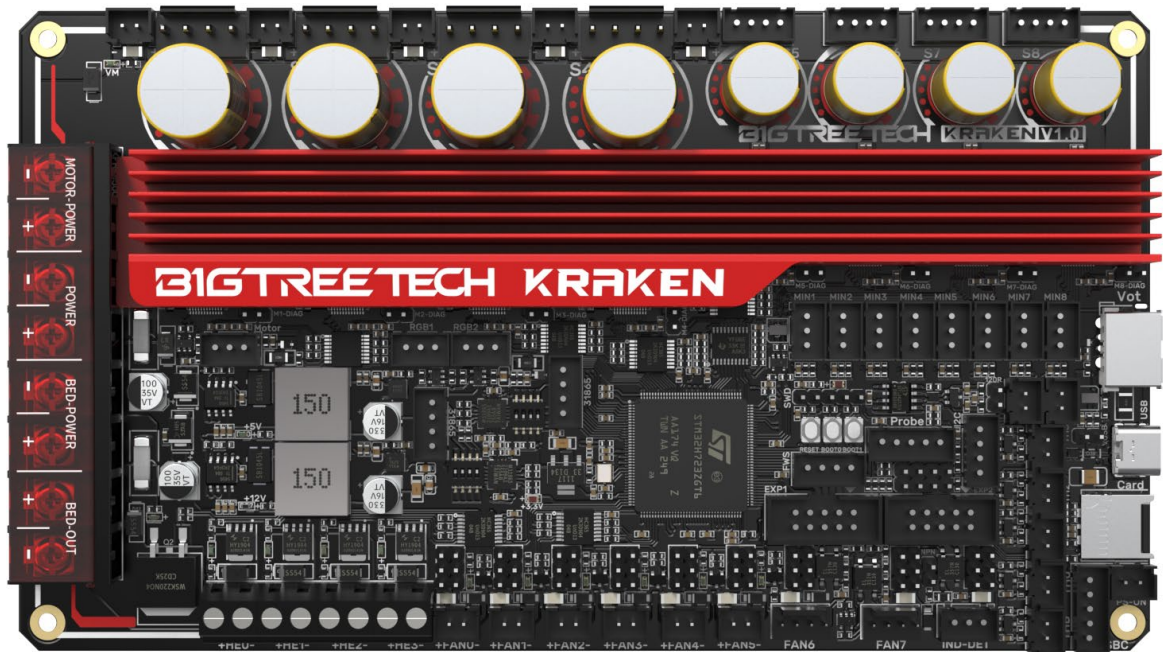


**BIGTREE TECH**

# KRAKEN V1.0

## User Manual



## Revision Log

Version	Date	Revisions
v1.00	22nd December 2023	Initial Version

---

## CONTENT

<b>Revision Log</b> .....	2
<b>Product Profile</b> .....	5
<b>Features Highlights</b> .....	5
<b>Specifications</b> .....	6
<b>Firmware Support</b> .....	7
<b>Dimensions</b> .....	8
<b>Peripheral Interface</b> .....	8
<b>Interface Diagram</b> .....	8
<b>Pin Description</b> .....	9
<b>Interface Introduction</b> .....	9
<b>USB Power Supply</b> .....	9
<b>Stepper Motor Drivers</b> .....	10
<b>Onboard TMC2160 in SPI mode</b> .....	10
<b>PWM Fan Voltage Selection</b> .....	10
<b>MicroProbe V2.0 Wiring</b> .....	11
<b>Auto Power Off (Relay V1.2) Wiring</b> .....	11
<b>EXP1+EXP2 and MINI12864 V2.0 Display Wiring</b> .....	12
<b>RGB Wiring</b> .....	12
<b>Servo Wiring</b> .....	13
<b>I<sup>2</sup>C Wiring (Temperature and Humidity Sensor)</b> .....	13
<b>2-pin Fan Connection</b> .....	14
<b>Proximity Switch Connection</b> .....	15
<b>4-pin PWM Fan and Water Cooling Connection (12V example)</b> .....	16
<b>Raspberry Pi Connection</b> .....	16
<b>Marlin</b> .....	17
<b>Install Compiling Environment</b> .....	17
<b>Download Marlin Firmware</b> .....	17

<b>Configure Firmware</b> .....	17
<b>Open Marlin Project</b> .....	17
<b>Compiling Environment</b> .....	17
<b>Configure Motherboard and Serial Port</b> .....	17
<b>Configure Stepper Driver</b> .....	18
<b>Sensorless Homing</b> .....	20
<b>100K NTC or PT1000</b> .....	21
<b>BLTouch</b> .....	21
<b>Auto Power Off (Relay V1.2)</b> .....	24
<b>RGB</b> .....	24
<b>Filament Sensor</b> .....	25
<b>Smart Filament Sensor (SFS V1.0)</b> .....	26
<b>Compile Firmware</b> .....	27
<b>Klipper</b> .....	28
<b>Compiling the Firmware</b> .....	28
<b>Configuring Klipper</b> .....	29
<b>Firmware Updates</b> .....	31
<b>Updating via microSD</b> .....	31
<b>Updating Klipper via DFU</b> .....	31
<b>Precautions</b> .....	32

## Product Profile

BIGTREETECH Kraken V1.0 is a 32-bit motherboard for large printers. It comes with onboard high-voltage, high-current stepper motor drivers, significantly simplifying the connection between the motherboard and high-voltage drivers and saving space in the chassis. The board uses silkscreened ID design with an ID-design heat sink for aesthetics and practical cooling.

## Features Highlights

- Utilizes a 32-bit ARM Cortex-M7 series STM32H723ZGT6 MCU with a main frequency of 550 MHz.
- TPS5450-5A power chip, supporting DC12/24V power input. This chip provides an output current of up to 5A, peaking at 6A, perfectly supporting Raspberry Pi power supply.
- The motherboard has a reserved BOOT button, allowing users to update the bootloader via DFU mode.
- The thermistor circuit is protected to prevent MCU damage from shorted heated bed and heater cartridge connections;
- Selectable voltage (24V, 12V, 5V) for PWM fan, eliminating the need for external voltage conversion modules, thereby reducing the likelihood of motherboard damage.
- Onboard two MAX31865 modules, supporting dual PT thermocouples and compatible with two or four-wire PT100/PT1000, facilitating DIY usage for customers.
- Firmware can be upgraded via MicroSD card or through the Klipper's `make flash` command via DFU.
- Onboard 8 TMC2160 drivers in SPI mode with DIAG function pins; simply plug and unplug jumpers for easy use.
- Reserved interfaces for Filament Detection, Auto Power-Off, Probe, RGB, I<sup>2</sup>C, Servo, EXP1+EXP2, CAN, UART (SBC), and USB A power output.
- High-performance MOSFETs to reduce heat generation.
- Replaceable fuses for easy replacement.
- 2x 4-pin fan interfaces with selectable voltages of 24V, 12V, and 5V, also suitable for water cooling setups.
- Onboard proximity switch port, supports NPN and PNP types, 24V, 12V, 5V voltage selectable;

- Reserved SPI interface for connecting an accelerometer for Klipper resonance compensation.
- External DC12V powers the MOSFET control power for TMC2160, reducing driver output impedance and chip heat generation.

## Specifications

MCU	ARM Cortex-M7 STM32H723ZGT6 550MHz
Driver Input Voltage	HV (24-60V)
Motherboard Input Voltage	DCIN = DC12V or DC24V
Heated Bed Input Voltage	BED IN = DC12V or DC24V
Logic Voltage	DC3.3V
Heating Interface	Heating Interface: Heated Bed (HB), Heater Cartridge (HE0, HE1, HE2, HE3)
Max Heated Bed Output Current	10A, peak 11A
Max Heater Cartridge Output Current	6A, peak 6.5A
Fan Interfaces	6x 2-pin PWM Fans (FAN0, FAN1, FAN2, FAN3, FAN4, FAN5), 2x 4-pin PWM Fans (FAN6, FAN7), 5x Always-On Fan, PWM Fans Voltage (5V, 12V, 24V) Selectable
Max Fan Output Current	1A, peak 1.5A
Total Current for Heater Cartridge + Fans	Less than 14A
Motherboard Max 5V Output Current	5A (peak 6A)
Motherboard Max 12V Output Current	5A (peak 6A)
Expansion Interfaces	Probe (Servos, Probe), Servo, Filament Sensor, PS-ON, I <sup>2</sup> C, RGBx2, SPI, SBC (UART), EXP1+EXP2,

