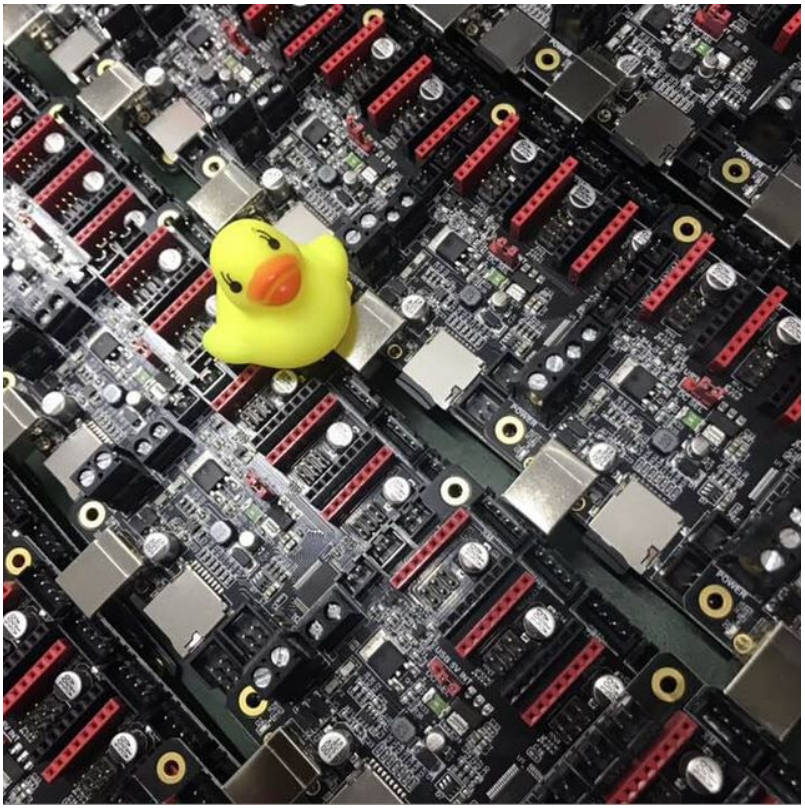
Guía BIGTREETECH SKR V1.3, V1.4 SKR Mini E3

Canal telegram https://t.me/SKR_board_32bits (Versión 29-05-2020. @LoKuS77)



Me ha llegado una bonita placa negra SKR de Aliexpress/Amazon **¿Y ahora qué?** ☺

Toda la información oficial, como esquemas, cableados, jumpers y hardware sobre esta placa de 32 bits se encuentra en el

Sitio oficial GITHUB de BIGTREETECH.

-**SKR 1.3 y 1.4 / 1.4 turbo:** https://github.com/bigtreetech/BIGTREETECH-SKR-V1.3

-**SKR Mini E3** https://github.com/bigtreetech/BIGTREETECH-SKR-mini-E3

-**SKR E3 Dip** https://github.com/bigtreetech/BIGTREETECH-SKR-E3-DIP-V1.0

-**Otras SKR:** https://github.com/bigtreetech?tab=repositories

Esta guía muestra cómo instalar el firmware Marlin 2.0 firmware en la placas SKR V1.3,SKR V1.4, SKR MINI E3 en general. No tiene en cuenta tu tipo-modelo de impresora, u otros ajustes específicos de tu hardware! Este documento se basa en S.O. Windows.

En placas de 32bits, no podemos compilar Marlin utilizando el IDE de arduino.
(Eso sólo sirve para electrónicas de 8 bits con cpu AVR Atmega atmega2560, atmega1284p, etc…)

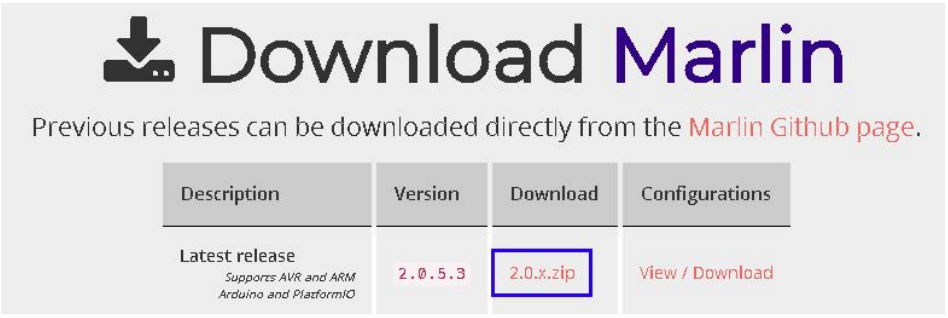
Para compilar Marlin 2.0 se recomienda usar Microsoft Visual Studio Code (VS Code) + el plugin + Platform IO.

* Esta guía da por hecho que vas a usar una pantalla LCD de tipo 128x64 estándar REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER con dos cables cinta de 10 pins.

Descarga de marlin 2.0

Para descargar la versión estable más reciente de Marlin 2.0, iremos a https://marlinfw.org/meta/download/ hy descargaremos 2.0.x.zip

- Marlin 2.0 Release (más estable)
Enlace directo al ZIP **https://github.com/MarlinFirmware/Marlin/archive/2.0.x.zip**



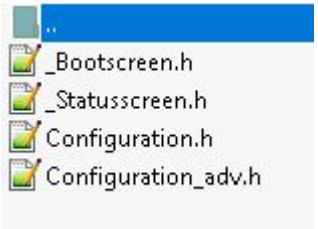
NOTAS: Si tienes previamente descargado un Marlin 2.0 de procedencia o fecha desconocida, puedes mirar de cuando es, revisando el fichero “versión.h” en la carpeta \Marlin-bugfix-2.0.x\Marlin\src\inc**Version.h**
Abrelo con block de notas o notepad++, y sobre la línea 50 encontrarás la fecha de ese marlín:
#define STRING_DISTRIBUTION_DATE "2020-03-31".
Como norma general, conviene evitar utilizar versiones de marlin 2 previas a la Release (2.0.0) de fecha 1 diciembre 2019.
Si tu marlín es más antiguo, o tiene más de un mes... bájate el último.

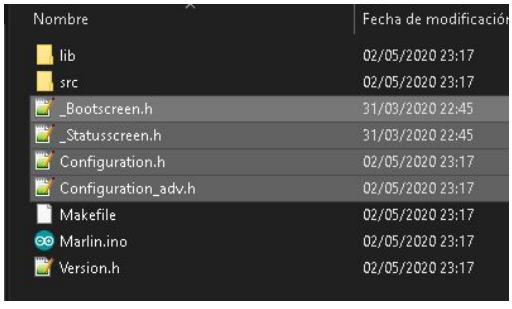
Aparte de marlin 2.0 Release (estable), está Marlin 2.0.x **Bugfix**, en el que prácticamente cada día corrigen de cosas.
Es una versión más actualizada que la última release, pero puede ser más inestable que marlín 2.0.XX Release.

Una vez descargado Marlin 2.0 Release o bugfix, la descomprimimos en una ruta conocida.
Es aconsejable utilizar una **ruta corta**, tipo c:\mi marlin\ .
Muchas veces Marlin da errores de compilación por utilizar **rutas en disco muy largas,** tipo al escritorio
(que realmente puede ser “c:\users\manolito perez gomez\desktop\cosasquemebajo\marlin\el de ayer\marlin2.0-bugfix\”)

Una vez descargado Marlin, procederemos a bajar el pack de configuraciones de ejemplo (antes venía todo junto, ahora lo han separado). Están en https://github.com/MarlinFirmware/configurations
Si has bajado Marlin 2.0.5.3 Release, el zip directo es este:
https://github.com/MarlinFirmware/Configurations/archive/release-2.0.5.zip
Si has bajado Marlin Bugfix, es este: https://github.com/MarlinFirmware/Configurations/archive/bugfix-2.0.x.zip
Podemos descomprimirlo entero, o bien coger sólo los config de ejemplo de nuestra impresora.
Por ejemplo para una Ender3, los ficheros de dentro de la carpeta \Configurations-2.0.5.3\config\examples\Creality\Ender-3\

Cogeremos estos ficheros (en el caso de la ender son 4 al llevar un logo de inicio propio) y los copiaremos en la carpeta \Marlin-2.0.x**Marlin**\ sobreescribiendo los que hubiera.
Con esto, tendremos la config de serie de la máquina elegida (velocidades, pasos por mm, y resto de configuraciones).





La carpeta en la que copiar los ficheros, tiene marlin.ino, subcarpetas Lib y Src, y esos mismos ficheros de configuración, que deberemos sobreescribir.

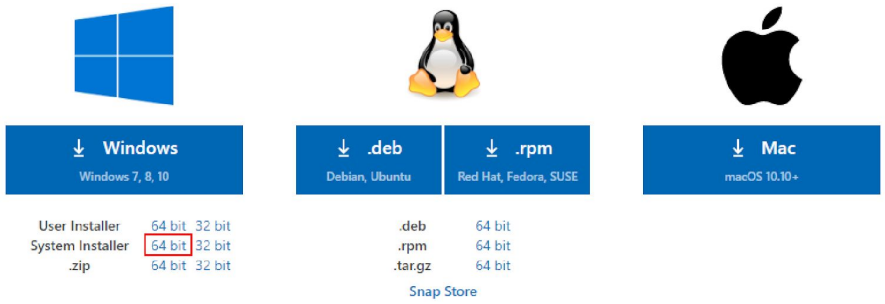
Instalación VS CODE + Platformio

(además de lo aquí escrito, existe una guía de instalación oficial en inglés)

Para compilar Marlin utilizamos ***Visual Studio Code.*** Es una herramienta gratuita, de microsoft, y es la recomendada por el equipo de Marlin. (Antes también se utilizaba Atom+ Platformio, pero actualmente no se le da soporte)

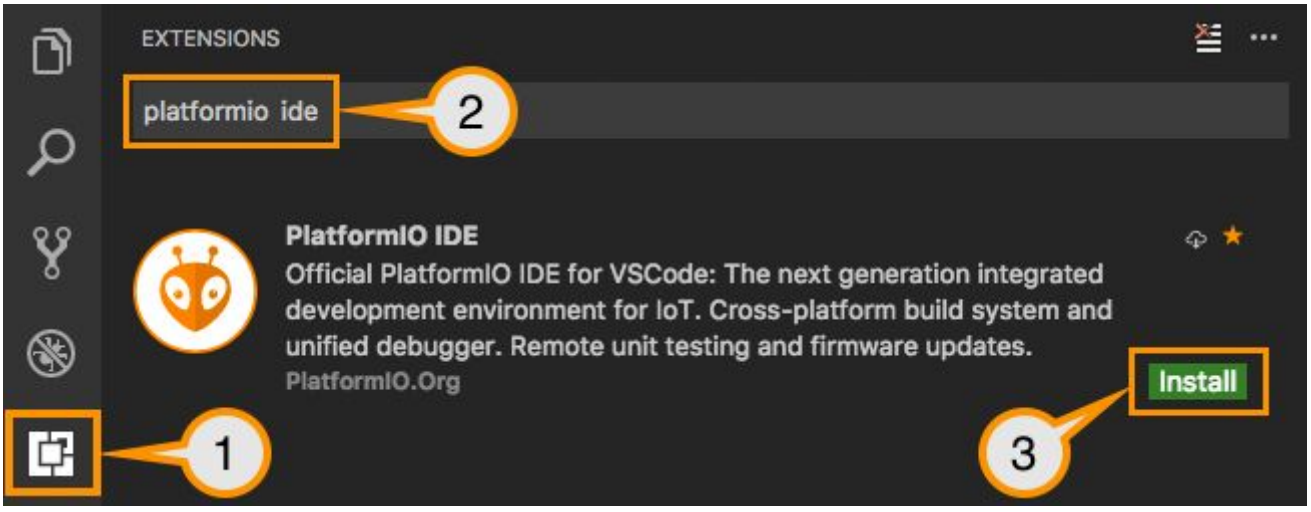
Descarga VScode aquí: https://code.visualstudio.com/Download

Elige la versión que coincida con tu sistema operativo.



Después de terminar la descarga, haz doble click para realizar la instalación. Una vez instalado, abrimos VScode.

Dentro de VScode, necesitaremos también instalar el plugin **PlatformIO.**



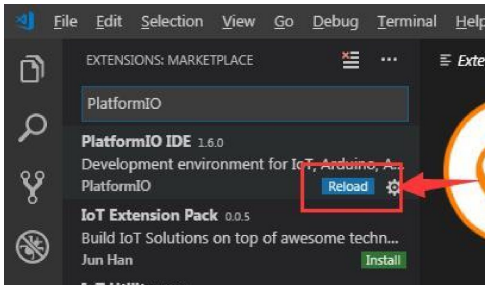
Para instalar el plugin **PlatformIO** realizaremos los pasos:

-Click en la figura  del paso 1

-Escribimos **PlatformIO** en el recuadro del paso 2.

-Hacemos click en Install en PlatformIO IDE 1.xx , del paso 3.

Después de completar la descarga, necesitaremos hacer una recarga (Reload)

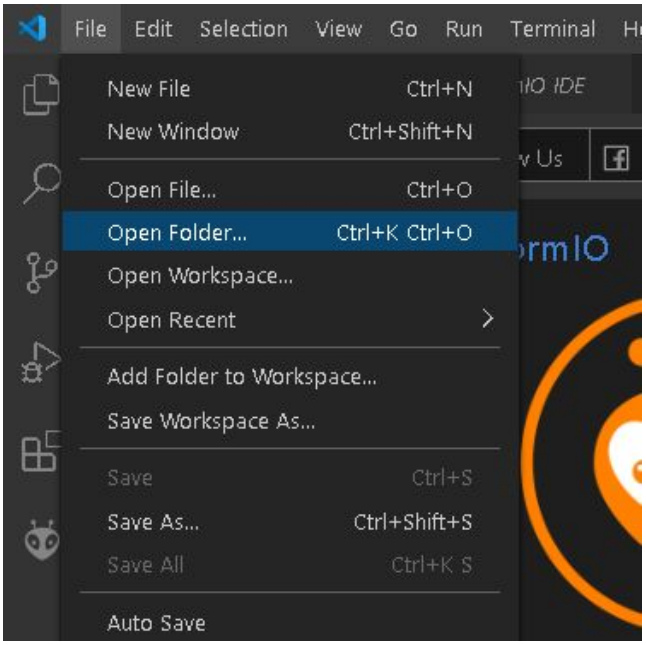


Una vez instalado, reiniciamos VScode.

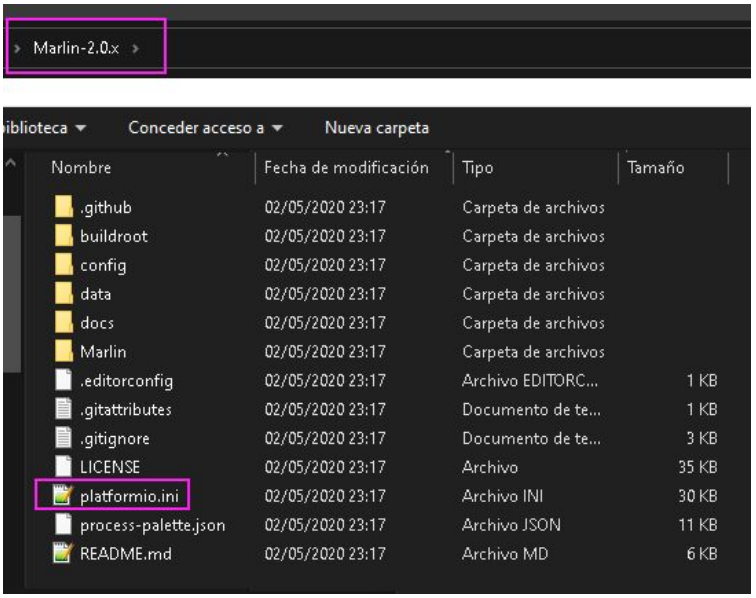
Si se ha instalado correctamente,en el panel izquierdo, veremos un icono hormiguita que es el del plugin PlatformIO.

Para abrir nuestro Marlin 2.0 que hemos descargado, iremos a File / Open folder

y elegiremos la **CARPETA** que contiene el fichero platformio.ini. Una vez elegida pulsamos el botón Seleccionar capeta.



Haremos click en él (1), y después en Open Project (2), para abrir el proyecto.

