

# Arduino-Bootloader brennen

Während der Blogartikel „Den MicroController überwachen“ entstand und dafür anfänglich der Arduino Nano genutzt wurde, kam schnell bei den Beispielcodes für den Watchdog Frustration auf. Da der verwendete Arduino Nanos älter als 5 Jahre war, kam es beim Entwickeln der Watchdog-Beispiele zu einem im Blogbeitrag beschriebenen Problem. Das Problem führt dazu, dass der Watchdog während dem Starten vom Arduino Nano in einer Startschleife hängen bleibt. Was also tun? Die Antwort ist das Brennen des Bootloaders auf den Arduino. Es ist möglich einen Arduino einen neuen Bootlader zu verpassen ohne einen ISP-Programmer besitzen zu müssen. Wie das geht, was Sie dazu brauchen und was es ggf. noch für interessante Informationen gibt, erfahren Sie in diesem Blogbeitrag.

## Benötigte Hardware

Damit Sie ihren Arduino mit einem neuen Bootloader brennen können benötigen Sie folgende Materialien, siehe Tabelle 1.

Pos	Anzahl	Bauteil	Link
1	1	Arduino Uno (empfohlen) im Folgenden als Programmer bezeichnet	<a href="https://az-delivery.com">https://az-delivery.com</a> <a href="https://www.amazon.de">https://www.amazon.de</a>
2	1	Das passende USB-Programmierkabel für Pos 1	
3	1	Jumper Wire „Female to female“	<a href="https://az-delivery.com">https://az-delivery.com</a>
4	1	OPTIONAL Breadboard Kit mit 3 x Jump Wire	<a href="https://www.az-delivery.de">https://www.az-delivery.de</a>

*Tabelle 1: Benötigte Materialien*

Behalten Sie bitte im Hinterkopf, dass Sie zwei Arduinos brauchen! Einmal den Arduino, der mit einem Bootloader geflasht werden soll, das sogenannte Target und der Arduino, der als Programmer dient. Diese Begriffe werden nachfolgend für beide Arduinos genutzt.

## Benötigte Software

Für diesen Blogteil brauchen Sie lediglich eine Arduino IDE, welche Sie von der [Arduino-Homepage](#) bekommen. Bitte achten Sie darauf, dass Sie die Arduino IDE herunterladen und nicht die Web Editor - Version starten. Letzteres kann zwar den Code für den Arduino der ISP-Programmer werden soll hochladen, aber nicht das Bootloader flashen.

## Vorwort bevor es zum Aufbau kommt

In diesem Blogbeitrag wird davon ausgegangen, dass ein Arduino Uno als Programmer und ein Arduino Nano als Target genutzt werden. Programmer

bedeutet in diesem Fall, dass der Arduino Uno die Aufgabe übernimmt, die Kommunikation zum Ziel, im englischen auch Target genannt, aufbaut und den Bootloader danach flasht. Natürlich können Sie auch jede andere Kombination nutzen, z.B. kann auch ein Arduino Nano als Programmer für einen Arduino Mega als Targe verwendet werden. In vielen englischen Foren wird aber empfohlen den Arduino Uno als Programmer einzusetzen.

## Den Bootloader flashen

Bevor Sie den Bootloader auf ein Target brennen können, braucht der Arduino erst einmal den passenden Programmcode. Dieser wird praktischerweise bei der Installation der Arduino IDE gleich mitgeliefert. Schließen Sie daher den Arduino Uno an Ihren PC an und öffnen Sie unter Datei -> Beispiele -> 11.ArduinoISP -> ArduinoISP, siehe Abbildung 1, den Sketch.