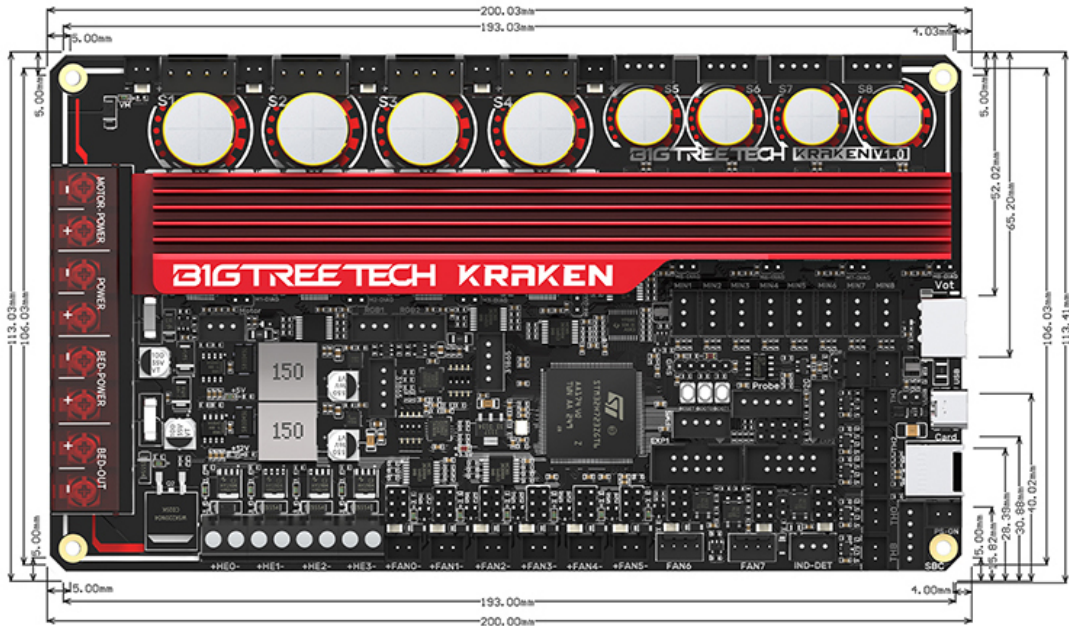
---

	CANx2, PT100/PT1000x2, USB-A 5V Power, Endstop Portx8, etc.
Motor Drivers	Onboard TMC2160, support 24-60V, Max 8A driving current for S1-S4(Rsense=22mR), Max 3A for S5-S8(Rsense=75mR)
Driver Modes	SPI
Motor Interfaces	S1, S2, S3, S4, S5, S6, S7, S8
Temp Sensor Interfaces	5x 100K NTC, 2x MAX31865
Display	LCD
PC Communication	Type-C
Supported Kinematics	Cartesian, Delta, Kossel, Ultimaker, CoreXY
Recommended Slicer/Console	Cura, Simplify3D, Pronterface, Repetier-host, Makerware
Dimensions	200 x 113mm
Mounting Dimensions	for details please refer to <b>BIGTREETECH Kraken V1.0-SIZE.png</b>

## Firmware Support

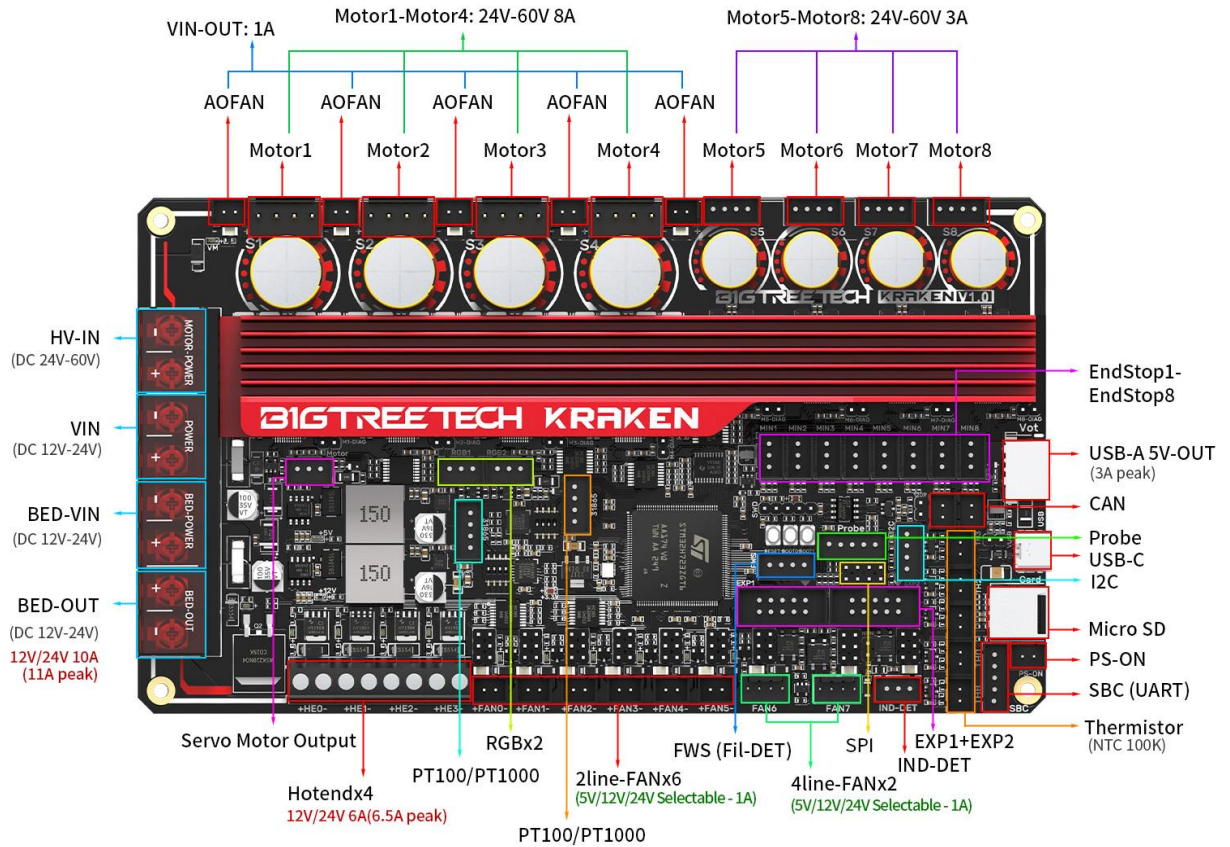
Compatible with Klipper, Marlin, RRF(RepRapFirmware)

## Dimensions



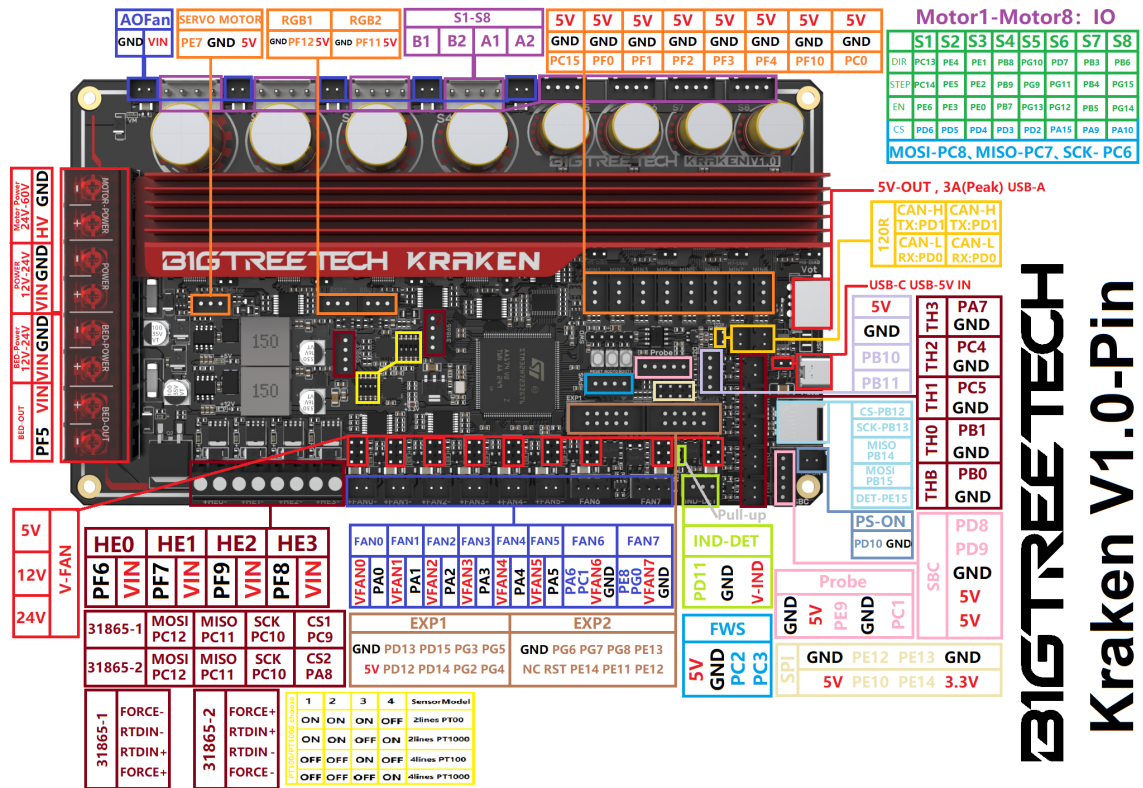
## Peripheral Interface

### Interface Diagram





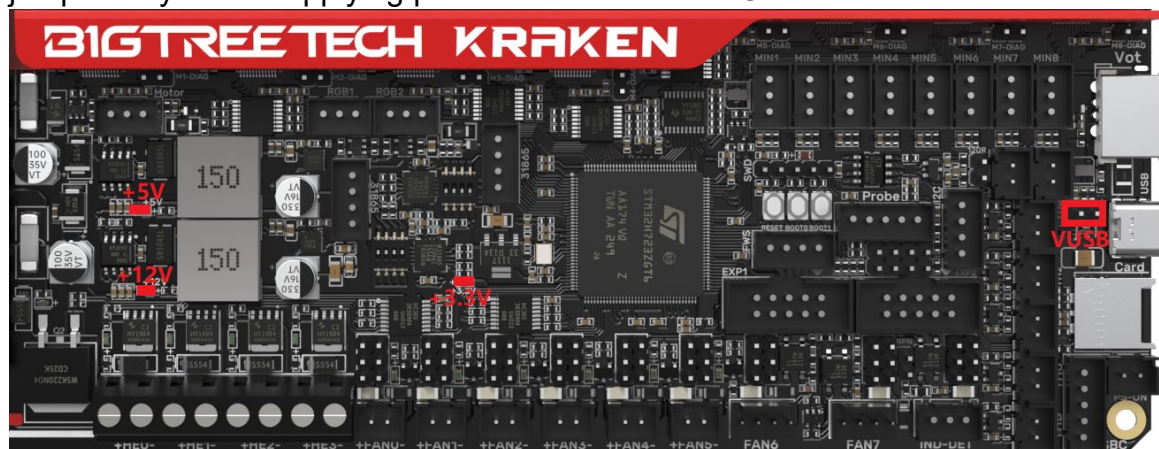
## Pin Description



## Interface Introduction

### USB Power Supply

When Kraken is powered on, the power indicator light turns red, indicating normal power supply. VUSB is the power select pin which needs to be shorted with a jumper only when supplying power to the board via USB.



