MRR ESPE

Version 0.5

Getting Started

Document version: 1.1

Document date: January 14, 2020

Contents

Features.... 4
  Specifications (ESP32 microcontroller).... 4
Board Layout.... 5
Connectors.... 6
  Power.... 6
  Heating elements.... 6
  Fans.... 7
  Thermistors.... 7
  Endstops.... 7
  Stepper drivers.... 8
  Motors.... 8
  Others.... 8
Jumpers.... 10
  USB power.... 10
  Bed MOSFET signal voltage.... 10
  Stepper motor drivers.... 10
Connecting the board.... 13
Setting Up the Firmware.... 16
  Flashing firmware.... 16
  Setting up WiFi (via Terminal).... 17
  Setting up WiFi (via WiFi).... 18
  Uploading the web UI.... 18
  Adding macros.... 19
Additional Notes.... 20
  Wi-Fi performance.... 20
  Connecting Trinamic stepper motor drivers (SPI mode).... 20
  Using 12V for heated bed.... 21
  Using GPIO0 (IO0).... 21
Troubleshooting.... 22
Warnings.... 23
Disclaimer.... 23
License.... 23

Additional Information....24
Common Marlin settings....24
Pins....26

Features

The MRR ESPE is a 32-bit 3D printer control board using an ESP32 microcontroller. Besides interpreting G-codes for 3D printing, the ESP32 microcontroller acts as a web server that provides a web interface for controlling the printer. The board comes with the following features:

- Able to use up to 5 stepper drivers—X, Y, Z, E0, and E1—and also allowing up to 2 motors to be connected in parallel for Z axis, with support to easily configure use of Trinamic SPI stepper drivers using jumpers
- 12V to 24V input power supply
- Allows separate power supply to be used for the heated bed through use of an optocoupler
- X, Y, and Z min endstops
- Allows the use of a Z-axis probe, such as an inductive sensor, running on the input supply voltage (12V to 24V)
- Firmware controllable fans (part cooling fan; heat sink cooling fan; case fan; and one extra fan)
- AUX1 connector for use with an external host, such as the closed-source MKS TFT32
- LCD connector for use with an LCD controller such as that found on Creality 3D printers
- Web interface over 802.11 b/g/n Wi-Fi (implemented via firmware)
- I2S stepper stream with 10 additional I2S output pins

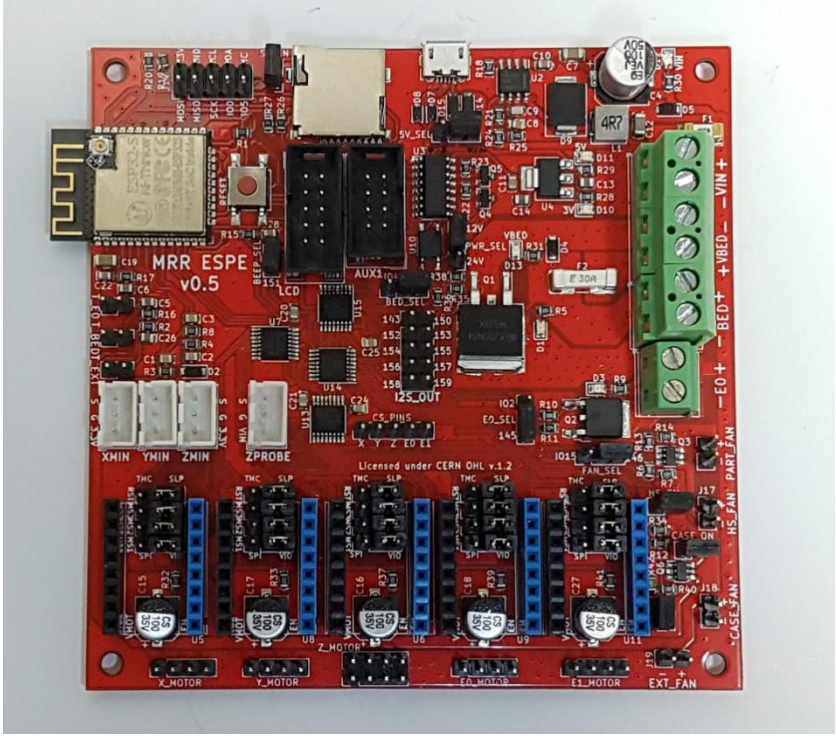
Specifications (ESP32 microcontroller)

- Tensilica Xtensa 32-bit LX6 microprocessor with dual cores
- Clock frequency up to 240 MHz
- 802.11 b/g/n Wi-Fi supporting WFA, WPA/WPA2 and WAPI
- Bluetooth v4.2 BR/EDR and Bluetooth Low Energy (BLE)
- ROM: 448 KB
- SRAM: 520 KB
- Flash memory: 4 MB
- Peripheral input/output: ADC, DAC, I²C, UART, CAN 2.0, SPI, I²S, RMII, PWM, and more.

