Universelle Bestimmung der Zählimpulsrate von Sensoren für Radioaktivität mit dem Arduino

Bernd Laquai 8.9.2014

Ob ein Pin-Dioden Sensormodul oder ein Geigermüller-Zählrohr als Sensor für Radioaktivität genutzt wird, am Ende ist in der Regel immer das Ziel sehr kurze digitale Zählpulse zu erzeugen, wenn man eine Zählrate als Maß für die Radioaktivität quantitativ erfassen möchte. Oft kann man sich bereits fertiger Mess-Module für radioaktive Strahlung bedienen, die lediglich eine Spannungsversorgung benötigen und einen Ausgang besitzen, an dem bereits digitale Zählimpulse abgeliefert werden, die man digital weiterverarbeiten kann. Man muss sich dann nicht mehr um eine entsprechende Hochspannungserzeugung oder Impulsverstärkung kümmern. Beispiele sind für die Zählrohr-basierten Sensoren der Pollin-Geigerzähler bzw. das SBM-20 Driver Modul von 4N-Galaxy und für die Pin-Dioden basierten Systeme das Stuttgarter Geigerle oder das Teviso-Modul.

Prinzipiell kann man also den Strahlungs-Sensor als Black-Box modellieren, die pro registriertem Strahlungsquantum einen sehr kurzen Zählimpuls liefet, der eine Dauer im Mikrosekundenbereich hat, rechteckförmig ist und zwischen 0 und 5V oder 0 und 3.3V wechselt und entweder low-aktiv oder high-aktiv ist.

Um aus diesen Zählimpulsen das berühmte Knacken eines Geigerzählers zu erzeugen muss man die Pulse in den Millisekundenbereich verlängern und in einen Strom umwandeln der einen Lautsprecher oder einen Signalgeber treiben kann. Wenn man sich aber nicht nur am Knacke erfreuen will, sonder auch eine quantitative Aussage über die Radioaktivität treffen will, muss man eine mittlere Zählrate bestimmen, mit der die Zählimpulse auftreten.

