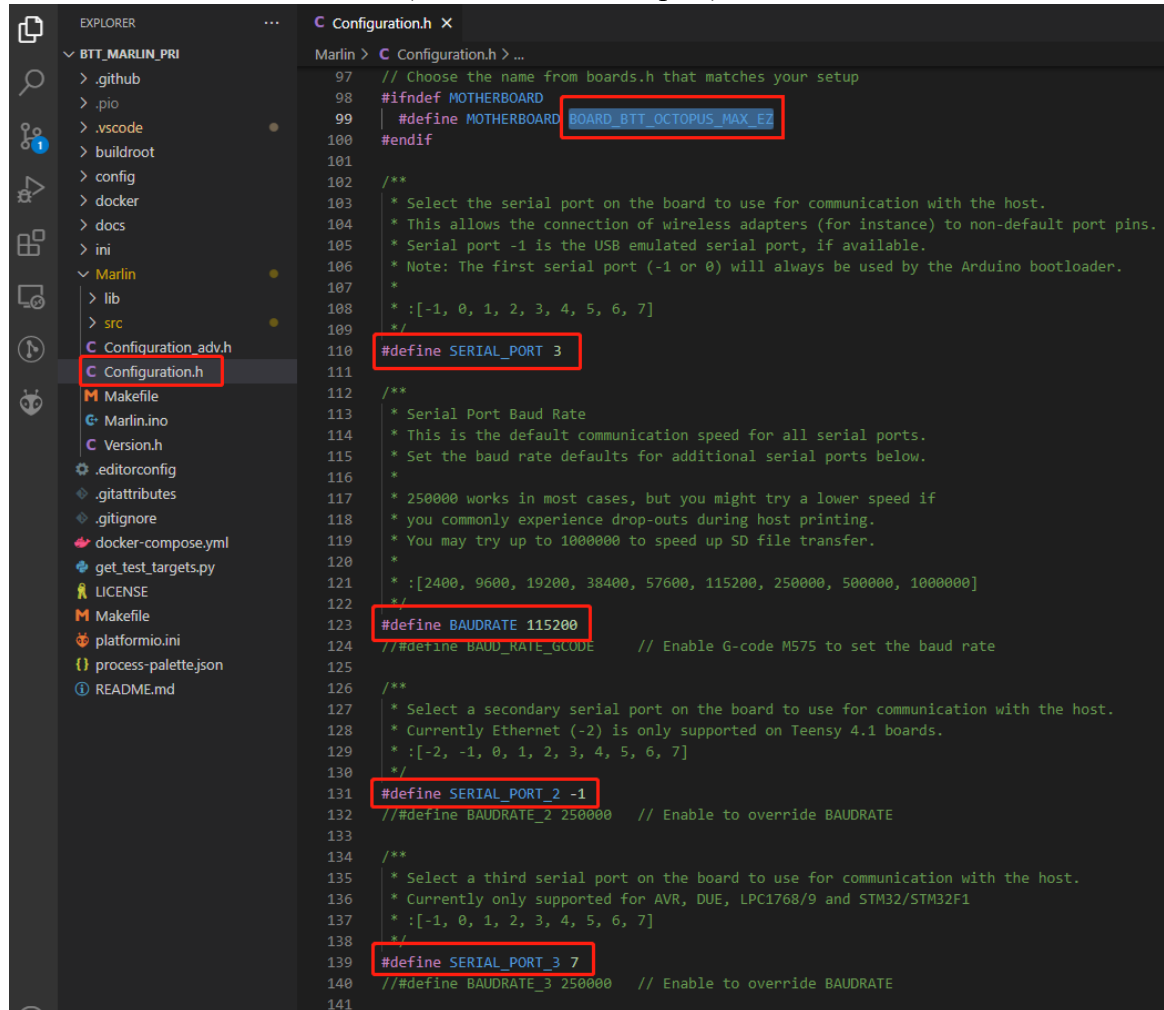
```
#define MOTHERBOARD BOARD_BTT_OCTOPUS_MAX_EZ
```

```
#define SERIAL_PORT 3 (enable TFT serial port)
```

```
#define BAUDRATE 115200 (set baudrate to the same as the communication device)
```

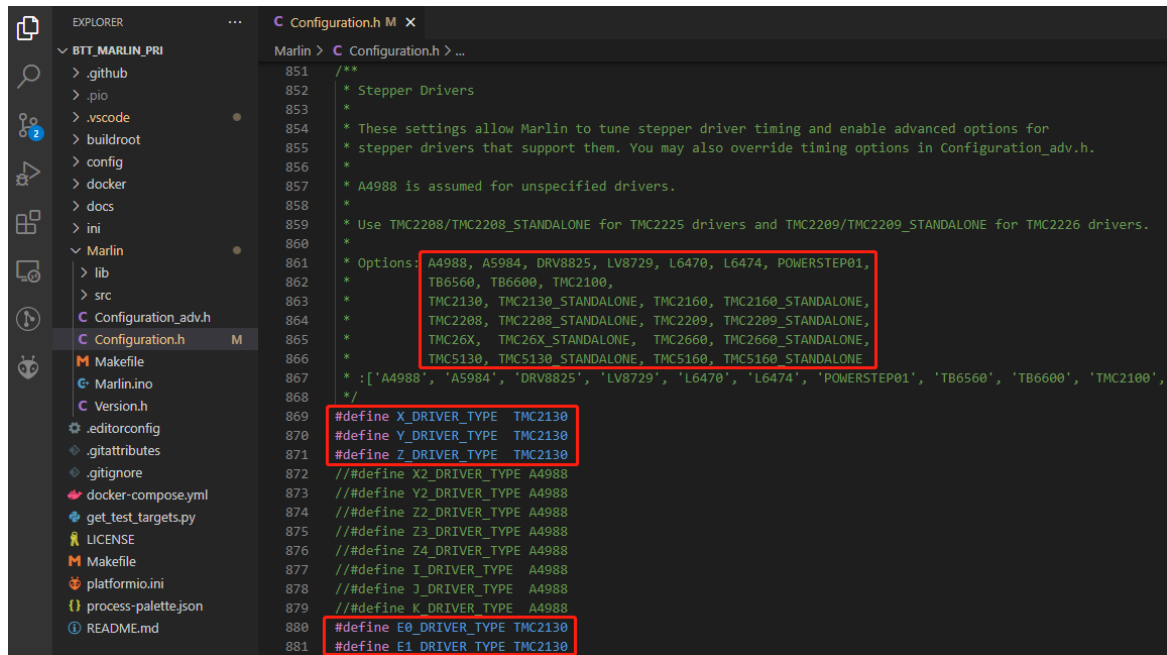
```
#define SERIAL_PORT_2 -1 (enable USB serial port)
```

```
#define SERIAL_PORT_3 7 (enable WIFI serial port)
```



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

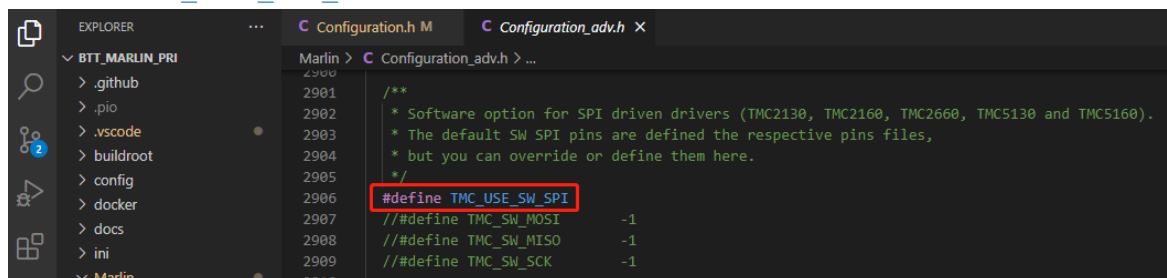

Configure Stepper Driver



```
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  */
868 #define X_DRIVER_TYPE  TMC2130
869 #define Y_DRIVER_TYPE  TMC2130
870 #define Z_DRIVER_TYPE  TMC2130
871 // #define X2_DRIVER_TYPE  A4988
872 // #define Y2_DRIVER_TYPE  A4988
873 // #define Z2_DRIVER_TYPE  A4988
874 // #define Z3_DRIVER_TYPE  A4988
875 // #define Z4_DRIVER_TYPE  A4988
876 // #define I_DRIVER_TYPE  A4988
877 // #define J_DRIVER_TYPE  A4988
878 // #define K_DRIVER_TYPE  A4988
879 #define E0_DRIVER_TYPE TMC2130
880 #define E1_DRIVER_TYPE TMC2130
```

When using SPI mode, you need to enable `TMC_USE_SW_SPI` in `Configuration_adv.h`

`#define TMC_USE_SW_SPI`



```
2901 /**
2902  * Software option for SPI driven drivers (TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160).
2903  * The default SW SPI pins are defined the respective pins files,
2904  * but you can override or define them here.
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
```

Sensorless Homing

```
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
```

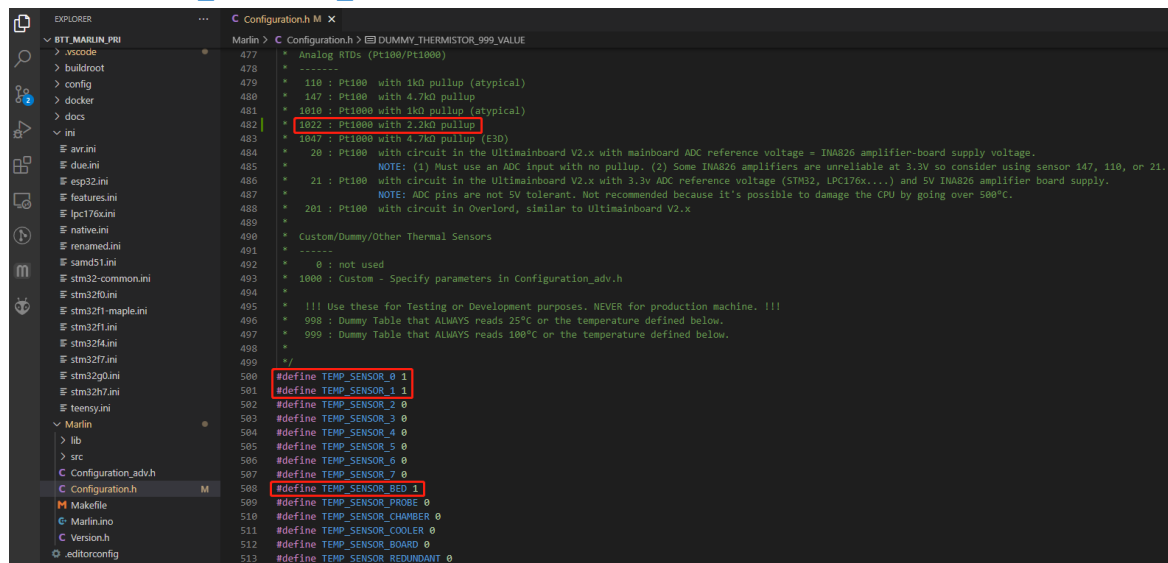
`#define SENSORLESS_HOMING` // enable sensorless homing
`#define xx_STALL_SENSITIVITY 8` // sensitivity setting, TMC2209 range from 0 to 255, higher number results in more sensitive trigger threshold, sensitivity too high will cause endpoint to trigger before gantry actually moves to the end, lower number results in less sensitive trigger threshold, too low of sensitivity will cause endpoint to not trigger and gantrying continue. Other drivers range from 63 to -64, lower numbers result in a 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.

