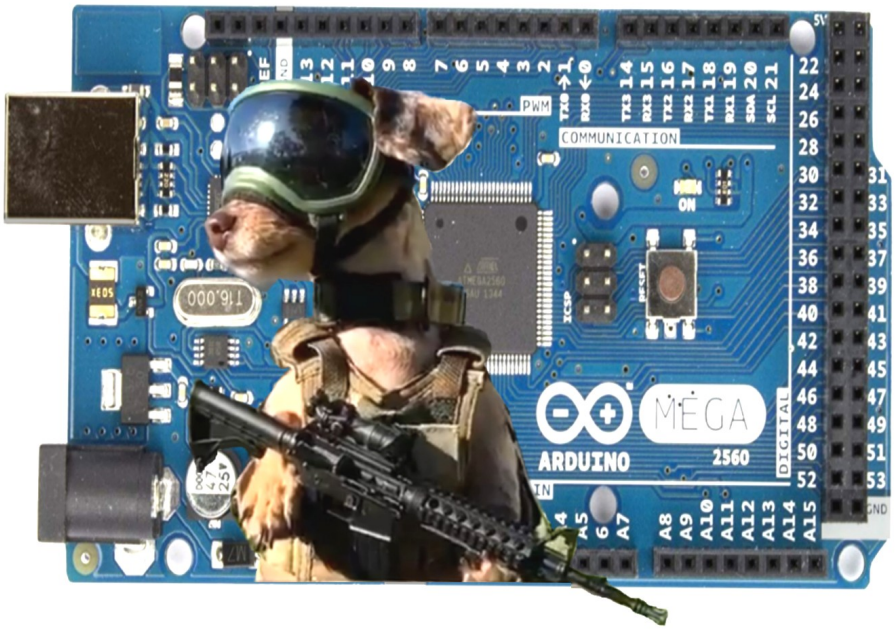
Arduino interener Watchdog Timer

Inhaltsverzeichnis



Watchdog-Timer und Soft-Reset

Achtung: Nur mit neuem Bootloader einsetzt bar !!! (siehe Bootloader flashen).

Der Watchdog Timer

Alle ATmega-Controller verfügen über einen Watchdog-Timer ("Watchdog"), der die Stabilität des Systems bei unvorhersehbaren Problemen, z. B. dem Ausfall eines Sensors oder die Unterbrechung einer Kommunikationsverbindung, sicherstellen soll. Der Watchdog-Timer ist bei den ATmega-Controllern ein **zusätzlicher** Zähler auf dem Chip mit **eigenem** 128-kHz-Taktoszillator. Einmal aktiviert, wird der Zähler unabhängig vom laufenden Programm regelmäßig erhöht und, falls er nicht rechtzeitig von der Software zurückgesetzt wird, beim Überlaufen des Zählers einen Reset auslösen.

Der Watchdog-Timer besitzt einen einstellbaren Vorteiler, um die Zeit bis zum Überlauf in Stufen festlegen zu können. Die Reaktionszeit muss vom Programmierer sorgfältig gewählt werden - ein Reset darf nur ausgelöst werden, wenn es wirklich nötig ist. Man rechnet also aus, wie lange die Hauptschleife `loop()` im ungünstigsten Fall für einen Durchgang benötigt und setzt den Watchdog-Timer eine Zeitstufe über diesen Wert.

Wird der Watchdog falsch konfiguriert (Ablaufzeit zu kurz) kann dies zur Folge haben, dass der Watchdog vor dem Bootloader auslöst und damit ein Herunterladen eines geänderten Programms verhindert.

Achtung:

1. Nur mit neuem Bootloader einsetzt bar! (siehe Bootloader flashen).
2. Wenn Sie Experimente mit dem Watchdog durchführen, solle die Reaktionszeit des Watchdog nicht zu niedrig gewählt werden. Im schlimmsten Fall muss dann nämlich der Arduino über die ISP-Schnittstelle programmiert werden oder man muss den Controllerchip mit Arduino als ISP wieder zum Leben erwecken (siehe Bootloader flashen).

