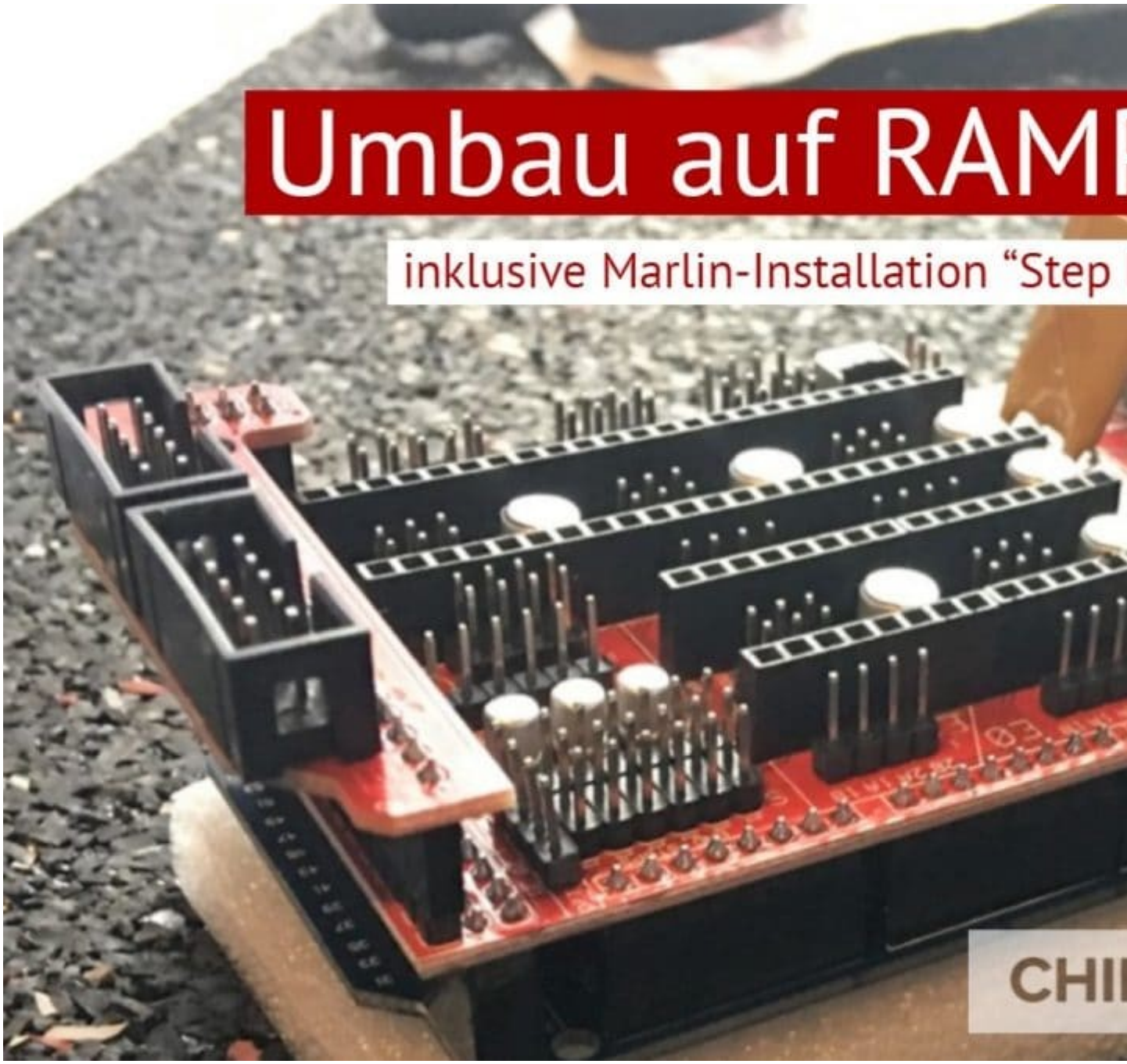
3D Drucker Aufbau / Aufbau auf RAMPS 1.4



in diesem Tutorial möchte ich Euch die Vorteile von RAMPS 1.4 an eurem 3D Drucker näher bringen.
Das RAMPS-Board hat hier noch ein wenig mehr Luft und sollte erstmal noch ausreichen in der aktuellen Board-Version. Mit einem RAMPS 1.4 Setup habt ihr außerdem die Möglichkeit das Heizbett eures 3D Druckers mit 24 Volt zu betreiben.

Inhalt

Benötigte Komponenten

- RAMPS + Arduino-Mega 2560 – die beiden Platinen, Arduino und Stepper Treiber-Board
- Reprap Discount 12864 LCD – vollgrafisches Display mit SD-Kartenleser
- 4 Steppertreiber-Platinen vom Typ A4988 (16 Schritt) ODER 4 Stepper Treiber vom Typ DRV8825 (32 Schritt)

Ich rate Euch zu diesem KIT mit dem LCD 12864 Grafik Smart Display Controller. Hier ist eigentlich Alles an neuen Komponenten enthalten was Ihr benötigt. Und das für einen deutlich günstigeren Preis als ich erzielt habe. –Gab es damals leider noch nicht. Natürlich könnt Ihr Euch auch die Arbeit machen und selber nochmal recherchieren. Dieses KIT ist mit den A4988 Steppern. Ihr könnt natürlich auch direkt auf DRV8825-Stepper. Dann müsst Ihr aber wie in diesem Guide ein Stück von der Standard-Konfiguration abweichen. Ich habe DRV8825-Steppertreiber verbaut. In diesem Guide sind beide Wege beschrieben.

RAMPS 1.4-Kit auf Amazon

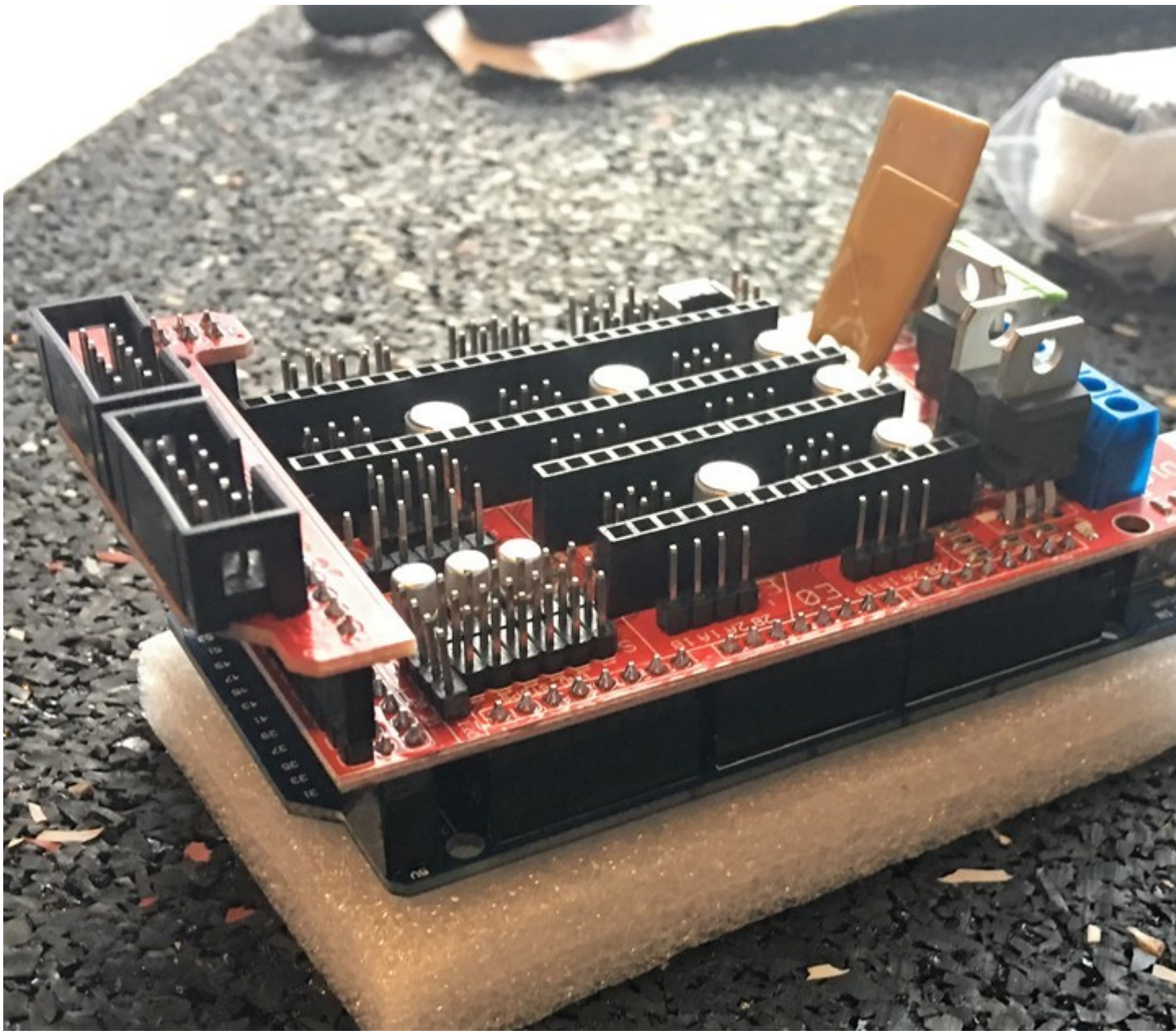
Darüber hinaus sollte eure Werkstatt für einen sauberen Aufbau / Umbau auch folgende Komponenten aufweisen, das meiste habt Ihr vielleicht sowieso:

- Litzen für Kabel-Verlängerung etc.
- Schrumpfschlauch zum ordentlichen Verkabeln.
- Lötkolben etc.
- Multimeter
- Crimpzange und Ader-Endhülsen
- Kabelschuhe
- Dupont-Stecker oder Jumper-Kabel für das RAMPS 1.4 Board
  Luxus-Variante für die Verkabelung – fertig kaufen

Schraubendreher, Seitenschneider und eine Spitz-Zange sehe ich mal als wirklich sowieso vorhanden an. Damit habt Ihr dann aber wirklich Alles und wir können loslegen.

Schritt 1: Zusammenstecken der Komponenten Arduino Mega 2560 und RAMPS 1.4 Shield

Als erstes packt Ihr natürlich die einzelnen Komponenten aus und dann könnt auch schon mal den Arduino mit dem RAMPS-Shield zusammenstecken. Seid dabei vorsichtig, passt eigentlich aber nur in einer Variante sauber zusammen und man kann nichts falsch machen.



RAMPS 1.4 aufgesteckt auf ARDUINO Mega 2560

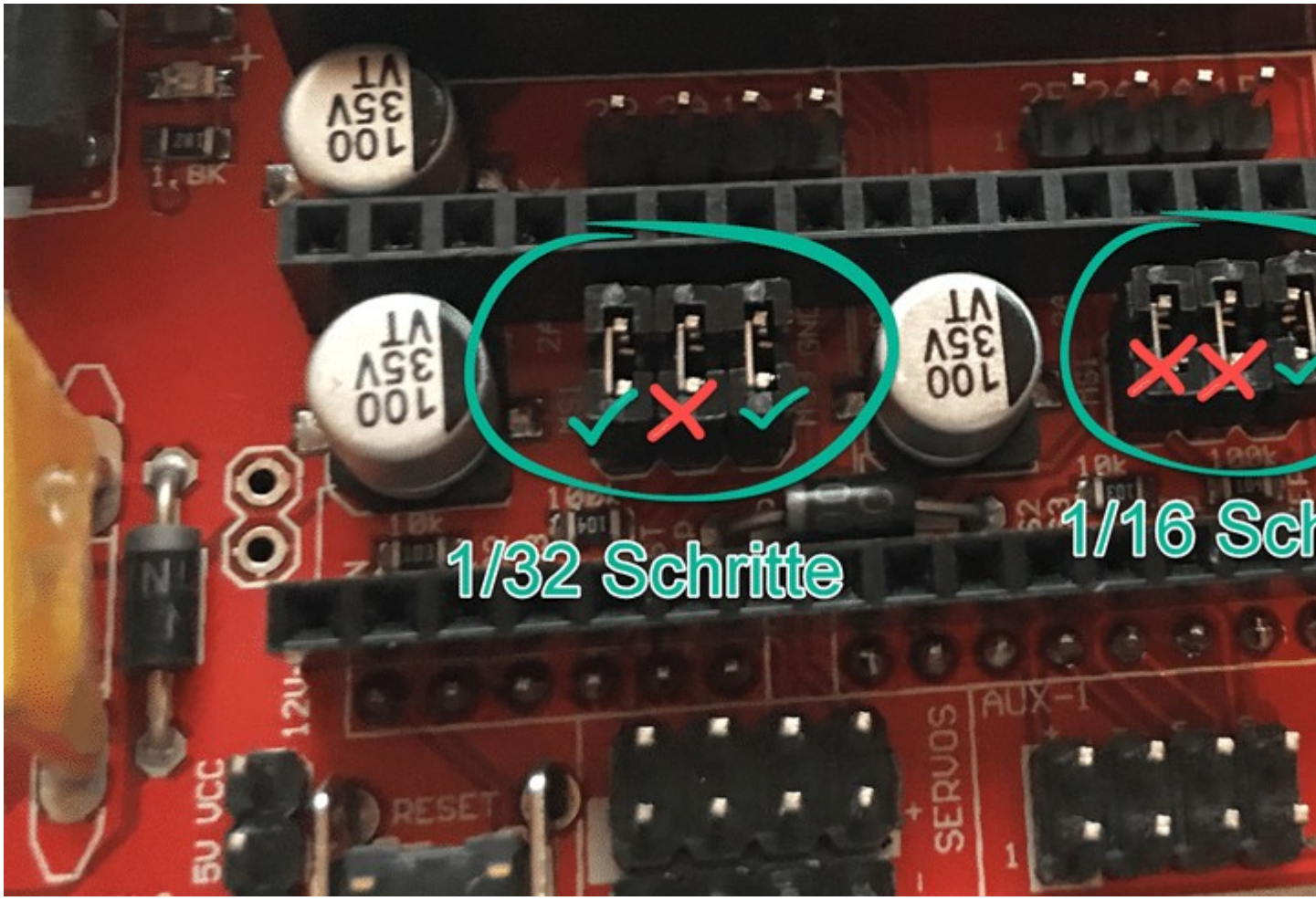
Schritt 2: Jumper stecken für die Stepper Treiber

Im nächsten Schritt stecken wir die Jumper für die Stepper-Treiber richtig. Mit diesen sagt ihr den Stepper Treibern mit wie viel Schritten sie arbeiten sollen. ACHTUNG, wie die Jumper gesteckt werden müssen ist neben euren gewünschten Schritt-Einstellungen davon abhängig, welche Steppertreiber ihr verwendet! Im Reprap-Wiki steht dazu folgendes:

```
jumper Yes/No step size
1 2 3
no no no full step
yes no no half step
```

```
no yes no 1/4 step
yes yes no 1/8 step
no no yes 1/16 step
yes no yes 1/32 step
no yes yes 1/64 step
yes yes yes 1/128 step
```

das sieht dann für uns quasi so aus:

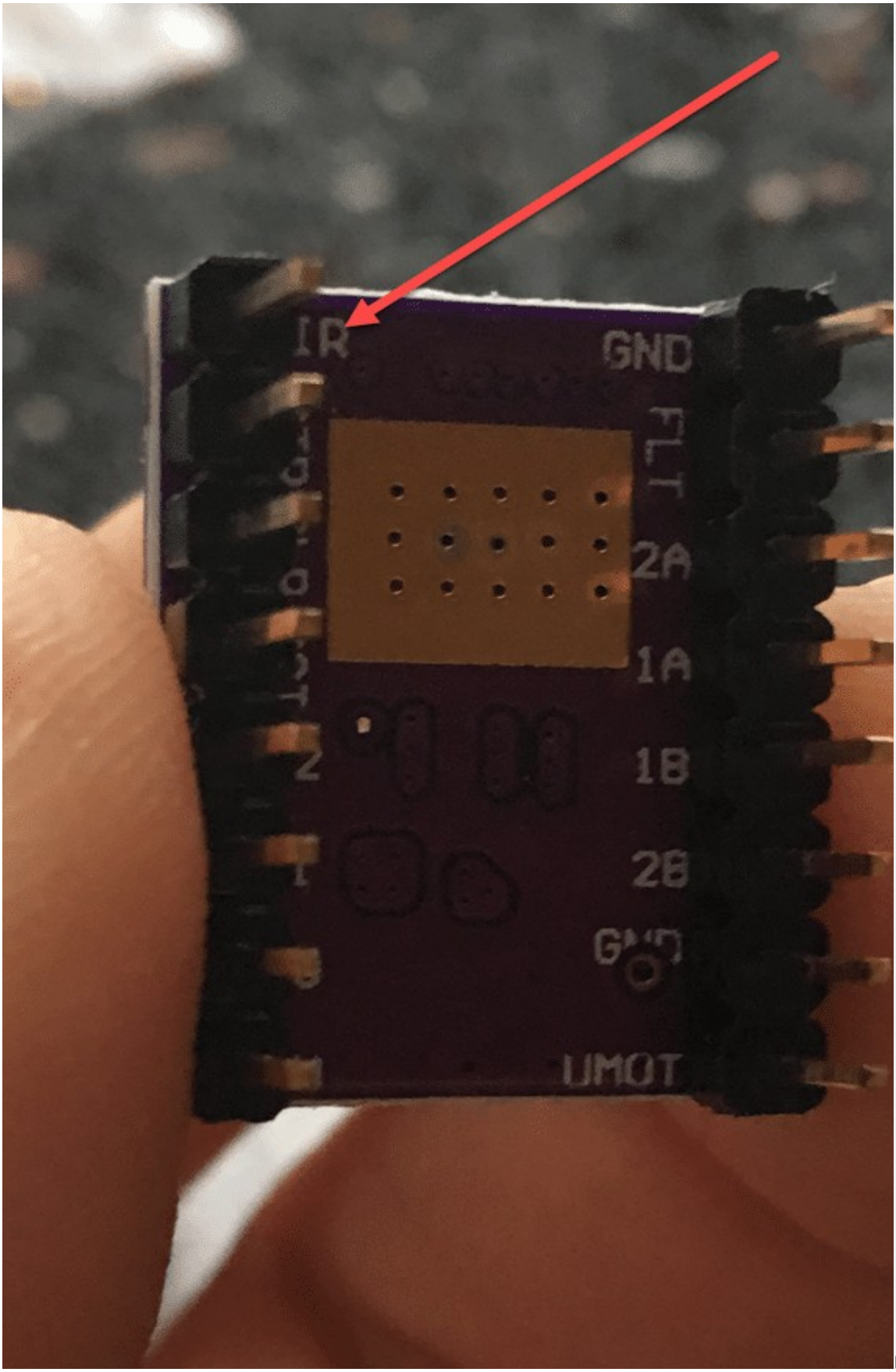


Jumper-Setting für Schrittmotoren-Treiber auf RAMPS 1.4

Aber nochmal, es ist wichtig welche Stepper Treiber Ihr euch zu eurem RAMPS 1.4 besorgt habt. A4988 können nur 1/16 Schritte.

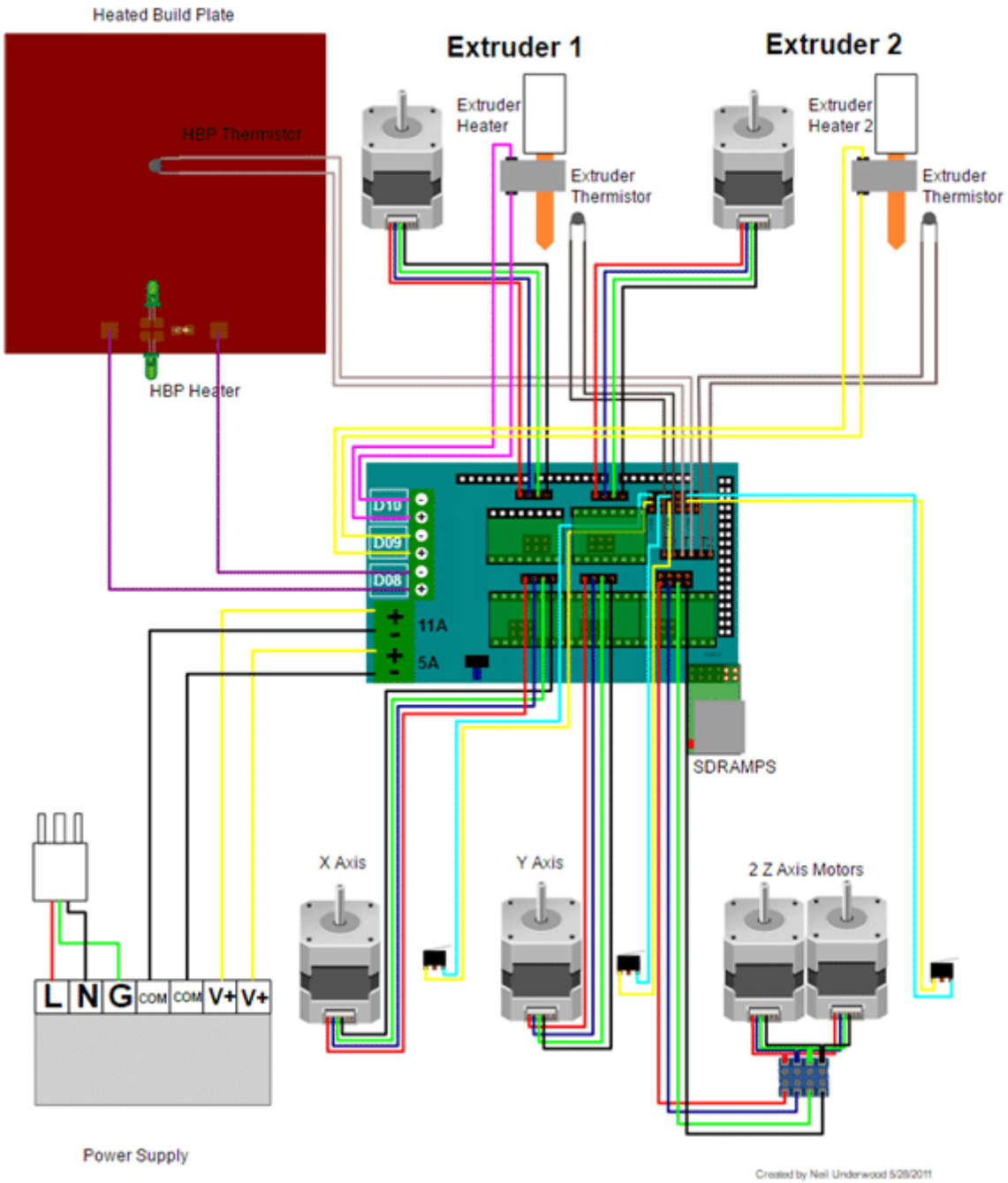
Schritt 3: Aufstecken der Stepper Treiber DRV8825 oder A4988 auf euer RAMPS-Shield

Nachdem Ihr nun die Jumper gesteckt habt könnt Ihr die Stepper Treiber aufstecken, welche später die Jumper die Ihr grade aufgesteckt habt verdecken. Hier ist es besonders wichtig diese korrekt aufzustecken, da sonst RAMPS-Board oder Stepper Treiber beschädigt werden könnten. Ist aber relativ leicht wenn Ihr auf die Markierungen an Stepper Treiber und Board achtet. Ich hab mich dabei an einem äußeren PIN orientiert (DIR) orientiert. Den gibt’s nur einmal..



