By: Astral, 2021

A Foreword…

I've written this guide to provide a better explanation on this board swap. There's a lot of information out there that can confuse someone like me, and maybe like you, and those who might be doing this for the first time. So, to save you some of the time, headaches, and problem solving, I've gone ahead and compiled everything I know to get this board working with the X1 and stock TFT screen. Here we go…

Oh, I almost forgot, before we go any further I highly recommend you join the Artillery 3D Discord server. There you can find many Artillery users who are familiar with troubleshooting and 3D printing in general. It's a great community and you can learn a lot here. *Shout out to Sgt. Taz and Daali who were most helpful to me when I was trying to figure this stuff out.*

So... you want to upgrade your X1?

The SKR board series by Big Tree Tech is a popular option for swapping out your old Gen L board. In general, the SKR 1.4 / 1.4 Turbo boards are much more powerful with their 32 bit processor, offering a very nice improvement to the performance of your printer. Here are some specs across board versions along with the benefits the board can give you:

Comparison Of Functions

	BTT SKR V1.3	BTT SKR V1.4	BTT SKR V1.4 Pro
Frequency	LPC1768 -100MHz	LPC1768 -100MHz	LPC1769 -120MHz
Fan interfaces	One fan interface	1-way CNC fan, 3-way Frequent fans	1-way CNC fan, 3-way Frequent fans
I2C, SPI and WiFi interfaces to DIY	X	✓	✓
The closed-loop drive interface	X	✓	✓
Double z-axis interface	X	✓	✓
RGB light belt	X	✓	✓
Support multifarious screen	✓	✓	✓
High performance MOSFET tube	✓	✓	✓
External connection power module	X	✓	✓

Comparison Of Parameters

	SKR V1.3	SKR V1.4
Motor driver	Support:TMC5161,TMC5160, TMC2209,TMC2225,TMC2208, TMC2130,ST820,LV8729, DRV8825,A4988, can be independently external motor drive	Support:TMC5161,TMC5160, TMC2209,TMC2225,TMC2208, TMC2130,ST820,LV8729, DRV8825,A4988, can be independently external motor drive
WIFI module	/	ESP-01S
Driver working mode support	SPI,UART,STEP/DIR	SPI,UART,STEP/DIR
Display	"2.4/2.8 /3.5" TFT, (SUPPORTS DUAL-MODE SCREENS) LCD2004, LCD12864	"2.4/2.8 /3.5" TFT, (SUPPORTS DUAL-MODE SCREENS) LCD2004, LCD12864
Supporting file formats	G-CODE	G-CODE
Motor drive interface	X, Y, Z, E0, E1, five (each has a reextendable interface), Up to 256 subdivisions	X, Y, Z, E0, E1, five (each has a reextendable interface), Up to 256 subdivisions

Pros:

- Improved processing speed. 32 bit > 8 bit.
- Easier firmware flashing via USB. No need to disconnect the TFT screen.
- Quiet printer operation using TMC 2208/2209 drivers utilizing StealthChop 2.
- UART configuration for stepper drivers allowing control of stepper driver voltage through the firmware.
- Marlin 2.0.x, and the features it has over Marlin 1.
- Optional sensorless homing (no end-stops required).

Drawbacks:

- SD card slot on the board does not line up with the X1 housing.
- Only one PWM controllable fan input. The rest are constantly powered.

Some things to consider…

In order to install this board, you will be voiding your printer's warranty. In addition to that, you should have at least some experience with rewiring and computer related hardware. If you are completely brand new to something like this - do not be afraid to give this a try, but also consider that there are times where you might accidentally break something and your printer will be down for a while. It happens to even the best of us. Take your time, and be careful!

Prerequisites:

For this install we will be using the BTT TMC 2209 1.2 drivers, as they are the easiest to work with and pair best with the board in my opinion. The board can support many other drivers listed in the specs above, but those will require slightly different setups. If you absolutely do not want to use BTT TMC 2209s, then you can find a guide HERE to how you can set your board up for them. Just keep in mind that everything I do in this guide will be for the 2209, but you can retrofit this information to loosely fit your setup's needs.

1. **SKR 1.4 / 1.4 TURBO 32Bit Control Board**
2. **TMC 2209 Stepper Drivers**
3. **SKR Board Adapter**
4. **4x M2 screws.**
5. **A brain.**

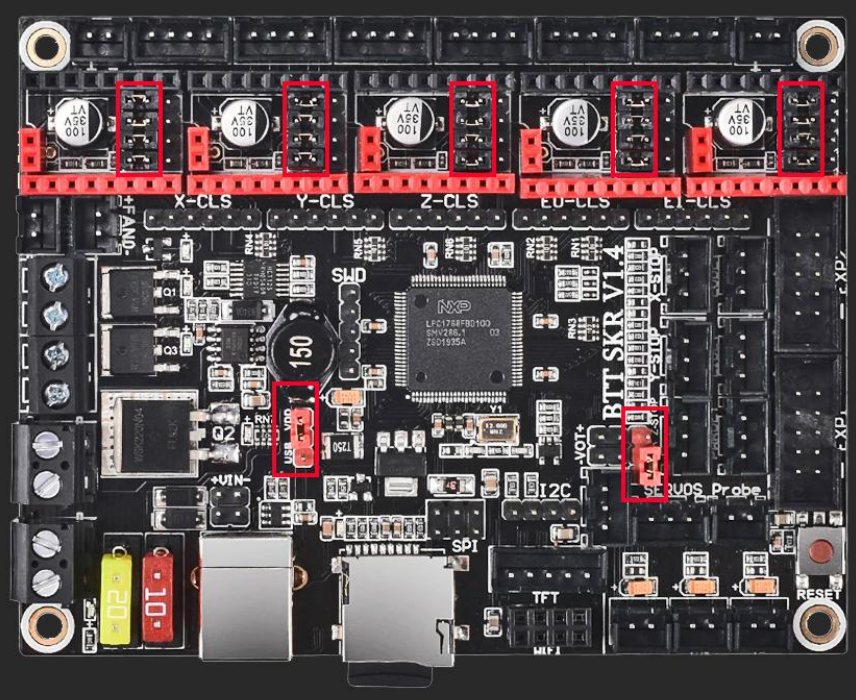
"What?! No BLTouch?"
How about no, buster. While the board offers support for it, this guide does not cover it. Google it. We're pros here. We know how to level a bed manually. Remember that episode of *Rick and Morty* where Rick made that sheet of concrete in the garage ***PERFECTLY*** flat? Yeah, that's us. I mean, what are you gonna do if your BLTouch dies? Uh, oh! Exactly.

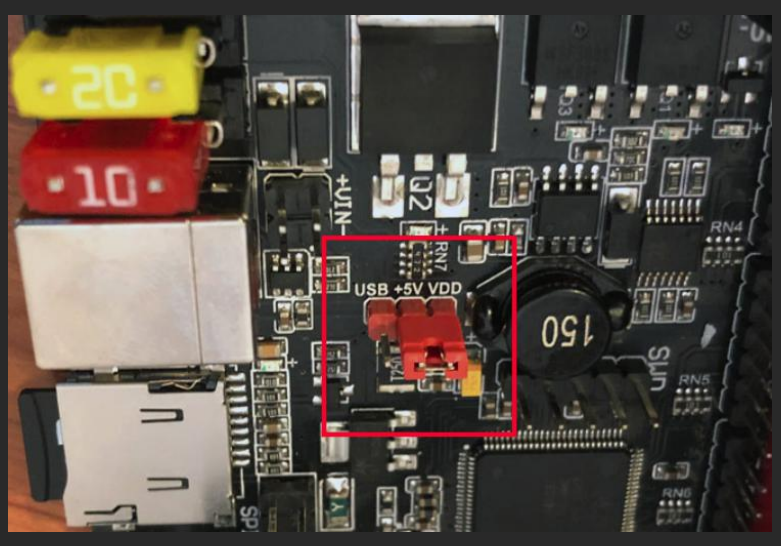
"Sensorless Homing…"
Also, keep in mind that in this guide we will not be utilizing Sensorless Homing. I personally haven't fully tested it, and prefer to use my existing end-stops. I only mention it in some steps because it is an option.

Now, let us begin.

Preparing your SKR 1.4 / 1.4 TURBO

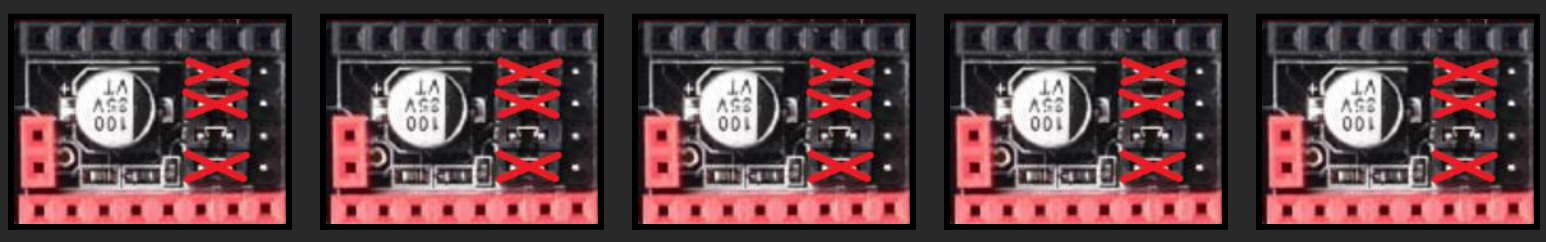
When your SKR board arrives, you'll find that it will come with numerous jumpers on it. If you don't know what those are, they're the tiny black and red plastic pieces with metal contacts that are helping connect pins together.





