Der Maker-Geiger: Geigerzähler und LoRaWAN Radioaktivitäts-Messstation auf Basis der Arduino Maker-Familie

Bernd Laquai, 27.12.2018

Manche hardware-orientierte Leute sagen ja spöttisch, dass der Unterschied zwischen einem echten Hardware-Elektronik-Experten und einem „Maker“ darin besteht, dass der „Maker“ lediglich fertige Boards zusammensteckt und die Eigen-Kreativität nur darin besteht, die Software dafür zu schreiben ohne die Hardware völlig zu verstehen. Es mag ja etwas Wahres daran sein, aber es ist auch definitiv so, dass es heute kaum mehr möglich ist, mit eigenen Mitteln Hardware zu bauen, welche moderne Elektronik-Komponenten enthält, und diese dann (inklusive der Chips) bis auf den letzten Transistor zu verstehen. Den ATMega 328P Prozessor eines Original Arduino Uno gibt es zwar noch in einem DIL-Gehäuse, so dass man rein theoretisch auch eine Platine dafür selbst machen könnte. Aber schon die leistungsfähigeren IoT-Mikrocontrollern wie z.B. der Atmel SAM D21 mit 32-Bit ARM® Cortex®-M0+ CPU kommen entweder im TQFP, QFN oder WLCSP Gehäuse daher, das sich nicht mehr mit dem klassischen Lötkolben selbst auf eine Platine auflöten lässt. Auch was die übrigen Komponenten und deren Performance anbelangt, lässt sich so etwas nicht mehr auf vernünftige Weise mit einer eigenen Platinen-Bestückung erreichen. Von daher wird jeder vernünftige Hardware-Entwickler bei Verfügbarkeit einer bereits erprobten, fertigentwickelten Platine zunächst auf diese zurückgreifen, bevor er etwas selbst entwickeln und die teure Herstellung und Bestückung an eine darauf spezialisierte Firma outsourcen würde. Da dies in gleichem Umfang auch für viele anspruchsvolle „Shields“ und „Break-Out“-Boards gilt, macht der moderne „Maker“-Ansatz durchaus seinen Sinn.

Deswegen soll hier auch ein „Maker“-Ansatz für einen Geigerzähler, der mit Vertretern der Arduino-Maker Familie als Controller und einem PiGI-Breakout-Board sehr einfach aufgebaut werden kann, vorgestellt werden. Dieser Geigerzähler kann auch ganz einfach mit Hilfe des Arduino MKRWAN 1300 zu einer Messstation für Umweltradioaktivität ausgebaut werden, die ihre Daten drahtlos über LoRaWAN und das bisher noch kostenfreie „The Things Network“ ins Internet überträgt.

Basis für den Sensor für Radioaktivität ist ein Geiger-Müller-Zählrohr. Obwohl es schon etliche Sensor-Lösungen für Radioaktivität auf Halbleiter-Basis gibt, bleibt das Geiger-Müller Zählrohr immer noch unerreicht, wenn es darum geht, mit geringen Kosten Zählraten zu erreichen, die für die Erfassung der Umweltradioaktivität im Niedrigdosis-Bereich geeignet sind. Prinzipiell stellt die hier vorgestellte Messstation nur sehr wenig Anforderungen an den Typ des Zählrohrs. Sehr kostengünstig erhältlich und sehr weitverbreitet ist das russische Zählrohr vom Typ SBM-20, das bei normaler Hintergrundstrahlung in der Umwelt (ca. 0.1µSv/h Ortsdosisleistung) etwa eine Zählrate von 30 Pulse pro Minute erreicht. Wartet man auf 100 Pulse um aus der dafür benötigten Zeit die Pulsrate zu berechnen, dann sind im Mittel 3.3 Minuten Messzeit für einen Messwert erforderlich. Rein aus statistischen Gründen (Poisson-Statistik des radioaktiven Zerfalls) bedeutet das, dass die Zählrate bei 100 Pulsen um ca. 10 Pulse schwankt (Streuung). Das heißt die Messunsicherheit rein auf Grund der unvermeidlichen Zufälligkeit der Pulse beträgt 10%. Ordnet man die Zählrate von 30 cpm der Ortsdosisleistung von 0.1µSv/h zu, dann ergibt sich daraus die Messunsicherheit von ca. 0.01µSv/h, was für die reine Überwachung der Umweltradioaktivität jedoch völlig ausreichend ist.

Natürlich hat so ein kostengünstiges Zählrohr noch weitere Quellen der Messunsicherheit. Eine sehr wesentliche ist beispielsweise die Energieabhängigkeit der Zählrate. Die Zählrate von etwa 30cpm pro 0.1µSv/h eines SBM20 Zählrohrs bezieht sich im Wesentlichen auf die Gamma-Strahlung von natürlichem Uran. Natürliches Uran, wie es in vielen Böden und im Gestein der Erdoberfläche vorkommt, enthält aber einen bunten Blumenstrauß von Radionukliden aus der Zerfallskette des U-

238 und des Thorium-232, die teilweise ebenfalls Gammastrahlung abgeben und mit ihrer spezifischen Energie der jeweiligen Gamma-Strahlung vor allem im Bereich zwischen 100 und 300keV zu dieser Zählrate beitragen. Vergleicht man das aber mit der Zählrate aus der Gammastrahlung des Cäsium-137 (Radiocäsium), das oft bei kerntechnischen Unfällen freigesetzt wird, dann stellt man fest, dass diese dann bei gleicher, mit Referenzmessgeräten festgestellten Ortsdosisleistung doch erheblich anders aussehen kann. Das liegt daran, dass im natürlichen Uran ein großer Teil der Gammastrahlung bei Energien entsteht, wo das Zählrohr sehr empfindlich ist und viele Pulse pro Minute liefert, im Vergleich zur Gammastrahlung bei 662keV des Radiocäsiums, wo ein SBM-20 Zählrohr doch schon deutlich weniger empfindlich ist. Das heißt aber, immer dann, wenn die Gammaenergie eines Strahlers arg vom natürlichen Uran abweicht, müsste man mit einer anderen Empfindlichkeit des Zählrohrs rechnen, was dann natürlich voraussetzt, dass man auch das entsprechende Radionuklid kennt, was eben oft nicht der Fall ist.

