Ein Zählimpulsgenerator mit Poisson-Statistik zur Prüfung der Auswerteelektronik von Geigerzählern und Szintillationszählern

Bernd Laquai, 23.12.23

Einleitung

Gelegentlich hat man auf einer Urlaubs- oder einer Geschäftsreise ein Hotel gebucht, wo man nachts aufwacht und die Fahrzeuge, welche auf der Straße direkt vor dem Fenster vorbeifahren, recht laut hört. Dies kann einen doch erheblich daran hindern wieder einzuschlafen. Manchmal hilft es dann, in der Zeit zwischen zwei Fahrzeugen gleichmäßig in Gedanken hochzuzählen. Das Zählergebnis für die Zeit zwischen zwei aufeinanderfolgenden Fahrzeugen ist recht zufällig. Das liegt daran, dass ein Fahrzeug in der Regel nichts vom nächsten Fahrzeug weiß und auch aus einer ganz anderen Motivation als der Vorgänger vor dem Fenster des Hotels vorbeifährt. Der Prozess des Eintreffens von Fahrzeugen vor dem Fenster ist deswegen völlig zufällig. Man kann nämlich nicht vorhersagen, wann das nächste Fahrzeug an der Stelle des Fensters eintrifft und dann vorbeifährt. Deswegen schläft man in der Regel beim Zählen auch recht schnell wieder ein.

Es ist aber für die hier vorgestellte Anwendung ganz interessant, diesen statistischen „Ankunftsprozess“ der Fahrzeuge mit anschließendem Vorbeifahren genauer zu analysieren. Wenn man das Zählergebnis für die Abstände der Fahrzeuge mittelt, dann kann man so die mittlere Verkehrsdichte für die jeweilige Uhrzeit bestimmen. Denn wenn man sich für das Hochzählen um eins etwa eine Sekunde zeitlässt, dann kommt man auf diese Weise zur Zahl der Fahrzeuge pro Sekunde (Kehrwert des mittleren Zählergebnisses). Man erkennt auch schnell, dass ein langer zeitlicher Abstand zwischen zwei Fahrzeugen, relativ zum mittleren zeitlichen Abstand, umso weniger häufig auftritt, je länger er ist. Kurze Abstände treten dagegen häufiger auf. Bei Regentropfen, der beginnen auf ein Zeltdach zu fallen, ist das ganz ähnlich, nicht jedoch bei einem tropfenden Wasserhahn. Der Wasserhahn tropft weitestgehend regelmäßig, was auch deutlich mehr hinderlich beim Einschlafen ist, weil man nämlich ziemlich genau vorhersagen kann, wann der nächste Tropfen fällt. Das ist so, weil das Tropfen des Wasserhahns von der Dichtigkeit ein und desselben Ventils gesteuert wird, und die Tropfen kein Eigenleben haben, was Zufälligkeit erzeugen könnte, so wie beim Regen.

Man nennt Zufallsprozesse wie den „Ankunftsprozess“ der Fahrzeuge oder der Regentropfen stochastische Prozesse im Gegensatz zu den deterministischen Prozessen wie dem Tropfen eines Wasserhahns. Die Eigenheiten eines Zufallsprozesses, sind natürlich schon seit langem ein Thema der mathematischen Statistik und bestens untersucht, weil sie technisch sehr relevant werden können. Wenn sich zum Beispiel vor dem Hotel eine Ampel befände, dann wäre dieser Ankunftsprozess bestimmend für die Länge der Schlange, die sich vor der Ampel bilden würde und so etwas interessiert dann die Verkehrsplaner in einer Stadt.

Ein weiteres Beispiel, wo zufällig eintreffende Ereignisse eine sehr große Rolle spielen, ist die Strahlungs-Messtechnik. Dort müssen die zufällig über der Zeit eintreffenden Strahlungsquanten gezählt werden und die Zählrate berechnet werden bzw. abgeschätzt werden um zum Beispiel die für den Menschen bedeutsame Dosisleistung zu messen. So ist die an einem Ort erzeugte Dosisleistung einer radioaktiven Quelle proportional zu dem von einem Geigerzähler erzeugten Knacken an dieser Stelle und genauso zufällig wie die vor dem Hotel vorbeifahrenden Fahrzeuge. Es gibt noch viele weitere solche zufälligen Ankunftsprozesse, wo es bei der Dimensionierung technischer Systeme entscheidend ist, dass man den Prozess mit Hilfe der Statistik genau beschreiben kann. Dazu gehört beispielsweise das Eintreffen von Datenpaketen in einem Internet-Router oder -Switch oder die

ankommenden Anfragen an ein Serversystem aber auch das Eintreffen von Feinstaub-Partikeln in der Messkammer eines Partikelzählers für Luftqualitätsmessungen.

Ein gewisses Problem, das man aber mit der Zufälligkeit eines Ankunftsprozesses haben kann, ist, dass die Ankunftsereignisse ja auch massiert auftreten können, dann wenn der zeitliche Abstand von gleich mehreren Ankunftsereignissen gegen Null geht. Damit kann eine technische Messeinrichtung, welche die mittlere Ankunftsrate durch Zählen bestimmen soll, schnell an ihre Grenzen geraten, selbst wenn die mittlere Ankunftsrate gar nicht hoch ist. Das ist die Folge der hohen zeitlichen Dynamik in der Zufälligkeit eines Ankunftsprozesses.

Bei der Entwicklung eines Strahlungsmessgeräts z.B. auf der Basis eines Geigerzählers, hat man ein ganz vergleichbares Problem, weil nämlich die Strahlungsquanten am Zählrohr genauso zufällig eintreffen und dabei die zählende Elektronik überfordern, wenn zu viele auf einmal eintreffen. Wenn das Strahlungsmessgerät am Ende eine Dosisleistung anzeigen soll, welche die Gefährdung für den Menschen darstellt, könnte dies zu einer gefährlichen Unterschätzung der Dosisleistung führen, dann nämlich, wenn schnell aufeinandertreffende Zählpulse von der Elektronik einfach „verschluckt“ werden, ohne dass der Anwender das angezeigt bekommt.

Daher ist eine Prüfeinrichtung sinnvoll, die für den Test von solchen Messgeräten statt regelmäßigen Zählimpulsen ohne zeitliche Dynamik, ganz bewusst zufällige Zähl-Ereignisse mit einer vorgegebenen Wahrscheinlichkeitsverteilung und damit hoher zeitlicher Dynamik generiert. Dies soll nun am Beispiel der Prüfung eines Geigerzählers, mit einem Zählimpuls-Generator, der einen Zufalls-Prozess nachbildet, dargestellt werden.