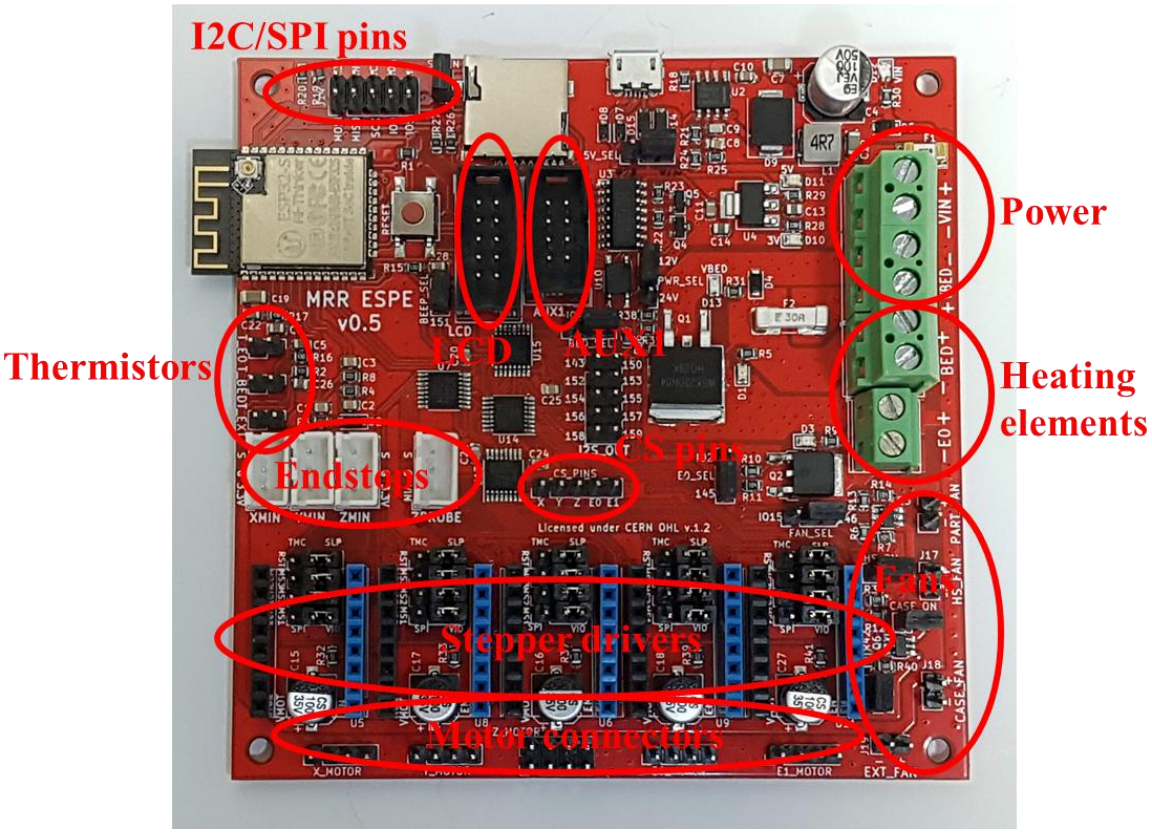
Board Layout

Physical dimensions

- 102 mm by 102 mm
- M3 mounting holes at each corner. Center of mounting hole is 4 mm from the edges of the board.
- Board thickness: 1.6 mm



Connectors



Power

VIN:

- Main power supply for the MRR ESPE.
- Accepts supply voltage from 12V to 24V.
- Used to power all devices connected to the MRR ESPE, except the heated bed. This includes the hot end heating block, motors, and fans.
- Maximum current allowed is 10A.

VBED:

- Power supply for the heated bed.
- Accepts supply voltage from 12V to 24V.
- Maximum current allowed is 15A.

Heating elements

BED:

- For connecting to a heated bed.
- Maximum current allowed is 15A.

E0:

- For connecting to the hot end heating block.
- Maximum current allowed is 5A.

Fans

PART_FAN:

- Connector for part cooling fan.
- Use a fan rated for a voltage that matches the voltage of VIN.
- Can be controlled by firmware.

HS_FAN:

- Connector for hot end's heatsink cooling fan.
- Use a fan rated for a voltage that matches the voltage of VIN.
- Can be controlled via firmware; use the HS_ON jumper to set it always on.

CASE_FAN:

- Connector for case fan.
- Use a fan rated for a voltage that matches the voltage of VIN.
- Can be controlled via firmware; use the CASE_ON jumper to set it always on.

EXT_FAN:

- Connector for extra fan.
- Use a fan rated for a voltage that matches the voltage of VIN.
- Can be controlled via firmware; use the EXT_ON jumper to set it always on.

Thermistors

T_BED:

- Thermistor for heated bed.

T_E0:

- Thermistor for hot end.

T_E1:

- Extra thermistor; can be used for enclosure thermistor.

Endstops

X_MIN:

- X minimum endstop.
- Externally pulled high to 3.3V.

Y_MIN:

- Y minimum endstop.
- Externally pulled high to 3.3V.

Z_MIN:

- Z minimum endstop.
- Externally pulled high to 3.3V.

ZPROBE:

- For connecting to a capacitive or inductive probe instead of a mechanical switch for the Z minimum endstop.
- The probe will be supplied with the same voltage as VIN. Please make sure that the connected probe is rated for this voltage.

Stepper drivers

- There are five sets of connectors for five stepper drivers.
- Note the orientation of the stepper driver. The EN pin on the stepper driver should be in the row **toward** the fan connectors.

Motors

X_MOTOR:

- For connecting to the X-axis stepper motor.

Y_MOTOR:

- For connecting to the Y-axis stepper motor.

Z_MOTOR:

- For connecting to the Z-axis stepper motor.

E0_MOTOR:

- For connecting to the extruder stepper motor.

E1_MOTOR:

- For connecting to an addition stepper motor, which can be a second extruder, or a second stepper motor for the Z axis.

Others

AUX1:

- For connecting to an external host controller like the MKS TFT32 using UART.
- Supply voltage on this connector is 5V.