100K NTC or PT1000

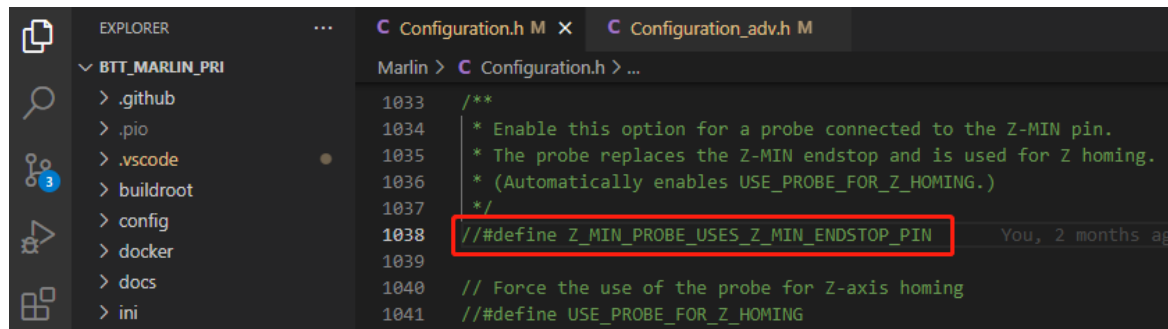
When using 100K NTC, pull-up resistance is 4.7K, when using PT1000, pull-up resistance is 2.2K, set sensor type to 1 for 100K NTC +4.7K pull-up resistance, 1022 for PT1000 + 2.2K pull-up resistance. (Note: this method has a much lower accuracy than the MAX31865 in reading temperature.)

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



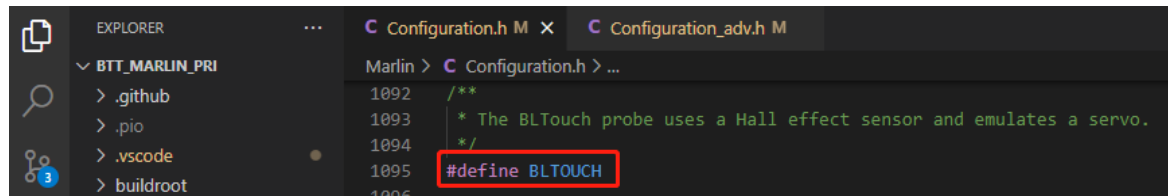
```
477 * Analog RTDs (Pt100/Pt1000)
478 * -----
479 * 110 : Pt100 with 1kΩ pullup (atypical)
480 * 147 : Pt100 with 4.7kΩ pullup
481 * 1810 : Pt1000 with 1kΩ pullup (atypical)
482 * 1022 : Pt1000 with 2.2kΩ pullup
483 * 1047 : Pt1000 with 4.7kΩ pullup (E3D)
484 * 20 : Pt100 with circuit in the Ultimainboard V2.x with mainboard ADC reference voltage = INA826 amplifier-board supply voltage.
485 * 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.
486 * 21 : Pt100 with circuit in the Ultimainboard V2.x with 3.3V ADC reference voltage (STM32, LPC176x...) and 5V INA826 amplifier board supply.
487 * NOTE: ADC pins are not 5V tolerant. Not recommended because it's possible to damage the CPU by going over 500°C.
488 * 201 : Pt100 with circuit in Overlord, similar to Ultimainboard V2.x
489 *
490 * Custom/Dummy/Other Thermal Sensors
491 * -----
492 * 0 : not used
493 * 1000 : Custom - Specify parameters in Configuration_adv.h
494 *
495 * !!! Use these for Testing or Development purposes. NEVER for production machine. !!!
496 * 998 : Dummy Table that ALWAYS reads 25°C on the temperature defined below.
497 * 999 : Dummy Table that ALWAYS reads 100°C on the temperature defined below.
498 *
499 */
500 #define TEMP_SENSOR_0 1
501 #define TEMP_SENSOR_1 1
502 #define TEMP_SENSOR_2 0
503 #define TEMP_SENSOR_3 0
504 #define TEMP_SENSOR_4 0
505 #define TEMP_SENSOR_5 0
506 #define TEMP_SENSOR_6 0
507 #define TEMP_SENSOR_7 0
508 #define TEMP_SENSOR_BED 1
509 #define TEMP_SENSOR_PROBE 0
510 #define TEMP_SENSOR_CHAMBER 0
511 #define TEMP_SENSOR_COOLER 0
512 #define TEMP_SENSOR_BOARD 0
513 #define TEMP_SENSOR_REDUNDANT 0
```

BLTouch



```
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
```

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



```
1092 /**
1093 * The BLTouch probe uses a Hall effect sensor and emulates a servo.
1094 */
1095 #define BLTOUCH
1096
```

```
#define BLTOUCH // Enable bltouch
```

```

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 * 0-- 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

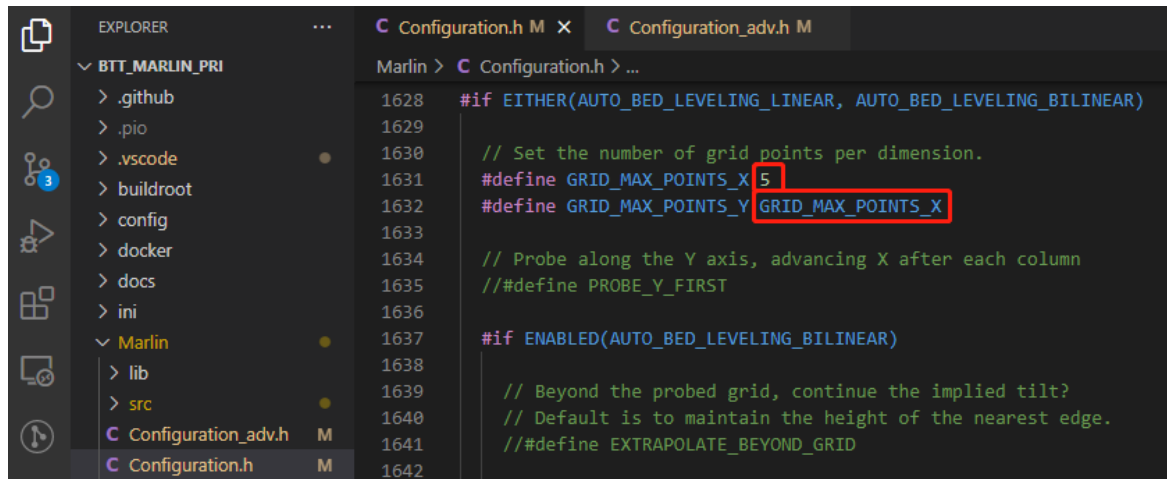
```

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

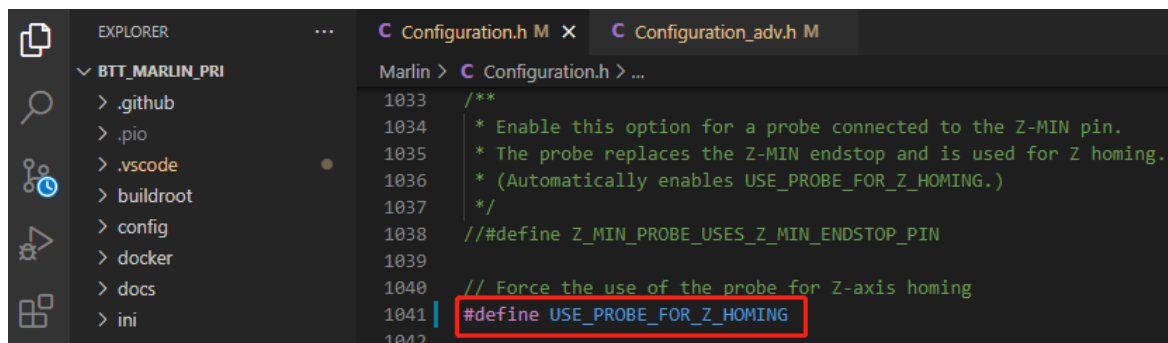


```
1628 #if EITHER(AUTO_BED_LEVELING_LINEAR, AUTO_BED_LEVELING_BILINEAR)
1629
1630 // Set the number of grid points per dimension.
1631 #define GRID_MAX_POINTS_X 5
1632 #define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X
1633
1634 // Probe along the Y axis, advancing X after each column
1635 // #define PROBE_Y_FIRST
1636
1637 #if ENABLED(AUTO_BED_LEVELING_BILINEAR)
1638
1639 // Beyond the probed grid, continue the implied tilt?
1640 // Default is to maintain the height of the nearest edge.
1641 // #define EXTRAPOLATE_BEYOND_GRID
1642
```

`#define GRID_MAX_POINTS_X 5` // set number of probe points for x axis, usually 5 point is sufficient

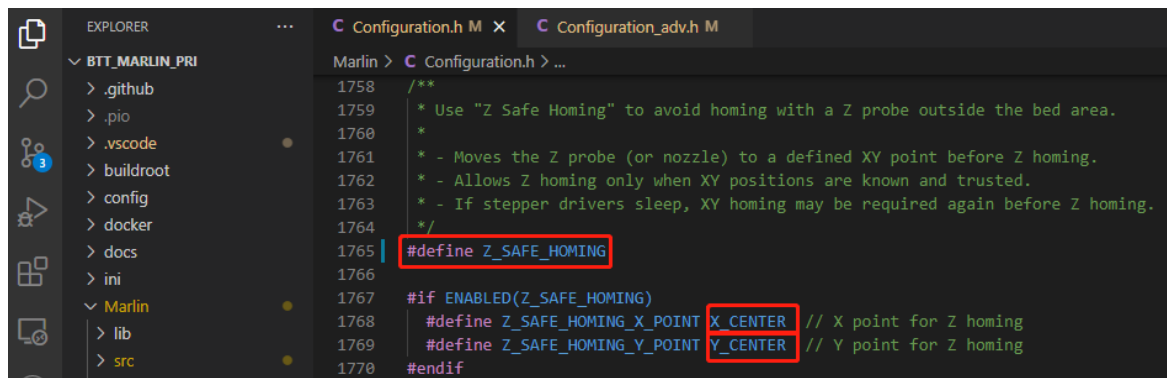
`#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X` // set the number of probe points for Y axis to the same as X axis.