Schrittmotoren-Treiber

Dann sucht Ihr einfach ebenfalls auf dem Board nach (DIR) auf den Steckplätzen für die Achsen X,Y,Z und den Extruder. Damit solltet Ihr die Stepper Treiber korrekt aufstecken können, seid sorgfältig! Die Z-Achse verwendet für beide Motoren bei RAMPS 1.4 nur einen Treiber was deutlich geschickter ist als bei der Original-Lösung beim Anet A8 Standard-Board. Dafür hat mir dieses Schaltbild aus dem Reprap-Wiki sehr weitergeholfen:



RAMPS 1.4 Schaltbild

Schritt 4: Aufspielen der Firmware und Einstellungen in Configuration.h

Ja Ihr lest richtig. Ich hab jetzt schon die Firmware aufgespielt, weil mein Drucker noch im Gehäuse stand und das ganze momentan ohne Verkabelung noch wunderbar mobil ist. Strom bekommen Arduino und RAMPS 1.4 dabei vom PC über das USB-Kabel. Hat wunderbar funktioniert. Als Firmware habe ich mich für MARLIN entschieden da mir Konfiguration und Features (Auto-Leveling) eigentlich zusagen. Marlin besorgt Ihr euch am besten direkt an der Quelle auf Github. Nachdem wir das ganze entpackt haben, widmen wir uns der Configuration.h und gehen die einmal von oben nach unten in Ruhe durch. Ich hab dafür eine Flasche Bier aufgemacht… Bitte seht die Zeilen-Nummern im nächsten Teil relativ. Kann sein das weicht bei Euch ein Stück ab, das ist aber für die Konfiguration völlig egal. Ich nehme meine Zeilen-Nummern mal als Referenz für diesen Artikel, so wisst Ihr ungefähr wo Ihr suchen müsst.
Ich habe die Dateien in einem externen Editor (Notepad++) bearbeitet. Ihr könnt das aber auch direkt in der ARDUINO IDE erledigen. Für das aufpsielen auf euer RAMPS1.4 benötigt Ihr diese sowieso. Ich habe die Version 18.2 genommen. Wenn Ihr in der Arduino IDE arbeitet, öffnet Ihr direkt in dem Ordner wo sich Marlin befindet die Datei „Marlin.ino“. Alle weiteren zugehörigen Dateien werden dann automatisch geöffnet.

Baudrate und RAMPS 1.4 Board definieren

Als erstes kümmern wir uns um die Deklaration der Baudrate bei ca. Zeile 125. Ich habe immer 115200 genommen und bin damit gut gefahren.

```
125   #define BAUDRATE 115200
```

In Zeile 133 sagen wir Marlin welche Board-Konfiguration wir verwenden. Man kann das RAMPS 1.4 mit unterschiedlichen Konfigurationen betreiben. Wenn man zum Beispiel einen Dual-Extruder verwendet, weicht die Konfiguration hier. Für einen 3D Drucker ähnlich dem Anet A8 ab Werk habt Ihr einen Extruder, einen Bridging-Fan sowie ein beheitztes Bett. Die passende Konfiguration für die Kombi lautet „E(xtruder)F(an)B(ed)“. Wir definieren also folgendes:

```
132   #ifndef MOTHERBOARD
133     #define MOTHERBOARD BOARD_RAMPS_14_EFB
134   #endif
135
```

Drucker-Namen vergeben

Wenn eurem Drucker einen Namen vergeben wollt könnt Ihr das in Zeile ca. 138 tun:

```
136  // Optional custom name for your RepStrap or other custom machine
137  // Displayed in the LCD "Ready" message
138  #define CUSTOM_MACHINE_NAME "Druckername"
```

Extruder-Anzahl und Temperatur-Sensoren

In Zeile 146 könnt Ihr die Extruder-Anzahl bestimmen. Standardmäßig gehört hier also eine 1 hin.

```
144  // This defines the number of extruders
145  // :[1, 2, 3, 4]
146  #define EXTRUDERS 1
```

In Zeile ca. 252 definiert Ihr die Temperatur-Fühler mit den eurer Drucker arbeiten soll. Standardmäßig werdet Ihr einen für Extruder und einen für das beheitze Bett haben. Dann sieht die Konfiguration wie folgt aus:

```
252  #define TEMP_SENSOR_0 1
253  #define TEMP_SENSOR_1 0
254  #define TEMP_SENSOR_2 0
255  #define TEMP_SENSOR_3 0
256  #define TEMP_SENSOR_BED 1
```

Minimal-Temperatur und Maximal-Temperatur für Sensoren definieren

An der Stelle um Zeile 280 könnt Ihr Mindest-Temperaturen definieren, welche mindestens erreicht sein müssen damit euer Extruder beim Start vom Aufheizen überhaupt mit Heizen beginnt. Das ist eine klasse Sache und stellt sicher, dass eure Thermistor-Verkabelung richtig ist. Standardmäßig sind dort 5 Grad eingestellt. Könnt Ihr gerne höher stellen aber ich habs so gelassen. Jedenfalls eine schicke Sache dass die Firmware hier prüft.

```
280  #define HEATER_0_MINTEMP 5
281  #define HEATER_1_MINTEMP 5
282  #define HEATER_2_MINTEMP 5
283  #define HEATER_3_MINTEMP 5
284  #define BED_MINTEMP 5
```

Um Zeile 289 könnt Ihr die maximale Temperatur die eingestellt werden kann für Extruder und Heizbett definieren. Solltet Ihr auf jedenfall tun, macht das ganze sicherer. Standardmäßig sind 275 Grad für Extruder und 130 Grad für das Druckbett definiert.

```
289 #define HEATER_0_MAXTEMP 275
290 #define HEATER_1_MAXTEMP 275
291 #define HEATER_2_MAXTEMP 275
292 #define HEATER_3_MAXTEMP 275
293 #define BED_MAXTEMP 130
```

PID-Tuning und Thermal Runaway aktivieren für Heizbett

Im Bereich um Zeile 345 könnt Ihr das PID-Tuning für euer Heizbett aktivieren. Solltet Ihr am besten gleich miterledigen. Damit könnt Ihr später das Heizen des Betts kalibrieren.

```
345 #define PIDTEMPBED
```

Marlin hat eine Funktion eingebaut die sicherstellen soll, dass Euch nicht die Bude abfackelt wenn ein Thermistor ausfällt. Stellt sicher dass Ihr die folgenden Zeilen nicht auskommentiert habt.

```
404 #define THERMAL_PROTECTION_HOTENDS // Enable thermal protection for all extruders
405 #define THERMAL_PROTECTION_BED     // Enable thermal protection for the heated bed
```

Endstop Settings

In Zeile 434-436 sagen wir der Firmware, das wir Endstops für alle 3 Achsen verwenden und das jeweils an der Minimal-Position. Also Ausgangsposition.

```
434 #define USE_XMIN_PLUG
435 #define USE_YMIN_PLUG
436 #define USE_ZMIN_PLUG
```

Schritte der Achse / Steppermotoren einstellen

In Zeile 489 definieren wir die Schritte für die Achsen die nötig sind, um einen Millimeter zu fahren.
Verwendet Ihr A4988 mit 1/16 Schritten sieht das standardmäßig folgendermaßen aus:

```
485  * Default Axis Steps Per Unit (steps/mm)
486  * Override with M92
487  *                                      X, Y, Z, E0 [, E1[, E2[, E3]]]
488  */
489 #define DEFAULT_AXIS_STEPS_PER_UNIT   { 100, 100, 400, 95 }
```

Wenn Ihr DRV8825 mit konfigurierten 1/32 Schritten verwendet (dazu müsst Ihr auch für 1/32 die Jumper gesteckt haben, siehe oben), solltet Ihr mit diesen Werten starten:

```
485  * Default Axis Steps Per Unit (steps/mm)
486  * Override with M92
487  *                                      X, Y, Z, E0 [, E1[, E2[, E3]]]
488  */
489 #define DEFAULT_AXIS_STEPS_PER_UNIT   { 200, 200, 800, 190 }
```