Poisson-Prozess und Poisson-Statistik

Beim Knacken eines Geigerzählers, wie bei den anderen oben genannten Ankunftsprozessen, liegt statistisch gesehen ein sogenannter Poisson-Prozess zu Grunde. Er entsteht immer dann, wenn die Ankunftsabstände der Ereignisse negativ-exponentiell verteilt sind. Ein Strahlungsmessgerät besteht typischerweise aus einer Detektor-Einheit und einer Zähl- und Auswerteeinheit. Die Detektor-Einheit liefert Zählimpulse immer dann, wenn ein Strahlungsquantum vom Detektor registriert wird. Die zeitlichen Abstände dieser Zählimpulse folgen einer negativ-exponentiellen Verteilung. Damit ist die Anzahl der Zählimpulse, die in einem vorgegebenen Zeitintervall von einem Zähler registriert werden, Poisson-verteilt. Wenn der Geigerzähler als Dosisleistungsmessgerät eingesetzt werden soll, wird die zu messende Dosisleistung meist unter der Annahme einer Proportionalität zwischen gemessener Zählrate und der Dosisleistung mit Hilfe eines Konversionsfaktors aus der Zählrate berechnet. Dieser Konversionsfaktor wird mit Hilfe einer Kalibriermessung an einer bekannten Quelle oder durch Vergleich mit einem Referenzinstrument bestimmt.

Die negativ exponentielle Abstandsverteilung ist nun wie folgt definiert:

$p = 0$ für $t < 0$ und $p(t) = \lambda e^{-\lambda t}$ für $t >= 0$

Sie beschreibt den zeitlichen Abstand zwischen zwei Zähl-Ereignissen. Der Mittelwert stellt den mittleren zeitlichen Abstand $1/\lambda$, dar, wobei der Parameter λ die mittlere Anzahl der Ereignisse pro Zeiteinheit bestimmt, was als Zählrate bezeichnet wird. Was man an dieser negativ exponentiellen Verteilung auch gleich erkennen kann, ist, dass die Wahrscheinlichkeit für das Auftreten eines Abstands zwischen zwei Zählpulse mit zunehmender Abstandszeit t exponentiell abnimmt. Das heißt kurze Abstände zwischen zwei Zählpulsen treten viel häufiger auf als lange Abstände.

Wenn nun ein Messzeitintervall T vorgegeben wird, um die Anzahl an detektierten Strahlungsquanten zu zählen, dann fällt eine zufällige Anzahl k Zählimpulse in dieses Intervall. Die Anzahl k, der in jedem Messzeitintervall gezählten Zählimpulse, folgt der Poisson-Verteilung, sofern die Abstände negativ-exponentiell verteilt sind.

Die Poisson-Verteilung ist gegeben durch:

$P(k, \mu) = N^k e^{-N} / k!$

Dabei ist der Parameter N der Mittelwert, also die im Mittel detektierten Strahlungsquanten bzw. erzeugten Zählimpulse. Dieser Mittelwert ergibt sich beim Poisson-Prozess auch aus $N = \lambda * T$ wenn T das Messzeitintervall ist und λ die mittlere Ereignisrate. Besonders bemerkenswert ist bei dieser Verteilung darüber hinaus, dass die Streuung $\sigma = \sqrt{N}$ beträgt. Das bedeutet, dass sie bereits durch den Mittelwert festgeschrieben ist, anders als beispielsweise bei der Gauß-Verteilung, wo die Streuung vom Mittelwert beliebig unabhängig sein kann.

Dividiert man nun die detektierte Anzahl k, durch die Dauer T eines Messzeitintervalls, ergibt sich auf Grund der Zufälligkeit nur ein Schätzwert R = k/T für die mittlere Zählrate λ. Werden aber genügend Messungen immer mit diesem Intervall T durchgeführt, dann kann die mittlere Zählrate durch Mittelwertbildung bestimmt werden, welche dann dem Parameter λ aus der Abstandsverteilung entsprechen muss. Interessant ist schließlich noch, dass die aus einer Messung mit dem Intervall T geschätzte Zählrate R eine Streuung um den Mittelwert λ hat, die sich wie folgt ausdrücken lässt:

$R = \lambda \pm \lambda / \sqrt{N}$.

Es ist also ein wesentliches Merkmal, dass die Streuung eines Poisson-Prozesses auch dazu führt, dass auch die im Messintervall T geschätzte Zählrate R um einen Faktor kleiner ist als sein Mittelwert, wobei der Faktor die Wurzel aus der Anzahl der in dem Messzeitintervall T im Mittel eintreffenden Ereignisse N ist.

Der Algorithmus zur Zählimpulserzeugung mit zufälliger Abstandsverteilung und seine Implementierung auf einem Mikrocontroller

Die Zähl- und Auswerteeinheit in einem Strahlungsmessgerät wird meist auf der Basis eines Mikrocontrollers implementiert. Dabei ist nun sehr wesentlich, dass die Zählimpulse in zufälligen Abständen eintreffen und die Anzahl k Zählimpulse entweder in einem vorgegebenen Zeitintervall T gezählt werden müssen (Zeitvorwahl), oder dass ein zu erreichender Zählerstand k vorgegeben wird, wobei dann die für das Erreichen des vorgegeben Zählerstands benötigte Messzeit T gemessen wird (Impulsvorwahl). Schließlich bestimmt die Auswerteeinheit aus Zählerstand und Messzeit den Schätzwert R für die mittlere Zählrate λ. Er ergibt sich wie oben beschrieben aus R = k/T.

Um diese Zähl- und Auswerteeinheit in realistischer Weise mit Hilfe von zufälligen Zählimpulsen mit der ganzen zeitlichen Dynamik eines Poisson-Prozesses zu testen, müssen zunächst Impulse erzeugt werden, die in der Amplitude, Impulsform und Impulsdauer den Impulsen der Detektor-Einheit entsprechen.

Eine Idee für einen Algorithmus zur Erzeugung der notwendigen Impuls-Abstandsverteilung ist, die in den üblichen Entwicklungsumgebungen (IDEs) für Mikrocontroller bereitgestellte Routine zur Zufallszahlenerzeugung random() zu nutzen. Sie erzeugt in aller Regel aber nur Zufallszahlen, die in

einem vorgebbaren Bereich gleichverteilt sind. Durch eine Skalierung kann man sie leicht auf Gleitkommazahlen, die zwischen 0 und 1 gleichverteilt sind, abbilden.