If bltouch also functions as your Z homing sensor, no wiring change is needed, just set it in the firmware.



```
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
1042
```

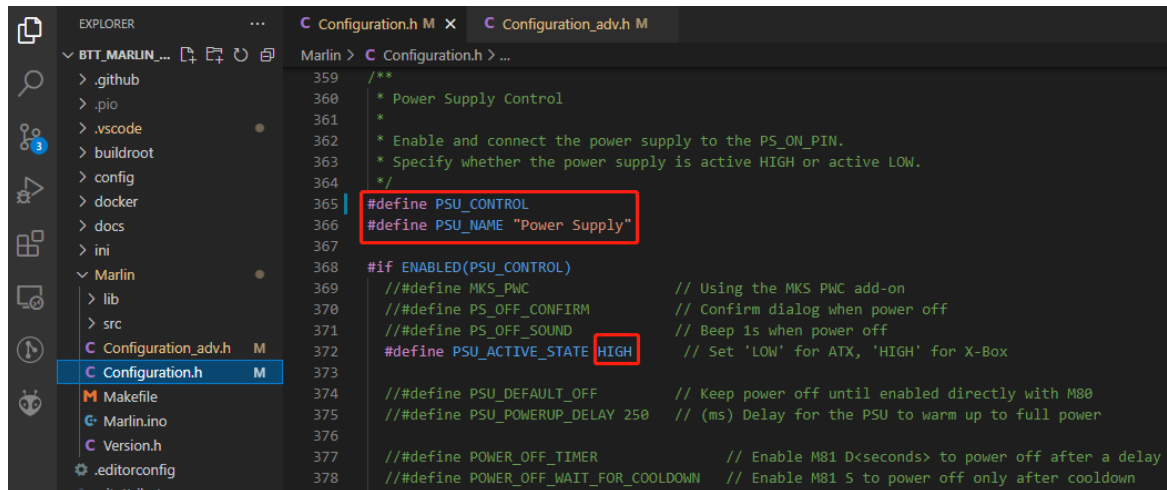
`#define USE_PROBE_FOR_Z_HOMING` // use Z Probe(BLtouch) for Z homing



```
1758 /**
1759  * Use "Z Safe Homing" to avoid homing with a Z probe outside the bed area.
1760  *
1761  * - Moves the Z probe (or nozzle) to a defined XY point before Z homing.
1762  * - Allows Z homing only when XY positions are known and trusted.
1763  * - If stepper drivers sleep, XY homing may be required again before Z homing.
1764  */
1765 #define Z_SAFE_HOMING
1766
1767 #if ENABLED(Z_SAFE_HOMING)
1768 #define Z_SAFE_HOMING_X_POINT X_CENTER // X point for Z homing
1769 #define Z_SAFE_HOMING_Y_POINT Y_CENTER // Y point for Z homing
1770 #endif
```

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

Auto Power Off(Relay V1.2)



```
359 /**
360  * Power Supply Control
361  *
362  * Enable and connect the power supply to the PS_ON_PIN.
363  * Specify whether the power supply is active HIGH or active LOW.
364  */
365 #define PSU_CONTROL
366 #define PSU_NAME "Power Supply"
367
368 #if ENABLED(PSU_CONTROL)
369 // #define MKS_PWC // Using the MKS PWC add-on
370 // #define PS_OFF_CONFIRM // Confirm dialog when power off
371 // #define PS_OFF_SOUND // Beep 1s when power off
372 #define PSU_ACTIVE_STATE HIGH // Set 'LOW' for ATX, 'HIGH' for X-Box
373
374 // #define PSU_DEFAULT_OFF // Keep power off until enabled directly with M80
375 // #define PSU_POWERUP_DELAY 250 // (ms) Delay for the PSU to warm up to full power
376
377 // #define POWER_OFF_TIMER // Enable M81 D<seconds> to power off after a delay
378 // #define POWER_OFF_WAIT_FOR_COOLDOWN // Enable M81 S to power off only after cooldown
```

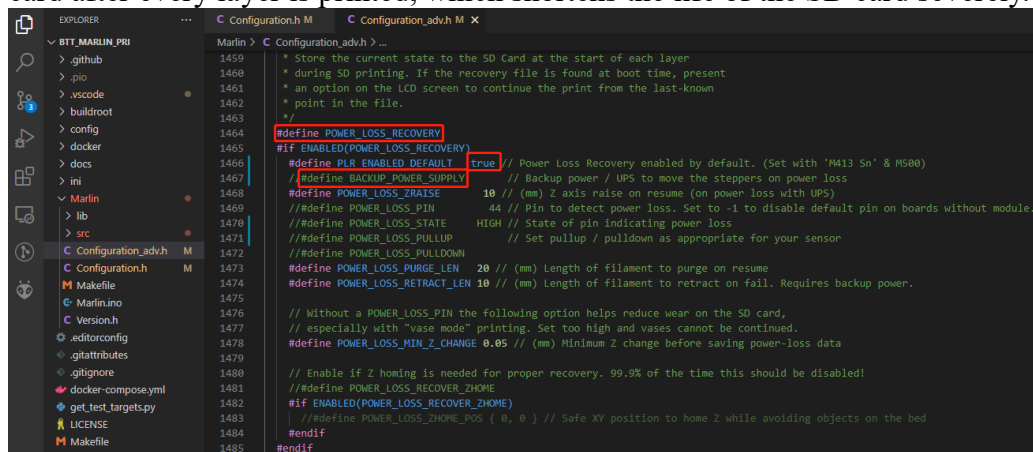
#define PSU_CONTROL // enable PSU control to turn on and off using M80 and M81

#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.

Power Loss Recovery

There are two methods for power loss recovery

1. 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.

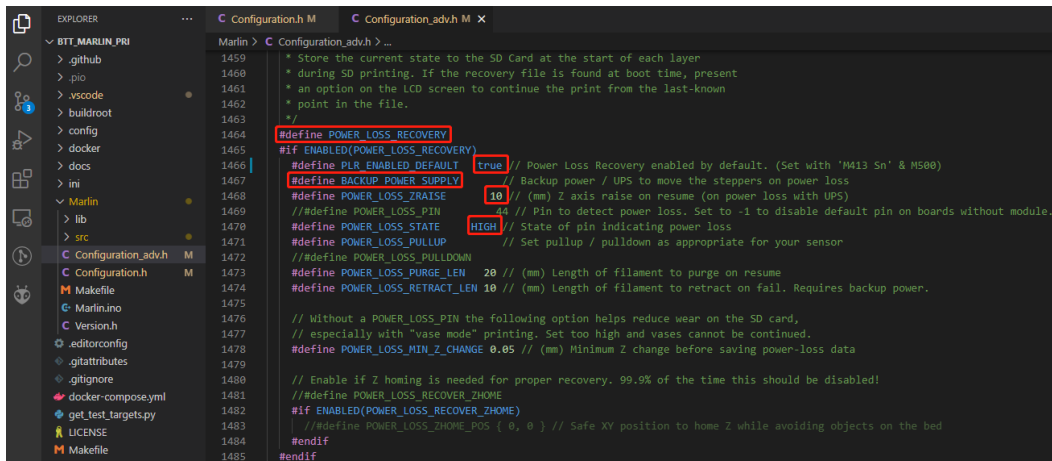


```
1459 * Store the current state to the SD card at the start of each layer
1460 * during SD printing. If the recovery file is found at boot time, present
1461 * an option on the LCD screen to continue the print from the last-known
1462 * point in the file.
1463
1464 #define POWER_LOSS_RECOVERY
1465 #if ENABLED(POWER_LOSS_RECOVERY)
1466 #define PLR_ENABLED_DEFAULT true // Power Loss Recovery enabled by default. (Set with 'M13 Sn' & M500)
1467 #define BACKUP_POWER_SUPPLY // Backup power / UPS to move the steppers on power loss
1468 #define POWER_LOSS_ZRAISE 10 // (mm) Z axis raise on resume (on power loss with UPS)
1469 // #define POWER_LOSS_PIN 44 // Pin to detect power loss. Set to -1 to disable default pin on boards without module.
1470 // #define POWER_LOSS_STATE HIGH // State of pin indicating power loss
1471 // #define POWER_LOSS_PULLUP // Set pullup / pulldown as appropriate for your sensor
1472 // #define POWER_LOSS_PULLDOWN
1473 #define POWER_LOSS_PURGE_LEN 20 // (mm) Length of filament to purge on resume
1474 #define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fail. Requires backup power.
1475
1476 // Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
1477 // especially with "vase mode" printing. Set too high and vases cannot be continued.
1478 #define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power-loss data
1479
1480 // Enable if Z homing is needed for proper recovery. 99.9% of the time this should be disabled!
1481 // #define POWER_LOSS_RECOVER_ZHOME
1482 #if ENABLED(POWER_LOSS_RECOVER_ZHOME)
1483 // #define POWER_LOSS_ZHOME_POS { 0, 0 } // Safe XY position to home Z while avoiding objects on the bed
1484 #endif
1485 #endif
```

