trueRNG & DICE - echte Zufallszahlen erzeugen

Rauschquelle, Zufallszahlengenerator und elektronischer Würfel in einem

Konzept

Das Grundprinzip dieser Schaltung ist nicht neu. Ein Rauschgenerator erzeugt ein chaotisches Ausgangssignal mit zufälligen Amplituden. Man drückt einen Knopf oder sendet eine Anweisung, um eine Zufallszahl zu generieren. Zu diesem Zeitpunkt wird ein Wert genommen, der völlig zufällig ist. Sind z.B. 1.000 Werte gleich verteilt? Die Antwort auf diese Frage ist uns wichtig, denn wir wollen einen echten Würfel haben. Wenn er oft genug geworfen wird, soll jede Würfelzahl annähernd gleich häufig auftreten.

Da wir es gerne genau nehmen, enthält unser Projekt einen Chiquadrat-Test, mit dem wir überprüfen können, ob die generierten Zahlen der Gleichverteilungshypothese genügen.

Folgendes leistet **trueRNG & DICE**:

- generiert eine einzelne Zufallszahl auf Abruf (Befehl oder Knopfdruck)
- generiert bis zu 200 Zufallszahlen auf Abruf (Die Anzahl ist einstellbar)
- wenn die generierten Zahlen den Chiquadrat-Test nicht bestehen, werden bis zu 9 weitere Läufe ausgeführt, um gleichverteilte Zufallszahlen zu gewinnen
- das Projekt bietet eine Abgleichhilfe für die Einstellung des Analogteils
- ein Rauschsignal von ~3 V_{ss} steht am Analogausgang bereit

Umfang des Projektes

Folgende Informationen stehen bereit:

- technische Beschreibung mit Schaltplan
- Sämtliches Sketche zum Projekt
- div. Abbildungen zum Prototyp und ein Video des korrekt eingestellten Rauschsignales
- BOM (Materialliste)

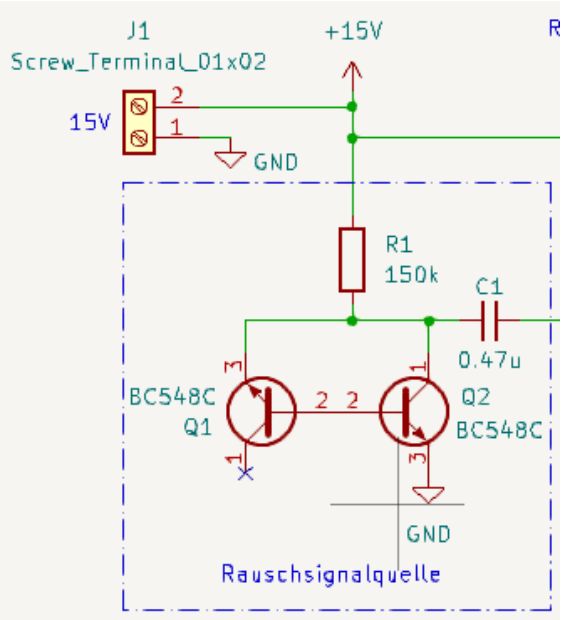
Es handelt sich um ein Open Source Projekt für den privaten Gebrauch. Dennoch unterliegt es den Bestimmungen des Copyright. Wer das Projekt für nicht-kommerzielle Anwendungen nutzen und/oder verändern möchte, kann dies gerne tun. Kommerzielle Anwendungen erfordern meine Zustimmung.

Wenn sich wieder ausreichend viele Interessenten für eine Leiterplatte finden, werde ich eine erstellen und ggfls. eine Sammelbestellung organisieren. Bitte bei Interesse bei mir melden: **dl1mkp@darc.de**

Grundlagen

Ein Transistor, der „falsch herum“ (siehe Schaltung) angeschlossen wird, leitet einen Strom, sobald seine maximale „reverse bias voltage“ überschritten wird. Diese Eigenschaft der pn-Diode bezeichnet man auch als „breakdown“, denn eigentlich soll der ideale Transistor in dieser Schaltung sperren. Wenn man die Spannung weiter erhöht, steigt dieser Strom weiter schnell an.