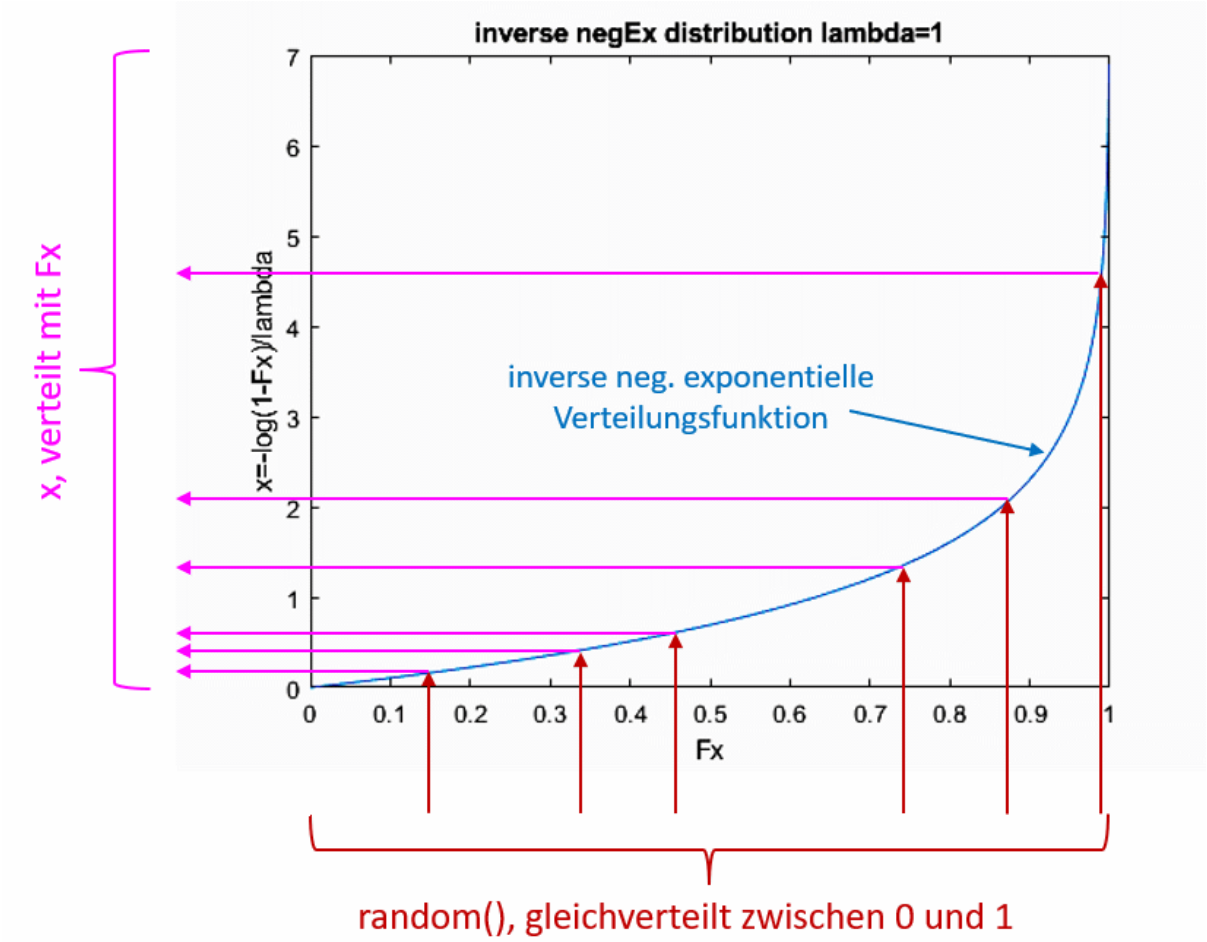

Abb. 1: Abbildung gleichverteilter Zufallszahlen auf die gewünschte Verteilungsfunktion

Nun kann man sich jedoch überlegen, dass die Verteilungsfunktion einer Zufallsvariable den Wertebereich der Zufallsvariablen auf Wahrscheinlichkeiten zwischen 0 und 1 abbildet. Umgekehrt muss dann die Inverse der Verteilungsfunktion Werte zwischen 0 und 1 auf den Wertebereich der Zufallsvariablen abbilden. Das bedeutet aber, dass gleichverteilte Zufallszahlen, die z.B. an der Inversen der negativ-exponentiellen Verteilungsfunktion in Zufallszahlen abgebildet werden, die gewünschte Verteilungs-Eigenschaft der negativ-exponentiellen Verteilung bekommen (Abb. 1).

Dieses Grundkonzept des Algorithmus lässt sich im Falle eines Poisson-Prozesses leicht in einem Mikrocontroller realisieren. Poisson-verteilte Zählimpulse müssen, wie oben beschrieben, diese negativ-exponentielle Abstandsverteilung haben. Wenn man in einer Schleife einen Impuls an einem I/O Pin generiert und in die Schleife eine Wartezeit einbaut, die negativ-exponentiell verteilt ist, dann muss die in einem Zeitintervall erzeugte Zählimpulsanzahl Poisson-verteilt sein, so die Theorie. Als Abbildungsfunktion benötigt man daher die Inverse zur negativ-exponentiellen Verteilungsfunktion. Diese lautet:

$x = -\log(1-Fx)/\lambda$

und lässt sich in den meisten IDEs mit Hilfe des Logarithmus aus der Mathematik Bibliothek sehr einfach berechnen. Darin stellt Fx die zufälligen Eingangswerte zwischen 0 und 1 dar, und x die Zufallszahlen, die in Folge der Abbildung negativ-exponentiell verteilt sind. In der Arduino-IDE stellt

sich der C++ Code für die Zählimpuls-Generierung mit Poisson-Statistik daher auf ganz einfache Weise so dar:

```
void setup() {
  Serial.begin(9600);
  pinMode(5,OUTPUT);
  digitalWrite(5,LOW);
}

void loop() {
  double lambda = 0.001; // --> 1000cps
  //double lambda = 0.0005; // --> 500cps
  double rnd = random(2147483648)/2147483647.00;
  int negEx = round(-log(1-rnd)/lambda);
  delayMicroseconds(negEx);
  // Serial.println(negEx);
  digitalWrite(5,HIGH);
  delayMicroseconds(10);
  digitalWrite(5,LOW);
}
```

Listing 1: Implementierung des Zählimpuls-Generators in der Arduino IDE mit C++

In der Setup-Routine, welche nur einmal zu Programmbeginn ausgeführt wird, wird lediglich eine Kommunikation des Mikrocontrollers mit dem PC über einen COM-Port festgelegt, der mit 9600 Baud auf der seriellen Schnittstelle läuft. Zudem wird noch der IO-Pin 5 als Pin festgelegt, welcher die Zählimpulse ausgeben soll, und der mit LOW initialisiert wird.