#define POWER_LOSS_RECOVERY // enable power loss recovery

#define PLR_ENABLED_DEFAULT true // true default to power loss recovery enabled

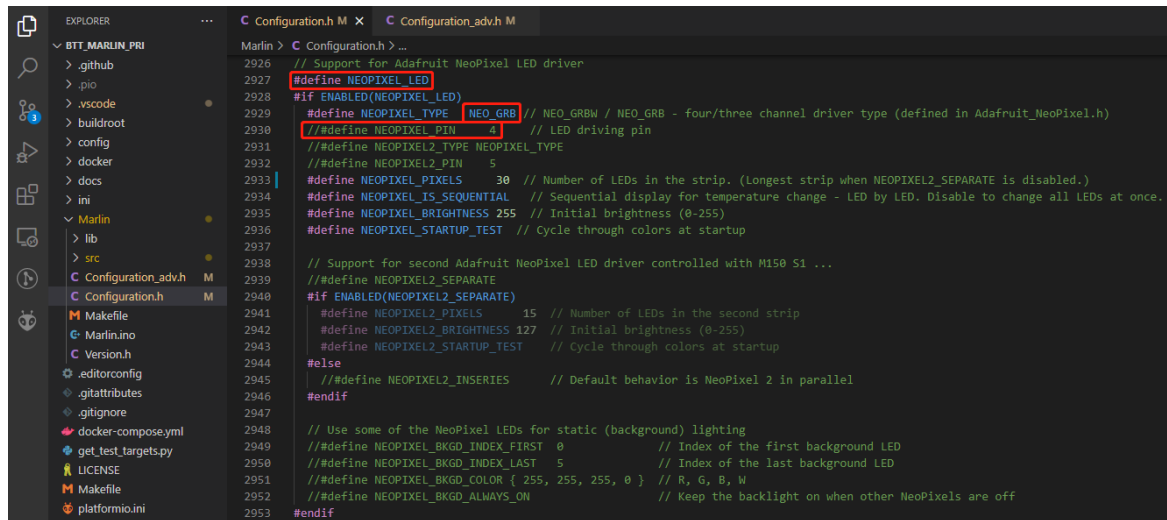
2. External UPS 24V V1.0 module, when power is cut, the module will provide power to the board and signal the board to save current print status to SD card. This method has virtually no effect on the life of the SD card.



```
1459 * Store the current state to the SD Card at the start of each layer
1460 * during SD printing. If the recovery file is found at boot time, present
1461 * an option on the LCD screen to continue the print from the last-known
1462 * point in the file.
1463 */
1464 #define POWER_LOSS_RECOVERY
1465 #if ENABLED(POWER_LOSS_RECOVERY)
1466   #define PLR_ENABLED_DEFAULT true // Power Loss Recovery enabled by default. (Set with 'M13 Sn' & M500)
1467   #define BACKUP_POWER_SUPPLY // Backup power / UPS to move the steppers on power loss
1468   #define POWER_LOSS_ZRAISE 10 // (mm) Z axis raise on resume (on power loss with UPS)
1469   #define POWER_LOSS_PIN 4 // Pin to detect power loss. Set to -1 to disable default pin on boards without module.
1470   #define POWER_LOSS_STATE HIGH // State of pin indicating power loss
1471   #define POWER_LOSS_PULLUP // Set pullup / pulldown as appropriate for your sensor
1472   #define POWER_LOSS_PULLDOWN
1473   #define POWER_LOSS_PURGE_LEN 20 // (mm) Length of filament to purge on resume
1474   #define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fall. Requires backup power.
1475
1476   // Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
1477   // especially with "vase mode" printing. Set too high and vases cannot be continued.
1478   #define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power-loss data
1479
1480   // Enable if Z homing is needed for proper recovery. 99.9% of the time this should be disabled!
1481   #define POWER_LOSS_RECOVER_ZHOME
1482   #if ENABLED(POWER_LOSS_RECOVER_ZHOME)
1483     #define POWER_LOSS_RECOVER_ZHOME_POS { 0, 0 } // Safe XY position to home Z while avoiding objects on the bed
1484   #endif
1485 #endif
```

#define POWER_LOSS_RECOVERY // enable power loss recovery
#define PLR_ENABLED_DEFAULT true // true default to power loss recovery enabled
#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

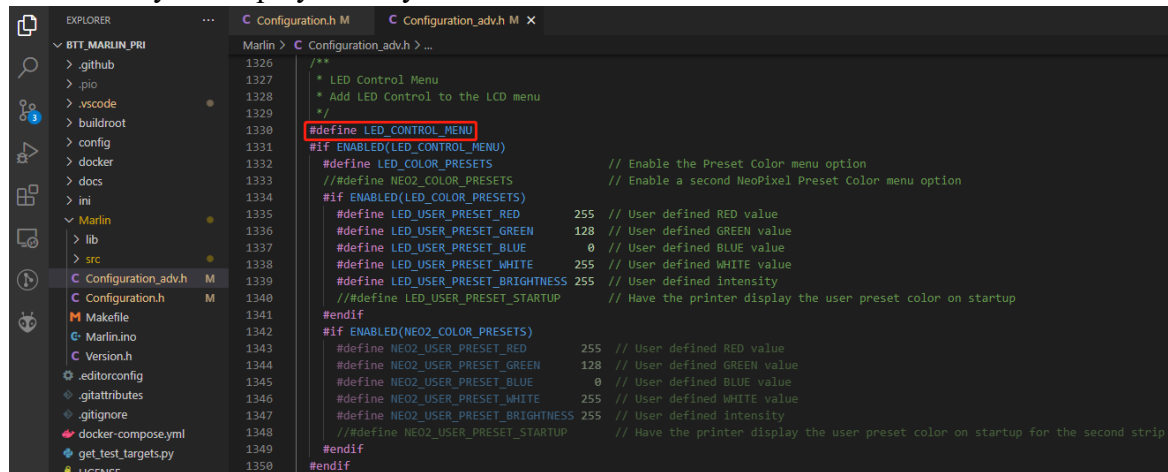
RGB



```
2926 // Support for Adafruit NeoPixel LED driver
2927 #define NEOPIXEL_LED
2928 #if ENABLED(NEOPIXEL_LED)
2929   #define NEOPIXEL_TYPE NEO_GRB // NEO_GRBW / NEO_GRB - four/three channel driver type (defined in Adafruit_NeoPixel.h)
2930   #define NEOPIXEL_PIN 4 // LED driving pin
2931   #define NEOPIXEL2_TYPE NEOPIXEL_TYPE
2932   #define NEOPIXEL2_PIN 5
2933   #define NEOPIXEL_PIXELS 30 // Number of LEDs in the strip. (Longest strip when NEOPIXEL2_SEPARATE is disabled.)
2934   #define NEOPIXEL_IS_SEQUENTIAL // Sequential display for temperature change - LED by LED. Disable to change all LEDs at once.
2935   #define NEOPIXEL_BRIGHTNESS 255 // Initial brightness (0-255)
2936   #define NEOPIXEL_STARTUP_TEST // Cycle through colors at startup
2937
2938   // Support for second Adafruit NeoPixel LED driver controlled with M150 S1 ...
2939   #define NEOPIXEL2_SEPARATE
2940   #if ENABLED(NEOPIXEL2_SEPARATE)
2941     #define NEOPIXEL2_PIXELS 15 // Number of LEDs in the second strip
2942     #define NEOPIXEL2_BRIGHTNESS 127 // Initial brightness (0-255)
2943     #define NEOPIXEL2_STARTUP_TEST // Cycle through colors at startup
2944   #else
2945     // #define NEOPIXEL2_INSERTIES // Default behavior is NeoPixel 2 in parallel
2946   #endif
2947
2948   // Use some of the NeoPixel LEDs for static (background) lighting
2949   #define NEOPIXEL_BKGD_INDEX_FIRST 0 // Index of the first background LED
2950   #define NEOPIXEL_BKGD_INDEX_LAST 5 // Index of the last background LED
2951   #define NEOPIXEL_BKGD_COLOR { 255, 255, 0 } // R, G, B, W
2952   #define NEOPIXEL_BKGD_ALWAYS_ON // Keep the backlight on when other NeoPixels are off
2953 #endif
```

#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.

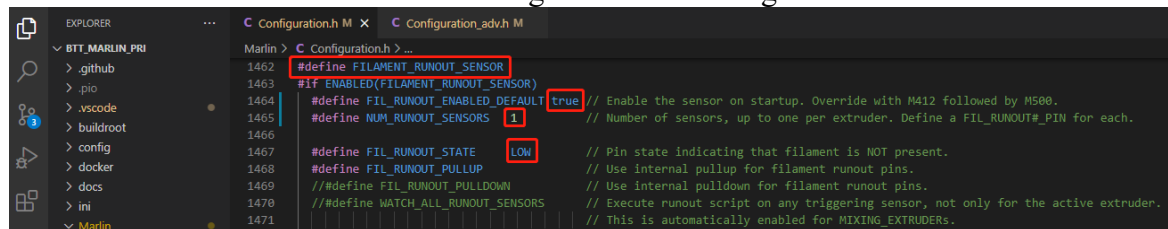


```
1326 /**  
1327  * LED Control Menu  
1328  * Add LED Control to the LCD menu  
1329  */  
1330 #define LED_CONTROL_MENU  
1331 #if ENABLED(LED_CONTROL_MENU)  
1332 #define LED_COLOR_PRESETS // Enable the Preset Color menu option  
1333 // #define NEO2_COLOR_PRESETS // Enable a second NeoPixel Preset Color menu option  
1334 #if ENABLED(LED_COLOR_PRESETS)  
1335 #define LED_USER_PRESET_RED 255 // User defined RED value  
1336 #define LED_USER_PRESET_GREEN 128 // User defined GREEN value  
1337 #define LED_USER_PRESET_BLUE 0 // User defined BLUE value  
1338 #define LED_USER_PRESET_WHITE 255 // User defined WHITE value  
1339 #define LED_USER_PRESET_BRIGHTNESS 255 // User defined intensity  
1340 // #define LED_USER_PRESET_STARTUP // Have the printer display the user preset color on startup  
1341 #endif  
1342 #if ENABLED(NEO2_COLOR_PRESETS)  
1343 #define NEO2_USER_PRESET_RED 255 // User defined RED value  
1344 #define NEO2_USER_PRESET_GREEN 128 // User defined GREEN value  
1345 #define NEO2_USER_PRESET_BLUE 0 // User defined BLUE value  
1346 #define NEO2_USER_PRESET_WHITE 255 // User defined WHITE value  
1347 #define NEO2_USER_PRESET_BRIGHTNESS 255 // User defined intensity  
1348 // #define NEO2_USER_PRESET_STARTUP // Have the printer display the user preset color on startup for the second strip  
1349 #endif  
1350 #endif
```

#define LED_CONTROL_MENU // add led control to your menu.