Anwendung des Arduino Watchdogs

Die Anwendung des Watchdogs ist recht einfach, die AVR-Watchdog-Library besitzt nur wenige Funktionen. Zuerst wird die Library auf die C-typische Art und Weise eingebunden:

```
#include <avr/wdt.h>
```

Für die Programmierung und Aktivierung werden nur drei Funktionen benötigt:

- `wdt_enable(zeit)` schaltet den Watchdog scharf und setzt gleichzeitig die Zeit bis zum Auslösen. Die möglichen Zeitangaben listet die Tabelle unten auf.
- `wdt_disable()` schaltet den Watchdog wieder ab.
- `wdt_reset()` setzt den internen Zähler zurück. Diese Funktion muss regelmäßig aufgerufen werden, um zu erhindern, dass der Zähler überläuft und der Watchdog einen Reset auslöst. Der Aufruf erfolgt beispielsweise am Ende oder Anfang der Hauptschleife (`loop()`).

Die Watchdog-Zeitkonstanten können dem ATmega328-Datenblatt entnommen werden, möglich sind folgende Werte:

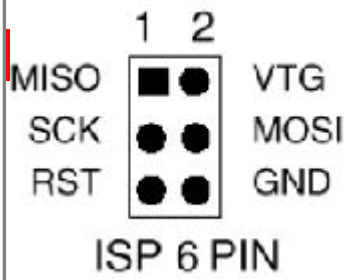
Zeit	**Konstante**	**Prescaler-Bits**			
		WDP 0	**WDP 1**	**WDP 2**	**WDP 3**
16 ms	WDTO_15MS	0	0	0	0
32 ms	WDTO_30MS	1	0	0	0
64 ms	WDTO_60MS	0	1	0	0
0,125 s	WDTO_120MS	1	1	0	0
0,25 s	WDTO_250MS	0	0	1	0
0,5 s	WDTO_500MS	1	0	1	0
1,0 s	WDTO_1S	0	1	1	0
2,0 s	WDTO_2S	1	1	1	0
4,0 s	WDTO_4S	0	0	0	1
8,0 s	WDTO_8S	1	0	0	1

ATMEGA328 Bootloader flashen

Anschlüsse

Manchmal kommt es vor, dass man sich - wie auch immer - den Bootloader des Arduino zerschießt. Das geht ganz leicht, wenn man den Arduino wie ein ganz gewöhnliches Controllerboard behandelt und die Software nicht über die serielle Schnittstelle in den Flash-Speicher des Prozessors lädt, sondern direkt über den ISP-Anschluss (ISP: (In System Programming). Die meisten Arduino-Boards haben dafür auch einen eigenen Anschluss, eine zweireihige Stiftleiste mit sechs Pins. Der Arduino UNO besitzt sogar zwei ISP-Anschlüsse, einen für den eigentlichen Arduino und einen für den USB-zu-seriell-Konverter.

Der ISP-Port ist bei allen Arduinos gleich belegt. Die folgende Tabelle zeigt die Zuordnung der Anschlüsse. In der mittleren Spalte stehen die Pins des ISP-Ports, links die Leitungen des Programmers (Arduino oder AVR-Programmer) und rechts die Anschlüsse am zu flashenden Controller.

	Arduino as ISP	AVR Programmer	ISP Header	ATmega328	ATmega32U4
	D12 (MISO)	MISO	Pin 1	D12 (Pin 18)	D14
	Vcc/5V	5V	Pin 2	Vcc (Pin 7)	Vcc
	D13 (SCK)	SCK	Pin 3	D13 (Pin 19)	D15
	D11 (MOSI)	MOSI	Pin 4	D11 (Pin 17)	D16
	D10 (Reset)	Reset	Pin 5	Reset (Pin 1)	Reset
	GND	GND	Pin 6	GND (Pin 8)	GND

Anmerkung: Falls nicht wie unten beschrieben ein Arduino UNO als Programmierer verwendet wird, sondern der ISP Header eines anderen Systems als Programmierer dient, ist zu beachten, dass der RST-Anschluss immer vom Output D10 (Reset) genommen werden muss. RST am ISP Header ist die Reset-Leitung des als Programmier dienenden Controllers, während D10 die Restleitung des zu programmierenden Controllers ist.

