# Firmware and Electronics

T. Sailesh Kumar and Seepana Prasanth Kumar

# Firmware:

- Software that runs on the 3D printer's microcontroller

- Firmware is the link between software and hardware: It converts inputs from software to an output that computer hardware can understand.

- In 3D printing, that process happens whenever you send a G-code file from your slicing software to the 3D printer: The firmware "works out" the code and accordingly gives an output to the stepper motors, heaters, display, and so on.

# G- code

- G-code is a language that humans use to tell a machine how to do something.
- With 3D printing, g-code contains commands to move parts within the printer.
- G-code consists of G- and M-commands that have an assigned movement or action.

- G-codes deal with moving motors and parts, while M-codes deal with machine-specific settings that tell the printer what to do.

- For example, an M-code exists for setting nozzle temperature on virtually every functional 3D printer. These M-codes vary by machine and are therefore not standardized.

# “G1 X50 Y20 E15”

- G1 instructs the microcontroller to prepare for a linear movement.
- Then the new coordinates, which are X = 50, Y = 20, and E = 15, are read in by the firmware.
- For 3D printers, the E stands for the extrusion drive. When an E is present in a G code coordinate then either filament is being deposited or retracted.
- At this point the firmware needs to perform some basic calculations to determine how far off each carriage is from the required positions. After determining directions and distances the microcontroller will send step pulses to the stepper drivers, which will move the appropriate stepper motors.

# Example code:

```
11 G1 F900 X197.600 Y29.900 E19.82400
```

11 → Indicates the line of code and is used for reference

G/M → Blue Text is a G- or M-command

Red Text defines certain parameters

F → Speed

X/Y/Z → Coordinates

E → Feeder movement

; → A semicolon behind the code is used for comment information. The comment is not part of the code.

# Slicer

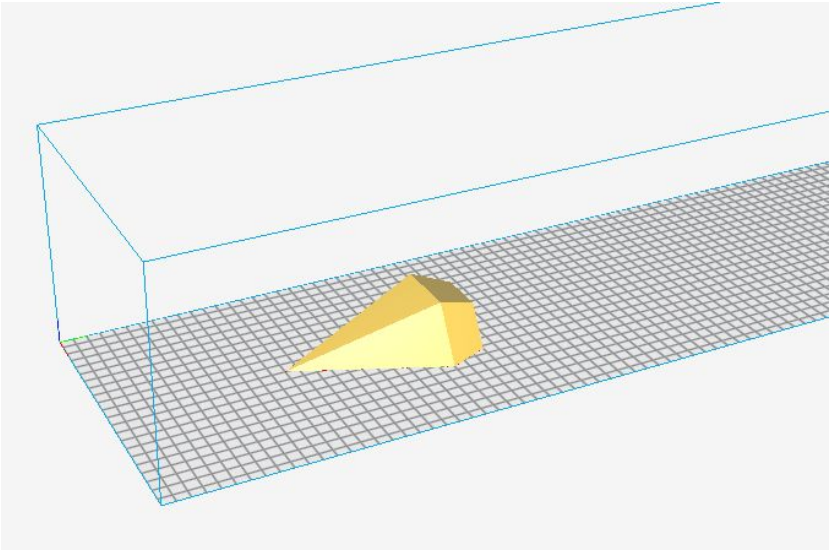
- The **slicer,** also called **slicing software,** is computer software used in the majority of 3D printing processes for the conversion of a 3D object model to specific instructions for the printer.

- In particular, the conversion from a model in STL format to printer commands in g-code format in fused filament fabrication and other similar processes.

# Changes we need to make for inf. 3D printer

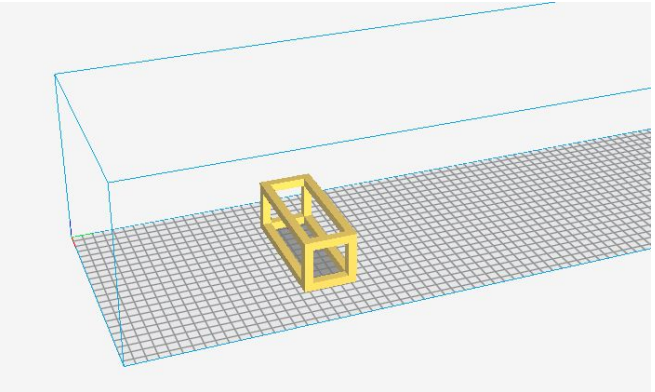
1. Position a flat edge as the first layer:

- If the very first layer is too short, the print may not adhere well to the conveyor belt. Not only will this create a poor-looking model, but could cause the model to get knocked off the conveyor belt, causing a print failure.

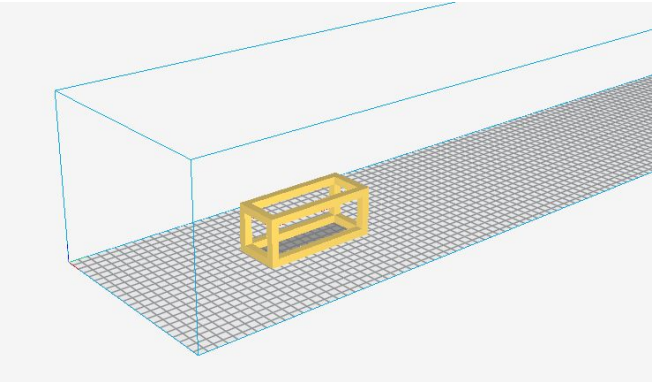


## 2. Position long ‘bridges’ along the belt axis

- On a conveyor belt 3D printer, short bridges should be positioned on the x-axis. Long bridges should be positioned along the y-axis.



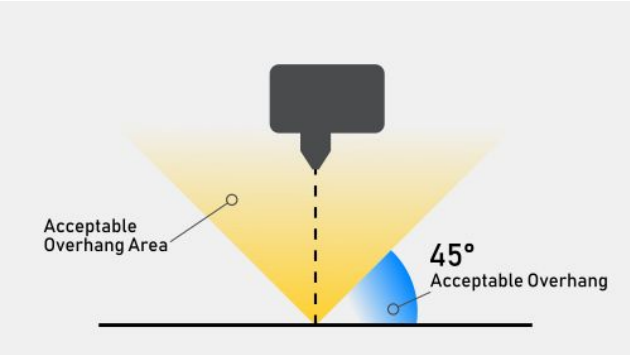
While this model is printable as-is, this is not the best orientation for a 3D printer with a conveyor belt.



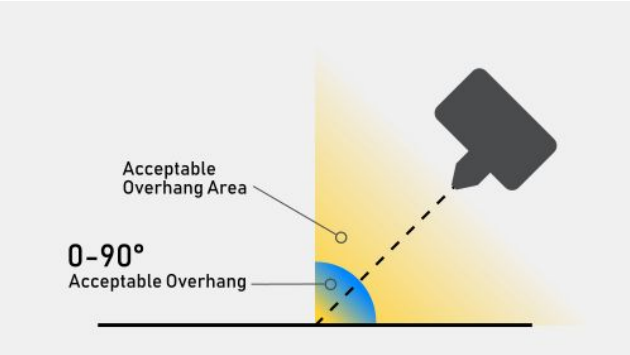
The longest overhangs should be positioned along the conveyor belt axis.

## 3. Adjust overhang areas

- Overhangs become more directional on a 3D printer that uses a conveyor belt. While conventional 3D printers are typically able to support overhangs up to 45 degrees, these machines can support any overhang angle, as long as it is in the y-axis direction.




Traditional 3D printers can print at an angle up to 45 degrees



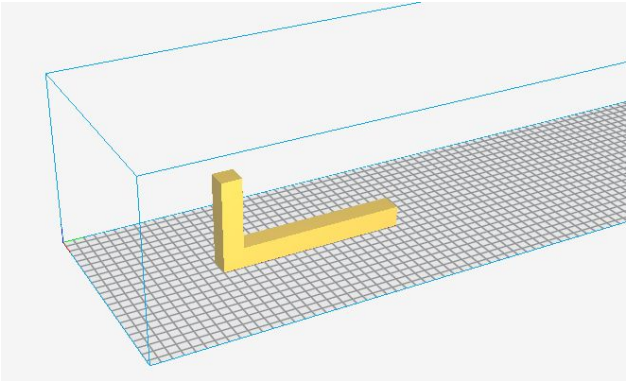

Conveyor belt 3D printers can print at any angle in the direction of the belt

## 4. Position the longest feature of your part along the belt axis

- This is a little up to personal preference, but generally speaking, the longest dimension of your part should be aligned with the conveyor belt axis. This will make the part the most stable and give the best surface finish.