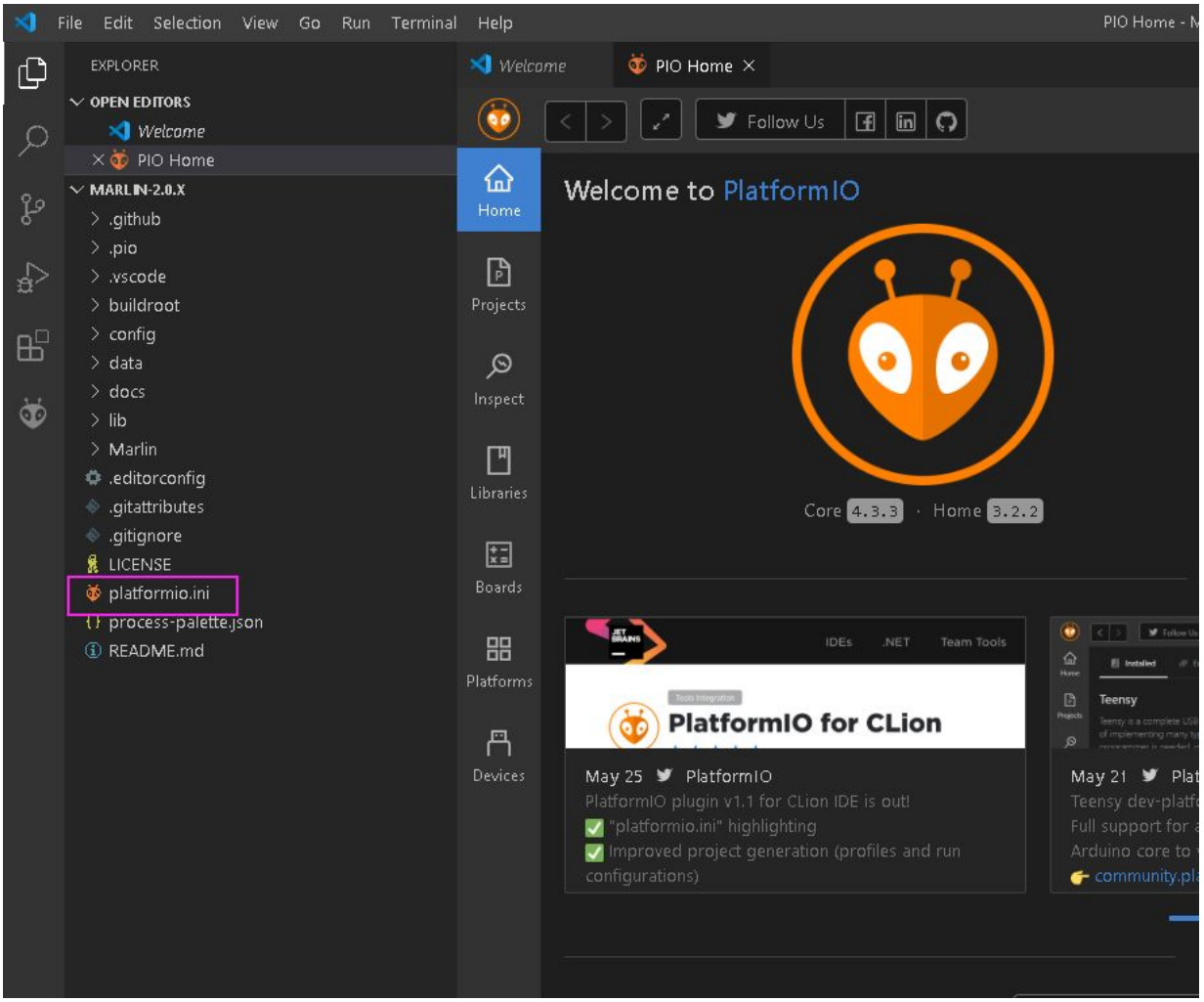


En este caso, he elegido la carpeta “Marlin-2.0.x” que es la que contiene plaformio.ini

Después de abrir el proyecto, se nos mostrará a la izquierda el árbol de múltiples ficheros y carpetas que conforman Marlin.

No hace falta asustarse :) De todos ellos, **sólo vamos a modificar 3 ficheros:**

- MARLIN-2.0.X**plattformio.ini**
- MARLIN-2.0.X\Marlin**configuration.h**
- MARLIN-2.0.X\Marlin**configuration_adv.h**



PLATFORMIO.INI

Inicialmente, iremos al fichero platformio.ini

En este fichero vamos a modificar una sóla línea. En la línea 22: “ **default_envs** = mega2560 “

Esta variable “**default_envs”** le indica a VScode qué tipo de CPU tiene nuestra placa.

El valor que marlin trae por defecto (mega2560) sirve sólo para placas de 8 bits con cpu arduino mega.

Cambiaremos esta línea por la que corresponda:

Para placas SKR 1.1, SKR 1.3, SKR 1.4:

default_envs = **LPC1768**

Para placas SKR 1.4 Turbo:

default_envs = **LPC1769**

Para placas SKR MINI E3, SKR MINI 1.1, SKR E3 DIP:

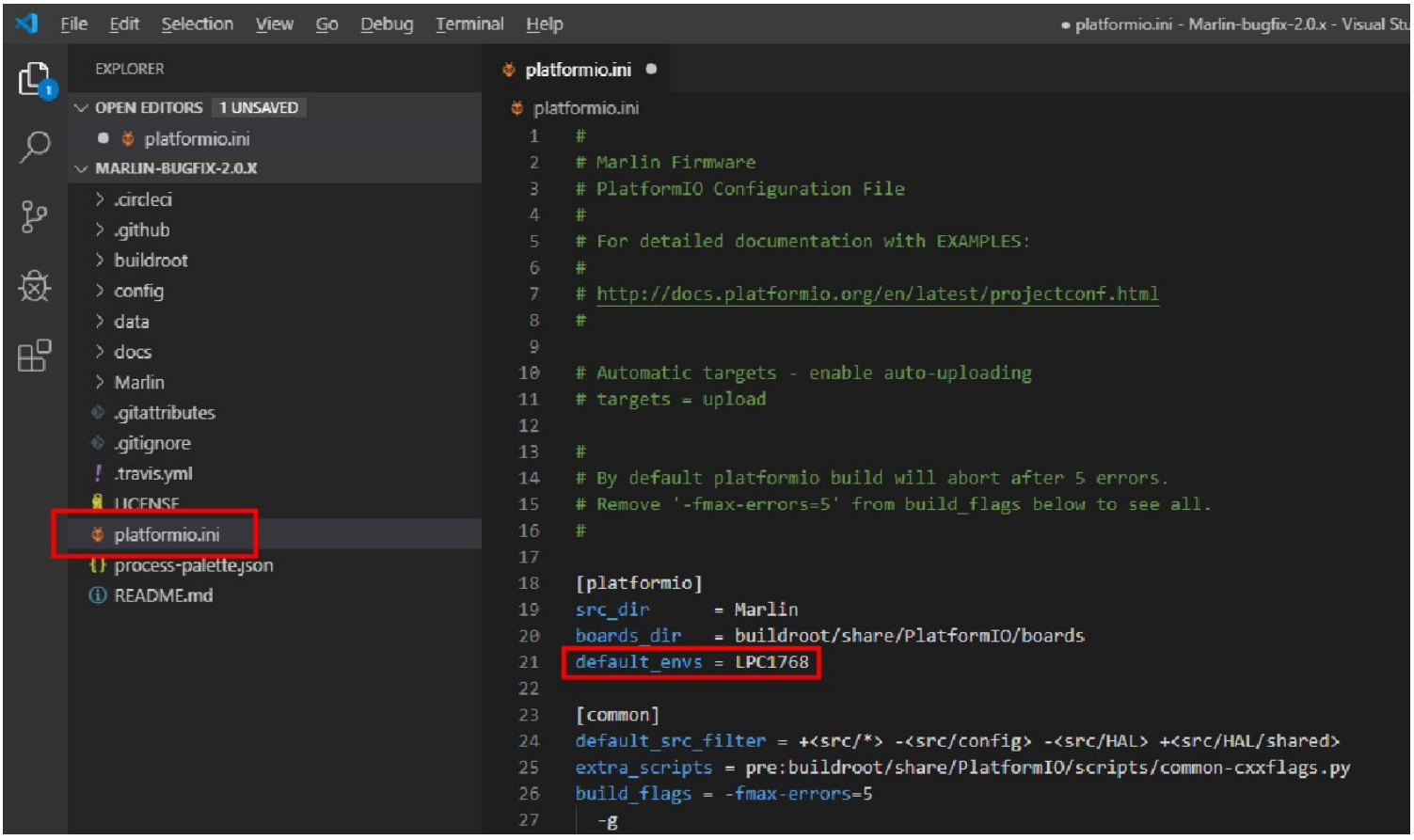
default_envs = **STM32F103RC_btt**

Nota: Alguna SKR MINI E3, SKR MINI 1.1, SKR E3 DIP en ocasiones lleva una cpu ligeramente distinta.

Si no conseguimos compilar mirar apartado RCT6 / RET6 . Si la cpu es RET6 será:

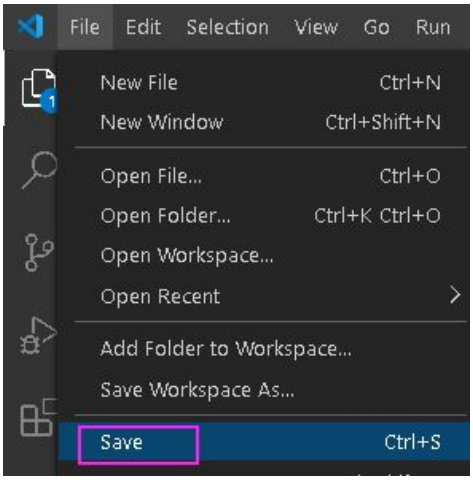
default_envs = **STM32F103RE_btt**

Para otras placas, consultar abajo en el **Apéndice Nº1: OTRAS PLACAS**:



En la imagen del ejemplo hemos puesto LPC1768 (Válido solo para SKR 1.1, SKR 1.3, SKR 1.4)

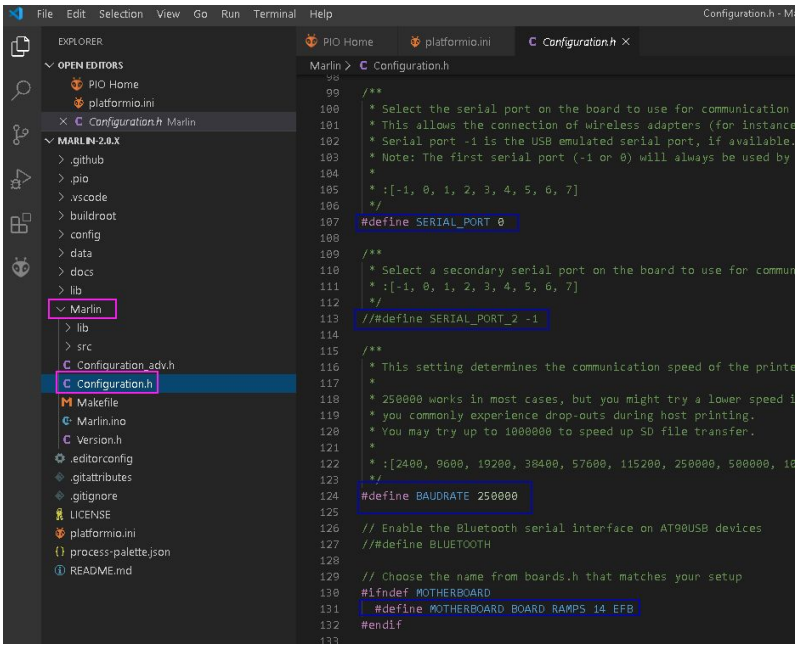
Una vez realizado el cambio, lo guardaremos en File / Save.



Elegir el modelo de placa y puertos serie

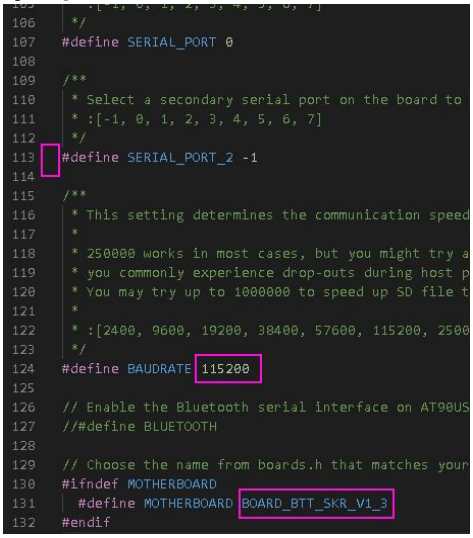
CONFIGURATION.H

Iremos ahora al fichero configuration.h , que se encuentra en la carpeta \Marlin\ ,por defecto viene así:



Si no están ya hechos los cambios, para placas SKR 1.3 pondremos (para resto placas ver tabla debajo):

#define SERIAL_PORT **0**
#define SERIAL_PORT_2 **-1** **(**descomentamos ,eliminar las dos barras //)
#define BAUDRATE **115200** **(**En ocasiones 250.000 da problemas con ciertos cables usb)
#define MOTHERBOARD **BOARD_BTT_SKR_V1_3**
Ejemplo:



La lista de posibles MOTHERBOARD que admite marlin, puede consultarse en el fichero Marlin/Src/Core/**Boards.h**

Para nuestras SKR es:

#define MOTHERBOARD **BOARD_BTT_SKR_V1_1**
#define MOTHERBOARD **BOARD_BTT_SKR_V1_3**
#define MOTHERBOARD **BOARD_BTT_SKR_V1_4**
#define MOTHERBOARD **BOARD_BTT_SKR_V1_4_TURBO**
#define MOTHERBOARD **BOARD_BTT_SKR_MINI_V1_1**
#define MOTHERBOARD **BOARD_BTT_SKR_MINI_E3_V1_0**
#define MOTHERBOARD **BOARD_BTT_SKR_MINI_E3_V1_2**
#define MOTHERBOARD **BOARD_BTT_SKR_MINI_E3_V2_0** ***NO DISPONIBLE AÚN** en marlin 2.0.5.3. Está en bugfix o marlin de BTT
#define MOTHERBOARD **BOARD_BTT_SKR_E3_DIP**
#define MOTHERBOARD **BOARD_BTT_SKR_PRO_V1_1**
#define MOTHERBOARD **BOARD_BTT_BTT002_V1_0**
#define MOTHERBOARD **BOARD_BTT_GTR_V1_0**

El número de puerto serie y puerto serie 2, cambia según el modelo de placa:

Placa	**puertos serie a definir:**
SKR 1.1 SKR 1.3 SKR 1.4 SKR 1.4 Turbo	#define SERIAL_PORT 0 #define SERIAL_PORT_2 -1
SKR E3 DIP SKR Mini E3 1.0 SKR Mini E3 1.2 SKR Mini E3 2.0	#define SERIAL_PORT 2 #define SERIAL_PORT_2 -1
SKR PRO 1.1 SKR MINI 1.1	#define SERIAL_PORT -1 #define SERIAL_PORT_2 1

*datos cogidos de los configs de ejemplo de cada placa en su documentación oficial

PANTALLAS LCD

Si usamos pantalla de tipo **LCD12864:**

En el fichero configuration.h, descomentaremos

#define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER

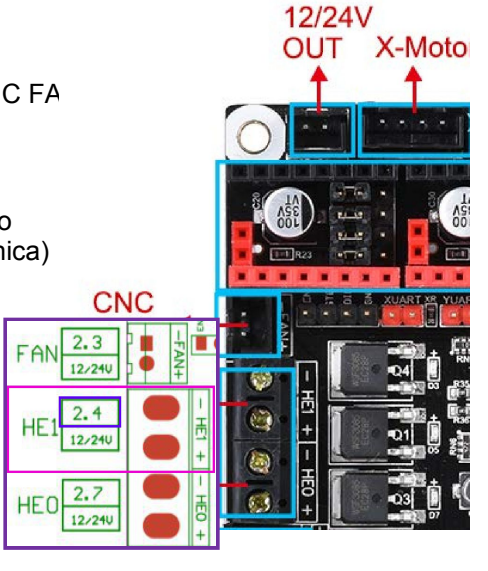
(Si utilizamos otro tipo de pantalla, des comentaremos la que utilicemos).

Pantallas habituales:

- Creality Ender3 / CR10 → #define CR10_STOCKDISPLAY
- Bigtreetech TFT24 v1.1/TFT35 v3.0 (con emulación 12864)
  → #define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER
- Anet A8: → #define ZONESTAR_LCD
- Tevo Tornado → #define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER
- Tevo Tarántula PRO → #define MKS_MINI_12864
- TFT24/TFT28/TFT35 MKS, o Bigtreeech táctil, no dual → NO se configura en marlín. Ver apartado TFT

  Si el LCD no da imagen, revisa el apartado Problemas LCD

Configuración de ventiladores.



-**Ventilador de capa:** Si quieres usar un ventilador de capa, conéctalo al conector CNC FA Marlin (y tu laminador al generar Gcode) regulan la potencia del ventilador por software. Más info: **Gcode M106**

-Ventilador Hotend (opcional): Lo habitual es tener el ventilador del hotend conectado directo a 12/24v de la fuente. Si quieres que el ventilador del hotend (o el de la electrónica) sean controlados **de forma automática ON/OFF al llegar a 50ºC el nozzle,** conéctalo a la salida HE1 (E1 heater) y modifica el fichero configuration_adv.h file:

//#define E0_AUTO_FAN_PIN -1
cambiarlo a
#define E0_AUTO_FAN_PIN FAN1_PIN

- Ventilador para la Electrónica y/o drivers:(opcional) Si quieres un ventilador para la placa o drivers, que se active cuando

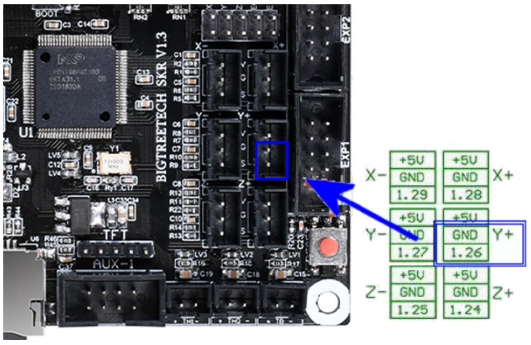
los drivers están activos iremos al fichero configuration_adv.h y cambiaremos:

#define USE_CONTROLLER_FAN

```
#if ENABLED (USE_CONTROLLER_FAN)
  #define CONTROLLER_FAN_PIN P1_26          // define un pin personalizado para el venti de placa
  #define CONTROLLERFAN_SECS 60             // duración en segundos de auto apagado
  #define CONTROLLERFAN_SPEED 255           // 255 == full speed
#endif
```

En la salida del endstop **Ymax** conectaremos un pequeño interruptor mosfet con un conector JST XH2.54 en el en los pins **P1-26 y GND.** No es algo habitual, y yo personalmente no lo he probado.