Bootloader auf einen UNO oder NANO brennen

Um einen neue Bootloader auf einen Arduino zu brennen, wird beispielsweise der Arduino UNO als ISP verwendet. Zum Programmieren des Bootloaders sind folgende Schritte notwendig:

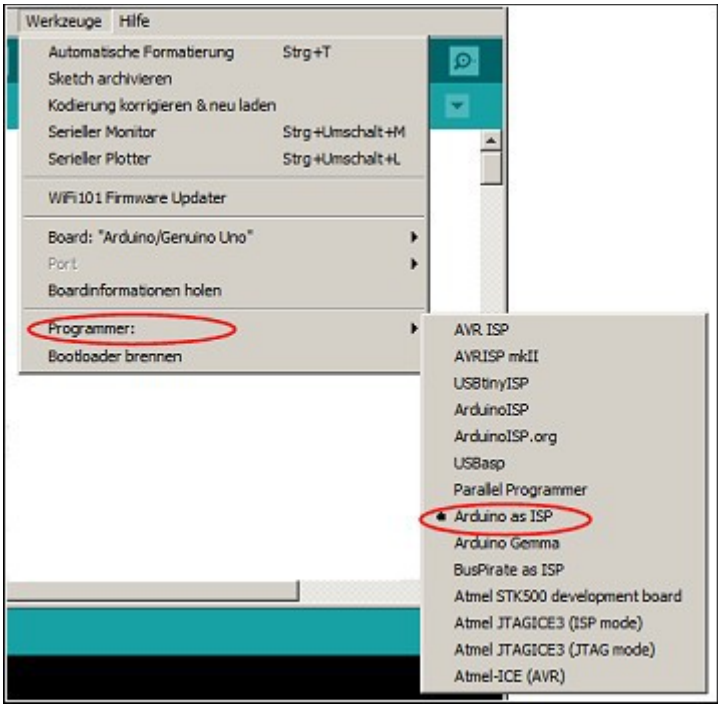
Schritt 1: Den UNO zum ISP-Programmer umfunktionieren
Dazu wird zunächst der UNO mit dem PC verbunden und mit der Programmer-Software geladen. In der Arduino-IDE finden Sie das entsprechende Programm unter *Datei → Beispiele → ArduinoISP*. Dieses wird in die IDE geladen. Falls nicht schon geschehen, unter *Werkzeuge → Board → Arduino UNO* den UNO als aktuelles System festlegen und die richtige COM-Schnittstelle wählen (*Werkzeuge → Port → COMxx*. Dann das Programm compilieren und in den UNO laden.

Schritt 2: Anschluß des zu flashenden Arduino
Das kann jeder Arduino mit einem ISP-Anschluss sein. Als Beispiel soll hier ein Arduino NANO dienen. Stecken Sie die UNO vom PC ab (stromlos machen) und verbinden Sie den ISP-Port des NANO mit den entsprechenden Pins des UNO gemäß der Tabelle oben. Also:

```
D12 des UNO → Pin 1 des NANO
+5V des UNO → Pin 2 des NANO
D13 des UNO → Pin 3 des NANO
D11 des UNO → Pin 4 des NANO
D10 des UNO → Pin 5 des NANO
GND des UNO → Pin 6 des NANO
```

Für die Verbindung der beiden Board eignen sich die Experimentierlitzen mit Stecker an einem und Buchse am anderen Ende.

Schritt 3: Bootloader programmieren
Stecken Sie den UNO wieder am PC an. Das "ArduinoISP"-Programm ist ja noch gespeichert und startet. Nun wählen Sie den zu flashenden NANO unter *Werkzeuge → Board → Arduino NANO* aus und überprüfen Sie, ob die Einstellung des seriellen Ports noch stimmt. Stellen Sie unter *Werkzeuge → Programmer → Ardunino as ISP* ein. Nun starten Sie das Brennen des Bootloaders mittels *Werkzeuge → Board → Bootloader brennen*. Machen Sie den UNO wieder stromlos und entfernen Sie die Verbindungen zum NANO.