In der Schleife Loop() sind einige Zeilen auskommentiert, die auf Bedarf aktiv geschaltet werden können, indem man das Kommentar entfernt. Es wird zunächst eine Variable lambda erklärt, welche die mittlere Zählrate λ als Wert zugewiesen bekommt. Das könnte man im Prinzip auch mit einer #define Anweisung bewerkstelligen. Wichtig ist hier aber, dass lambda in der Zeiteinheit Mikrosekunden angegeben werden muss, weil die weiter unten verwendete delayMicroseconds() Routine ihren Delay-Parameter in Mikrosekunden benötigt. Da die Absicht bestand, den Code auf einem 32bit-Prozessor ablaufen zu lassen, wurde hier angenommen, dass 2^31-1 die größte Zahl ist, welche von der Routine random() generiert werden kann, wenn man sie mit dem Parameter 2^31 aufruft. Durch Division mit 2^31-1 erreicht man somit eine Skalierung auf double-Zufallszahlen in der kleinstmöglichen Granularität, die zwischen 0 und 1 gleichverteilt sind. Mit diesen Zahlen geht man in die Inverse der negativ-exponentiellen Verteilungsfunktion und erhält damit eine negativ-exponentiell verteilte Zufallszahl als Wert der Variablen negEx, der bei jedem Schleifendurchlauf erneuert wird. Danach pausiert die Programmausführung um die zufällige Impuls-Abstandszeit negEx. Optional kann man sich diese Abstandszeit z.B. mit dem Serial Monitor auf den PC ausgeben lassen, um damit ihre Statistik zu untersuchen. Danach wird auf dem IO-Pin 5 der Zählimpuls erzeugt, dessen Impulsdauer

hier zu 10us gewählt wurde, was ausreichend ist, um die Zähl- und Auswerteschaltung, die getestet werden soll, zu triggern.

Unter den Mikrocontrollern, die untersucht wurden, erwies sich der Arduino Portenta H7 Lite als am geeignetsten, um den Zählimpuls-Generator zu implementieren. Er basiert auf einem STM32H747XI 32-Bit Prozessor, der über einen ARM Cortex®-M7 Kern verfügt, der auf 480MHz läuft, und einen M4-Core auf 240 MHz, der hier aber nicht gebootet werden muss. Er hat gegenüber anderen Mikrocontroller Boards den Vorteil, dass sich durch die Dauer der Abarbeitung der Schleife praktisch keine zusätzliche Totzeit ergibt, und die Zählimpulse mit einem vernachlässigbaren Minimalabstand voneinander erzeugt werden können.

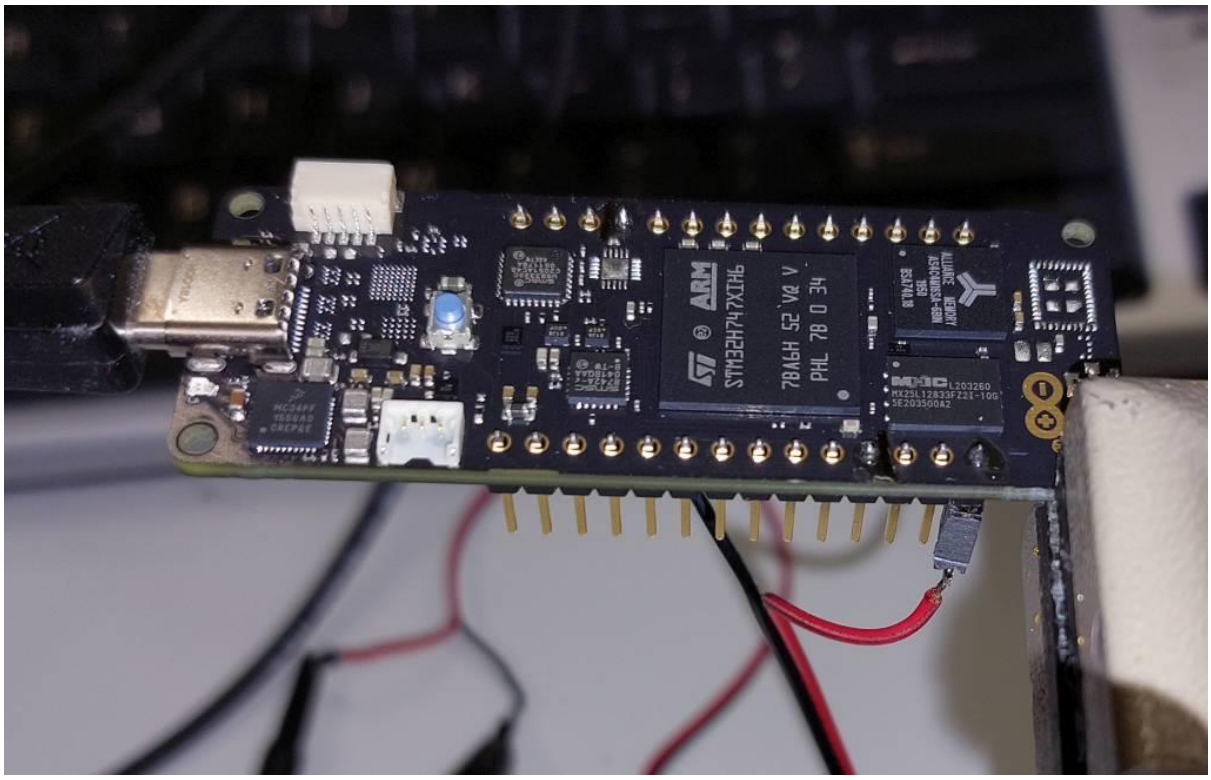

Abb. 2: Implementierung des Zählimpuls-Generators auf einem Arduino Portenta H7 Lite

Prüfung der Zähl- und Auswerteeinheit eines Geigerzählers für die Dosisleistungs-Messung mit dem Zählimpuls-Generator

Zunächst wurde die Ausgabe der Werte negEx aktiviert, um die Korrektheit der berechneten Abstandswerte der Zählimpulse zu prüfen. Die Daten wurden aufgezeichnet und geplottet (blaue Kurve) und mit der theoretischen negativ-exponentiellen Abstandsverteilungsfunktion (rot gestrichelt) verglichen. Dabei ergibt sich eine sehr gute Übereinstimmung (Abb. 3). Hierzu soll noch auf eine Eigenheit der Arduino IDE hingewiesen werden. Mit neueren Versionen (z.B. 2.2.1) ist es nicht mehr möglich, Daten aus dem Fenster des Serial Monitor einfach mit Copy und Paste in eine Datei oder in Excel zu kopieren. Dazu sollte eine ältere Version z.B. 1.8.x verwendet werden und ein vergleichbares Board ausgewählt werden, wenn es das verwendete Board in dieser Version noch nicht gab (also z.B. Due statt Portenta).