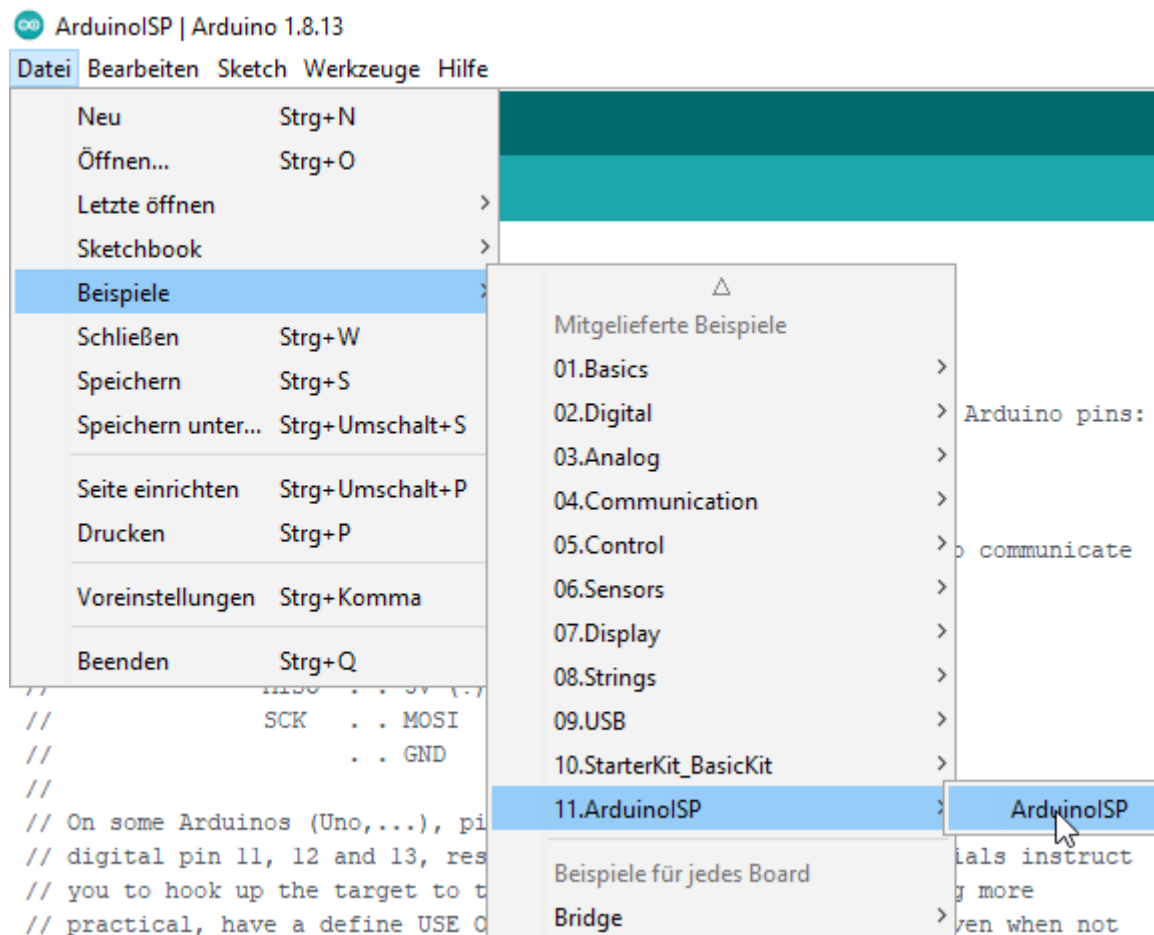


Abbildung 1: Öffnen vom Sketch ArduinoISP

Haben Sie die aktuellste Version der Arduino IDE, beim Erstellen des Blogbeitrags war dies die Version 1.8.13, wird der Arduino und der zugehörige COM-Port automatisch erkannt. Laden Sie daher den Sketch direkt auf den Programmer hoch, indem Sie den Sketch über Sketch -> Hochladen oder die Tastenkombination Strg +U oder der Funktionstaste, siehe Abbildung 2 rote Umrandung nutzen.

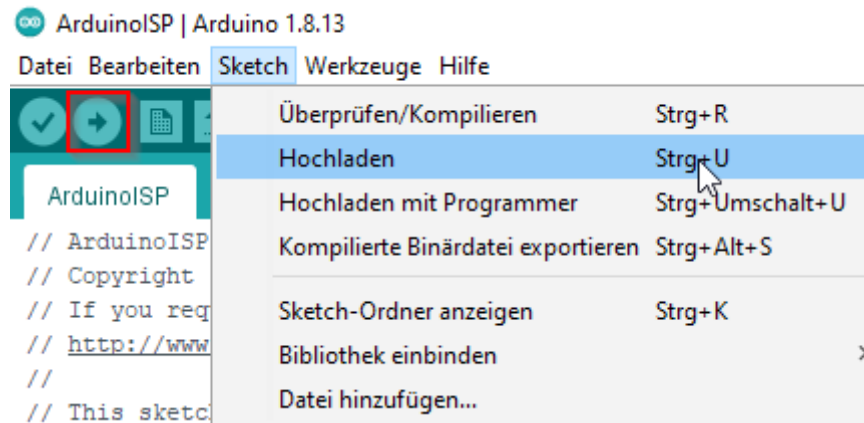


Abbildung 2: Sketch auf Arduino hochladen

Damit ist der Arduino Uno zu einem ISP-Programmer geworden. Den Quellcode können Sie sich gerne einmal ansehen, interessant ist der Ausschnitt, den Sie in Code 1 sehen.

```
// Configure which pins to use:
```

```
// The standard pin configuration.
#ifndef ARDUINO_HOODLOADER2
```

```
#define RESET    10 // Use pin 10 to reset the target rather than SS
#define LED_HB    9
#define LED_ERR   8
#define LED_PMODE 7
```

```
// Uncomment following line to use the old Uno style wiring
// (using pin 11, 12 and 13 instead of the SPI header) on Leonardo, Due...
```

```
// #define USE_OLD_STYLE_WIRING
```

```
#ifdef USE_OLD_STYLE_WIRING
```

```
#define PIN_MOSI  11
#define PIN_MISO  12
#define PIN_SCK    13
```

```
#endif
```

```
// HOODLOADER2 means running sketches on the ATmega16U2 serial converter
chips
```

```
// on Uno or Mega boards. We must use pins that are broken out:
#else
```

```
#define RESET     4
#define LED_HB     7
#define LED_ERR    6
#define LED_PMODE  5
```

```
#endif
```

```

// By default, use hardware SPI pins:
#ifndef PIN_MOSI
#define PIN_MOSI MOSI
#endif

#ifndef PIN_MISO
#define PIN_MISO MISO
#endif

#ifndef PIN_SCK
#define PIN_SCK SCK
#endif

// Force bitbanged SPI if not using the hardware SPI pins:
#if (PIN_MISO != MISO) || (PIN_MOSI != MOSI) || (PIN_SCK != SCK)
#undef USE_HARDWARE_SPI
#endif

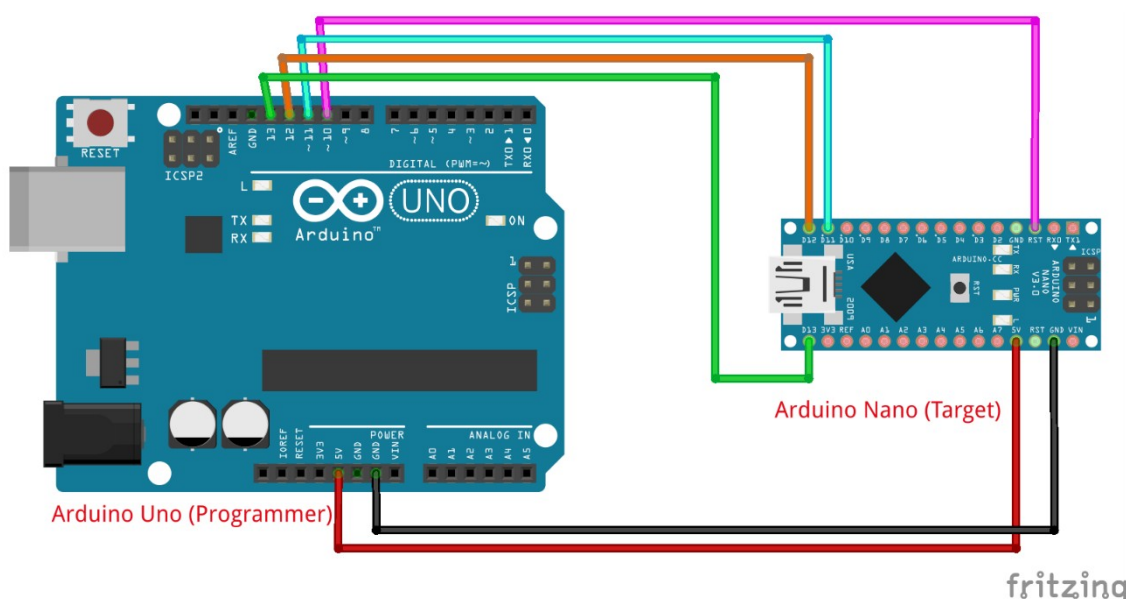
```

*Code 1: Pinbelegung um Code*

An dieser Stelle wird festgelegt, welche Pins für die Programmer genutzt werden, damit dieser als ISP-Programmer funktionieren kann. Damit das klappt braucht es immer die Pins

- 5V (Spannungsversorgung)
- GND (Spannungsversorgung)
- Reset (Reset vom Target)
- MOSI (Kommunikation)
- MISO (Kommunikation)
- SCK (Serielle Uhr)

Nachdem das Hochladen auf den Programmer abgeschlossen ist, trennen Sie die USB-Verbindung zum PC und schließen Sie das Target, wie in Abbildung 3 gezeigt an.