## Stepper Motor Drivers

### Onboard TMC2160 in SPI mode

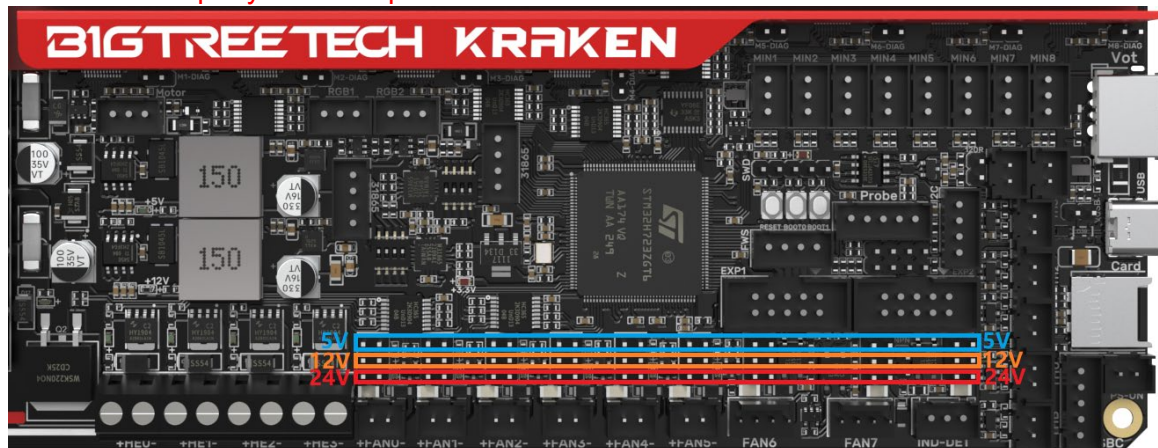
No jumper is needed to select the mode; directly use the SPI mode. When using Sensorless Homing, plug in the jumper; if not, leave it unplugged. DIAG connection as follows:



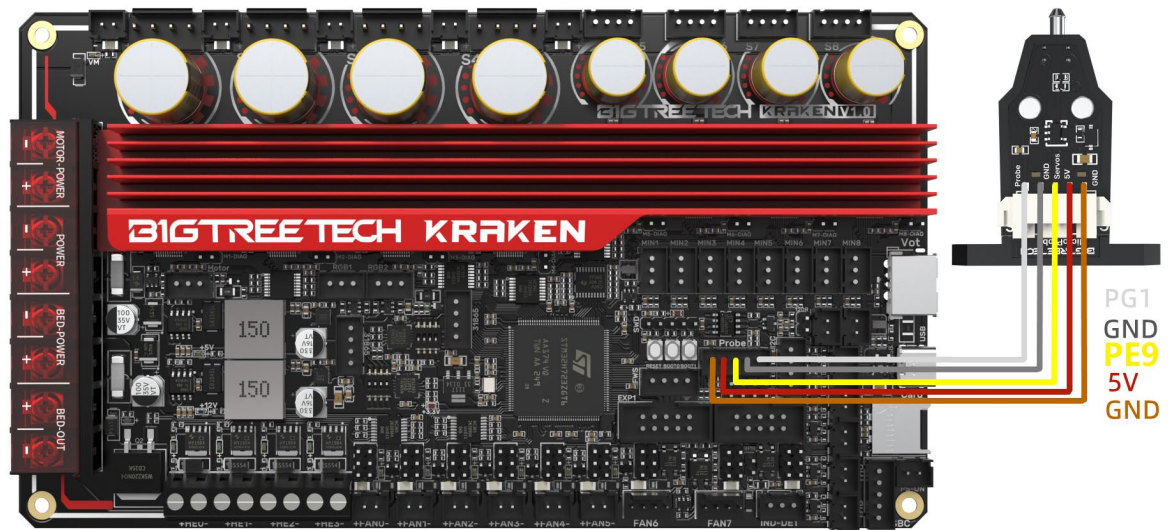
### PWM Fan Voltage Selection

Set the output voltage to 5V, 12V, or 24V via a jumper. The fan interface output current is 1A.

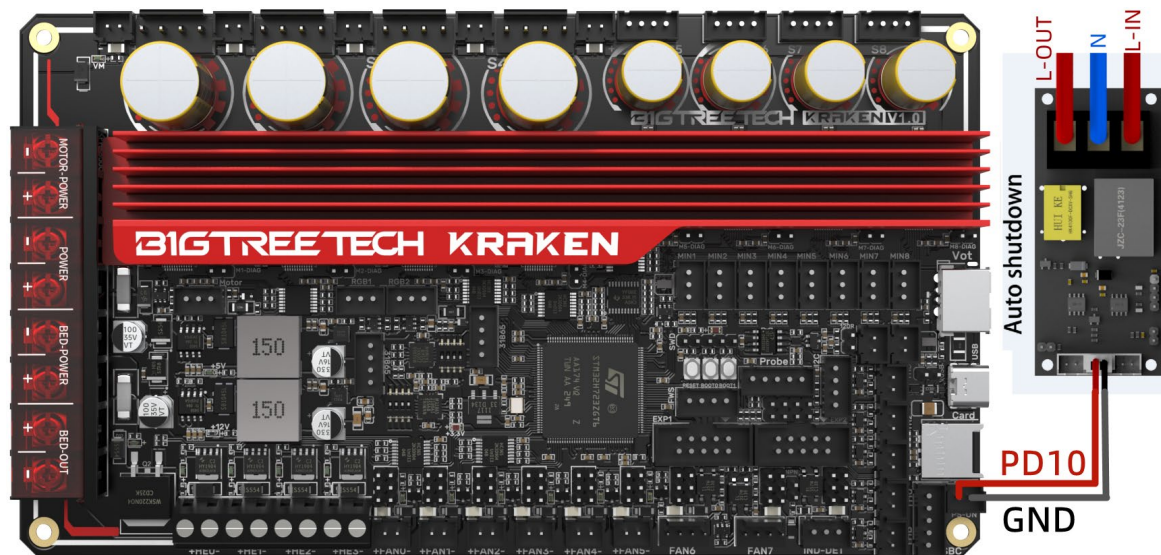
Be sure to confirm the supported voltage of the fan before selection to avoid damage for which our company is not responsible.