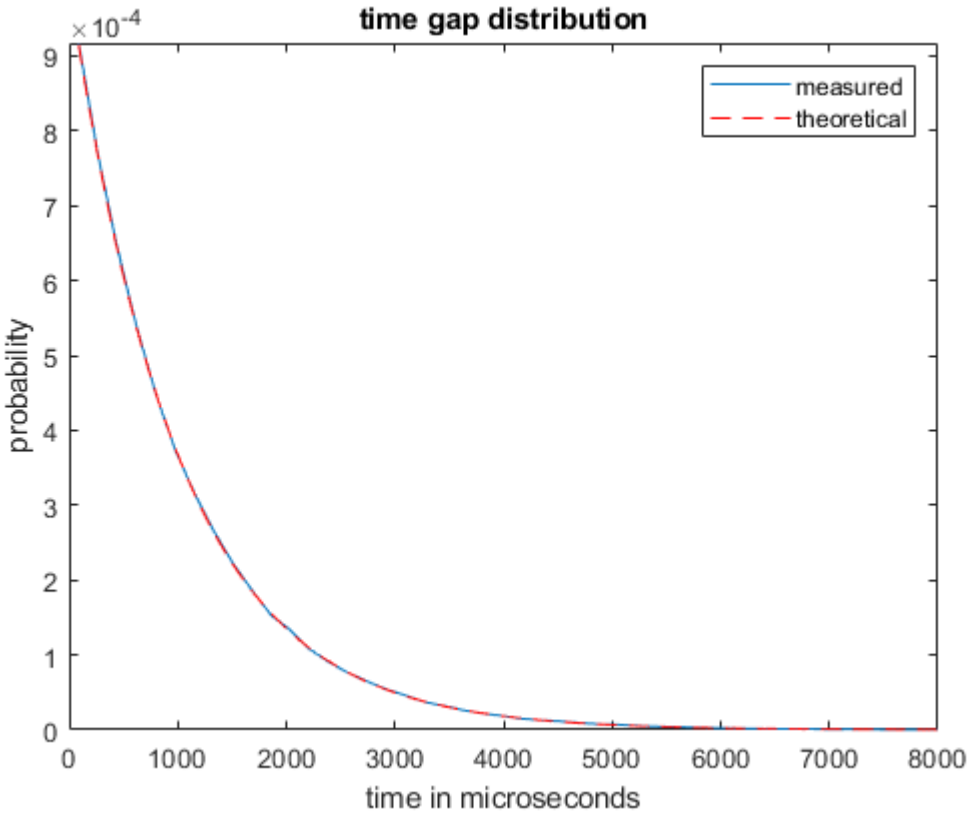

Abb. 3: Vergleich der vom Mikrocontroller berechneten und der theoretischen Abstandsverteilung der zufälligen Zählimpulse

Lässt man den Code nun für λ=1000cps ablaufen (lambda = 0.001), und gibt das Signal an Pin 5 auf ein Oszilloskop, dann kann man sehr deutlich die zufällig erzeugten Zählimpulse erkennen, die teilweise nah aufeinander auftreten, teilweise aber auch mit größerem Abstand (Abb. 4a). Schaltet man die Nachleuchtdauer des Oszilloskops (Persistence) auf unendlich, dann überlagert das Oszilloskop-Display die generierten Zählimpulse und man kann verfolgen, wie sich allmählich die Zeitachse füllt, wobei Impulse in größeren Zeitabständen, relativ zum triggernden Impuls, umso seltener dazukommen, je größer der Abstand ist. Das bestätigt die negativ-exponentielle Verteilung der Abstände (Abb. 4b).

Außerdem kann man mit dem Oszilloskop auch feststellen, dass auch Zählimpulsabstände unter 20µs noch korrekt auftreten. Das bedeutet, dass auch Zählimpulsraten über 10000cps möglich sein müssten.

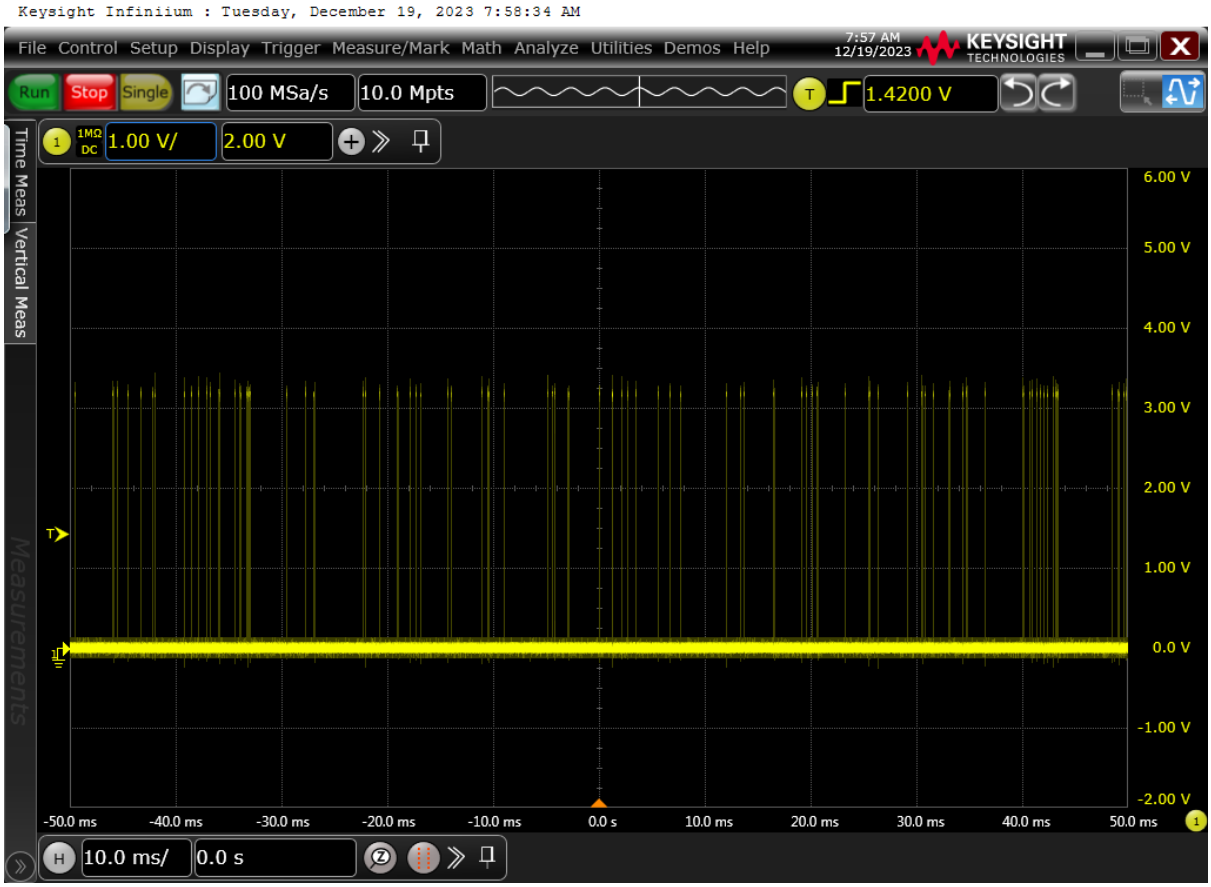

Abb. 4a: Darstellung der erzeugten Zählimpulse auf dem Oszilloskop (einzelne Aufzeichnung)