*Abbildung 3: Programmer und Target verbinden*

Möglich ist auch der Anschluss über die Pins mit der Beschriftung ICSP, dazu benötigen Sie jedoch das genaue Pinout, welches Sie für den Arduino Uno unter dem Link [https://content.arduino.cc/assets/Pinout-UNOrev3\\_latest.pdf](https://content.arduino.cc/assets/Pinout-UNOrev3_latest.pdf) und für den hier verwendeten Arduino Nano unter dem Link [https://content.arduino.cc/assets/Pinout-NANO\\_latest.pdf](https://content.arduino.cc/assets/Pinout-NANO_latest.pdf) finden. Der sicherere Weg ist aber tatsächlich der Anschluss über die Funktions- bzw. Digitalpins der Boards. In manchen Fällen ist ein 10µF-Kondensator zwischen dem RESET- und GND-Pin notwendig, siehe Abbildung 4.

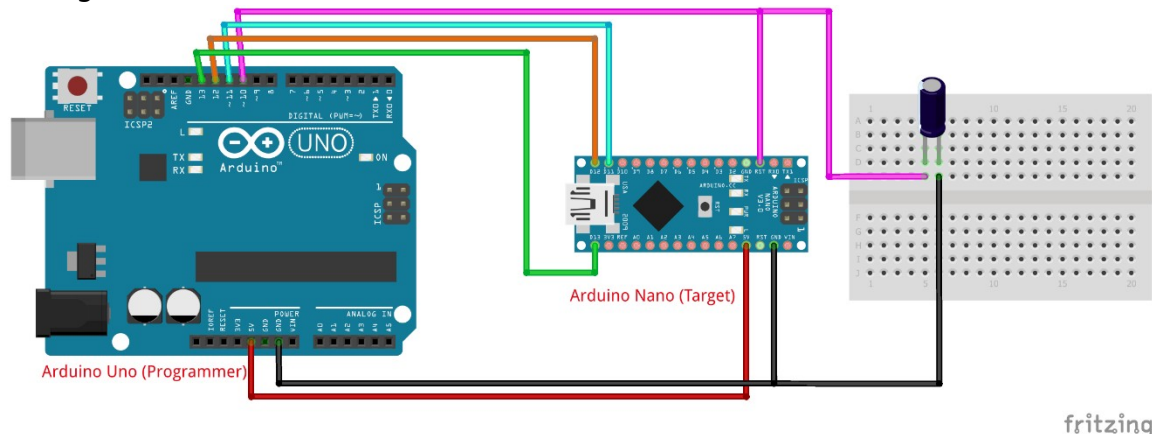


Abbildung 4: Programmer und Target verbinden mit Kondensator

Nun muss in der Arduino IDE noch bekannt gegeben werden, dass der Arduino UNO ein ISP-Programmer ist. Wählen Sie dazu in der Menüleiste Werkzeuge -> Programmer -> Arduino as ISP aus, siehe Abbildung 5.