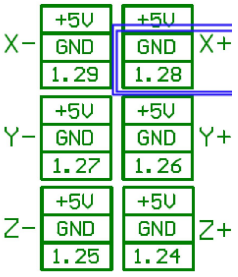
Foto de ejemplo y ENLACE (1€)
¿más info?--> Youtube



Sensor de Filamento (opcional)

Si vamos a utilizar sensor de filamento, la configuración por defecto utiliza el pin de endstop Xmax (**1.28**)
Deberemos editar el fichero **configuration.h:**

-Descomentar **#define FILAMENT_RUNOUT_SENSOR**
-Cambiar de False a True: #define FIL_RUNOUT_INVERTING **true**



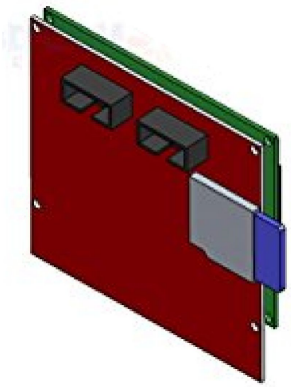
Fichero **configuration_adv.h:**

-Descomentar **#define ADVANCED_PAUSE_FEATURE**
y también **#define NOZZLE_PARK_FEATURE**

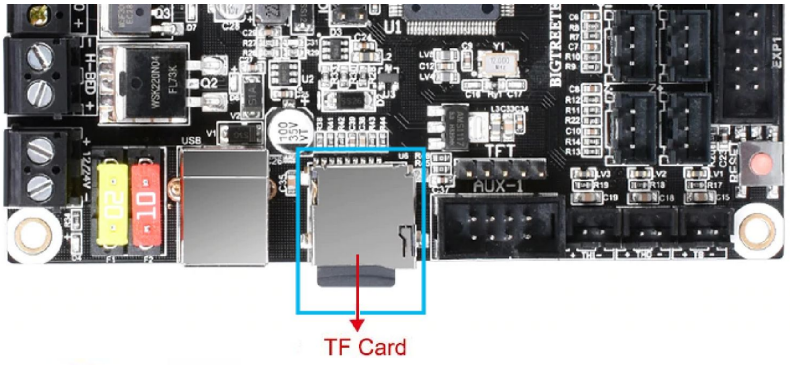
Si quisiéramos utilizar el sensor de filamento en otro lugar distinto de XMAX, deberemos modificar el fichero de pins SKR V1.3 (\Marlin\src\pins\lpc1768**pins_BIGTREE_SKR_V1.3.h** y también **pins_BTT_SKR.h**), ya que por defecto es:
#define FIL_RUNOUT_PIN **P1_28** (conector Endstop Xmax)

Selección de tarjeta SD

Si vamos a utilizar tarjeta SD (habremos descomentado en configuration.h el valor **#define SDSUPPORT**), deberemos elegir si vamos a utilizar la tarjeta SD integrada con nuestra pantalla/TFT, o si vamos a utilizar la propia tarjeta microSD incorporada en la placa Skr V1.3
(Existe también una tercera opción, menos habitual, que es utilizar una tarjeta externa, en el pin que definamos nosotros)



VS (o bien)



Para elegir cuál de los 3 modos queremos usar, deberemos editar en el fichero **Configuration_Adv.h** la línea

```
#if HAS_SDCARD_CONNECTION
    //#define SDCARD_CONNECTION ONBOARD
```

descomentándola y cambiándola a

```
#define SDCARD_CONNECTION ONBOARD        //Para utilizar la tarjeta interna
#define SDCARD_CONNECTION LCD            //Para utilizar la tarjeta de nuestra pantalla lcd/tft
#define SDCARD_CONNECTION CUSTOM_CABLE   //Para tarjetas externas definidas en el fichero de pins.
```

No es habitual que haya fallos de lectura en SD interna de la placa, pero sí son habituales en la SD del lcd, sobre todo si utilizamos un **cable largo** y no está apantallado con papel de aluminio o Ferrita:

Bltouch

Diagrama de pins Bigtreetech SKR V1.3 y Bltouch:

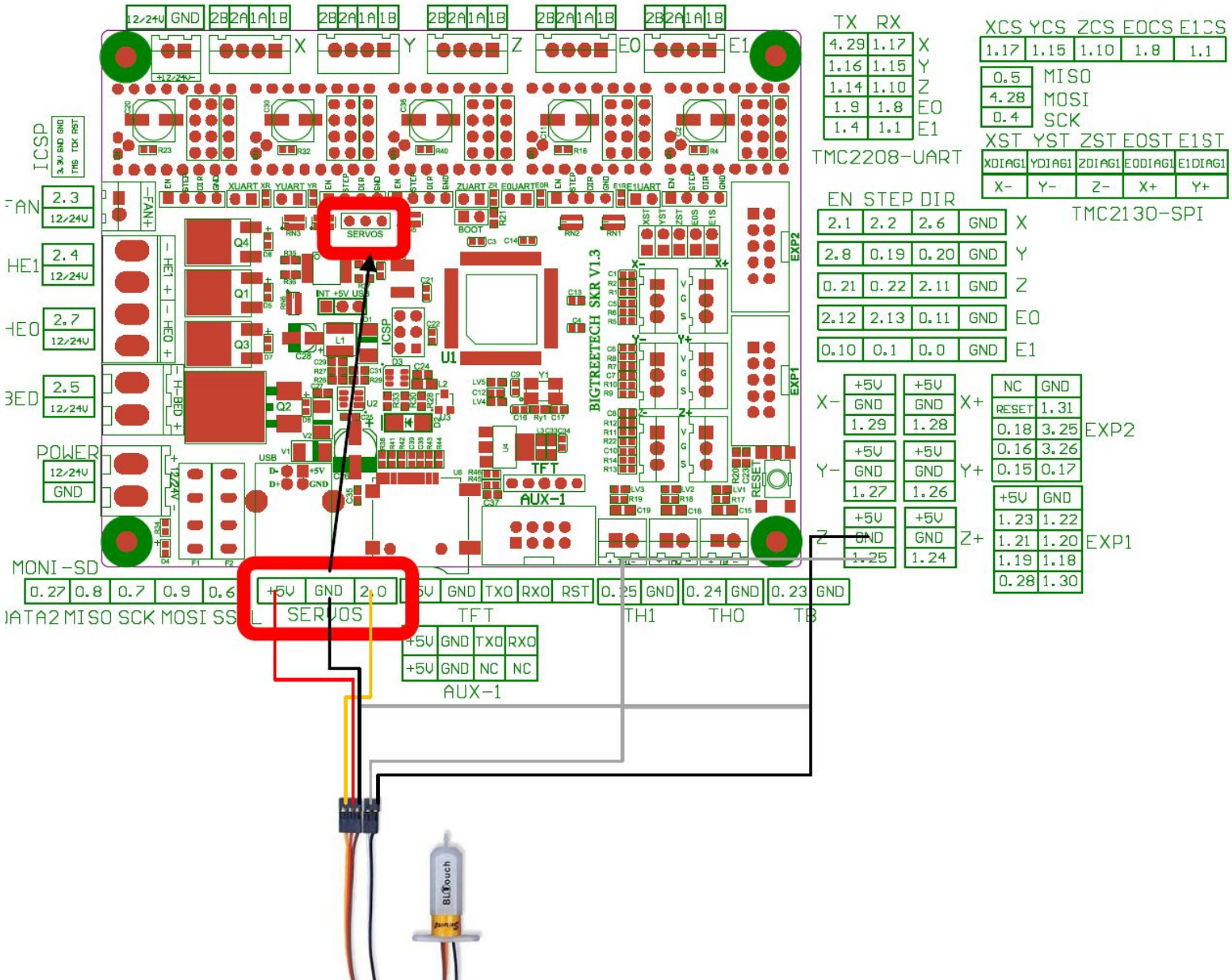
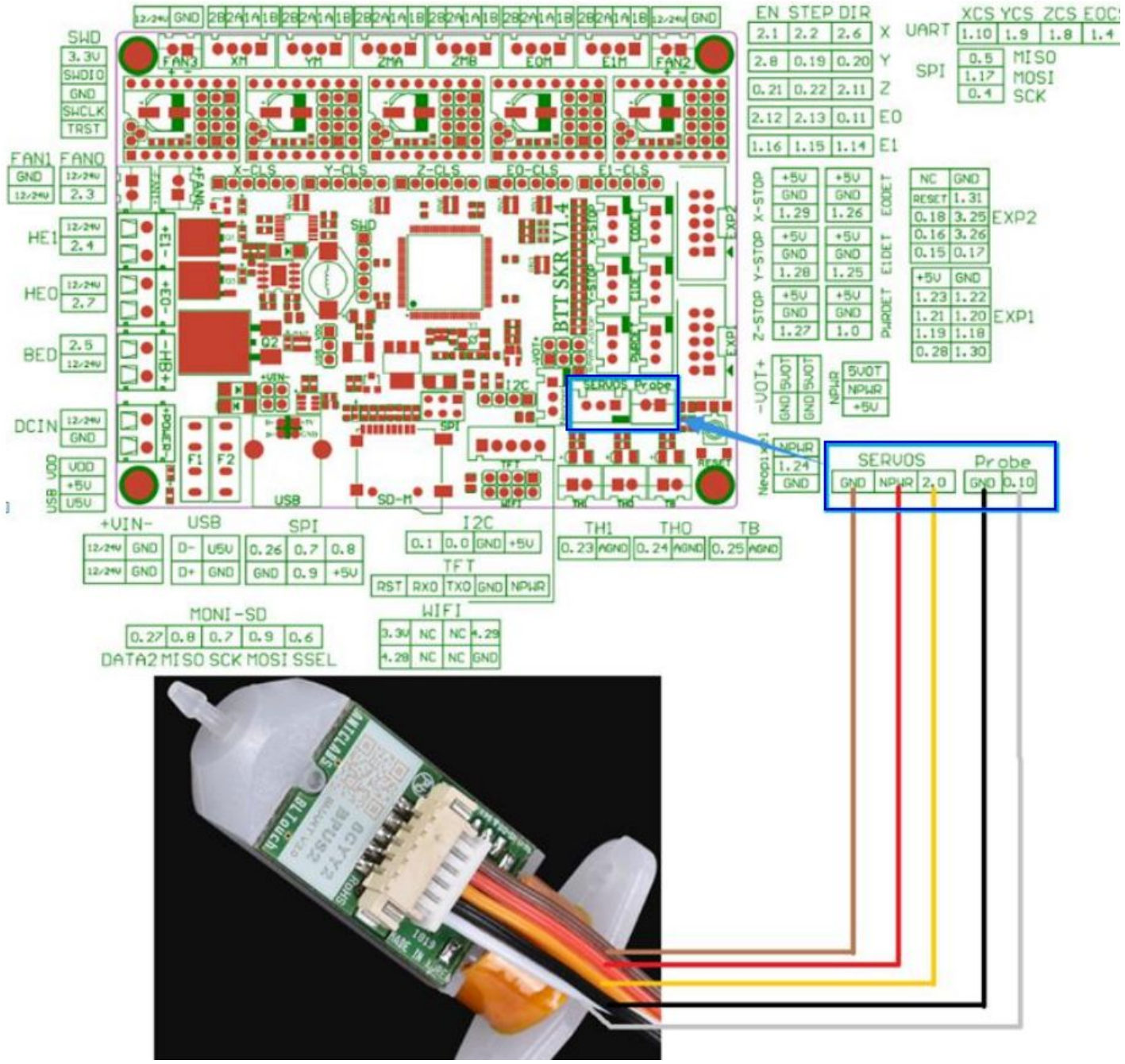


Diagrama de pins Bigtreetech SKR V1.4 / 1.4 Turbo, y Bltouch:



Configuración BL-Touch

La placa SKR V1.3 tiene un Puerto dedicado para servos. **Se debe revisar el cableado antes de usarlo!**

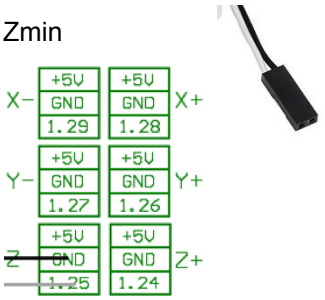
Para conectar un ANTCLABS BL-touch se utilizan 3 cables al Puerto servo.

En la placa SKR v1.3 el puerto servo tiene = (+)(-)(pulso). En cambio el BL-touch es = (-)(+)(puls).
El pin servo es “2.0” (#define SERVO0_PIN P2_00)
Se deben intercambiar el positivo (+) y negativo (-) en el conector Bltouch.
(o bien usar un cable intermedio adicional que realice el cruce).

Si necesitas reordenar los cables, es muy sencillo cambiarlos, con un cutter o pinza, levantando la pestañita negra y sacando el conector metálico:

En el BLTouch original (puede cambiar en los clones, o 3Dtouch chinos) los cables son:
- ROJO +5V (+)
- MARRÓN GND, masa. (-)
- NARANJA señal,(pulso)

El conector de 2 cables se conecta al puerto ENDSTOP de Zmin (endstop Z-, entre pin 1_25 y masa)



Para activar en Marlin BL-Touch, deberemos realizar los siguientes cambios en el fichero **Configuration.h**

Cambiaremos las siguientes líneas:

//#define BLTOUCH cambiar a
#define BLTOUCH

//#define AUTO_BED_LEVELING_BILINEAR cambiar a
#define AUTO_BED_LEVELING_BILINEAR

// #define Z_SAFE_HOMING cambiar a
#define Z_SAFE_HOMING

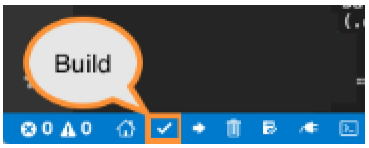
Problemas conocidos Bltouch: Si tu Bltouch original, TL-Touch, o clon baja el pincho durante la impresión, vete al fichero Marlin/scr/feature/**bltouch.h** ay cambia el siguiente código:

#define BLTOUCH_STOW   100   // por defecto es 90
#define BLTOUCH_SELFTEST 130   // por defecto es 120

Compilación

Una vez terminadas las modificaciones en la config de Marlin, compilaremos.
(Ante cualquier duda consultad la **biblia de marlin https://www.staticboards.es/blog/marlin-instalacion-configuracion/** de StaticBoards o la guía de marlin 2 desde cero: https://3dwork.io/configurar-marlin-2-0-x-desde-cero/)

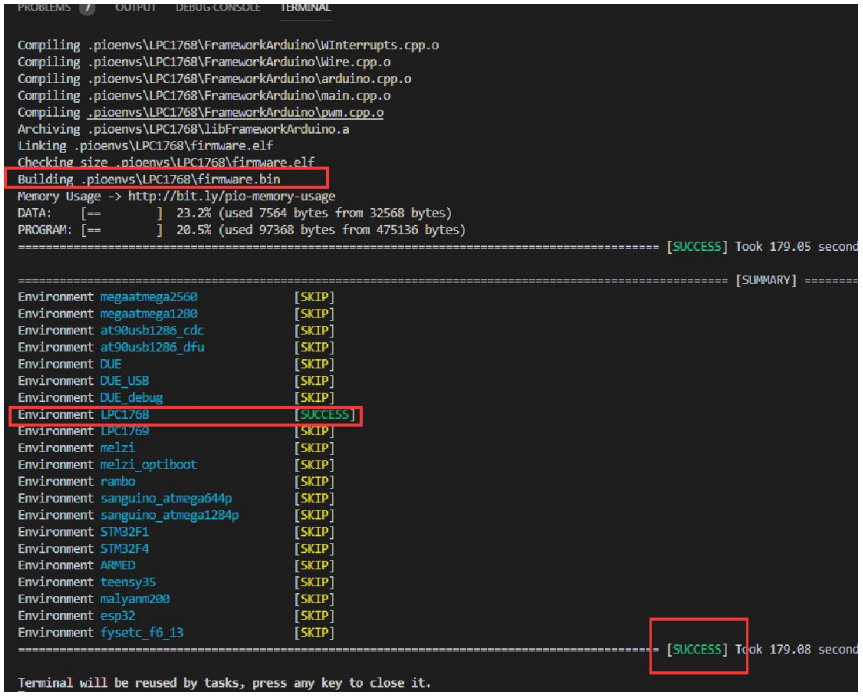
Para comenzar a compilar pulsaremos en VScode las teclas **Ctrl+Mayúsculas+B**, o bien click en el icono BUILD.



PlatformIO comenzará automáticamente a descargar las librerías que necesite, y compilar los componentes.

Puede costarle hasta 5 minutos, tiempo más que de sobra para ir a mear ☺… o mirar el correo, o bien saludar

a la gente del canal **SKR V1.3 32bits** de Telegram (https://t.me/SKR_board_32bits)



SUCCESS!! :
Tras finalizar la compilación con éxito (**SUCCESS**), se habrá generado un fichero "**firmware.bin**" dentro de la carpeta **\.pio\build\LPC1768** firmware.bin

(…Ruta de marlín en tu PC\ Marlin-2.0.0\.pio\build\LPC1768\firmware.bin)

Copia ese fichero firmware.bin en la tarjeta microSD, ponla en la placa, y resetea o reiníciala.
Durante el inicio, la placa grabará ese firmware en la memoria interna de la placa SKR. El fichero una vez subido a la placa se renombra automáticamente en la SD a "firmware.cur" (de Current, actual…)

NO ES NECESARIO que esté la SD insertada, el firmware está ya almacenado en la propia memoria interna de la placa.
Es aconsejable renombrar y guardar como backup tu actual firmware.cur, si tu actual marlin 2.0 compilado estaba funcionando OK. Este podrá volver a flashearse, si lo dejamos de nuevo renombrado como firmware.bin

Esa misma tarjeta puede utilizarse para almacenar ficheros Gcode e imprimir desde ellos (si la has configurado como ONBOARD en la sección SD)

NO SUCCESS!! 😭 Si la compilación hubiera dado errores. Revisa en la misma ventana negra de la consola más arriba, en busca **del primer error (en rojo).**
La mayoría de errores son muy descriptivos, el propio texto del error te indica en qué has fallado. Cosas tipo a "Activar XXX requiere que pongas fulanito=TRUE". Puedes buscar en san google, busca otra vez más, y ya por fin.. busca otra vez… y después pregunta en telegram a alguien en busca de ayuda.