Nun kann man aber zusätzlich noch die Poisson-Statistik dadurch überprüfen, dass man die Zähl- und Auswerte-Einheit eines Geigerzählers, die hier zum Beispiel auf einem Arduino MKR ZERO Mikrocontroller implementiert wurde, als Testobjekt verwendet und sich pro Messung die ermittelten Schätzungen R für die mittlere Zählrate λ ausgeben lässt. Hier wurde eine Auswerteeinheit mit Impulsvorwahl verwendet, welche die Zeit misst, die benötigt wird, bis 100 Zählimpulse eingetroffen sind. Wertet man nun ca. 2000 Messwerte aus, dann ergibt sich ein Mittelwert von 998.4cps und eine Streuung von 99.6cps (Abb. 5). Das aber bedeutet, dass die Bedingung für die Poisson-Statistik $\sigma = \lambda / \sqrt{N}$ sehr gut erfüllt ist. Daher kann man nun mit sehr großer Sicherheit sagen, dass die Poisson-Statistik bei der Zählimpuls-Generierung zuverlässig erfüllt ist, und der Zählimpuls-Generator wie gewünscht funktioniert.

Das Listing für die Zähl- und Auswerteeinheit des Geigerzählers, die auf dem Arduino MKR ZERO Mikrocontroller verwendet wurde, ist in Listing 2 gezeigt.

Um auf die zufällig eintreffenden Zählimpulse reagieren zu können, die von der Detektor-Einheit her geliefert werden, wird hier die Interrupt-Logik des Prozessors verwendet. Sie unterbricht die aktuelle Programmausführung, verzweigt in die Interrupt Service Routine (ISR), kehrt dann zurück und fährt mit der normalen Programmausführung fort. In der Interrupt Service Routine wird lediglich die globale Variable counter hochgezählt, welche die Anzahl der registrierten Zählimpulse in der laufenden Messung darstellt.

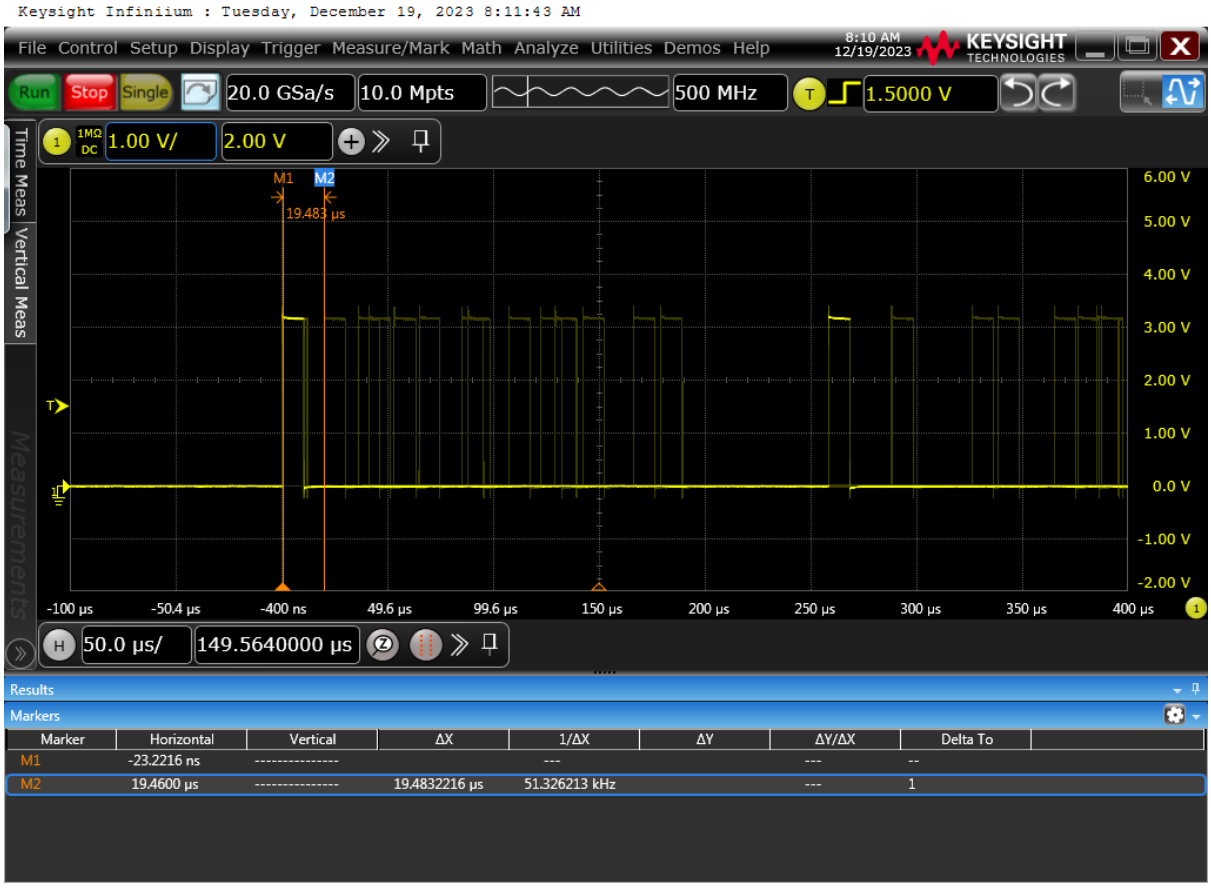

Abb. 4b: Darstellung der erzeugten Zählimpulse auf dem Oszilloskop (mehrfache Aufzeichnung mit „infinite persistence“ Einstellung des Displays)

```
#define MAXCNT 100
volatile int counter = 0;
unsigned long oldTime = 0;

void count()
{
  counter++;
}

void setup(){
Serial.begin(9600);
attachInterrupt(digitalPinToInterrupt(0), count, FALLING);
}

void loop(){
  unsigned long time;
  unsigned long dt;
  float rate;
  int err;
  if (counter == MAXCNT) {
    detachInterrupt(digitalPinToInterrupt(0));
    time = millis();
    dt = time-oldTime;
    rate = (float)MAXCNT*1000.0/(float)dt;
    Serial.println(round(rate));
```

```
    oldTime = millis();
    counter = 0;
    attachInterrupt(digitalPinToInterrupt(0), count, FALLING);
  }
}
```

Listing 2: Code für die Zähl- und Auswerteeinheit auf dem Arduino MKR ZERO

Im hier verwendeten Code wird dazu mit dem Statement:

```
attachInterrupt(digitalPinToInterrupt(0), count, FALLING)
```

vereinbart, dass die Zählimpulse an Pin 0 anliegen um den Interrupt mit der fallenden Flake auszulösen um damit in die ISR count() zu verzweigen. Die normale Programmausführung läuft in der Schleife Loop(). In dieser Schleife wird abgeprüft, ob der Zählerstand den Wert MAXCNT erreicht hat. Das ist jedoch die meiste Zeit nicht der Fall, so dass damit die Schleife auch bereits abgearbeitet ist. Erreicht jedoch der Zählerstand den Wert MAXCNT, dann wird die vereinbarte Interrupt-Behandlung vorübergehend deaktiviert, und es wird die Auswertung gemacht. Für die Auswertung wird die Zeit bestimmt, die zum Erreichen des Zählerstands nötig war (Impuls-Vorwahl-Methode). Aus MAXCNT und der gemessenen Zeit wird dann der Schätzwert R für die mittlere Zählrate bestimmt, und der Wert davon über den Serial Monitor ausgegeben.