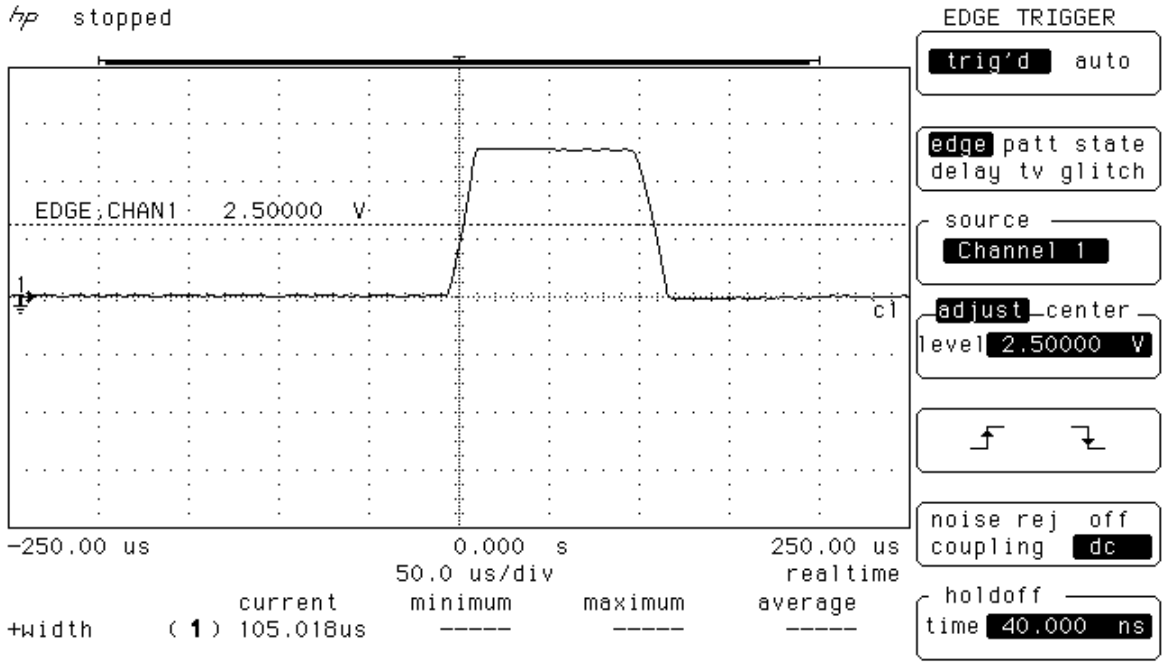

Abb. 1: High aktiver Puls mit 5V des Teviso Pin Dioden Sensor Moduls

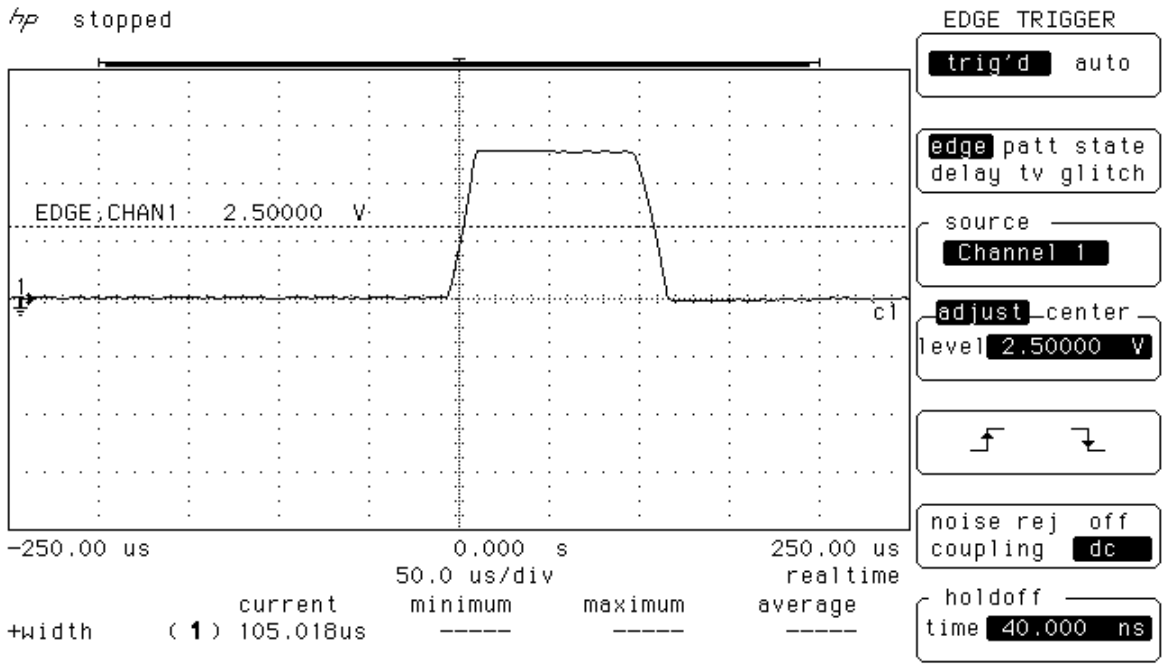

Abb. 2: Low aktiver Puls mit 3V des Zählrohr Moduls von 4N-Galaxy

Dazu bietet sich ein Mikrocontroller und dabei vor allem der Arduino wegen seiner einfachen Bedienbarkeit an. Bereits das einfachste Arduino-Modell, der Uno ist dafür bestens geeignet. In der einfachsten Form kann er die Zählrate über den Serial Monitor auf den Bildschirm eine angeschlossenen PC ausgeben und gleichzeitig die Impulsverlängerung bewerkstelligen und ein Ausgangssignal liefern, das einen digitalen Signalgeber ansteuern kann, so dass man auch gleich das Knacken als Funktionskontrolle bekommt.