Nota: Durante la compilación aparecen **alertas** (**en amarillo**), pero estas pueden ignorarse en el 99% de los casos.
Son los **errores** (**en rojo**) los que harán que la compilación no termine con éxito.

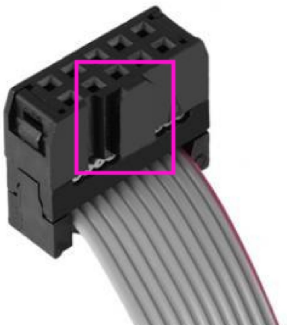
LCD. Problemas

Si tu LCD 128x64 pixeles (REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER) no está funcionando, revisa lo siguiente:

-Si la retroiluminación (luz de fondo) está encendida, pero no se muestran caracteres, has elegido un tipo de pantalla incorrecto en la configuración del firmware en el fichero configuration.h. Cámbialo.

-Si la pantalla permanece apagada, quizás el cable EXP1 esté puesto en el conector EXP2.
Conecte el cable EXP1 en el conector EXP1 de la placa.

-Si la pantalla permanece apagada estando EXP1 bien conectado en su lugar (exp1),
deberemos quitar la pestaña de bloqueo del conector plástico, para poder conectarlo girado 180 grados (es decir, del revés). Esta pestaña puede cortarse con un cutter, o alicates, de forma sencilla.



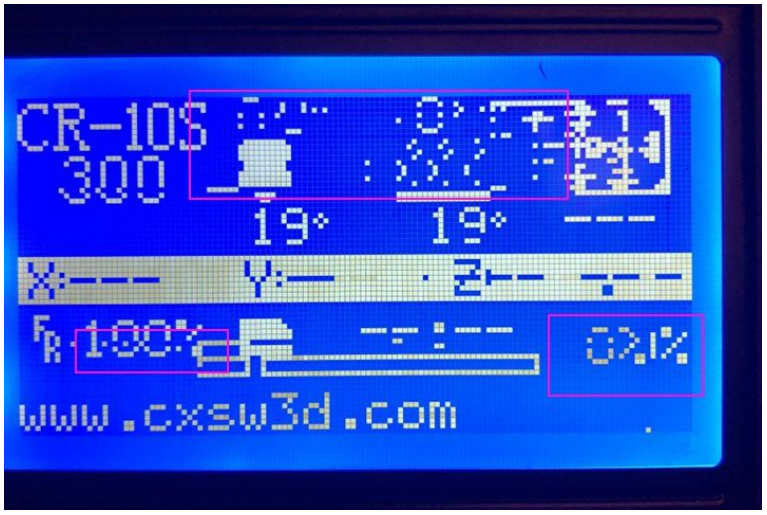
Una vez cortado en ambos EXP1 y EXP2, los conectaremos girados.
NOTA: Esta operación puede realizarse sin miedo, **NO es posible romper el LCD o la placa** por conectarlos del revés.

-Problemas conocidos: Si una vez instalada la pantalla, la ruedecita se comporta raro, podremos cambiar en el fichero Configuration.h lo siguiente:

Descomentar
#define ENCODER_PULSES_PER_STEP 4
y
#define REVERSE_ENCODER_DIRECTION

Caracteres raros: Si la pantalla muestra caracteres raros, puede ser debido a unos retardos,

que de serie a veces dan problemas:



En ese caso, añadiremos al final del configuration.h:

#define ST7920_DELAY_1 DELAY_NS(50)

#define ST7920_DELAY_2 DELAY_NS(188)

#define ST7920_DELAY_3 DELAY_NS(50)

CAPÍTULO Nº827: LOS DRIVERS / JUMPERS

Dado que tienes una placa con soporte de drivers controlados por software (Drivers Trinamics TMC con Uart o Spi), lo óptimo es utilizar drivers TMC, configurados en modo software. De esta manera se pueden gestionar mediante Gcode, o desde el LCD. Si vas a utilizar sólo drivers clásicos que no tienen control por software: **Pololu Stepstick A4988, DRV8825, LV8729** o bien vas a utilizar los drivers TMC, pero en modo clásico “pinchar, ajustar potenciómetro y ya”, puedes saltarte toda esta parte, e ir directo AQUÍ : DRIVERS STANDALONE - JUMPERS

Ajuste de jumpers de driver para uso en modo SPI / UART

En el modo spi/uart la electrónica se comunica con el driver utilizando un puerto serie (puerto SPI o puerto UART).

Este puerto serie puede ser de tipo UART (en el caso de los TMC2208, TMC2209) o de tipo SPI (en el caso de los TMC2130, TMC5160, TMC5161, etc).

El modo spi/uart es más eficiente, y podremos configurar todos los parámetros del driver, así como su amperaje, desde marlin o desde la pantalla LCD.

MODO SPI (TMC2130, TMC5160)

JUMPERS SPI: Siguiendo las instrucciones de Bigtreetech, para poner drivers TMC2130 o TMC5160 en modo SPI, deberemos utilizar jumpers en los 4 zócalos ROJOS, tal que así:

Aparte de los jumpers de la placa el propio driver deberá estar preparado/configurado para SPI.
Si nuestro driver no está preparado para SPI, **deberemos SOLDARLO. (ver guía soldar drivers aquí)**

Hay fabricantes (aliexpress) que venden el mismo driver preparado ya para SPI, o sin preparar (DIT, Do it yourself, sin soldar).

Configuración SPI en marlin

Para indicarle a Marlin que vamos a utilizar drivers TMC2130 ó TMC5160, debemos definirlos en el fichero **configuration.h,**

descomentaremos las líneas X_driver_type, Y, Z, y extrusor E0.

En la imagen del ejemplo le estamos diciendo a Marlin que vamos a utilizar drivers TMC2130 en X,Y, Z y Extrusor.

'**TMC2130'** **equivale a "TMC2130 en SPI".**
'TMC2130_standalone' equivale a "TMC2130 en modo clásico step/dir", con jumpers, Vref potenciómetro y sin control por soft.

```
/**
 * Stepper Drivers
 *
 * These settings allow Marlin to tune stepper driver timing and enable adva
 * stepper drivers that support them. You may also override timing options i
 *
 * A4988 is assumed for unspecified drivers.
 *
 * Options: A4988, A5984, DRV8825, LV8729, L6470, L6474, POWERSTEP01,
 *          TB6560, TB6600, TMC2100,
 *          TMC2130, TMC2130_STANDALONE, TMC2160, TMC2160_STANDALONE,
 *          TMC2208, TMC2208_STANDALONE, TMC2209, TMC2209_STANDALONE,
 *          TMC26X,  TMC26X_STANDALONE,  TMC2660, TMC2660_STANDALONE,
 *          TMC5130, TMC5130_STANDALONE, TMC5160, TMC5160_STANDALONE
 * :['A4988', 'A5984', 'DRV8825', 'LV8729', 'L6470', 'L6474', 'POWERSTEP01',
 */
#define X_DRIVER_TYPE  TMC2130
#define Y_DRIVER_TYPE  TMC2130
#define Z_DRIVER_TYPE  TMC2130
//#define X2_DRIVER_TYPE A4988
//#define Y2_DRIVER_TYPE A4988
//#define Z2_DRIVER_TYPE A4988
//#define Z3_DRIVER_TYPE A4988
//#define Z4_DRIVER_TYPE A4988
#define E0_DRIVER_TYPE TMC2130
//#define E1_DRIVER_TYPE A4988
//#define E2_DRIVER_TYPE A4988
```

Además del fichero configuration.h, deberemos editar en el fichero **configuration_adv.h:**

#define INTERPOLATE true // Lo dejamos activo, en True, así el driver interpola pasos hasta 256 él solo.

#define X_CURRENT 850
#define Y_CURRENT 850
#define Z_CURRENT 850
#define E0_CURRENT 600
Estos valores son los de serie. **pondremos el valor que queremos en MILIAMPERIOS** para cada motor. 600mA = 0,6Amperios. Dependerá del amperaje de nuestros motores, de la impresora,etc

#define X_MICROSTEPS , Y_MICROSTEPS Z_MICROSTEPS E0_MICROSTEPS etc: Conviene dejarlos en **16**

Descomentamos **#define TMC_USE_SW_SPI**

dejamos #define CHOPPER_TIMING CHOPPER_DEFAULT_**24V**
(o bien lo cambiamos a #define CHOPPER_TIMING CHOPPER_DEFAULT_**12V**) según el voltaje de nuestra fuente.

Descomentamos **#define MONITOR_DRIVER_STATUS** //para activar las protecciones de temperatura del driver, entre otras cosas.

Descomentamos **#define SQUARE_WAVE_STEPPING** **//** descomentamos para tener una señal cuadrada más precisa.

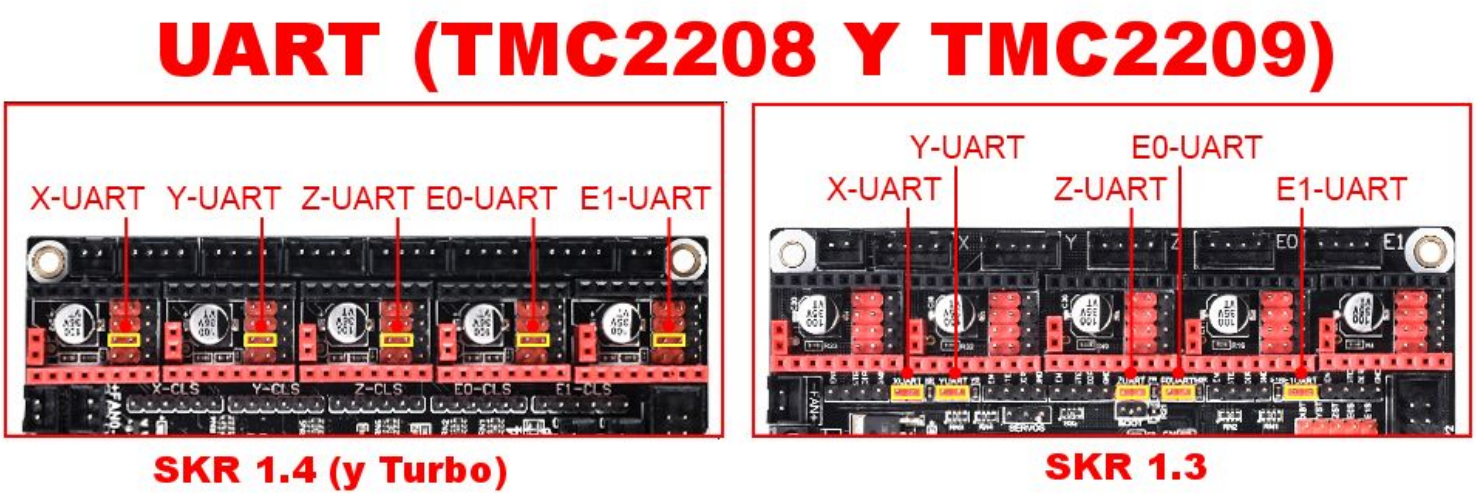
Descomentamos **#define TMC_DEBUG** // para que al lanzar un comando M122 (status de los TMC) nos devuelva mucha más información útil.

Nota: Con drivers TMC5160 deberemos cambiar R_SENSE de ejes X, Y, Z de 0.11 a 0.075. Con TMC2130 no se toca, dejamos todos ellos a 0.11

Con esto, ya tenemos Marlin configurado para utilizar drivers tmc2130 y/o tmc5160 en modo inteligente, comunicacion puerto serie de tipo SPI.

MODO UART (TMC2208, TMC2209)

JUMPERS UART: Siguiendo las indicaciones del fabricante, para poner drivers en UART, deberemos quitar los 4 jumpers bajo el driver en la SKR 1.3 y poner el jumper Uart de cada uno. En la SKR 1.4, y SKR E3 DIP, quitaremos 3 de los 4 jumpers según la imagen:



Las placas SKR Mini E3 no requieren jumpers para UART, vienen ya preparadas de serie puesto que el driver está soldado en la placa.

Aparte de los jumpers de la placa el propio driver deberá estar preparado/configurado para UART.

Si nuestro driver no está preparado para UART, **deberemos SOLDARLO. (ver guía soldar drivers aquí)**
Hay fabricantes (aliexpress) que venden el mismo driver preparado ya para uart, o sin preparar (DIT, Do it yourself, sin soldar).

Configuración UART en marlin

Para indicarle a Marlin que vamos a utilizar drivers TMC2208 / TMC2209, debemos definirlos en el fichero **configuration.h,**

descomentaremos las líneas X_driver_type, Y, Z, y extrusor E0.

En la imagen del ejemplo le estamos diciendo a Marlin que vamos a utilizar drivers TMC2209 en X,Y, Z, Z2 y Extrusor.

'**TMC2209' equivale a "TMC2209 en SPI".**
'TMC2209_standalone' equivale a "TMC2209 en modo clásico step/dir", con jumpers, Vref potenciómetro y sin control por soft.

```
664     *
665     * Options: A4988, A5984, DRV8825, LV8729, L6470, L6474, POWERSTEP01,
666     *          TB6560, TB6600, TMC2100,
667     *          TMC2130, TMC2130_STANDALONE, TMC2160, TMC2160_STANDALONE,
668     *          TMC2208, TMC2208_STANDALONE, TMC2209, TMC2209_STANDALONE,
669     *          TMC26X,  TMC26X_STANDALONE,  TMC2660, TMC2660_STANDALONE,
670     *          TMC5130, TMC5130_STANDALONE, TMC5160, TMC5160_STANDALONE
671     * :['A4988', 'A5984', 'DRV8825', 'LV8729', 'L6470', 'L6474', 'POWERSTEPC
672     */
673    #define X_DRIVER_TYPE  TMC2209
674    #define Y_DRIVER_TYPE  TMC2209
675    #define Z_DRIVER_TYPE  TMC2209
676    //#define X2_DRIVER_TYPE A4988
677    //#define Y2_DRIVER_TYPE A4988
678    #define Z2_DRIVER_TYPE TMC2209
679    //#define Z3_DRIVER_TYPE A4988
680    //#define Z4_DRIVER_TYPE A4988
681    #define E0_DRIVER_TYPE TMC2209
682    //#define E1_DRIVER_TYPE A4988
683    //#define E2_DRIVER_TYPE A4988
684    //#define E3_DRIVER_TYPE A4988
685    //#define E4_DRIVER_TYPE A4988
686    //#define E5_DRIVER_TYPE A4988
687    //#define E6_DRIVER_TYPE A4988
688    //#define E7_DRIVER_TYPE A4988
```

Además del fichero configuration.h, deberemos editar en el fichero **configuration_adv.h:**

#define INTERPOLATE true // Lo dejamos activo, en True, así el driver interpola pasos hasta 256 él solo.

#define X_CURRENT 850
#define Y_CURRENT 850
#define Z_CURRENT 850
#define E0_CURRENT 600
Estos valores son los de serie. **pondremos el valor que queremos en MILIAMPERIOS** para cada motor. 600mA = 0,6Amperios. Dependerá del amperaje de nuestros motores, de la impresora,etc

#define X_MICROSTEPS , Y_MICROSTEPS Z_MICROSTEPS E0_MICROSTEPS etc: Conviene dejarlos en **16**

dejamos #define CHOPPER_TIMING CHOPPER_DEFAULT_**24V**
(o bien lo cambiamos a #define CHOPPER_TIMING CHOPPER_DEFAULT_**12V**) según el voltaje de nuestra fuente.

Descomentamos **#define MONITOR_DRIVER_STATUS** //para activar las protecciones de temperatura del driver, entre otras cosas.

Descomentamos **#define SQUARE_WAVE_STEPPING** **//** descomentamos para tener una señal cuadrada más precisa.

Descomentamos **#define TMC_DEBUG** // para que al lanzar un comando M122 (status de los TMC) nos devuelva mucha más información útil.

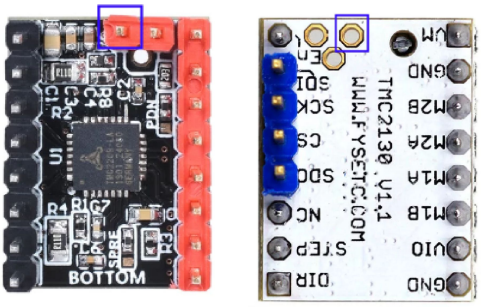
Con esto, ya tenemos Marlin configurado para utilizar drivers TMC2208 y/o TMC2209 en modo inteligente, comunicacion puerto serie de tipo UART.

SENSORLESS_HOMING (STALLGUARD) - Modo sin Endstops.

En el modo sin endstops, el propio driver TMC detecta (utilizando Stallguard) que algo está frenando al motor (al haber llegado éste al final de carrera), y esa detección se envía como señal al pin de Endstop.
Cuando algo frena el motor, el driver Tmc envía por el pin Diag0 una señal. Esta señal, llega al pin de ENDSTOP.