## 5. Position largest/tallest feature last

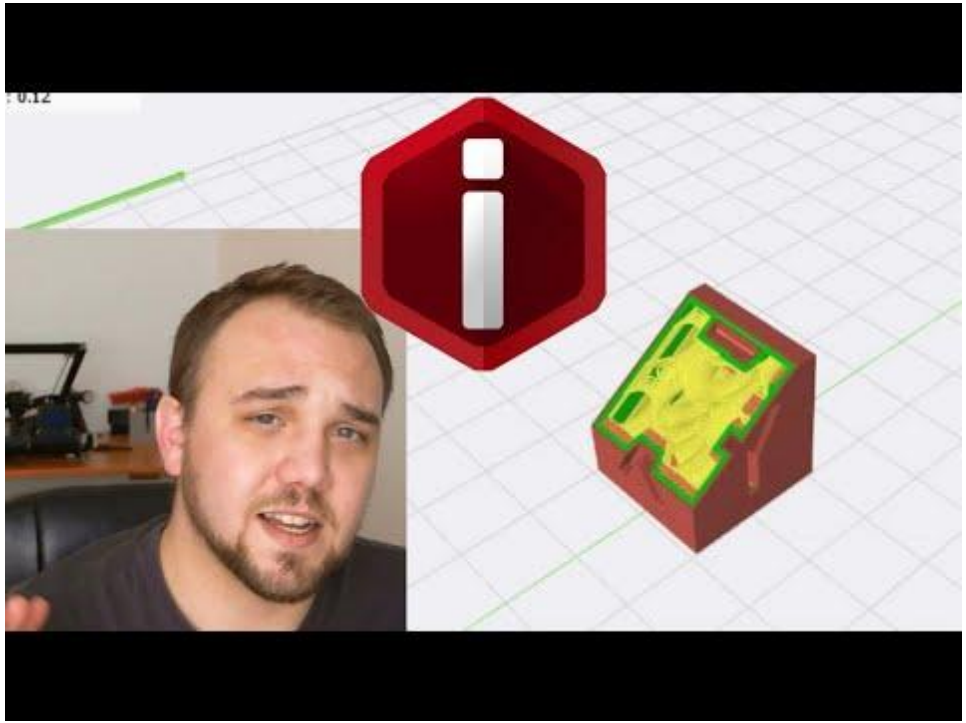
- In general, the shortest part of the print should be printed – and leave the conveyor belt – first. This helps keep the print stable during printing.



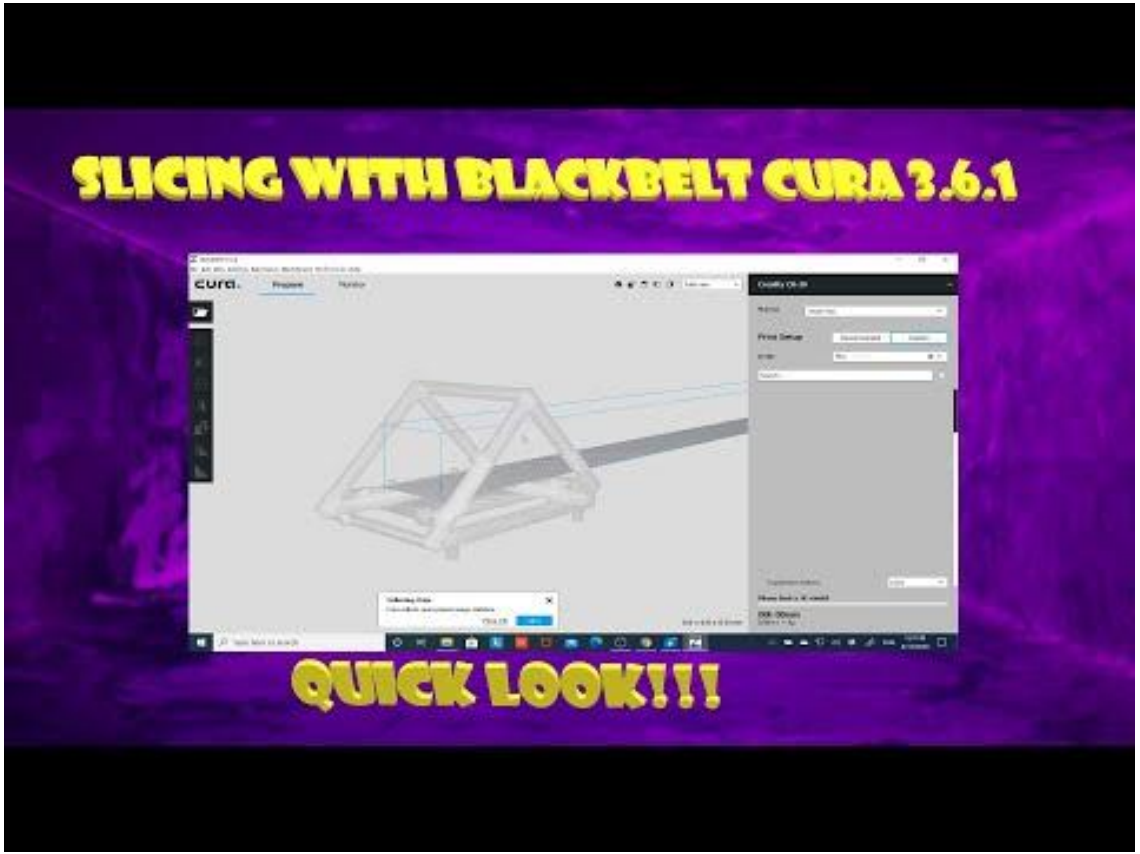
The tallest feature should be positioned towards the start of the conveyor belt (rear of the printer) to make the model as stable as possible.

# Options available for us:

1. IdeaMaker from Raise3D



# G Code From Blackbelt Cura



# Configuration Variables

- The appropriately named configuration variables are where your printer's specific information is stored.
- These variables are typically located in an accessible file known as a configuration file.
- Imagine that you are the microcontroller for a 3D printer, and you have a long list of G-code that you need to turn into a physical 3D object. What information do you need to be stored in the firmware to make that happen?

# Example List:

- Number of extruders – How many extruders does the 3D printer have?
- 
- Heated bed – Is there a heated bed?
- Location of limit switches – Are the limit switches at the minimum or maximum of their respective linear guides?
- Z probe – Is there a Z-probe? If not, then how far away is the extruder from the build plate?
- Stepper motor rotation to linear motion – For each axis, how far does the carriage move per rotation of the stepper motor?
- Build Volume – What is the maximum travel for each axis?
- Direction – Does clockwise or counterclockwise rotation of each stepper motor result in positive movement?
- Filament extrusion – How much filament is extruded per rotation of the extrusion drive stepper?
- Speed – What is the maximum speed each motor is capable of?

# “M92 X80 Y80 Z1200 E410”

- The M92 command tells the RepRapFirmware to set how many step pulses it takes to cover **1 mm** of linear distance for the following axes.

- In this example, the X and Y carriages move 1 mm every 80 step pulses while the Z axis moves 1 mm every 1200 pulses.

- The E parameter once again refers to the extruder. Here it takes 410 step pulses to extrude 1mm filament.

# Different Firmwares Available:

# Marlin


- Most widely known and commonly used 3D printer firmware options
- Marlin's strengths are in its high level of customization and strong community support.
- Comes in two different versions: one for 8-bit boards and one for 32-bit boards. The 32-bit version offers better and more up-to-date features.

- **Website:** Marlin
- **Key features:** Supports a large variety of machines, highly compatible, widely used, frequently updated
- **Compatibility:** Virtually every 8-bit controller or 32-bit controller boards
- **Requirements:** Arduino IDE

# Klipper

- newer 3D printer firmware

- The main highlight of Klipper is its ability to do calculations at a high speed, which results in faster 3D printing speeds. Klipper firmware can enable your 8-bit 3D printer to run at speeds greater than 80-100 mm/s

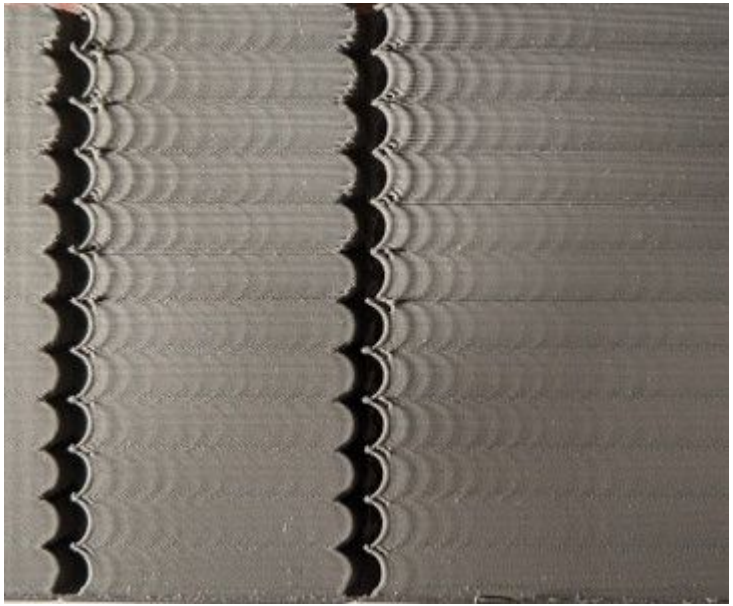
  .

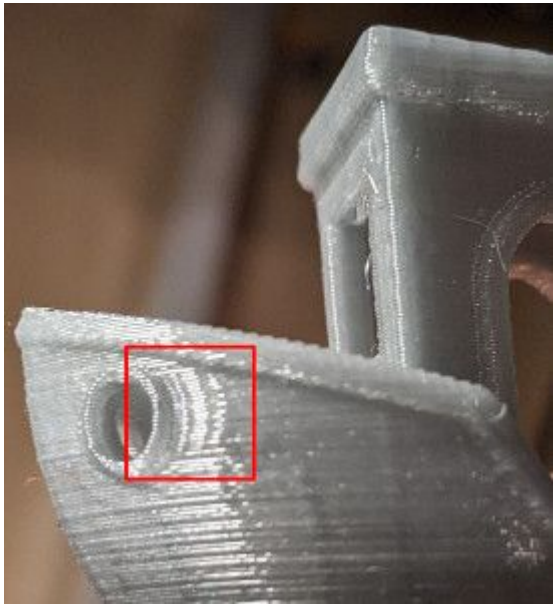
- To achieve this, it makes use of an additional single-board computer like a Raspberry Pi

  .