Es sind der quantenmechanische Tunneleffekt und der „Lawineneffekt“ - ein Elektron stößt mehrere andere Elektronen „aus dem Gitter“ - (Physikerkollegen mögen jetzt bitte Kaffee trinken gehen und auf der nächsten Seite weiter lesen), die Ladungsträger „chaotisch“ durch die Bandgap schubsen. Wie stark die Effekte sind, hängt von vielen Faktoren ab, z.B. der Dotierung und der Sperrschichttemperatur.



Die Rauschquelle besteht in unserem Fall aus zwei npn-Kleinsignaltransistoren BC548C. Die Rauschquelle ist Q1, während Q2 als Stromquelle dient.

Der minimale Strom durch R1 reicht völlig aus, um das Rauschsignal zu erzeugen. Wichtig ist die Betriebsspannung von + 15 V, die deutlich über der reverse bias voltage des BC548 liegt (ca. 8,5 V).

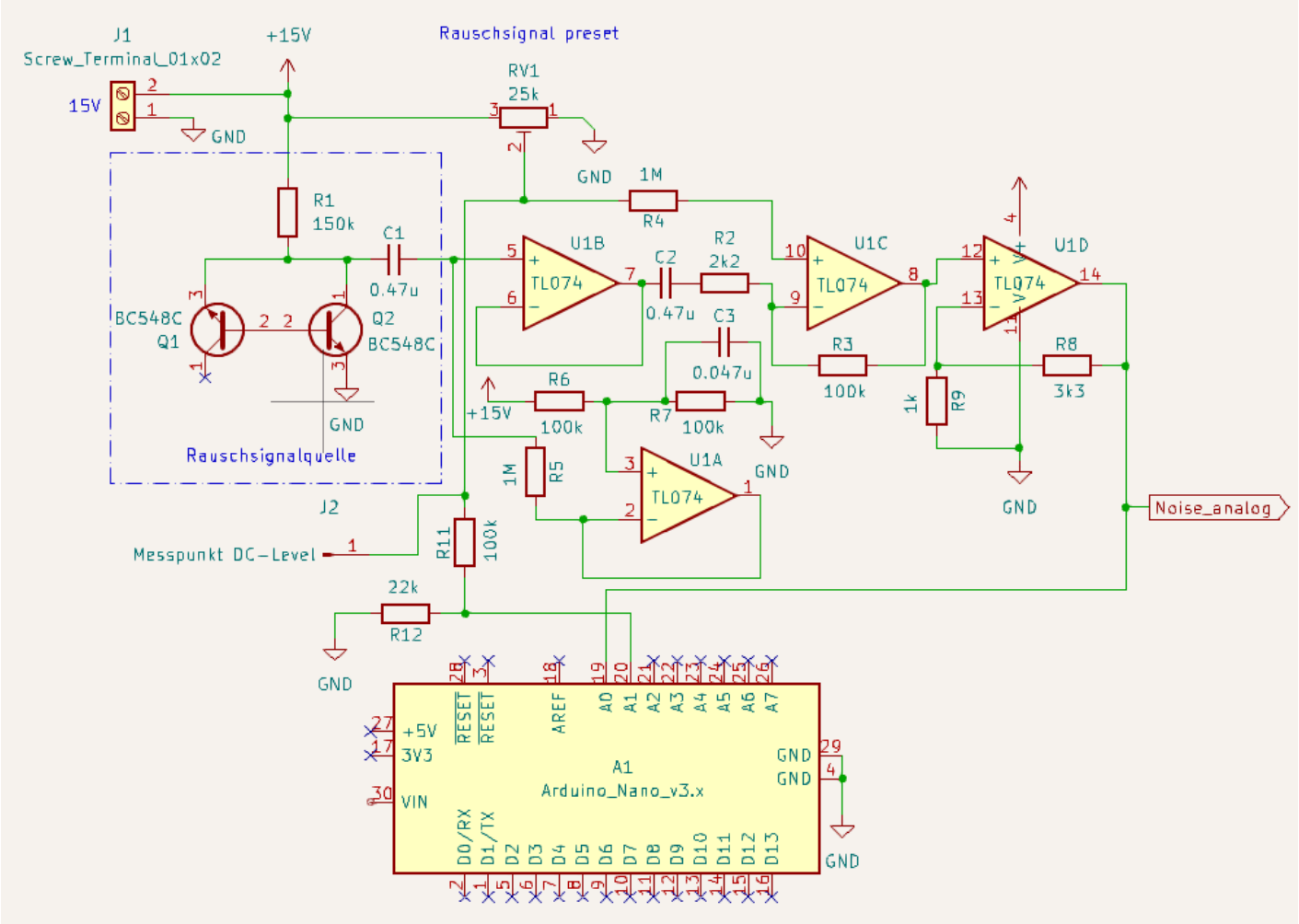
Das Rauschsignal liegt bei Zimmertemperatur im Bereich von 35 - 100 mV_{pp}. Es gelangt über den Koppelkondensator C1 zu dem nachfolgenden Verstärker.