LCD:

- For connecting to a LCD controller like those used on Creality 3D printers.
- Supply voltage on this connector is 5V.

I2C/SPI:

- Breakout headers for I2C and SPI pins.
- GPIO 0 (IO0) is also available for troubleshooting purposes. Connect GPIO 0 to ground, then boot/reset the board to manually enter flash mode.

USB:

- For connecting to an external host computer.
- When connected to a host computer, a serial terminal can be used to send gcodes to the MRR ESPE.

MicroSD:

- For use with a microSD card.
- The microcontroller can work with partitions up to 4 GB in size.
- There is no automatic detection of microSD card. Each time a card is inserted, the MRR ESPE needs to be informed via the gcode M21.

Jumpers

USB power

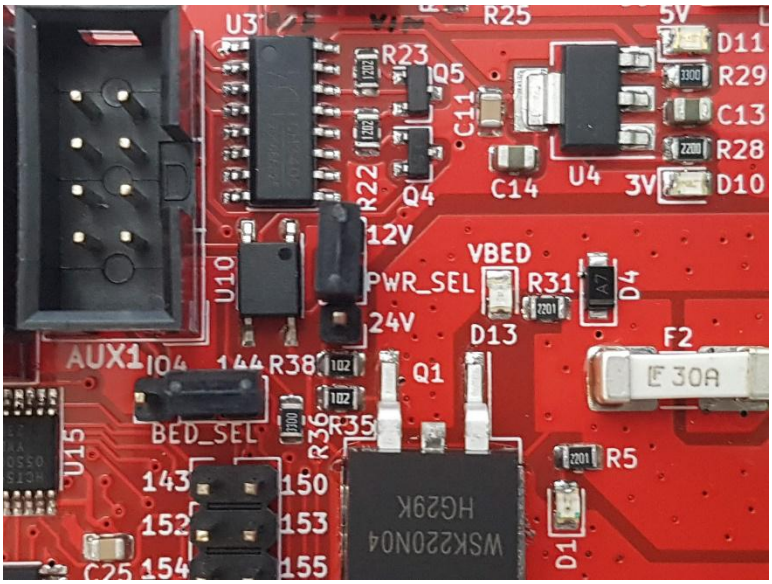
5V_SEL:

- Allows the board to be powered via USB or VIN.
- This jumper should usually be connected to VIN.
- Connect this jumper to VUSB only when flashing the firmware, or when you need to power the board via USB. Note: Powering the board via USB while heating the heated bed can cause unstable operation.

Bed MOSFET signal voltage

PWR_SEL:

- Allows the bed MOSFET's gate to be powered by a higher voltage so as to reduce $R_{DS(on)}$.
- When powered using 12V to 18V for VBED, set the jumper to connect the middle pin with the pin labeled "12V".
- When powered by a voltage above 18V for VBED, set the jumper to connect the middle pin with the pin labeled "24V".



Stepper motor drivers

MS1:

- Connect to VIO to set MS1 to high for microstepping; connect to SPI side to enable SPI mode for Trinamic drivers.

MS2:

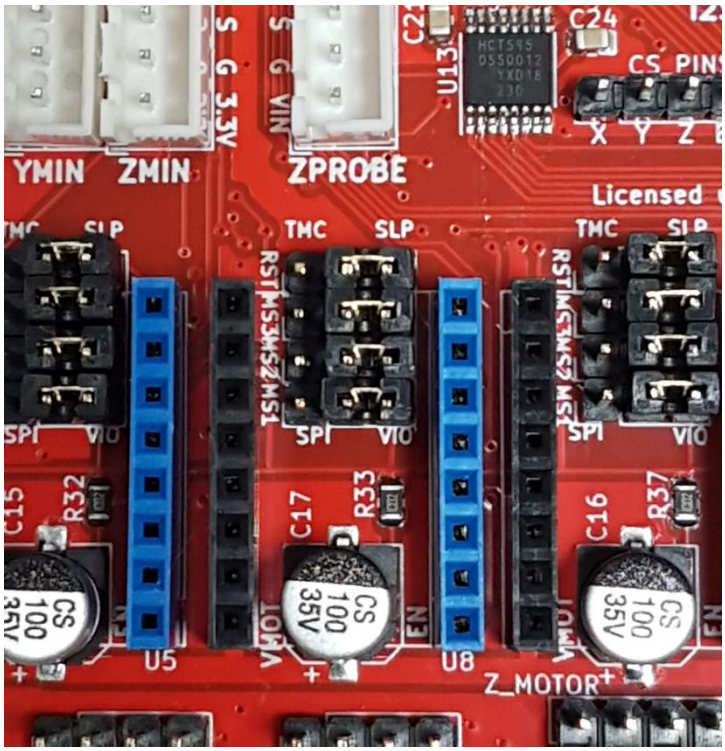
- Connect to VIO to set MS2 to high for microstepping; connect to SPI side to enable SPI mode for Trinamic drivers.

MS3:

- Connect to VIO to set MS3 to high for microstepping; connect to SPI side to enable SPI mode for Trinamic drivers.

RST:

- Connect to SLP if using A4988 or DRV8825 drivers, or if using Trinamic drivers in standalone mode; connect to SPI side to enable SPI mode for Trinamic drivers.



