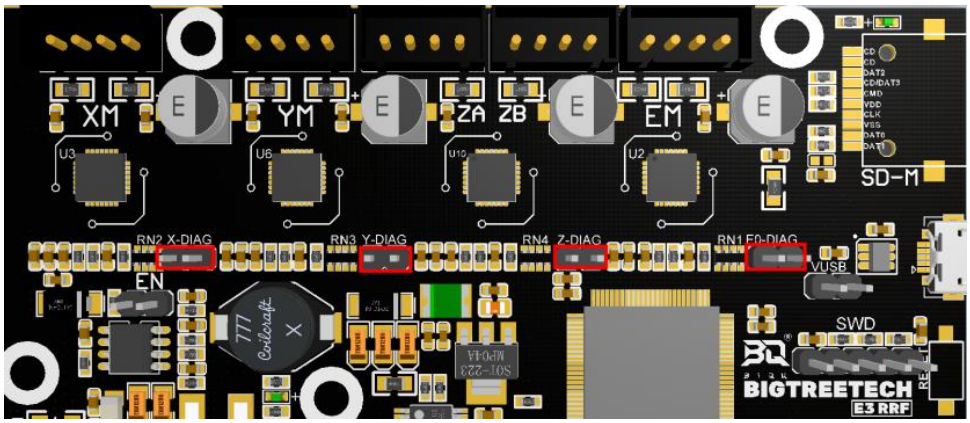
Introduction

This guide is most useful for users who are looking for advanced DIY features from their E3 RRF. If you are a user who has a standard i3 style installation then you can use the pre-compiled firmware.bin file available on our github site.

This guide is arranged to show how each additional feature on the E3 RRF can be used by configuring the hardware and firmware. Each sub-heading covers a separate feature.

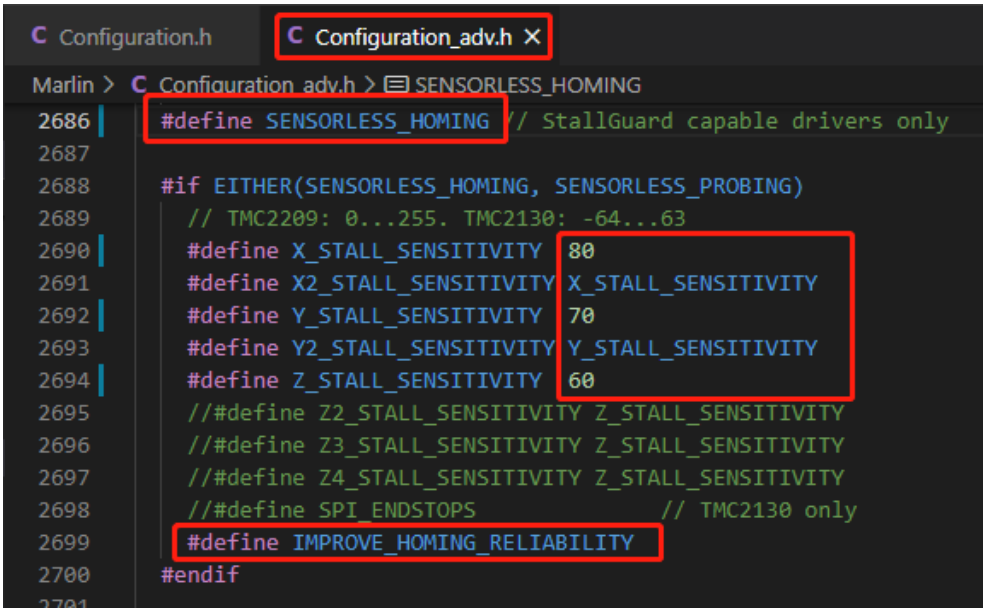
TMC2209 Sensorless homing

1. Jumper configuration



Disconnect the limit switch from the motherboard for each axis that uses sensorless homing and close the corresponding DIAG circuit with a jumper. Note that it does not make sense to close the DIAG jumper on the extruder since a stall on the extruder does not indicate a runout.

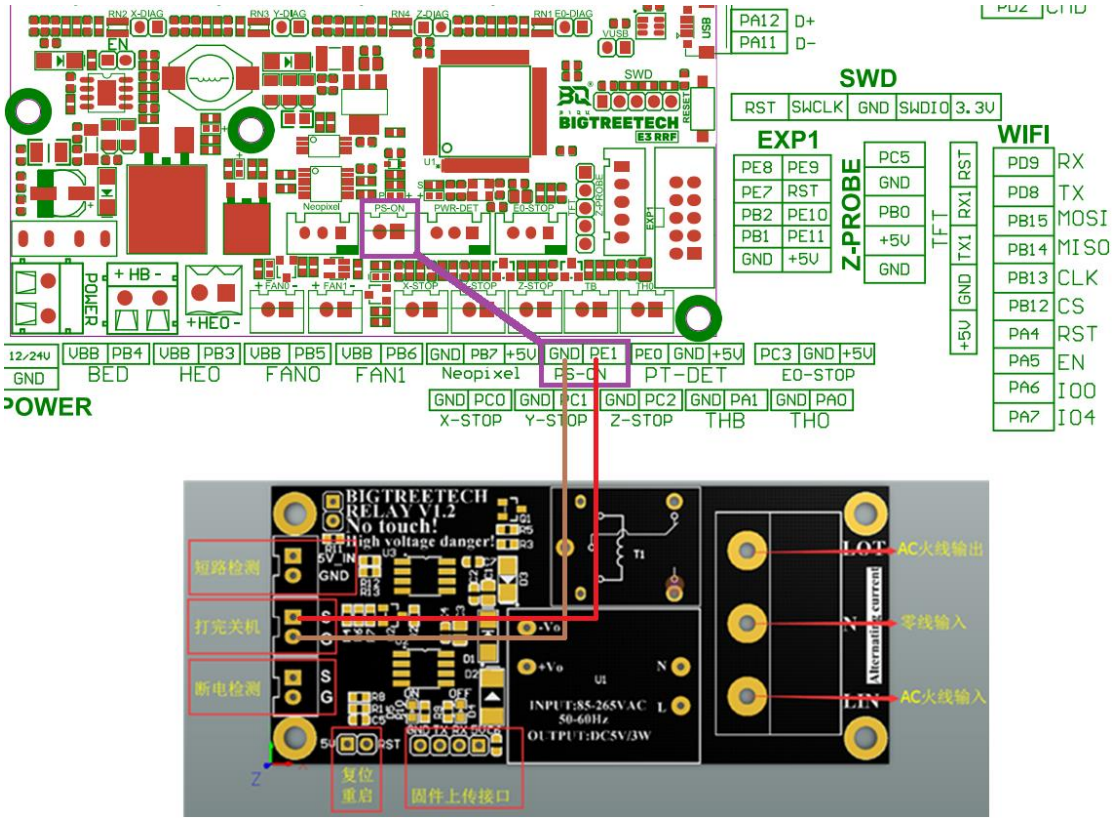
2. Firmware configuration



#define SENSORLESS_HOMING // enable sensorless homing with StallGuard
#define xxx_STALL_SENSITIVITY 80 // For sensorless homing on the TMC2209 the sensitivity needs to be calibrated for each installation. If you set it too high then the axis will stop before it reaches the correct endstop point. If you set it too low then the axis will grind up against the endstop point without stopping. You can use M914 to change the sensitivity on an endstop during runtime. Our test on ender3 shows it would be more appropriate to set “X=80, Y=70, Z=60” although this can vary with belt tightness and other factors.
#define IMPROVE_HOMING_RELIABILITY//You can individually set the homing current (X_CURRENT_HOME) to improve homing reliability. Generally sensorless homing works better when the homing current is set lower.

Automatic power shutdown after print (Relay V1.2 module required)

1. Wiring diagram



Insert the control PIN of the module into the PS-ON port of the motherboard as per the diagram above.