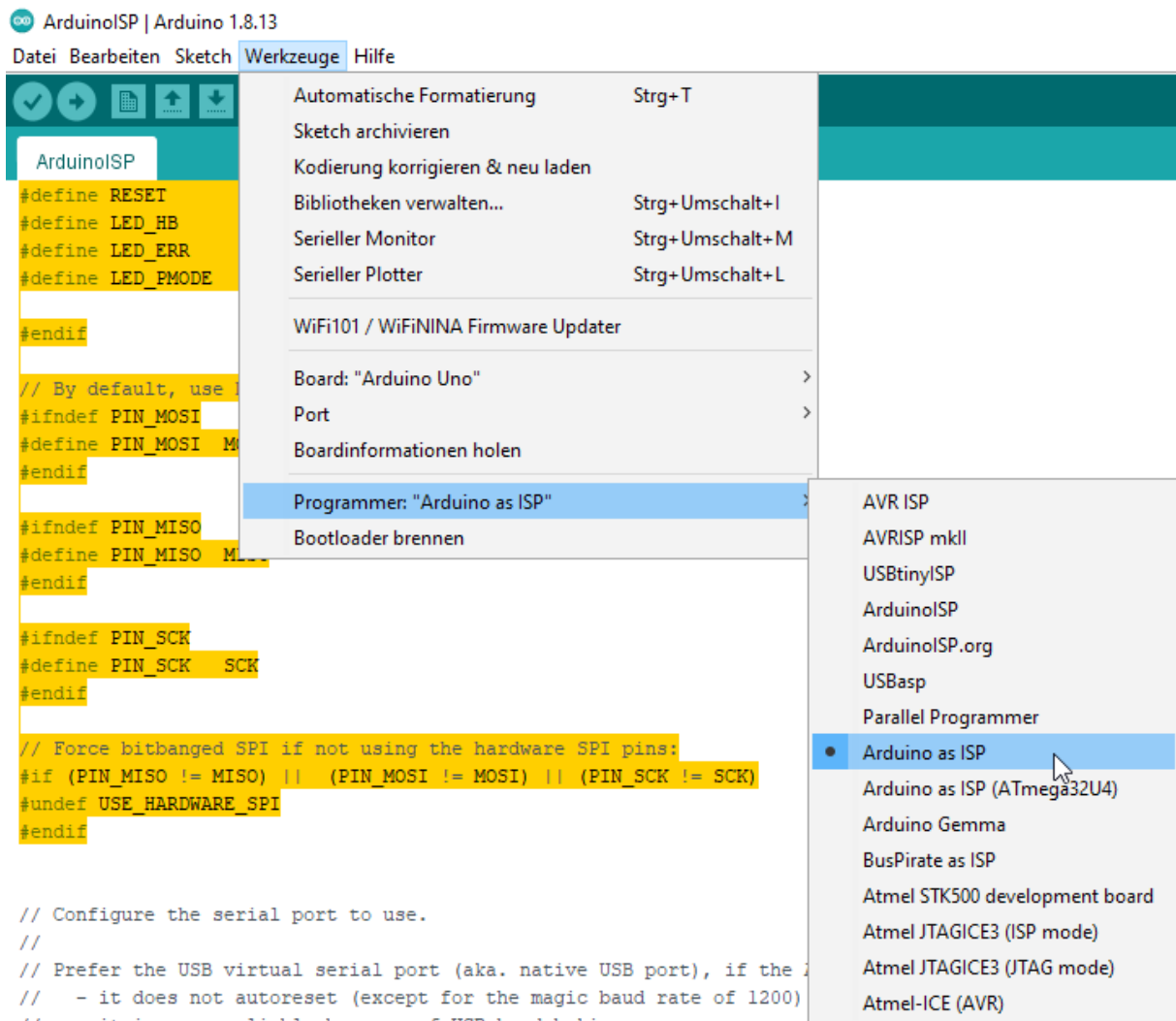


Abbildung 5: Arduino Uno in der Arduino IDE als ISP-Programmer bekannt machen

Im letzten Schritt wählen Sie den Boardtyp vom Target aus. Hierzu navigieren Sie in der Menüleiste zu Werkzeuge -> Board -> Arduino AVR Boards -> Arduino Nano, siehe Abbildung 6.

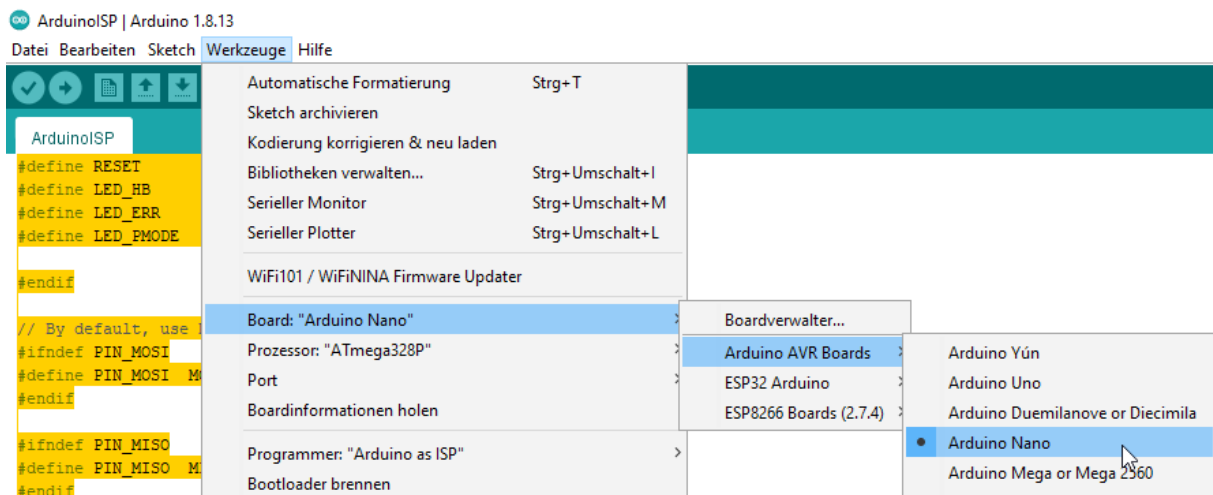


Abbildung 6: Wähle Boardtyp vom Target

Gerade bei den Arduino Nano und z.B. dem Arduino Mega kann es sein, dass Sie noch den korrekten Prozessor auswählen müssen, siehe Abbildung 7.

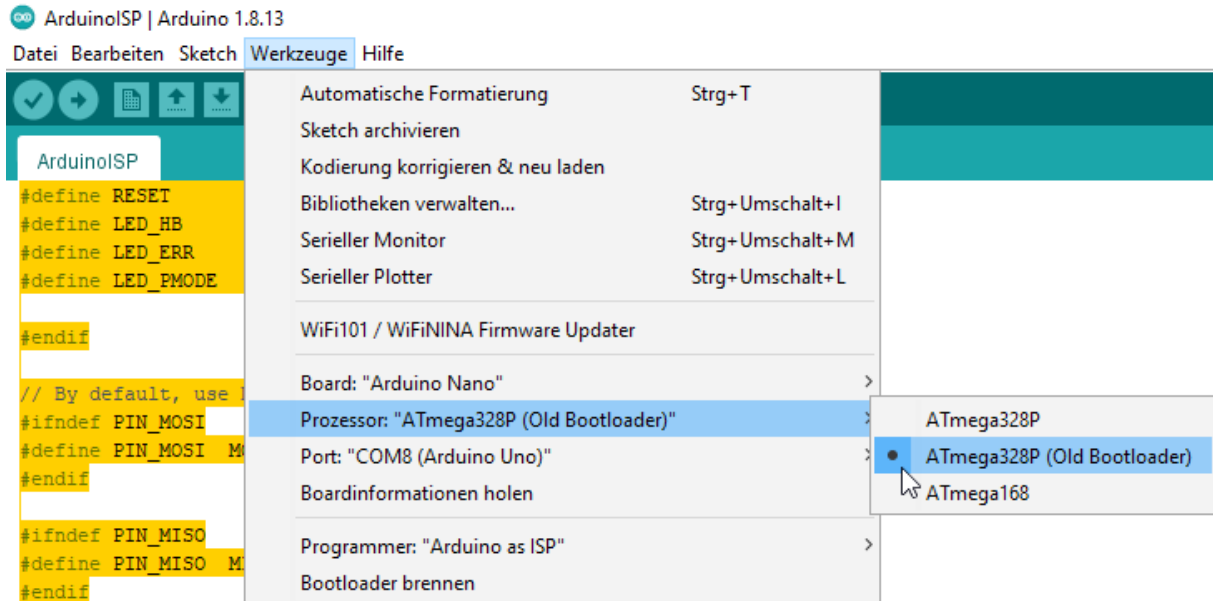


Abbildung 7: Passenden Prozessor für Target auswählen

Nun schließen Sie wieder den Programmer mit dem USB-Kabel an dem PC an. Sie werden feststellen, dass beide Arduinos nun Spannung haben, was daran liegt, dass der Programmer das Target, über die Pins 5V und GND, entsprechend mitversorgt. Wählen Sie nun in der Werkzeugleiste Werkzeuge -> Bootloader brennen aus, siehe Abbildung 8.