Fans

HS_ON:

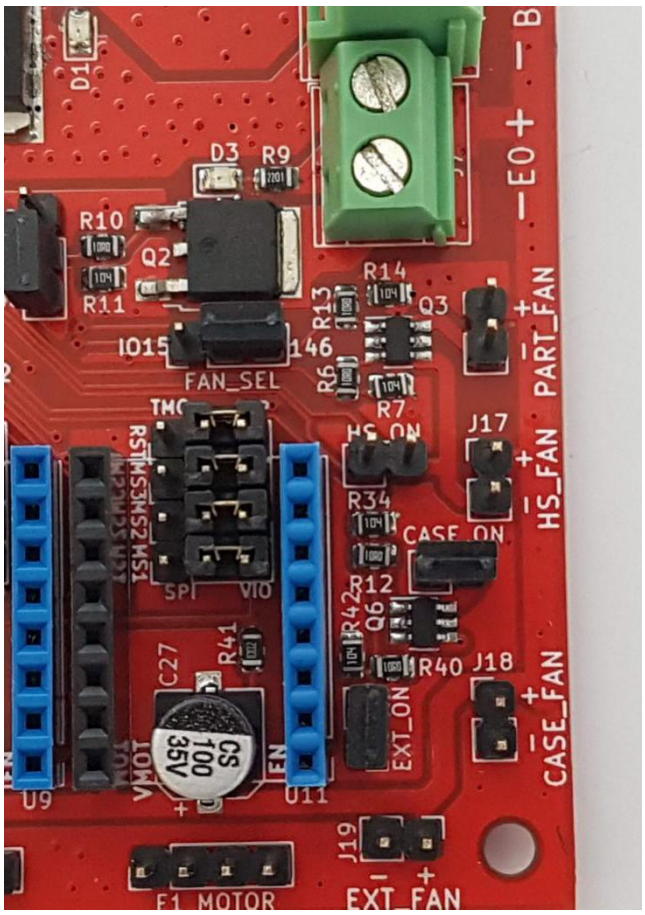
- Connect (close) to keep the heat sink fan always on.
- Leave open to allow the firmware to control the heat sink fan.

CASE_ON:

- Connect (close) to keep the case fan always on.
- Leave open to allow the firmware to control the case fan.

EXT_ON:

- Connect (close) to keep the extra fan always on.
- Leave open to allow the firmware to control the extra fan.



PWM signal

BED_SEL:

- Connect BED_SEL pin to either IO2 (native ESP32 pin) or 145 (I2S pin).

E0_SEL:

- Connect E0_SEL pin to either IO5 (native ESP32 pin) or 144 (I2S pin).

FAN_SEL:

- Connect FAN_SEL pin to either IO15 (native ESP32 pin) or 146 (I2S pin).

Connecting the board

Endstops:

- Connect the cables for the X, Y, and Z minimum endstops to the board's XMIN, YMIN, and ZMIN respectively.
- Note the polarity: **V** for 3.3V, **G** for ground, and **S** for signal.

Probe:

- If using a Z probe, such as an inductive probe, connect the probe's cable to ZPROBE instead of ZMIN.
- Note the polarity: **V** for 12V to 24V (depends on VIN voltage), **G** for ground, and **S** for signal. **Connecting the probe wrongly may damage the probe.**

Thermistors:

- Connect the hotend thermistor to T_E0.
- If using a heated bed, connect the heated bed thermistor to T_BED.
- An additional thermistor (such as for enclosure temperature) can be connected to T_E1.

Stepper drivers and motors:

- Set the jumpers for each stepper driver to the required microstepping (if using A4988, DRV8825, or Trinamic steppers in standalone mode) or to the SPI side (if using Trinamic steppers in SPI mode).
- Place the stepper drivers for each motor (X, Y, Z, E0, and E1) into the set of connectors corresponding to their motor connectors.
- Note the orientation of the stepper driver. The EN pin on the stepper driver should be in the row **toward** the fan connectors. **Placing the stepper driver in the wrong orientation can damage the stepper driver.**
- Connect the cables from the X, Y, Z, and extruder motors to X_MOTOR, Y_MOTOR, Z_MOTOR, and E0_MOTOR on the board respectively. An additional motor can be connected to E1_MOTOR if used.
- For Z axis, 2 motors can be connected in parallel.

Fans:

- Connect the case fan (for cooling the control board) to CASE_FAN. Leave CASE_ON jumper open to control this fan via firmware; close (short) CASE_ON

jumper to make this fan always on.

- Connect the hotend heatsink cooling fan (for cooling the heat sink on the hot end) to HS_FAN. Leave HS_ON jumper open to control this fan via firmware; close (short) HS_ON jumper to make this fan always on.
- Connect the part cooling fan (controlled by gcode) to PART_FAN. Set the FAN_SEL jumper to either IO15 (use native pins) or 146 (use I2S).
- An extra fan can be connected to EXT_FAN. Leave EXT_ON jumper open to control this fan via firmware; close (short) EXT_ON jumper to make this fan always on.
- Note the polarity; usually, the red or yellow wire is positive (+), and the black or blue wire is negative (-). **Reversing polarity may damage the fan.**

Hotend:

- Connect the cable for the hotend's heating cartridge to E0.
- Polarity matters if you are connecting this to an external MOSFET.
- Set the E0_SEL jumper to either IO2 (use native pins) or 145 (use I2S).

Heated bed:

- If using a heated bed, connect the cable for the heated bed to BED.
- Polarity matters if you are connecting this to an external MOSFET.
- Set the BED_SEL jumper to either IO4 (use native pins) or 144 (use I2S).