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.



```
1462 #define FILAMENT_RUNOUT_SENSOR  
1463 #if ENABLED(FILAMENT_RUNOUT_SENSOR)  
1464 #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M500.  
1465 #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT#_PIN for each.  
1466 #define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT present.  
1467 #define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins.  
1468 // #define FIL_RUNOUT_PULLDOWN // Use internal pulldown for filament runout pins.  
1469 // #define WATCH_ALL_RUNOUT_SENSORS // Execute runout script on any triggering sensor, not only for the active extruder.  
1470 // 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. Set according to the actual situation of the module. If the module sends a low level when the filament is abnormal, set it to LOW.

Smart Filament Sensor (SFS V1.0)

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

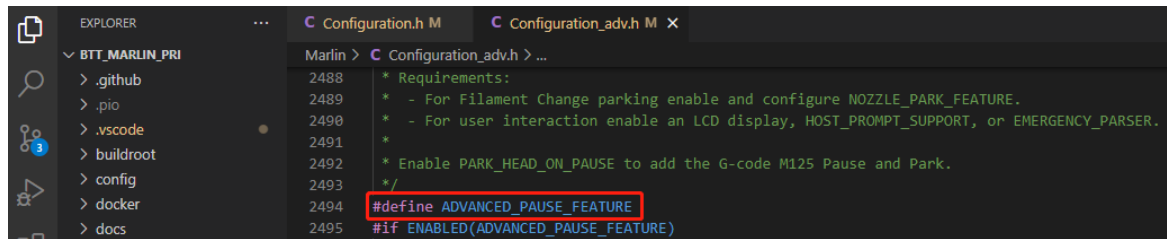
```
1462 #define FILAMENT_RUNOUT_SENSOR
1463 #if ENABLED(FILAMENT_RUNOUT_SENSOR)
1464 #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M580.
1465 #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT#_PIN for each.
1466
1467 #define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT present.
1468 #define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins.
1472
1473 // Override individually if the runout sensors vary...
1477
1478 // #define FIL_RUNOUT2_STATE LOW...
1481
1482 // #define FIL_RUNOUT3_STATE LOW...
1485
1486 // #define FIL_RUNOUT4_STATE LOW...
1489
1490 // #define FIL_RUNOUT5_STATE LOW...
1493
1494 // #define FIL_RUNOUT6_STATE LOW...
1497
1498 // #define FIL_RUNOUT7_STATE LOW...
1501
1502 // #define FIL_RUNOUT8_STATE LOW...
1505
1506 // Commands to execute on filament runout.
1507 // With multiple runout sensors use the %c placeholder for the current tool in commands (e.g., "M600 T%c")
1508 // NOTE: After 'M412 H1' the host handles filament runout and this script does not apply.
1509 #define FILAMENT_RUNOUT_SCRIPT "M600"
1510
1511 // After a runout is detected, continue printing this length of filament
1512 // before executing the runout script. Useful for a sensor at the end
1513 // of a feed tube. Requires 4 bytes SRAM per sensor, plus 4 bytes overhead.
1514 #define FILAMENT_RUNOUT_DISTANCE_MM 7
1515
1516 #ifndef FILAMENT_RUNOUT_DISTANCE_MM
1517 // Enable this option to use an encoder disc that toggles the runout pin
1518 // as the filament moves. (Be sure to set FILAMENT_RUNOUT_DISTANCE_MM
1519 // large enough to avoid false positives.)
1520 #define FILAMENT_MOTION_SENSOR
1521 #endif
1522 #endif
```

`#define FILAMENT_MOTION_SENSOR` // set encoder type
`#define FILAMENT_RUNOUT_DISTANCE_MM 7` // 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.

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

```
1907 #define NOZZLE_PARK_FEATURE
1908
1909 #if ENABLED(NOZZLE_PARK_FEATURE)
1910 // Specify a park position as { X, Y, Z raise }
1911 #define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 }
1912 // #define NOZZLE_PARK_X_ONLY // X move only is required to park
1913 // #define NOZZLE_PARK_Y_ONLY // Y move only is required to park
1914 #define NOZZLE_PARK_Z_RAISE_MIN 2 // (mm) Always raise Z by at least this distance
1915 #define NOZZLE_PARK_XY_FEEDRATE 100 // (mm/s) X and Y axes feedrate (also used for delta Z axis)
1916 #define NOZZLE_PARK_Z_FEEDRATE 5 // (mm/s) Z axis feedrate (not used for delta printers)
1917 #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

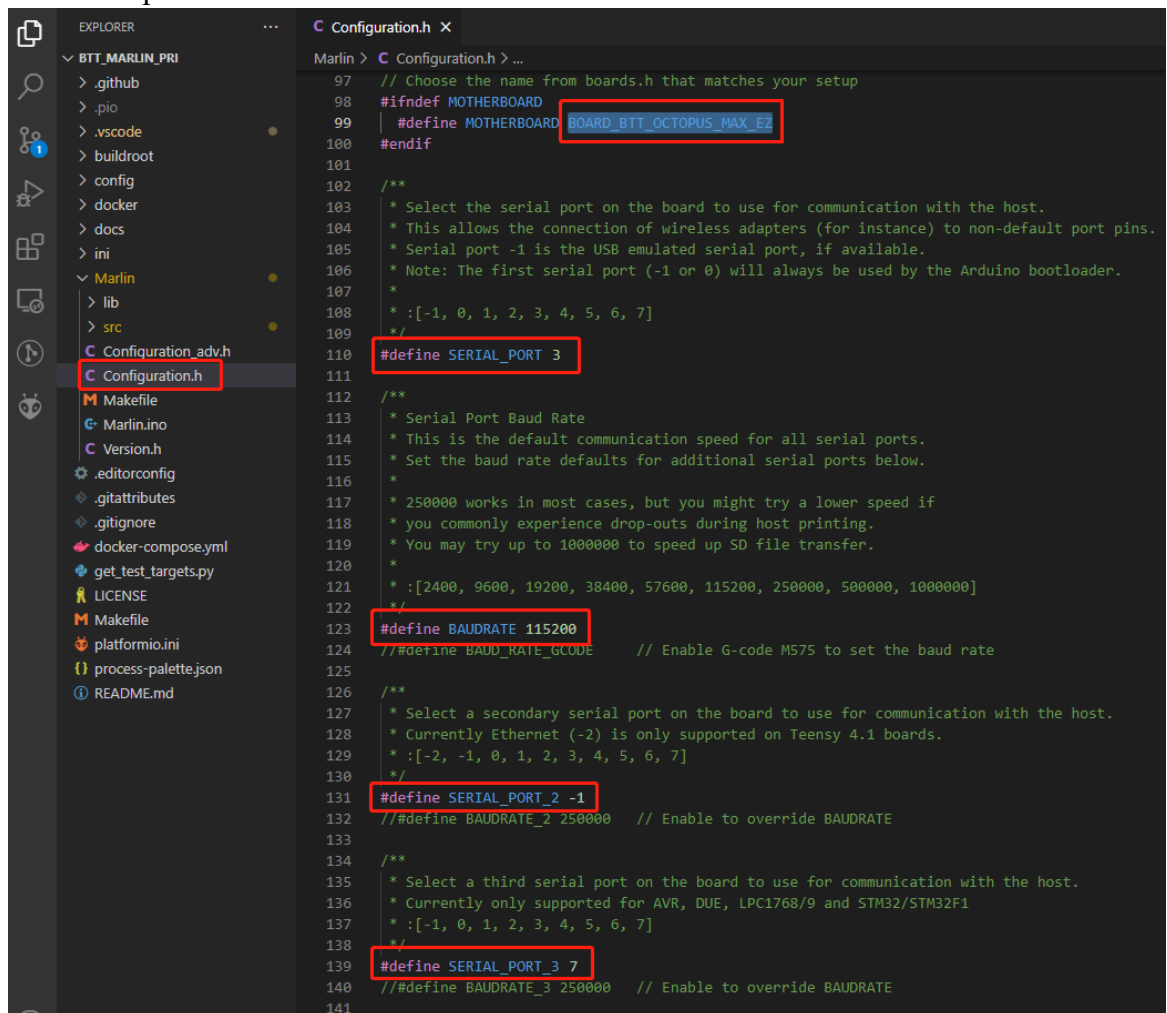


```
2488 * Requirements:
2489 * - For Filament Change parking enable and configure NOZZLE_PARK_FEATURE.
2490 * - For user interaction enable an LCD display, HOST_PROMPT_SUPPORT, or EMERGENCY_PARSER.
2491 *
2492 * Enable PARK_HEAD_ON_PAUSE to add the G-code M125 Pause and Park.
2493 */
2494 #define ADVANCED_PAUSE_FEATURE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

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

ESP3D

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

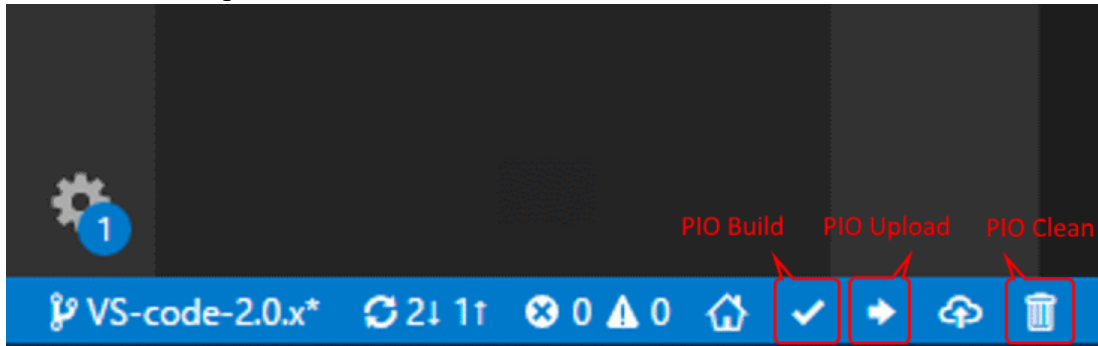