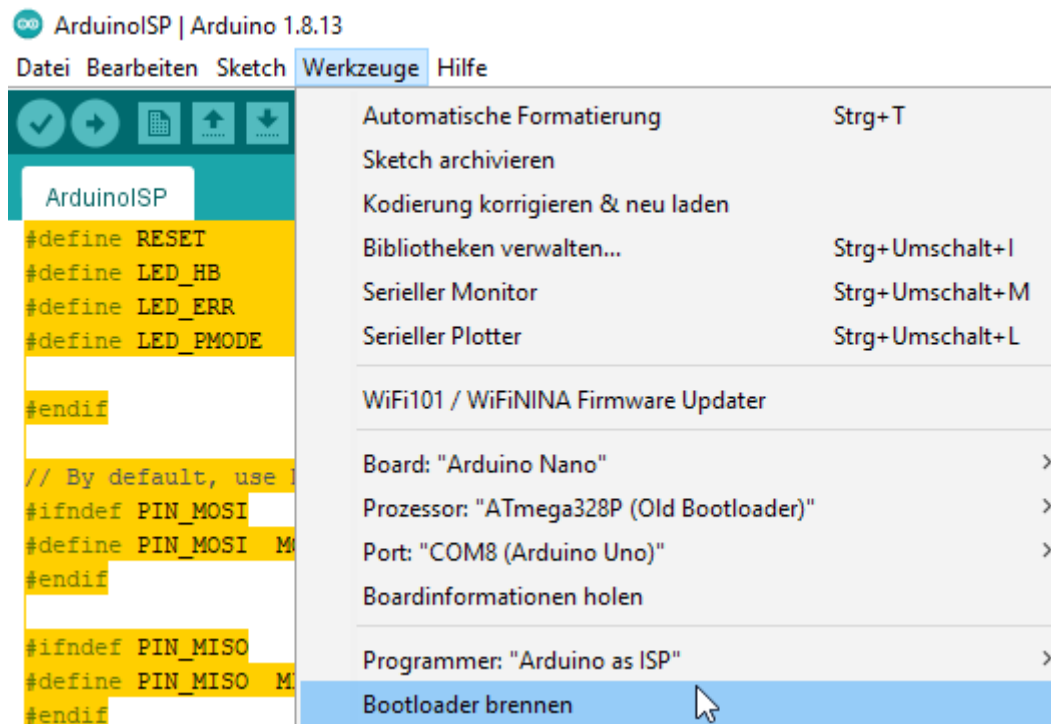


Abbildung 8: Bootloader brennen

In der Arduino IDE werden Sie nun, je nach Voreinstellung, diverse Ausgabe sehen. Gleichzeitig sehen Sie, dass beim Programmer während dem Brennvorgang die LEDs RX und TX flackern und auch die eingebaute LED vom Target flackert. Auf jeden Fall sollten die letzten Meldungen wie in Abbildung 9 gezeigt aussehen.

```
Der Bootloader wurde gebrannt.

Writing | ##### | 100% 0.02s

avrdude: 1 bytes of lock written
avrdude: verifying lock memory against 0x0F:
avrdude: load data lock data from input file 0x0F:
avrdude: input file 0x0F contains 1 bytes
avrdude: reading on-chip lock data:

Reading | ##### | 100% 0.01s

avrdude: verifying ...
avrdude: 1 bytes of lock verified

avrdude done. Thank you.
```

118 - 68 Arduino Nano auf COM8

Abbildung 9: Meldung in der Arduino IDE für Erfolg

Danach ist das Target, der Arduino Nano, fertig und sollte wieder laufen. Testen können Sie das, indem Sie z.B. das Beispiel-Sketch „Blink“ auf den Arduino Nano laden. Sollte, trotz Erfolgsmeldung beim Brennen des Bootloaders, der Arduino Nano nicht programmierbar sein, prüfen Sie noch einmal die Einstellungen vom Target und brennen den Bootloader erneut.

## Weitere Informationen

### Arduino mit Optiboot-Bootloader

Wer im Internet nach Informationen über den Bootloader für die Arduino-Boards sucht, der wird schnell über das Projekt optiboot stolpern. Hierbei handelt es sich für die gängigsten Arduino-Boards, mit ATmega-Chipsatz, um einen verbesserte Bootloadervariante. Dieser ist ebenfalls, wie beim Arduino Uno, nur 0,5kByte groß. Die Versprechen sind aber weitaus höher, als bei Arduino. Bei optiboot ist die Rede davon, dass mehr Speicher auf allen kompatiblen Arduino-Board zu verfügbar sein soll, dass Sketche schneller hochgeladen werden können und gerade höhere Baudraten stabiler laufen sollen. Sucht man ein bisschen weiter, stellt man fest, dass Teile von optiboot anscheinend in die offiziellen Arduino-Bootloader eingeflossen sind.

Um einen optiboot-Bootloader auf den Arduino zu brennen, brauchen Sie zunächst die passende URL für eine zusätzliche Boardkonfiguration, die Sie in den Arduino IDE Voreinstellungen eintragen müssen. Die URL finden Sie unter der Adresse <https://github.com/Optiboot/optiboot/releases>, siehe Abbildung 10.







# Optiboot Verison 8.0

 WestfW released this on 20 Sep 2018 · [64 commits](#) to master since this release

Now with `do_spm()` feature (allows applications to write to flash memory by calling code i  
This should mean that this repository is now up-to-date with respect to features and platf  
/mighty, and the SpenceKonde Attiny cores.

## Assets 4

-  [Optiboot-8.0.zip](#)
-  [package\\_optiboot\\_optiboot-additional\\_index.json](#)
-  [Source code \(zip\)](#)
-  [Source code \(tar.gz\)](#)

- Link in neuem Tab öffnen
- Link in neuem Fenster öffnen
- Link in neuem privaten Fenster öffnen
- Lesezeichen für diesen Link hinzufügen
- Ziel speichern unter...
- Link in Pocket speichern
- Link-Adresse kopieren**
- Google-Suche nach "package\_optiboo..."

## Interim 7.0 release...

Abbildung 10: Optiboot-Adresse kopieren

Die Adresse fügen Sie in der Arduino IDE unter Datei -> Vorsteinstellungen wie in Abbildung 11 hinzu.