2. Firmware configuration

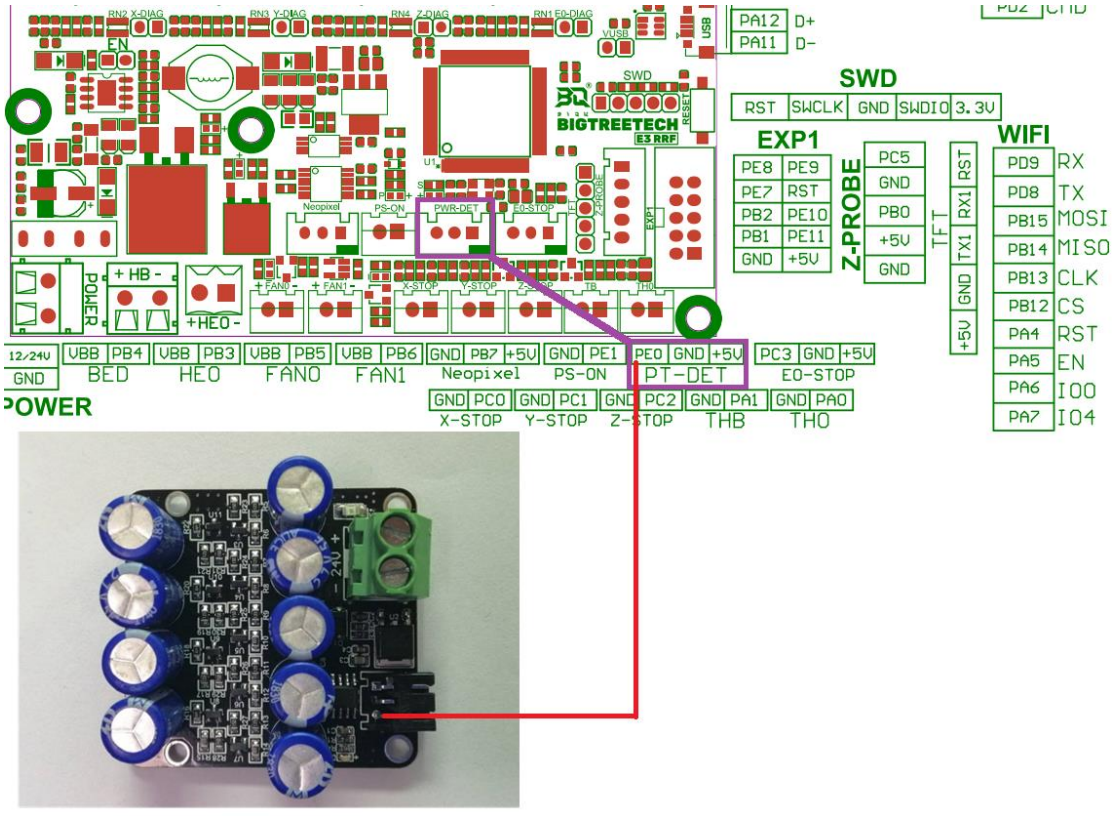
```
Configuration.h ×   Configuration_adv.h
Marlin > Configuration.h > PSU_CONTROL
314  #define PSU_CONTROL
315  #define PSU_NAME "Power Supply"
316
317  #if ENABLED(PSU_CONTROL)
318    #define PSU_ACTIVE_STATE HIGH     // Set 'LOW' for ATX, 'HIGH' for X-Box
319
320    //#define PSU_DEFAULT_OFF         // Keep power off until enabled directly with M80
321    //#define PSU_POWERUP_DELAY 250   // (ms) Delay for the PSU to warm up to full power
322
323    //#define PSU_POWERUP_GCODE  "M355 S1"  // G-code to run after power-on (e.g., case light on)
324    //#define PSU_POWEROFF_GCODE "M355 S0"  // G-code to run before power-off (e.g., case light off)
325
326    //#define AUTO_POWER_CONTROL      // Enable automatic control of the PS_ON pin
327    #if ENABLED(AUTO_POWER_CONTROL)
328      #define AUTO_POWER_FANS         // Turn on PSU if fans need power
329      #define AUTO_POWER_E_FANS
330      #define AUTO_POWER_CONTROLLERFAN
331      #define AUTO_POWER_CHAMBER_FAN
332      //#define AUTO_POWER_E_TEMP        50 // (°C) Turn on PSU if any extruder is over this temperature
333      //#define AUTO_POWER_CHAMBER_TEMP  30 // (°C) Turn on PSU if the chamber is over this temperature
334      #define POWER_TIMEOUT            30 // (s) Turn off power if the machine is idle for this duration
335      //#define POWER_OFF_DELAY          60 // (s) Delay of poweroff after M81 command. Useful to let fans run for extra time.
336    #endif
337  #endif
```

#define PSU_CONTROL // Power on/off PSU control with M80/M81

#define PSU_ACTIVE_STATE HIGH // Our Relay V1.2 module is on at a high (logic) level and off at a low level, so the state needs to be set to HIGH, which is different from other shutdown after print modules.

UPS Module (BTT UPS 24V)

1. Wiring diagram

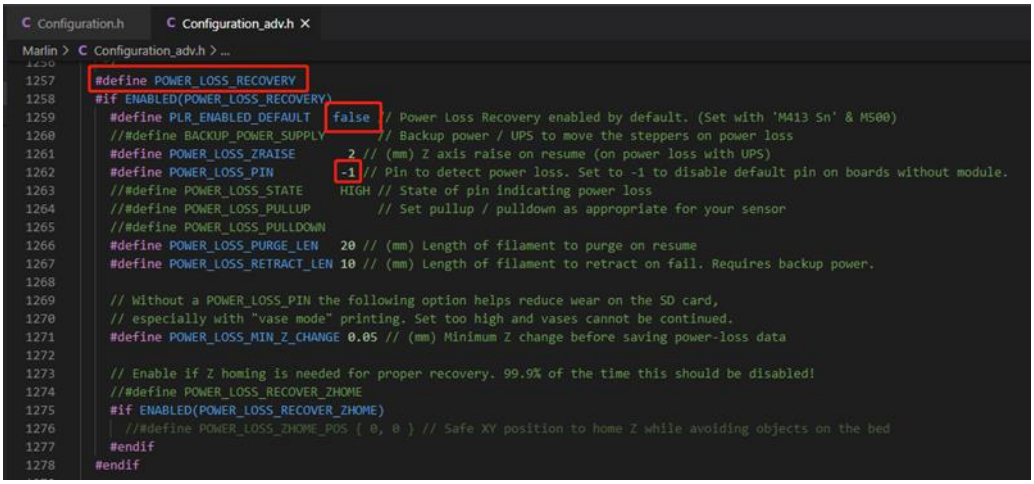


Insert the feedback signal line (PIN) of the module into the PWR-DET port on the motherboard

2. Firmware configuration

Power loss recovery can be achieved with the following methods:

a. No external module is required. The printing status is saved to the SD card periodically throughout the print. If the power is removed during a print then the print will be resumed from the last known save point on the SD card. The saving frequency may be as low as once per layer depending on your firmware. The resume location may therefore not be exactly from where the print stopped. Additionally constant writes to the SD card can shorten its life meaningfully.



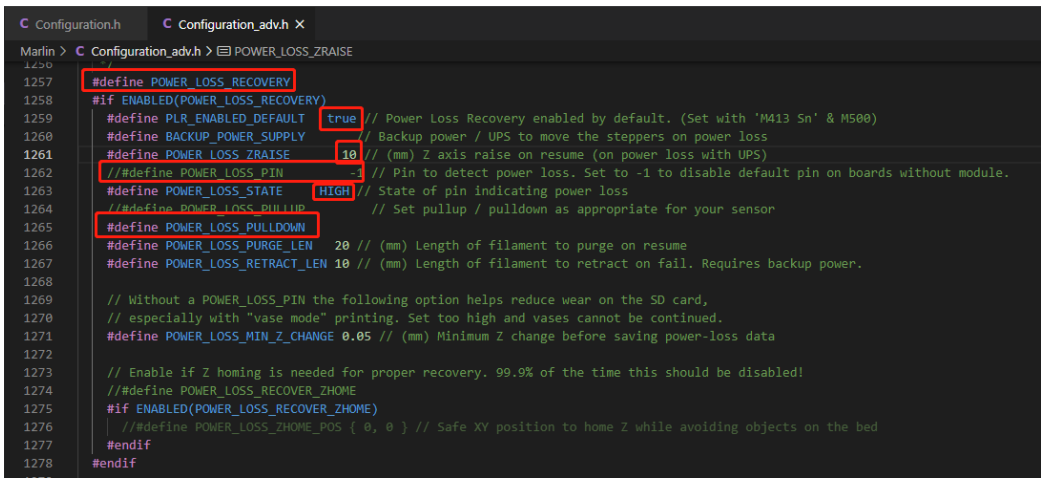
#define POWER_LOSS_RECOVERY // enable power-loss recovery feature in the firmware

#define PLR_ENABLED_DEFAULT false // false means disable the feature by default. Can be changed with M413 S1.

#define POWER_LOSS_PIN -1 // Pin to detect power loss. Set to -1 to disable default pin on boards without external detection module.

Power loss recovery is an enabled feature in the firmware.bin file on our Github but as it affects the life of SD card it is set to OFF by default. You can turn it on in "Configuration->Power Outage" on the 12864 screen and save settings in "Configuration->Store Settings", or enable power-loss recovery with “M413 S1” followed by “M500”.