Bei meinen Tests habe ich gelernt, dass einige Transistoren deutlich besser funktionieren, als andere. Empfehlen kann ich grundsätzlich: BC107, BC546, BC547, BC548 und den Klassiker 2N2222. Einzelne 2N2222 bringen einen Output von bis zu 400 mV_{pp}.

Die technische Herausforderung

Die tatsächliche Herausforderung liegt darin, jede systematische Störung des Rauschsignales zu vermeiden. So verursacht eine 50 Hz – Stromnetzeinstreuung einen systematischen Fehler, der uns den ganzen schönen Zufall zerstören kann. Also aufpassen, wo Trafos und andere Störquellen positioniert sind.

Die Schaltung



Unser Rauschsignal ist am Ausgang der Schaltung bei „Noise_analog“ verfügbar. Sie können es in einen Verstärker einspeisen und am Ausgang des Verstärkers mit einem Spektrumanalyzer überprüfen, ob der Signalverlauf über die Bandbreite des Verstärkers Amplitudenverluste erlitten hat.

Das Herz der Schaltung ist die Rauschsignalquelle mit den Transistoren Q1 und Q2. Dieser Teil wurde gerade erklärt.

Die nachgeschalteten Operationsverstärker U1B, U1C und U1D dienen als Buffer (U1B) und Verstärker. Dabei wird U1C über das Trimmpoti RV1 am nichtinvertierenden Eingang mit einer Vorspannung belegt. Damit wird sowohl die Verstärkung beeinflusst, als auch der Arbeitspunkt. Wir müssen sicherstellen, dass der Operationsverstärker - der bis zu 15 V liefern kann - nur das verstärkte Rauschsignal liefert, das maximal 3 V haben darf, denn mehr verträgt der **nano V3** an seinem Analogeingang A0 nicht.

Wir bauen zuerst den Rauschgenerator auf und gleichen ihn ab. Erst wenn dieser Teil perfekt läuft und die Ausgangsspannung stimmt, wird der digitale Teil aufgebaut und verbunden.

Da wir von der Rauschquelle an Pin 5 des TL074 ein kapazitiv ausgekoppeltes Signal (von C1) erhalten, das um den Nullpunkt schwankt, addiert U1A über einen 1MΩ-Widerstand R5 ca. ½ der 15 V, also ca. 7,5 V Gleichspannung dazu und hebt so die Nulllinie von -7,5 V auf ~ +0,25 Volt. Das Rauschsignal bewegt sich jetzt zwischen +0,25 Volt und der Maximalamplitude. So vermeiden wir die Notwendigkeit einer negativen Spannungsquelle und kommen mit einer Stromversorgung von +15 V aus.

Die Schaltung benötigt Ihre Aufmerksamkeit an mehreren Punkten. Zum einen ist es so, dass es bei den Transistoren ziemliche Exemplarstreuungen gibt. Ich habe mehrere Versionen der Schaltung aufgebaut. Bei einer Version brauchte ich U1D nur als Buffer (Verstärkung = 1), aber nicht mehr als Verstärker zu schalten, weil die Transistoren 2N2222 viel Rauschsignal brachten. In diesem Fall wird R8 durch eine Drahtbrücke ersetzt, R9 entfällt ganz. Das können Sie leider nur ausprobieren und am Oszilloskop testen.

Des weiteren ist die Einstellung von RV1 kritisch. Das Poti wird zum Abgleich ganz zur Gnd-Seite gedreht. Am besten messen sie die Spannung am Potiabgriff. Sie muss anfangs 0 V sein. Dann einschalten und das 10-Gang Poti langsam aufdrehen, bis sie ein sauberes Rauschsignal bekommen, das eine Amplitude von ca. 2,5 V hat. Dazu stelle ich Ihnen ein Video [2] als Hilfe zur Verfügung.