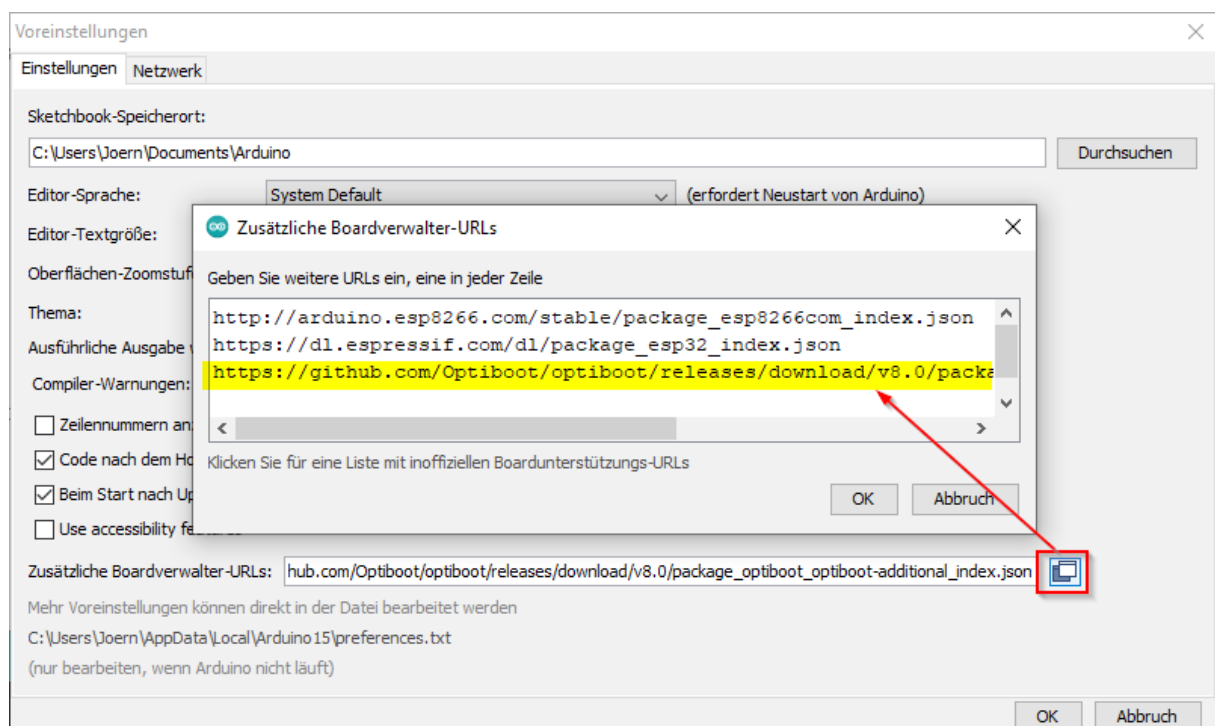


Abbildung 11: Optiboot-URL der Arduino IDE hinzufügen

Bestätigen Sie alles mit OK und installieren Sie dann über Werkzeuge -> Board -> „Boardverwaltung...“ die Boardbibliothek Optiboot, siehe Abbildung 12.

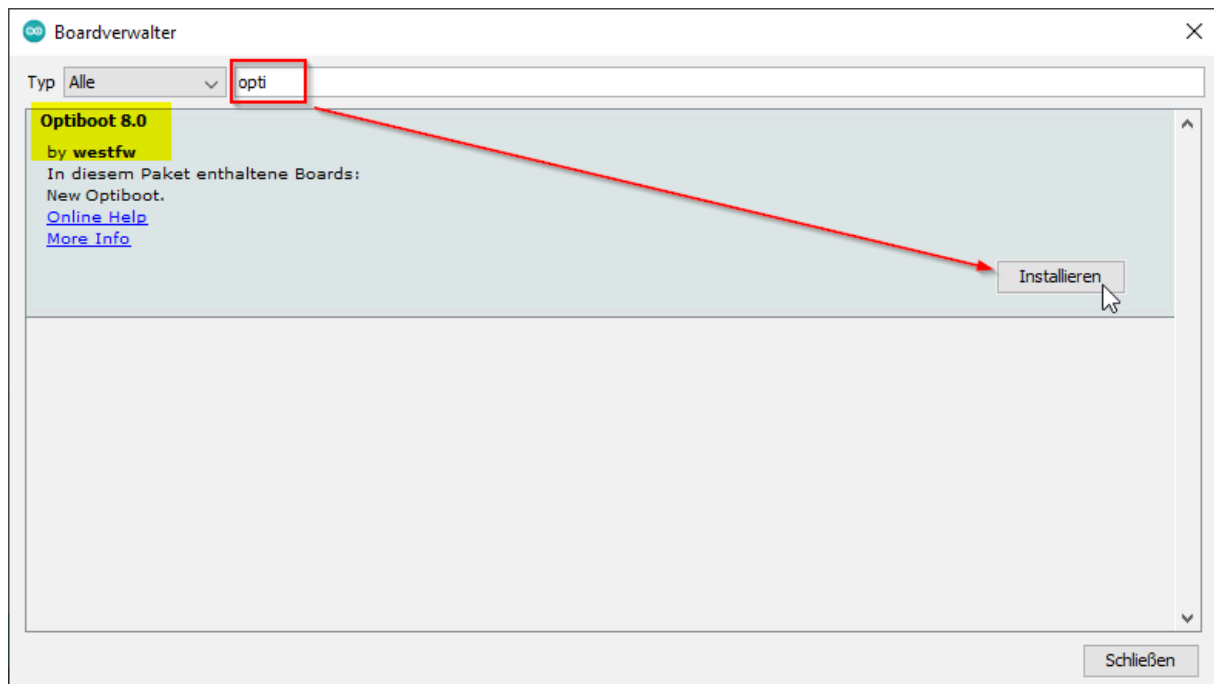


Abbildung 12: Installiere Optiboot-Boardbibliothek

Danach können Sie z.B. für den Arduino Nano, das Board „Optiboot on 28-pin cpus“ auswählen, siehe Abbildung 13 und wie schon im Hauptteil dieses Artikels erläutert, den Optiboot-Bootloader über den Arduino Uno auf den Arduino Nano brennen.

