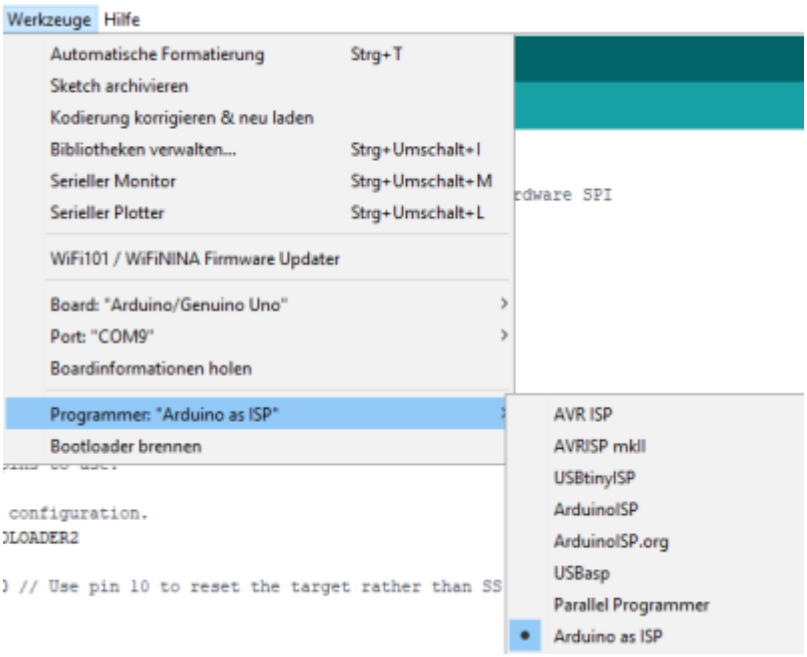
Bootloader programmieren

Arduino Uno als Programmer

Aufbauend auf den Arduino-Seiten



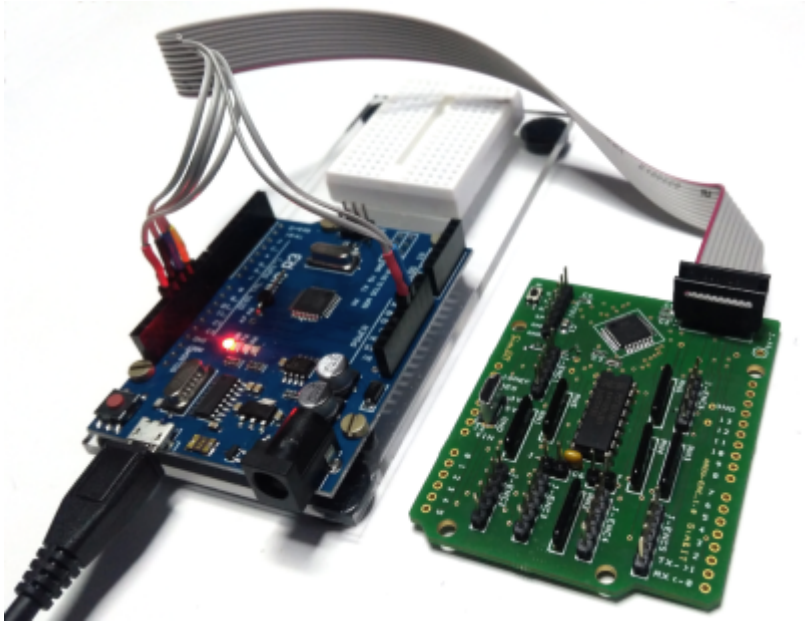
- ArduinoISP
- ArduinoToBreadboard

habe ich den Encoder-Controller folgendermaßen programmiert:

- Herstellen eines Verbindungskabels von den Arduino-Pins (am Controller Steckverbinder J1)
  - `Arduino 5 V → 5 V (J1-Pin9)`
  - `Arduino D10 → RESET (J1-Pin5)`
  - `Arduino D11 → MOSI (J1-Pin6)`
  - `Arduino D12 → MISO (J1-Pin8)`
  - `Arduino D13 → SCK (J1-Pin2)`
  - `Arduino GND → GND (J1-Pin7)`
- Sketch in einen Arduino Uno laden: Datei → Beispiele → ArduinoISP
- Board und Port auswählen, auf das der **ArduinoISP** installiert werden soll
  - Werkzeuge → Board → Arduino/Genuino Uno (Beispiel)
  - Werkzeuge → Port ...
- Sketch → Hochladen

Jetzt ist der Arduino als Programmer nutzbar.