- The features such as “smooth pressure advance” and “input shaping” deliver a highly reliable and good quality 3D printing experience.

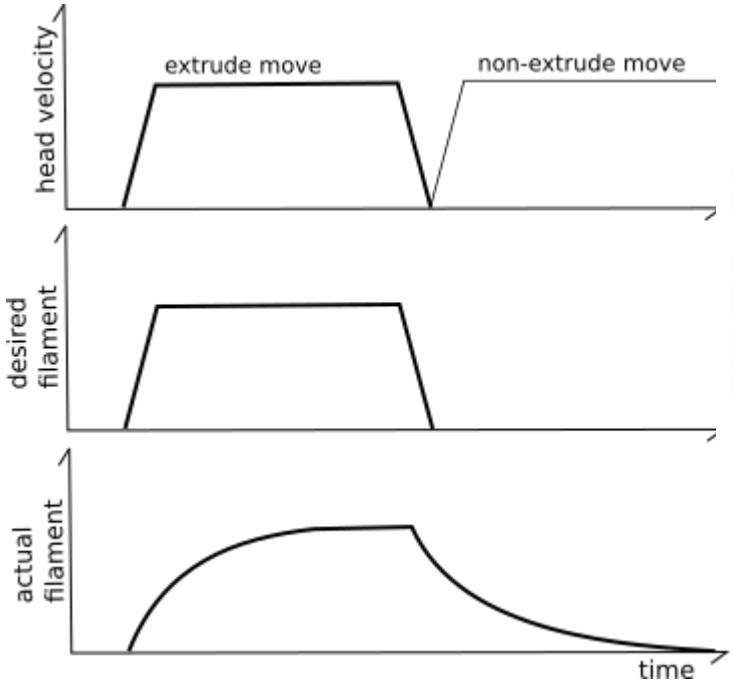
# Input Shaping

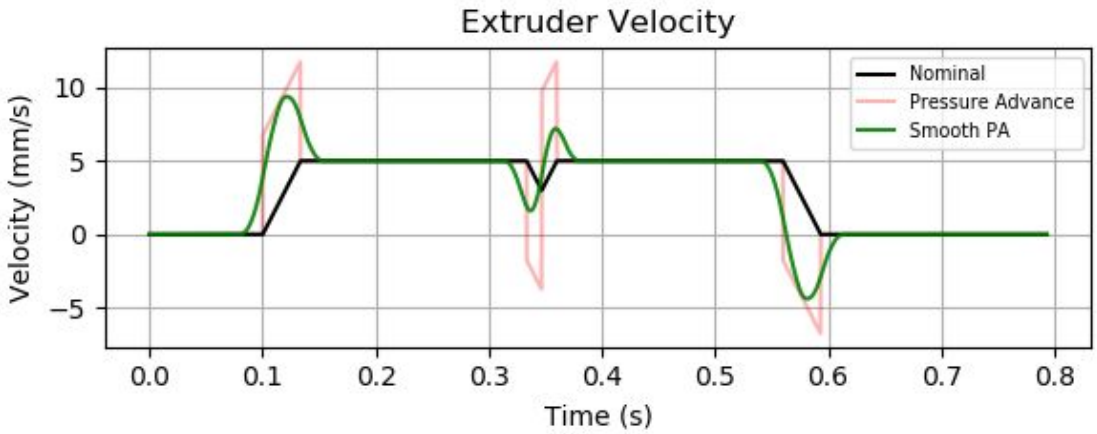




- Input shaping is an open-loop control technique which creates a commanding signal that cancels its own vibrations.

# Smooth Pressure Advance



# Klipper

- It can run on many Cartesian and delta-style 3D printers with little additional hardware (other than the Pi).

- Klipper firmware is written in Python, but with simpler coding.

- **Website:** Klipper
- **Key features:** Allows printing at higher speeds, OctoPrint compatibility, precise stepper motor movement, well documented
- **Compatibility:** Atmel ATmega-based microcontrollers, ARM-based microcontrollers, Beaglebone PRU-based printers
- **Requirements:** Raspberry Pi

# Prusa

- The Prusa firmware is one of the modified versions of Marlin

- It's tuned to work with the Einsy Rambo boards that come with Prusa 3D printers. This means that the firmware is only compatible with Prusa's own printers

    - **Website:** Prusa
    - **Key features:** Clean and easy to use, well documented, open source
    - **Compatibility:** Einsy Rambo board
    - **Requirements:** Prusa 3D printer

# Repetier


- Repetier firmware generates very reliable and fast 3D prints and works for both Cartesian and delta 3D printers.

- Another great feature is that the Repetier firmware is highly customizable.

- Also, being open source and free to download helps with regular feature updates and maintenance of this firmware.

- **Website:** Repetier
- **Key features:** Excellent documentation, compatible with Duet-based boards, optimized for use with Repetier-Host, easy to customize
- **Compatibility:** A range of 8-bit boards, plus RADDS (RepRap Arduino-Due Driver Shield)
- **Requirements:** Arduino IDE

# RepRap


- Originally designed for the Duet controller board, RepRap firmware was one of the earlier options to support 32-bit boards.

- There's also a great online configuration tool that can help you define or update your firmware configuration file.

- One limitation is that RepRap firmware only works on a few boards, specifically ones that use 32-bit AVR chips

.

- **Website:** RepRap
- **Key features:** Highly modular, web-based configuration and control
- **Compatibility:** Duet, RADDS, Smart-RAMPS
- **Requirements:** A text editor, Pronterface host software, Bossa (depending on the board)

# Smoothieware

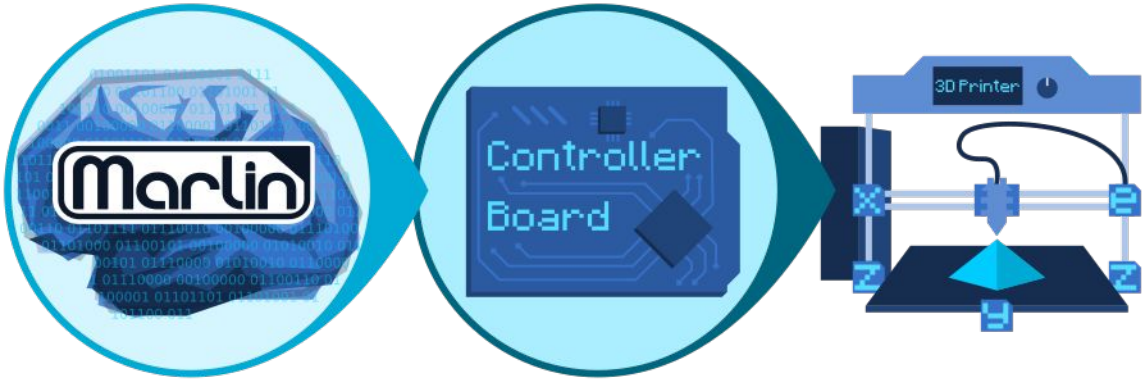
- Smoothieware is a multipurpose, feature-rich program that was developed with performance in mind.
- It also allows for a huge amount of custom configuration, including the ability to add extra stepper motors and configure the board for laser cutting or CNC milling.
- Similar to RepRap firmware, Smoothieware is limited to very few boards, the only major ones being the Smoothieboard itself and the Azteeg X5 Mini.
- The user and development community is also relatively new and small. But with great documentation, this open-source and free firmware is easy to explore.

# Smoothieware

- **Website:** Smoothieware
- **Key features:** 32-bit compatibility, easy and flexible configuration, support for laser cutters and CNC mills
- **Compatibility:** Smoothieboard, R2C2 Electronics Board, Azteeg X5 mini
- **Requirements:** Text editing software

# Marlin Firmware

https://www.youtube.com/watch?v=qjjDF0HIrcY



## Configuration _H_Version:

Denotes the version of the firmware and it is used to report bugs in the firmware to the open community

## Marlin logotype (SHOW_BOOTSCREEN)

It’s the Marlin logo that appears when the firmware boots. By default it appears active, we can deactivate it if we wish without any problem. It is not essential to configure Marlin 2 but we may even free up some memory.

## Communication ports (SERIAL_PORT y SERIAL_PORT_2)

They are the communication ports of our electronics. In principle, the first one should not be modified unless you have some exotic electronics. The second I leave it disabled because I have an electronic **MKS Gen v1.4** .

If your electronic board is a **SKR V1.4** or **SKR V1.4 Turbo** you need activate **SERIAL_PORT_2** and put the value -1 (as is by default). And for electronics **SKR GTR V1.0** you must put the values -1 and 3 respectively in **SERIAL_PORT** and **SERIAL_PORT_2**.

These are certainly the electronics that I am most familiar with. For others you will have to search the values online, something that will not take you too long to locate.

# ARDUINO BAUD RATE:

This setting defines the speed at which the 3D printer will communicate with your computer's USB port. The default setting is 250000 but can be changed to 115200 if your computer is having issues communicating with the 3D printer.