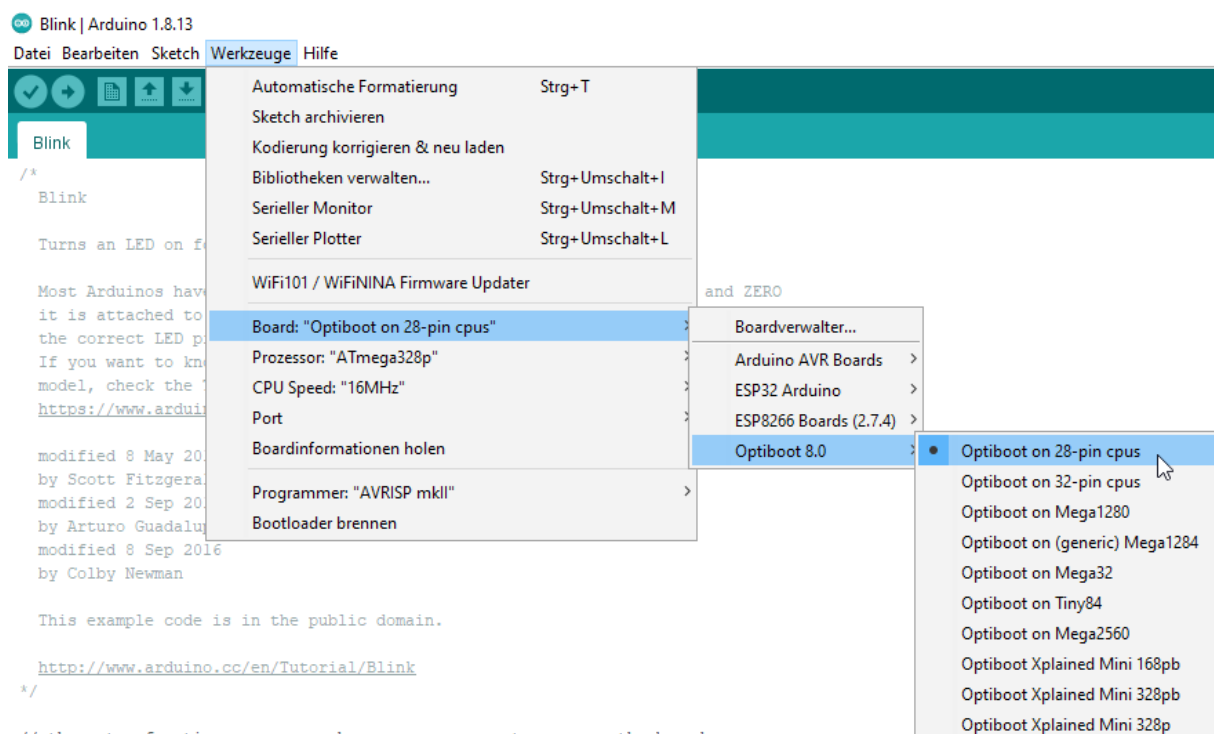


Abbildung 13: Passendes Optiboot-Board für Arduino Nano suchen

Sollten Sie zu einem späteren Zeitpunkt wieder einen anderen Bootloader auf Ihr Target brennen wollen, ist das jederzeit möglich.

## Weitere Bootloader für Arduino

Optiboot ist nicht der einzige Dritthersteller von Bootloader für die Arduino-Produktfamilie. Es gibt noch einige mehr, die unter <https://playground.arduino.cc/Main/ArduinoCoreHardware/#Bootloader> einsehbar sind. Da an dieser Stelle nicht alle getestet werden können, können Sie sich einen Eindruck der Möglichkeiten machen. Der Brennvorgang dürfte in den meisten Fällen identisch zu der hier vorgestellten Variante sein.

## Das Pinout der beliebtesten Arduino-Boards

Wie bereits erwähnt, soll hier noch nicht Schluss sein, sondern noch weitere Informationen folgen. Zunächst einmal, damit Sie nicht jedes Datenblatt vom Arduino durcharbeiten müssen, soll Ihnen Tabelle 2 helfen, die richtigen Pins auf den Boards zu finden. Dabei geht die Tabelle auf die beliebtesten Boards ein.

<b>Funktion</b>	<b>Arduino Uno</b>	<b>Arduino Nano</b>	<b>Arduino Micro</b>	<b>Arduino Mega 2560</b>	<b>Arduino Due</b>
<b>MOSI</b>	Digitalpin 11	Digitalpin 11	Digitalpin 16	Digitalpin 11	Digitalpin 11
<b>MISO</b>	Digitalpin 12	Digitalpin 12	Digitalpin 14	Digitalpin 12	Digitalpin 12
<b>SCK</b>	Digitalpin 13	Digitalpin 13	Digitalpin 15	Digitalpin 13	Digitalpin 13

*Tabelle 2: Arduino Pin-Funktionsübersicht*

Unter <https://store.arduino.cc/arduino-genuino/boards-modules> können Sie die gesamte Arduino-Boards einsehen. Hier können Sie zu jedem Board die technischen Spezifikationen und die Dokumentation einsehen. Interessant für Sie ist bei der Dokumentation, dass Sie direkt unter dem Pinout-Bild sich eine PDF mit dem Pinout der letzten bekannten Revision vom Board runterladen können, siehe Abbildung 14 rote Umrandung.

## OSH: Schematics

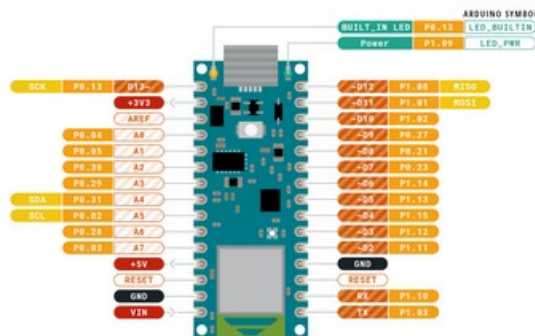
The Arduino Nano 33 BLE is open-source hardware! You can build your own board using the following files:



## Pinout Diagram



### ARDUINO NANO 33 BLE



32kByte Speicher besitzt, könne Sie 1,5kByte mehr Speicher für Ihre Programme herausholen. Gleichzeitig hat der Bootloader vom Arduino Nano auch einige ungefixte Bugs, die es aber in den Bootloader des Arduino Unos geschafft haben. Gerade der Bug mit dem Watchdog bei niedrigen Watchdog-Zeiten sei hier zu nennen. Die Kehrseite beim Brennen vom Arduino Uno Bootloaders auf den Nano ist, dass Sie bei der Auswahl vom Board in der Arduino IDE nicht mehr den Arduino Nano auswählen dürfen, sondern den Arduino Uno! Sie sollten daher den Arduino Nano, mit geflashten Arduino Uno Bootloader, entsprechend markieren.