```
97 // Choose the name from boards.h that matches your setup
98 #ifndef MOTHERBOARD
99 #define MOTHERBOARD BOARD_BTT_OCTOPUS_MAX_E2
100 #endif
101
102 /**
103 * Select the serial port on the board to use for communication with the host.
104 * This allows the connection of wireless adapters (for instance) to non-default port pins.
105 * Serial port -1 is the USB emulated serial port, if available.
106 * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
107 *
108 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
109 */
110 #define SERIAL_PORT 3
111
112 /**
113 * Serial Port Baud Rate
114 * This is the default communication speed for all serial ports.
115 * Set the baud rate defaults for additional serial ports below.
116 *
117 * 250000 works in most cases, but you might try a lower speed if
118 * you commonly experience drop-outs during host printing.
119 * You may try up to 1000000 to speed up SD file transfer.
120 *
121 * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
122 */
123 #define BAUDRATE 115200
124 // #define BAUD_RATE_GCODE // Enable G-code M575 to set the baud rate
125
126 /**
127 * Select a secondary serial port on the board to use for communication with the host.
128 * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
129 * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
130 */
131 #define SERIAL_PORT_2 -1
132 // #define BAUDRATE_2 250000 // Enable to override BAUDRATE
133
134 /**
135 * Select a third serial port on the board to use for communication with the host.
136 * Currently only supported for AVR, DUE, LPC1768/9 and STM32/STM32F1
137 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
138 */
139 #define SERIAL_PORT_3 7
140 // #define BAUDRATE_3 250000 // Enable to override BAUDRATE
141
```

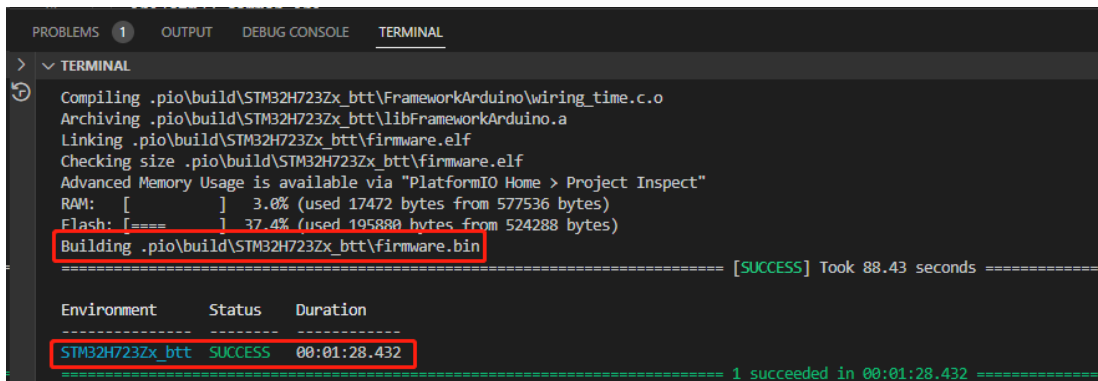
The newest ESP3D firmware can be found at <https://github.com/luc-github/ESP3D>, compile your own binary file and rename it to "esp3d.bin", copy it to the root directory of the SD card, insert into the motherboard and press the reset button. The bootloader will update the firmware to ESP8266 automatically.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