Für die Zählraten-Bestimmung hat man prinzipiell zwei Möglichkeiten. Entwender man gibt eine Zeit vor und zählt wie viele Impulse man in dieser Zeit bekommt oder aber man gibt die Zahl der Impulse vor, die man abwarten will und man misst die Zeit die man benötigt bis man die vorgegebene Impulsanzahl erreicht. In beiden Fällen ergibt sich die Impulsrate in dem man die Impulszahl durch die Zeit dividiert. In vielen Fällen wird die erste Variante gewählt, weil sie naheliegender scheint. Die zweite Variante hat aber einen großen Vorteil gegenüber der ersten: Die Statistik (die Streuung) des Ergebnisses bleibt unabhängig von der Aktivität. Das liegt daran, dass durch die Zufälligkeit der Zählimpulse ja immer eine Unsicherheit in der berechneten Zählrate bleibt, die sich darin äußert, dass sie bei gleichbleibender Aktivität streut, da man sich auf eine endliche Anzahl an Zählpulsen zur Berechnung beschränken muss. Diese Streuung hängt von der Zahl an Zählimpulsen ab und nicht von der Zeit. Wenn man daher immer die gleich Zahl an Zählimpulsen abwartet, dann bleibt auch die Streuung des Ergebnisses gleich. Dabei passt sich die Messzeit automatisch an die Verhältnisse an, wenn man eine hohe Aktivität hat, erhält man aktualisierte Zählraten in rascher Folge, nur wenn die Zählrate niedrig ist, dauert die Messung länger. Damit wartet man bei einer Änderung der Zählrate immer die kürzestmögliche Zeit (bei gegebener Statistik) für den nächsten Messwert. Daher wird in diesem Beispiel die zweite Methode gezeigt.