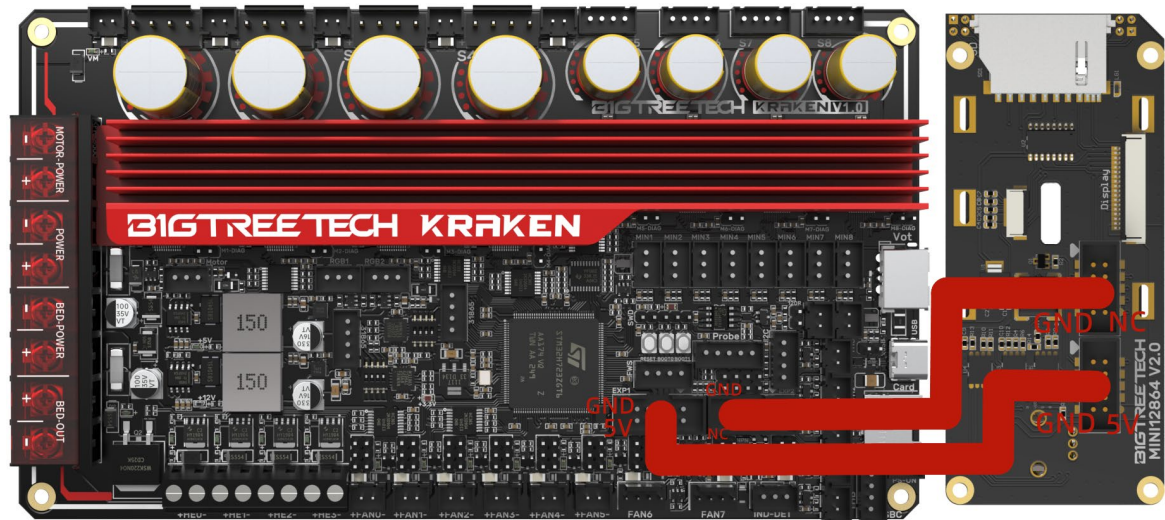
## MicroProbe V2.0 Wiring



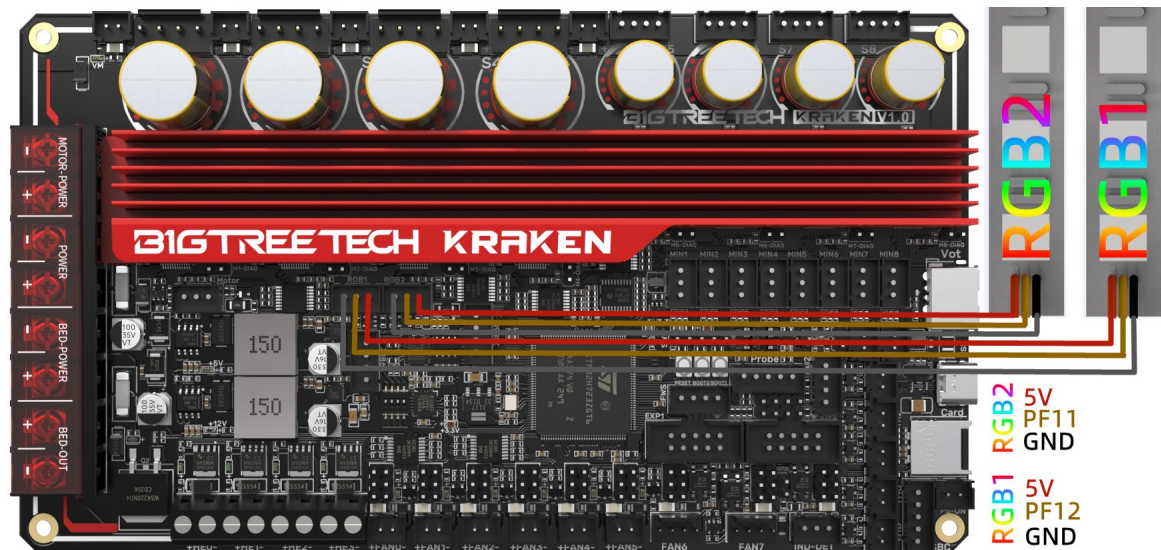
## Auto Power Off (Relay V1.2) Wiring



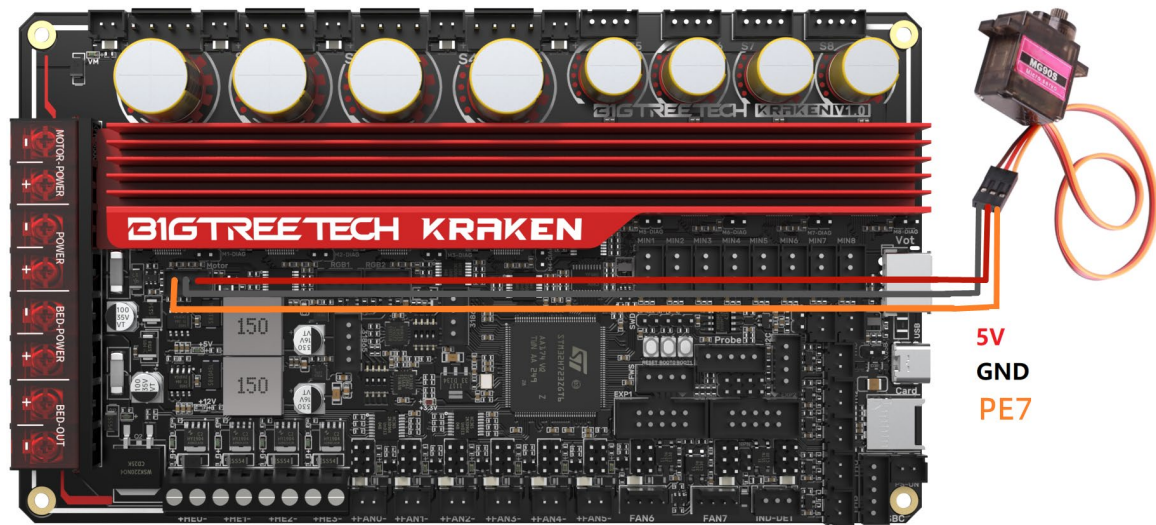
## EXP1+EXP2 and MINI12864 V2.0 Display Wiring