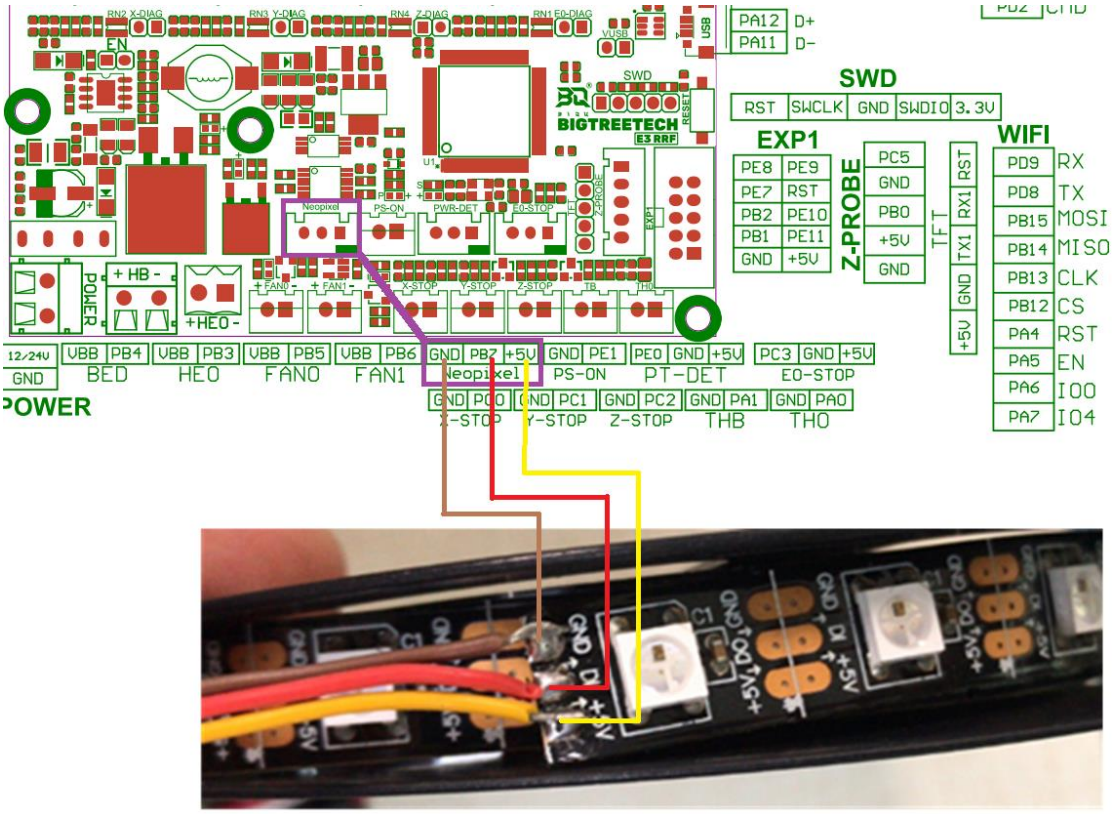
By using UPS modules like BTT UPS 24V, you can keep the motherboard powered for a brief period even after the mains power is removed. The module will then send a signal to the motherboard to indicate that mains power has been lost which will prompt the motherboard to save the printing state and raise the z axis away from the print. This method will only write date to the SD card once the mains power is removed and therefore will not affect the life of the SD card.



#define POWER_LOSS_RECOVERY // include power-loss recovery feature in the firmware
#define PLR_ENABLED_DEFAULT true // true means power-loss recovery enabled by default
#define POWER_LOSS_ZRAISE 10 // the nozzle will raise 10mm, which protects against damaged print part
//#define POWER_LOSS_PIN -1 // Don’t set the power loss pin here. The correct pin is already set within the pins file for the motherboard.
#define POWER_LOSS_STATE HIGH // This means that marlin will detect a power loss when the input pin goes high. Since the BTT UPS 24V module outputs a high level when the mains power is lost, this is the correct setting.
#define POWER_LOSS_PULLDOWN // This is an essential line since it will ensure that there are no false triggers detected from the BTT UPS 24V module.

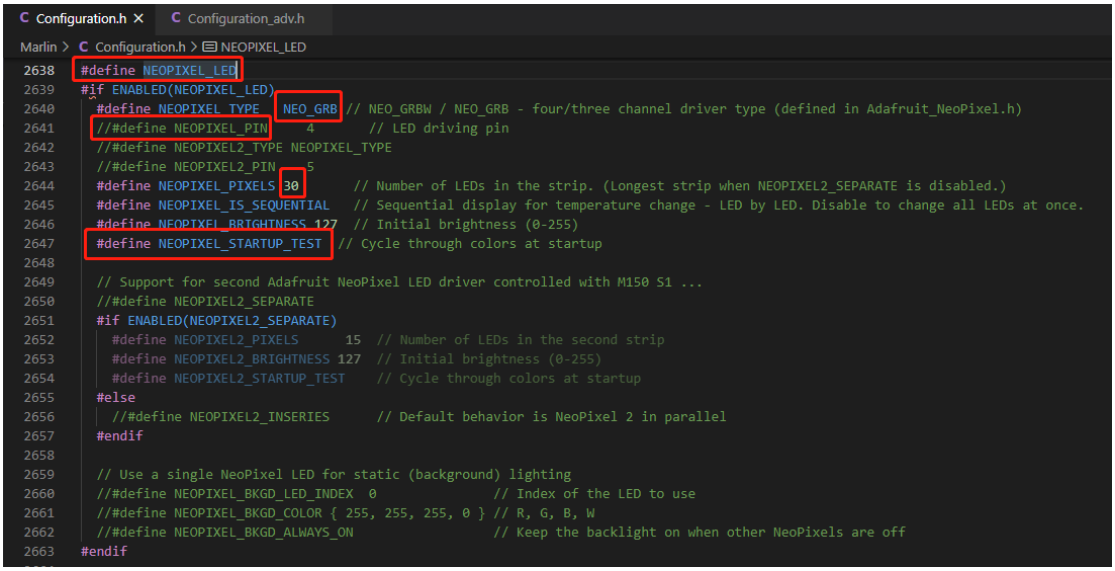
RGB strip (WS2812,etc)

1. Wiring diagram



When connecting an RGB light onto the Neopixel port of the motherboard, pay close attention to the wiring of the GND, signal, and +5V lines. Incorrect wiring could result in damage to the motherboard or the peripheral board.

2. Firmware configuration



#define NEOPIXEL_LED //enable Neopixel feature

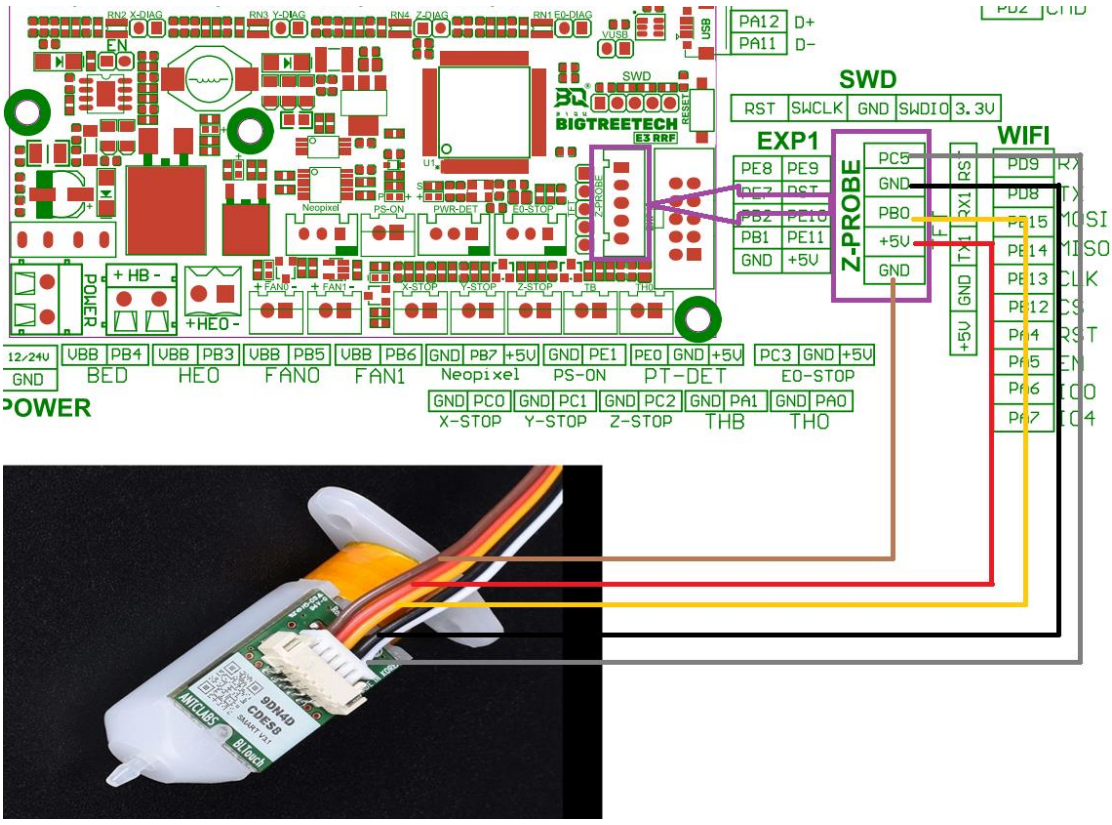
#define NEOPIXEL_TYPE NEO_GRB // set the type of RGB light