Main power supply:

- Connect the board's power supply (12V to 24V) to VIN.
- Note the polarity; the positive wire (usually red) should go to +, while the ground wire (usually black) goes to -. **Reversing polarity can damage the board and render it unusable. It may also damage the hotend MOSFET, causing the hotend to heat up uncontrollably when power is turned on. This may become a potential fire hazard.**

Heated bed power supply:

- Connect the power supply for the heated bed (12V to 24V) to VBED. A separate power supply from VIN can be used as the heated bed is controlled via an optocoupler.
- Note the polarity; the positive wire (usually red) should go to +, while the ground wire (usually black) goes to -. **Reversing polarity can damage the board and cause**

the bed to heat up uncontrollably when power is turned on. This may become a potential fire hazard.

- Set the PWR_SEL jumper according to the voltage being used for VBED: when using 12V to 18V for VBED, set the jumper to connect the middle pin with the pin labeled “12V”; when using a voltage above 18V for VBED, set the jumper to connect the middle pin with the pin labeled “24V”.

LCD controller:

- A LCD controller, such as those used on Creality 3D printers, can be connected to LCD.
- Close (short) BEEP_SEL to enable the beeper on the LCD controller; leave BEEP_SEL open if you do not want the beeper to sound.

MicroSD card:

- Insert a microSD card into the microSD card slot.
- Close (short) SD_EN jumper to use the microSD card on the board. Leave SD_EN jumper open to use an external SD card reader; the CS_SEL of this external card reader needs to be connected to IO5 on the breakout pins.

External host:

- An external host controller, such as the MKS TFT32, can be connected to AUX1.

Setting Up the Firmware

The board is sold without the firmware being flashed. Therefore, when using it for the first time, the firmware must be configured and flashed.

Flashing firmware

- Download a copy of the firmware, such as from https://github.com/maplerainresearch/Marlin or the main Marlin repository at https://github.com/MarlinFirmware/Marlin.If this is downloaded as a ZIP file, it will need to be uncompressed to a directory on your local hard drive.
- Connect the board via USB to your computer. Remember to set the 5V_SEL jumper to VUSB so that the board gets powered through USB. If your computer does not detect the board, you may need to install drivers to support CH340 on your operating system.
    - Windows: http://www.wch.cn/download/CH341SER_EXE.html
    - Mac: http://www.wch.cn/downloads/CH341SER_MAC_ZIP.html
    - Linux (should already come with the OS): http://www.wch.cn/downloads/CH341SER_LINUX_ZIP.html
- If using Marlin, refer to the instructions at http://marlinfw.org/docs/basics/install.html for the basics on configuring, building, and uploading Marlin firmware. Here, we will use Arduino IDE to build and upload the firmware. (Note: It is highly recommended to compile Marlin firmware using PlatformIO on VSCode. The specific instructions can be found here http://marlinfw.org/docs/basics/install_platformio_vscode.html.)
    - First, you need Arduino IDE installed on your computer (Windows, Mac, or Linux). Go to Arduino IDE's download page to download the version of the IDE for your computer. Arduino IDE's installation guide can be found here.
    - Once Arduino IDE has been installed, follow the instructions here to install the ESP32 core for Arduino IDE. It is recommended to install the stable release.
    - Launch Arduino IDE, and open the sketch for the firmware. This file is named `Marlin.ino` and can be found in the downloaded firmware's `Marlin` subdirectory.
    - In the IDE's menu, go to “Tools”, then “Board”, and choose “ESP32 Dev Module” which is the generic ESP32 module. Then, under “Tools”, “Port”, choose the COM port that your board is connected to. For upload speed, go to “Tools”, “Upload speed”, and set to “115200”.
    - Edit the `Configuration.h` and `Configuration_adv.h` files to define

the size of your printer, endstops, and other settings. See the section below for some basic settings which need to be changed.

- Click on the "Upload" icon of the IDE to compile the firmware and flash it onto the board. This will take quite a while.
- Once the firmware has been flashed onto the board, you can open the IDE's serial monitor (by pressing Ctrl+Shift+M, or from the menu via Tools, then Serial monitor) to make sure the firmware has been flashed properly. Enter M503 into the serial monitor and you should get a response from the board giving the current settings such as steps/mm and PID values.
- Via the serial monitor, you can configure your WiFi settings. See the section below on how to do so. You can also set up WiFi using a web browser (see relevant section below).

- When uploading firmware for the first time, the board needs to be connected via USB to the computer. Once over-the-air (OTA) update has been enabled, future uploads of firmware to the board can be carried out via WiFi.

Setting up WiFi (via Terminal)

Once the firmware has been flashed for the first time, there is a need to do an initial setup for the WiFi settings. In Arduino IDE, open up the serial monitor to connect to the board. You can also use any other terminal program to open the COM port that the board is connected to. Then, type in the following commands 1, 2, and 5 to set up the board in station mode (connect to an existing WiFi router). To set up in AP mode, use commands 3, 4, and 5.

1. `[ESP100]abcd`
   `[ESP101]12345`
   WiFi settings; the above will connect to a SSID of `abcd` with password `12345`. Change `abcd` and `12345`.

2. `[ESP103]STA`
   This will start up WiFi in station mode. The IP address of the webserver will be displayed in the terminal. Use this IP address to connect to the webserver.

3. `[ESP105]abcd`
   `[ESP106]12345`
   WiFi settings; the above will create an access point with SSID of `ESP32` and password `12345`. WPA2 security will be used by default.

4. `[ESP103]AP`
   This will start up WiFi in AP mode.