For the black jumpers, extract all of them ***except*** the ones in the third position down. So, for each motor, that is the top two and then the bottom to be removed. In addition, the red jumper towards the center of the board controls which source

we will choose to power the board. You have two choices of either USB or VDD (power supply). Choose VDD. Your board also should have come with an SD card (mine did). Go ahead and insert it into the board for now.
By keeping a jumper on each motor's 3rd position we're configuring the board so that we can have UART mode.
Again, here's what each group of jumpers you should remove across your 5 stepper driver jumper pins:



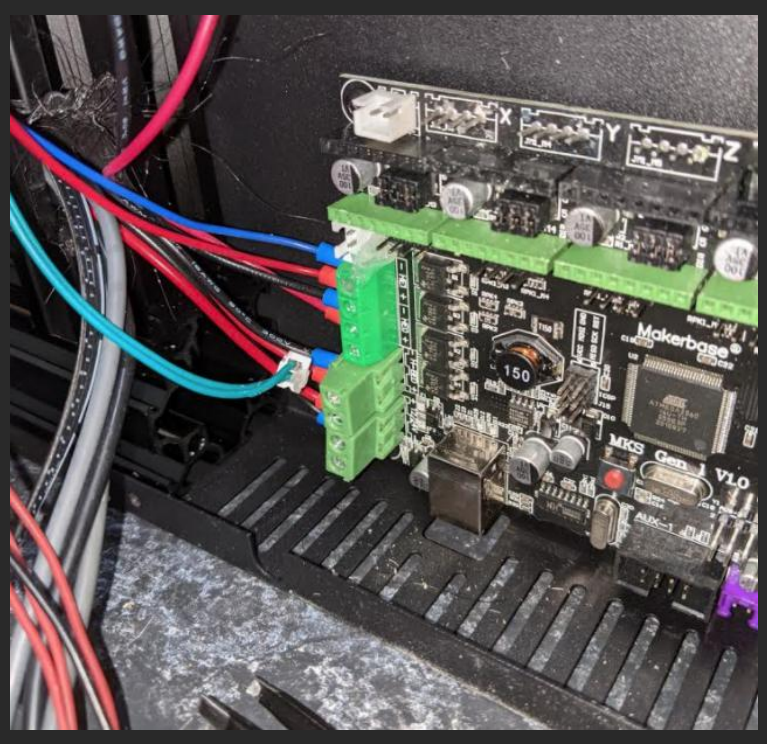
Now we can move on to the printer…

Preparing your printer.

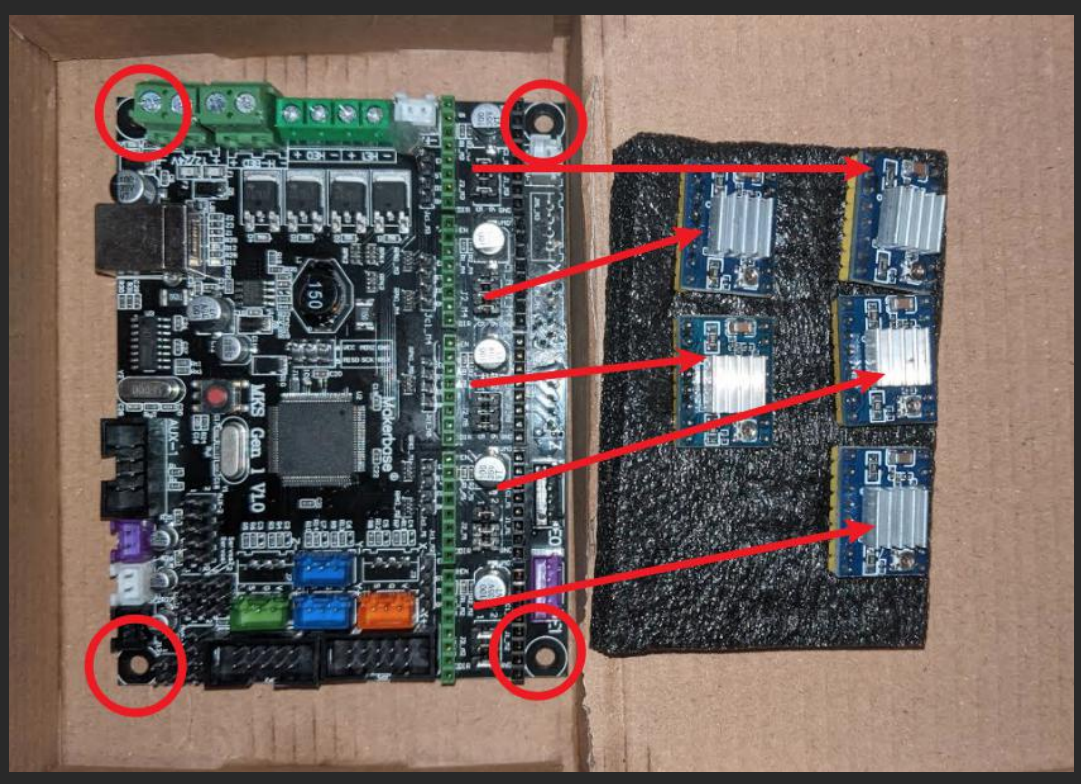
WARNING: This process will void your warranty. But if you're like me, you don't care. This is totally worth it. 😎

First disconnect the power cable from your X1 and carefully flip the printer on its side. Don't forget to disconnect any filament and spools previously being used. Then proceed to remove the bottom M2 screws holding the housing cover. Be careful removing the case cover, as the cooling fan will be attached to your printer's main board. Disconnect it.

With the cover off, we can begin retiring our MKS GEN L board. It was a fine board, but if you're like me, it might have crapped out on you because of all of the glue on it. If this is your first time looking at your board, don't be surprised seeing super glue EVERYWHERE. Why, you ask? Well Artillery has an employee we all like to call Glue Guy. He makes more money than anyone in the world, and his job is to be a jackass and cover your entire motherboard in glue. Since there's nothing we can do about this, we just have to press on.



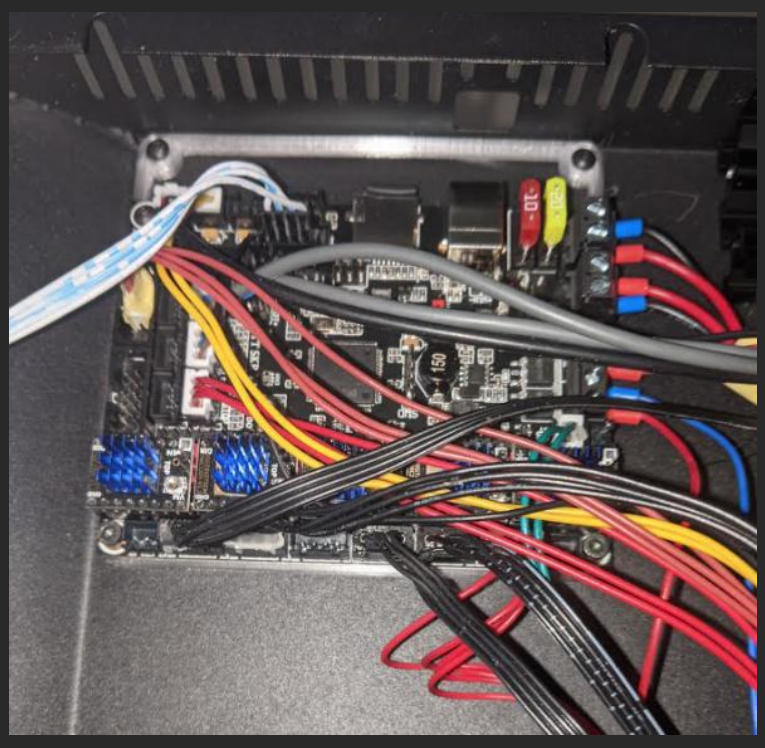
Be careful removing each wire accordingly. I used some painter's tape laying around and taped cable groups together in order to organize them, so as to not lose track of what's what. To remove any excessive glue, using tweezers and pliers will help a lot. Just don't yank on any wires or connectors. As you can see, my X Y Z plugs were glued so badly together that they came off of the board! I had to clip them off the wire connectors later.



You can then proceed to unscrew the four M2 screws holding the board to the printer. Keep them, we'll use them for our printed mount. You can also remove the stepper drivers. The TMC2100s are nice, and can fetch a couple bucks if you want to sell them. I kept mine to reuse in case of a rainy day.

Mounting the SKR

Next, we can install our SKR mount using the previous 4 screws. Do not over tighten these screws! Just get them a little snug. You'll want to position it in your case as such to allow some space between the side of the printer and the board. This will give us room to insert and remove the SD card from the board if we need to.