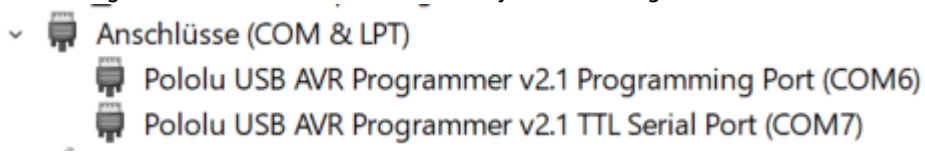
- Board und Port auswählen, auf das der **Bootloader** installiert werden soll
  - Werkzeuge → Board → Arduino Pro or Pro Mini (Beispiel)
  - Werkzeuge → Port ...
- Werkzeuge → Programmer → Arduino as ISP
- Werkzeuge → Bootloader brennen



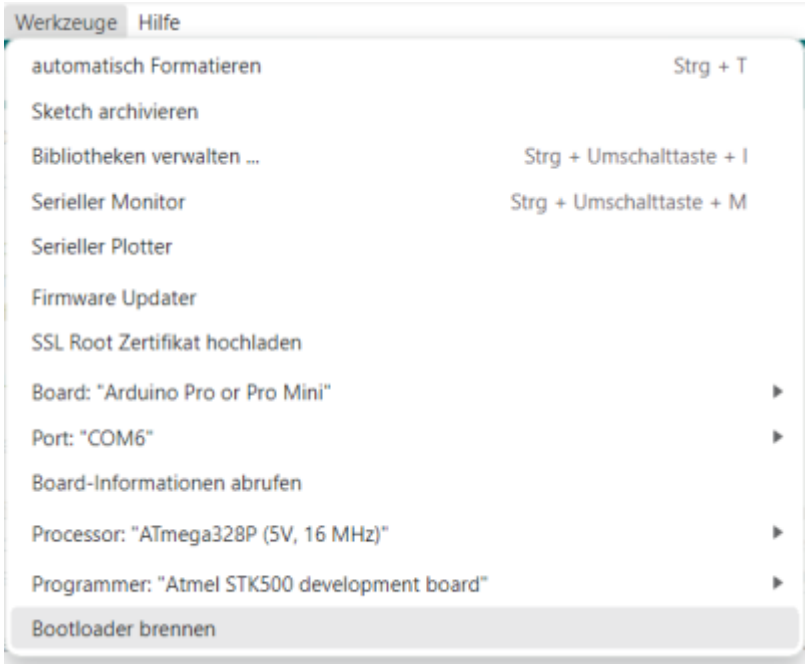
Anschließend kann der Controller mit einem USB-Seriell-Adapter am Steckverbinder FTDI mit der IDE wie ein Arduino Pro Mini programmiert werden.

STK500 Programmer

Von Pololu gibt es einen AVR-USB-Programmer, der kompatibel ist zum STK500. Im System meldet sich der Programmer mit zwei COM-Ports an. Ein Port dient als Programmer-Port und der andere als TTL-kompatibler serieller Port. Der Programmer kann für 5V- und 3,3V-Systeme konfiguriert werden:



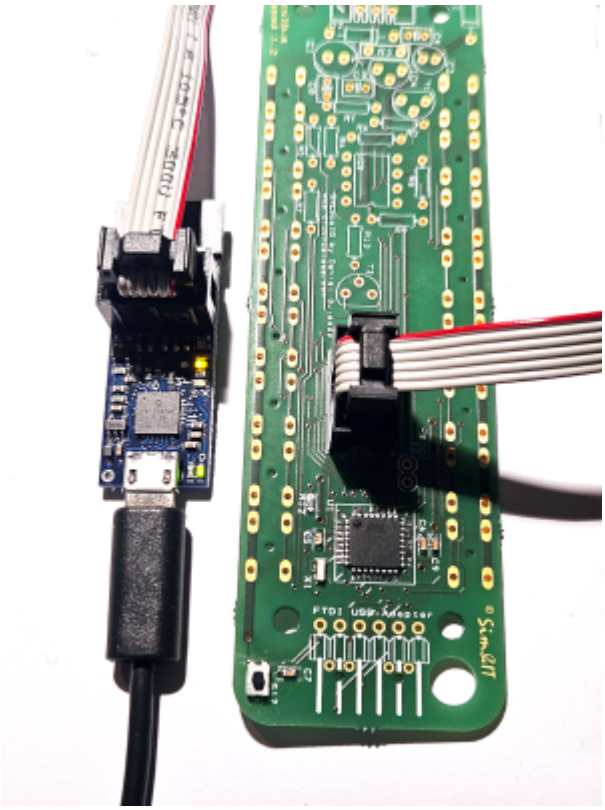
In der Arduino-IDE muss unter Werkzeuge das **Zielsystem**, der **Programmer-Port** und der **Programmer** eingetragen werden.