Das Video zeigt Ihnen, dass das Signal oberhalb der Nulllinie „tanzt“. Dennoch ist zwischen der unteren Markierung am Oszilloskop und der kleinsten Amplitude des Rauschsignales ein kleiner Gleichspannungsanteil von ca. +0,25 V zu erkennen, den wir später im Sketch „heraus rechnen“.

RV1 ist typisch so eingestellt, dass die Vorspannung ≤ 0,3 V liegt. Bitte achten Sie darauf, dass die Ausgangsamplitude des Rauschsignales in jedem Fall unterhalb von 3 V bleibt.

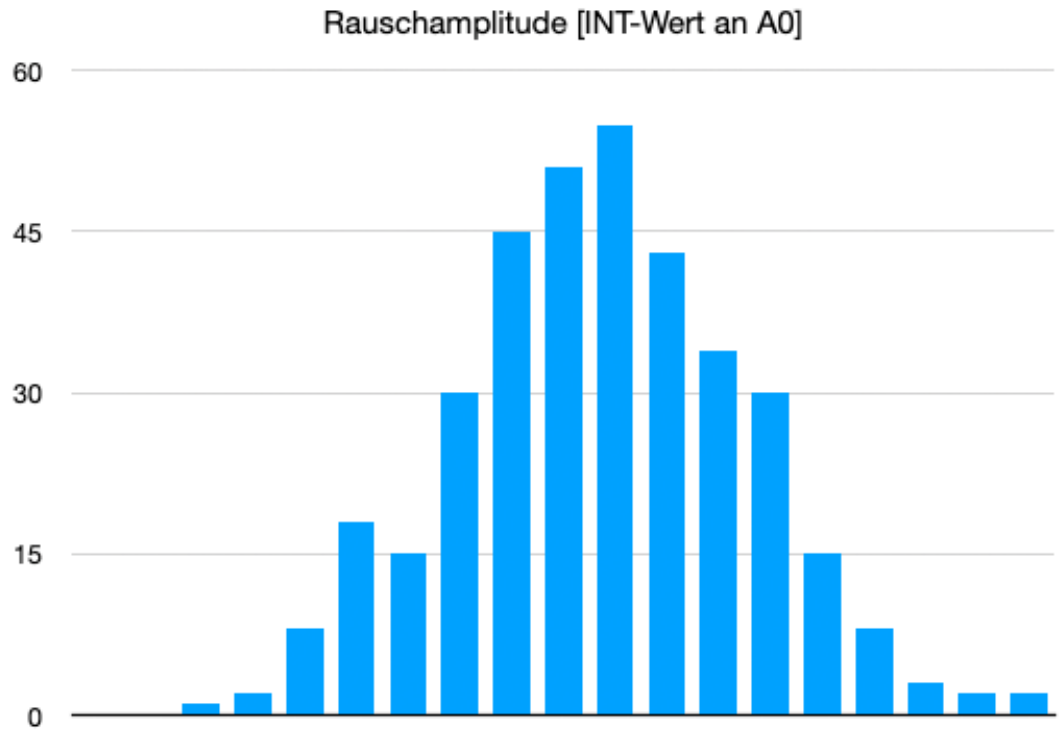
Noch ein Hinweis zum Aufbau der Schaltung. Der StepUp-Wandler MT3608 erzeugt zwar nur minimale Störungen, doch bei hoher Verstärkung und der empfindlichen Schaltung kann der Wandler sein 150 kHz-Signal einstreuen und damit einen systematischen Fehler auslösen. Der Zufall bleibt dann auf der Strecke. Wenn Sie den Wandler neben dem nano V3 platzieren, in etwa so, wie sie es weiter unten auf den Fotos des Musteraufbaus sehen, dann gibt es nach meiner Erfahrung keinerlei Probleme.

Das Herz des Projektes: Der Arduino-Sketch

Bei diesem Projekt ist es wichtig, dass die Abfrage des Rauschsignales durch eine Messung am Analogeingang A0 zu einem *zufälligen* Zeitpunkt erfolgt. Denn tatsächlich sind die Amplituden des Rauschsignales ja das Ergebnis eines natürlichen Prozesses und Gauss-verteilt.