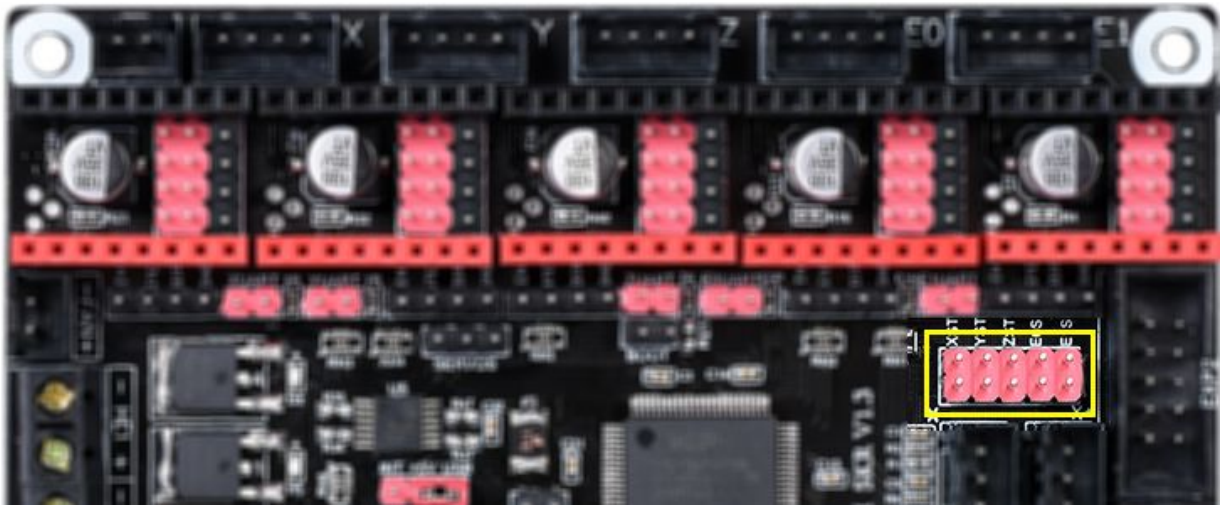
Requiere utilizar drivers TMC que tengan la función STALLGUARD. Es decir, drivers TMC2130, TMC2209 ó TMC516x.
Los drivers **TMC2208 no son compatibles con esta funcionalidad**.

Estos drivers además deberán tener soldado el pin DIAG0 (azul), para que conecte con el de la placa SKR.

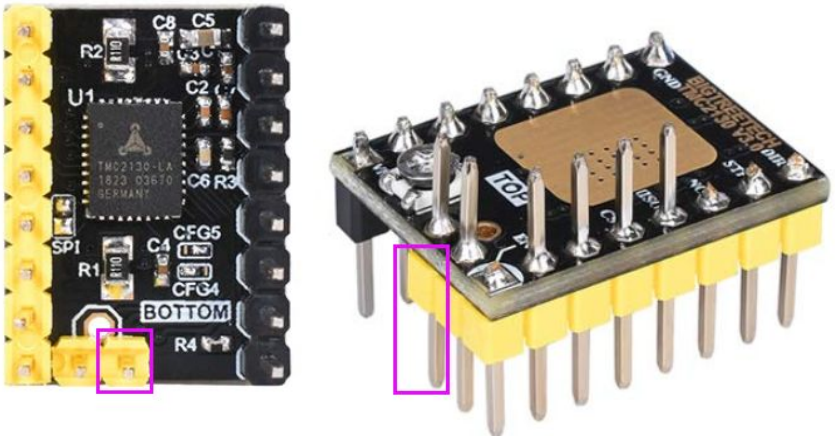


Modo CON endstops:

En **SKR1.3**: Quitaremos los 5 jumpers rojos Xdiag Ydiag Zdiag etc, que están bajo E1. No hace falta hacer nada más.



En **SKR1.4 y turbo**: Dado que la placa no tiene jumpers para evitar que la señal diag llegue al pin endstop, deberemos QUITAR o DOBLAR el pin diag de cada driver TMC2209 (en caso de que el driver lo tenga):

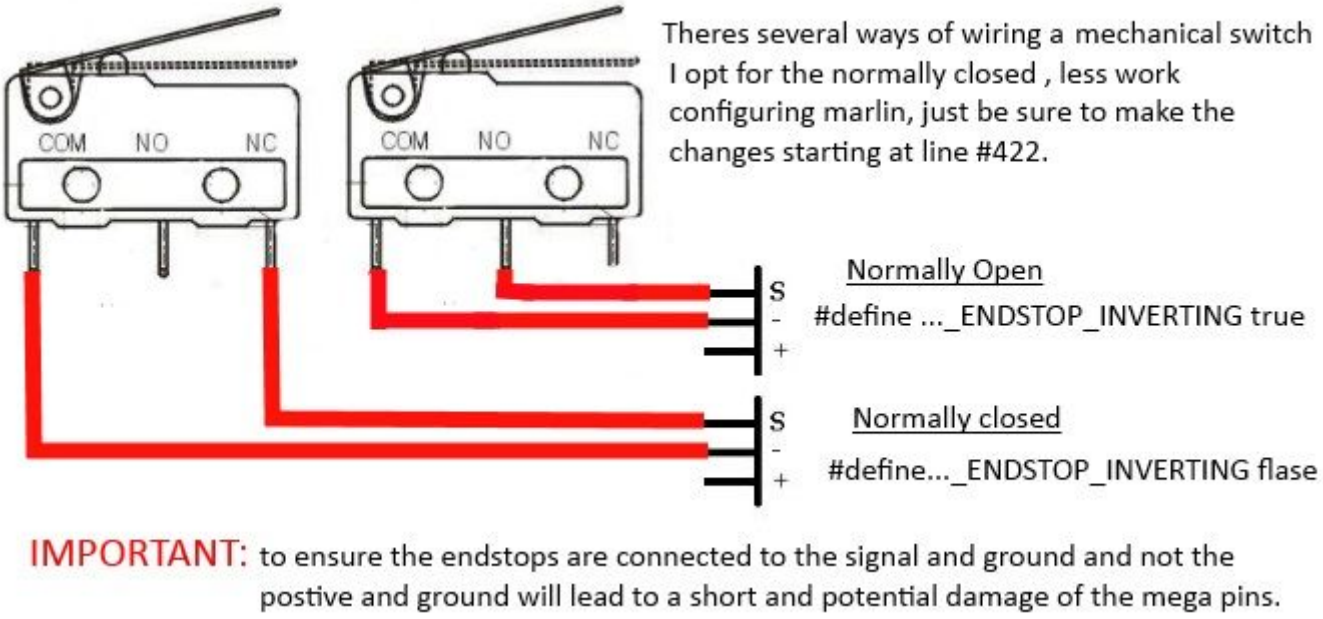


Si el driver tiene este pin, y no vamos a usar sensorless (es decir, vamos a usar endstops normales) deberemos

quitarlo o doblarlo. De esta forma evitamos falsos positivos y que nuestros endstops normales se vean afectados, haciendo los motores cosas raras como pararse sin venir a cuento.

Para el endstop clásico se utilizan dos pins. Signal y gnd. el tercer pin, es +5v para aquellos endstop que requieran corriente (Leds, etc).

#define X_MIN_ENDSTOP_INVERTING false (o true)
#define Y_MIN_ENDSTOP_INVERTING false (o true)



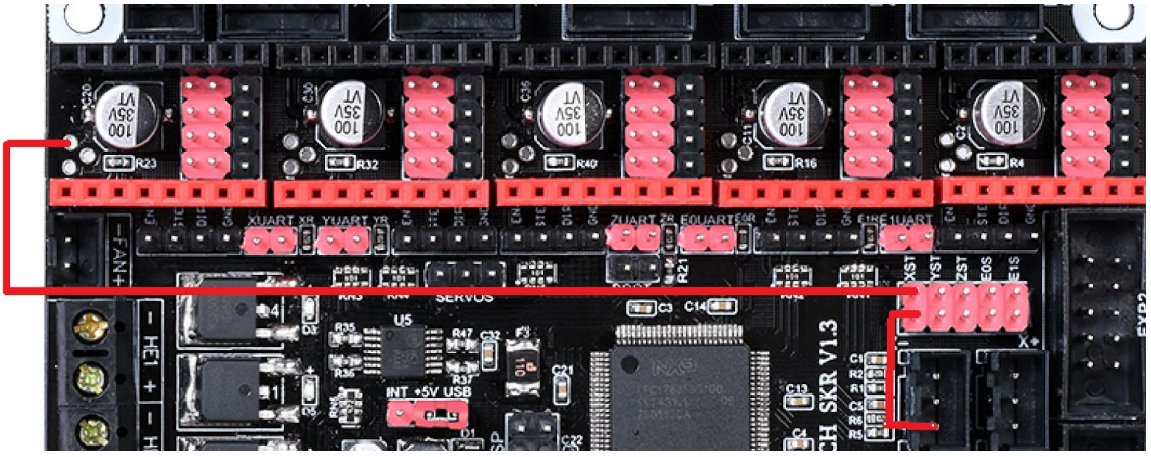
Para comprobar el estado de un endstop enviaremos el gcode: **M119**

Este nos devolverá si el endstop está pulsado (Triggered) o sin pulsar (Open). Si se comporta al revés de lo deseado, cambiaremos true por false (o false por true), según proceda.

Modo SIN endstops:

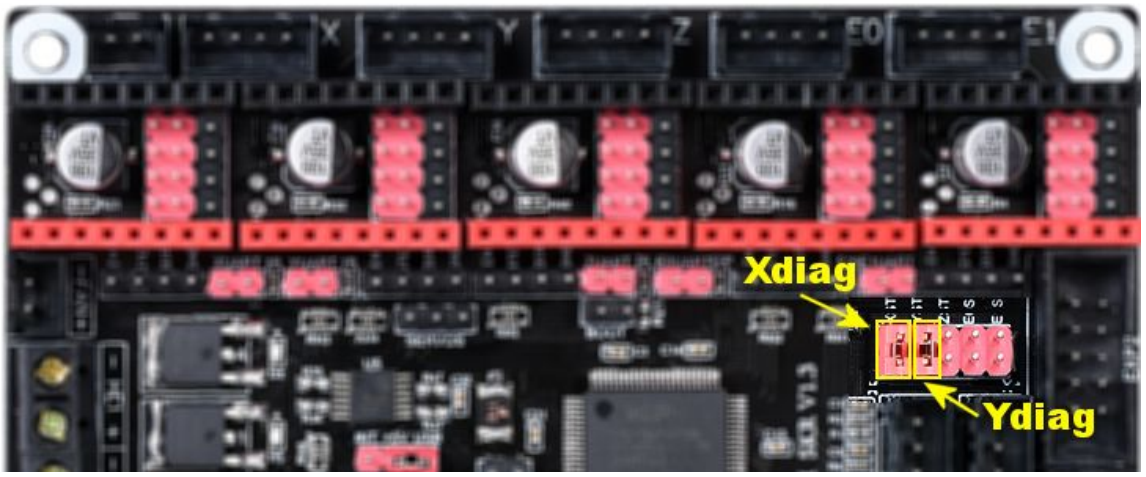
Si queremos usar modo sin Endstops (sensorless, stallguard), deberemos utilizar jumpers en XDiag, YDiag.
No es recomendable utilizar el modo sin endstops en Z.

Esto, internamente, está conectado así. Cuando el driver detecta que el motor está frenado, envía por el pin DIAG una señal hacia el endstop. Se requiere que el Jumper esté puesto para que esta señal llegue.



Si vamos a usar modo sin endstops en los ejes X e Y, pondremos estos dos jumpers.

NO ES RECOMENDABLE utilizar sensorless en el eje Z.
(puesto que hace falta mucha más fuerza para frenar un motor con relación 1:1, es decir, con husillo)
Es peligroso, y podríamos dañar el hotend y/o la cama caliente.



El modo sin endstops, se habilita en Marlin descomentando en configuration_adv.h

#define SENSORLESS_HOMING

Para habitarlo: Necesitaremos tener ambos dos invertir_endstop en FALSE si el driver es **TMC2209.**
En el fichero configuration.h:
#define X_MIN_ENDSTOP_INVERTING false
#define Y_MIN_ENDSTOP_INVERTING false

O bien en TRUE, si el driver es **TMC2130/TMC2160/TMC2660/TMC5130/TMC5160.**
En el fichero configuration.h:
#define X_MIN_ENDSTOP_INVERTING true
#define Y_MIN_ENDSTOP_INVERTING true

Volviendo de nuevo al fichero configuration_adv.h, pondremos a cero estos dos valores:

#define X_HOME_BUMP_MM 0
#define Y_HOME_BUMP_MM 0
(Es la configuración de rebote del home, que por defecto es 5mm. Es decir, hace home una vez, rebota 5mm y hace un segundo home más suave). Para sensorless debe estar OFF, en cero.
Tras habilitar el modo sensorless, deberemos configurar la sensibilidad de cada eje

En Marlin por defecto (justo debajo, mismo fichero)

#define X_STALL_SENSITIVITY 8
#define Y_STALL_SENSITIVITY 8

Un valor intermedio con el que empezar a probar sería 120
#define X_STALL_SENSITIVITY **120**
y
#define Y_STALL_SENSITIVITY **120**

Podemos ajustar la sensibilidad en caliente con el comando Gcode M914

Los TMC2130 van de lo más duro +63 a lo más sensible -64

Los TMC2239 van de lo más duro 0 a lo más sensible 255

Si lo configuramos muy sensible, el motor ni llegará a moverse, falso positivo, endstop siempre pulsado.
Si lo configuramos muy duro, el motor puede que nunca detecte endstop o al hacerlo pegue golpes fuertes al tope físico.

No hay medidas mágicas, depende de cada motor. Algo intermedio suele estar bien (entre 60 y 150 para un TMC2209 y entre -20 y 20 para un TMC2130).
Si queremos poner, por ejemplo, para un TMC2209 un valor de X de 80, y un valor de Y de 105, usando el Gcode M914 sería:

M914 X80
M914 Y105
M500 (guardar cambios eeprom)
Mas info: http://marlinfw.org/docs/hardware/tmc_drivers.html
Para resto de comprobaciones y configuraciones de drivers TMC ver apartado **DRIVERS TMC**

DRIVERS EN MODO CLÁSICO STEP/DIR (STANDALONE)

Ajuste de jumpers de driver para uso en modo standalone (ni UART ni SPI)

El modo standalone, también es conocido como el modo modo clásico STEP/DIR.

Es el modo de toda la vida (placas Ramps, etc). En modo clásico, los drivers se configuran con jumpers, se pincha el driver, y se le ajusta la tensión Vref, desde el potenciómetro.

Es el único modo que soportan los drivers A4988, DRV8825, LV8729. Si tenemos drivers TMC, podremos elegir ambos dos modos. Dado que nuestra placa soporta los modos SPI/UART (drivers TMC) para controlar los drivers por software con jumpers, sin utilizar un solo cablecillo externo, el modo STEP/DIR no es el más recomendado. Es menos versátil, aunque más sencillo de configurar para algunos usuarios, o marcas-modelo de drivers.

En el modo SPI/DIR, los drivers se configuran con los zócalos para jumpers MS0, MS1 y MS2, siguiendo las tablas del fabricante de cada driver. En este modo sólo se utilizan (poniendo o quitando jumpers) las zonas indicadas en los 3 recuadros ROSA. El resto no se usa.

TMC2100

Los drivers de tipo TMC2100 se controlan mediante 3 posibles estados.

0 = pin conectado a GND/MASA mediante cablecillo.

1 = pin conectado a +V con jumper entre rosa y negro.

OPEN = pins al aire SIN jumper

TMC2100	steps	MS0	MS1	MS2	Interpolado	modo
	Full	0	0	Open	No	Spreadcycle
	1/2	1	0	Open	No	Spreadcycle
	1/4	Open	1	Open	SI 256	Spreadcycle
	1/16	0	1	Open	No	Spreadcycle
	1/4	1	1	Open	No	Spreadcycle
	1/4	Open	1	Open	SI 256	Spreadcycle
	1/16	0	1	Open	SI 256	Spreadcycle
	1/4	1	Open	Open	SI 256	Stealthchop1
	1/16	Open	Open	Open	SI 256	Stealthchop1

TMC2208

Los drivers de tipo TMC2208 se controlan mediante 2 posibles estados:

0 = SIN jumper.

1 = Pin conectado a +V con jumper entre rosa y negro

TMC2208	steps	MS0	MS1	MS2	Interpolation	Mode
	1/2	0	0	0	1/256	Stealthchop2
	¼	0	1	0	1/256	Stealthchop2
	1/8	0	0	0	1/256	Stealthchop2
	1/16	1	1	0	1/256	Stealthchop2

TMC2209

Los drivers de tipo TMC2209 se controlan mediante 2 posibles estados, más un pin aparte SPREAD:

0 = SIN jumper.

1 = Pin conectado a +V con jumper entre rosa y negro

steps	MS0	MS1	Interpolado	
8	0	0	SI 256	
32	0	1	SI 256	Distinto a TMC2208!
64	1	0	SI 256	Distinto a TMC2208!
16	1	1	SI 256	

PIN SPREAD:

GND O ABIERTO □ **STEALTHCHOP**

VCC_IO □ **SPREADCYCLE**

TMC2130

Los driver TMC2130 drivers tienen 3 posibles estados

Los drivers de tipo TMC2130 se controlan mediante 3 posibles estados:

0 = pin conectado a GND/MASA mediante cablecillo.

1 = pin conectado a +V con jumper entre rosa y negro.

OPEN = pins al aire SIN jumper

TMC2130	steps	MS0	MS1	MS2	Interpolation	Mode
	Full	0	0	Open	No	Spreadcycle
	½	1	0	Open	No	Spreadcycle
	½	Open	1	Open	1/256	Spreadcycle
	¼	0	1	Open	No	Spreadcycle
	1/16	1	1	Open	No	Spreadcycle
	¼	Open	1	Open	1/256	Spreadcycle
	1/16	0	Open	Open	1/256	Spreadcycle
	¼	1	Open	Open	1/256	Stealthchop1
	1/16	Open	Open	Open	1/256	Stealthchop1

A4988, DRV8825, LV8729

Los siguientes drivers tienen 2 posibles estados.