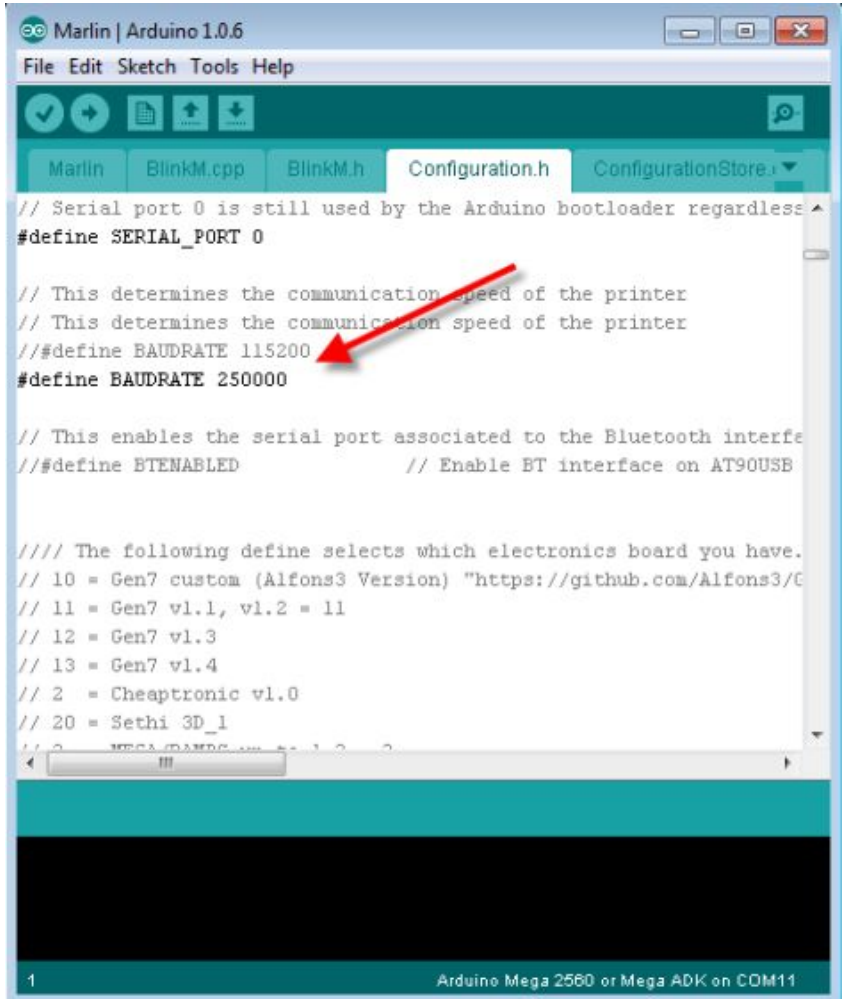
Use the scroll bar to locate the line #define BAUDRATE

If you wish to utilize the default baud rate leave it as is

If you wish to change the baud rate to 115200

o Remove the two slashes in front of the line #define BAUDRATE 115200

o Add two dashes in front of the line #define BAUDRATE 250000


Marlin | Arduino 1.0.6
File Edit Sketch Tools Help
Marlin
BlinkM.cpp
BlinkM.h
Configuration.h
ConfigurationStore.
// Serial port 0 is still used by the Arduino bootloader regardless
#define SERIAL_PORT 0
// This determines the communication speed of the printer
// This determines the communication speed of the printer
//#define BAUDRATE 115200
#define BAUDRATE 250000
// This enables the serial port associated to the Bluetooth interfe
//#define BTENABLED // Enable BT interface on AT90USB
//// The following define selects which electronics board you have.
// 10 = Gen7 custom (Alfons3 Version) "https://github.com/Alfons3/
// 11 = Gen7 v1.1, v1.2 = 11
// 12 = Gen7 v1.3
// 13 = Gen7 v1.4
// 2 = Cheaptronic v1.0
// 20 = Sethi 3D_1
1
Arduino Mega 2560 or Mega ADK on COM11

## #ifndef :

In the C Programming Language, the #ifndef directive allows for conditional compilation. The preprocessor determines if the provided macro does **not** exist before including the subsequent code in the compilation process.

## RAMPS board

RepRap Arduino Mega Polulu Shield, or RAMPS, is a board that serves as **the interface between the** Arduino Mega — the controller computer — and the electronic devices on a RepRap 3D printer.

## Motherboard or electronics (MOTHERBOARD)

Very important parameter. Here we will select the electronics that we have installed in our 3D printer. As there are many electronics available, we will have to find which one is ours and the value to place here.

You have a list of compatible electronics to use in the file "**boards.h**". This is located in the following path of your Marlin firmware:
**\Marlin\src\core\boards.h**

In my case, my 3D printer uses a **MKS Gen v1.4** electronic boards. For it I must write "**BOARD_MKS_GEN_13**" as you can see in the example. You configure the one you have.

## Name of your machine (CUSTOM_MACHINE_NAME)

This is easy, the name you want your new configured printer to have. By default you will notice that it is disabled, hence these 2 bars appear **“//”** in front of **#define**.

## #define MACHINE_UUID:

Printer's unique ID, used by some programs to differentiate between machines.

## LINEAR_AXES

`LINEAR_AXES` : The number of axes that are not used for extruders (axes that benefit from endstops and homing). `LINEAR_AXES` > `3` requires definition of `[[I, [J, [K]]]_STEP_PIN`, `[I, [J, [K]]]_ENABLE_PIN`, `[I, [J, [K]]]_DIR_PIN`, `[I, [J, [K]]]_STOP_PIN`, `USE_[I, [J, [K]]][MIN || MAX]_PLUG`, `[I, [J, [K]]]_ENABLE_ON`, `DISABLE_[I, [J, [K]]]`, `[I, [J, [K]]]_MIN_POS`, `[I, [J, [K]]]_MAX_POS`, `[I, [J, [K]]]_HOME_DIR`, possibly `DEFAULT_[I, [J, [K]]]JERK`, and definition of the respective values of `DEFAULT_AXIS_STEPS_PER_UNIT`, `DEFAULT_MAX_FEEDRATE`, `DEFAULT_MAX_ACCELERATION`, `HOMING_FEEDRATE_MM_M`, `AXIS_RELATIVE_MODES`, `MICROSTEP_MODES`, `MANUAL_FEEDRATE` and possibly also values of `HOMING_BUMP_DIVISOR`, `HOMING_BACKOFF_POST_MM`, `BACKLASH_DISTANCE_MM`. For bed-leveling, `NOZZLE_TO_PROBE_OFFSETS` has to be extended with elemets of value 0 until the number of elements is equal to the value of `LINEAR_AXES`.

Allowed values: [3, 4, 5, 6]

## AXIS4_NAME

`AXIS4_NAME`, `AXIS5_NAME`, `AXIS6_NAME` : Axis codes for additional axes: This defines the axis code that is used in G-code commands to reference a specific axis.

- 'A' for rotational axis parallel to X
- 'B' for rotational axis parallel to Y
- 'C' for rotational axis parallel to Z
- 'U' for secondary linear axis parallel to X
- 'V' for secondary linear axis parallel to Y
- 'W' for secondary linear axis parallel to Z

Regardless of the settings, firmware-internal axis names are I (AXIS4), J (AXIS5), K (AXIS6).

Allowed values: ['A', 'B', 'C', 'U', 'V', 'W']

## Number of extruders (EXTRUDERS)

Here we are going to configure the number of extruders that our 3D printer has. By default, the initial value is 1, since most have an extruder. If this is not your case, enter the correct number of extruders.

## Filament diameter (DEFAULT_NOMINAL_FILAMENT_DIA)

In this section we will select the type of diameter that our printer uses. By default 3.0 comes, something strange since now most printers already usually operate at 1.75 in diameter. We will have to change it, very important

# SINGLE NOZZLE

If you have multiple extruders but a single nozzle,

# Prusa Multi Material upgrade:

## Prusa Multi Material Kit (PRUSA_MMU2)

Activate it if you have installed the original Josef Prusa multimaterial kit or some other **chinese clon** that are available in the market. If not, leave it as it is, with the two bars on.

## Temperature sensors (TEMP_SENSORS)

In this section of the Marlin firmware we will define the existing temperature sensors in our 3D printer. In this latest version 2.0.7.2, has been added support for more extruders (with a total of 11 sensors supported), including a sensor for the chamber temp.

As you can see, there are a lot of them, and a list appears in the code itself. In the case that you do not have information about yours, try consulting Google as it will probably be documented by some other user with your same printer.

By default, only **TEMP_SENSOR_0** is configured with value “1”. In my case, I must put the value “5” there.

As I also have a heated bed, to improve adhesion during printing and avoid **Warping issues**, I setup **TEMP_SENSOR_BED** with the value 1. I leave the rest at 0 since I don’t have any more extruders or a closed chamber at this printer

## Temperature limiters (MINTEMP & MAXTEMP)

Configure in Marlin 2 these parameters is very important, since they control the maximum and minimum temperatures of our printer.

We will define in which ranges each and every one of the available sensors must be. In this way, if any sensor does not reach or exceed this temperature, Marlin emits an error on the display and deactivates the Hotend and the heated bed.

In my case I usually set the extruder temperatures at 5 degrees (minimum) and 265 degrees (maximum). In the event that an extruder or the bed does not reach 5º degrees, we will deduce that the Thermistor is not installed correctly, or is physically damaged.

In the same way, it will protect us in the event that the maximum temperature is accidentally exceeded, something crucial for our safety and that of our home.

# PID

PID tuning refers to the parameters adjustment of a proportional-integral-derivative control algorithm used in most repraps for hot ends and heated beds.

PID needs to have a P, I and D value defined to control the nozzle temperature. If the temperature ramps up quickly and slows as it approaches the target temperature, or if it swings by a few degrees either side of the target temperature, then the values are incorrect

## PID Tuning (Extruders)

Section that controls the settings of the algorithm that manages the temperature of the extruders and the heated bed.