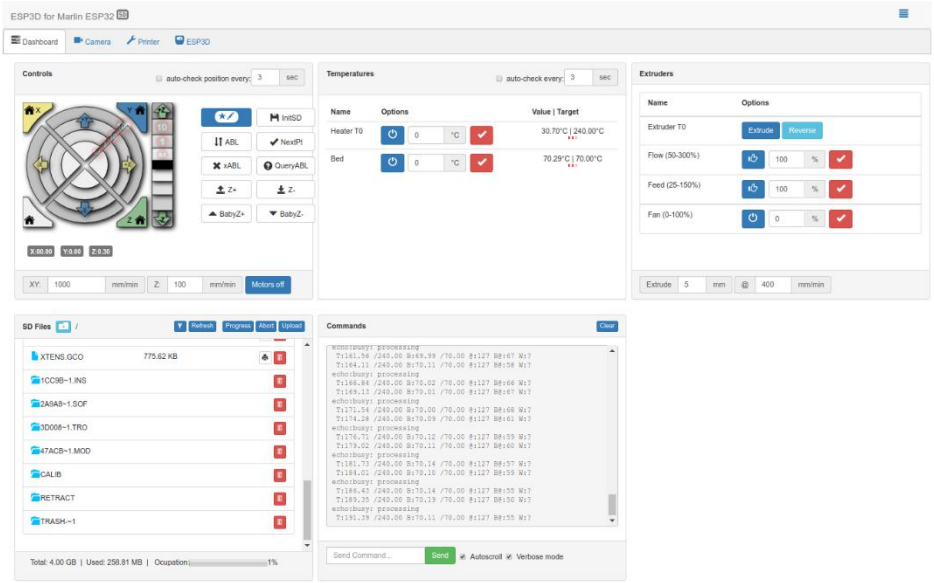
5. `[ESP110]ON`
   This will turn on WiFi.

Setting up WiFi (via WiFi)

When WiFi is not configured, the board will automatically become an access point (AP). The SSID of this AP is `MARLIN_ESP` and the default password is `12345678`. Connect your computer to this AP, then open a web browser and go to the address `192.168.0.1` to open up a web interface to configure WiFi settings. For login credentials, use the user name `admin` and the default password `admin` to change the settings.

Uploading the web UI

- When you first go to the web UI address (if you have mDNS, this should be http://marlinesp.local/ by default), you will be asked to upload the web UI files. Click on "Choose files", select `index.html.gz`, then upload. Then, refresh your browser page; the webUI should load. A copy of `index.html.gz` can be found in the firmware directory, under `<firmware directory>/Marlin/data/`.
- If you have the Arduino ESP32 filesystem uploader, you can also choose to upload the required files from Arduino IDE. After flashing the firmware, go to Tools, then choose ESP32 Sketch Data Upload.



Adding macros

The webUI (based on ESP3D) allows user-defined macros to be created. This allows "buttons" for custom commands (GCODE), such as buttons for babystepping, to be added to the webUI.

For example, to add a command to move to babystep Z by 0.01 downwards:

1. Create a text file with the gcode. In this case the gcode would be "`M290 Z-0.01`". Save it with a "`.g`" extension. In this case, it can be named `babyzdn.g`.
2. Upload that file to the ESP3D File System (not to the SD card); this is the SPIFFS file system, where `index.html.gz` is stored.
3. Click on the macro editor button in the controls panel. click the plus icon to add a macro. Give it a name, select a color, set the target as ESP and enter the path to the gcode file you uploaded.
4. Click Save.

Some ready-made macros can be found at:

https://github.com/maplerainresearch/MRR_ESPA/tree/master/firmware/macros

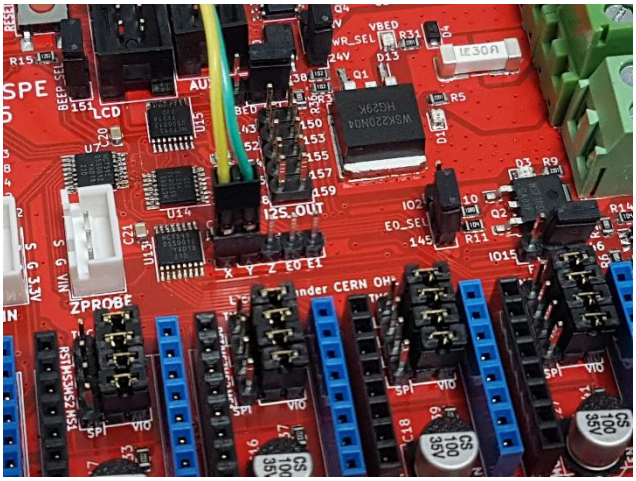
Additional Notes

Wi-Fi performance

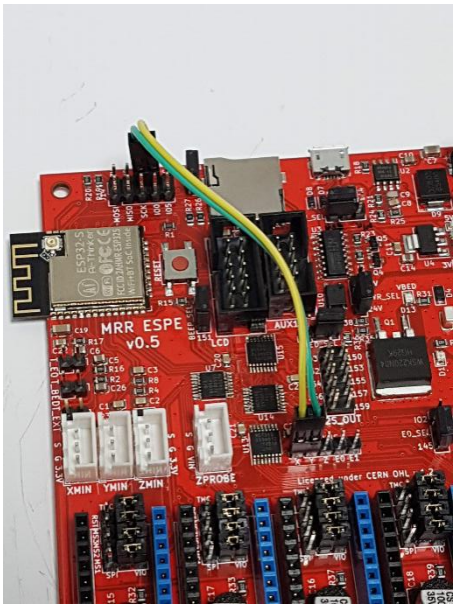
- Avoid placing the MRR ESPE within a metal case as it may adversely affect Wi-Fi performance. If necessary, expose the PCB antenna outside the case.