Compile Firmware

1. Click "√" to compile firmware.



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



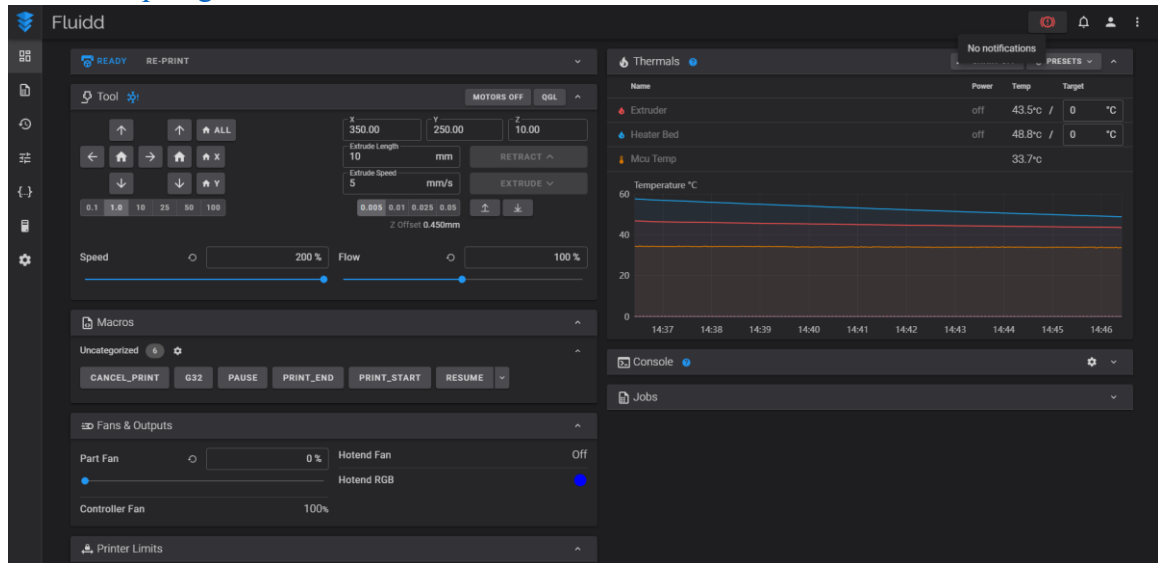
Klipper

Preparation

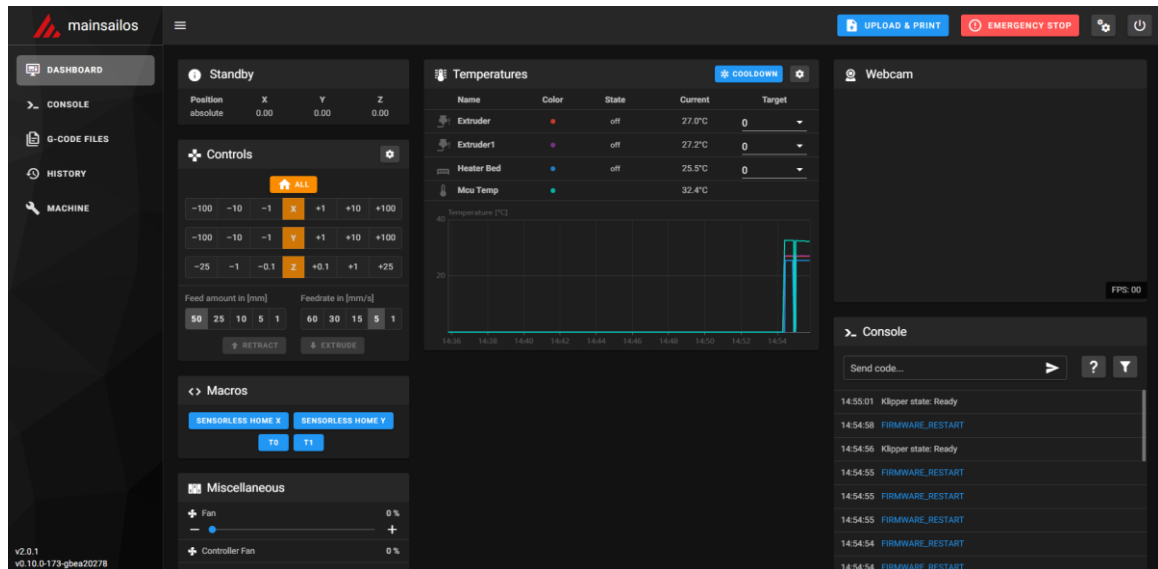
Download OS Image

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

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



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



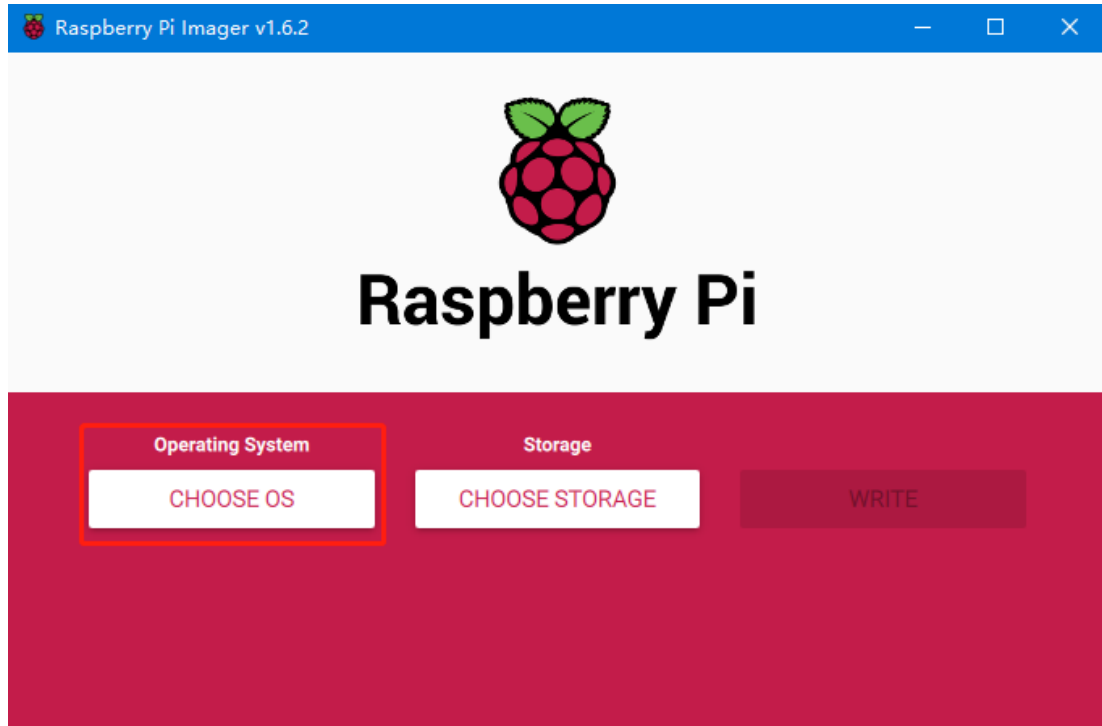
Or refer to [Klipper official installation guide](#) using Octoprint.

Download and Install Raspberry Pi Imager

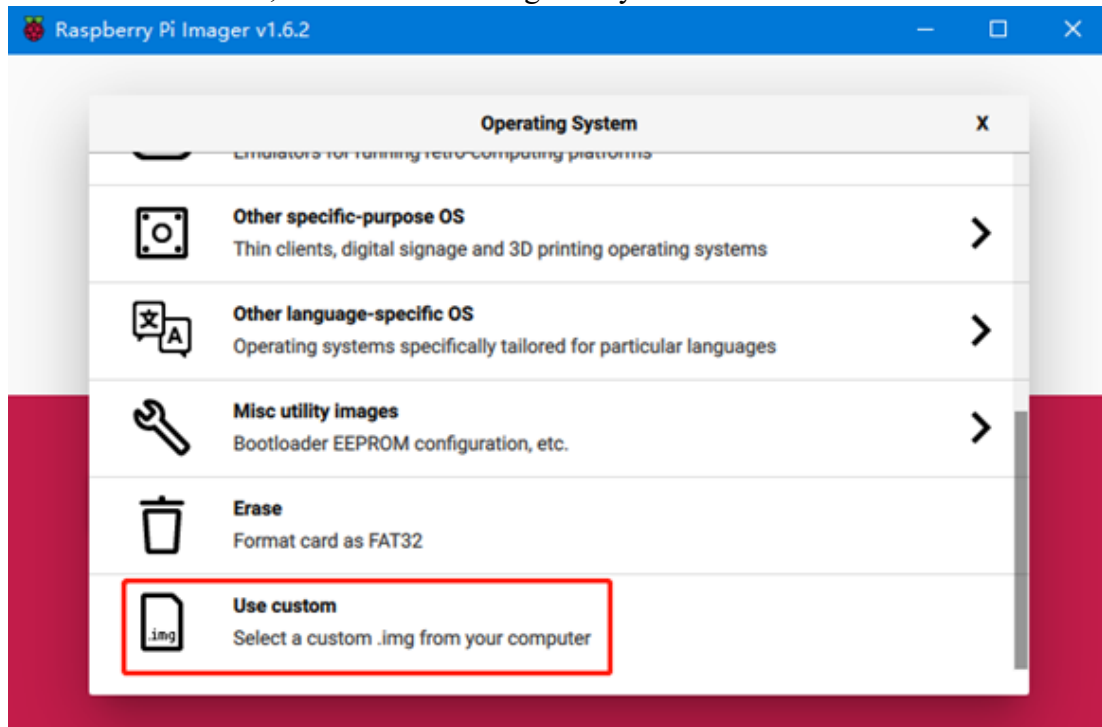
Install the official Raspberry Pi Imager <https://www.raspberrypi.com/software/>

Write Image

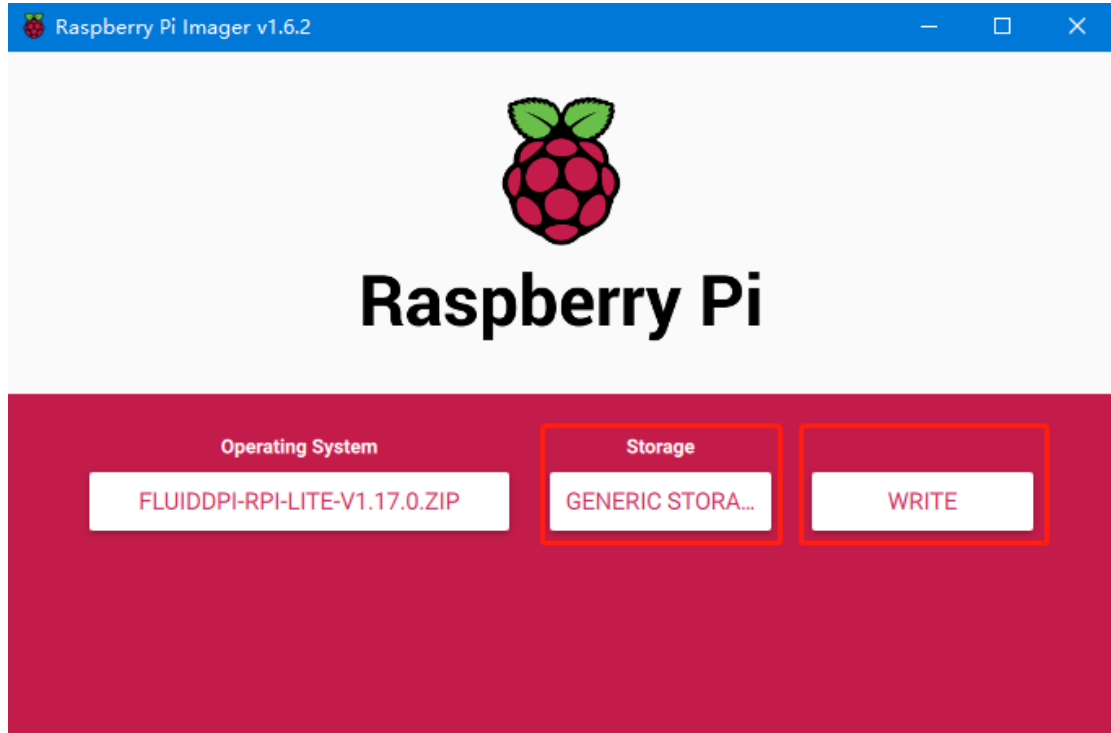
1. Insert microSD into your computer via a card reader.
2. Choose OS.



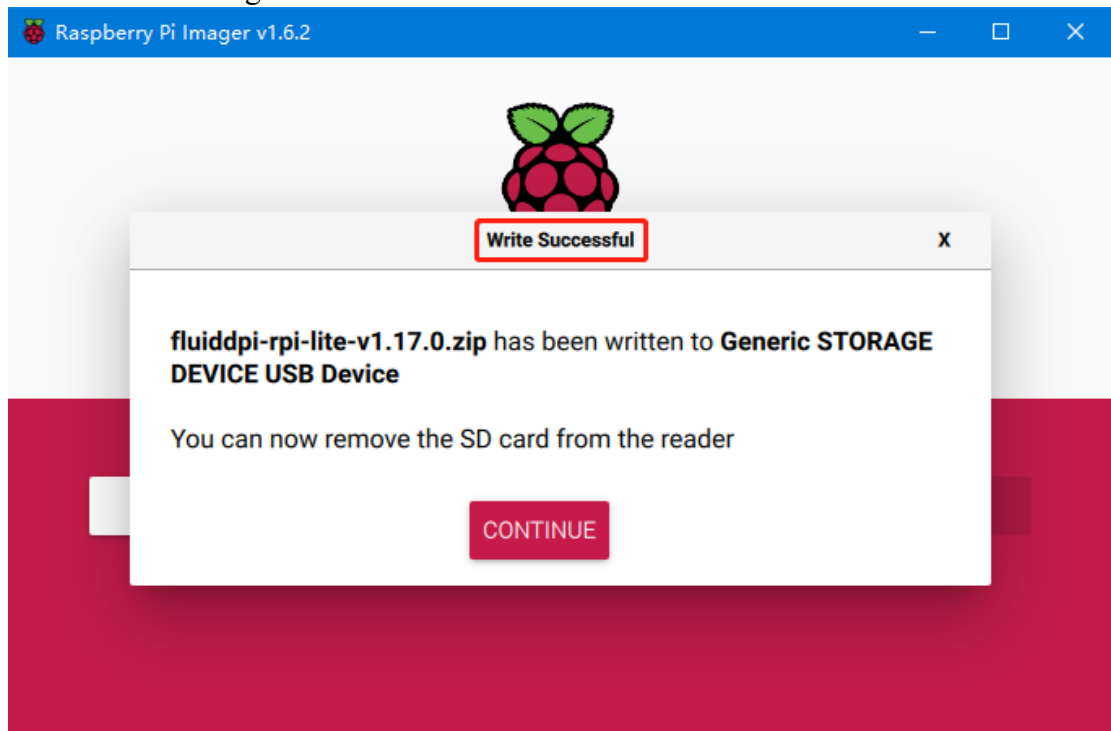
3. Select "Use custom", then select the image that you downloaded.



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



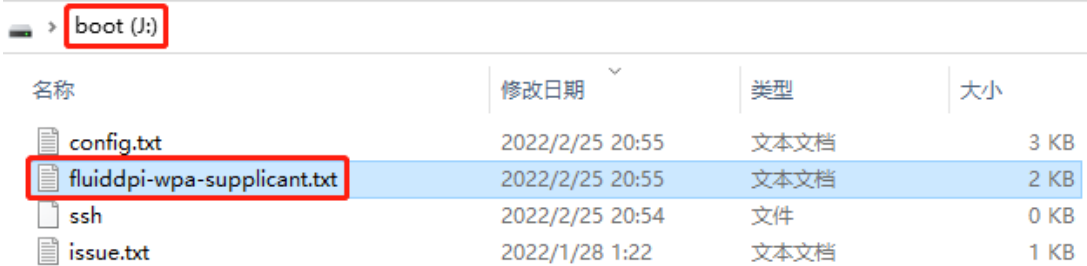
5. Wait for the writing to finish.



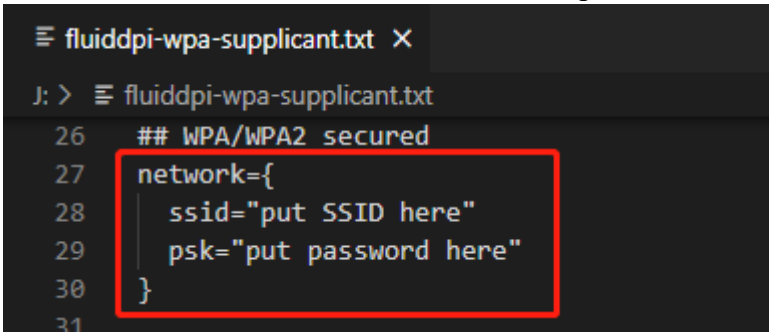
WiFi Setting

Note: this step can be skipped if you are using a network cable connection.

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)

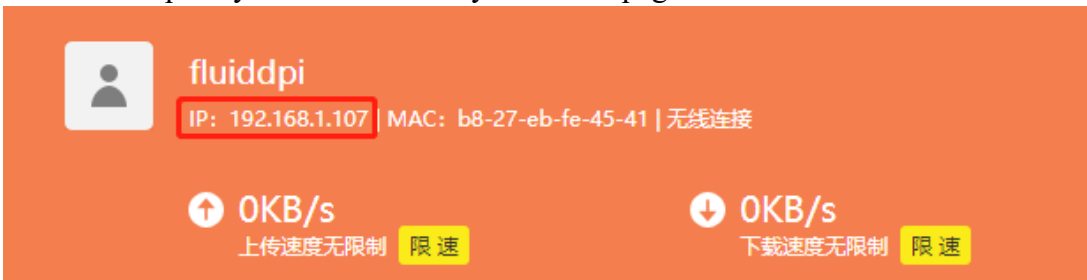


3. Delete "#", insert the correct WiFi SSID and password then save the file.

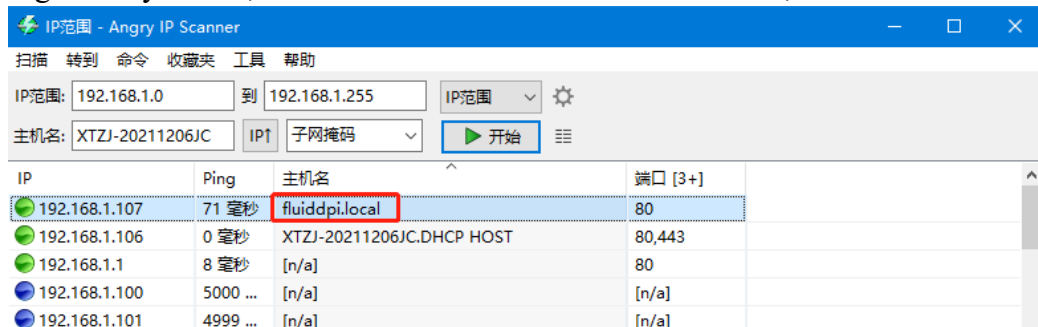


SSH Connect to Raspberry Pi

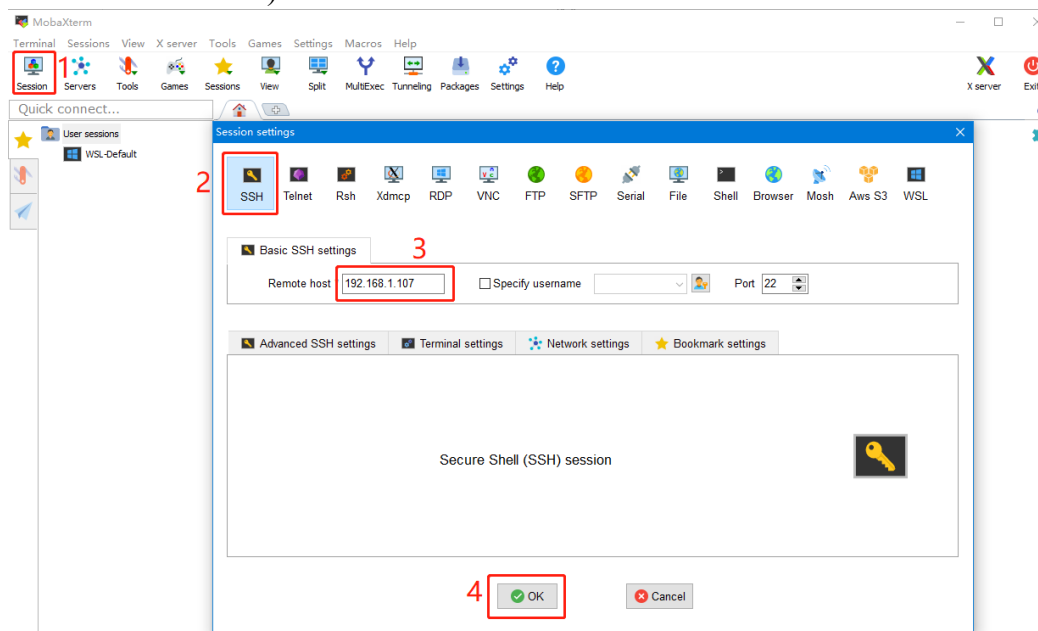
1. Install the SSH application MobaXterm: <https://mobaxterm.mobatek.net/download-home-edition.html>
2. Insert SD card to Raspberry Pi, wait for system to load after power on, approx. 1-2min.
3. The Raspberry Pi will automatically be assigned an IP address after being successfully connected to the network.
4. Find the Raspberry Pi IP address on your router page.



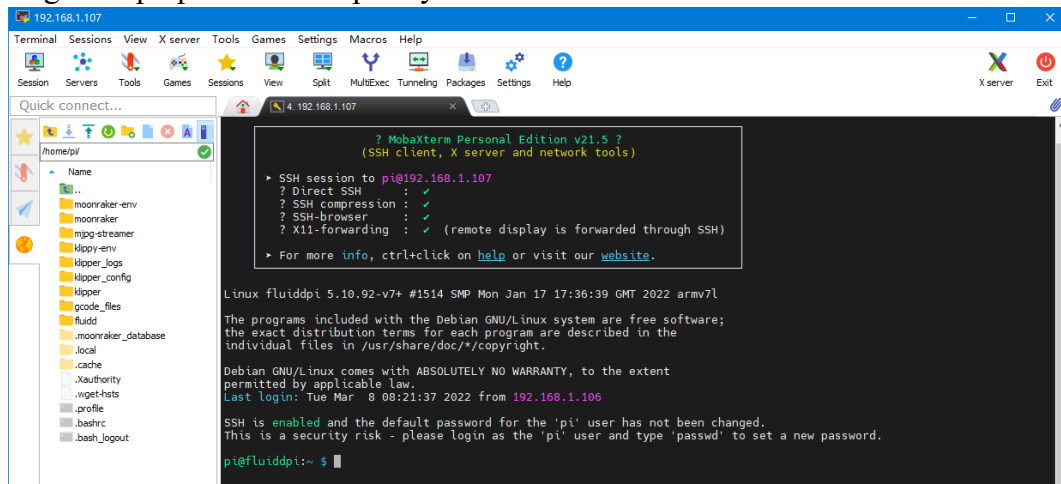
5. Or use the <https://angryip.org/> tool, scan all IP addresses in the current network organize by names, and find the IP named Fluidd or Mailsail, as shown below.