//#define NEOPIXEL_PIN 4 // Don't set the pin here. It's already set in the motherboard pins file.
#define NEOPIXEL_PIXELS 30 // Set this to the number of LEDs that are connected.
#define NEOPIXEL_STARTUP_TEST // during startup, it will display red, green, and blue in turn for easy testing
#define LED_CONTROL_MENU // Adds a menu to control the colors of LED on the screen

```
Configuration.h   Configuration_adv.h ×
Marlin > Configuration_adv.h > LED_CONTROL_MENU
1128  /**
1129   * LED Control Menu
1130   * Add LED Control to the LCD menu
1131   */
1132  #define LED_CONTROL_MENU
1133  #if ENABLED(LED_CONTROL_MENU)
1134    #define LED_COLOR_PRESETS                 // Enable the Preset Color menu option
1135    //#define NEO2_COLOR_PRESETS              // Enable a second NeoPixel Preset Color menu option
1136    #if ENABLED(LED_COLOR_PRESETS)
1137      #define LED_USER_PRESET_RED        255  // User defined RED value
1138      #define LED_USER_PRESET_GREEN      128  // User defined GREEN value
1139      #define LED_USER_PRESET_BLUE         0  // User defined BLUE value
1140      #define LED_USER_PRESET_WHITE      255  // User defined WHITE value
1141      #define LED_USER_PRESET_BRIGHTNESS 255  // User defined intensity
1142      //#define LED_USER_PRESET_STARTUP       // Have the printer display the user preset color on startup
1143    #endif
1144    #if ENABLED(NEO2_COLOR_PRESETS)
1145      #define NEO2_USER_PRESET_RED        255  // User defined RED value
1146      #define NEO2_USER_PRESET_GREEN      128  // User defined GREEN value
1147      #define NEO2_USER_PRESET_BLUE         0  // User defined BLUE value
1148      #define NEO2_USER_PRESET_WHITE      255  // User defined WHITE value
1149      #define NEO2_USER_PRESET_BRIGHTNESS 255  // User defined intensity
1150      //#define NEO2_USER_PRESET_STARTUP       // Have the printer display the user preset color on startup for the second strip
1151    #endif
1152  #endif
```

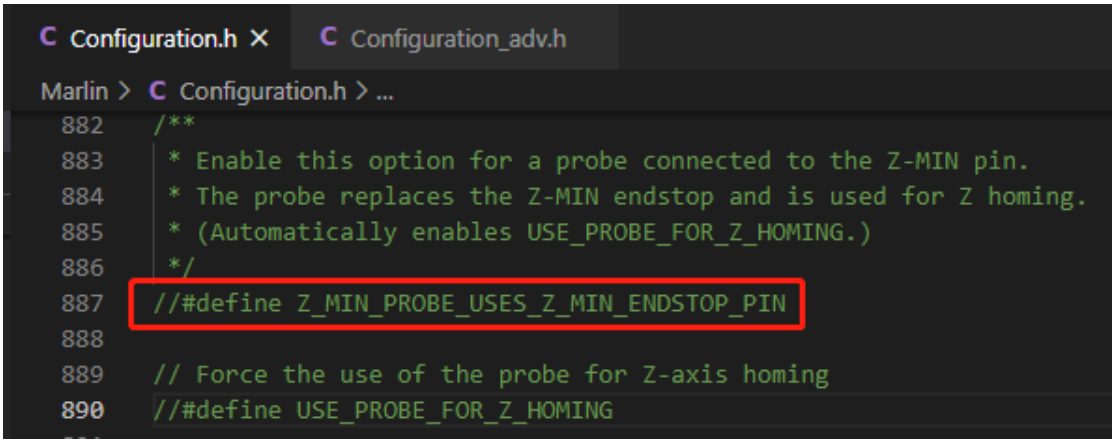
BL Touch

1. Wiring diagram

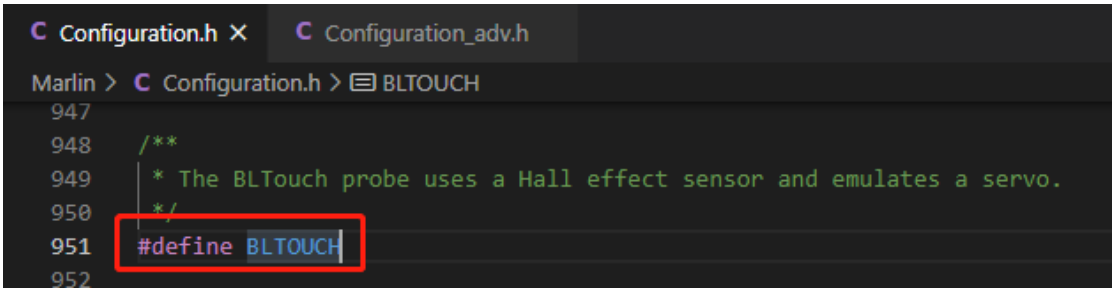


Connect the 3pin and a 2pin signals of the BL Touch to the 5pin Z-PROBE port of the motherboard according to the wiring shown in the image. If your BL touch does not have the same colour wiring as the one shown in the image then remove the connector and check whether there is a silk screen marking showing the pinout. If there is then make sure that the pinout is wired in the correct order to the z-probe connector.

2. Firmware configuration



//#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN Commenting this out will cause the probe to use the dedicated probe input pin. If you want to use the probe as the homing endstop then you can use the "USE_PROBE_FOR_Z_HOMING" definition below that statement. This is explained a little later in this document.



In configuration_adv.h make the following changes:

#define BLTOUCH // enable BLTOUCH

#define NOZZLE_TO_PROBE_OFFSET { -40, -10, -1.85 } // You will need to adjust this to the offset between your probe and nozzle.
#define PROBING_MARGIN 10 // set the probe margin for leveling, for example, the maximum probe area for Ender3 is 0-235 however, probing the full volume will likely hit bed clips or miss the bed slightly and therefore a margin is recommended.
PROBING_MARGIN = 10 reduces the probe area for leveling to 10-225.

Configuration.h | Configuration_adv.h

Marlin > Configuration.h > ...

```
1372   */
1373  //#define AUTO_BED_LEVELING_3POINT
1374  //#define AUTO_BED_LEVELING_LINEAR
1375  #define AUTO_BED_LEVELING_BILINEAR
1376  //#define AUTO_BED_LEVELING_UBL
1377  //#define MESH_BED_LEVELING
1378
1379  /**
1380   * Normally G28 leaves leveling disabled on completion. Enable one of
1381   * these options to restore the prior leveling state or to always enable
1382   * leveling immediately after G28.
1383   */
1384  #define RESTORE_LEVELING_AFTER_G28
1385  //#define ENABLE_LEVELING_AFTER_G28
1386
```

#define AUTO_BED_LEVELING_BILINEAR // configure bed leveling

#define RESTORE_LEVELING_AFTER_G28 //restore leveling after G28

Configuration.h | Configuration_adv.h

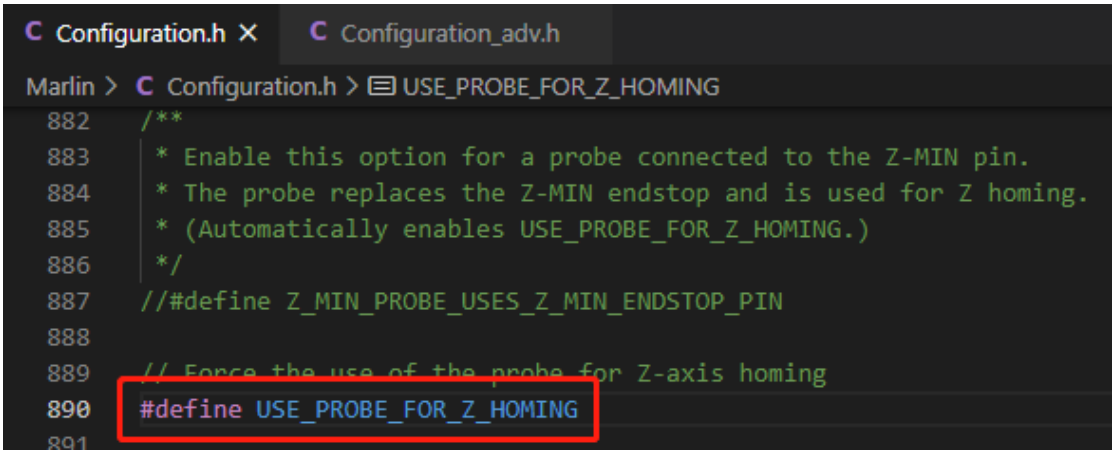
Marlin > Configuration.h > AUTO_BED_LEVELING_BILINEAR

```
1430    #endif
1431
1432  #endif
1433
1434  #if EITHER(AUTO_BED_LEVELING_LINEAR, AUTO_BED_LEVELING_BILINEAR)
1435
1436    // Set the number of grid points per dimension.
1437    #define GRID_MAX_POINTS_X 5
1438    #define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X
1439
1440    // Probe along the Y axis, advancing X after each column
1441    //#define PROBE_Y_FIRST
1442
1443    #if ENABLED(AUTO_BED_LEVELING_BILINEAR)
1444
1445      // Beyond the probed grid, continue the implied tilt?
1446      // Default is to maintain the height of the nearest edge.
1447      #define EXTRAPOLATE_BEYOND_GRID
1448
1449      //
1450      // Experimental Subdivision of the grid by Catmull-Rom method.
1451      // Synthesizes intermediate points to produce a more detailed mesh.
1452      //
1453      //#define ABL_BILINEAR_SUBDIVISION
1454      #if ENABLED(ABL_BILINEAR_SUBDIVISION)
1455        // Number of subdivisions between probe points
1456        #define BILINEAR_SUBDIVISIONS 3
1457      #endif
1458
1459    #endif
```