By default, some values are already predetermined for several of the best known printers on the market. Since my printer does not match the description, I have simply added my own values.

To get the values from your printer, you can consult them on the Internet or you can make them yourself with an internal Marlin function called **PID Autotune**.

To do this, connect your printer to Repetier/Simplify3D or any other software that allows you to use Gcode commands. Once connected, send the command **M303 E0 S200 C8** and the process will begin.

After heating and cooling the hotend 8 times, it will finish and display the following information. You only have to take the values of Kp, Ki and Kd and enter them in your firmware, in the fields **DEFAULT_Kp**, **DEFAULT_Ki** and **DEFAULT_Kd**.

```
bias: 92 d: 92 min: 196.56 max: 203.75
Ku: 32.59 Tu: 54.92
Clasic PID
Kp: 19.56
Ki: 0.71
Kd: 134.26
PID Autotune finished ! Place the Kp, Ki and Kd constants in the configuration.
```

## PID Bed Tuning (Heatbed)

The same as before, but for our heated bed. By default in the new firmware it will be disabled. If you have a heated bed, you must uncomment the **#define PIDTEMPBED** option as you will see in my configuration.

You can also run another internal Marlin function to get the optimal values for your bed. This is done with the Gcode command **M303 E-1 C8 S90**.

## Cold extrusion (PREVENT_COLD_EXTRUSION)

Both functions manage cold extrusion at Marlin. As we all know, it is not advisable to extrude filament if our Hotend is not turned on.

The option **PREVENT_COLD_EXTRUSION** it will directly prevent us from extruding material if our hotend does not reach the minimum temperature defined.

We can configure at what minimum temperature we will allow Marlin to extrude filament with the **EXTRUDE_MINTEMP** parameter. By default both are active, so you should not worry.

## Extrusion lenght (PREVENT_LENGTHY_EXTRUDE)

As indicated in the comments, it prevents extrusion of filament from a distance greater than the one we have configured. This means that Marlin will only allow you to extrude a maximum of 200mm in a single GCODE command.

If you want to extrude much more than 200mm in a single command, for example in the case of having a function to load the filament from the display using Bowden, you will have to modify the parameter **EXTRUDE_MAXLENGTH** for a higher value. Both are enabled by default in Marlin 2.

## Thermal protections (THERMAL_PROTECTION)

These three parameters activate the additional thermal protection that Marlin 2.0.x offers to prevent damage to our printers. They must always be active, in fact they come in Marlin 2 by default, so we will not touch them in any way.

## Mechanical parameters (COREXY kinematics)

There are different printers with different kinematics on the market. For example Cartesian kinematics, deltas, tripteron, corexy (and their variants), etc

In the case that your 3D printer uses COREXY kinematics or any of its variants, you should activate its corresponding parameter.

## Endstop availables (USE_XMIN_PLUG & USE_XMAX_PLUG)

In this section we are going to indicate to our Marlin firmware the limit switches that we have connected to our motherboard. The most normal thing is to have 3 limit switches to indicate the minimum travels (or stops). In this way Marlin determines the initial position (0) in all axes when doing a HOMING.

Extra sensors can also be installed to determine the maximum travels. They will serve to prevent the machine from moving beyond what is strictly necessary. They are not essential, but they are a good option to have all the unforeseen under control.

## Drivers configuration (DRIVER_TYPE)

Configuring the Marlin 2 firmware with the drivers that we have is something essential. But do not worry, it is a very simple process.

By default you will see that all the options are disabled, so you must enable the ones you are going to use, which will be where you have installed drivers on your board.

In my printer I have 3 motors in the 3 axes (X, Y, Z), and a single motor for the extruder. So I enable the necessary options. The first extruder is always named E0 as you can see.

But not only we must enable the option, but also indicate the driver that you have mounted on your electronics. Since I have **DRV8825** , I have indicated it to the compiler, and that’s it.

In the comments you can see all the types of drivers available in Marlin 2, so locate yours and write it down as it appears.

## Motor Steps configuration (DEFAULT_AXIS_STEPS_PER_UNIT)

In this section we will tell our printer how many steps will be carried out for each movement unit. Units may be defined in millimeters or inches.

In the case of using inches, you must activate the **INCH_MODE_SUPPORT** parameter. We are not going to activate that parameter, since we usually work with millimeters, and in this section we will introduce our steps for each of the axes and the extruder.

You first need to know the values to enter, so I recommend that you extract them from some firmware already available for your printer (to be safe).

Another option is to search for it on the Internet from some other user, browse forums, consult the manufacturer or, as a last option, calculate them yourself as you can see **in this video**.

## Reverse direction of motors (INVERT_X_DIR)

We will modify these parameters if when doing a HOME or when printing any part with our printer, any of the motors goes in the wrong direction. By default they are all configured with the FALSE option, but in my case I had to reverse the direction of my bed (Y axis) as you can see below.

However, another option is to invert the motor connection cable, so you can choose the more comfortable solution for you.

## Printing volume (X_BED_SIZE & Y_BED_SIZE)

We will define here the exact size of your Heatbed. My printer has a 285x206mm horizontal bed, so it would be configured this way:

## Movement limits (MIN_POS & MAN_POS)

Once the size of our printing base (Heatbed) have been defined, we must configure Marlin 2 with the minimum and maximum limits of movement.

By default, for the minimums Marlin comes configured with the value 0, which we should not modify since it is **HOME**. For the maximum limits, Marlin directly uses the current size of our Heatbed in X and Y, which we had previously left configured in the previous parameter.

To finish with the configuration, in the Z axis the measure that we must add is the maximum printing height of your 3D Printer. In my particular case, I can print approximately 296mm, so I enter its value.

## Movement restrictions (MIN_SOFTWARE_ENDSTOPS)

For security reasons, and to avoid damaging the components of our printer, Marlin 2 comes by default configured so that we cannot exceed the minimum limits.

This means that when reaching a limit switch or sensor, it will not allow the engine to advance further even if you do it manually.

## EEPROM storage (EEPROM_SETTINGS)

Marlin offers us the possibility to change the configuration of our printer from the display itself and then save the changes in the Eeprom.

If we want to have this possibility active, we must activate this parameter in our firmware.

## SD Card support (SDSUPPORT)

More than likely, it is essential to activate this option. It allows you to load the STL files using an SDCard.

Believe it or not, it is disabled by default in our Marlin firmware. You will need to activate it, as I have done in my firmware.

# Steps to convert into a infinite 3D printer

1. Find CoreXY(or any other) and uncomment it
2. Change the default_axis_steps_per_unit to the desired numbers
3. Change Z_max position to maximum value (99,999 mm, depends on the version)

That’s all

# Link to 2 open-sourced inf. 3D printing projects

EnderLoop by mcsgroi

Opensource Belt Printer (BlackBelt Prototype) by scalda

# Let's see some G-Code for conveyor belt printers!!

```
G90                ; Set to Absolute Positioning
M82                ; Set extruder to absolute mode
G21                ; Metric values
G92 X0 Y{blackbelt_z_offset} Z0 E0   ; Set all axis to 0

G1 Y2              ; Move Y axis off the bed
G1 E15             ; Extrude 5mm more
G1 Z10 E18 F500    ; Move belt 5mm and keep extruding
```

```
G1 Z30            ; Move the belt a bit further without extruding
G92 Z0            ; Zero Belt
G92 E0            ; Zero the extruded length
G1 E-4 F3900      ; Retract 4mm at 65mm/s
;prepare printing
G1 E0             ; Move extruder back to 0
G92 E-1           ; Add 1mm start distance
M117 Printing...
```

# Electronics

# Electronics

- Electronics system acts upon the instructions given by the user and can be considered as the backend of 3D printer
- Connects the software with the mechanical system of 3D printer.
- These parts are present in Electronics system of a 3D Printer:

1. Ramps Board(Microcontroller boards)
2. NEMA Stepper motors
3. End Stops
4. Motor Driver

# Electronic Boards

- Microcontroller boards control the entire printing process and many option are available for 3D Printers and all are open source.
- Commonly used- RAMPS, a DIY shield board for Arduino MEGA, Sanguinololu, a DIY board with microprocessor on board
- Functions:

  Processes G-code instructions.

  Controls and regulates the four stepper motor controllers.

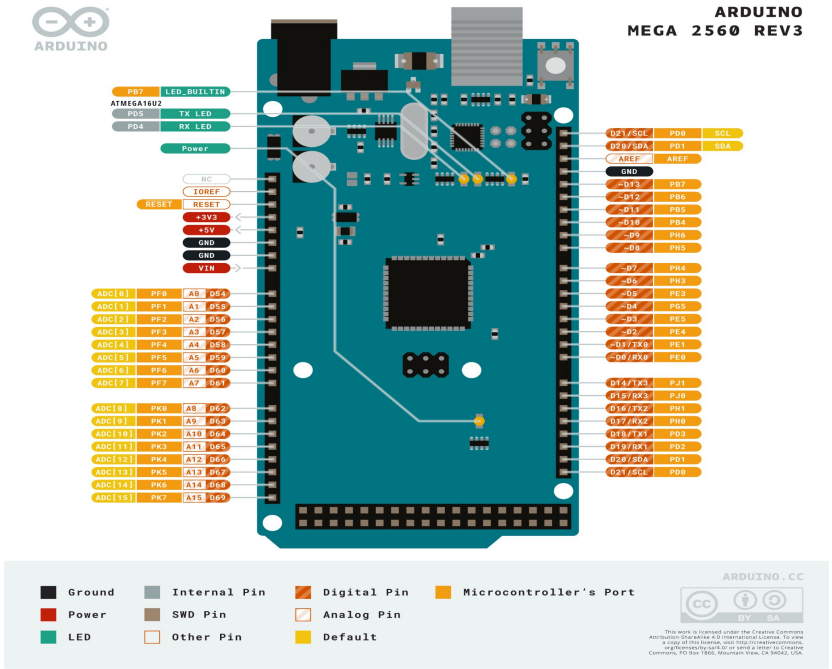
  Controls the temperature of the heated bed.