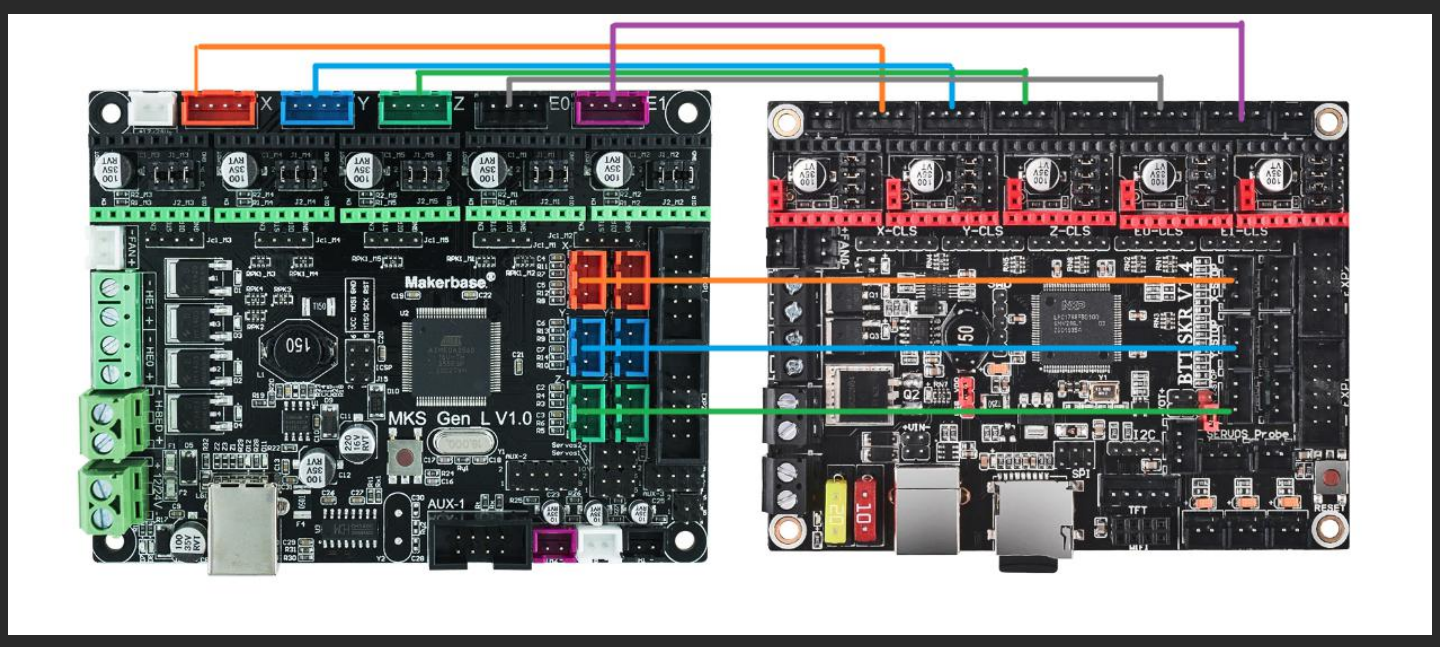


You'll need to use some small M2 screws for installing your board onto the mount. You can find these at most hardware stores or online. Again, do not over-tighten. Just get the board a little snug. That's it!

Hooking things back up...

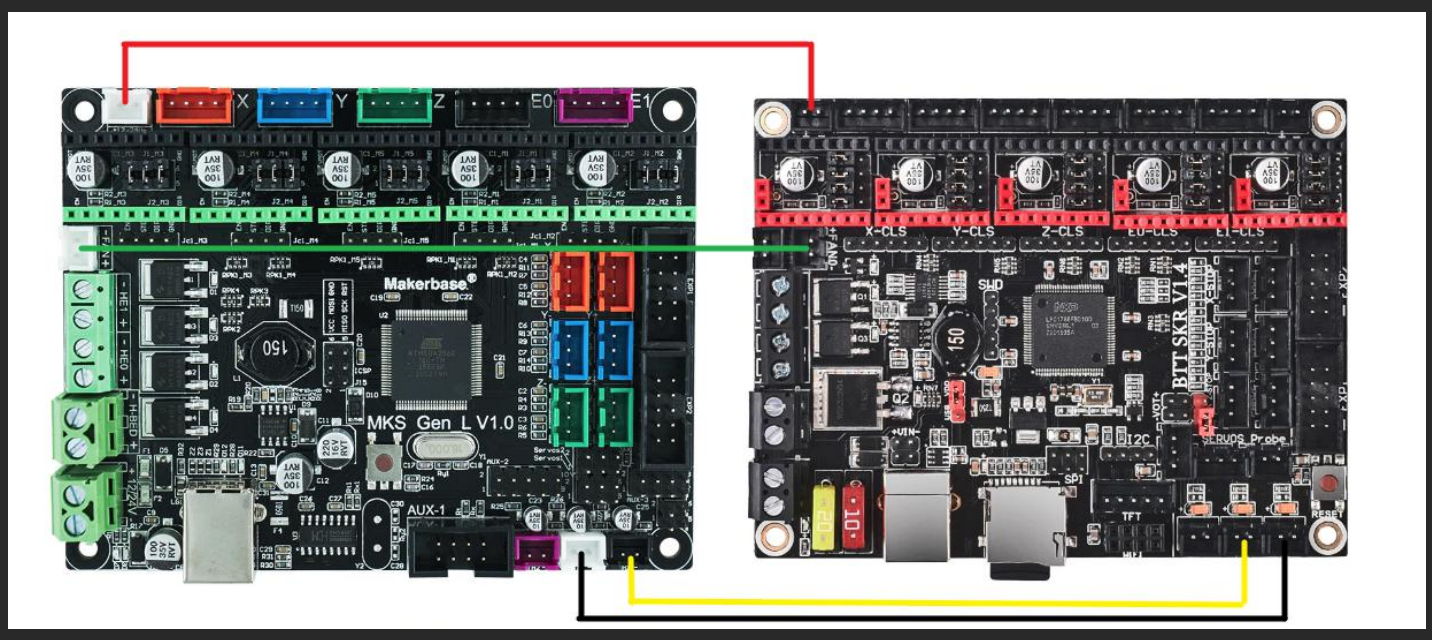
Now we can begin to reconnect our motors and hardware to the motherboard! Pay close attention to this part to avoid any power issues as the connections from the old board do not exactly translate to the new board.

Here's a diagram to where we connect our motors and end-stops.:

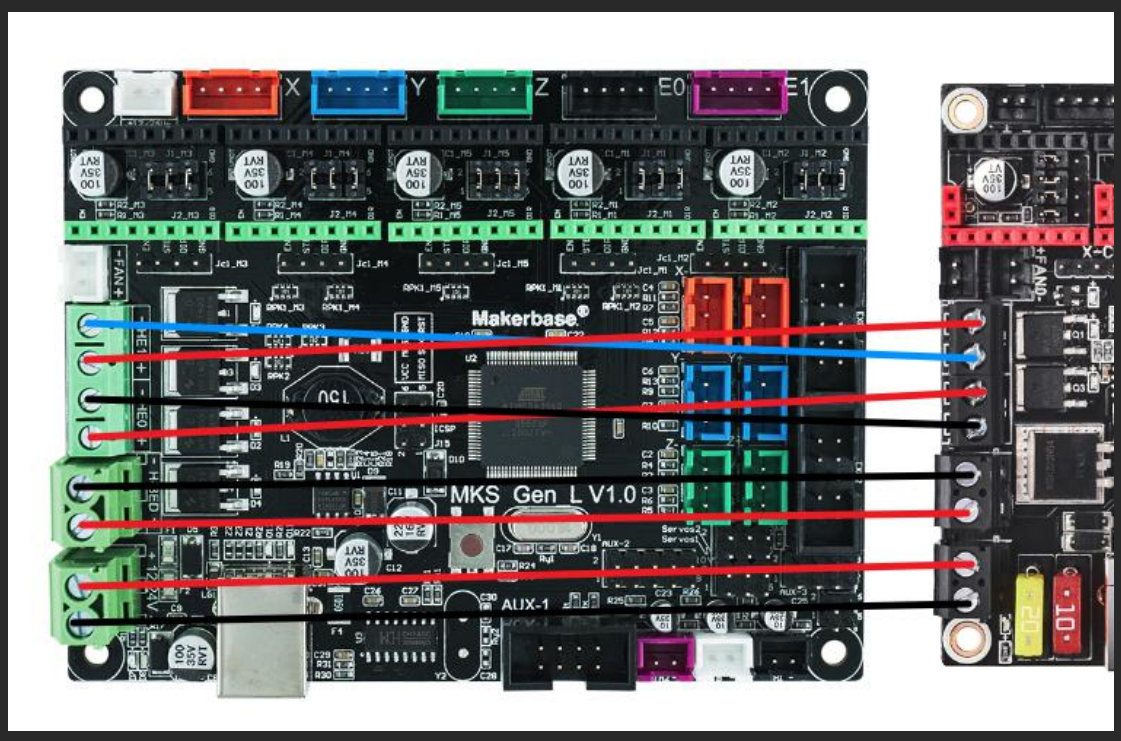


Note: If you want to use sensorless homing, you can remove the end-stop cables entirely. I've chosen to continue to use and keep my end-stops. In my opinion, they are more accurate than sensorless homing. To each their own.

Here's where your fans, hotend, and bed connectors go:



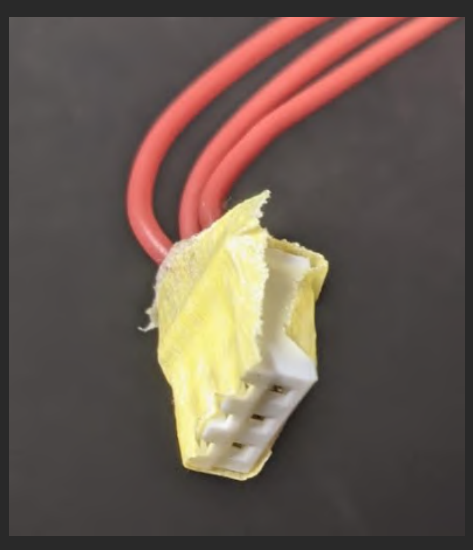
And finally, here's where your power cables go. Pay close attention to the order of positive and negative. You'll know if a ***wire*** is negative if it's black and positive when red.