Anschließend wird der Bootloader programmiert.

Je nach Zielsystem kann jetzt das Zielsystem über einen USB-Adapter, eigenen USB-Anschluss oder auch über den ISP-Steckverbinder mit einem Sketch programmiert werden.

ATmega328



Auf dem Zielsystem muss der entsprechende 6-polige ICP-Steckverbinder vorhanden sein oder eine andere Möglichkeit diese Signale zu nutzen (z.B. Lötpunkte o.ä).

Zuordnung des ISP-Steckverbinders zum **ATmega328**

```
Pin1 → PB4 - MISO
Pin2 → 5P/3,3P je nach Zielsystem
Pin3 → PB5 - SCK
Pin4 → PB3 - MOSI
Pin5 → PC6 - RESET
Pin6 → GND
```

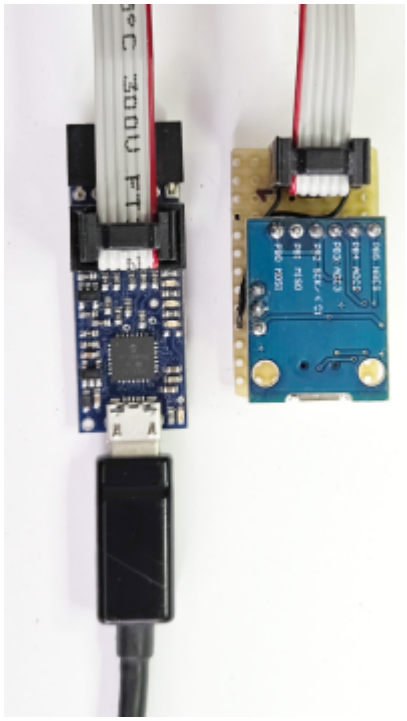
Anschließend kann der Bootloader programmiert werden. Das Programmieren kann nach der Bestückung aller Bauteile rund um den ATmega328 erfolgen.

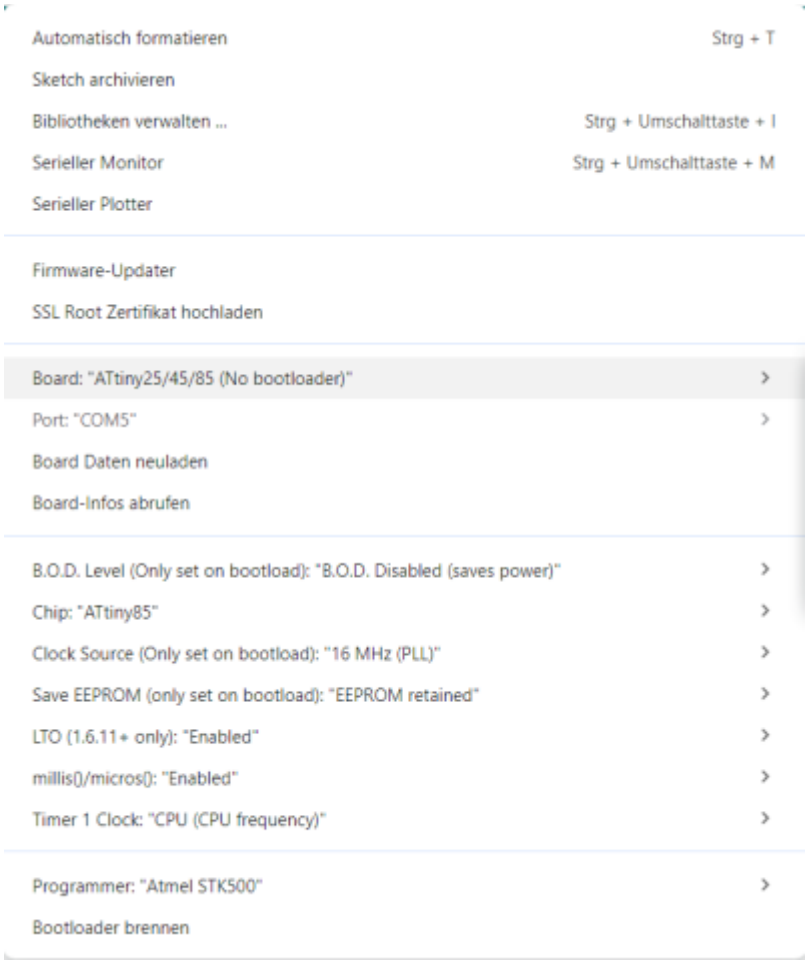
Wenn der Bootloader vorhanden ist, kann der Controller über den sechspoligen Anschluss (FTDI-kompatibel) und einem USB-Seriell-Adapter mit der Arduino-IDE programmiert werden.