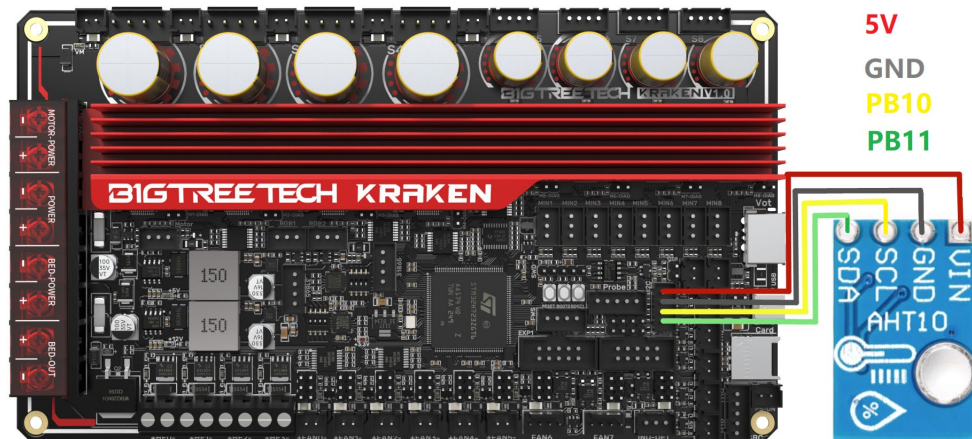
## RGB Wiring



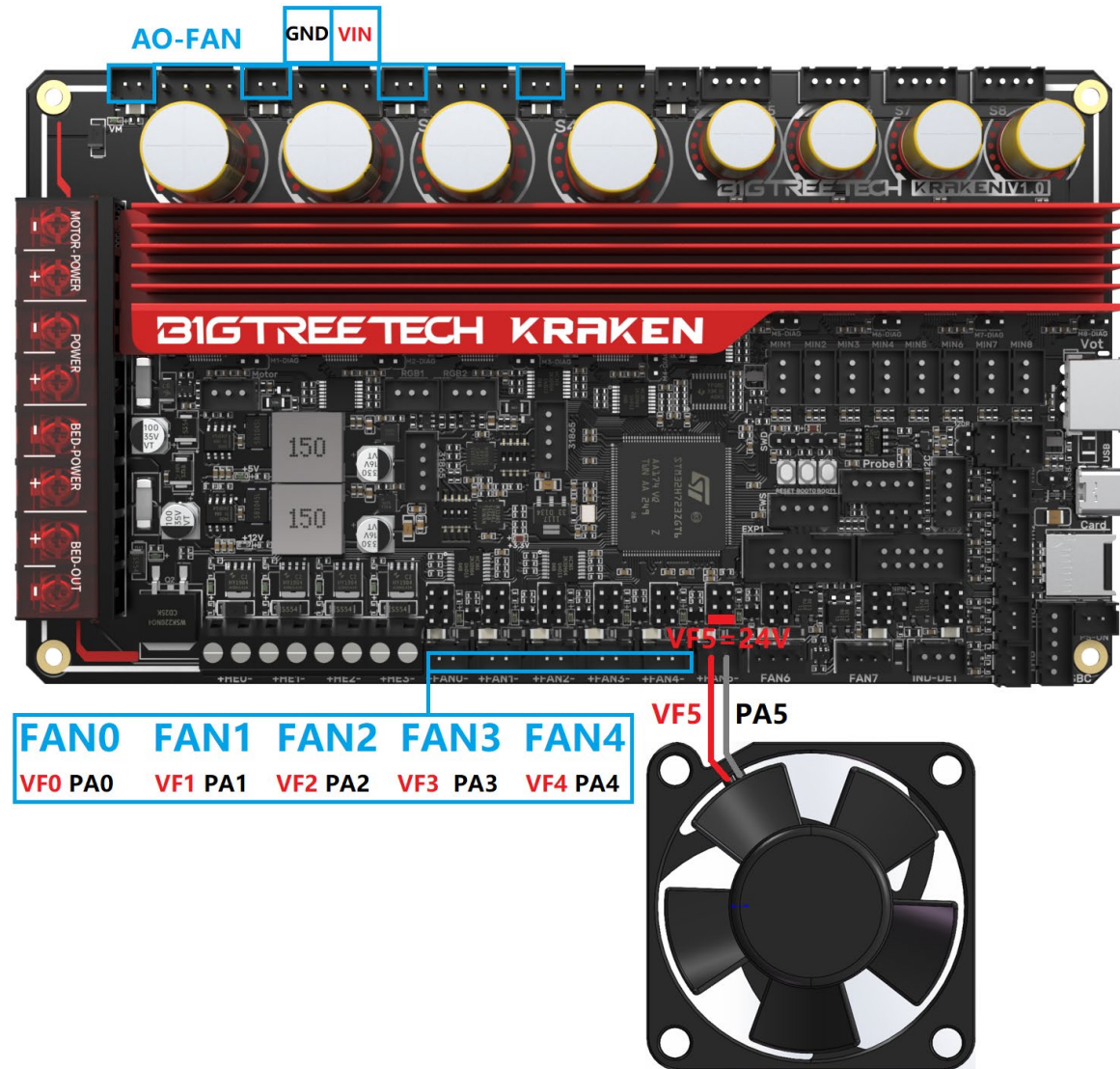
## Servo Wiring



## I<sup>2</sup>C Wiring (Temperature and Humidity Sensor)

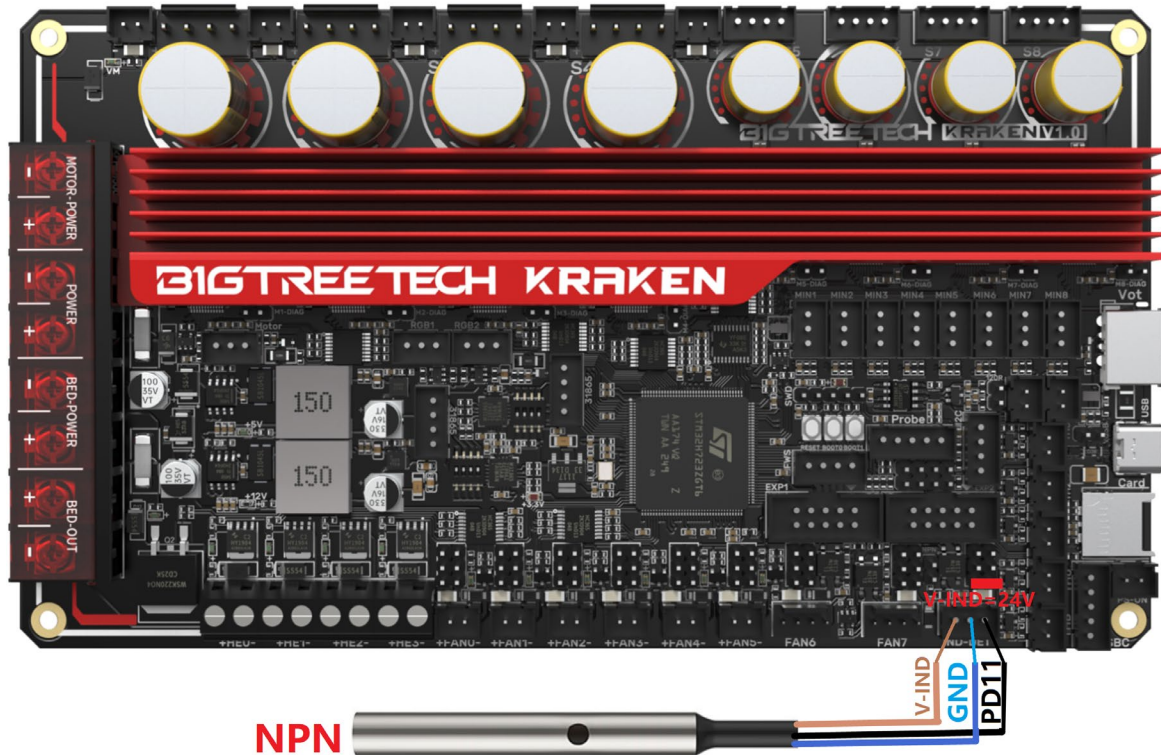


## 2-pin Fan Connection

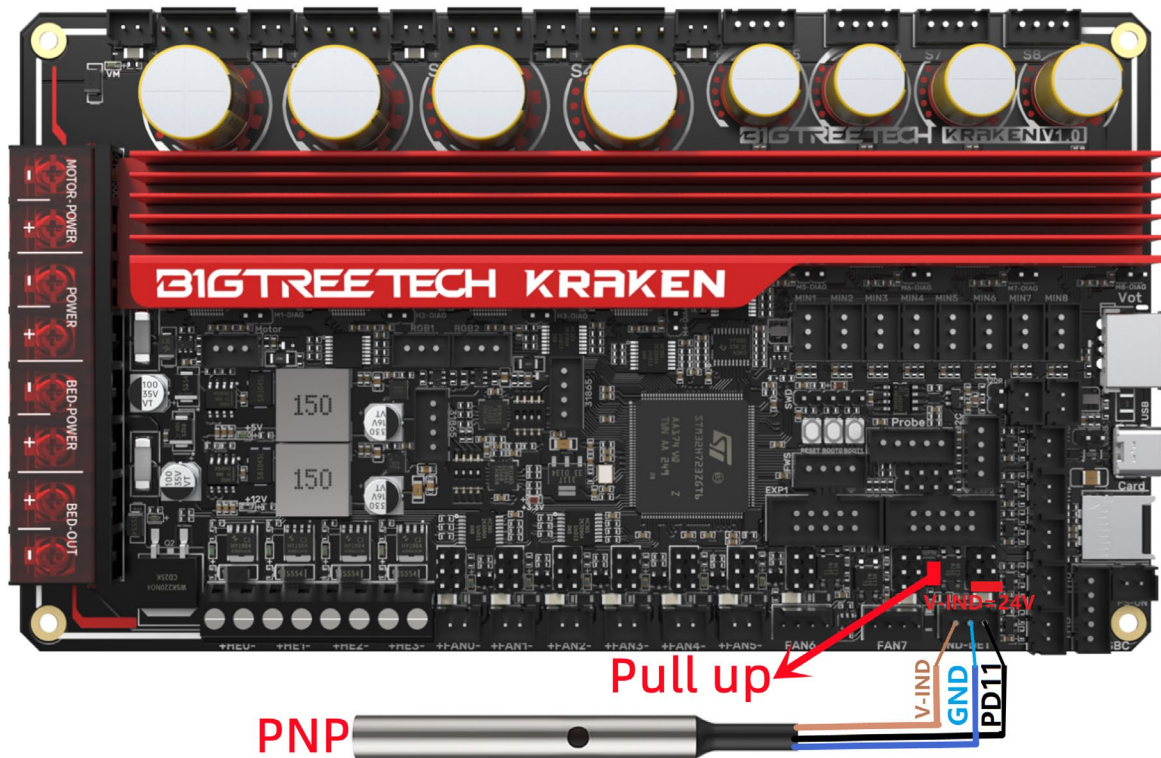


## Proximity Switch Connection

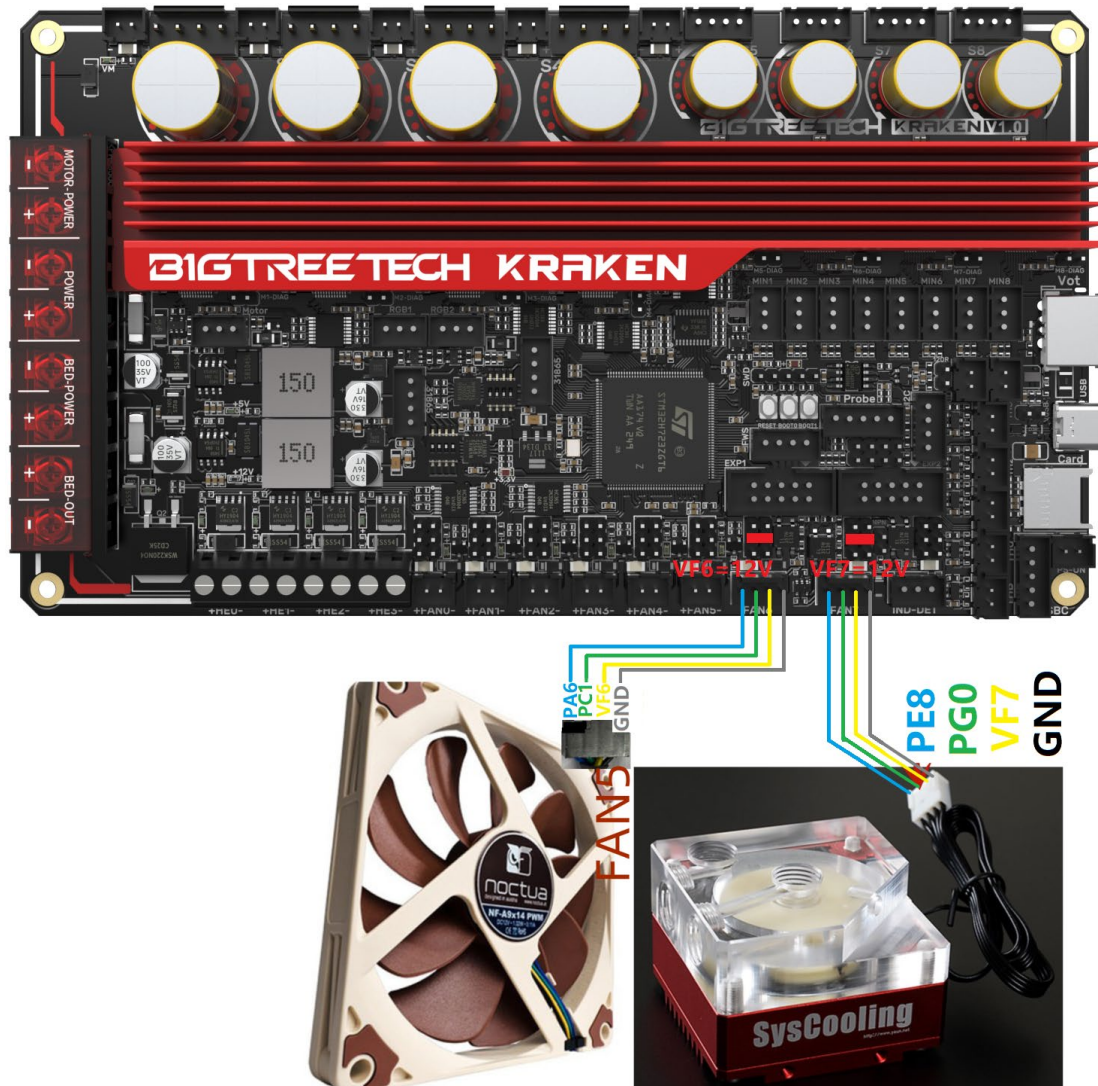
Normally open (NPN type), no jumper is required, as shown in the 24V example.



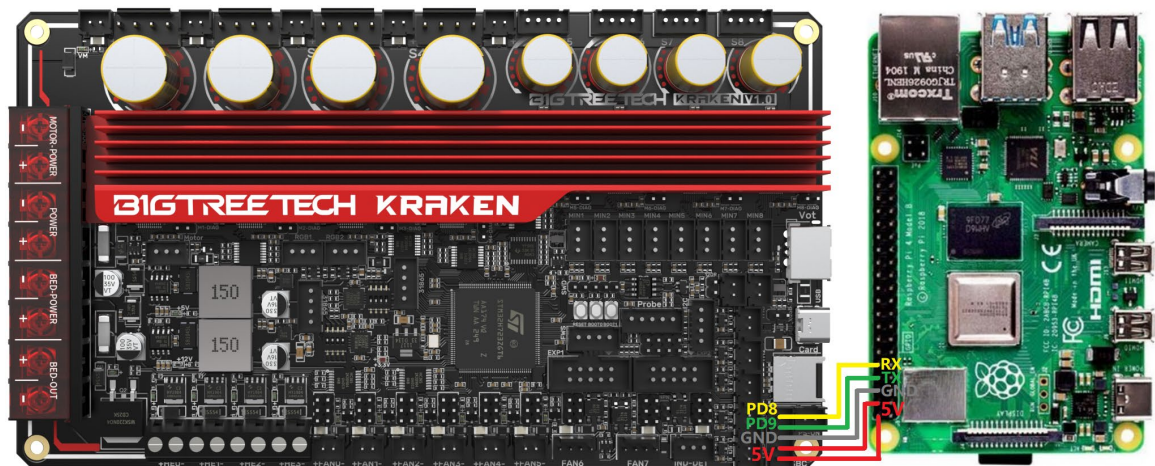
Normally closed (PNP type), a jumper is needed, as shown in the 24V example.