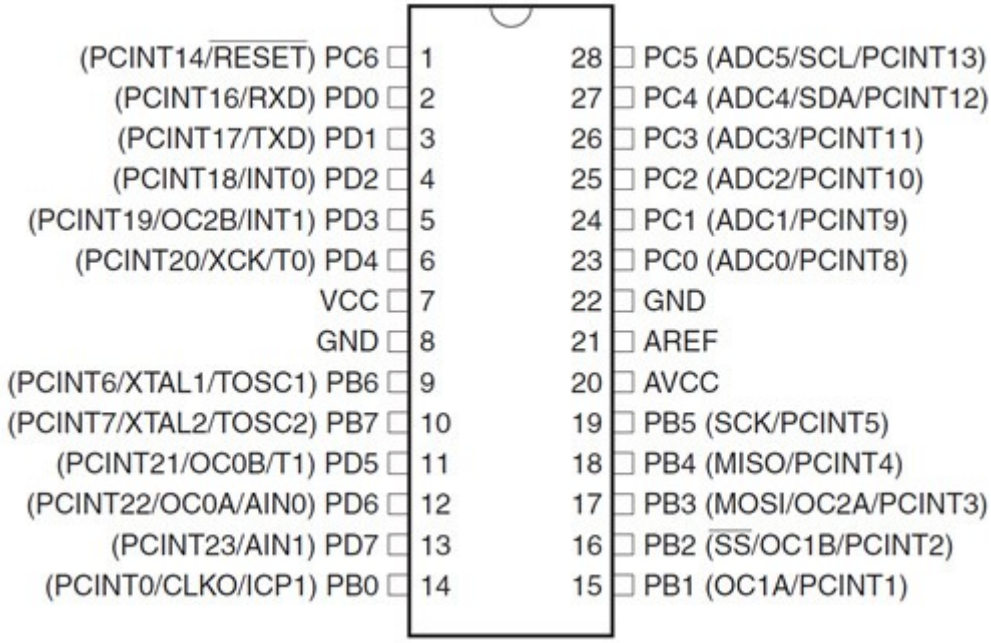


Bootloader auf einen ATMEGA328P brennen

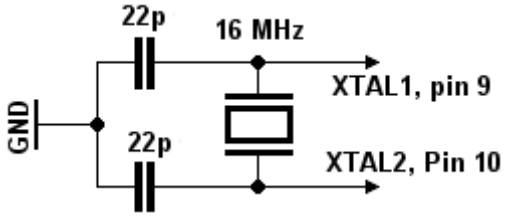
Ab und zu verwende ich den Arduino zum Programmieren, aber andere Hardware für das fertige Projekt. Oft lohnt sich die Anfertigung einer Platine nicht, weil sich nur einige wenige Bauteile rund um den ATMEGA328P tummeln. Also wird die Schaltung auf einer Lochrasterplatte aufgebaut und der Controller dann aus der DIL-Fassung des Arduino gezogen und in die DIL-Fassung auf der Lochrasterplatte gesteckt. Also wird öfter mal ein ATMEGA328P (genauer ATMEGA328P-PU) mit Bootloader als Ersatz benötigt (manchmal auch, wenn der ATMEGA328P in der Schaltung "gegrillt" wurde). Hier kann man fast genauso verfahren, wie oben - lediglich der zweite Schritt ist unterschiedlich.

Schritt 1: Den UNO zum ISP-Programmer umfunktionieren
Dazu wird zunächst der UNO mit dem PC verbunden und mit der Programmer-Software geladen. In der Arduino-IDE finden Sie das entsprechende Programm unter *Datei > Beispiele > ArduinoISP*. Dieses wird in die IDE geladen. Falls nicht schon geschehen, unter *Werkzeuge > Board > Arduino UNO* den UNO als aktuelles System festlegen und die richtige COM-Schnittstelle wählen (*Werkzeuge > Port > COMxx*. Dann das Programm compilieren und in den UNO laden.

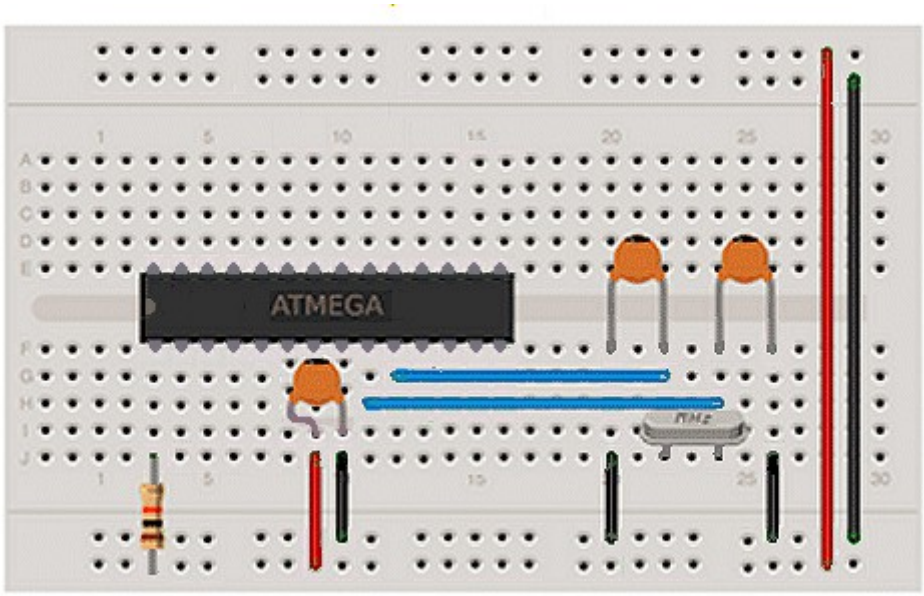
Schritt 2: ATMEGA328P auf dem Steckbrett vorbereiten
Diesmal wird das Steckbrett mit dem ATMEGA328P bestückt. Die Pinbelegung des Dual-Inline-Gehäuses zeigt die folgende Grafik.