Connecting Trinamic stepper motor drivers (SPI mode)

- Jumpers are available to allow Trinamic stepper motor drivers using SPI mode to be easily configured. By setting the jumpers correctly, the only wiring needed for each driver is a wire for the CS pin.
- To configure a driver for SPI mode, connect the jumpers MS1, MS2, MS3, and RST for that respective driver to the TMC side.



- Use a jumper wire to connect the respective CS_PIN (either X, Y, Z, E0, or E1) to an unused pin (IO0, IO4, IO2, IO15, SDA, or SCL).



Using 12V for heated bed

- When using 12V for the heated bed, the current drawn by the heated bed can be in excess of 10A. This can cause the board to become very hot; **do not touch the board during operation**. (Warning: Parts of the board, such as the VBED fuse, can reach temperatures above 100 degrees Celsius when heating the heated bed. Also, due to the high currents involved, wires use for the heated bed, including wires being used to connect to the PSU, may get very hot.)
- It is highly recommended to have another fan to cool the underside of the board. This fan can be connected in parallel with the case fan. It is also recommended to place a small heat sink on the heated bed MOSFET (**Q1**).

Using GPIO0 (IO0)

On the ESP32, GPIO0 has a special function: it is pulled to GND when booting to enter flash mode. Therefore, it is not recommended to use this pin for functions which require it to be physically pulled to GND, as it will affect the booting up of the board.

On the other hand, if you are facing problems trying to flash the board via USB, you can connect GPIO0 to GND, then reset the board to enter flash mode.

Troubleshooting

A common cause of issues with the board is a loose connection. Check to make sure all cables are firmly connected. Reverse polarity can also damage the board, so check that all cables are connected with correct polarity.

- Issue: The endstops are not triggering.
  Check the endstops are connected correctly, especially the V, G, and S wires. Mechanical endstops usually only require two wires: ground (connect to G) and the signal line (connect to S).

- Issue: The motors do not move.
  Make sure the jumpers under the stepper drivers are connected correctly. For A4988 or DRV8825 drivers, the RST jumper must be connected to the SLP side. The other jumpers (MS1, MS2, MS3) can be connected to VIO to pull them high (to set microstepping), or left unconnected.

- Issue: The heated bed does not heat up.
  The heated bed is powered via VBED, so make sure there is a power supply connected to VBED.

- Issue: The motors keep stopping during the print.
  Make sure the stepper drivers are adequately cooled. A case fan should be used to cool the stepper drivers; this fan is connected to CASE_FAN.

- Issue: The motor is turning the wrong way.
  You can either reverse the cable connection on the board, or edit the firmware's configuration to reverse the direction of the motor.

- Issue: Files uploaded to the microSD card using the webUI are corrupted or partially truncated.
  Disable #define MONITOR_DRIVER_STATUS in Configuration_adv.h by commenting it out (adding double backslashes in front of it).

Warnings

- This product uses high currents which may cause fires. Use with caution.
- Do not use input voltages higher than 24V. Input current must not exceed 15A.
- Do not attempt to use more than 15A for the heated bed. While the MOSFET is capable of such currents, the connectors are not. If you need more than 15A, use an external MOSFET. In any case, it is highly recommended to use an external MOSFET for the heated bed to avoid drawing high currents through this product.
- Do not attempt to use more than 5A for the hot end. While the MOSFET is capable of such currents, the connectors are not. If you need more than 5A, use an external MOSFET.
- Do not reverse polarity.
- The breakout and endstop pins are rated for 3.3V, which is the voltage the ESP32 operates at. Attempting to feed inputs above 3.3V to these pins may damage the board.
- It is recommended to power the board only via USB (i.e. turn off PSU power supply) when flashing the board.

Disclaimer

This product is provided as-is. While care has been taken to design a safe product, it is the user's responsibility to make sure precautions are taken to prevent damage or injury when using this product. This includes adhering to all warnings provided. By using this product, the user accepts responsibility to bear all liabilities from any damage or injury arising from the use of this product.

License

Additional Information

Common Marlin settings

Below are some common settings in Marlin firmware which should be set to reflect your machine setup. This is not a comprehensive nor exhaustive list; the user is recommended to go through each setting in `Configuration.h` and `Configuration_adv.h` to make sure all settings are correct.

Note: Enabling a feature/setting means to uncomment it, i.e. make sure there is no `//` (double slashes) in front of the `#define` statement.

Configuration.h

#define MOTHERBOARD

This value defines which board is being used. It should be set to `#define MOTHERBOARD BOARD_MRR_ESPE` for the MRR ESPE.

#define BAUDRATE

This is the speed used for serial communications. It should be set to `#define BAUDRATE 115200` for the MRR ESPE.

#define TEMP_HYSTERESIS

#define TEMP_BED_HYSTERESIS

These values define what is considered “close enough” when trying to reach a certain temperature for the hotend or heated bed. Due to the ADC of the ESP32, they should be set to a value of `5`.

#define PIDTEMPBED

This allows PID bed heating, and should be enabled.

#define THERMAL_PROTECTION_HOTENDS

#define THERMAL_PROTECTION_BED

#define THERMAL_PROTECTION_CHAMBER

These settings enable safeguards against a broken or disconnected thermistor wire, and **must be enabled**.