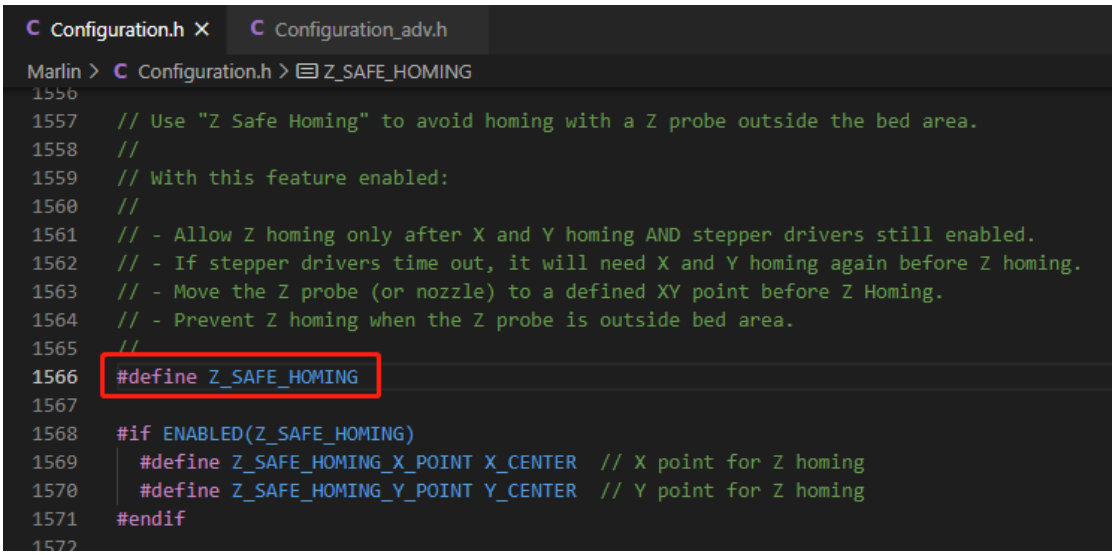
#define GRID_MAX_POINTS_X 5 //set the number of grid points, and probe 5 points along the X-axis. Configure this according to your needs.

#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X // probe 5 points along the Y-axis

If you use BLTouch instead of Z-axis mechanical switch as Z-axis limit switch, you just modify the firmware setting without changing the wiring of BLTouch.



#define USE_PROBE_FOR_Z_HOMING // use Z Probe (BLTouch) as Z-axis homing limit switch

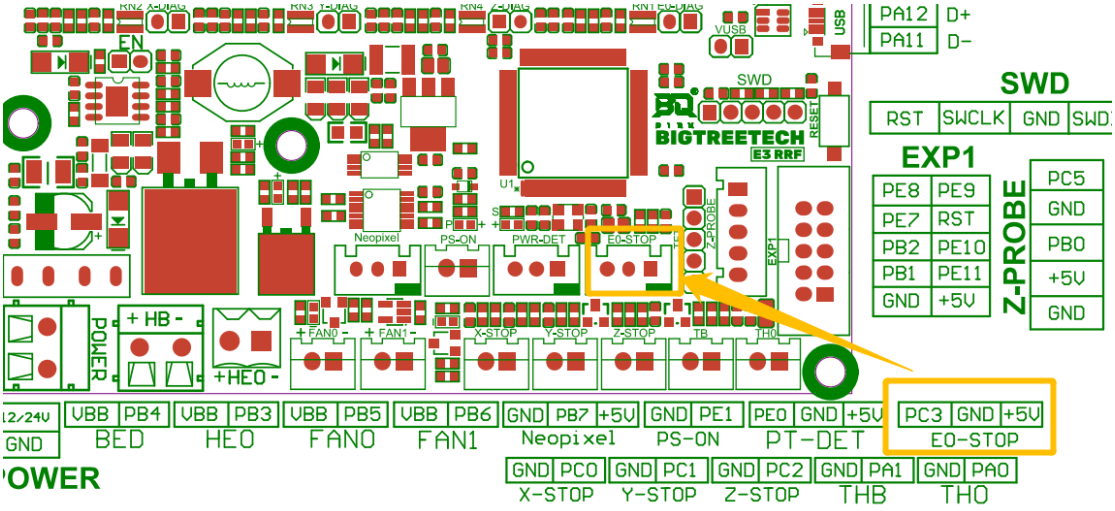


If you decide to use the z-probe for homing then it is recommended that you enable z safe homing.

#define Z_SAFE_HOMING // prevents the Z-Probe (BLTouch) is outside bed area by moving to a defined XY point (by default, the middle of the bed) before Z homing.

Filament runout detection module

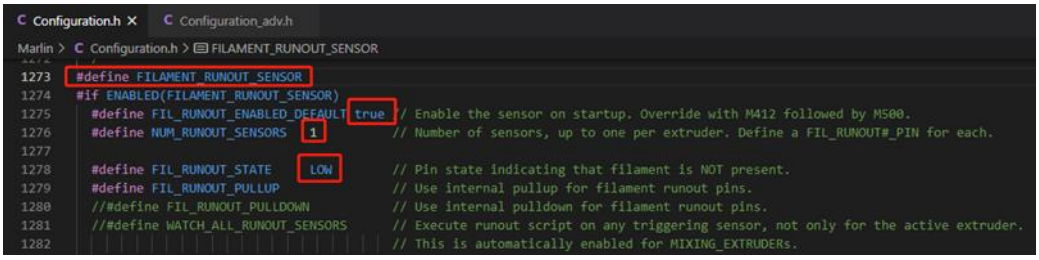
1. Wiring diagram



2. Firmware configuration

This motherboard currently supports two filament runout detection modules:

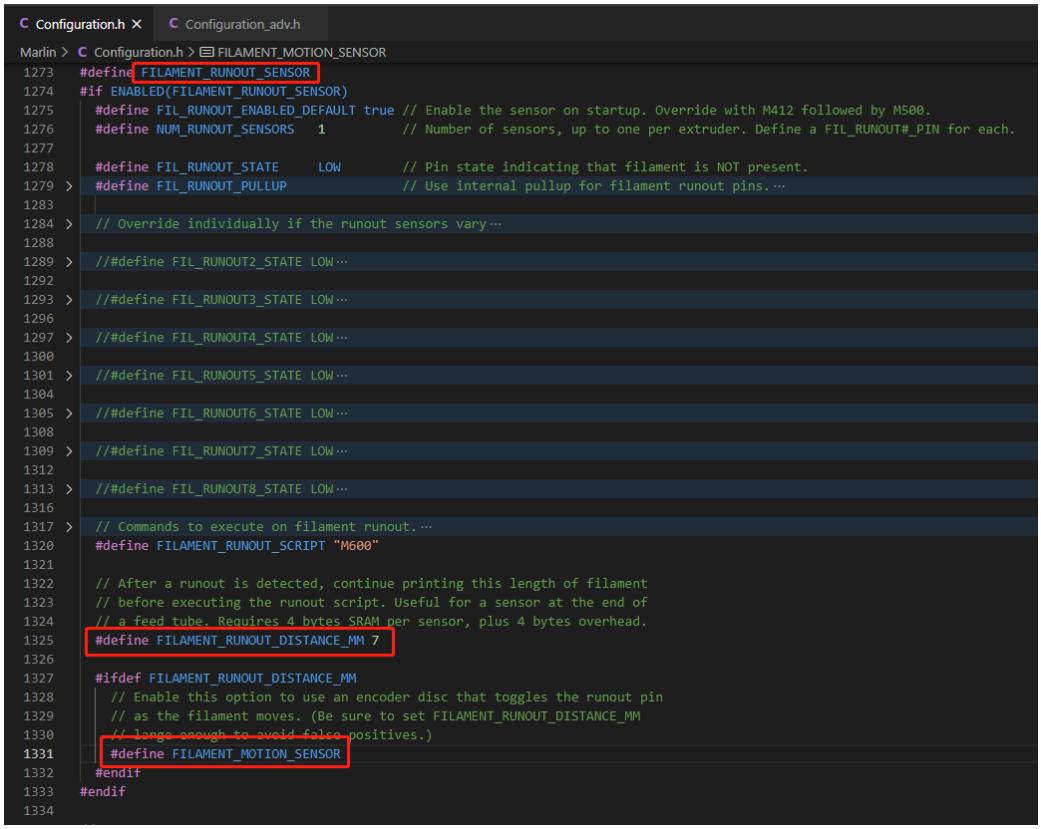
a. The standard filament runout detection module. This is generally nothing more than a mechanical switch which can detect whether filament is present within the path that runs over the switch.