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



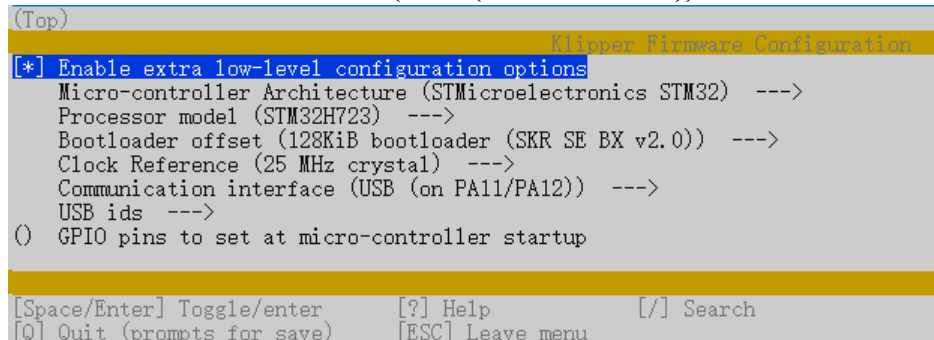
Compile Firmware

1. After SSH successfully connected to the Raspberry Pi, enter in terminal:

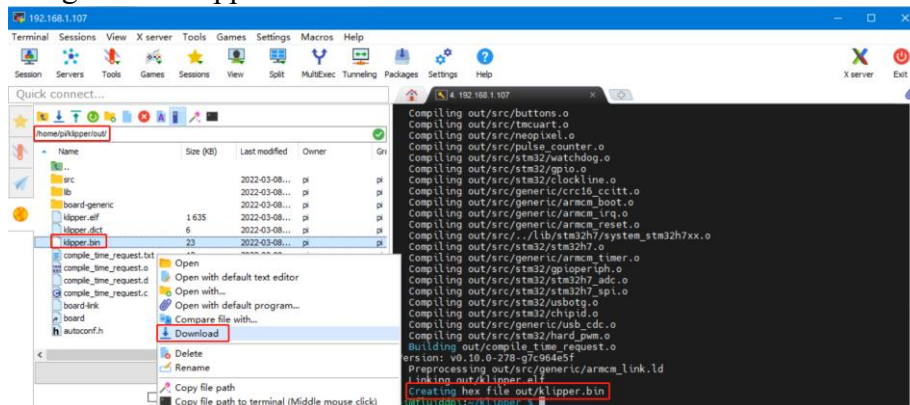
```
cd ~/klipper/  
make menuconfig
```

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

- * [*] 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)) --->



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, as shown below.

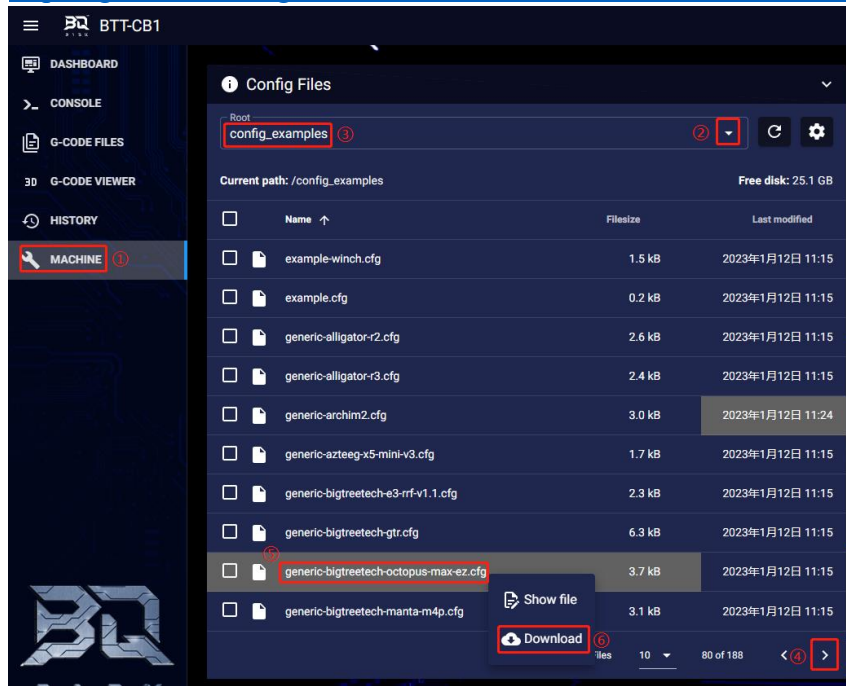
```
pi@fluiddpi:~/klipper $ ls /dev/serial/by-id/  
usb-Klipper_stm32h723xx_41003D001751303232383230-if00  
pi@fluiddpi:~/klipper $
```

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

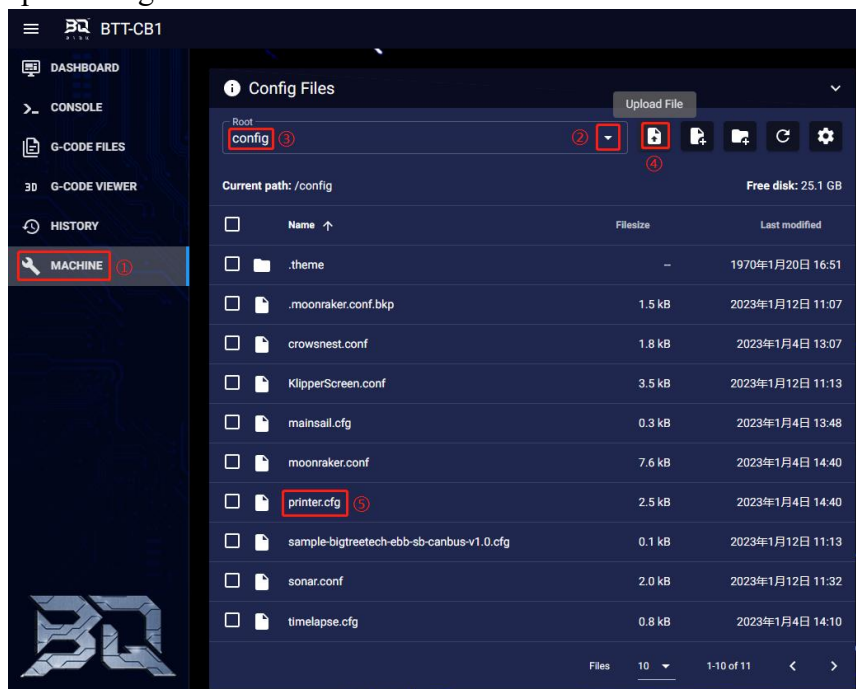
Configure Klipper

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:

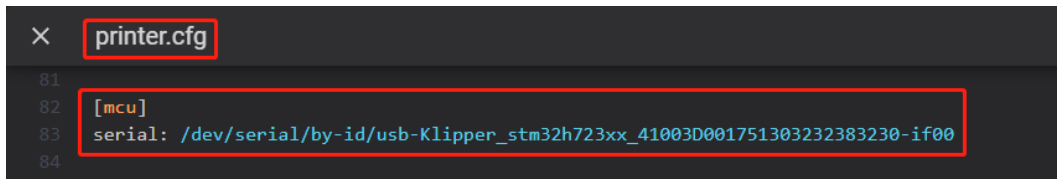
<https://github.com/bigtreetech/BIGTREETECH-OCTOPUS-Max-EZ>



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



3. Insert the correct motherboard ID.



```
× printer.cfg
81
82 [mcu]
83 serial: /dev/serial/by-id/usb-Klipper_stm32h723xx_41003D001751303232383230-if00
84
```

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

Firmware Update

Update using microSD

1. Make sure microSD 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 microSD into 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.

Cautions

1. Unplugging and plugging operations should be performed under the condition of power off.
2. Ensure that the voltage selection matches the fan's working voltage to prevent damage.

If you need other resources for this product, please visit <https://github.com/bigtreetech/> and find them yourself. If you cannot find the resources you need, you can contact our after-sales support.

If you encounter other problems during use, feel free to contact us, and we are answering them carefully; any good opinions or suggestions on our products are welcome, too and we will consider them carefully. Thank you for choosing BIGTREETECH. Your support means a lot to us!