Neben den ISP-Leitungen werden noch der Quarz mit seinen beiden Kondensatoren und ein Pullup-Widerstad für den Reset-Eingang benötigt. Die Schaltung des Quarzes ist recht einfach, der Quarz wird zwischen den Pins 9 und 10 angeschlossen, die Kondensatoren sind gegen Masse geschaltet.



Man kann den Controller noch einen 100-nF-Kondensator an der Spannungsversorgung (Pin 7 und Pin 8) gönnen. Des weiteren wird ein 10-kΩ-Widerstand zwischen Reset (Pin 1) und +5V gelegt. Ihr Steckbrett sollte dan etwa so aussehen:



Nun werden noch die sechs ISP Leitungen vom UNO zum Steckbrett gezogen:

```
UNO GND → ATMEGA328 Pin 8
UNO +5V → ATMEGA328 Pin 7
UNO  10 → ATMEGA328 Pin 1  Reset
UNO  11 → ATMEGA328 Pin 17 MOSI
UNO  12 → ATMEGA328 Pin 18 MISO
UNO  13 → ATMEGA328 Pin 19 SCK
```

Wer es ganz komfortabel haben möchte, platziert noch eine LED mit Vorwiderstand (ca. 1 kΩ) auf dem Steckbrett, wobei der Widerstand an Pin 19 des ATMEGA angeschlossen wird. Die andere Seite des Widerstandes kommt an die Anode der LED (langes Bein) und die Kathode der LED (kurzes Bein) wird mit GND verbunden. Dann hat man eine Funktionsanzeige.

Schritt 3: Bootloader programmieren
Stecken Sie den UNO wieder am PC an. Das "ArduinoISP"-Programm ist ja noch gespeichert und startet. Nun wählen Sie diesmal den zu flashenden ATMEGA unter *Werkzeuge > Board > Arduino UNO* aus und überprüfen Sie, ob die Einstellung des seriellen Ports noch stimmt. Stellen Sie unter *Werkzeuge → Programmer → Ardunino as ISP* ein. Nun starten Sie das Brennen des Bootloaders mittels *Werkzeuge > Board > Bootloader brennen*. Machen Sie den UNO wieder stromlos und entfernen Sie die Verbindungen zum ATMEGA.

Testbeispiel Programm

```
#include <avr/wdt.h>

void setup() {
//Die Dauer, bis der Watchdog auslösen soll
//wdt_enable(WDTO_15MS);
//wdt_enable(WDTO_30MS);
//wdt_enable(WDTO_60MS);
//wdt_enable(WDTO_120MS);
//wdt_enable(WDTO_250MS);
//wdt_enable(WDTO_500MS);
//wdt_enable(WDTO_1S);
//wdt_enable(WDTO_2S);
//wdt_enable(WDTO_4S);
//wdt_enable(WDTO_8S);

pinMode(3, OUTPUT); // Test LED Reset
pinMode(4, OUTPUT); // Test LED loop
pinMode(5, INPUT); // Reset Watchdog Eingang

//--------- Reset LED ein - aus -ein - aus
digitalWrite(3, HIGH);
delay(500);
digitalWrite(3, LOW);
delay(500);
digitalWrite(3, HIGH);
delay(500);
digitalWrite(3, LOW);

// ---- Watchdog Befehle ---
wdt_disable(); // WD ausschalten
wdt_enable(WDTO_8S); // WD einschalten
wdt_reset(); // WD zurücksetzen
}

void loop() {
digitalWrite(4, HIGH);
delay(500); // 0,5 Sekunden
digitalWrite(4, LOW);
delay(500); // 0,5 Sekunden

// ----- Reset Watchdog solange Eingang 5 = Low
if (digitalRead(5) == LOW) {wdt_reset();}
}
```