ATtiny85

Digispark-Board

Für die ICP/ISP-Programmierung der ATtiny85-Boards habe ich einen kleinen Adapter gebaut. Damit ist es dann möglich, das Board auch ohne den Micronucleus-Bootloader zu verwenden, wenn man den USB-Anschluss nicht mehr benötigt. Das spart ca. 20% des vorhandenen Programmspeichers. Ich nutze 16MHz als CPU-Frequenz, u.a. weil bei einigen DCC-Dekodern die Dimmung des PB3 in der Software realisiert werden muss und sonst leichtes Flimmern bemerkbar war.





Zuordnung des ISP-Steckverbinders zum **ATtiny85**

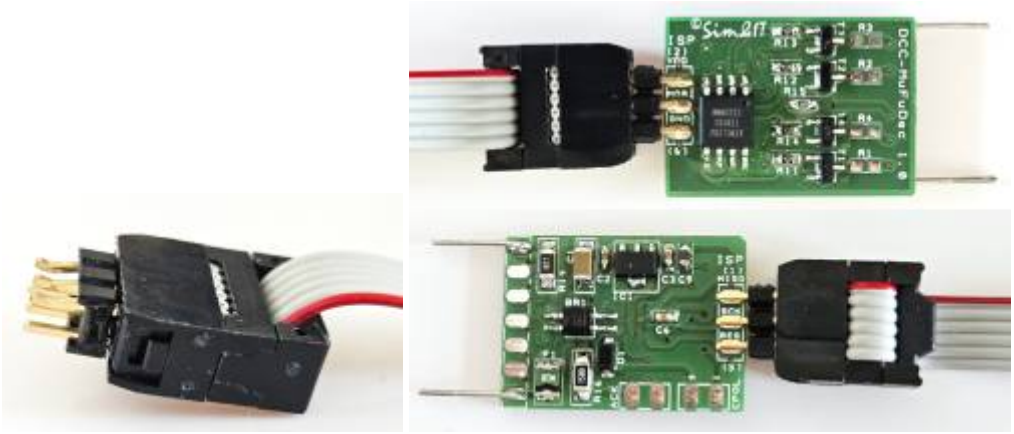
```
Pin1 → PB1 - MISO
Pin2 → 5P
Pin3 → PB2 - SCK
Pin4 → PB0 - MOSI
Pin5 → PB5 - RESET
Pin6 → GND
```

Als erstes muss auch hier der Menüpunkt **Bootloader brennen** ausgeführt werden, das bewirkt aber nur das Programmieren der Fuses. Ich nutze zuletzt die B.O.D-Level-Einstellung 4,3 V.

Anschließend wird in der Arduino-IDE der Menüpunkt **Sketch → Mit Programmer hochladen** für die Programmübertragung verwendet.

DCC-Dekoder-Hardware 4-Port MuFuDec

Eine komplette DCC-Dekoderplatine mit ATtiny85 hat an einer Seite 6 Pads für die ISP-Programmierung. An diese Pads könnte eine 6-polige Stiftleiste so angelötet werden, dass die Platine zwischen den Stiften liegt. Da die Programmierung eigentlich nur einmalig erfolgen muss, kann aber auch eine Stiftleiste so modifiziert/leicht gebogen werden, dass diese straff auf die Kontaktflächen aufgeschoben werden kann.



From:
https://www.simandit.de/simwiki/ - **Wiki**

Permanent link:
https://www.simandit.de/simwiki/doku.php?id=weiteres:arduino:bootloader

Last update: **2026/09/06 22:16**