### 4-pin PWM Fan and Water Cooling Connection (12V example)



### Raspberry Pi Connection





## Marlin

### 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](https://marlinfw.org/docs/basics/install_platformio_vscode.html)

### Download Marlin Firmware

Get pre-configured firmware source code from our GitHub:

<https://github.com/bigtreetech/BIGTREETECH-Kraken>

### Configure Firmware

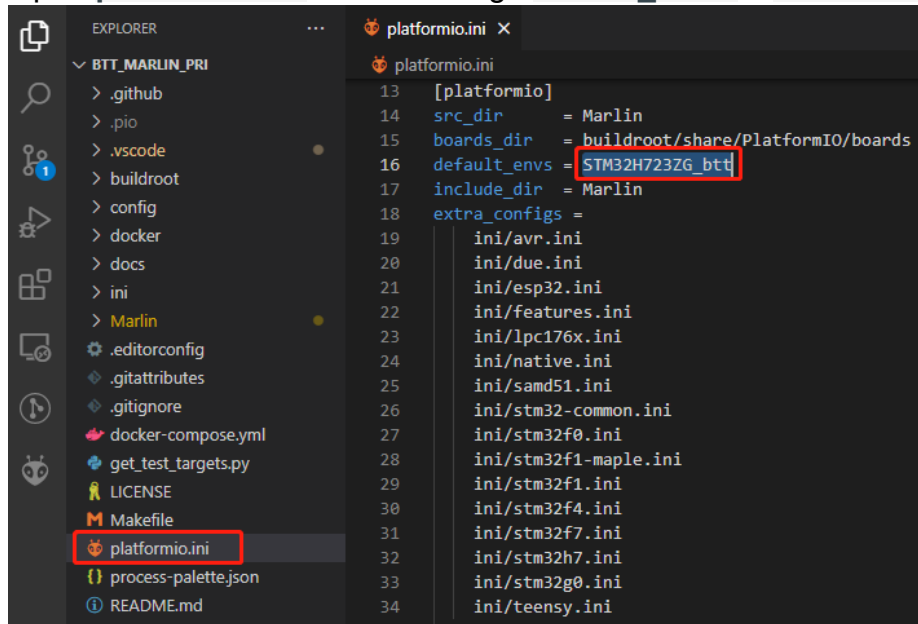
#### Open Marlin Project

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 **STM32H723ZG\_bt**.



#### Configure Motherboard and Serial Port

Set **MOTHERBOARD** as **BOARD\_BTT\_KRAKEN\_V1\_0**

`#define MOTHERBOARD BOARD_BTT_KRAKEN_V1_0`

`#define SERIAL_PORT 3` (Enable SBC serial port)  
`#define BAUDRATE 115200` (Set baudrate to the same as the communication device)

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

The above settings can be enabled as needed.

```

69 // Choose the name from boards.h that matches your setup
70 #ifndef MOTHERBOARD
71 #define MOTHERBOARD BOARD_BTT_KRAKEN_V1_0
72 #endif
73
74 /**
75  * Select the serial port on the board to use for communication with the host.
76  * This allows the connection of wireless adapters (for instance) to non-default port pins.
77  * Serial port -1 is the USB emulated serial port, if available.
78  * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
79  *
80  * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
81  */
82 #define SERIAL_PORT 3
83
84 /**
85  * Serial Port Baud Rate
86  * This is the default communication speed for all serial ports.
87  * Set the baud rate defaults for additional serial ports below.
88  *
89  * 250000 works in most cases, but you might try a lower speed if
90  * you commonly experience drop-outs during host printing.
91  * You may try up to 1000000 to speed up SD file transfer.
92  *
93  * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
94  */
95 #define BAUDRATE 115200
96
97 // #define BAUD_RATE_GCODE // Enable G-code M575 to set the baud rate
98
99 /**
100  * Select a secondary serial port on the board to use for communication with the host.
101  * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
102  * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
103  */
104 #define SERIAL_PORT_2 -1
105 // #define BAUDRATE_2 250000 // :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
106

```

## Configure Stepper Driver

Kraken has 8 TMC2160 drivers onboard

```

142 #define X_DRIVER_TYPE TMC2160
143 #define Y_DRIVER_TYPE TMC2160
144 #define Z_DRIVER_TYPE TMC2160
145 // #define X2_DRIVER_TYPE A4988
146 // #define Y2_DRIVER_TYPE A4988
147 // #define Z2_DRIVER_TYPE A4988
148 // #define Z3_DRIVER_TYPE A4988
149 // #define Z4_DRIVER_TYPE A4988
150 // #define I_DRIVER_TYPE A4988
151 // #define J_DRIVER_TYPE A4988
152 // #define K_DRIVER_TYPE A4988
153 // #define U_DRIVER_TYPE A4988
154 // #define V_DRIVER_TYPE A4988
155 // #define W_DRIVER_TYPE A4988
156 #define E0_DRIVER_TYPE TMC2160
157 // #define E1_DRIVER_TYPE A4988
158 // #define E2_DRIVER_TYPE A4988
159 // #define E3_DRIVER_TYPE A4988
160 // #define E4_DRIVER_TYPE A4988
161 // #define E5_DRIVER_TYPE A4988
162 // #define E6_DRIVER_TYPE A4988
163 // #define E7_DRIVER_TYPE A4988

```

We need to enable `TMC_USE_SW_SPI` in `Configuration_adv.h`

`#define TMC_USE_SW_SPI`

```

2980
2981
2982 /**
2983  * Software option for SPI driven drivers (TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160).
2984  * The default SW SPI pins are defined the respective pins files,
2985  * but you can override or define them here.
2986  */
2987 #define TMC_USE_SW_SPI
2988 // #define TMC_SW_MOSI -1
2989 // #define TMC_SW_MISO -1
2990 // #define TMC_SW_SCK -1
    
```

The Rsense of S1-S4 is 22mΩ, so the firmware needs to be set to 0.022. The Rsense of S5-S8 is 75mΩ, so the firmware needs to be set to 0.075.

```

2950 #if AXIS_IS_TMC_CONFIG(X)
2951 #define X_CURRENT 800 // (mA) RMS current. Multiply by 1.414 for peak current.
2952 #define X_CURRENT_HOME X_CURRENT // (mA) RMS current for sensorless homing
2953 #define X_MICROSTEPS 16 // 0..256
2954 #define X_RSENSE 0.022 // Multiplied x1000 for TMC26X
2955 #define X_CHAIN_POS -1 // -1..0: Not chained. 1: MCU MOSI connected. 2: Next in chain, ...
2956 // #define X_INTERPOLATE true // Enable to override 'INTERPOLATE' for the X axis
2957 // #define X_HOLD_MULTIPLIER 0.5 // Enable to override 'HOLD_MULTIPLIER' for the X axis
2958 #endif
2959
2960 #if AXIS_IS_TMC_CONFIG(X2)
2961 #define X2_CURRENT X_CURRENT
2962 #define X2_CURRENT_HOME X_CURRENT_HOME
2963 #define X2_MICROSTEPS X_MICROSTEPS
2964 #define X2_RSENSE X_RSENSE
2965 #define X2_CHAIN_POS -1
2966 // #define X2_INTERPOLATE true
2967 // #define X2_HOLD_MULTIPLIER 0.5
2968 #endif
2969
2970 #if AXIS_IS_TMC_CONFIG(Y)
2971 #define Y_CURRENT 800
2972 #define Y_CURRENT_HOME Y_CURRENT
2973 #define Y_MICROSTEPS 16
2974 #define Y_RSENSE 0.022
2975 #define Y_CHAIN_POS -1
2976 // #define Y_INTERPOLATE true
    
```

## 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

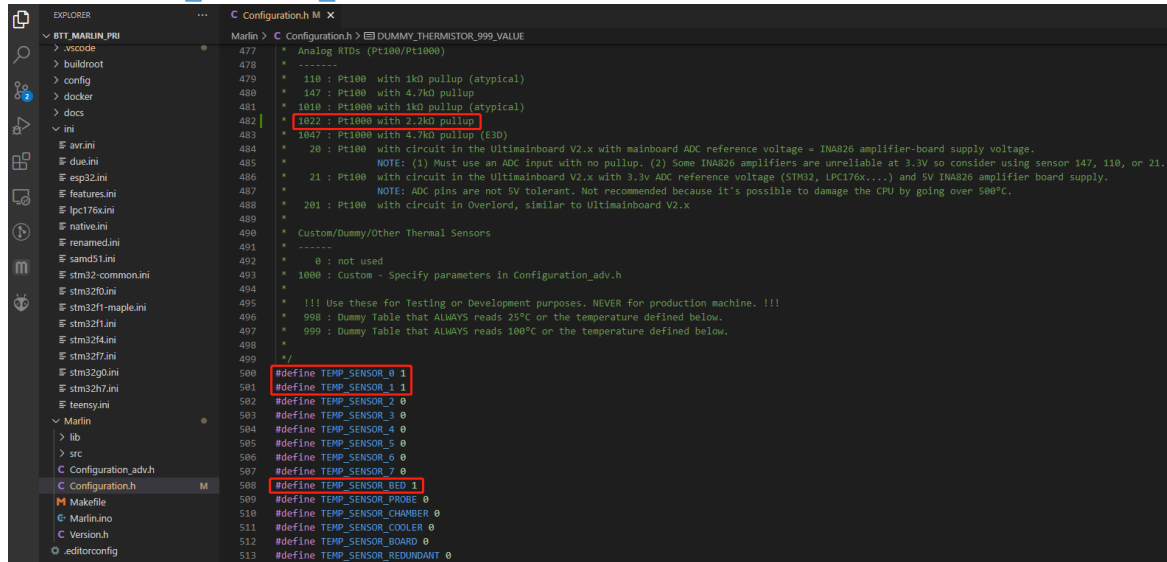
`#define xx_STALL_SENSITIVITY 8` // sensitivity setting, TMC2209 range from 0 to 255, higher number results in more sensitive trigger threshold, sensitivity too high will cause endpoint to trigger before gantry actually moves to the end, lower number results in less sensitive trigger threshold, too low 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.