Max Feed Rate definieren

In Zeile 496 definieren wir die Max Feed Rate:

```
491  /**
492   * Default Max Feed Rate (mm/s)
493   * Override with M203
494   *                                      X, Y, Z, E0 [, E1[, E2[, E3]]]
495   */
496  #define DEFAULT_MAX_FEEDRATE          { 250, 250, 8, 25 }
```

Maximale Beschleunigung und Jerk-Settings

An dieser Stelle sagen wir der Firmware, wie schnell die Achsen maximal beschleunigt werden dürfen. Diese Einstellungen sind global und sagen, dem Drucker die Achsen niemals schneller zu bewegen. Ich habe einen verstärkten Rahmen und habe deshalb recht hohe Werte. Solltet Ihr einen Acryl-Rahmen verwenden rate ich zu niedrigeren Maximal-Werten von 400 mm /s. Am besten mal niedrig starten und schrittweise anheben wenn Ihr mit dem Aufbau fertig seid und ein paar Testdrucke macht. auch bei Default_Max_Acceleration solltet Ihr eher nach unten Schrauben wenn Ihr in Geschwindigkeitsprobleme lauft. Auch hier sind im folgendem Bild die Standardwerte.

```
498  /**
499   * Default Max Acceleration (change/s) change = mm/s
500   * (Maximum start speed for accelerated moves)
501   * Override with M201
502   *                                    X, Y, Z, E0 [, E1[, E2[, E3]]]
503   */
504  #define DEFAULT_MAX_ACCELERATION      { 3000, 3000, 100, 10000 }
505
506  /**
507   * Default Acceleration (change/s) change = mm/s
508   * Override with M204
509   *
510   *   M204 P    Acceleration
511   *   M204 R    Retract Acceleration
512   *   M204 T    Travel Acceleration
513   */
514  #define DEFAULT_ACCELERATION          400    // X, Y, Z and E acceleration for prin
515  #define DEFAULT_RETRACT_ACCELERATION  3000    // E acceleration for retracts
516  #define DEFAULT_TRAVEL_ACCELERATION   3000    // X, Y, Z acceleration for travel (n
517
518  /**
```

Die Jerk-Settings sagen der Firmware welches der minimale Geschwindigkeits-Unterschied zwischen Bewegungen ist, welche eine Beschleunigung erforderlich machen. Leichte Schwankungen in der Geschwindigkeit werden somit ignoriert. Folgende Werte sind standardmäßig für den Anet A8 hintelegt. Ich würde Euch raten diese Werte zu halbieren.

```
518  /**
519   * Default Jerk (mm/s)
520   *
521   * "Jerk" specifies the minimum speed change that requires acceleration.
522   * When changing speed and direction, if the difference is less than the
523   * value set here, it may happen instantaneously.
524   */
525  #define DEFAULT_XJERK                20.0
526  #define DEFAULT_YJERK                20.0
527  #define DEFAULT_ZJERK                 0.4
528  #define DEFAULT_EJERK                 5.0
```

Max-Geschwindigkeit und Travel-Limits für das Homing

Bei ca. Zeile 886 definiert Ihr die Geschwindigkeit fürs Homing von X und Y Achse

```
585  // X and Y axis travel speed (mm/m) between probes
586  #define XY_PROBE_SPEED 8000
```

So um Zeile 720 definiert Ihr die Travel-Limits für eure Achsen. Sprich, wie weit die Achsen gefahren werden können. Je nach eurem Aufbau könnt Ihr hier später noch Fein-Tuning betreiben. Wenn Ihr euren Bau-Raum voll ausreizen wollt, müsst Ihr hier sogar anfassen. Gute und sichere Startwerte sind:

```
719  // Travel limits after homing (units are in mm)
720  #define X_MIN_POS 0
721  #define Y_MIN_POS 0
722  #define Z_MIN_POS 0
723  #define X_MAX_POS 200
724  #define Y_MAX_POS 200
725  #define Z_MAX_POS 200
726
```

Filament-Runout Sensor

So um Zeile 729 könnt Ihr euren Filament-Runout-Sensor konfigurieren wenn Ihr euch einen gebaut habt. Diesen Punkt sehe ich mal als optional und werde Ihn in einem anderem Artikel beschreiben ähnlich dem Auto-Leveling.

Auto-Leveling / Auto Bed-Leveling

Von oben nach unten gesehen ist jetzt das Auto-Leveling in der Configuration.h dran. Ich werde in diesem Artikel nicht darauf eingehen da ich einen sehr beliebten gesonderten Artikel geschrieben habe.

EEPROM aktivieren

Wenn Ihr nicht wegen jeder Einstellung den Drucker neu mit der Firmware bespielen wollt bietet MARLIN EEPROM-Settings. Hier könnt Ihr im Betrieb Werte in der Konfiguration auslesen, überschreiben und neu abspeichern. Das solltet Ihr in jeden Fall aktivieren.

```
898  // EEPROM
899  //
900  // The microcontroller can store settings in the EEPROM, e.g. max velocity...
901  // M500 - stores parameters in EEPROM
902  // M501 - reads parameters from EEPROM (if you need reset them after you changed t
903  // M502 - reverts to the default "factory settings".  You still need to store them
904  //define this to enable EEPROM support
905  #define EEPROM_SETTINGS
```

Temperaturen für Preheat via Menü

Wahrscheinlich kennt Ihr diese Einstellungen schon aus dem Menü eures Druckers. In der Firmware können Standard-Temperatur-Kombinationen eingestellt werden die später im Menü auswählbar sind. E.g. „Preheat PLA“ und „Preheat ABS“ Die Einstellungen dazu finder Ihr bei ca. Zeile 938.

```
938  // Preheat Constants
939  #define PREHEAT_1_TEMP_HOTEND 205
940  #define PREHEAT_1_TEMP_BED     60
941  #define PREHEAT_1_FAN_SPEED     0 // Value from 0 to 255
942
943  #define PREHEAT_2_TEMP_HOTEND 220
944  #define PREHEAT_2_TEMP_BED    65
945  #define PREHEAT_2_FAN_SPEED     0 // Value from 0 to 255
946
```

Menü-Sprache festlegen

In Zeile 1064 könnt Ihr die Menüsprache festlegen. Da die Übersetzungen sehr gut sind, hab ich deutsch gewählt.

```
1055  // LCD LANGUAGE
1056  //
1057  // Here you may choose the language used by Marlin on the LCD menus, the followi
1058  // list of languages are available:
1059  //    en, an, bg, ca, cn, cz, de, el, el-gr, es, eu, fi, fr, gl, hr, it,
1060  //    kana, kana_utf8, nl, pl, pt, pt_utf8, pt-br, pt-br_utf8, ru, tr, uk, test
1061  //
1062  // :{ 'en':'English', 'an':'Aragonese', 'bg':'Bulgarian', 'ca':'Catalan', 'cn':'
      'el-gr':'Greek (Greece)', 'es':'Spanish', 'eu':'Basque-Euskera', 'fi':'Finnish',
      'kana':'Japanese', 'kana_utf8':'Japanese (UTF8)', 'nl':'Dutch', 'pl':'Polish', '
      'pt-br_utf8':'Portuguese (Brazilian UTF8)', 'pt_utf8':'Portuguese (UTF8)', 'ru':'
1063  //
1064  #define LCD_LANGUAGE de
```

RAMPS 1.4 SD-Karten Support

Wenn Ihr ein Display mit SD-Kartenslot genommen habt. Wie oben beschrieben z. B. das Reprap Discount 12864 LCD, dann müsst Ihr hier der Firmware sagen dass Ihr einen SD-SLOT habt und Ihn verwenden wollt. Das findet Ihr in Zeile 1109

```
1104  // SD CARD
1105  //
1106  // SD Card support is disabled by default. If your controller has an SD slot,
1107  // you must uncomment the following option or it won't work.
1108  //
1109  #define SDSUPPORT
```