Dieser Code hat sich als relativ robust erwiesen und konnte auf den schnelleren Arduino-Boards bis mehrere 10kcps erfolgreich getestet werden.

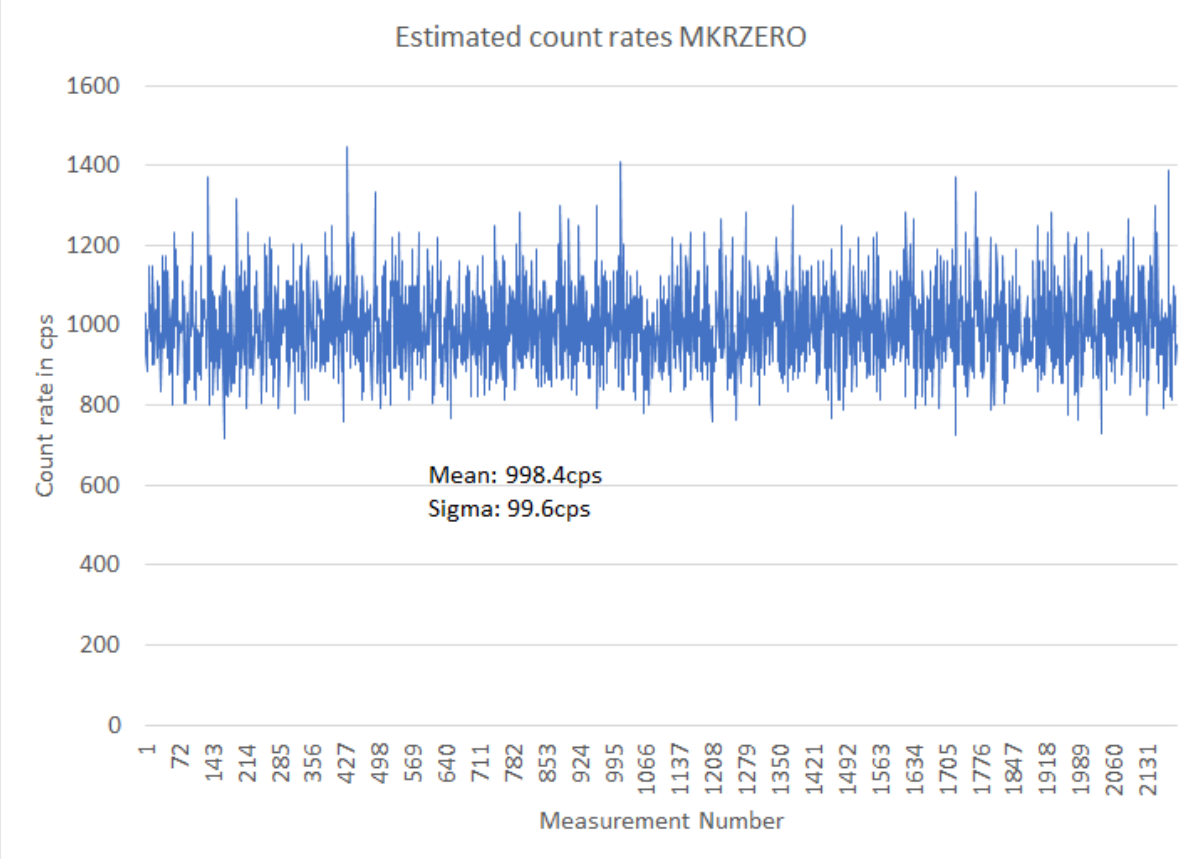

Abb. 5: Prüfung der mit einer Auswerteeinheit gemessenen Schätzwerte für die mittlere Zählrate, berechnet aus der Zeit für 100 Zählimpulse

Spektrale Analyse des zufälligen Zählimpuls-Signals

Ein weiterer interessanter Aspekt, die sich mit dem Zählimpuls-Generator mit zufälliger Impulsabstands-Verteilung leicht beantworten lässt, ist die Frage, wie das Fourier-Spektrum (bzw. die spektrale Leistungsdichte) eines idealen Geigerzähler-Signals aussieht. Es ist ja hinreichend bekannt, dass das Fourier-Spektrum eines periodischen Rechteck-Signals ein Sin(x)/x bewertetes Linienspektrum ist, welches durch das Tastverhältnis bestimmt ist. Aber wie sieht das Fourier-Spektrum aus, wenn die Zählimpulse zufällig kommen? Das könnte für die Bemessung der Bandbreite einer analogen Nachbearbeitung durchaus von Bedeutung sein, zumindest wenn ein Szintillationsdetektor mit großem Detektor-Kristall die Zählimpulse mit einer Rate in der Größenordnung von 10kcps schickt.

Um diese Frage zu beantworten, kann man den Zählimpuls-Generator mit einem Spektrum-Analysator vermessen. Das erzeugte Spektrum sieht dabei absolut plausibel aus, was ebenfalls unter Beweis stellt, dass der Zählimpuls-Generator wie erwartet funktioniert. Das Fourier-Spektrum (Wurzel der spektralen Leistungsdichte) eines Geigerzählers ist demnach ein gleichverteiltes, lückenloses Rauschspektrum, welches mit einem Sin(x)/x bewertet ist. Die Sin(x)/x Bewertung hat ihre Nullstellen bei N/Ti, mit N = 1…∞, wenn Ti die Breite des rechteckförmigen Zählimpulses ist. Da im gezeigten Beispiel des Zählimpuls-Generators die Impulsdauer zu 10µs gewählt wurde, ergibt sich so die erste Nullstelle bei 100kHz. Die Änderung der mittleren Zählimpulsrate ändert die Form des Spektrums nicht, sie skaliert nur die Amplitude. Bei niedrigeren Raten ist es jedoch schwierig zu erreichen, dass der Spektrum-Analysator auch die volle Zufälligkeit sieht, da es dann geraume Zeit dauert, bis genügend Zählimpulse eingetroffen sind, um ein repräsentatives Spektrum darstellen zu können.