#define FILAMENT_RUNOUT_SENSOR // include filament runout detection in the firmware
#define FIL_RUNOUT_ENABLED_DEFAULT true // the filament runout detection is ON by default. You can disable it in " Configuration->Runout Sensor" and save setting in "Configuration->Store Settings" on the LCD12864, or disable it with "M412 S0" followed by "M500".

#define NUM_RUNOUT_SENSORS 1 // number of filament runout sensors
#define FIL_RUNOUT_STATE LOW // If the filament runs out, the module produces a low logic level and the state is set to "runout".

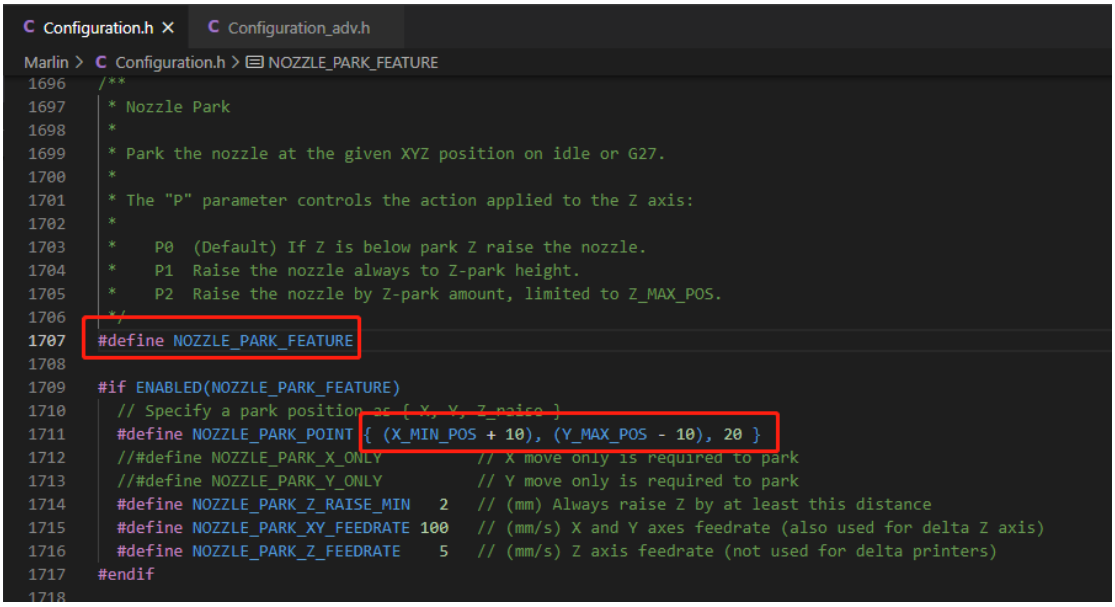
b. BIGTREETECH Smart Filament Sensor – This sensor continuously sends a pulse train to the motherboard when filament is passing through it. When there is no filament passing through it or when then filament jams it will stop sending a pulse train to the motherboard, thereby allowing the motherboard to detect a runout or jammed condition.



#define FILAMENT_MOTION_SENSOR // set the type of filament runout sensor

#define FILAMENT_RUNOUT_DISTANCE_MM 7 // This allows for 7mm of filament to be extruded without the sensor producing a pulse. You may need to change this in the firmware to suit your setup if you find that you are getting false triggers.

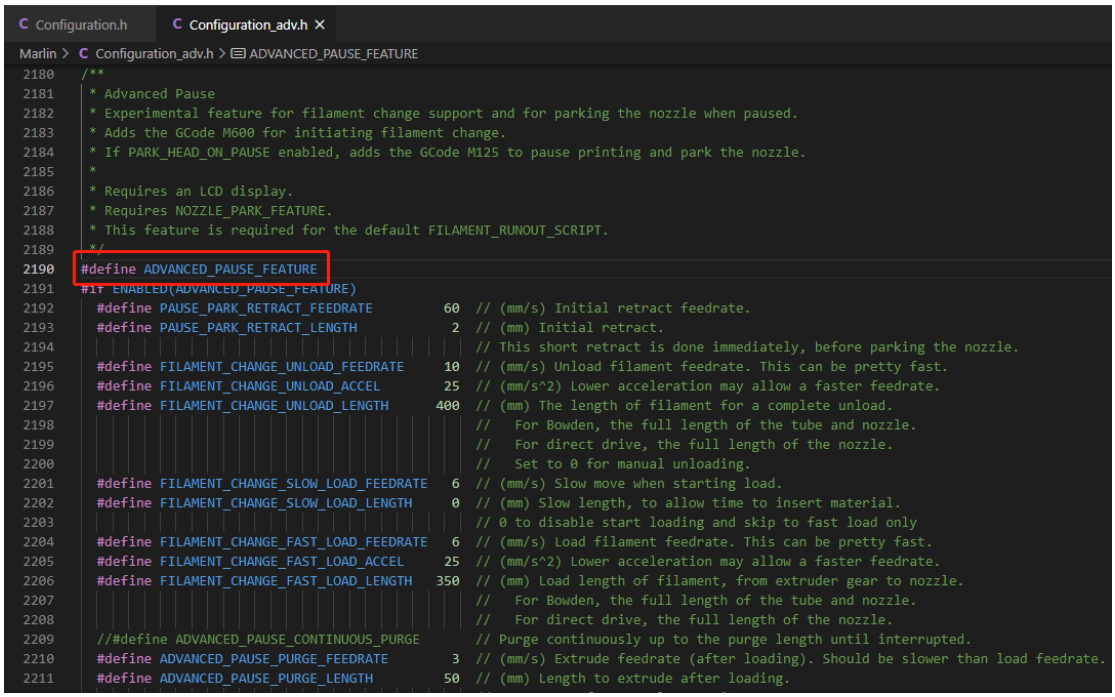
If a runout or blockage is detected, then the firmware can park the nozzle while you correct the issue. Make the following changes to enable this feature.



#define NOZZLE_PARK_FEATURE // nozzle pause feature

#define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 } //specify the XY position and Z-

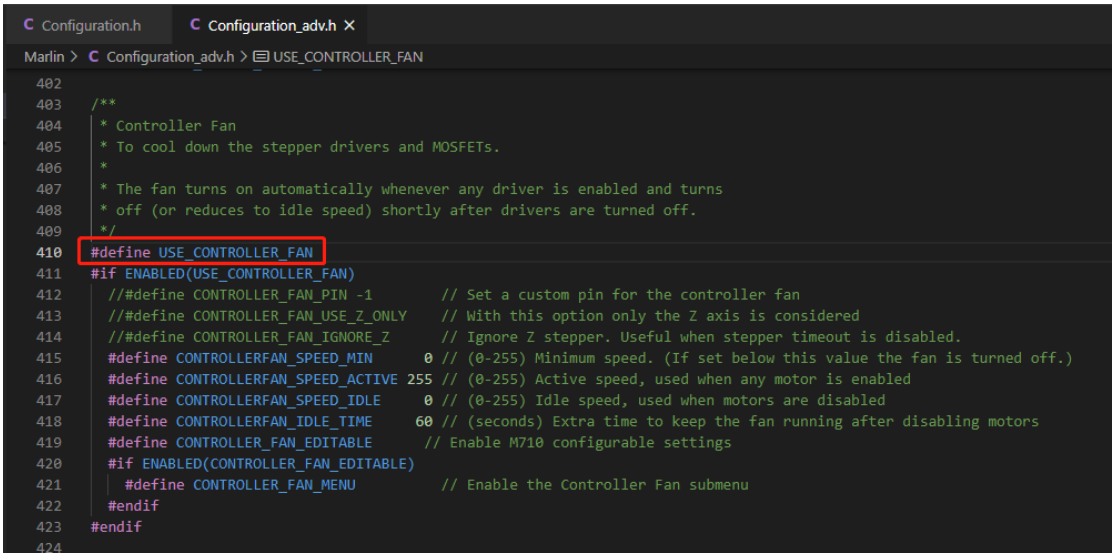
raise height when the nozzle pauses
#define ADVANCED_PAUSE_FEATURE // set the filament retraction length and speed when it pauses, and filament unload length once the nozzle reaches the park point.



Fan configuration

For the Ender3, by default, the cooling fan for the hotend is always-on and directly connected to the 24V power supply.