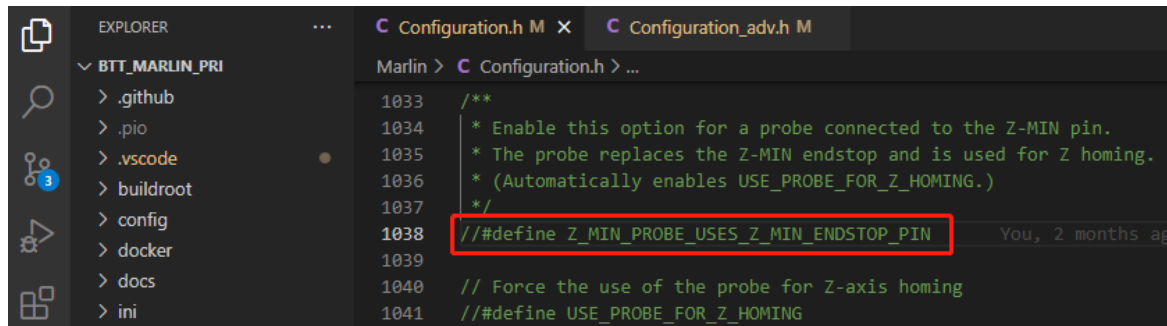
## 100K NTC or PT1000

In Marlin firmware, "1" represents a 100K NTC with a 4.7K pull-up resistor.

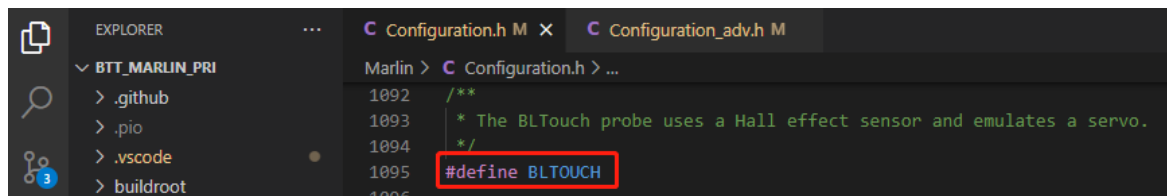
```
#define TEMP_SENSOR_0 1
#define TEMP_SENSOR_1 1
#define TEMP_SENSOR_BED 1
```



## BLTouch



`//#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN` // Do not remap the Z\_PROBE\_PIN to the Z\_MIN port.



`#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.

## 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_INSERT // 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, 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
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.
1469  // #define FIL_RUNOUT_PULLDOWN // Use internal pulldown for filament runout pins.
1470  // #define WATCH_ALL_RUNOUT_SENSORS // Execute runout script on any triggering sensor, not only for the active extruder.
1471  // 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...
1469
1470 // Override individually if the runout sensors vary...
1471
1472 // #define FIL_RUNOUT2_STATE LOW...
1473 // #define FIL_RUNOUT3_STATE LOW...
1474 // #define FIL_RUNOUT4_STATE LOW...
1475 // #define FIL_RUNOUT5_STATE LOW...
1476 // #define FIL_RUNOUT6_STATE LOW...
1477 // #define FIL_RUNOUT7_STATE LOW...
1478 // #define FIL_RUNOUT8_STATE LOW...
1479
1480 // Commands to execute on filament runout.
1481 // With multiple runout sensors use the %c placeholder for the current tool in commands (e.g., "M600 T%c")
1482 // NOTE: After 'M412 H1' the host handles filament runout and this script does not apply.
1483 #define FILAMENT_RUNOUT_SCRIPT "M600"
1484
1485 // After a runout is detected, continue printing this length of filament
1486 // before executing the runout script. Useful for a sensor at the end
1487 // of a feed tube. Requires 4 bytes SRAM per sensor, plus 4 bytes overhead.
1488 #define FILAMENT_RUNOUT_DISTANCE_MM 7
1489
1490 #ifndef FILAMENT_RUNOUT_DISTANCE_MM
1491 // Enable this option to use an encoder disc that toggles the runout pin
1492 // as the filament moves. (Be sure to set FILAMENT_RUNOUT_DISTANCE_MM
1493 // large enough to avoid false positives.)
1494 #define FILAMENT_MOTION_SENSOR
1495 #endif
1496 #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

