Interrupts

Bei den Arduino Mikrocontrollern sind Interrupts im Grunde ein Signal, das es ermöglicht, eine Funktion in einem Sketch **jederzeit** direkt aufzurufen, gleichgültig, womit der Arduino sonst gerade „beschäftigt“ ist.

Wird ein Interrupt ausgelöst, wird der Ablauf des Programms angehalten, die Interrupt-Funktion ausgeführt [1)], wenn die ISR abgearbeitet ist, wird das das Programm an der Stelle der Unterbrechung fortgeführt.

Wie macht mans?

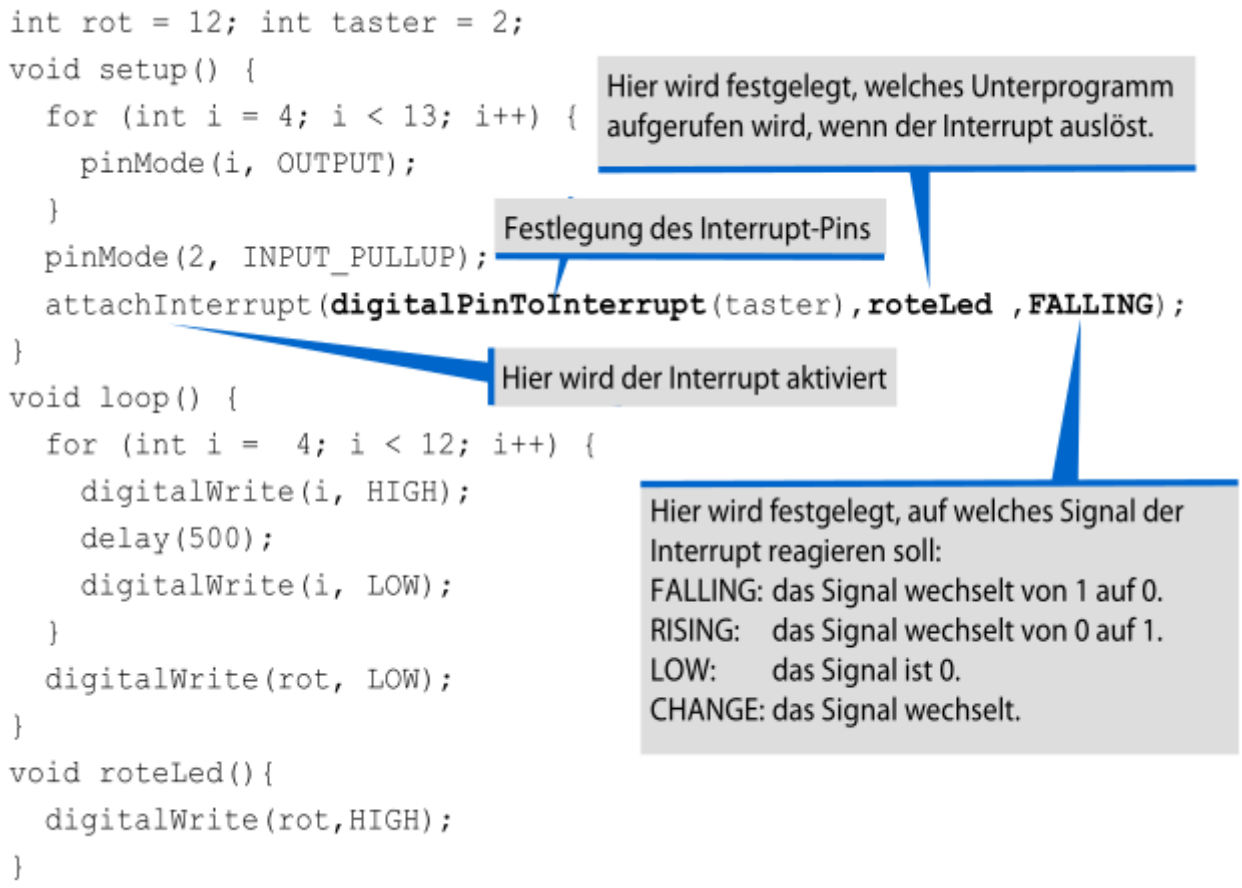
Der Ardunio Uno hat zwei Interrupts, die Pins 2 und 3 können verwendet werden. Um eine Interrupt-Service-Routine mit einem Interrupt zu verknüpfen, verwendet man die folgende Syntax:

```
attachInterrupt(PIN, ISR, mode)
```

dabei ist:

- PIN: Der Pin, der die ISR Triggern soll
- ISR: Der Name der Funktion, die aufgerufen werden soll
- mode: Definiert, wann der Interrupt getriggert werden soll. Dafür sind bereuts 4 Konstanten definiert:
  - LOW Interrupt wird getriggert, wenn der Pin LOW ist
  - CHANGE Interrupt wird getriggert, wenn der Pin den Wert ändert
  - RISING Interrupt wird getriggert, wenn der Pin von LOW auf HIGH wechselt
  - FALLING Interrupt wird getriggert, wenn der Pin von HIGH auf LOW wechselt

Beispiel



Für ISR gelten einige besondere Regeln:

- ISR sollten möglichst kurz gehalten werden.
- ISR können keine Argumente bekommen oder Rückgabewerte zurückgeben. Stattdessen werden - ausnahmsweise! - globale Variablen benutzt, um Daten zwischen Interrupt Service Routinen und dem Hauptprogramm zu tauschen. Damit die Variablen dabei korrekt geändert werden, sollten sie als `volatile` deklariert werden.
- Einige Funktionen verhalten sich in ISRs anders als gewohnt oder funktionieren gar nicht:
  - `millis()` verlässt sich zum Zählen auf Interrupts, wird also in einer Interrupt Service Routine niemals hochzählen.
  - `delay()` benutzt ebenfalls Interrupts und wird deshalb gar nicht in einer Interrupt Service Routine funktionieren - sollte dort aber aucch n iemals benutzt werden, weil, die ISR ja schnell abgearbeitet werden sollte!
  - `micros()` wird anfangs gut funktionieren, sich aber nach etwa 1 bis 2 ms unvorhersehbar verhalten - möglichst nicht benutzen.
  - `delayMicroseconds()` benutzt keine Zähler und wird deshalb normal funktionieren.

1)

„ISR“ - Interrupt Service Routine