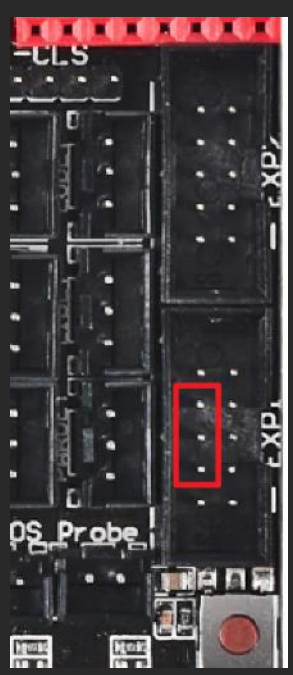


If you've messed this up, you'll end up short circuiting something. I accidentally did with my hot-end wires while figuring this out, but fortunately did not cause any damage. Whew!

Now for the RGB LED connectors…

I've not yet tested, but the X1's LEDs are not compatible with the existing pins on the SKR board. So, we'll just make use of some pins not being used. We'll designate what they're for later on in the firmware. To make this work, we'll wrap the dark red LED connector in tape to prevent the metal contacts on one side from touching other pins. It'll look like this connected to our EXP1 port:

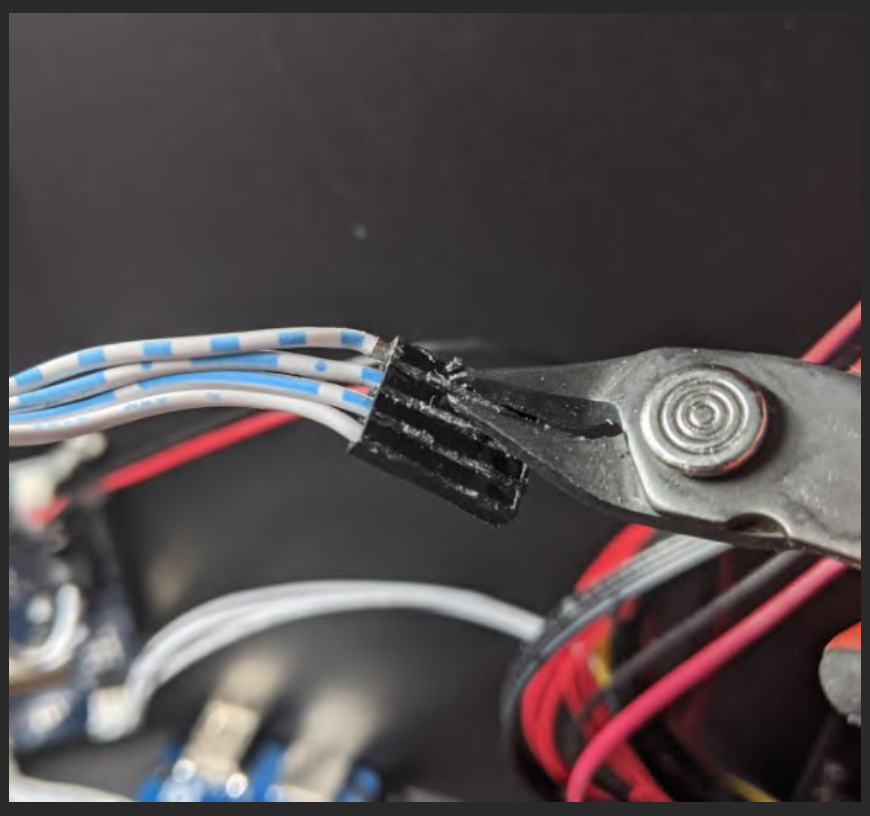




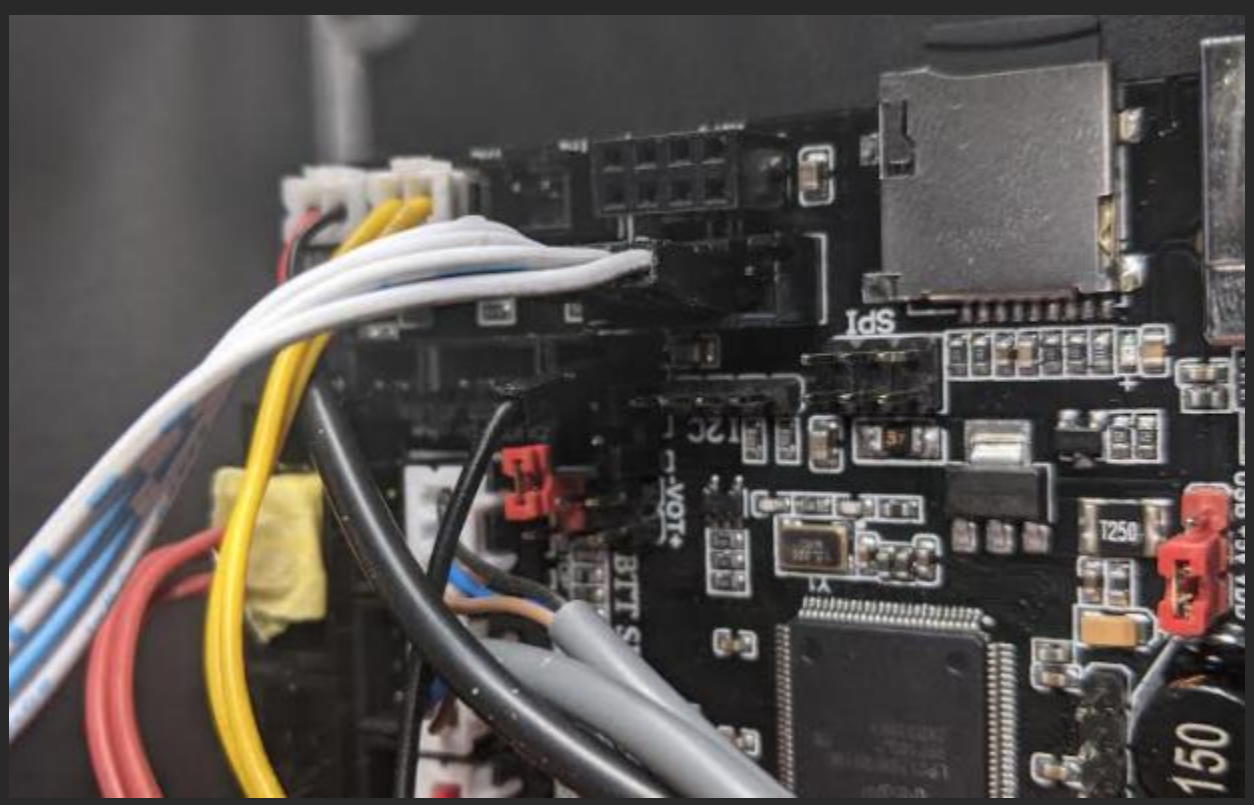
There is also a lone black pin connector which is the ground for our LED. We'll connect it to this third pin on the board's RGB connector.

Now for the TFT screen…

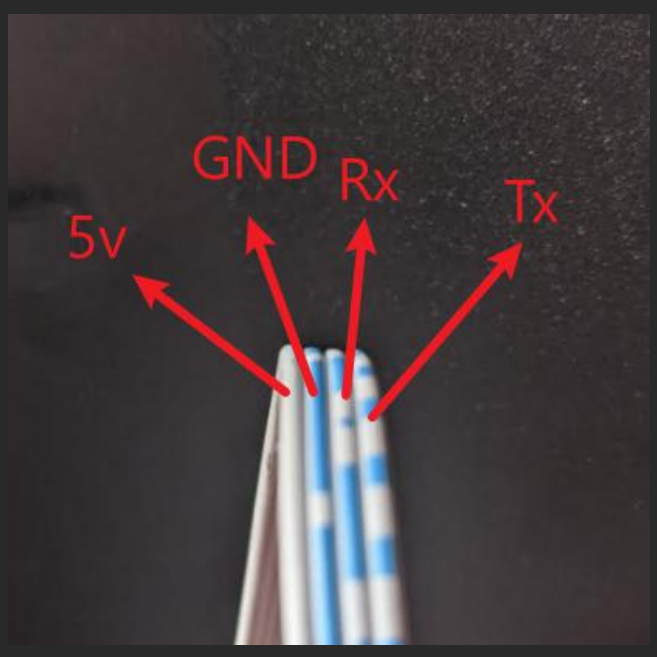
The stock screen connector by default does not fit into the SKR 1.4 board. But fear not, we can make an easy modification to the connector so it will. Grab some flush cutters (you should have these if you're 3D printing by the way) and clip off the excess row of empty female ports. It should look something like this, leaving only the used row of female connections.

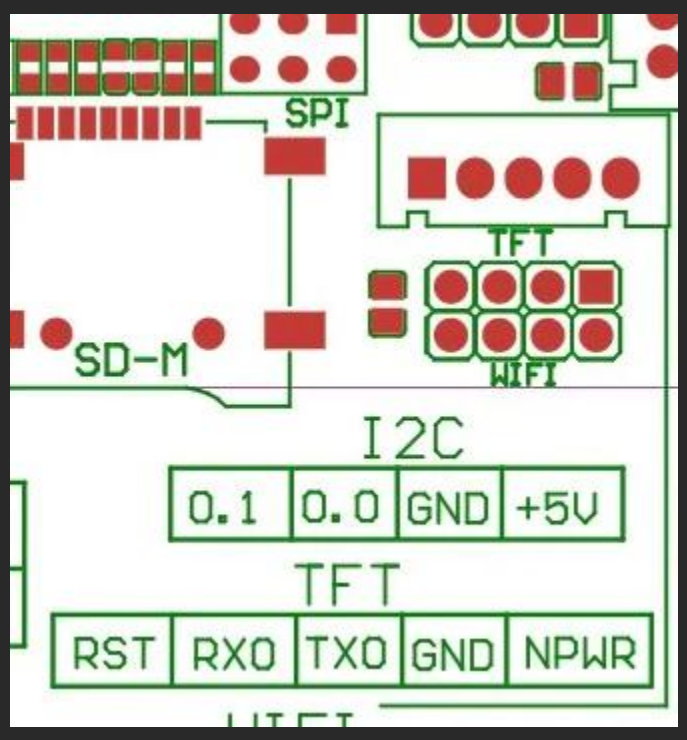


And with the connector now retrofitted, it will fit onto our board like this, leaving the single 5th RST pin exposed.