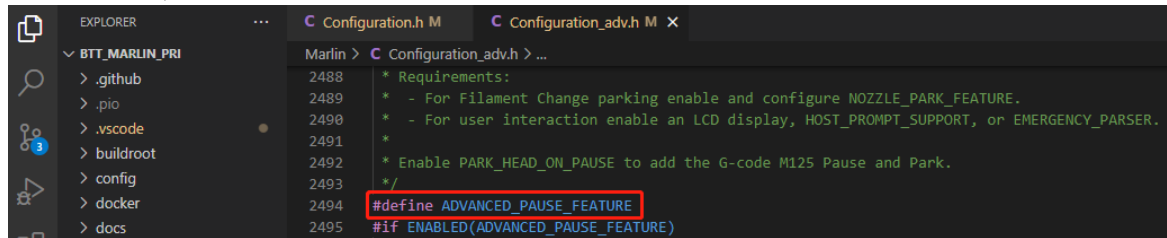
```

```

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
```



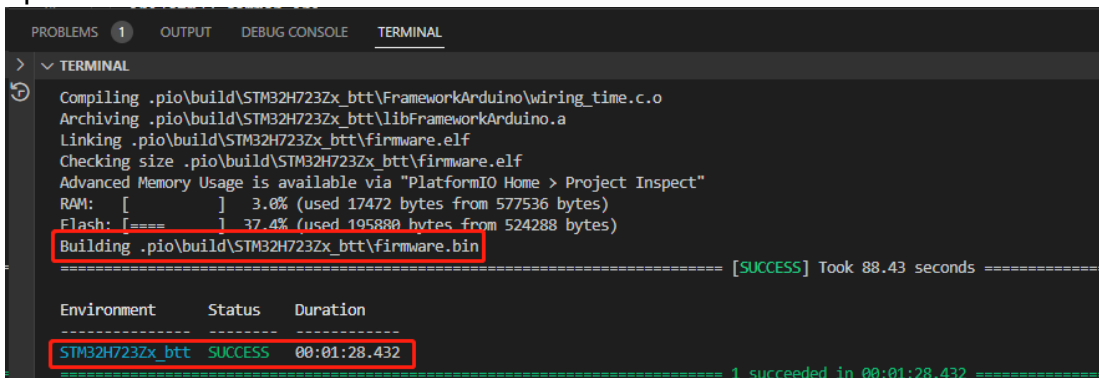
```
#define ADVANCED_PAUSE_FEATURE // retraction setting of nozzle park
movement and filament purge distance after the print is resumed.
```

## Compile Firmware

1. Click "√" to compile firmware.



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



## Klipper

### Compiling the Firmware

- Use the following configuration to compile the firmware (if these options are not available, please update the Klipper firmware source code to the latest 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) --->**

USB Interface

- \* **Communication interface (USB (on PA11/PA12)) --->**

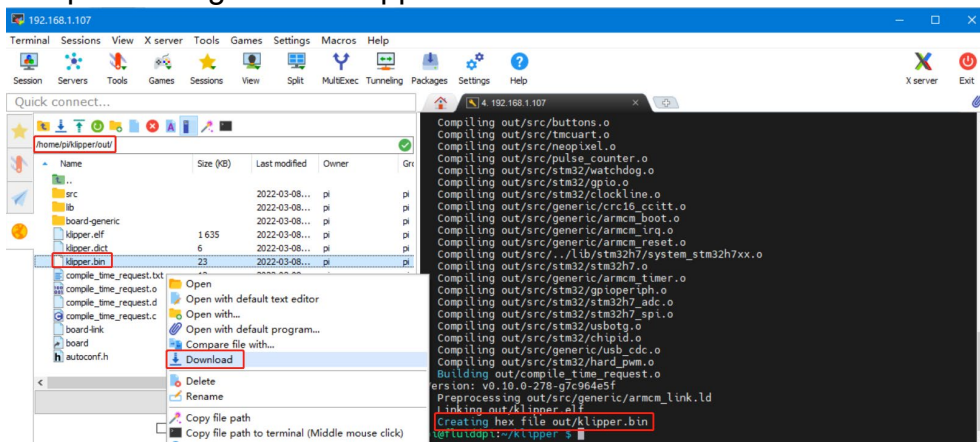
CANBUS Interface

- \* **Communication interface (CAN bus (on PD0/PD1)) --->**

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

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

- Press **q** to exit, and **Yes** when asked to save the configuration.
- 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.



- 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.

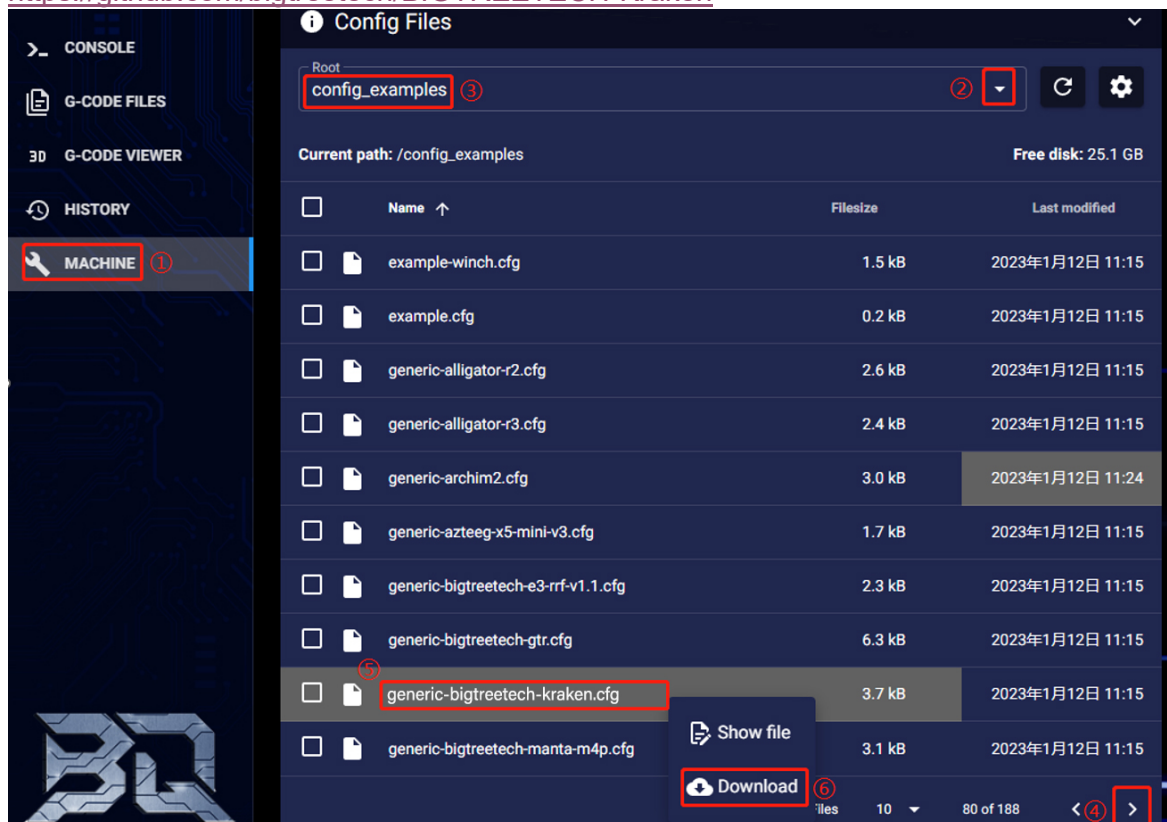
```
pi@fluiddpi:~/klipper $ ls /dev/serial/by-id/  
usb-Klipper_stm32h723xx_41003D001751303232383230-if00  
pi@fluiddpi:~/klipper $
```

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

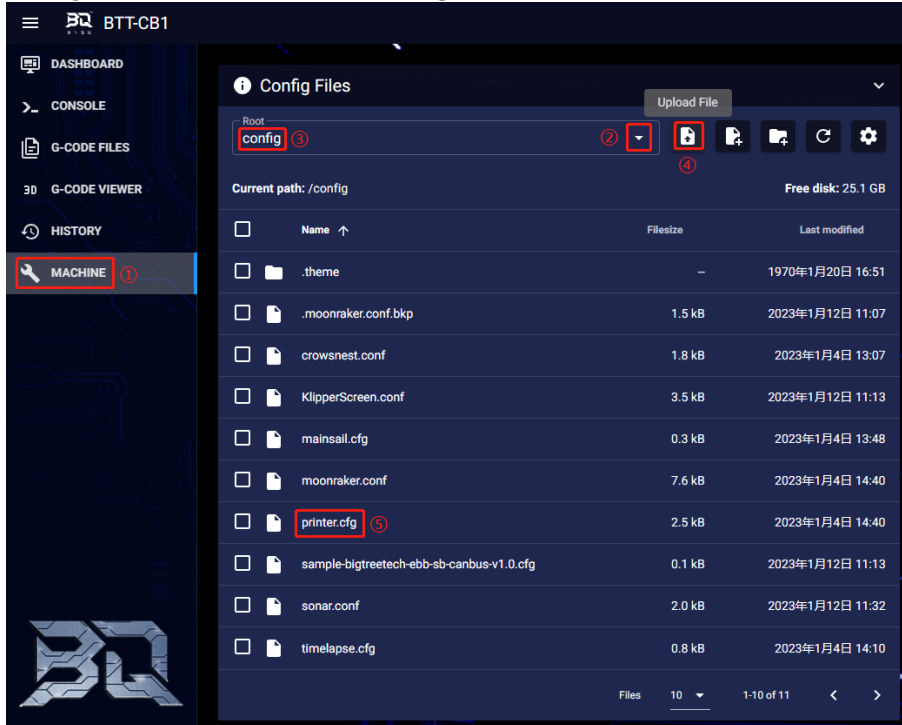
## Configuring Klipper

1. Enter the Raspberry Pi's IP in a browser to access it. Download the motherboard's reference configuration in the path shown in the image below. If you cannot find this file, update the Klipper firmware source code to the latest version or download from GitHub:

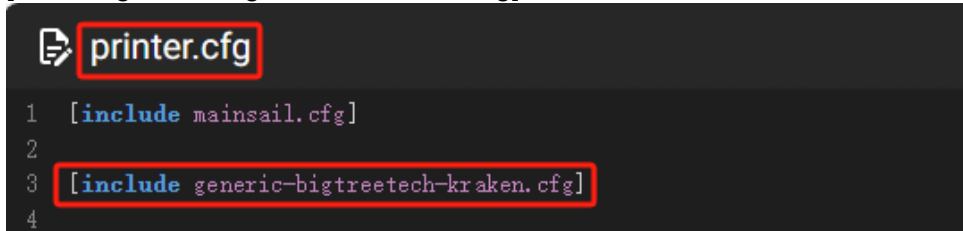
<https://github.com/bigtreetech/BIGTREETECH-Kraken>



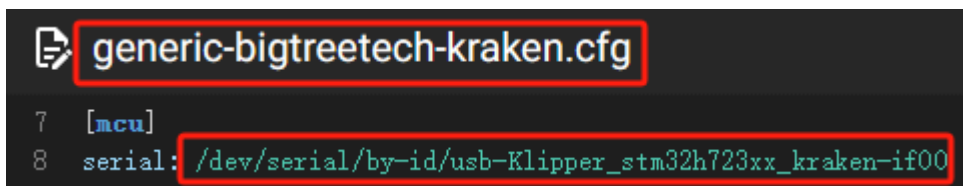
2. Upload the motherboard's configuration file to **Configuration Files** and include this configuration file in the **printer.cfg** file.



[include generic-bigtreetech-kraken.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.

## Firmware Updates

### Updating via microSD

1. Ensure the microSD card is formatted as FAT32.
2. Rename the compiled firmware or the firmware downloaded from GitHub to "firmware.bin" (**note**: make sure the computer system's extension settings are clear, as some users hide the extension, and "firmware.bin" actually displays as "firmware").
3. Copy "firmware.bin" to the root directory of the microSD card.
4. Insert the microSD card into the motherboard's slot, power on the motherboard, and the bootloader will automatically update the firmware.
5. The status LED will blink during update.
6. When it stops and the file is renamed "FIRMWARE.CUR", the update is complete.

### Updating Klipper via DFU

1. Run `ls /dev/serial/by-id/` to get the board ID. If Klipper is running, it will return a klipper ID.

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

2. If `ls /dev/serial/by-id/` is able to find the MCU's klipper device ID, you can directly input: `cd ~-klipper`  
`make flash FLASH_DEVICE=/dev/serial/by-id/usb-Klipper_stm32h723xx_41003D001751303232383230-if00`  
to write firmware (Note: Replace `/dev/serial/by-id/xxx` with the actual ID queried in the previous step)

```
pi@fluiddpi:~/klipper $ make flash FLASH_DEVICE=/dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000050415833323520-if00
Building hid-flash
/bin/sh: 1: pkg-config: not found
hid-flash requires libusb-1.0, please install with:
sudo apt-get install libusb-1.0
Flashing out/klipper.bin to /dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000050415833323520-if00
Entering bootloader on /dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000050415833323520-if00
Device reconnect on /sys/devices/platform/soc/52000000.usb/usb/l1/1-1/1-1.1:1.0
sudo dfu-util -p 1-1.1-a 0 -s 0x08020000:leave -D out/klipper.bin
dfu-util 0.9

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

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
Runtime device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 1024
DFU interface name: "Internal Flash"
Downloading to address = 0x08002000, size = 25264
Download [=====] 100% 25264 bytes
Download done.
File downloaded successfully
dfu-util: Error during download get_status

Failed to flash to /dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000050415833323520-if00: Error running dfu-util
If the device is already in bootloader mode it can be flashed with the
following command:
make flash FLASH_DEVICE=0483:df11
OR
make flash FLASH_DEVICE=1209:beba
If attempting to flash via 3.3V serial, then use:
make serialflash FLASH_DEVICE=/dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000050415833323520-if00
```

After writing completes, there may be an error message **dfu-util: Error during download get\_status**, just ignore it.

## Precautions

1. Forbidden to switch driver voltage when stepper motors are in motion;
2. When switching stepper motor driver voltage, ensure no control signals are being output from the MCU to the driver chips;
3. When driver current exceeds 7A, it is recommended to add a cooling fan for the driver for heat dissipation.

If you need further resources for this product, you can find them at [GitHub](<https://github.com/bigtreetech/>). If you cannot find what you need, you may contact our after-sales support([service005@biqu3d.com](mailto:service005@biqu3d.com)).

If you encounter any other problems during use or have suggestions or feedback, please contact us. Thank you for choosing BIGTREETECH products.