Bei einem anders als rechteckig geformten Zählimpuls eines realen Zählrohrs oder eines Photomultipliers muss man das gleichverteilte Rauschspektrum statt mit dem Sin(x)/x Spektrum eines einzelnen Rechteck-Impulses mit dem Spektrum des realen einzelnen Detektorimpulses multiplizieren. Insgesamt erhält man also ganz dasselbe Spektrum wie bei einer langen Pseudorandom-Zufallsfolge (PRBS), wie sie in der Datenkommunikation üblich ist, die mit einer Periodizität von 2^31-1 abläuft und eine Datenrate von 1kbps aufweist. Das ist auch insofern logisch, da ja die Autokorrelierte beider Zufallssequenzen Null ist, sobald die Verschiebung mehr als Ti beträgt und für kleinere Verschiebungen eine Dreiecks-Funktion aufweist. Das ist wiederum so, weil beide außerhalb von Ti völlig zufällig sind, was die Lage der Zählimpulse auf der Zeitachse anbelangt. Das Fourierspektrum hat also in beiden Fällen die Sin(x)/x Bewertung, weil das Leitungsdichte-Spektrum das Betragsquadrat davon ist und die Fouriertransformierte der dreieckförmigen Autokorrelations-Funktion darstellt (Abb. 6).

Wenn man sich aber das Spektrum mit sehr hoher Frequenzauflösung anschauen würde, dann würde man natürlich merken, dass es kein ganz kontinuierliches Spektrum ist, weil sich das Zeitsignal mit 2^31-1 wiederholt. Es liegt also tatsächlich doch ein sehr dicht verteiltes Linienspektrum dahinter, mit einem Linienabstand von 1/((2^31-1)*Ti), die der Spektrum-Analysator im gezeigten Spektrum nicht auflöst, der aber im µHz Bereich liegt und daher ignoriert werden kann.

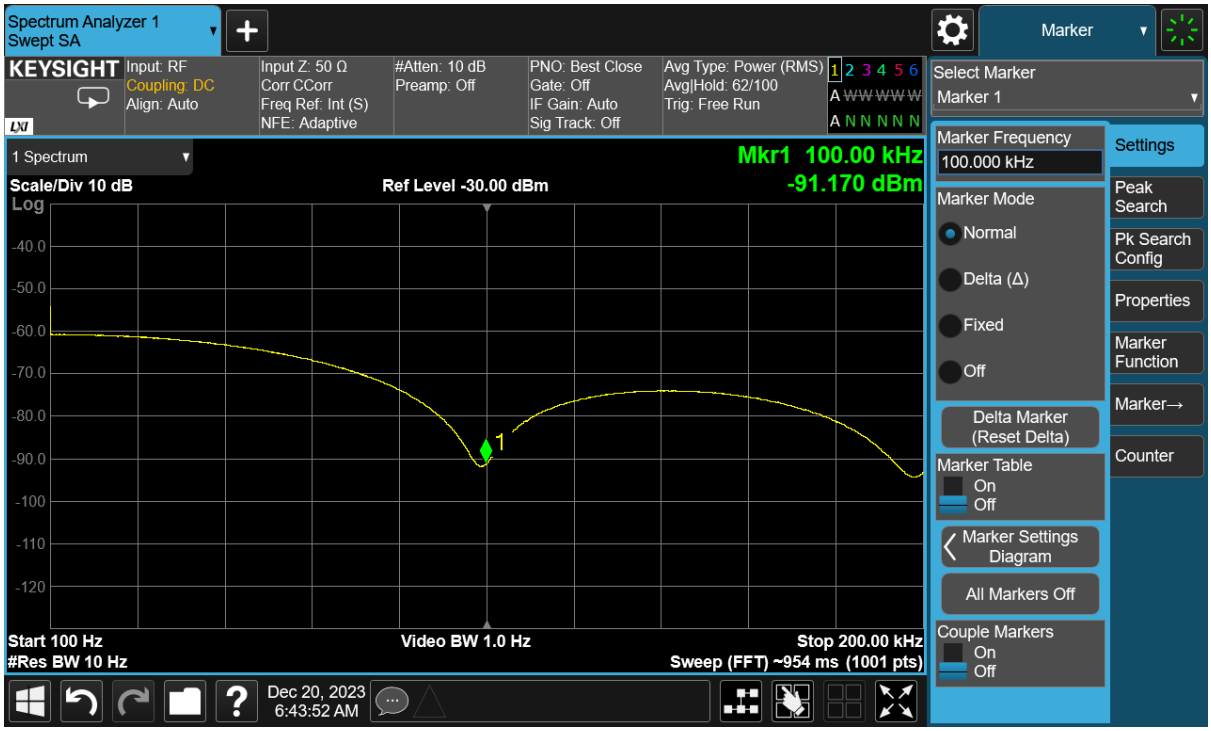

Abb. 6: Fourier-Spektrum (in dB) des zufälligen Zählimpuls-Signals mit einer Impulsdauer von 10µs

Zusammenfassung

Abschließend kann gesagt werden, dass die Implementierung des Zählimpuls-Generators mit einer zufälligen Impuls-Abstandsverteilung, welche die Poisson-Statistik erfüllt, auf einem marktüblichen Mikrocontroller sehr einfach möglich ist. Bei der beispielhaften Prüfung einer Zähl- und Auswerte-Einheit für einen Geigerzähler auf der Basis eines Arduino MKR ZERO Mikrocontrollers, liefert die Implementierung des Zählimpuls-Generators auf dem Arduino Portenta H7 zuverlässige Ergebnisse für Zählraten bis 1kcps, was für die meisten Selbstbau-Dosisleistungsmessgeräte mit Geiger-Müller Zählrohr mehr als ausreichend ist. Da das Konzept nicht nur auf die hier verwendete spezielle, negativ-exponentielle Verteilungsfunktion beschränkt ist, lässt sich der implementierte Algorithmus auch auf beliebig andere Verteilungen der Impuls-Abstände anpassen, sofern diese geschlossen mathematisch darstellbar sind, und entsprechende Bibliotheksroutinen für die dafür notwendigen Inversen zu den Verteilungsfunktionen verfügbar sind. Aus den Oszilloskop-Messungen kann geschlossen werden, dass auf dem Arduino Portenta H7, genau wie auf anderen modernen Mikrocontroller Boards der oberen Leistungsklasse, auch maximale Zahlimpulsraten von über 10kcps erreichbar sein müssten.

Literatur

/1/ J. Henniger, R. Schwierz; Grundlagenpraktikum Radiometrische Messung 1 - Geiger Müller Zählrohr, Poisson-Verteilung

https://tu-dresden.de/mn/physik/ressourcen/dateien/studium/lehrveranstaltungen/praktika/pdf/RM1_NP.pdf

/2/ Bernd Laquai; Die Statistik des Zerfalls; http://www.opengeiger.de/StatistikDesZerfalls.pdf