Das kann man wunderschön nachvollziehen, wenn man mit einem einfachen Arduino-Sketch eine loop programmiert, die in regelmässigen Abständen an A0 Werte abholt und in ein Array schreibt. Ich habe

das ausprobiert und komme sofort zur Andeutung einer klassischen Gaussglocke - nicht ganz perfekt - wie bei 200 Werten auch nicht anders zu erwarten:



Klar zu erkennen: Diese Werte sind ganz sicher nicht *gleich*verteilt. Opa Gauss lässt grüßen

Wir werden später z.B. 200 Zufallszahlen abrufen und in ein Array schreiben. Deswegen wird auch der Zeitpunkt des Abrufs - aber nur der *Zeitpunkt*! - durch eine Pseudo-Zufallszahl bestimmt. Tatsächlich funktioniert dieses Vorgehen, wie der gelungene Chiquadrat-Test danach ganz klar zeigt. Dass die Amplituden Gauss-verteilt sind, sagt nicht, dass die Zufallswerte auch Gauss-verteilt sein müssen. Dazu ist die Wahl des Zeitpunkts der Abfrage wichtig und auch, wie ich lernen durfte, eine Einschränkung der Maximalamplitude, die man zulässt. Sie erkennen das Vorgehen in der Funktion „Werterfassung".

Nun zum Inhalt des Sketch. Es gibt wohl wenig Sinn, auf jedes Detail ein zu gehen. Ein wenig Erfahrung in Arduino C++ ist schon nötig, um den Sketch nach zu vollziehen. Um Übersichtlichkeit habe ich mich bemüht. Zuviel Erklärung bringt auch nicht viel - also in groben Zügen:

Wir nutzen hier den Command Prozessor **<MD_cmdProcessor.h>** den ich wärmstens empfehlen kann. Damit wird der Sketch klar strukturiert und gut lesbar. Man gibt eine Anweisung ein. Der entsprechende Teil des Sketches bzw. eine bestimmte Funktion läuft ab. Fertig - übersichtlich !

Die „handler" - also die jeweiligen Programmteile, die ausgeführt werden sollen, stelle ich immer direkt hinter die erste systemeigene Funktion: void handlerHelp(char* param); noch vor void setup() und alle Systemfunktionen. Setup wird somit phantastisch kurz (Zeile 130ff), ebenso die eigentliche Haupt-Programmloop, die zusammenschrumpft auf:

```
void loop() {
  CP.run();
}
```

Weniger geht nicht.

Schauen wir mal, welche Anweisungen der Command Prozessor bietet:

```
const MD_cmdProcessor::cmdItem_t PROGMEM cmdTable[] =
{
  { "?",  handlerHelp,  "",    "Help", 0 },
  { "h",  handlerHelp,  "",    "Help", 0 },
  { "zz", handlerZZ,    "",    "Eine Zufallszahl erzeugen", 0 },
  { "zu", handlerZU,    "",    "Zufallswerte erzeugen", 0 },
  { "zi", handlerZI,    "",    "Zufallswerte erzeugen und prüfen", 0 },
  { "aw", handlerAW,    "a",   "Anzahl Werte eingeben z.B.: aw 200", 0 },
  { "be", handlerBE,    "",    "Anzahl der Bereiche festlegen, z.B.: be 20",
0 },
  { "ta", handlerTA,    "",    "Ergebnis als Tabelle ausgeben", 1 },
  { "cs", handlerCS,    "",    "Ergebnis als CSV ausgeben", 1 },
  { "st", handlerST,    "",    "Statistische Ergebnisanalyse", 1 },
  { "sp", handlerSP,    "",    "Statistische Ergebnisanalyse als Plot", 1 },
  { "ch", handlerCH,    "",    "Signifikanzanalyse mit Chi-Quadrat Test", 1 },
  { "ab", handlerAB,    "",    "Abgleich der Steuerspannung", 2 },
};
```

Anweisungen werden im seriellen Monitor eingegeben. Achten Sie bitte unbedingt darauf, dass als Zeilenende nur „Linefeed“ eingestellt ist (Neue Zeile). Ansonsten kommt eine Fehlermeldung.

zz	Eine Zufallszahl wird erzeugt
zu	Ein Array wird mit Zufallszahlen gefüllt. Default sind 200 Werte. Dauert in etwa 10 - 20 Sekunden.
zi	wie „zu“, nur intelligent, deshalb zi. Der Chiquadrat-Test prüft und löst ggfls. neue Durchläufe aus.
aw	wie viele Zufallswerte sollen generiert werden? Eingabe z.B. „**aw 100“.** Zwischenraum nicht vergessen
be	Anzahl der „Boxen“ um die statistische Ergebnisanalyse durch zu führen (mehr im Text)
ta	Werte als Tabelle auf dem Bildschirm ausgeben
cs	Ergebnis als CSV-Datei über serielle Schnittstelle ausgeben
st	Statistische Ergebnisanalyse durchführen, z.B. in 20 „Boxen“ | Zeigt, wie viele Werte in jeder Box sind
sp	wie st, aber als Plot über den seriellen Plotter der Arduino IDE
ch	Chiquadrat-Test auf 1% Signifikanzniveau
ab	Die Steuerspannung (siehe Erklärung im Text zum Analogteil) anzeigen

Das klingt doch sicher schon mal ganz gut?