Individuelles Menü für das Achsen-Homing

Wenn Ihr jede Achse dediziert nullen können wollt, dann braucht Ihr dafür das individuelle Menü. Dieses könnt Ihr optional in Zeile 1172 aktivieren.

```
1167  //
1168  // Individual Axis Homing
1169  //
1170  // Add individual axis homing items (Home X, Home Y, and Home Z) to the LCD menu.
1171  //
1172  #define INDIVIDUAL_AXIS_HOMING_MENU
1173
```

„Biepser“ aktivieren an eurem Display

Wenn euer Display einen kleinen Speaker verbaut hat, könnt Ihr mit Marlin die akustischen Feedbacks verwenden. Dazu müsst Ihr die folgenden Zeilen aktivieren. Das Signal lässt sich in Frequenz und Dauer einstellen. In habe folgende Werte verwendet:

```
1175  // SPEAKER/BUZZER
1176  //
1177  // If you have a speaker that can produce tones, enable it here.
1178  // By default Marlin assumes you have a buzzer with a fixed frequency.
1179  //
1180  #define SPEAKER
1181
1182  //
1183  // The duration and frequency for the UI feedback sound.
1184  // Set these to 0 to disable audio feedback in the LCD menus.
1185  //
1186  // Note: Test audio output with the G-Code:
1187  //  M300 S<frequency Hz> P<duration ms>
1188  //
1189  #define LCD_FEEDBACK_FREQUENCY_DURATION_MS 5
1190  #define LCD_FEEDBACK_FREQUENCY_HZ 1000
```

Display in Marlin konfigurieren

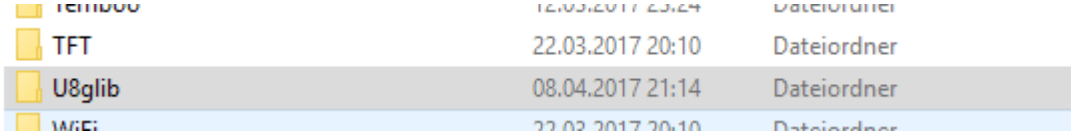
Ihr müsst Marlin außerdem noch mitteilen welches Display Ihr verwendet. Wenn Ihr das 12864 LCD verwendet, müsst Ihr die Zeile 1267 aktivieren.

```
1264  // RepRapDiscount FULL GRAPHIC Smart Controller
1265  // http://reprap.org/wiki/RepRapDiscount_Full_Graphic_Smart_Controller
1266  //
1267  #define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER
```

Wenn Ihr das LCD 2004 habt, wie zum Beispiel in dem KIT von oben. Dann müsst Ihr Zeile 1249 aktivieren.

```
1248  // RepRapDiscount Smart Controller.
1249  // http://reprap.org/wiki/RepRapDiscount_Smart_Controller
1250  //
1251  // Note: Usually sold with a white PCB.
1252  //
1253  #define REPRAP_DISCOUNT_SMART_CONTROLLER
```

Damit seid Ihr dann fertig fürs erste mit der Firmware-Konfiguration und Ihr könnt speichern und das ganze kompilieren. Wenn Ihr ein Vollgrafik-Display verwendet braucht Ihr für eure ARDUINO-Umgebung noch weitere Bibliotheken. Mit der U8glib bekommt Ihr das ganze zum Laufen. Dafür ladet Ihr diese Bibliothek. Diese entpackt Ihr und legt Sie in der Installation der Arduino IDE in das Verzeichnis „libraries„.



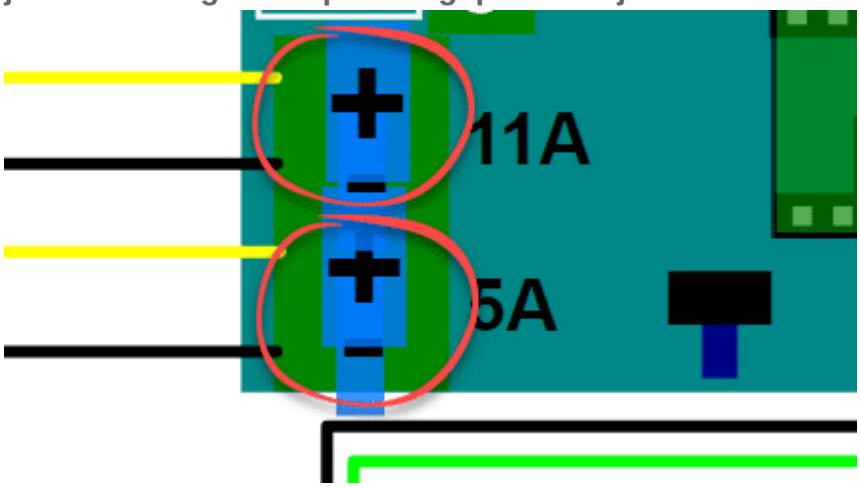
Wenn Ihr die Firmware kompilieren könnt ohne Fehler, wählt Ihr bei Devices das Arduino Mega 2560 aus und den entsprechenden COM-Port wo sich euer Arduino befindet. Bei mir war es COM-3. Das kann / wird bei Euch anders sein. Am besten alles unnötige USB ausstecken dann werden die Auswahlmöglichkeiten schnell kleiner und Ihr findet euer ARDUINO mit dem RAMPS schneller. Nach dem Aufspielen der Firmware auf euer RAMPS 1.4 solle Marlin auch schon booten. Wenn Ihr das Display angeschlossen habt, solltet Ihr das ganze schon fertig sehen.

Schritt 5: Anschließen der Komponenten

Nun müsst Ihr noch dafür sorgen das Alles richtig mit dem RAMPS 1.4 verkabelt ist. Dazu gehen wir Schritt für Schritt die einzelnen Komponenten durch.

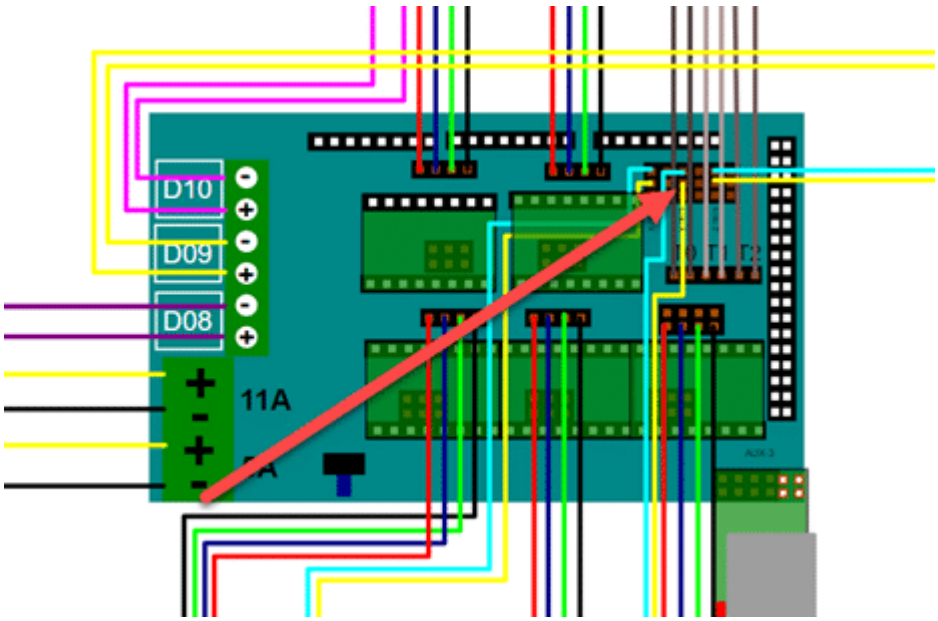
Stromversorgung anschließen RAMPS 1.4 vom Ausgang 12 Volt Netzteil

Das ist schnell erledigt. Führt den Zuleitungen von eurem Netzteil ausgehend jeweils ein eigenes Spannungspaar von jeweils einem Netzteil-Ausgang zu.