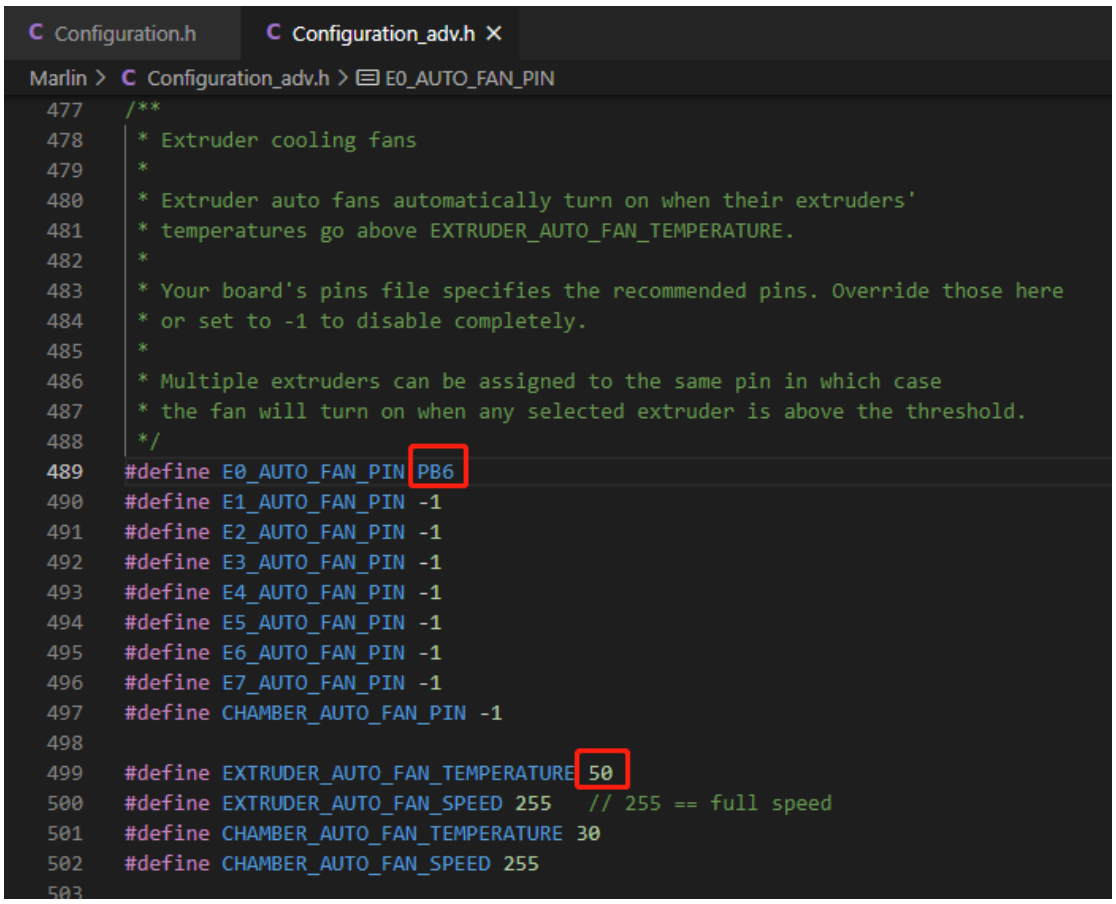
The FAN1 port on the motherboard is a PWM controllable fan. By default, the motherboard cooling fan is connected to the FAN1 port and you can turn this fan on to cool the motherboard during prints or when the bed is being heated. Use the following firmware configuration to do so.



If, instead, you want to control the hotend cooling fan, you can connect the hotend to the FAN1 port and then apply the configuration below. Note that you can also connect the motherboard cooling fan in parallel with the hotend cooling fan using the configuration below since it will cause the motherboard fan to turn on when the hotend is on which is the general condition during a print when the motherboard needs cooling.

First, disable the config that only enables the motherboard cooling fan:
#define USE_CONTROLLER_FAN

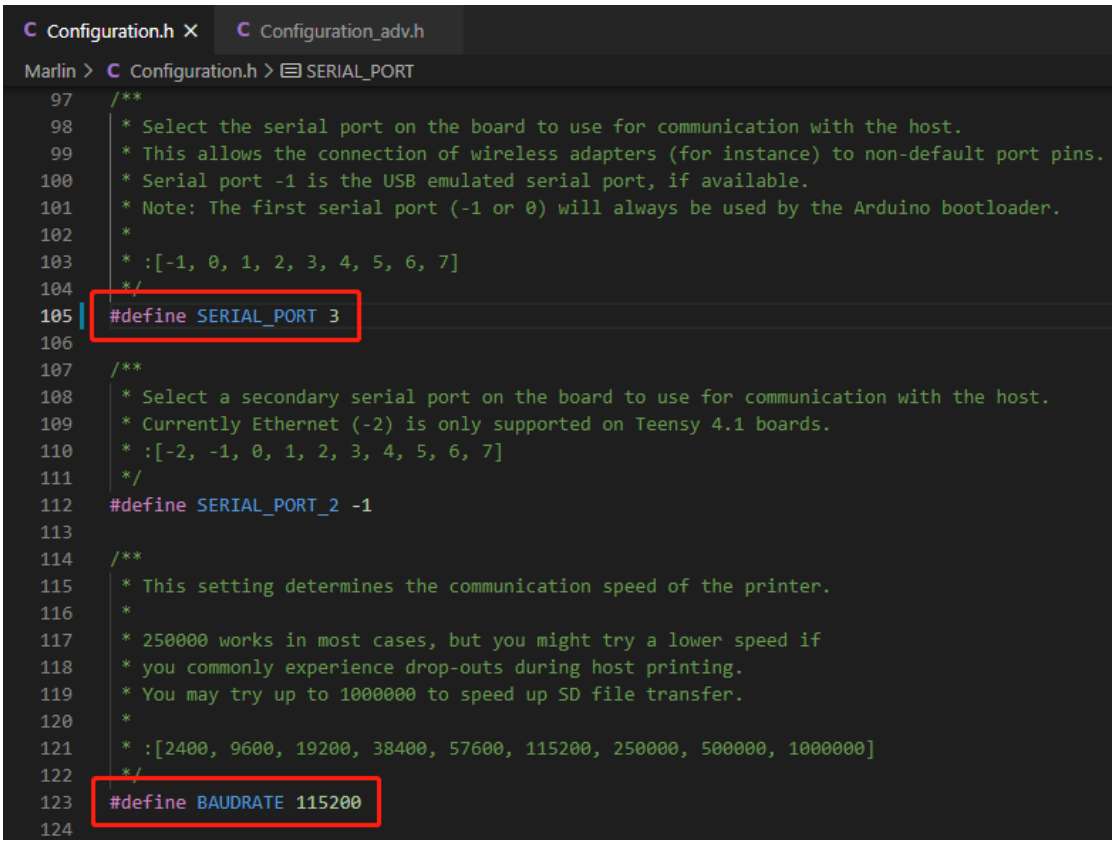
Next, enable config that will allow the hotend cooling fan to be controllable by setting the E0_AUTO_FAN_PIN as shown below.



#define E0_AUTO_FAN_PIN PB6 // set PB6(FAN1) to E0 AUTO FAN
#define EXTRUDER_AUTO_FAN_TEMPERATURE 50 // turn on the cooling fan when the nozzle temperature exceeds 50

ESP3D

The onboard ESP8266 module allows Marlin to use the Wi-Fi control panel of the open source ESP3D software without any additional hardware. Just set the correct “SERIAL_PORT” and “BAUDRATE” in Marlin. The serial port for communication between ESP8266 onboard and Marlin is UART3, so SERIAL_PORT is set to 3.