Statistische Ergebnisanalyse

Wenn man einen Würfel mit 6 Flächen hat, dann möchte man bei z.B. 1.000 Würfen gerne haben, dass die Zahlen 1 … 6 gleich häufig auftreten … jedenfalls so in etwa.

Wenn man die Zahlen z.B. von 1 … 400 hat, dann ist es bequemer, sich 10 oder 20 Boxen zu bauen. Die Zahlen 1 - 20 kommen in die erste Box, 21 - 40 in die zweite Box usw. Wenn die Boxen hinterher einigermaßen gleich gefüllt sind, dann sieht das nach Gleichverteilung aus.

Chiquadrat-Test

Im Studium haben uns die Mathematiker mit dem Chiquadrat-Test traktiert. Jetzt lohnt es sich, aufgepasst zu haben, ansonsten bitte nachlesen. Die Voreinstellungen des eingebauten Tests sind auf ein Signifikanzniveau von 1% festgelegt. Das kann man jederzeit ändern.

Der Test liefert eine Aussage der Art: "Die Zufallswerte erfüllen die Nullhypothese einer Gleichverteilung.“ Oder eben nicht. Dann startet in der Betriebsart „zi“ der Arduino nano V3 einen neuen Durchlauf und versucht es einfach noch

einmal. Was nach meiner Erfahrung die absolute Ausnahme ist. Wenn es doch mal nötig ist, kommt er auf 2 oder 3 Durchläufe maximal. Wenn tatsächlich nach 10 Läufen noch keine signifikanten Werte vorhanden sind, dass sollte der Analogabgleich noch einmal unter die Lupe genommen werden. Ein sauberer Durchlauf mit allen Funktionen sieht in der praktischen Anwendung so aus:

```
200 Zufallswerte werden erzeugt
Prozess abgeschlossen.

Der größte Wert im Array ist: 397

Verteilung der Werte in den Bereichen:
Bereich 0 - 38: 14
Bereich 39 - 77: 24
Bereich 78 - 116: 17
Bereich 117 - 155: 11
Bereich 156 - 194: 14
Bereich 195 - 233: 27
Bereich 234 - 272: 28
Bereich 273 - 311: 20
Bereich 312 - 350: 22
Bereich 351 - 397: 23

Deine Zufallszahlen:
378     270     255     348     50      288     249     312     48      284
119     26      242     226     94      210     58      300     332     53
356     170     239     213     355     145     32      217     178     374
222     177     161     280     30      19      264     35      293     242
275     41      370     234     246     243     156     118     218     278
206     34      172     369     68      221     213     246     259     228
112     45      39      241     397     292     172     185     217     2
286     93      49      156     236     112     353     347     221     287
118     40      388     328     114     317     264     115     119     187
382     187     202     170     286     311     42      38      391     396
228     55      219     251     392     57      366     320     247     342
232     105     49      344     322     79      339     340     44      58
239     90      118     79      236     198     96      49      305     203
355     112     156     126     283     204     209     306     109     260
328     90      343     378     271     268     212     315     71      53
135     48      126     368     327     40      298     321     344     371
174     30      251     195     300     36      383     380     268     222
122     234     131     195     305     62      86      248     110     378
82      212     288     30      331     200     347     369     313     264
44      314     39      25      304     26      237     220     379     38

Der größte Wert im Array ist: 397

Chi-Quadrat-Test mit Signifikanzniveau α = 0,01 [1%]

Chi-Quadrat-Wert: 15.20   |  Kritischer Wert = 21,67
Die Zufallswerte erfüllen die Nullhypothese einer Gleichverteilung.
```

Sie sehen nach dem Start das Ergebnis der statistischen Ergebnisanalyse. Die Boxen zeigen minimal 11 und maximal 28 Zahlen. Bei nur 200 Werten, verteilt auf 10 Boxen kann dennoch eine Gleichverteilung vorliegen, obwohl wir subjektiv daran zweifeln mögen. Es folgen danach die Zufallswerte (ta-Anweisung).