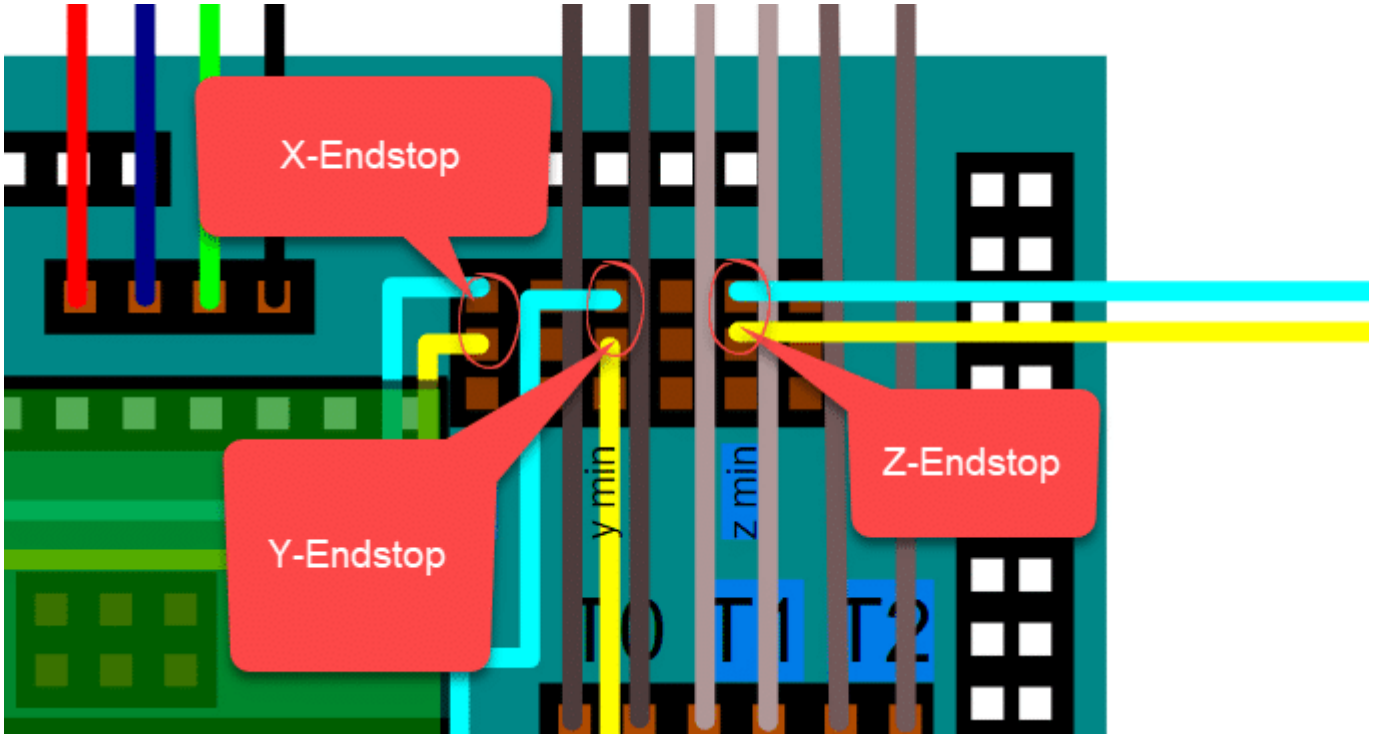


Endstops anschließen

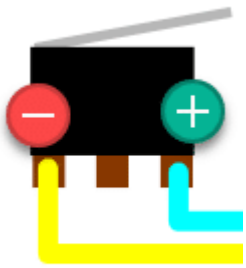
Dazu ist diese Stelle auf dem Board interessant.



Ihr solltet dafür am besten außerdem einen Multimeter verwenden um eure Verkabelung und die Schaltung zu kontrollieren. Im Detail sieht die Verkabelung folgendermaßen aus:



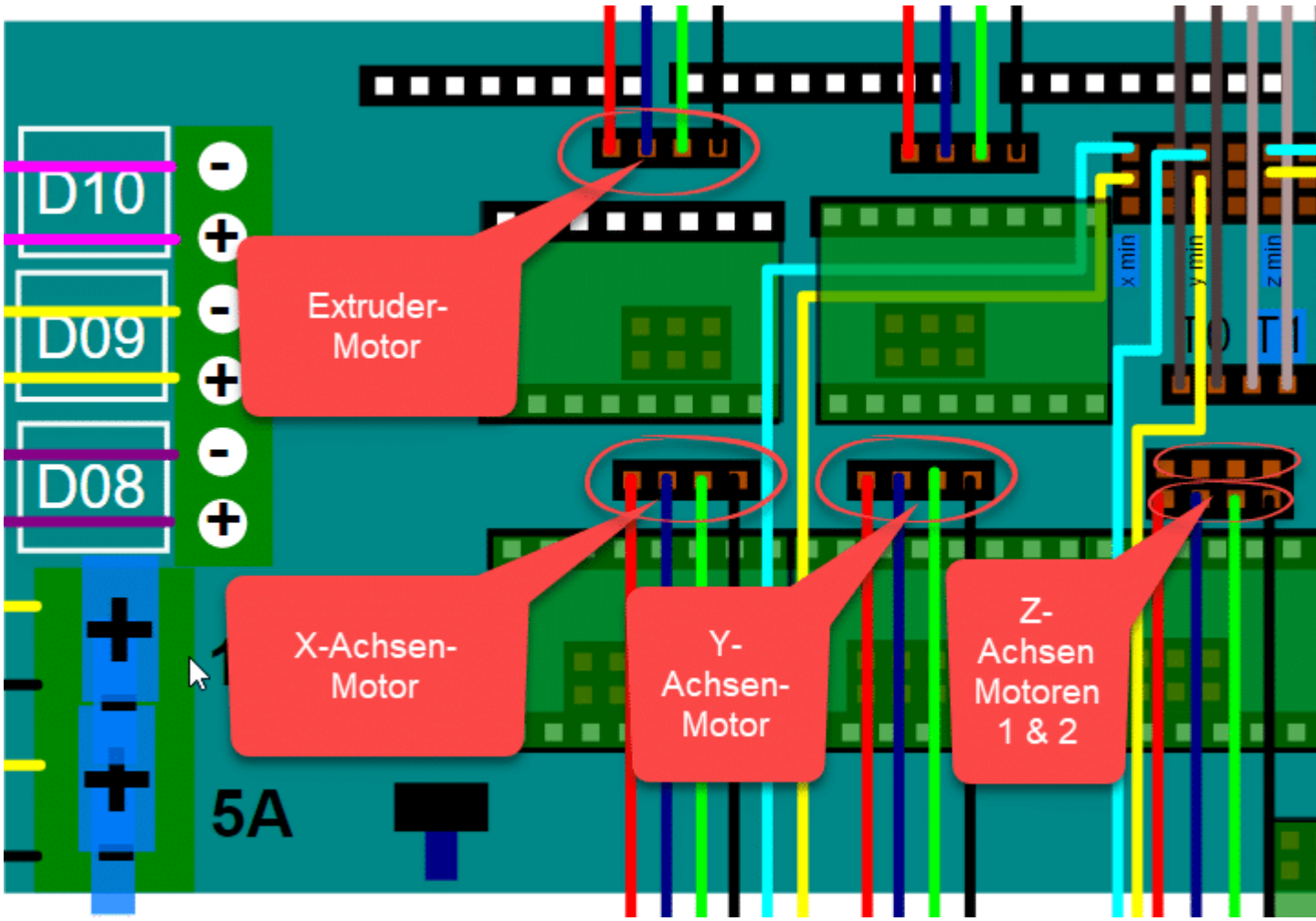
Zieht genügend lange Kabel weg vom Board die später bis zu euren jeweiligen Endstops reichen. Die Endstops verkabelt Ihr wie folgt:



Das Schaltverhalten solltet Ihr aber nochmal mit dem Multimeter prüfen anschließend. Bei betätigten Endstop solltet Ihr eine Spannung von 5 V Volt messen. Gemessen an verbunden Kontakten des Endstops. Bei nicht betätigten Endstop darf keine Spannung anliegen. Das macht Ihr für alle 3 Achsen wie beschrieben und schon ist das Thema Endstops erledigt.