0 = SIN jumper entre rosa y negro

1 = Pin conectado a V+ (CON jumper entre rosa y negro)

A4988	steps	MS0	MS1	MS2	Interpolation	Mode
	Full	0	0	0	No	None
	½	1	0	0	No	None
	¼	0	1	0	No	None
	1/8	0	1	0	No	None
	1/16	1	1	1	No	None

DRV8825	steps	MS0	MS1	MS2	Interpolation	Mode
	Full	0	0	0	No	None
	½	1	0	0	No	None
	¼	0	1	0	No	None
	1/8	1	1	0	No	None
	1/16	0	0	1	No	None
	1/32	1	1	1	No	None

LV8729	Steps	MS0	MS1	MS2	Interpolation	Mode
	Full	0	0	0	No	None
	1/2	1	0	0	No	None
	1/4	0	1	0	No	None
	1/8	1	1	0	No	None
	1/16	0	0	1	No	None
	1/32	1	0	1	No	None
	1/64	0	1	1	No	None
	1/128	1	1	1	No	None

Sentido de giro de los motores.

Si tras compilar, subir marlin y probar la impresora alguno de los motores va en dirección contraria, deberemos cambiar para ese eje el valor de INVERT_eje_DIR true por false, o false por true. **En los drivers trinamic, el sentido de giro es al contrario** que resto de drivers (a4988, drv8825, lv8729, etc)

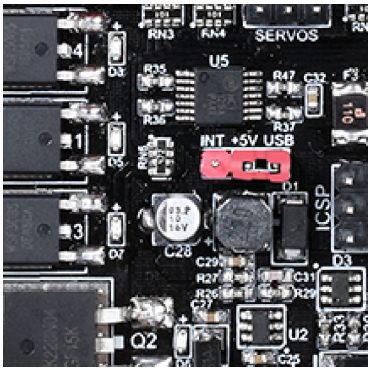
```
#define INVERT_X_DIR false
#define INVERT_Y_DIR false
#define INVERT_Z_DIR false
#define INVERT_E0_DIR false
```

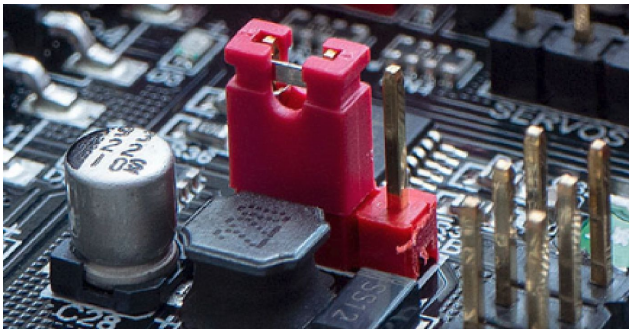
Alimentación USB:

Hay un jumper rojo con el texto INT +5V USB.
Con él elegiremos si la placa recibe voltaje del usb, o de la fuente de alimentación.

Si queremos realizar pruebas (o por ejemplo cargarle un Marlin) sin tener la impresora disponible, podemos cambiar el jumper para alimentarse de 5V del USB (Jumper a la derecha). Una vez realizadas las pruebas, la posición normal del jumper es INT, para que coja tensión de la fuente.

Nota: Los drivers Trinamic TMC dirán “TMC CONECTION ERROR” si alimentamos la placa del USB.
Se requiere alimentar la placa a 12 ó 24v para que los TMC funcionen en Uart/SPI.



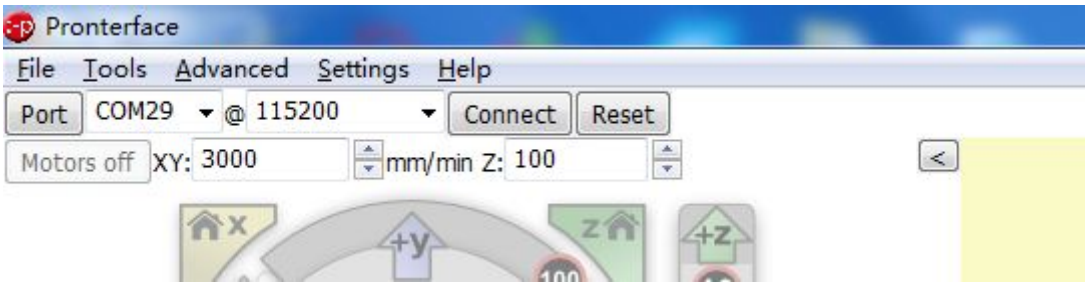


Pruebas / Conexión USB / Status

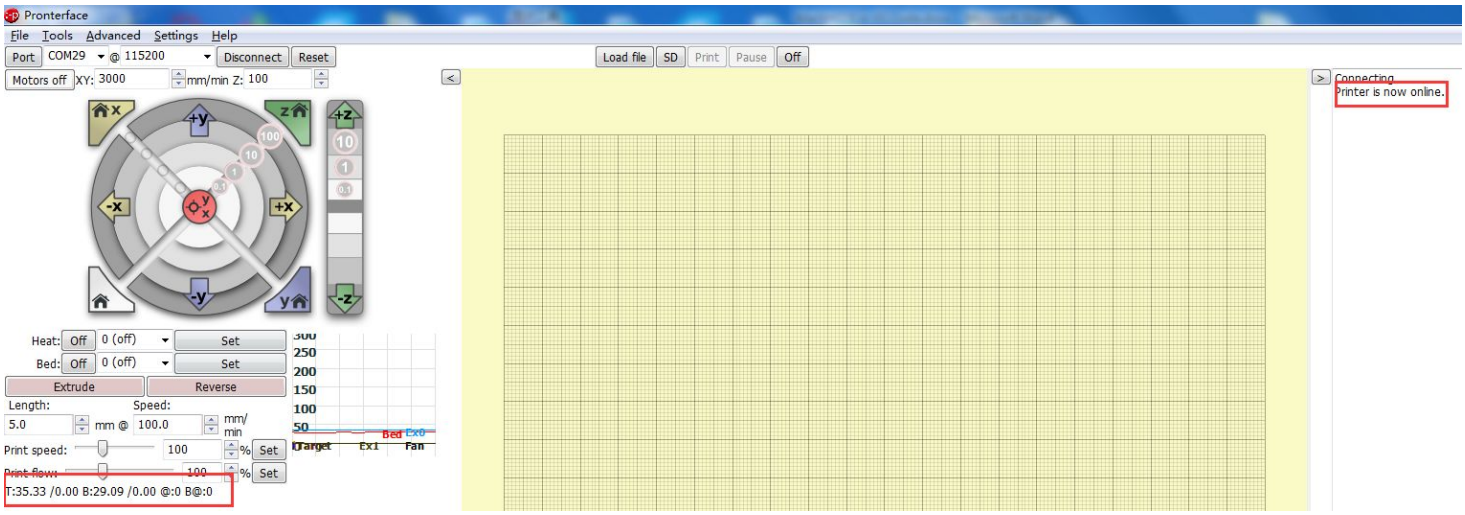
Abre la consola de Printrun/Pronterface o bien la de Repetier Host (por ejemplo), elige tu número de puerto serie y velocidad en baudios (ej 115200), y haz click en Connect.

Simplify también tiene una consola manual parecida, pero es bastante floja.

(El valor de la velocidad en baudios será 115200, o la velocidad que hayas puesto en tu configuration.h , en la variable #define BAUDRATE 115200)

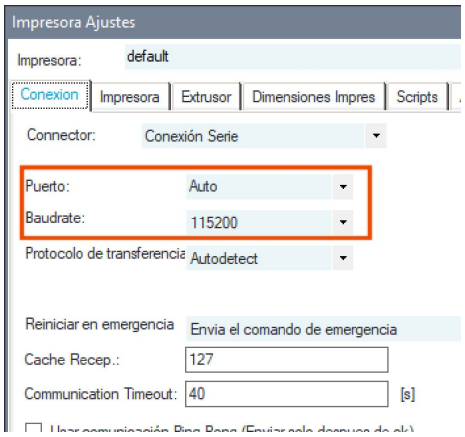


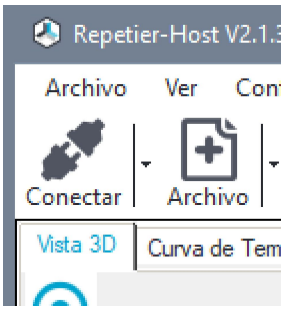
Tu número de puerto posiblemente sea distinto. Una vez le damos a connect, en la ventana amarilla a la derecha, podremos ver si ha conectado OK, mostrando que la impresora está Online.
Esto indicará que ha conectado bien. Ahora puedes utilizar tu PC para controlar la máquina, hacer endstop de cada eje, mover los ejes, o lanzar comandos Gcode.



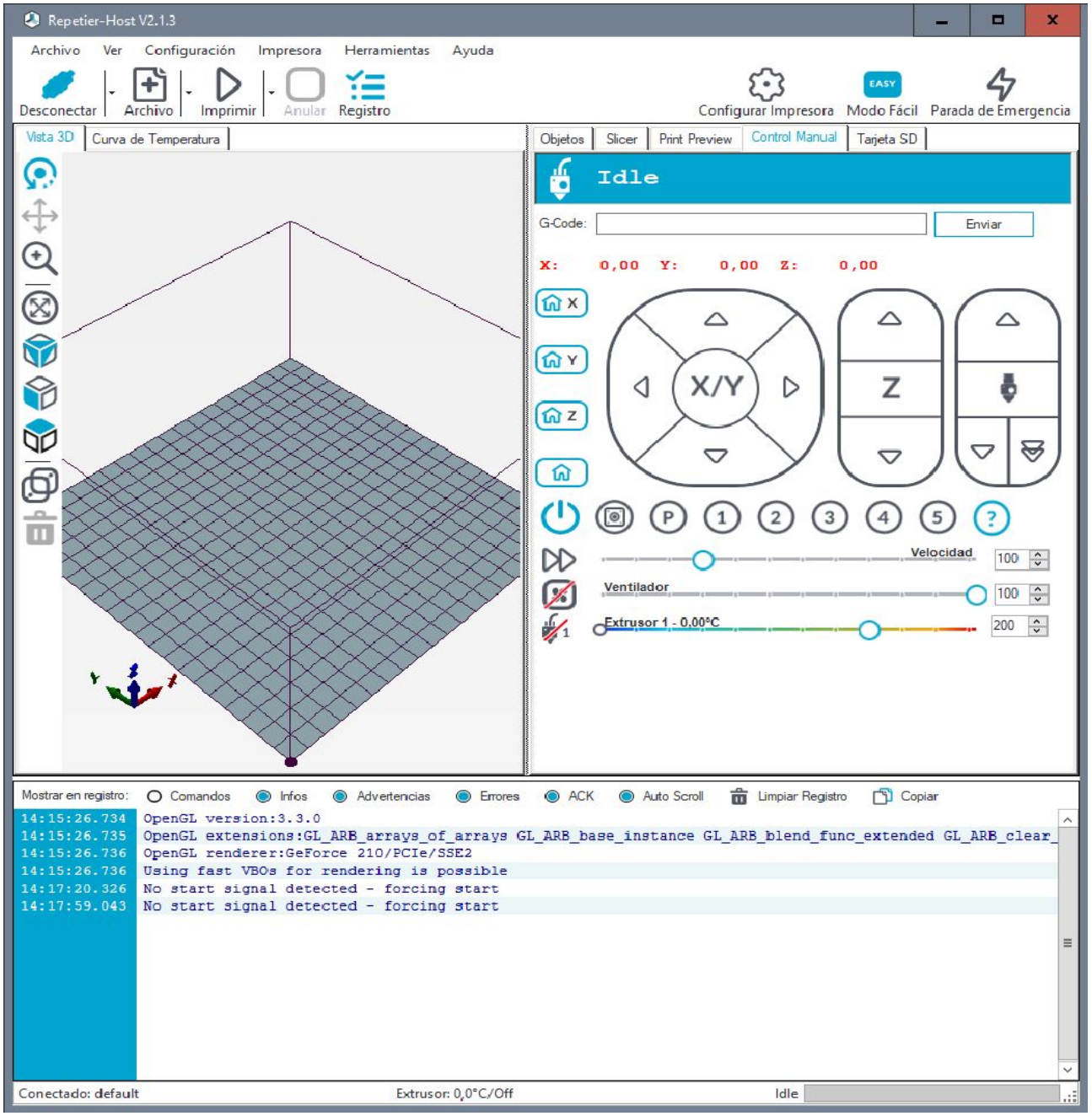
Si en lugar de Proncterface utilizas **Repetier Host**, (Enlace descarga) en configuración / configurar impresora, pon el puerto en AUTO y elige velocidad en baudios.

y tras aceptar, pulsamos en Conectar.





Una vez conectado, podremos utilizar/probar manualmente la impresora desde el control manual, enviar Gcodes sueltos, y ver su respuesta en la parte inferior, como por ejemplo la configuración actual de la máquina tras enviar un Gcode **M503** (mostrar datos eeprom) o **M122** (mostrar estado drivers Trinamics TMC).
Si no apareciera la ventana inferior con el log de texto, pulsaremos sobre el botón Registro.



Drivers TMC

Una vez compilado y subido Marlin a la máquina, si estamos usando drivers TMC, es posible que la pantalla muestre TMC CONNECTION ERROR.

Esto es debido a que Marlin no consigue “hablar” con el driver de forma correcta, bien sea por el puerto serie uart o el spi.

Las causas pueden ser:

- A) El driver no tiene suficiente corriente, está apagado: Si estamos alimentando la placa sólo del USB, se mostrará este error. Los drivers requieren 12 ó 24v de la fuente para funcionar. Asegurarse de que la fuente esté encendida, y de tener el jumper de alimentación en la posición correcta. ver apartado Alimentación USB.
- B) Alguno de los drivers está roto. Ver más adelante estado M122
- C) Hemos cambiado de versión de marlin y la Eeprom tiene datos antiguos, y esto está dando problemas con drivers TMC. Para ello borramos EEPROM:

  OPCIÓN 1: Desde el la propia pantalla:
  -Menú configuración / Rest. Fábrica (que sería como lanzar un M502)
  y después
  -Menú configuración / Guardar EEPROM (que sería como lanzar un M500)
  -Reiniciamos la impresora.
  --
  OPCIÓN2: Conectando a la impresora por USB, y enviamos los comandos gcode:
  M502
  (que borra la eeprom, leyendo la configuración de nuevo desde marlin)
  M500
  (guarda los cambios en la eeprom)
  Reiniciamos la impresora.

Ver estado drivers TMC (M122)

Existe un comando Gcode, M122 que nos muestra el estado de todos los drivers TMC (cuando están en SPI o UART, en standalone NO)

Al lanzar un comando Gcode **M122** recibimos lo siguiente (si muestra menos datos, es porque no está TMC_DEBUG descomentado):

```
X  Y   Z   Z2  E
Address        0   0           0
Enabled        false   false   false   false   false
Set current    850 900 600 600 600
RMS current    826 887 581 581 581
MAX current    1165    1251    819 819 819
Run current    26/31   28/31   18/31   18/31   18/31
Hold current   10/31   11/31   7/31    7/31    7/31
CS actual  10/31   11/31   7/31    7/31    7/31
PWM scale  12  13  9   9   9
vsense     1=.18   1=.18   1=.18   1=.18   1=.18
stealthChop    true    true    true    true    true
msteps     16  16  16  16  16
tstep      max max max max max
PWM thresh.
[mm/s]
OT prewarn false   false   false   false   false
triggered
OTP        false   false   false   false   false
off time   4   4   4   4   4
blank time 24  24  24  24  24
hysteresis
-end       2   2   2   2   2
-start     1   1   1   1   1
Stallguard thrs    110 100         0
DRVSTATUS  X   Y   Z   Z2  E
Driver registers:
X  0xC0:0A:00:00
Y  0xC0:0B:00:00
Z  0xC0:07:00:00
Z2 0xC0:07:00:00
E  0xC0:07:00:00
Testing X connection... OK
Testing Y connection... OK
Testing Z connection... OK
Testing Z2 connection... OK
Testing E connection... OK
```

Es texto en un formato de tabla poco claro, y al principio asusta verlo. Podemos copiarlo y pegarlo en Excel, o verlo como si fuera una tabla excel en nuestra cabeza. Son filas y columnas:

	X	Y	Z	Z2	E
Address	0	0			0
Enabled	false	false	false	false	false
Set current	850	900	600	600	600
RMS current	826	887	581	581	581
MAX current	1165	1251	819	819	819
Run current	26/31	28/31	18/31	18/31	18/31
Hold current	10/31	11/31	7/31	7/31	7/31
CS actual	10/31	11/31	7/31	7/31	7/31
PWM scale	12	13	9	9	9
vsense	1=.18	1=.18	1=.18	1=.18	1=.18
stealthChop	true	true	true	true	true
msteps	16	16	16	16	16
tstep	max	max	max	max	max
OT prewarn Triggered	false	false	false	false	false
OTP	false	false	false	false	false
off time	4	4	4	4	4
blank time					
hysteresis	24	24	24	24	24
-end	2	2	2	2	2
-start	1	1	1	1	1
Stallguard	110	100			0
DRVSTATUS	X	Y	Z	Z2	E
sg_result	0	0			0

Driver registers:	
X	0xC0:0A:00:00
Y	0xC0:0B:00:00
Z	0xC0:07:00:00
Z2	0xC0:07:00:00
E	0xC0:07:00:00
Testing X connection...	**OK**
Testing Y connection...	**OK**
Testing Z connection...	**OK**
Testing Z2 connection...	**OK**
Testing E connection...	**OK**

De toda esa información, mucha no nos es muy útil. En el ejemplo de la foto vemos una columna Z2 por utilizar dos drivers, uno para cada motor de Z.

El final del status, nos muestra el test de conexión de cada EJE.
Si devuelve OK, el uart/spi tiene comunicación correcta con el driver. Si devuelve ALL HIGH, o ALL LOW, ese driver no está comunicando de forma correcta.