If for whatever reason your screen isn't detecting your printer, check whether the Rx and Tx wires go from the display to the board properly. An Rx should connect to a Tx pin, and vice versa. You can tell which wire is what by the pattern on the protector. Just rewire the connector if needed.

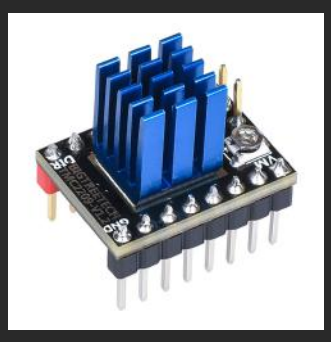




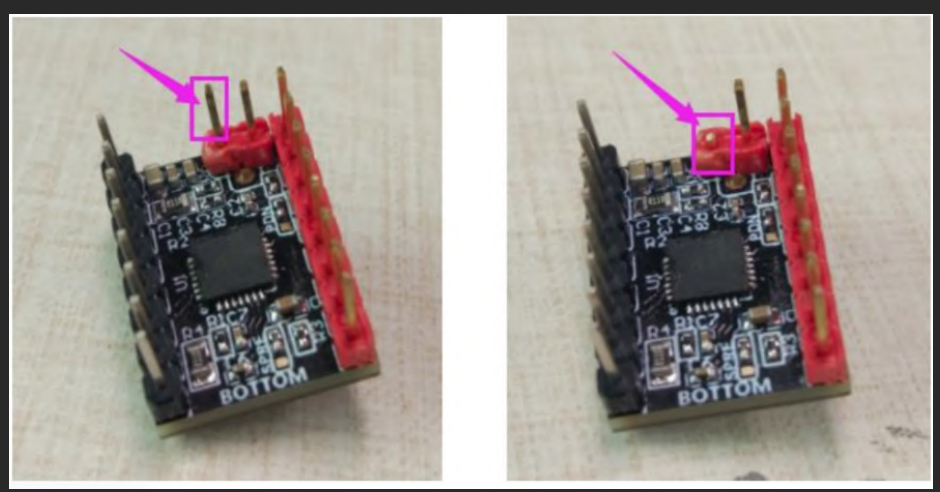
Installing The BTT TMC 2209 Stepper Drivers

Before we install the drivers, you should have decided whether you want to use the Sensorless Homing feature that the 2209s offer. This is also known as Stallguard. Essentially, these stepper drivers are capable of gauging resistance on the motors so when an axis hits its limiting point of movement the driver will interpret this physical force and spike in current as an end-stop. In my opinion, it is a pointless feature of convenience. So, we will be using our end-stops. If you want to use Sensorless Homing, you should only use it for your X and Y motors. Also note Sensorless Homing requires different firmware setup not covered in this guide.

Each of your drivers should come with an adhesive heatsink. Apply them. Make sure they're covering the gold vented plate on the driver. Also make sure that the heatsinks themselves are not touching any of the pins nearby.



In order for us to use our end-stops, we'll need to do one more thing to our drivers. We'll need to remove or bend this outer pin which will prevent sensorless homing communication on the driver. I just used my flush cutters and trimmed them off. Easy.



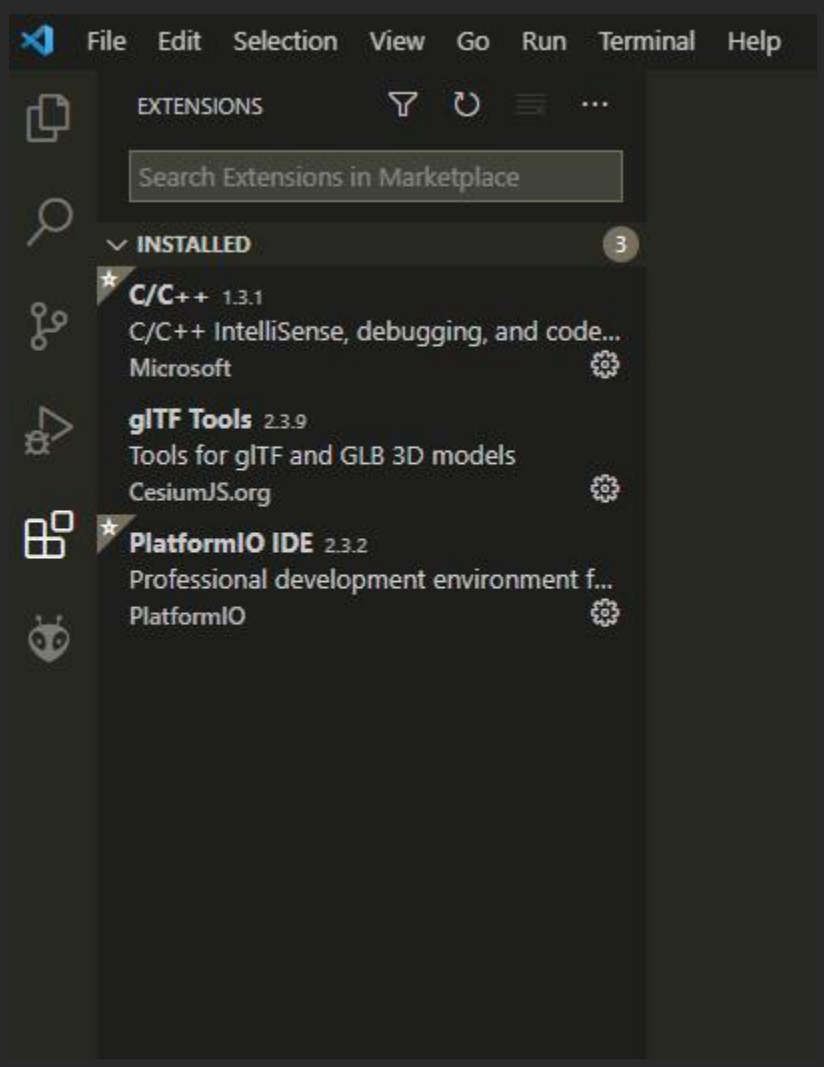
If you're using Sensorless Homing, don't remove the pins from the drivers that will be used for the affected motors (X/Y). Now you can install your stepper drivers into the board.

And that's it! Your SKR 1.4 is now assembled! You can flip your printer back up. I wouldn't close it all up yet - just in case! But before we can begin printing and testing, we've got to set up the firmware that tells the motherboard everything we want set up in it. After all, that is one of the purposes and advantages as to why we're doing all of this in the first place..

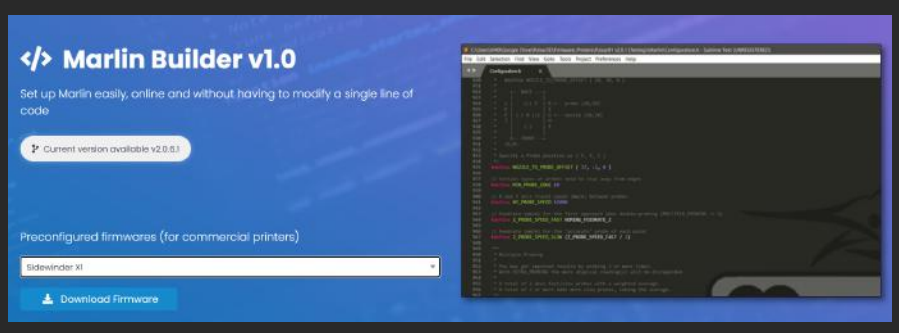
Setting Up The Firmware

Before we begin, we'll need to acquire some software that will help us in editing and compiling our firmware. I recommend using Microsoft's Visual Studio Code which you can download from here. Download it, install it, and load it up.

On the left hand side there's a pane menu we can open for adding extensions. The main extensions we need are **C/C++ and PlatformIO IDE**. You can look them up in the built-in marketplace and download/install them. You'll know PlatformIO is installed properly when a little alien head logo appears in the left pane. Once you've acquired these extensions, we can now move on to gathering the base of our firmware.



In this guide, we'll be using Marlin 2.0.x bugfix. To do less work we'll use the Marlin Builder web tool to gather firmware that already contains most of the parameters that are tuned for the Sidewinder X1.



Download the X1 firmware and unzip it to your desktop. You can rename the folder to whatever you'd like. I renamed mine to "SWX1-MARLIN-2-SKR_1.4T". We can now start making edits to our firmware!

Open up your firmware folder and open the platform.ini file in Studio Code. We'll only need to modify this one line. Change your "default_envs" value to "LPC1769". This will designate the board environment we're compiling with.

```
[platformio]
src_dir      = Marlin
boards_dir   = buildroot/share/PlatformIO/boards
default_envs = LPC1769
include_dir  = Marlin

```

Press CTRL + S to save your changes. You can close this file now.

Configuration.h

Now head to the Marlin folder in your firmware. We'll first open up our "Configuration.h" in Studio Code. You'll find that there are a lot of documented settings in this. For the most part we are just changing values and enabling features by uncommenting (removing "//"). If you're inexperienced just follow what this guide changes. On the left side you can see each line of code numbered, so you can find roughly where and what we're adjusting as you scroll down.