#define X_DRIVER_TYPE

#define Y_DRIVER_TYPE
#define Z_DRIVER_TYPE
#define E0_DRIVER_TYPE
Change the values of these to reflect the stepper driver being used for each axis. The default value is `A4988`.

#define INVERT_X_DIR
#define INVERT_Y_DIR
#define INVERT_Z_DIR
#define INVERT_E0_DIR
Toggle these values between `true` and `false` to switch the direction of the stepper motors.

#define X_HOME_DIR
#define Y_HOME_DIR
#define Z_HOME_DIR
The values of these should be `-1` to use the minimum endstops.

#define X_BED_SIZE
#define Y_BED_SIZE
#define Z_MAX_POS
These values should be set to the maximum sizes for your X, Y, and Z axes respectively.

#define X_MIN_POS
#define Y_MIN_POS
These values reflect the position of the nozzle when the respective X and Y minimum endstops are triggered. Set to values of `0` if the endstops are triggered at the edges of the print area. Set to negative values if the endstops are triggered with the nozzle outside the print area.

#define Z_MIN_POS
This value reflects the position of the nozzle when the Z minimum endstop is triggered. Set to `0` if the endstop are triggered at same height as the bed, or if using an autolevel probe. Set to a negative values if the endstop is triggered with the nozzle below the bed.

#define Z_SAFE_HOMING
Enable this if you are using an autolevel probe and want to home the Z axis using the center

of the bed instead of at the edge (where the probe may fail to detect the bed).

#define SDSUPPORT
This must be enabled to allow use of the onboard microSD card.

#define SPI_SPEED SPI_HALF_SPEED
Enable this if you have “volume init” errors.

Configuration_adv.h
#define TX_BUFFER_SIZE
Set to `128` for optimal speed.

#define RX_BUFFER_SIZE
Set this to at least `1024`.

#define X_CS_PIN -1
#define Y_CS_PIN -1
#define Z_CS_PIN -1
#define E0_CS_PIN -1
#define E1_CS_PIN -1
If using Trinamic stepper drivers in SPI mode, uncomment the respective line for the axis, and change the `-1` to the pin being used for CS (such as `21` or `22`).

#define ESP3D_WIFISUPPORT
#define WEBSUPPORT
#define OTASUPPORT
#define WIFI_CUSTOM_COMMAND
Enable this to allow the MRR ESPA to communicate via WiFi using the ESP3D webUI, and allow over-the-air (OTA) updates.

Pins

Function	Marlin name	Pin
Endstops		
X min endstop	X_MIN_PIN	35

Y min endstop	Y_MIN_PIN	32
Z min endstop	Z_MIN_PIN	33
I2S stream		
I2S word select clock	I2S_WS	26
I2S bit clock	I2S_BCK	25
I2S data	I2S_DATA	27
Stepper drivers		
X step	X_STEP_PIN	129
X direction	X_DIR_PIN	130
X motor enable	X_ENABLE_PIN	128
Y step	Y_STEP_PIN	132
Y direction	Y_DIR_PIN	133
Y motor enable	Y_ENABLE_PIN	131
Z step	Z_STEP_PIN	135
Z direction	Z_DIR_PIN	136
Z motor enable	Z_ENABLE_PIN	134
E0 step	E0_STEP_PIN	138
E0 direction	E0_DIR_PIN	139
E0 motor enable	E0_ENABLE_PIN	137
E1 step	E1_STEP_PIN	141
E1 direction	E1_DIR_PIN	142
E1 motor enable	E1_ENABLE_PIN	140
Thermistors		
Hotend thermistor	TEMP_0_PIN	36
Heated bed thermistor	TEMP_BED_PIN	39
Extra thermistor	TEMP_1_PIN	34
Heating elements		
Hotend MOSFET gate	HEATER_0_PIN	2 or 145
Heated bed MOSFET gate	HEATER_BED_PIN	4 or 144
Fan		
Part cooling fan	FAN_PIN	15 or 146
Case fan	CONTROLLER_FAN_PIN	147
Heat sink fan	E0_AUTO_FAN_PIN	148
Extra fan	FAN1_PIN	149

MicroSD card		
SD card select	SDSS (or SS_PIN)	5
SPI pins		
MOSI	MOSI_PIN	23
MISO	MISO_PIN	19
SCK	SCK_PIN	18
LCD controller		
LCD_RS	LCD_PINS_RS	13
LCD_EN	LCD_PINS_ENABLE	17
LCD_D4	LCD_PINS_D4	16
Beeper	BEEPER_PIN	151
BTN_EN1	BTN_EN1	0
BTN_EN2	BTN_EN2	12
BTN_ENC	BTN_ENC	14
I2C pins		
SDA	SDA (not used)	21
SCL	SCL (not used)	22
Others		
I2S output 143	(not used)	143
I2S output 150	(not used)	150
I2S output 152	(not used)	152
I2S output 153	(not used)	153
I2S output 154	(not used)	154
I2S output 155	(not used)	155
I2S output 156	(not used)	156
I2S output 157	(not used)	157
I2S output 158	(not used)	158
I2S output 159	(not used)	159

Document Revision History

November 1, 2019

- Document created.

January 14, 2020

- Updated with new ESP3D commands and configuration requirements due to ESP3DLib integration into Marlin.
- Added note about issue with MONITOR_DRIVER_STATUS causing problems with SD file uploads.