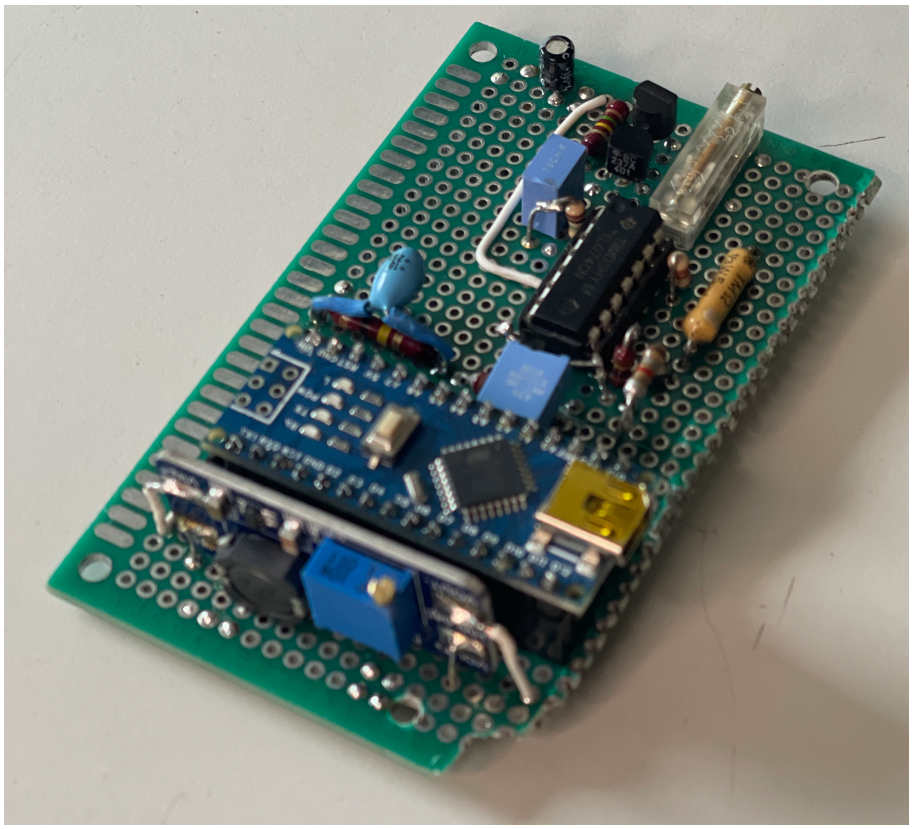
Ganz zu Schluss sehen Sie das Ergebnis des Chiquadrat-Tests, dessen kritischer Wert deutlich unterschritten wird. Inzwischen habe ich bei meinem Prototypen so weit die Analog-Einstellungen optimiert, dass meine Chiquadrat-Tests meist noch deutlich besser ausfallen. Das gelingt mir bislang nur empirisch - durch ausprobieren.

Zufällig sind die Ergebnisse aber immer und das ist ja auch so gewollt.

Aufbau der Hardware

Da wir eine Versorgungsspannung von +15 V für den Rauschgenerator brauchen, das Gerät aber gerne mit 5 V über den USB-Anschluss des nano V3 betreiben wollen, wurde ein StepUp-Wandler MT3608 eingesetzt. Er lässt sich sehr schön einstellen und liefert zuverlässig und stabil die gewünschte Ausgangsspannung.

Damit er seine 150 kHz nicht in die Schaltung einstreut, wurden verschiedene Anordnungen der Baugruppen getestet. Als problemlos erwies sich der Platz des Wandlers direkt neben dem Arduino nano V3:



Bei Rückfragen und Anmerkungen aller Art wenden Sie sich bitte an mich: dl1mkp@darc.de

>>> In Kürze stelle ich die Ergänzung vor, mit der dieses Projekt als elektronischer Würfel mit einer einstellbaren Anzahl Würfelflächen (6 … X) genutzt werden kann.

Viele Grüße - viel Spaß beim Nachbau
Auf alle Anmerkungen, Verbesserungsvorschläge und Anregungen freue ich mich.

Euer
Michael Klein | DL1MKP