Scroll down to find your serial port values. Change them to what you see here: -1, and 0.

```
/**
 * Select the serial port on the board to use for communication with the host.
 * This allows the connection of wireless adapters (for instance) to non-default port pins.
 * Serial port -1 is the USB emulated serial port, if available.
 * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
 *
 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
 */
#define SERIAL_PORT -1

/**
 * Select a secondary serial port on the board to use for communication with the host.
 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
 */
#define SERIAL_PORT_2 0
```

You can also set your baudrate. By default it's 25000. You can keep it this way. If you have connection drop outs to your computer and printer, I recommend lowering this. You'll also have to then configure it to match in your TFT firmware.

```
/**
 * This setting determines the communication speed of the printer.
 *
 * 250000 works in most cases, but you might try a lower speed if
 * you commonly experience drop-outs during host printing.
 * You may try up to 1000000 to speed up SD file transfer.
 *
 * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
 */
#define BAUDRATE 250000
```

Now we define what motherboard we'll be using.

If you're using the 1.4 your value will be "`BOARD_BTT_SKR_V1_4`".

For 1.4 Turbo you'll use "`BOARD_BTT_SKR_V1_4_TURBO`". I have a turbo, so that's what I'll use.

```
// Choose the name from boards.h that matches your setup
#ifndef MOTHERBOARD
  #define MOTHERBOARD BOARD_BTT_SKR_V1_4_TURBO
#endif

// Name displayed in the LCD "Ready" message and Info menu
#define CUSTOM_MACHINE_NAME "Sidewinder X1"

```

Since I’m using end-stops, these settings are defined.

```
627 //===========================================================================
628 //============================= Endstop Settings ============================
629 //===========================================================================
630
631 // @section homing
632
633 // Specify here all the endstop connectors that are connected to any endstop or probe.
634 // Almost all printers will be using one per axis. Probes will use one or more of the
635 // extra connectors. Leave undefined any used for non-endstop and non-probe purposes.
636 #define USE_XMIN_PLUG
637 #define USE_YMIN_PLUG
638 #define USE_ZMIN_PLUG
639 //#define USE_XMAX_PLUG
640 //#define USE_YMAX_PLUG
641 //#define USE_ZMAX_PLUG
```

This determines end-stop signal behavior. Ensure your values match so your end-stops work properly.

```
669 // Mechanical endstop with COM to ground and NC to Signal uses "false" here (most common setup).
670 #define X_MIN_ENDSTOP_INVERTING true // Set to true to invert the logic of the endstop.
671 #define Y_MIN_ENDSTOP_INVERTING true // Set to true to invert the logic of the endstop.
672 #define Z_MIN_ENDSTOP_INVERTING true // Set to true to invert the logic of the endstop.
673 #define X_MAX_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
674 #define Y_MAX_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
675 #define Z_MAX_ENDSTOP_INVERTING false // Set to true to invert the logic of the endstop.
676 #define Z_MIN_PROBE_ENDSTOP_INVERTING false // Set to true to invert the logic of the probe.
```

Now we can define what stepper drivers we’re using: the TMC2209! You can clarify if you’re using other drivers here as well. Just make sure the SKR 1.4 supports it.

```
678 /**
679  * Stepper Drivers
680  *
681  * These settings allow Marlin to tune stepper driver timing and enable adva
682  * stepper drivers that support them. You may also override timing options i
683  *
684  * A4988 is assumed for unspecified drivers.
685  *
686  * Options: A4988, A5984, DRV8825, LV8729, L6470, L6474, POWERSTEP01,
687  *          TB6560, TB6600, TMC2100,
688  *          TMC2130, TMC2130_STANDALONE, TMC2160, TMC2160_STANDALONE,
689  *          TMC2208, TMC2208_STANDALONE, TMC2209, TMC2209_STANDALONE,
690  *          TMC26X,  TMC26X_STANDALONE,  TMC2660, TMC2660_STANDALONE,
691  *          TMC5130, TMC5130_STANDALONE, TMC5160, TMC5160_STANDALONE
692  * :['A4988', 'A5984', 'DRV8825', 'LV8729', 'L6470', 'L6474', 'POWERSTEP01',
693  */
694 #define X_DRIVER_TYPE  TMC2209
695 #define Y_DRIVER_TYPE  TMC2209
696 #define Z_DRIVER_TYPE  TMC2209
697 //#define X2_DRIVER_TYPE A4988
698 //#define Y2_DRIVER_TYPE A4988
699 #define Z2_DRIVER_TYPE TMC2209
700 //#define Z3_DRIVER_TYPE A4988
701 //#define Z4_DRIVER_TYPE A4988
702 #define E0_DRIVER_TYPE TMC2209
703 #define E1_DRIVER_TYPE TMC2209
704 //#define E2_DRIVER_TYPE A4988
705 //#define E3_DRIVER_TYPE A4988
706 //#define E4_DRIVER_TYPE A4988
707 //#define E5_DRIVER_TYPE A4988
708 //#define E6_DRIVER_TYPE A4988
709 //#define E7_DRIVER_TYPE A4988
710
```

Here you can invert the direction each axis travels. If you’re noticing a carriage moving in the opposite direction intended, you invert the value in the firmware here or flip the affected connector. These values worked for me.

```
1103 // Invert the stepper direction. Change (or reverse the motor connector) if an axis goes the wrong way.
1104 #define INVERT_X_DIR true
1105 #define INVERT_Y_DIR true
1106 #define INVERT_Z_DIR false
```

This determines which direction your extruder motor is driven. I have a Bondtech LGX, so my value is “true”. If you have the regular X1 stock extruder, use “false”.

```
// For direct drive extruder v9 set to true, for geared extruder set to false.
#define INVERT_E0_DIR true
#define INVERT_E1_DIR false
#define INVERT_E2_DIR false
#define INVERT_E3_DIR false
#define INVERT_E4_DIR false
#define INVERT_E5_DIR false
#define INVERT_E6_DIR false
#define INVERT_E7_DIR false
```

Make sure these values match for proper homing behavior.

```
// Direction of endstops when homing; 1=MAX, -1=MIN
// :[-1,1]
#define X_HOME_DIR -1
#define Y_HOME_DIR -1
#define Z_HOME_DIR -1
```

Here we enable EEPROM. Now we can save settings if we’ve changed any values through the firmware.

```
/**
 * EEPROM
 *
 * Persistent storage to preserve configurable settings across reboots.
 *
 *   M500 - Store settings to EEPROM.
 *   M501 - Read settings from EEPROM. (i.e., Throw away unsaved changes)
 *   M502 - Revert settings to "factory" defaults. (Follow with M500 to init the EEPROM.)
 */
#define EEPROM_SETTINGS       // Persistent storage with M500 and M501
//#define DISABLE_M503        // Saves ~2700 bytes of PROGMEM. Disable for release!
#define EEPROM_CHITCHAT       // Give feedback on EEPROM commands. Disable to save PROGMEM.
#define EEPROM_BOOT_SILENT    // Keep M503 quiet and only give errors during first load
#if ENABLED(EEPROM_SETTINGS)
  //#define EEPROM_AUTO_INIT  // Init EEPROM automatically on any errors.
#endif
```

Here we enable SD card support for the board.

```
/**
 * SD CARD
 *
 * SD Card support is disabled by default. If your controller has an SD slot,
 * you must uncomment the following option or it won't work.
 *
 */
#define SDSUPPORT
```

And here we define the board RGB pins for our LED light! Copy these values:

```
#define RGB_LED
//#define RGBW_LED

#if EITHER(RGB_LED, RGBW_LED)
  #define RGB_LED_R_PIN P1_21
  #define RGB_LED_G_PIN P1_19
  #define RGB_LED_B_PIN P1_23
  #define RGB_LED_W_PIN -1
#endif
```

This concludes what we need out of this file. I encourage you to read through all of your options and features that the firmware offers. Press CTRL + S to save and close.

Now for the last file we’ll modify…

Configuration_adv.h

Open your “Configuration_adv.h” file in Visual Studio Code. There’s not much we’ll change in here. Follow along!

Scroll down to these lines. For our extruder fan to adjust to hotends temp properly, we’ll define what pin we’re using here:

```
455   #define E0_AUTO_FAN_PIN P2_04
456   #define E1_AUTO_FAN_PIN -1
457   #define E2_AUTO_FAN_PIN -1
458   #define E3_AUTO_FAN_PIN -1
459   #define E4_AUTO_FAN_PIN -1
460   #define E5_AUTO_FAN_PIN -1
461   #define E6_AUTO_FAN_PIN -1
462   #define E7_AUTO_FAN_PIN -1
463   #define CHAMBER_AUTO_FAN_PIN -1
```

These are nice features in Marlin 2.0.x and should help with our printing quality. Enable them by uncommenting.

```
950   * Adaptive Step Smoothing increases the resolution of multi-axis moves, particularly at step frequencies
951   * below 1kHz (for AVR) or 10kHz (for ARM), where aliasing between axes in multi-axis moves causes audible
952   * vibration and surface artifacts. The algorithm adapts to provide the best possible step smoothing at the
953   * lowest stepping frequencies.
954   */
955  #define ADAPTIVE_STEP_SMOOTHING
```

```
1606  * Linear Pressure Control v1.5
1607  *
1608  * Assumption: advance [steps] = k * (delta velocity [steps/s])
1609  * K=0 means advance disabled.
1610  *
1611  * NOTE: K values for LIN_ADVANCE 1.5 differ from earlier versions!
1612  *
1613  * Set K around 0.22 for 3mm PLA Direct Drive with ~6.5cm between the drive gear and heatbreak.
1614  * Larger K values will be needed for flexible filament and greater distances.
1615  * If this algorithm produces a higher speed offset than the extruder can handle (compared to E jerk)
1616  * print acceleration will be reduced during the affected moves to keep within the limit.
1617  *
1618  * See https://marlinfw.org/docs/features/lin_advance.html for full instructions.
1619  */
1620 #define LIN_ADVANCE
1621 #if ENABLED(LIN_ADVANCE)
1622   //#define EXTRA_LIN_ADVANCE_K // Enable for second linear advance constants
1623   #define LIN_ADVANCE_K 0.13    // Unit: mm compression per 1mm/s extruder speed
1624   //#define LA_DEBUG            // If enabled, this will generate debug information output over USB.
1625   #define EXPERIMENTAL_SCURVE // Enable this option to permit S-Curve Acceleration
1626 #endif
```