El valor **SET CURRENT** indica qué amperaje (en miliamperios) hemos definido en Marlin para ese eje.
El valor **RMS Current** indica cual es el **amperaje actual real**, del driver. Debiera ser muy similar al “Set Current”.
El valor **Stealchop** indica TRUE si ese driver está en modo silencioso, o FALSE si ese driver está en modo Spreadcycle (ruidoso)
El valor **msteps** indica los micropasos del driver.
El valor **OT Prewarn Triggered**: Si está en TRUE, indica que ese driver ha sobrepasado el umbral de temperatura.
Eso, como supondrás es malo. Marlin puede bajarle el amperaje a ese driver de forma automática.
Lo hará aunque el driver esté frío, si no reseteamos ese “flag”. Existe un comando para resetear el valor de avisos temperatura, M912
El valor **Stallguard** indica el ajuste de fuerza (sensibilidad) del modo sin endstops, sensorless (stallguard).
Driver Registers muestra esos mismos datos de cada driver, pero en formato propio hexadecimal (cada bit significa una cosa).

as info: http://marlinfw.org/docs/hardware/tmc_drivers.html

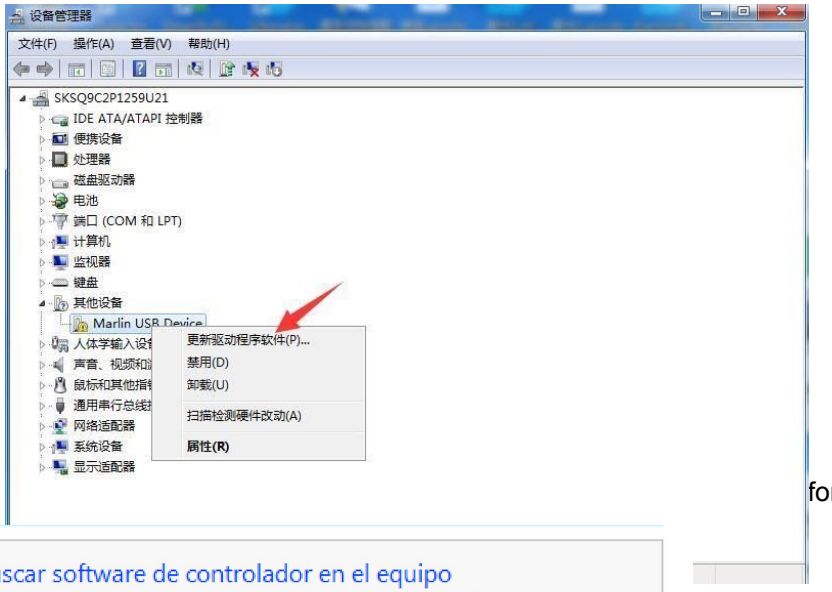
Driver sistema operativo

Debido al Plug & Play, en Windows 10 se instalará de forma automática un driver llamado MARLIN al conectar por USB la placa. Aparecerá un puerto serie MARLIN. No instales ningún otro driver, ni los drivers de Smoothieware si usas Windows 10!

Para el resto de Sistemas operativos Windows, se necesitará instalar manualmente un driver de puerto serie USB , que está disponible en la carpeta debajo indicada.
Como la ruta es muy larga, copia el fichero .INF al escritorio.

Abriremos el administrador de dispositivos (Tecla Windows+Pausa y click admin dispositivos), o bien ejecutar *compmgmt.msc.* y buscaremos el dispositivo con símbolo de error/advertencia.

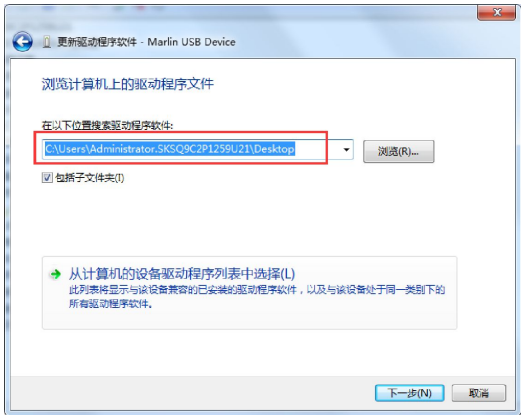
Botón derecho en el dispositivo, click en Actualizar controlador

Buscar de 

forma manual.

Elegiremos la ruta de la carpeta donde tenemos el fichero **lpc176x_usb_driver.inf**.

Si la habíamos copiado en el escritorio estará en C:\Users\(tu usuario)\Desktop\ , y hacemos click en siguiente.



Si aparecen alertas del cortafuegos, haremos click en Instalar Siempre.

Si la instalación del driver finaliza con éxito, el driver ya estará instalado. **Recuerda el número de Puerto.**

WINDOWS 10: SÓLO DRIVERS FIRMADOS WINDOWS

Windows 10: Deshabilitar el uso obligatorio de controladores firmados

¿Cómo puedo instalar controladores que no están firmados digitalmente?

Windows 10 predeterminadamente fuerza el uso obligatorio de controladores firmados. Esto puede desactivarse para instalar controladores que no están firmados digitalmente. Siga los pasos a continuación para deshabilitar el uso obligatorio de controladores firmados.

Forma sencilla: Abrimos un CMD como administrador, y ejecutamos el comando:

```
bcdedit.exe /set nointegritychecks on
```

Tras ello, reinicia el ordenador. Este cambio es permanente.

(Si alguna vez quisieras volver a habilitar el uso de sólo drivers firmados bcdedit.exe /set nointegritychecks off)

Forma "Microsoft" (80 pasos, más incómodo)**:**

Siga los pasos a continuación para deshabilitar el uso obligatorio de controladores firmados:

1. Click en botón de inicio , y configuración
2. Click **Actualización y seguridad**.
3. Click en **Recuperación**. 

4. Click **Reiniciar ahora** en el apartado I**nicio avanzado**.
5. Click **Solucionar problemas**.
6. Click **opciones avanzadas**.
7. Click **Configuración de inicio**.
8. Click on **Reiniciar**.
9. En la pantalla de Configuración de inicio pulse 7 o F7 para deshabilitar el uso obligatorio de controladores firmados.

El equipo se reiniciará y podrá instalar controladores que no tengan firma digital. Si has usado el método Microsoft, al reiniciar el equipo nuevamente se habilitará de nuevo el uso obligatorio de controladores firmados.

Este documento no ha sido creado por BIGTREETECH!

Documento original creado por 2019 Jupa Creations. Netherlands.

Traducción inicial y múltiples añadidos/correcciones posteriores por @lokus77 telegram

Apéndice Nº 1. OTRAS PLACAS

Fichero platformio.ini: Valores de default_envs para otras placas distintas a "SKR V1.3"

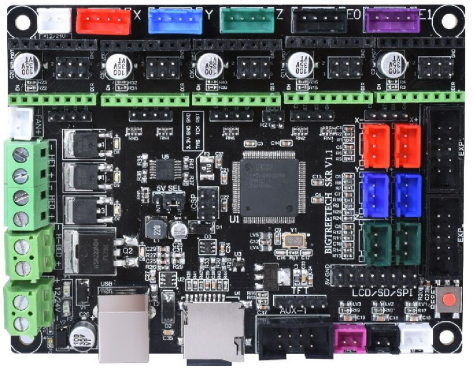
NOTA: En ocasiones cambian los nombres de la variable **default_envs.** Puedes consultar los existentes abriendo con notepad++ tu fichero platformio.ini

La placa SKR V1.3 utiliza 'default_envs = LPC1768' dado que su CPU es una **LPC1768.**

Para otras placas SKR, el 'default_envs' es distinto:

- ARDUINO MEGA AVR, y compatibles: mks gen, gen L, creality melzi, etc, etc, el clásico (y por defecto)

  default_envs = megaatmega2560



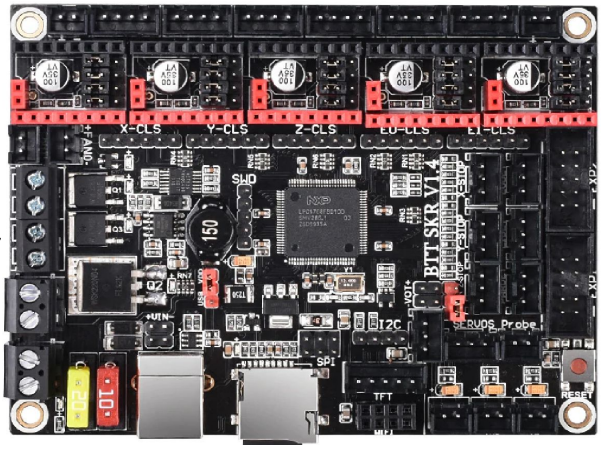
- SKR V1.1 (CPU: LPC1768)

  Versión anterior a la actual SKR V1.3, sin fusibles ni jumpers para UART/SPI. Misma CPU. Por lo demás igual a SKR1.3.

  default_envs = LPC1768

En el fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_SKR_V1_1



- SKR V14 / V1.4 TURBO(CPU: LPC1768/ LPC1769)

  Nueva versión de la 1.3 con alguna mejora, más salidas ventiladores, I2C, salida Led RGB, 2 conectores Z a un driver, Esp01, protección termistor

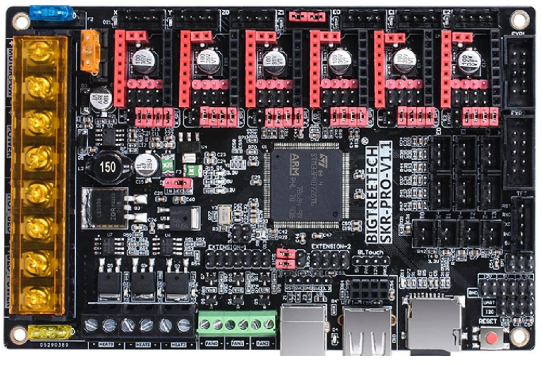
  1.4: **default_envs = LPC1768**

  1.4 Turbo: **default_envs = LPC1769**

En el fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_SKR_V1_4

#define MOTHERBOARD BOARD_BTT_SKR_V1_4_TURBO

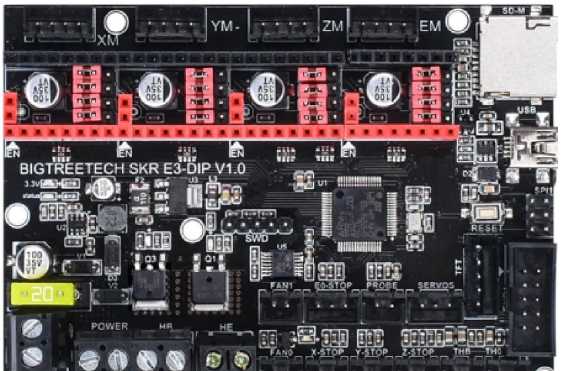


- SKR PRO 1.1 (CPU: STM32F407ZGT6) misma cpu que Lerdge

  default_envs = BIGTREE_SKR_PRO

En fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_SKR_PRO_V1_1



- SKR MINI E3 DIP (CPU: STM32F103 RCT6 / RET6)

  default_envs = STM32F103RC_btt

En el fichero configuration.h:
#define MOTHERBOARD BOARD_BTT_SKR_E3_DIP

rct6 RCT6 / RET6: Ver nota debajo: Enlace

- SKR MINI E3 V1.0 / SKR MINI E3 V1.2 (CPU: STM32F103 RCT6 / RET6)

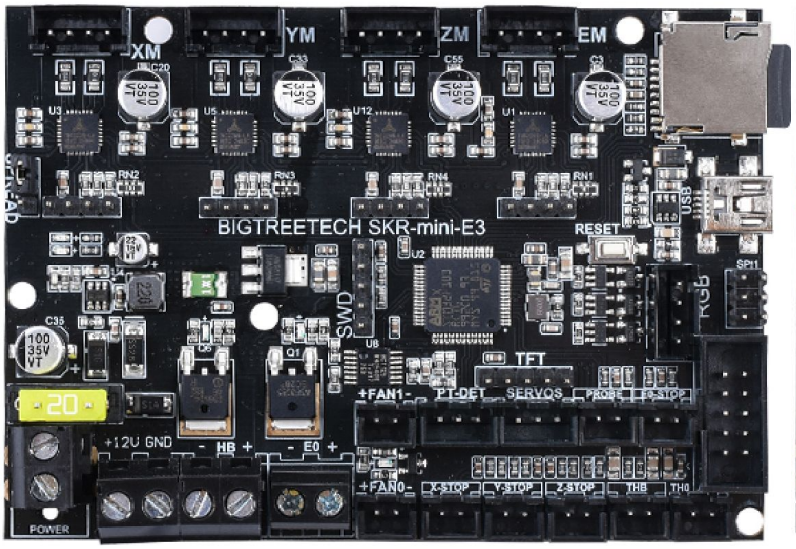
default_envs = STM32F103RC_btt

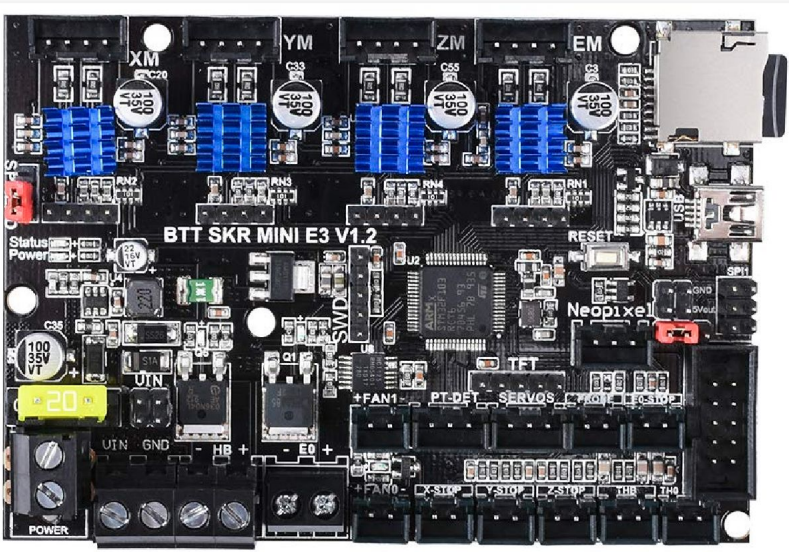
En el fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_SKR_MINI_E3_V1_0

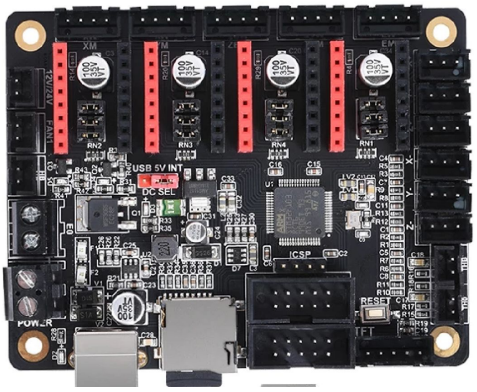
#define MOTHERBOARD BOARD_BTT_SKR_MINI_E3_V1_2

RCT6 / RET6: Ver nota debajo: Enlace





SKR Mini E3 1.0 **SKR Mini E3 1.2**

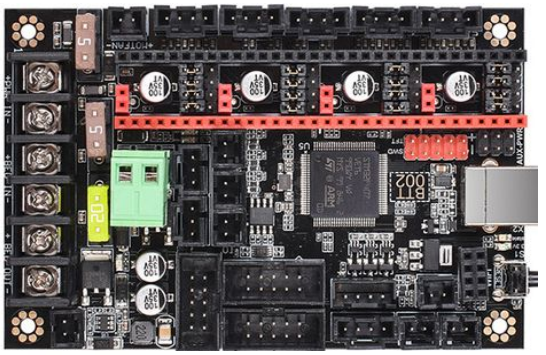


SKR MINI v1.1 (CPU: STM32F103RCT6)

default_envs = STM32F103RC_btt

fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_SKR_MINI_V1_1



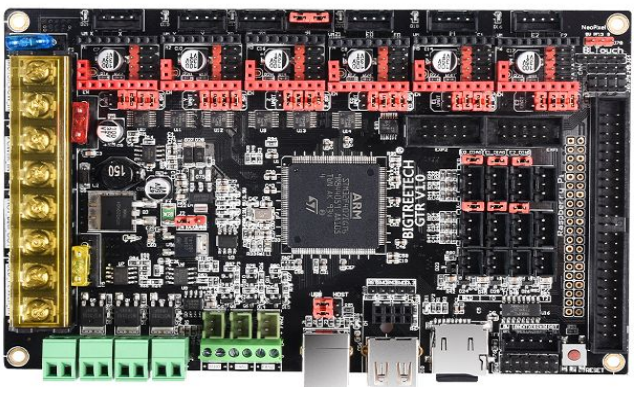
- BigTreeTech BTT002 (CPU: STM32F407VE) = cpu Lerdge X

default_envs = BIGTREE_BTT002

Placa orientada a sustituir a placa de Prusa MK3 original.

fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_BTT002_V1_0



- Bigtreetech GTR V1.0 (STM32F407IGT6

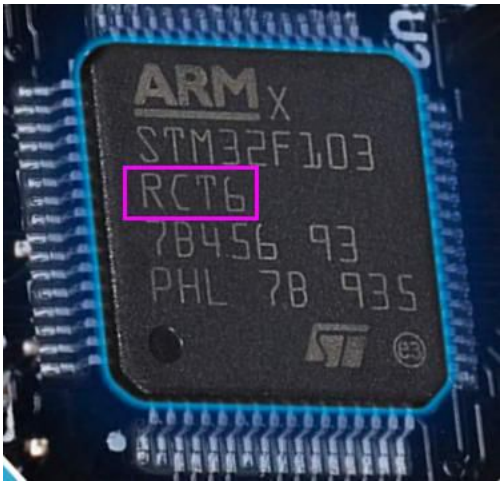
default_envs = BIGTREE_GTR_V1_0

fichero configuration.h:

#define MOTHERBOARD BOARD_BTT_GTR_V1_0

- CPU ATÍPICA - STM32F103 **RCT6** VS STM32F103 *RET6.*

Algunas SKR MINI E3 DIP, SKR MINI E3 1.0 Y SKR MINI E3 1.2 vienen una cpu distinta al STM32F103**RCT6** habitual. Es la cpu STM32F103**RET6**. Ambas son STM32F103, pero a la hora de compilar el entorno cambia.