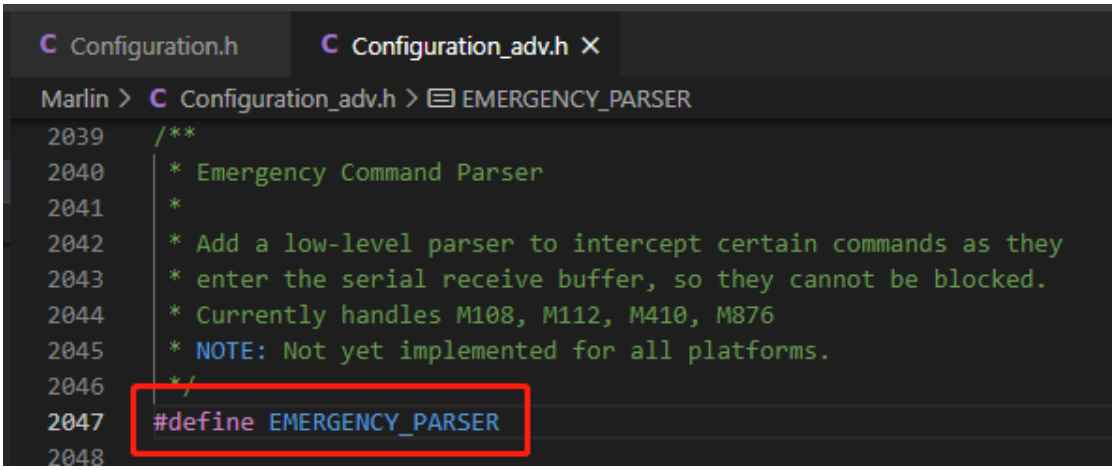


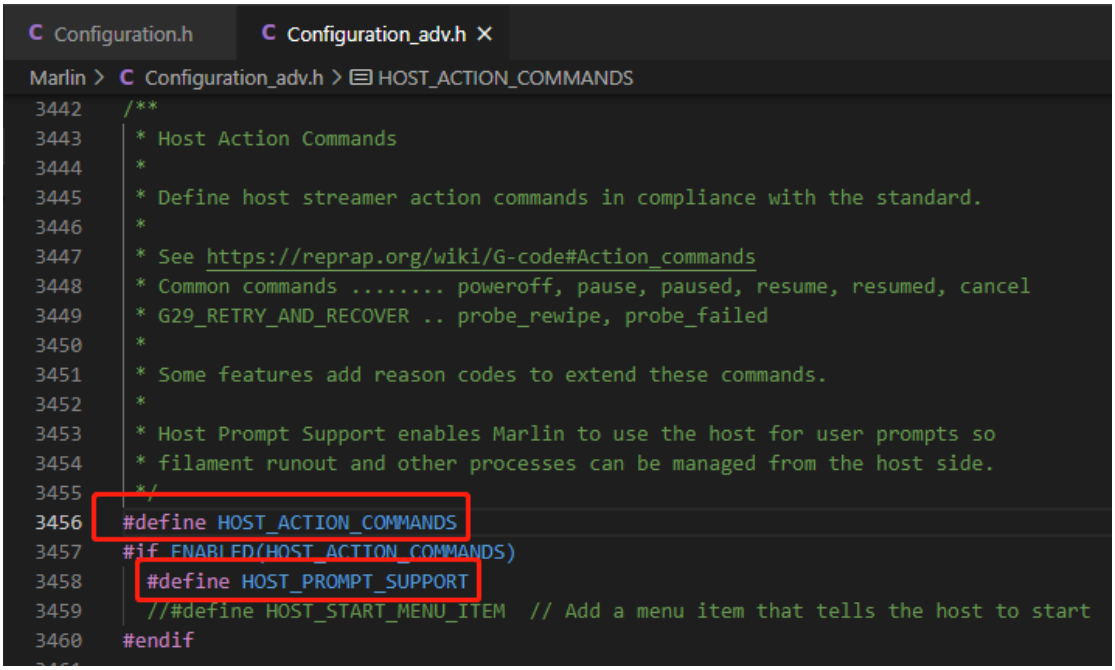
If you would like to update the firmware on the ESP module, you can download the latest ESP3D firmware at https://github.com/luc-github/ESP3D and compile your binary file. Rename it “ESP3D.bin”, copy it to the root directory in the microSD card, and then reset the motherboard. The bootloader on the motherboard will automatically update ESP3D.bin to the ESP8266, and the file will be renamed "ESP3D.CUR" after the update has completed.

Notes:

If you use the BTT TFT serial screen and connect any of the above-mentioned expansion modules to the motherboard, you need to enable the following within Marlin:

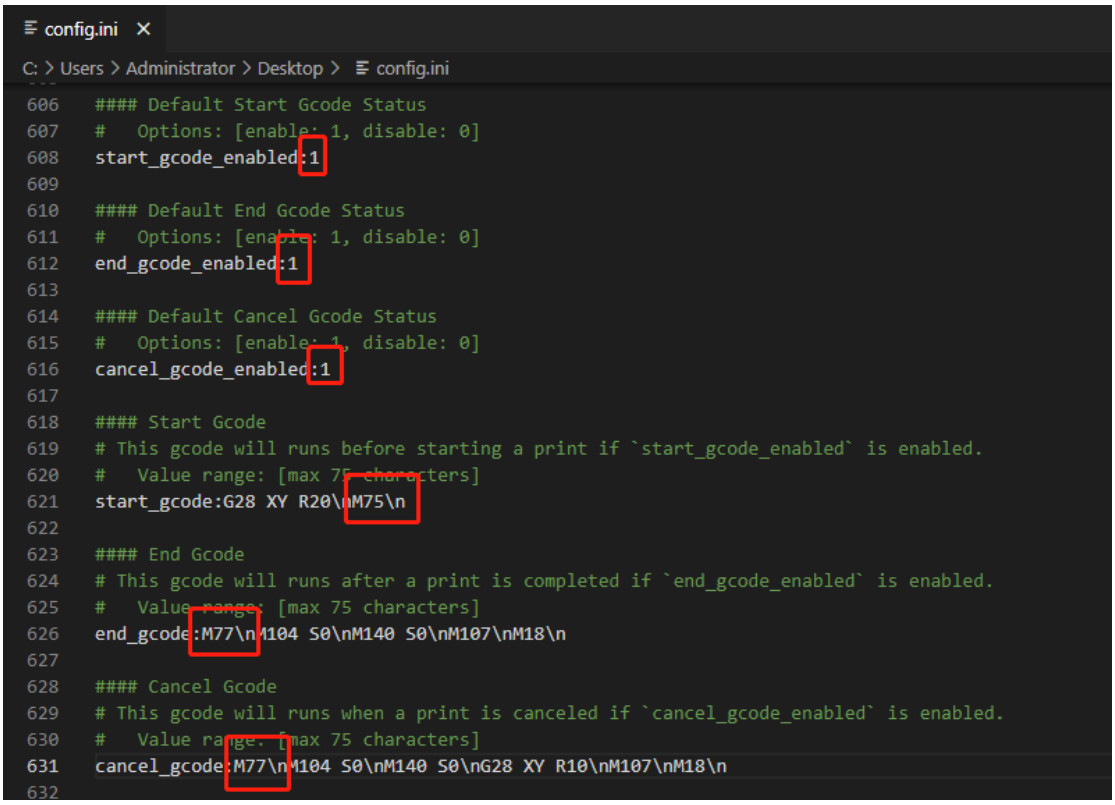
#define EMERGENCY_PARSER
#define HOST_ACTION_COMMANDS



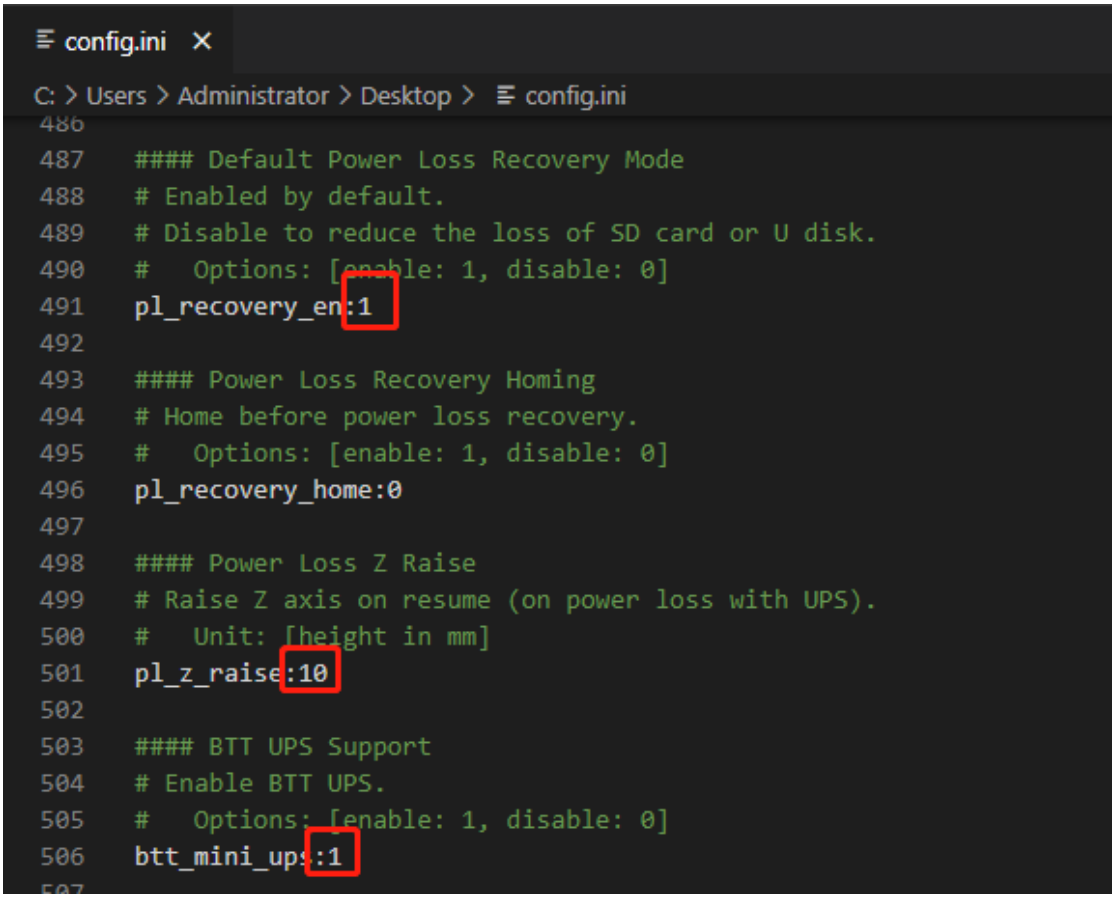


Additionally, you will need to set the following parameters in the config.ini file of the TFT:

start_gcode_enabled:1
end_gcode_enabled:1
cancel_gcode_enabled:1



Other settings are also required on the TFT to make power loss recovery module compatible with the touch screen.



pl_z_raise should be as high as #define POWER_LOSS_ZRAISE set in Marlin

If connecting smart filament sensor to the touch screen, you should enable “#define M114_DETAIL ”and disable #define LIN_ADVANCE”. LIN_ADVANCE in Marlin. At the time of writing, linear advance causes an error in the reporting of the extruded distance to the TFT.