Then we will indicate the voltage and the micro steps to which each of the motors must operate. For the X1 we have X, Y, Z, E0, and E1, so I will modify the values in the corresponding section. I used the same values for each motor as a start. Scroll through this entire section and check.

```
2233  #if AXIS_IS_TMC(X)
2234    #define X_CURRENT       800        // (mA) RMS current. Multiply by 1.414 for peak current.
2235    #define X_CURRENT_HOME  X_CURRENT  // (mA) RMS current for sensorless homing
2236    #define X_MICROSTEPS     16    // 0..256
2237    #define X_RSENSE          0.11
2238    #define X_CHAIN_POS      -1    // <=0 : Not chained. 1 : MCU MOSI connected. 2 : Next in chain, ...
2239  #endif
2240
2241  #if AXIS_IS_TMC(X2)
2242    #define X2_CURRENT      800
2243    #define X2_CURRENT_HOME X2_CURRENT
2244    #define X2_MICROSTEPS    16
2245    #define X2_RSENSE         0.11
2246    #define X2_CHAIN_POS     -1
2247  #endif
2248
2249  #if AXIS_IS_TMC(Y)
2250    #define Y_CURRENT       800
2251    #define Y_CURRENT_HOME  Y_CURRENT
2252    #define Y_MICROSTEPS     16
2253    #define Y_RSENSE          0.11
2254    #define Y_CHAIN_POS      -1
2255  #endif
```

Check each motor!

Enabling Stealth Chop option for silent operation. As if the X1 wasn't quiet enough! Oohh yeaaahh…

```
2421    * TMC2130, TMC2160, TMC2208, TMC2209, TMC5130 and TMC5160 only
2422    * Use Trinamic's ultra quiet stepping mode.
2423    * When disabled, Marlin will use spreadCycle stepping mode.
2424    */
2425   #define STEALTHCHOP_XY
2426   #define STEALTHCHOP_Z
2427   #define STEALTHCHOP_E
```

Make sure the power supply here is correct. By default the X1 has a 24V power supply.

```
2442    * { <off_time[1..15]>, <hysteresis_end[-3..12]>, hysteresis_start[1..8] }
2443    */
2444   #define CHOPPER_TIMING CHOPPER_DEFAULT_24V
2445
```

With this enabled, we can monitor our drivers and modify their current in Marlin using these machine commands! Sweet!

```
2447    * Monitor Trinamic drivers
2448    * for error conditions like overtemperature and short to ground.
2449    * To manage over-temp Marlin can decrease the driver current until the error condition clears.
2450    * Other detected conditions can be used to stop the current print.
2451    * Relevant G-codes:
2452    * M906 - Set or get motor current in milliamps using axis codes X, Y, Z, E. Report values if no axis codes given.
2453    * M911 - Report stepper driver overtemperature pre-warn condition.
2454    * M912 - Clear stepper driver overtemperature pre-warn condition flag.
2455    * M122 - Report driver parameters (Requires TMC_DEBUG)
2456    */
2457   #define MONITOR_DRIVER_STATUS
2458
2459   #if ENABLED(MONITOR_DRIVER_STATUS)
2460     #define CURRENT_STEP_DOWN     50  // [mA]
2461     #define REPORT_CURRENT_CHANGE
2462     #define STOP_ON_ERROR
2463   #endif
```

This will help improve the signal to our stepper drivers. Optionally enable.

```
2531    * TMC Homing stepper phase.
2532    *
2533    * Improve homing repeatability by homing to stepper coil's nearest absolute
2534    * phase position. Trinamic drivers use a stepper phase table with 1024 values
2535    * spanning 4 full steps with 256 positions each (ergo, 1024 positions).
2536    * Full step positions (128, 384, 640, 896) have the highest holding torque.
2537    *
2538    * Values from 0..1023, -1 to disable homing phase for that axis.
2539    */
2540    //#define TMC_HOME_PHASE { 896, 896, 896 }
2541
2542   /**
2543    * Beta feature!
2544    * Create a 50/50 square wave step pulse optimal for stepper drivers.
2545    */
2546   #define SQUARE_WAVE_STEPPING
```

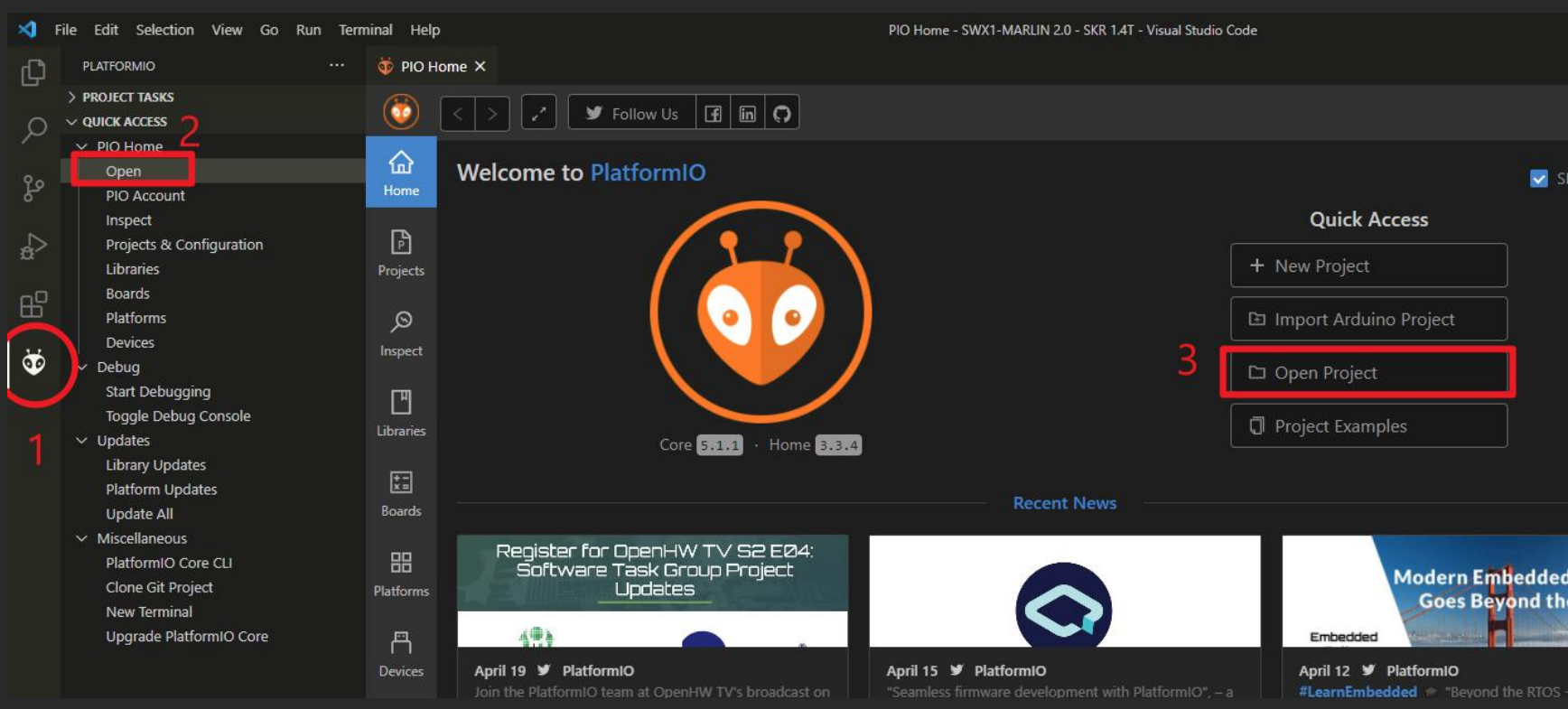
We'll need this to also monitor our drivers. Enable.

```
2548   /**
2549    * Enable M122 debugging command for TMC stepper drivers.
2550    * M122 S0/1 will enable continous reporting.
2551    */
2552   #define TMC_DEBUG
2553
```

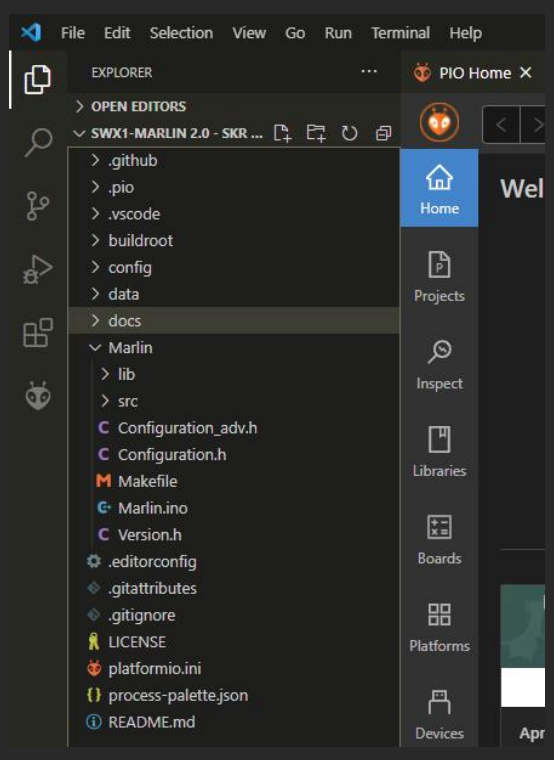
Whew! Now we're done! Save your file. We can now prepare to compile our firmware.