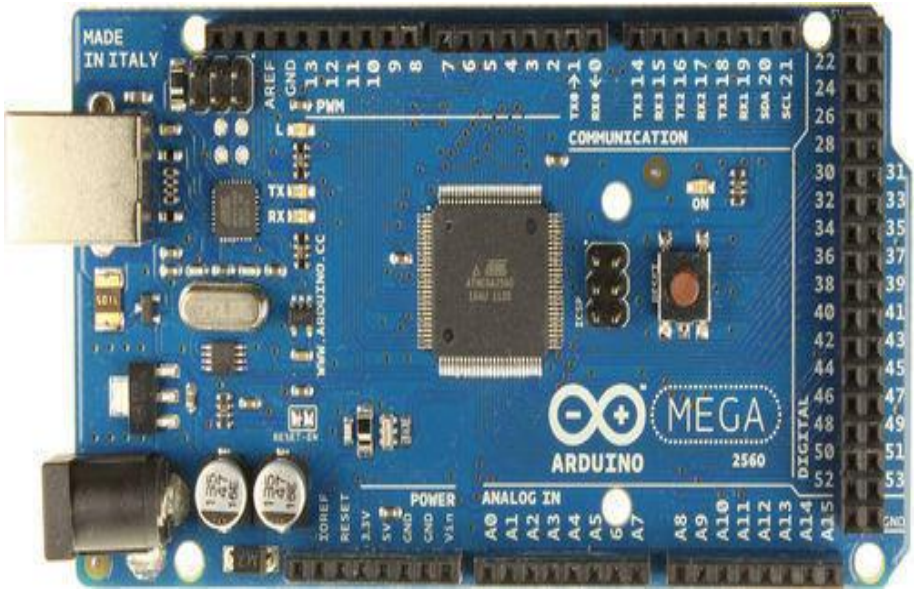
  Monitors the end stops.

# Microcontroller Board

- Arduino Mega 2560 is mostly used due to its capability to support complex projects as it has many features such as:

1. 54 digital input output pins
2. 16 analog inputs
3. 16 MHz crystal oscillator
4. Large space for sketch.


ARDUINO
ARDUINO
MEGA 2560 REV3
ATMEGA16U2
PB7 LED_BUILTIN
PD5 TX LED
PD4 RX LED
Power
NC
IOREF
RESET
+3V3
+5V
GND
GND
VIN
ADC[0] PF0 A0 D54
ADC[1] PF1 A1 D55
ADC[2] PF2 A2 D56
ADC[3] PF3 A3 D57
ADC[4] PF4 A4 D58
ADC[5] PF5 A5 D59
ADC[6] PF6 A6 D60
ADC[7] PF7 A7 D61
ADC[8] PK0 A8 D62
ADC[9] PK1 A9 D63
ADC[10] PK2 A10 D64
ADC[11] PK3 A11 D65
ADC[12] PK4 A12 D66
ADC[13] PK5 A13 D67
ADC[14] PK6 A14 D68
ADC[15] PK7 A15 D69
D21/SCL PD0 SCL
D20/SDA PD1 SDA
AREF AREF
GND
~D13 PB7
~D12 PB6
~D11 PB5
~D10 PB4
~D9 PH6
~D8 PH5
~D7 PH4
~D6 PH3
~D5 PE3
~D4 PG5
~D3 PE5
~D2 PE4
~D1/TX0 PE1
~D0/RX0 PE0
D14/TX3 PJ1
D15/RX3 PJ0
D16/TX2 PH1
D17/RX2 PH0
D18/TX1 PD3
D19/RX1 PD2
D20/SDA PD1
D21/SCL PD0
Ground
Power
LED
Internal Pin
SWD Pin
Other Pin
Digital Pin
Analog Pin
Default
Microcontroller's Port
ARDUINO.CC
BY SA


MADE IN ITALY
COMMUNICATION
POWER
ANALOG IN
DIGITAL
ARDUINO
MEGA
2560
WWW.ARDUINO.CC

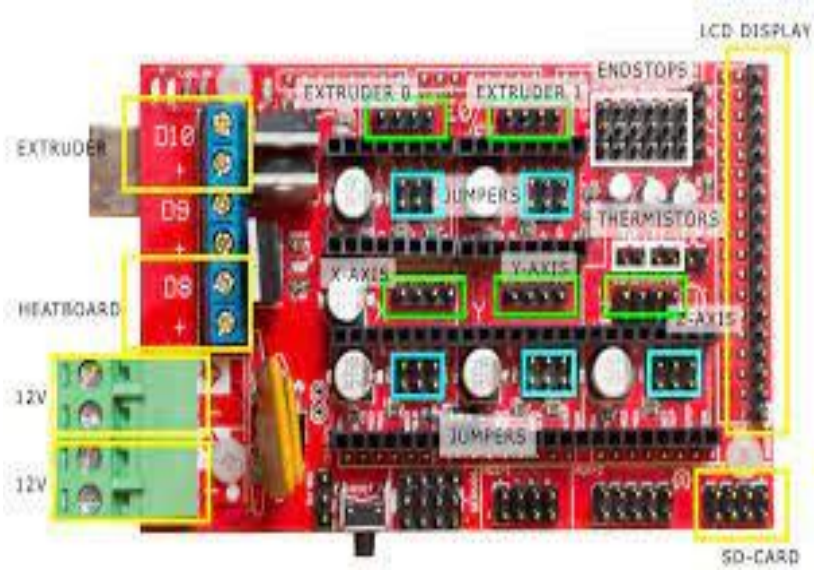
# Stepper Motors

- Stepper motor is a brushless DC electric motor that divides a full rotation into number of equal steps.
- Advantages:Low cost, high torque at low speed
- Disadvantages:Resonance effect,Decreasing torque with increasing speed.
- There are five stepper motors used in the 3D printer are One to control the Y-axis, One to control the X-axis, Two to control the Z-axis, One to control the extruder.
- The X-axis motor drives the extruder assembly along a timing belt.
- The number of motors in Infinite 3D Printer depends on what type of Printer it is, its functions and the axes to be printed.

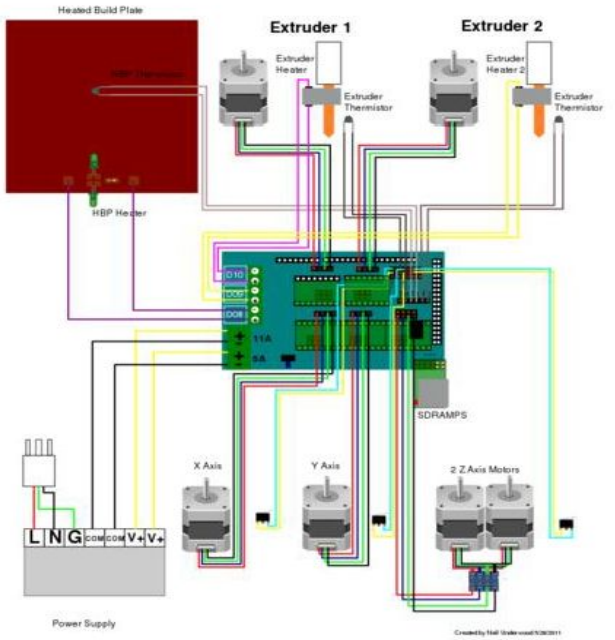
# RAMPS(RepRap Arduino Mega Pololu Shield)

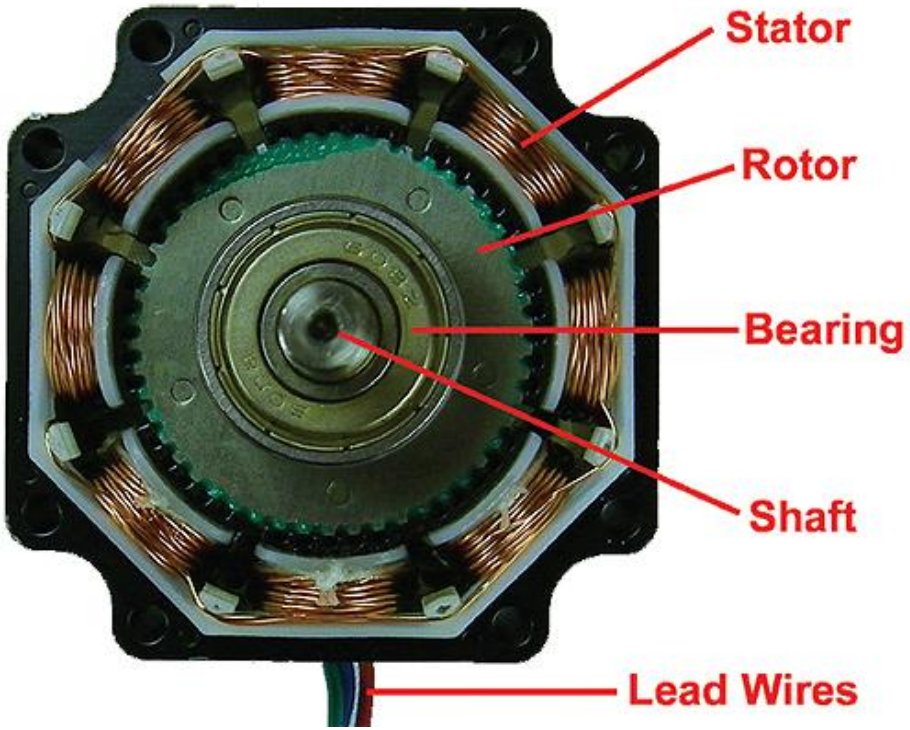
- Shields are the boards that can be plugged on the top of the arduino PCB extending its capabilities:

1. Motor controls for 3D printing
2. Liquid crystal display,
3. Ethernets and Global Positioning System



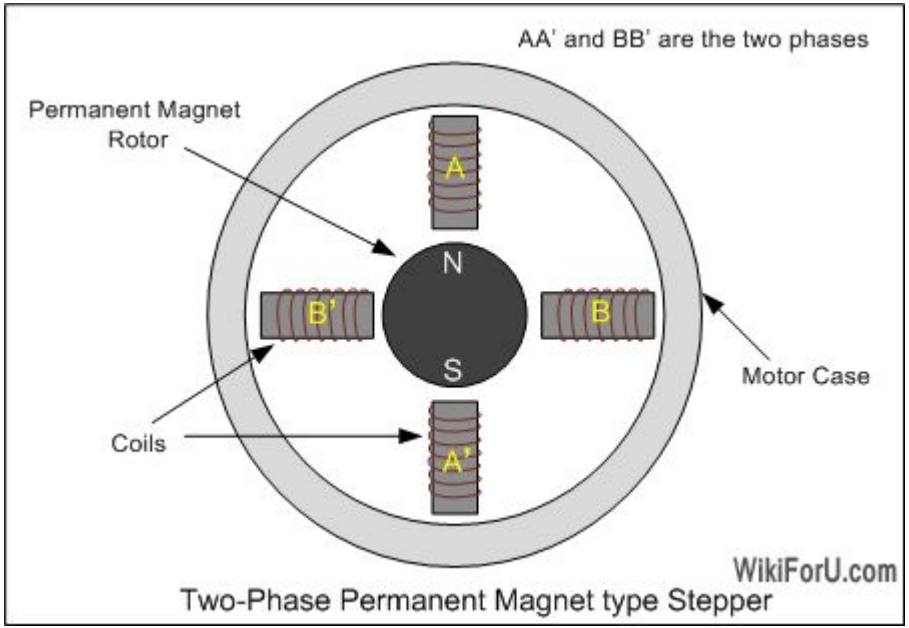

RepRap Arduino Mega Pololu Shield 1.4



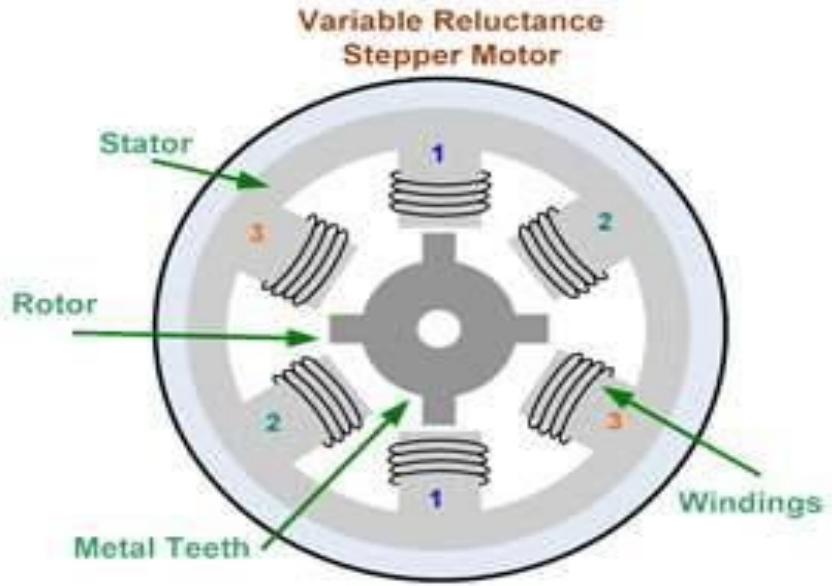

Stator
Rotor
Bearing
Shaft
Lead Wires

# Types of Stepper Motors

- There are three types of stepper motors,
- Permanent Magnet-Uses current flow through a magnetized rotor to achieve the steps of the motor
- Variable Reluctance-Uses a nonmagnetic geared rotor for its steps.
- Hybrid-Combines elements of the Permanent Magnet and Variable Reluctance.
- The permanent magnet and Hybrid motors have sub variations- Unipolar and Bipolar motors.
- Unipolar motor allows only half winding and supports single direction flow.
- Bipolar motor allows each stator to be magnetized to South or North pole. It generates more torque and requires complex circuits.

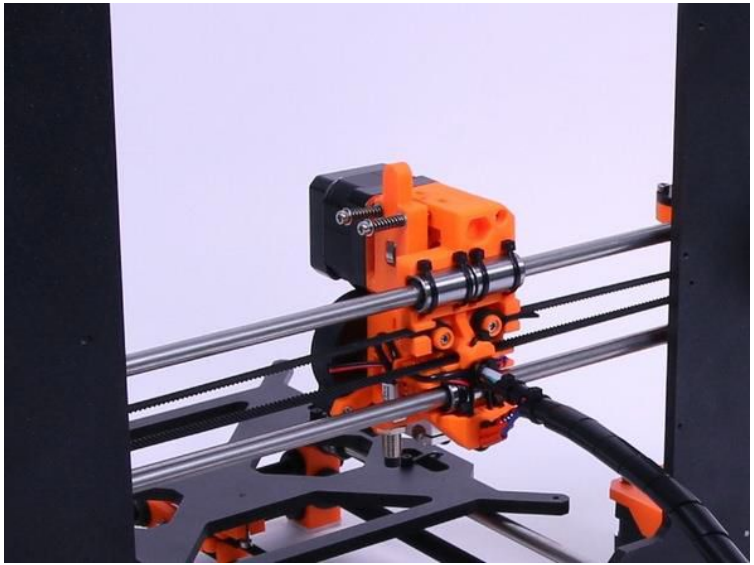


Two-Phase Permanent Magnet type Stepper



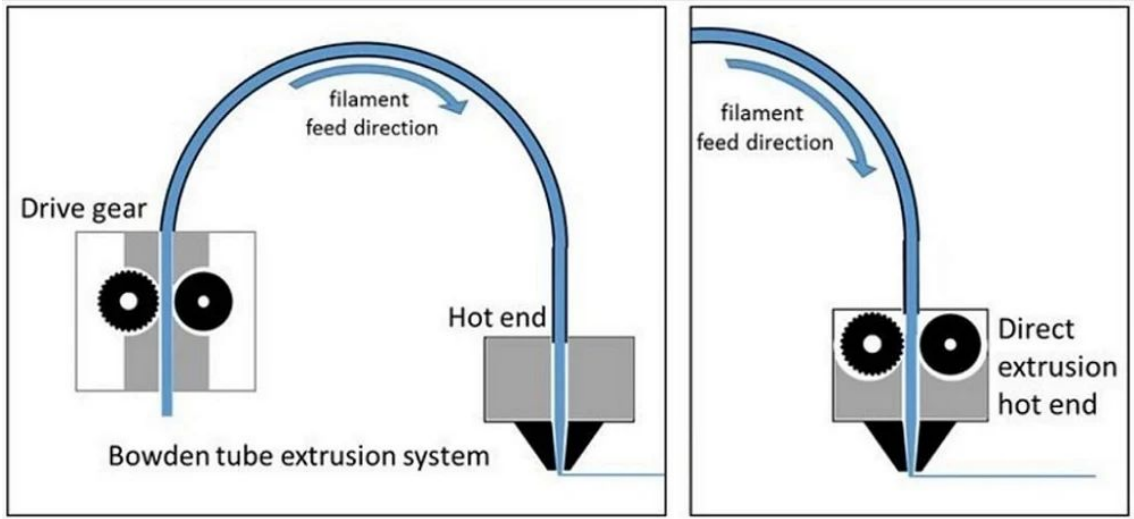
# Extruder

- Responsible for passing filament into hot end.
- Uses Stepper Motor.
- Uses Gears and bolts mechanism.
- Can make precise movements while printing.
- Angled Printing in Infinite 3D Printers.
- Vertical Resolution