Stepper-Motoren anschließen und Referenz-Spannung einstellen

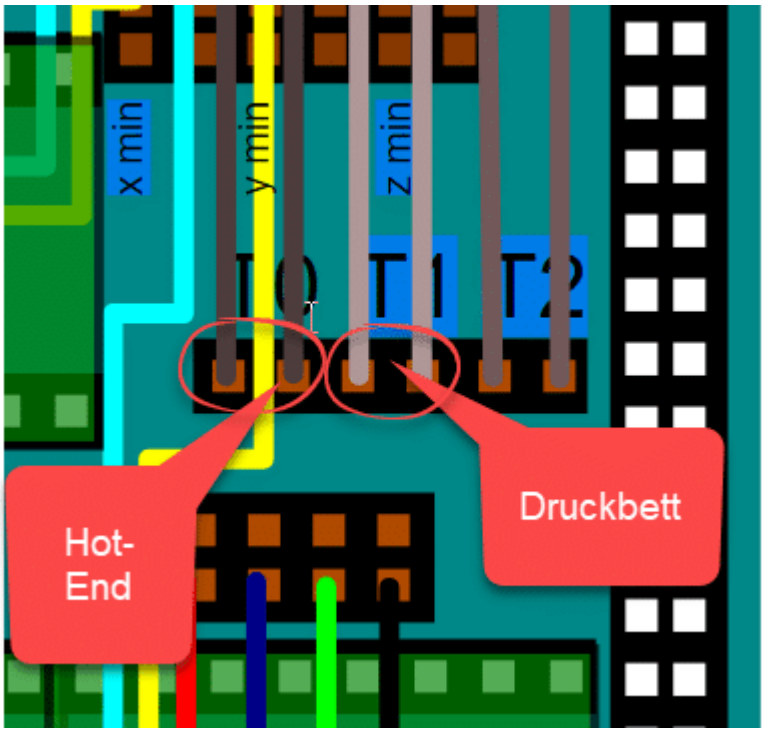
Bevor Ihr die Motoren auf dem RAMPS 1.4 anschließt solltet Ihr die Referenz-Spannung richtig einstellen. Dazu braucht Ihr in jedem Fall ein Messgerät (e.g. Multimeter) wie oben beschrieben. Wie Ihr die Referenz-Spannung einstellt ist hier bereits sehr schön beschrieben. Ein Anet-Motor arbeitet mit maximal 0,9 Ampere laut Gearbest. Sprich Ihr solltet die Referenzspannung auf 0,45 Volt einstellen. Wie Ihr das messt ist schön in dem Artikel beschrieben.
Danke an User Andreas dass ich das ursprünglich im Artikel vergessen hatte!
Motoren anschließen ist für alle Achsen und den Extruder schnell erledigt. Mit Ausnahme der Z-Achse.
Für X, Y und Extruder könnt Ihr eure vorhandene Verkabelung nehmen vom Anet A8. Für die Z-Achse ist der Platz auf der Platine leider nicht ausreichend für die Standard-Stecker.



Ignoriert am besten die Farben eurer vorhandenen Kabel. Wichtig ist es gibt 2 Spannungspaare pro Motor. Standardmäßig sind die beim Anet A8 auf dem Stecker schon richtig angeordnet, außen und ein paar innenliegend. Damit Ihr wisst wie die Stecker draufgehören könnt Ihr das mit einem Motor einfach ausprobieren. Schließt dazu einen Motor mit dem vorhanden Stecker an und schaut ob sich richtig herum dreht wenn Ihr Ihn per Menü bewegt. Wenn er sich falsch herum bewegt, dreht den Stecker einfach um. Das geht für alle Motoren so.
Für die beiden Z-Motoren nehmt Ihr die Dupont-Stecker oder die Jumperkabel um die richtige Verkabelung zu erreichen. Ist ein wenig fummelig aber geht ohne Probleme.

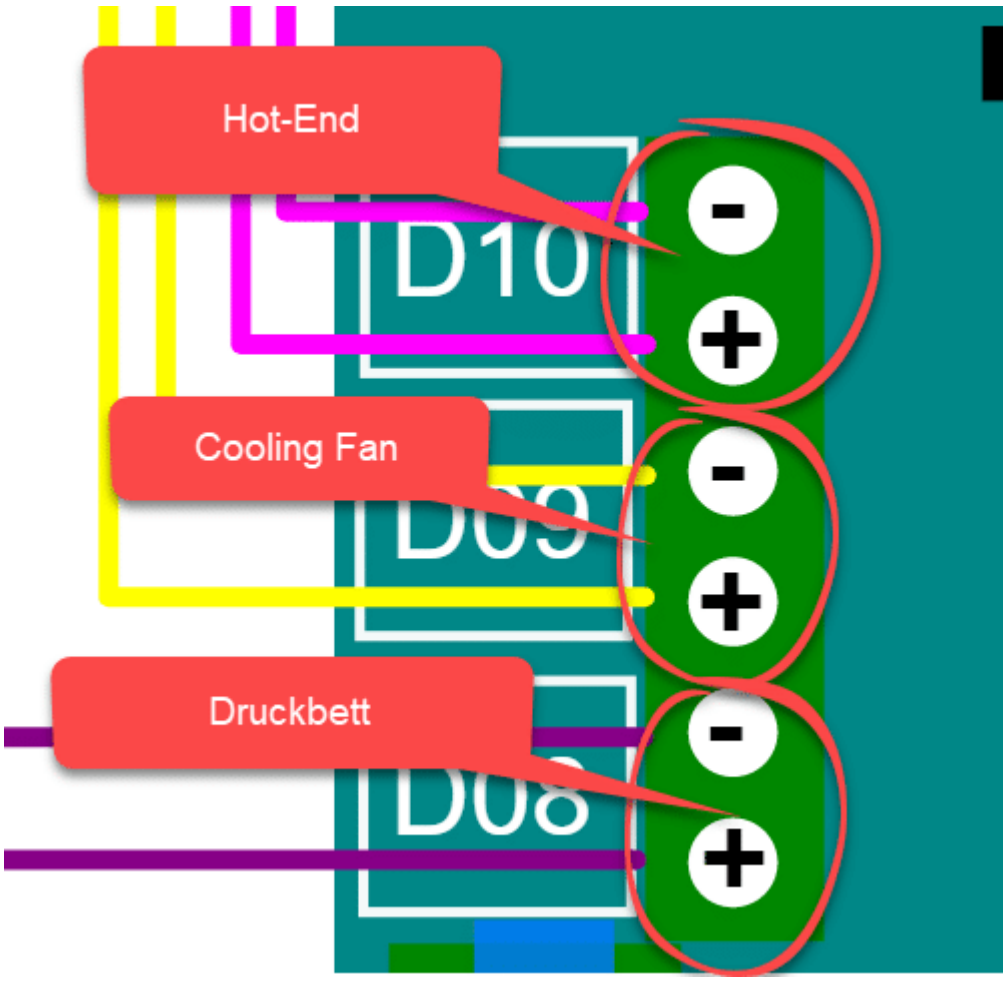
Thermistor von Hot-End und Heizbett anschließen

Da kann man nichts falsch machen. Die Thermistore sind nur Widerstände und haben keine Polarität. Jetzt einfach an den vorgesehen Stellen verkabeln:



Extruder, Heizbett und Cooling Fan verbinden

So jetzt müsst Ihr noch den Extruder das Druckbett und euren Cooling Fan verbinden. Das macht Ihr folgendermaßen:



Damit habt Ihr alle Komponenten verbunden.Nun macht Ihr am Besten eine provisorische Aufräumaktion damit alles einigermaßen ordentlich verkabelt und verstaut ist und startet einen Testdruck. Danach solltet Ihr das Ganze noch ordentlich an eurem Drucker anbringen.