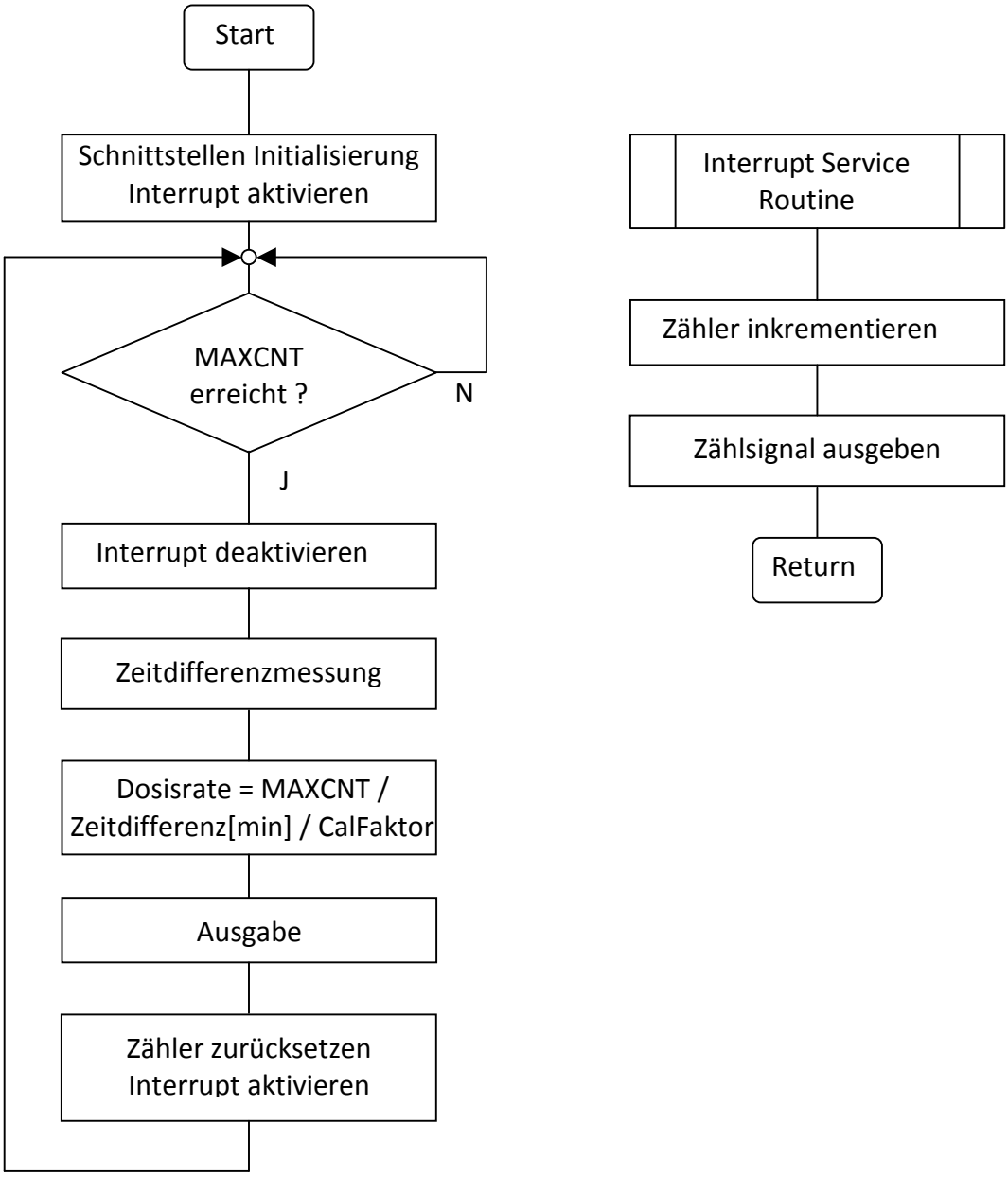

Abb.3: Flussdiagramm des universellen Arduino Zählprogramms

Bei einem Mikrocontroller wie dem Arduino bietet es sich an zur Erfassung der völlig unvorhersehbar eintreffenden Zählimpulse die Interrupt-Logik zu benutzen um die Zählimpulse zu erfassen. Damit kann sichergestellt werden, dass in einem Zählintervall Zählimpulse keine Zählimpulse verloren gehen, was mit einer Abfrage eines digitalen Eingangs in einer Schleife nicht unbedingt der Fall wäre. Die Interrupt-Logik sorgt dafür, dass die normale Programmausführung unterbrochen wird und eine spezielle Interrupt Service Routine (ISR) abgearbeitet wird, die in dieser Anwendung einen Zähler hochzählt. Durch diese Ereignissteuerung und Priorisierung wird also jeder Impuls erfasst auch wenn er nur ein paar Mikrosekunden lange anliegt. Die Interrupt Service Routine sollte allerdings so kurz wie möglich gehalten werden, da sonst das Betriebssystem eventuell Probleme bekommt,

wenn es in dieser Zeit gewisse wichtige Dinge nicht tun kann. Daher sollte man Berechnungen und Bildschirmausgaben dann in der normalen Programmschleife machen und eigentlich nur das Impulse Zählen in der Interrupt Service Routine bewerkstelligen.

Das unten aufgeführte Programm (Sketch) erledigt nun den besagten Job der Zählraten-Bestimmung auf dem Arduino. Zunächst wird die Zahl der Impulse MAXCNT vereinbart, auf die gewartet werden soll. Diese Zahl bestimmt die Statistik. Bei einem Pin-Dioden Zähler ist 10 ein sinnvoller Wert für MAXCNT. Da ein Zählrohr aber unter Umständen deutlich mehr Zählpulse liefert kann man MAXCNT für ein Zählrohr durchaus auch auf den Wert 100 oder höher setzen.

```
#define MAXCNT 10
#define CalFactor 1

volatile int counter = 0;
unsigned long oldTime = 0;
int speaker = 5;

void setup() {
  pinMode(speaker, OUTPUT);
  digitalWrite(speaker, LOW);
  Serial.begin(9600);
  attachInterrupt(0, count, FALLING);
}

void loop() {
  unsigned long time;
  unsigned long dt;
  float rate;

  if (counter == MAXCNT) {
    detachInterrupt(0);
    time = millis();
    dt = time-oldTime;
    rate = (float)MAXCNT*60.0*1000.0/(float)dt/CalFactor;
    Serial.println(rate);
    oldTime = time;
    counter = 0;
    attachInterrupt(0, count, FALLING);
  }
}

void count()
{
  counter++;
  digitalWrite(speaker, HIGH);
  delayMicroseconds(50000);
  digitalWrite(speaker, LOW);
}
```

Der Wert, der für CalFactor definiert wird dient einer eventuellen Umrechnung der Zählrate in eine Dosisleistung in µSv/h und wird in cpm / (µSv/h) angegeben. Das Zählprogramm bestimmt die Zählrate in counts per minute (cpm). Wenn der Hersteller eines Sensors einen Kalibrierfaktor in cpm / (µSv/h) angibt, so wie beispielsweise der Hersteller Teviso, dann kann dieser Wert hier eingetragen werden (z.B. 3.4 für das Teviso Modul RD2007), damit stellt der ausgegebene Wert die Dosisleistung in µSv/h dar. Man kann natürlich auch

versuchen selbst einen solchen Kalibrierfaktor zu finden, indem man die Zählrate für einen bekannten Prüfstrahler vermisst. Eine genaue Kalibrierung im Sinne von internationalen Standards ist aber denkbar schwierig und kann kaum mit den üblichen Hobbymitteln erreicht werden.