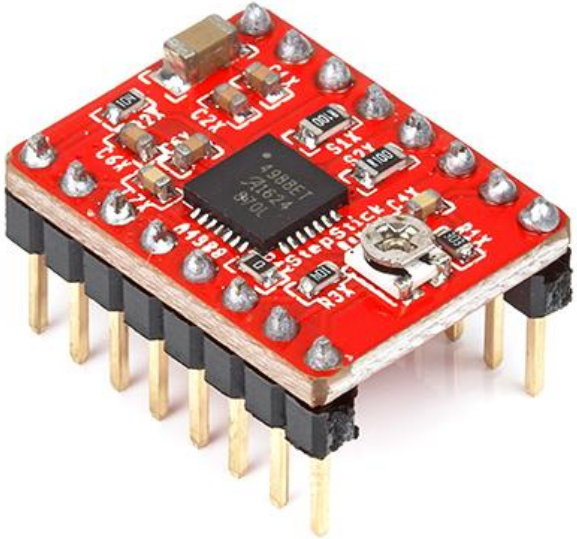
# Types of Extruder

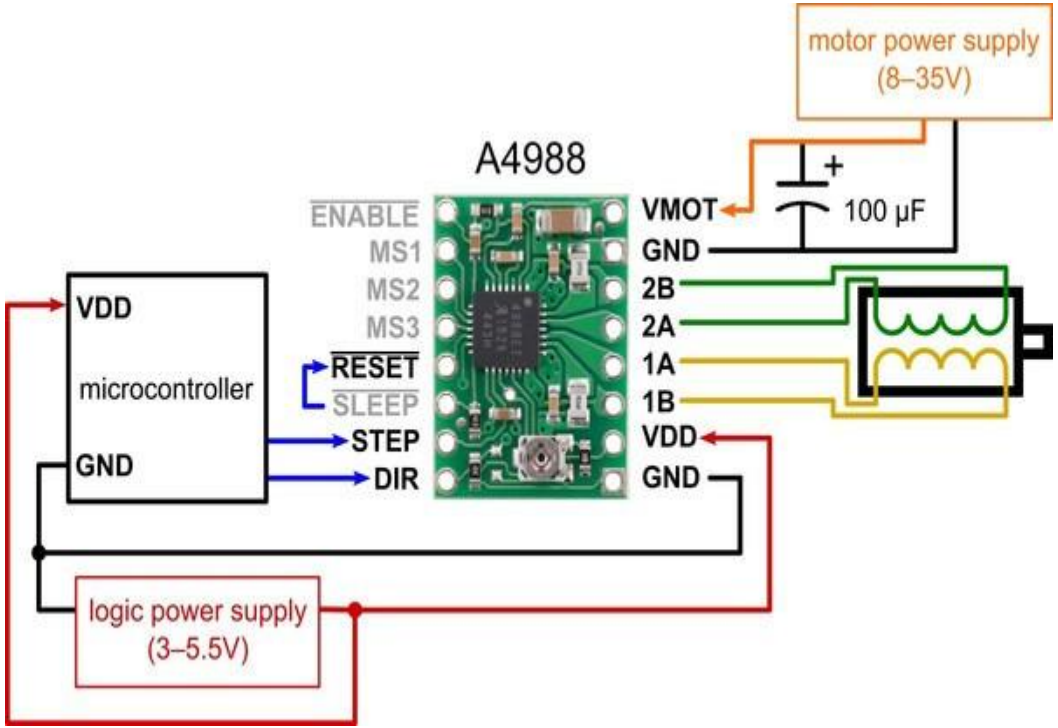
- Direct Drive Extruder- Filament runs directly from extruder to hot end.
- Bowden Tube Extruder-Hot end is connected via PTFE tubing.



# Motor Driver

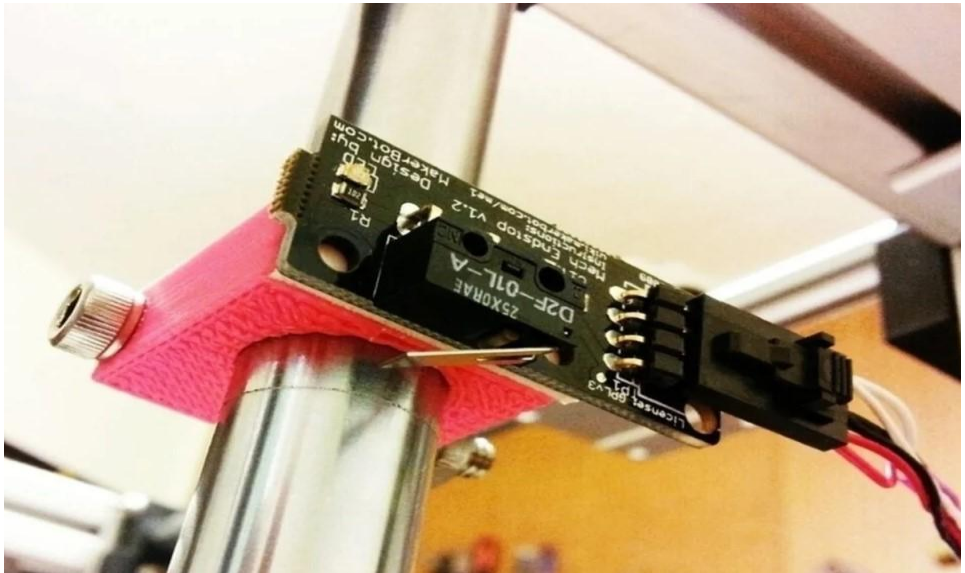
- A stepper motor of a 3D printer need to be precisely controlled to produce a good quality print. One of the components responsible for this function is the stepper motor driver.
- A motor driver is a little current amplifier, the function of motor driver is to take a low current control signal and then turn it into a higher current signal that can drive a motor.
- Using a driver the microcontroller can control the speed and position of the stepper motors while powering the motors directly from the power supply.
- Using a driver the microcontroller can control the speed and position of the stepper motors while powering the motors directly from the power supply.
- Infinite 3D Printers will be using Allegro’s A4988 micro-stepping bi-polar stepper motor driver.




motor power supply
(8–35V)
A4988
+
100 µF
ENABLE
MS1
MS2
MS3
RESET
SLEEP
STEP
DIR
VMOT
GND
2B
2A
1A
1B
VDD
GND
VDD
microcontroller
GND
logic power supply
(3–5.5V)

# END STOPS

- Endstops ensure that a machine, through one of various methods, keeps on object on an axis.
- This stops the object from derailing or jamming at the end of that axis
- Mechanical end-stops are popular choice in 3D printer because they are very inexpensive and simple to use
- In electrical terms,an end-stop is a switch operated by the motion if the machine par as the presence of an object.



# Types of End Stops

- Optical endstops use both infrared or other forms of electromagnetic radiation and an optical sensor to determine the distance between themselves and the objects immediately in the path of their emitted light.
- Mechanical endstops can achieve very accurate results and can be repeated large number of times because they rely on the physical collision of two objects.

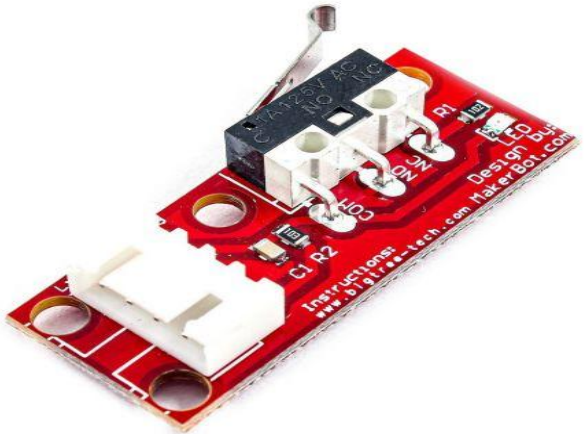



Photo by ElectroPeak

# Resonance Effects

- Every stepper motor has a resonance point, and sometimes motor vibration can affect motor performance and motor life When motor moves continuously, the oscillation of the rotor will come with a frequency. Once the frequency matches motor natural frequency, oscillation will become resonance and causes noise.
- When the printer is printing at speeds which are nearer to the resonant frequency, the printer would vibrate, which in turn can introduce ripples in the print. To test the printer for its behavior in such conditions, we use X-Y resonance and Z-resonance tests.

# References:


https://www.drdflo.com/pages/Guides/How-to-Build-a-3D-Printer/Firmware.html

https://all3dp.com/2/3d-printer-firmware-which-to-choose-and-how-to-change-it/

https://www.youtube.com/watch?v=Ki6Wo7qzdrs

https://www.klipper3d.org/Kinematics.html

https://www.klipper3d.org/Resonance_Compensation.html

https://3dprinterchat.com/how-to-choose-a-3d-printer-firmware/#What_are_the_options

https://www.youtube.com/watch?v=KzSUz7qWyME

https://all3dp.com/2/3d-printer-g-code-commands-list-tutorial/

https://powerbelt3d.com/how-to-orient-models-for-a-conveyor-belt-3d-printer/

https://3dwork.io/en/complete-guide-configure-marlin-2-0-x-from-scratch/

https://reprap.org/wiki/G-code

https://marlinfw.org/docs/configuration/probes.html

https://github.com/zechyc/Tilted-Bed-Conveyor/blob/master/Program.cs

https://powerbelt3d.com/how-to-calibrate-a-conveyor-belt-3d-printer/

https://all3dp.com/2/3d-printer-belt-all-you-need-to-know/

https://all3dp.com/2/easy-g-code-examples-to-begin-with/

https://all3dp.com/2/3d-printer-g-code-commands-list-tutorial/

https://nraynaud.github.io/webgcode/

https://marlinfw.org/docs/configuration/configuration.html

https://powerbelt3d.com/how-to-orient-models-for-a-conveyor-belt-3d-printer/

https://www.youtube.com/watch?v=OaFZRBWn4T0&t=470s

https://www.raise3d.com/download/

https://en.wikipedia.org/wiki/Slicer_(3D_printing)

http://solidutopia.com/marlin-firmware-user-guide-basic/

https://shop.prusa3d.com/en/upgrades/183-original-prusa-i3-mmu2s-upgrade-kit-for-mk25-mk3s.html

https://www.youtube.com/watch?v=0jc8xzotkdA&t=104s

https://all3dp.com/2/octoprint-bed-level-visualizer-guide/

https://www.youtube.com/watch?v=Ki6Wo7qzdrs

https://3dwork.io/en/complete-guide-configure-marlin-2-0-x-from-scratch/