## Der richtige Programmierer für die Entwicklung

Sie haben beim Flashen vom Bootloader den Programmierer umgeschaltet. Der eingestellte „Arduino as ISP“ können Sie dafür nicht verwenden. Damit die wieder Ihren Code auf ihr Arduinoboard laden können, müssen Sie wieder den Programmierer umstellen auf „AVRISP mkII“, siehe Abbildung 15.

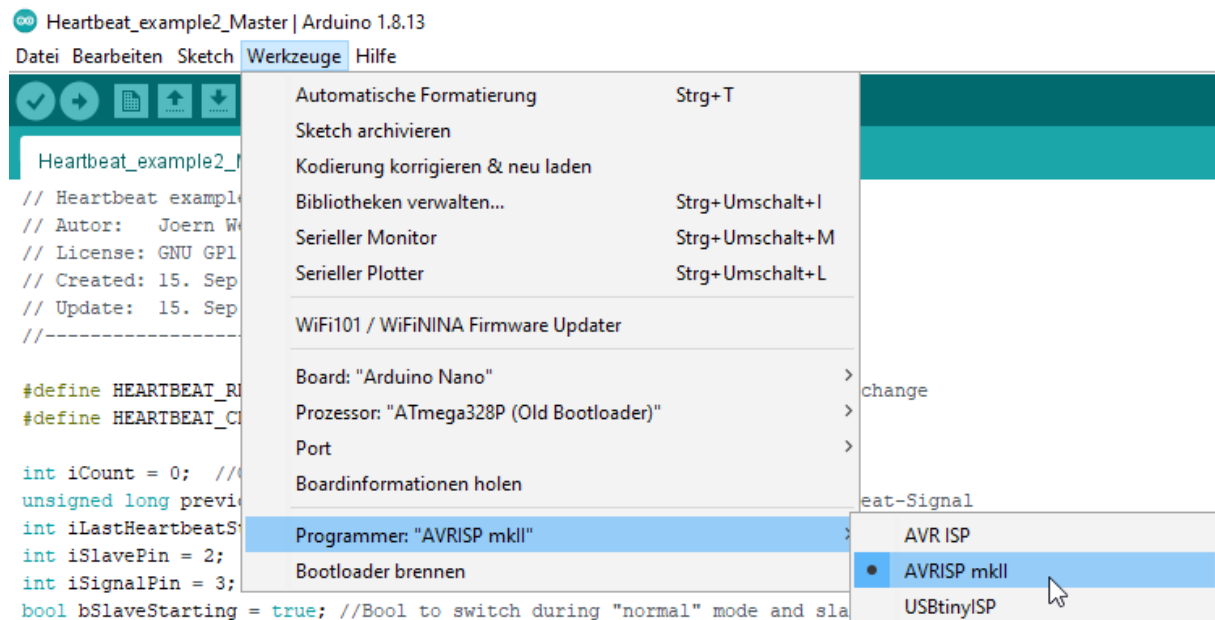


Abbildung 15: Programmierer für Entwicklung auswählen

## Der richtige Programmierer fürs Brennen vom Bootloader

Wenn Sie sich einmal die Programmierer genauer ansehen, werden Sie die Einträge „ArduinoISP“ und „ArduinoISP.org“ gesehen haben, siehe Abbildung 16.

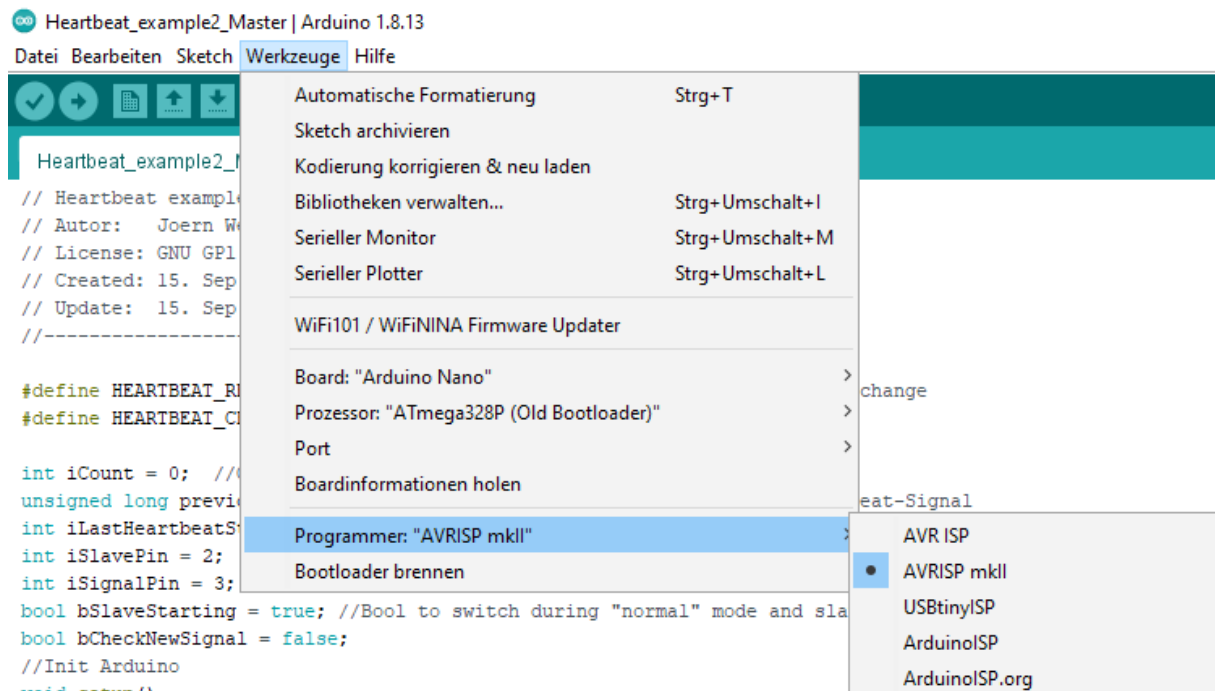


Abbildung 16: Nicht nutzbare ISP-Programmer

Was es mit den Programmern genau auf sich hat, soll hier nicht näher beschrieben werden. Jedoch soll hier erwähnt sein, sollten Sie nicht „Arduino as ISP“ auswählen, wird das Brennen vom Bootloader nicht gelingen.

Weitere Projekte für AZ-Delivery von mir finden Sie unter <https://github.com/M3taKn1ght/Blog-Repo>.