Building The Firmware

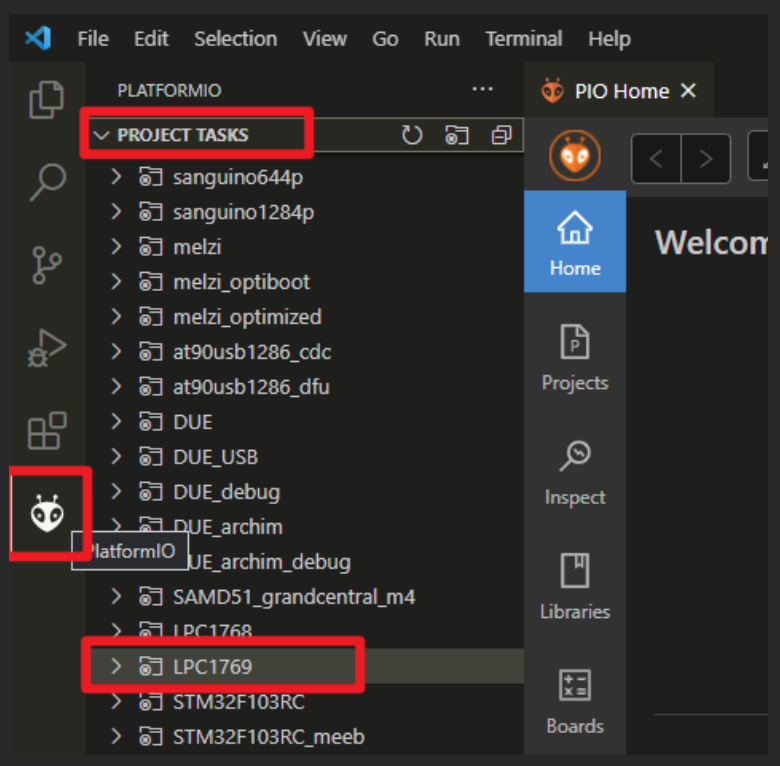
We'll start by opening our firmware project with PlatformIO. Open it by using "Open Project", then navigate to and select your firmware folder on your desktop.



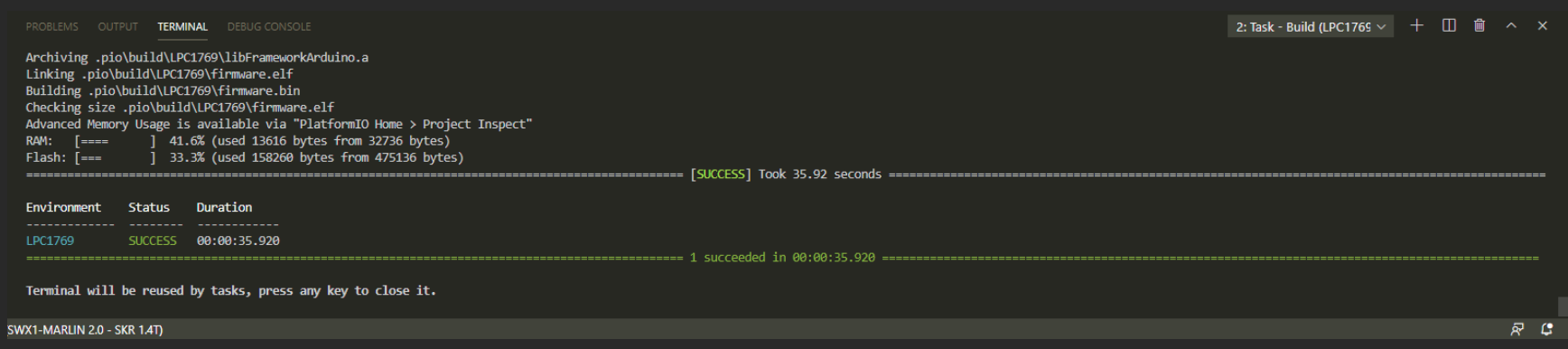
If you have the file explorer menu selected in Studio Code, you can see our entire project directory here.



And now we can begin with compiling our firmware. Remember that value we changed in our platformio.ini file? LPC1769 is the build environment for our SKR 1.4 board. So we'll open the Platform IO menu and look for our board.



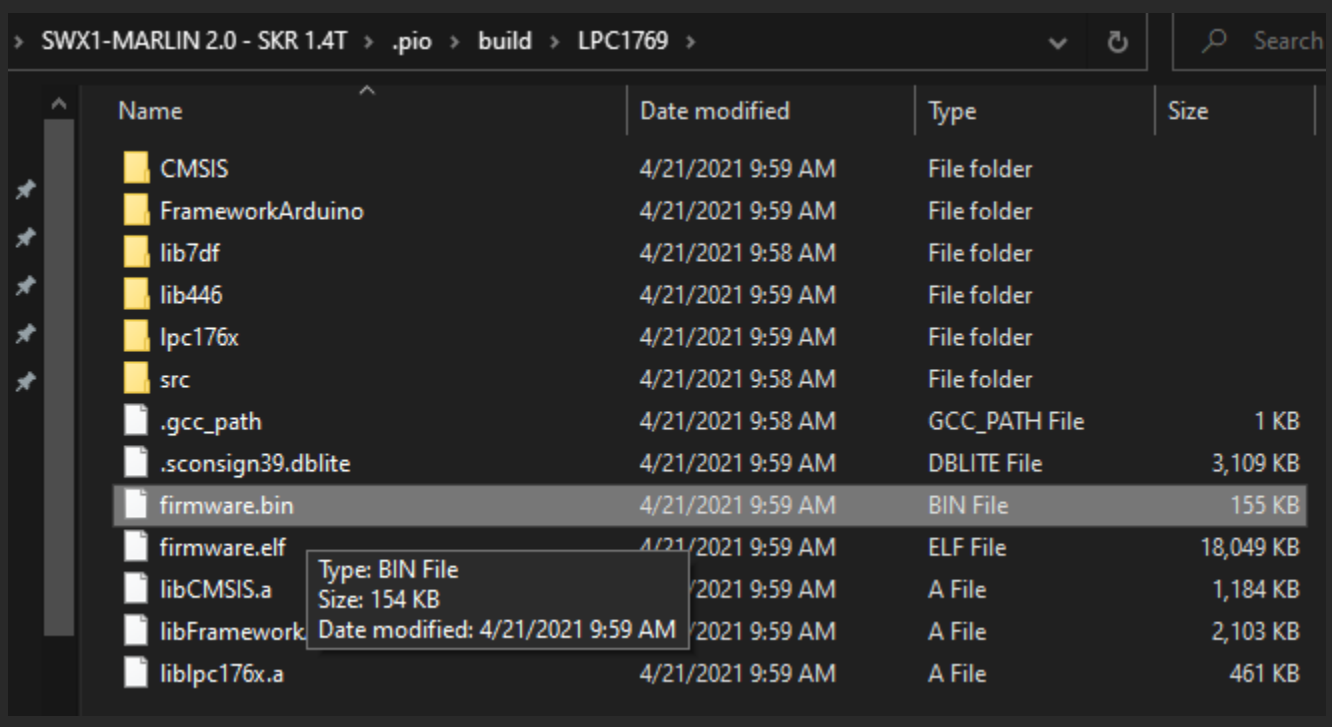
In the LPC1769 submenu we have some options. Click "Build" to begin compiling our firmware. A terminal at the bottom of Studio Code will open and begin displaying the process. Depending on your machine's specs, this can take anywhere from 30 seconds to a few minutes. WIth the changes we've made, we shouldn't have any compiling issues. If you do, don't worry. The terminal will point out what conflicting values are causing the problem. Once you've identified them, find them in either the "Configuration.h" or "Configuration_adv.h" files and address them. This typically means you forgot to enable or disable a feature, or there is a set value that is incorrect. Some features require other features to be enabled to work. Even typos can cause issues, so double check for them! Once you're sure your code is good, save. Click "Clean" in the Platform IO menu to remove previous builds, then click "Build" to compile again.



Flashing the Firmware

Now with our firmware compiled, we can flash the new board. Connect your printer to your PC via the USB port on the board and power it on. You should already have the SD card inserted into the board that came with it. With the SD card inserted, it will basically act like a USB drive when we connect the printer. Go to the SD card's drive on disc.

In another file explorer window, go to your firmware folder and you'll see Platform IO added some new folders when building our firmware. Dive into .pio > build > LPC1769. If you cannot find this then your firmware wasn't compiled. Find your "firmware.bin" folder and copy it into your SD card drive.



Disconnect your USB and reboot your printer, then wait 30 seconds.

And you're finished!

I recommend testing some commands from the TFT screen, like movement and homing, load/unload filament, to ensure things are behaving properly. Also test that the hotend and bed heat up properly.

If you need to adjust settings in your firmware, simply make the changes, save, recompile, and drag your bin to the SD card's drive. Reboot printer each time to flash. You'll notice your LEDs will go a solid color when the flashing is done.

Troubleshooting

"My printer shuts off and reboots whenever I heat up the bed/hotend."

It's likely you have the positive and negative wires hooked up wrong to the board. Isolate whether it's the bed, hotend, or both. Switch the wire positions and test again.

"My printer's carriage is moving in the wrong direction!"

Looks like your firmware setting is wrong. You can flip the connector around or change the firmware value. Check the bottom of page 9 in this guide to see how we set the direction of the axis.

"My printer is homing incorrectly!"

Check your firmware. See page 10 in this guide where you can set homing direction values with your end-stops.

If you have any other issues, I recommend joining the Discord mentioned at the beginning of this guide. There are many experienced users there who are happy to help and answer questions to the best of their ability.

HAPPY PRINTING!