La RET6 es realmente mejor Cpu que la STM32F103RCT6, porque tiene más ram, y 512kb de flash en lugar de 256k, a veces puede ser más lioso compilar marlin para ella, al no ser típica.

Si nuestra CPU es de tipo RET6, en platform.io deberemos usar:

default_envs = STM32F103RE_btt

STM32F103RET6	
Attribute	Value
Package	LQFP-64_10x10x05P
Manufacturer	STMicroelectronics
Brand Category	International Brands
Core Size	32-Bit
Program Memory Type	FLASH
Core Processor	ARM® Cortex®-M3
Speed	72MHz
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Program Memory Size	**512KB**
RAM Size	**64KB**
EEPROM Size	-
Number of I/O	51
A/D	16x12bit
D/A	2x12bit
PWM	18
LCD	LCD parallel interface, 8080/6800 modes
UART/USART	3 USART+2 UART
SPI	3
I2C/SMBUS	2
USB Device	1
USB Host/OTG	-
CAN	1
Ethernet	0
Features	I2S
Packaging	Tray

STM32F103RCT6	
Attribute	Value
Package	LQFP-64_10x10x05P
Manufacturer	STMicroelectronics
Brand Category	International Brands
Core Size	32-Bit
Program Memory Type	FLASH
Core Processor	ARM® Cortex®-M3
Speed	72MHz
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Program Memory Size	**256KB**
RAM Size	**48KB**
EEPROM Size	-
Number of I/O	51
A/D	16x12bit
D/A	2x12bit
PWM	18
LCD	LCD parallel interface, 8080/6800 modes
UART/USART	3 USART+2 UART
SPI	3
I2C/SMBUS	2
USB Device	1
USB Host/OTG	-
CAN	1
Ethernet	0
Features	I2S
Packaging	Tray

TFT Bigtreetech

Para conectarse en modo TFT táctil a la placa, no es necesario hacer nada en marlin.

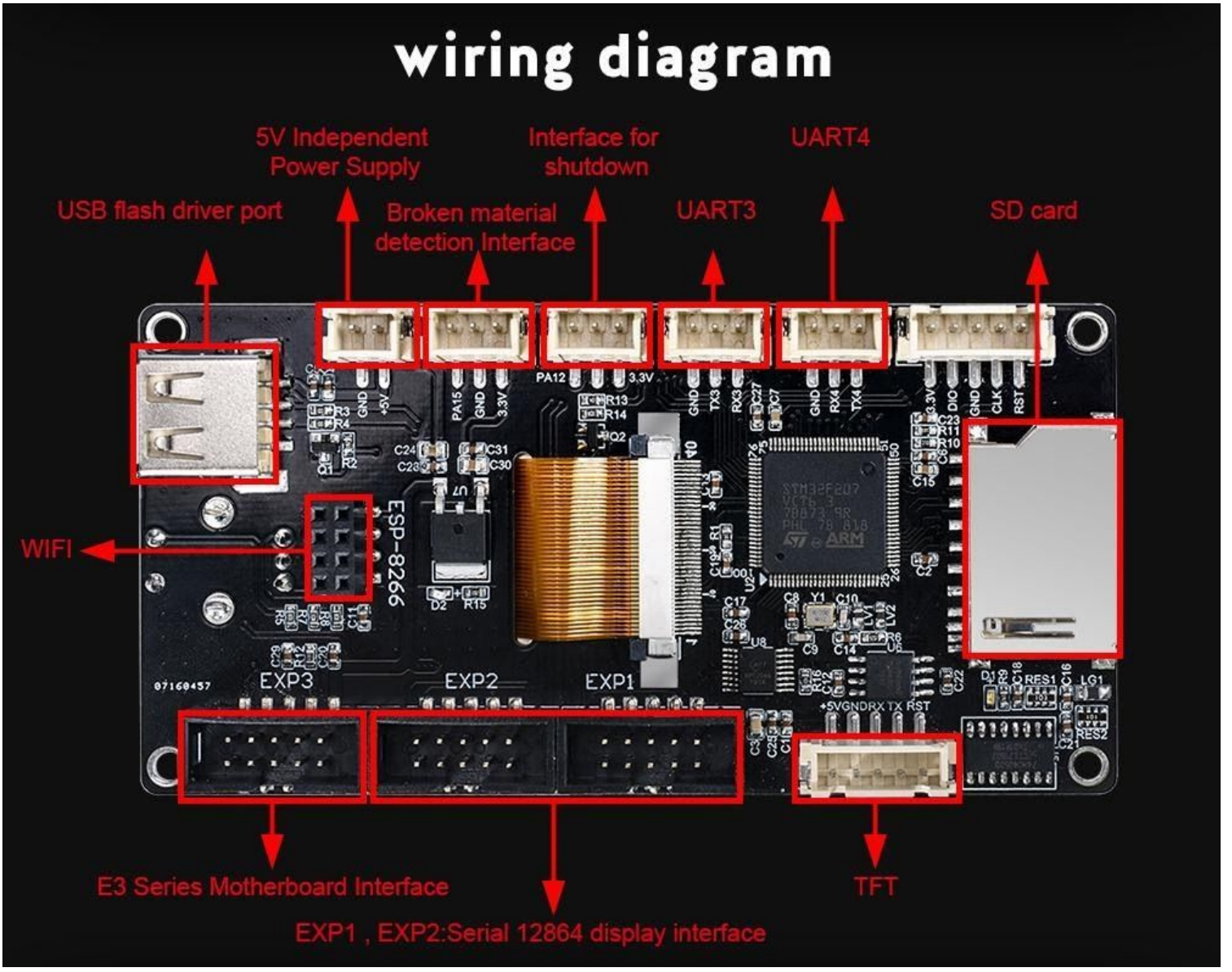
Deberemos ir a los menús del propio TFT y en configuración, definir los mismos baudios (velocidad de conexión) que tengamos en marlin. En el fichero config de marlin pusimos: #define BAUDRATE **115200** Deberemos por tanto indicarle que utilice 115200 para conectarse.

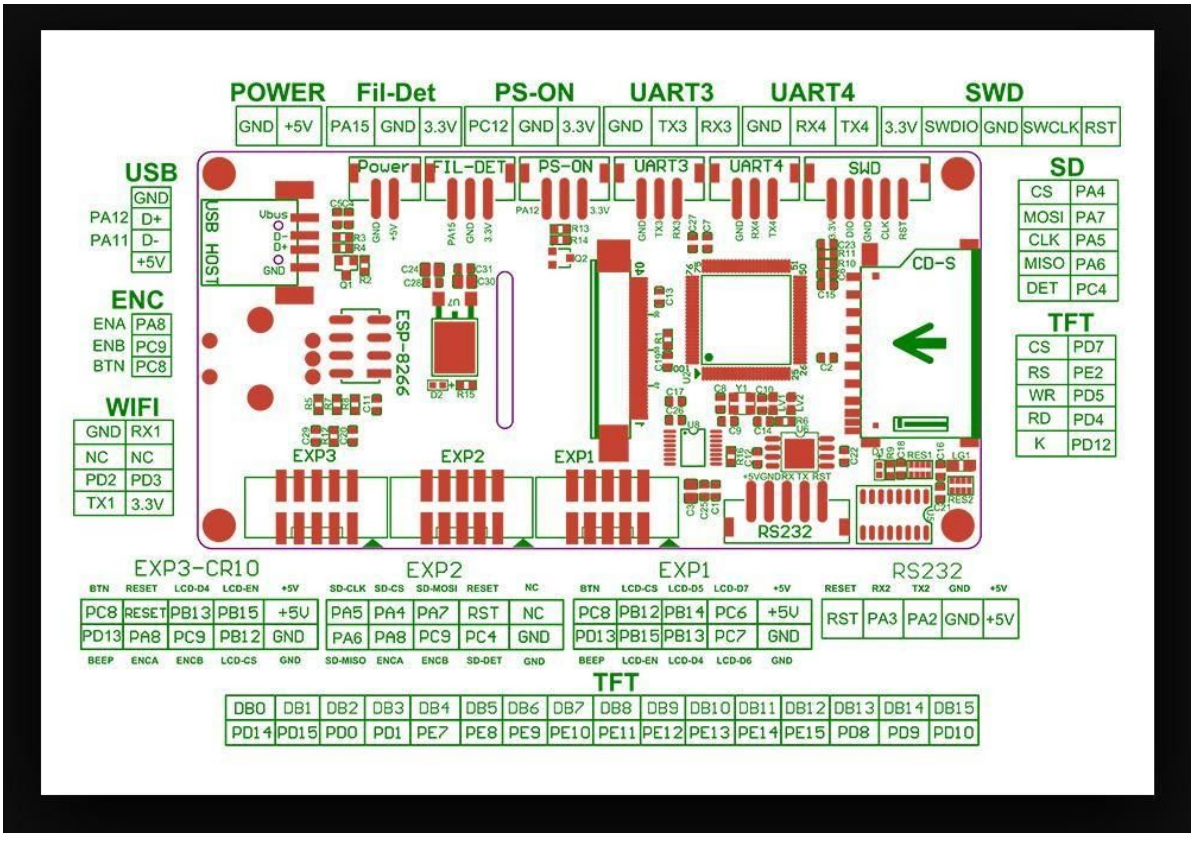
TFT TÁCTIL SOLO : Versiones anteriores a 3.0 de tft35
TFT DUALES (modo táctil TFT + modo emulación LCD12864)

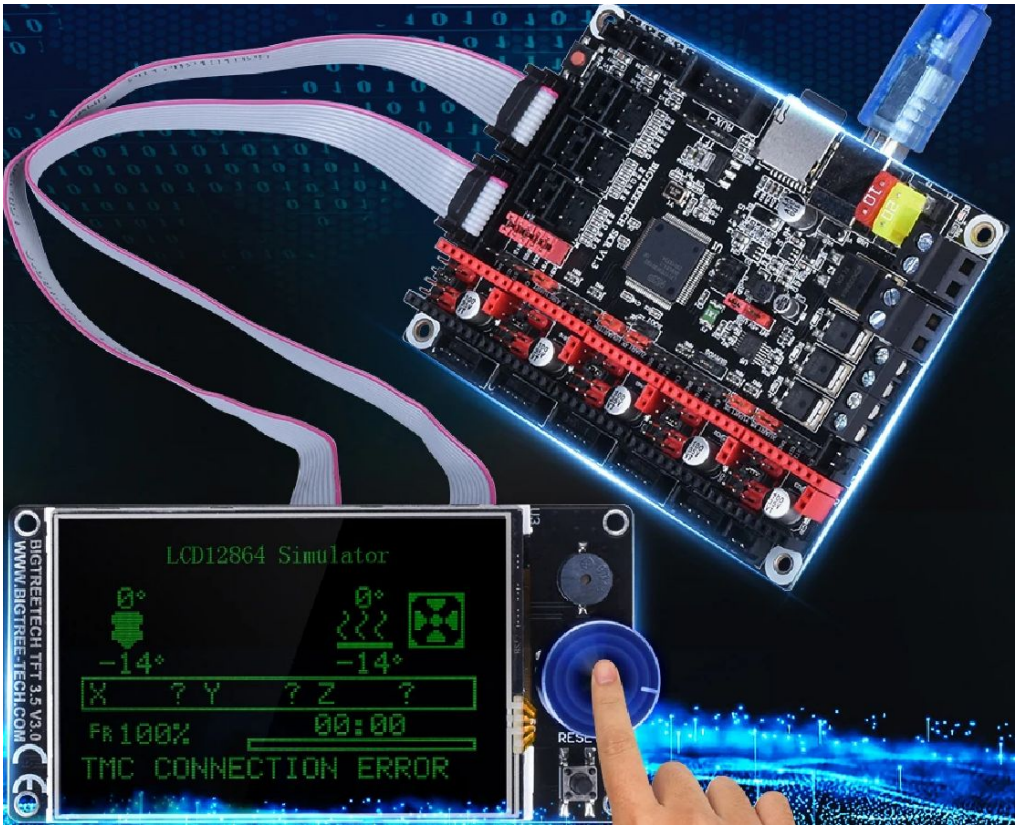
-TFT35 V3.0 (3.5 PULGADAS)
https://www.biqu.equipment/products/bigtreetech-tft35-v3-0-display-two-working-modes

-TFT24 V1.1 (2.4 PULGADAS)
https://www.biqu.equipment/collections/all-product/products/bigtreetech-tft24-v1-1-display-two-working-modes

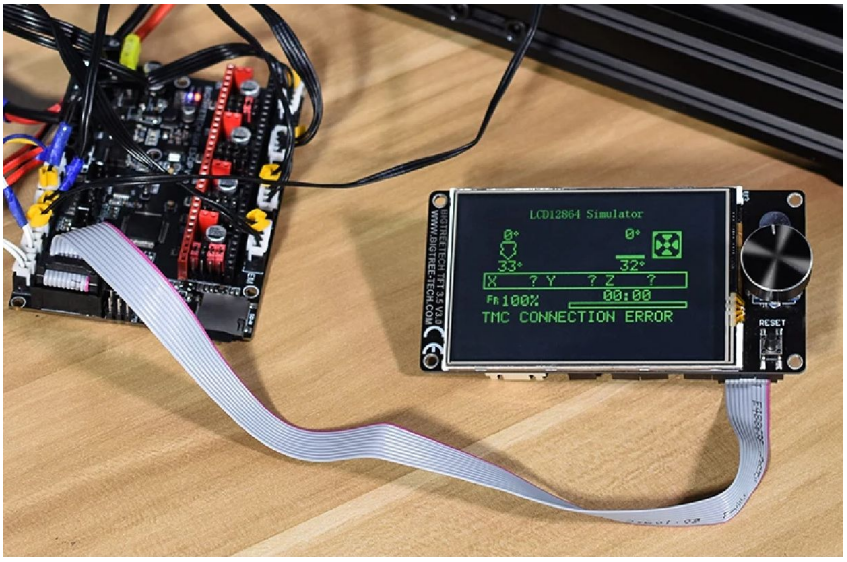
wiring diagram






LCD12864 Simulator
0°
-14°
0°
-14°
X ? Y ? Z ?
FR 100%
00:00
TMC CONNECTION ERROR
BIGTREETECH TFT 3.5 V3.0
WWW.BIGTREE-TECH.COM

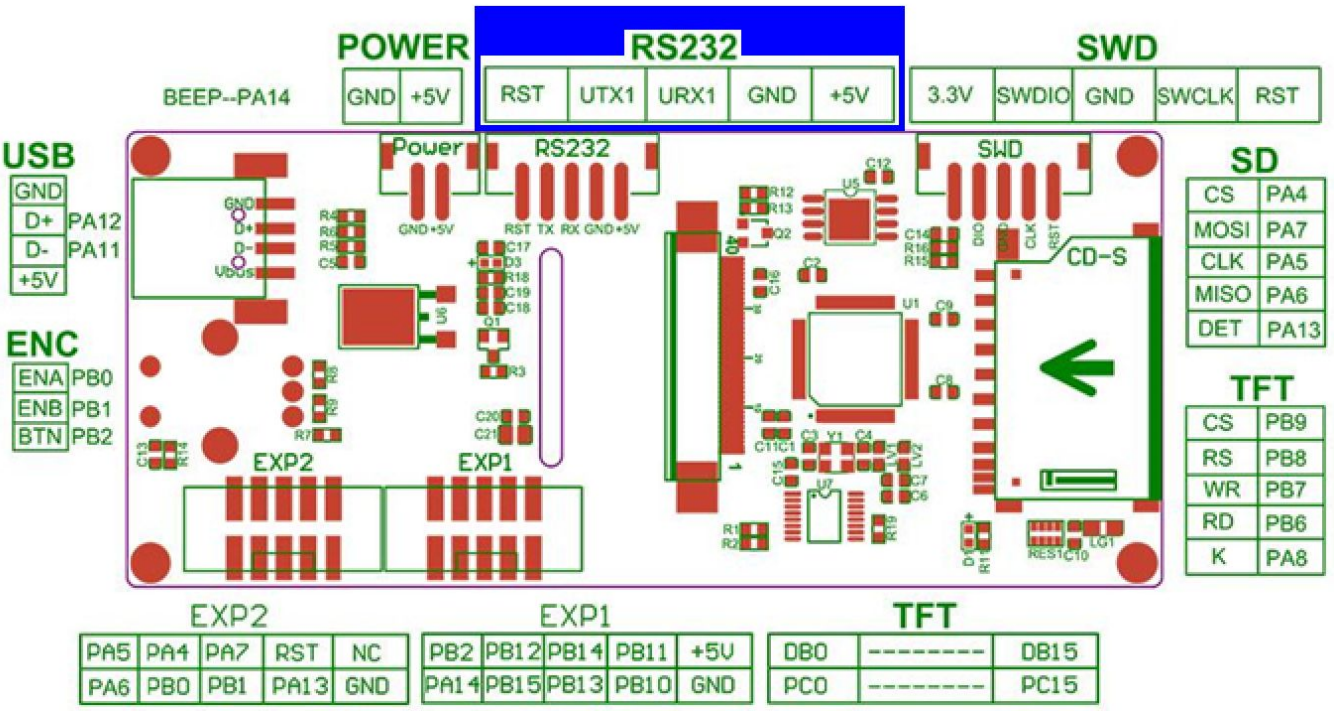

Ready
Heat
Move
Home
Print
Extrude
Fan
Settings
Leveling
Ready
Heat
Move
Home



BIGTREETECH TFT24 V1.1 (2.4 PULGADAS)

BIQU®

Conexión TFT24 en placa SKR 1.3



PLACA SKR V1.3