Zur Messunsicherheit kommt noch der Einfluss der Betastrahlung hinzu. Will man gegen jede Art von radioaktiver Strahlung maximal empfindlich sein, die für eine äußere Exposition relevant ist, betreibt man das Zählrohr typischerweise ohne wirksame Schirmung vor Betastrahlung oder bei einer Messstation mit einer minimal möglichen Schirmung z.B. der eines wasserdichten Gehäuses für den Außenraum (dünnwandiges Kunststoffgehäuse). Dann sieht man natürlich auch die Betastrahlung aus einer nassen Deposition (Regen), gegen die ein SBM-20 Rohr ebenfalls recht empfindlich ist. Gerade beim radioaktiven Cs137, welches bei kerntechnischen Unfällen als aerosolgebundene Radioaktivität in die Luft freigesetzt wird, hat man auch eine starke Betaaktivität, die auch gänzlich anders verteilt ist, als die des U nat. Von daher kann man sagen, kann so eine einfache Messstation die Radioaktivität in der Umwelt zwar sehr empfindlich detektieren, sie kann aber im Sinne einer Gamma-Ortsdosisleistung quantitativ nicht sehr korrekt messen, wen man das beispielsweise mit den ODL Messstationen des BfS (siehe odlinfo.bfs.de) vergleicht. Dazu wären dann ganz spezielle, aufwändige Maßnahmen zur Energiekompensation, Schirmung und eine umfassende Kalibrierung nötig. So viel also vorab zur erreichbaren Genauigkeit.

Ein Geiger-Müller Zählrohr braucht nun zum Betrieb eine Hochspannung von 350-500V (SBM-20 ca. 450V). In diesem Spannungsbereich ist die Zählrate nur wenig vom Wert der Hochspannung abhängig. Der Strombedarf eines Zählrohrs ist bei Zählraten, wie sie in der natürlichen Umwelt vorkommen extrem gering und liegt beim SBM-20 unter 1µA. Ein größeres Zählrohr mit größerer Oberfläche und entsprechend hoher Zählrate hat natürlich einen höheren Strombedarf, der aber immer noch so gering ist, dass man als Hochspannungsquelle lediglich eine Kapazität von etwa 10nF aus einem Spannungskonverter mit niedrigem Leistungsbedarf aufladen muss. Damit ist dann auch die Gefährlichkeit bei Berühren minimal, da die Spannung sofort zusammenbrechen würde, falls man die Kapazität berührt. Die Tatsache, dass für die Hochspannungserzeugung nur eine sehr geringe Leistung notwendig ist, passt auch sehr gut zur Anwendung einer IoT-Outdoor-Messstation. Die modernen „Maker“-Mikrocontroller sind nämlich ebenfalls auf kleine Leistungsaufnahme getrimmt, daher ergibt sich so auch die Option der Versorgung mit einer kleinen netzunabhängigen Photovoltaik-Anlage.

Für die Bereitstellung der Hochspannung in einem solchen „Maker“-System eignet sich ganz besonders das im Apollo-NG Projekt entwickelte PiGI-Board. Es wurde ursprünglich als Aufsteckboard für den Raspberry Pi entwickelt, ist aber als Break-Out Modul genauso für die Arduino-Maker-Familie geeignet.

Das PiGI-Modul ist mit ca. 5mA Stromaufnahme für die 450V Betriebsspannung des SBM-20 Zählrohrs sehr sparsam und kann daher auch vom Mikrocontroller-Board her versorgt werden (5V Anschluss). Neben der Bereitstellung der Hochspannung, die in einem weiten Bereich an einem Poti eingestellt werden kann, enthält es auch den Zählimpulsverstärker, der die analogen Zählimpulse in digitale wandelt, so dass ein Mikrocontroller z.B. mit einem Interrupt darauf reagieren kann.

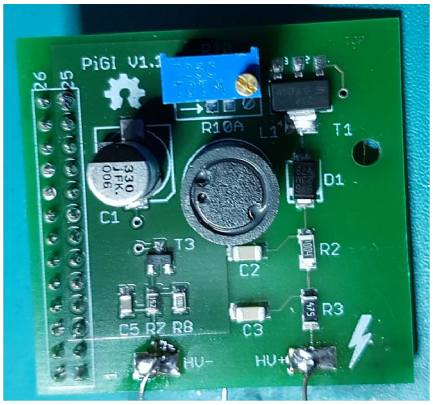

Abb. 1: Das PiGI- Geiger-Müller Zähler-Modul

Dieses Board wurde recht erfolgreich von der Regionalgruppe Aachen des „Forum InformatikerInnen für Frieden und gesellschaftliche Verantwortung e.V.“ (FiFF) eingesetzt um das TDRM Messnetz (Tihange Doel Radiation Monitoring) aufzubauen. Über ein Mitglied dieser Gruppe kann das Board auch als fertiges Modul bezogen werden (eMail: PeterKa@TreeDev.eu).

Die Funktion des Moduls ist etwas trickreich und nicht gleich auf Anhieb zu verstehen. Kern der Schaltung ist ein Step-Up Konverter, bestehend aus dem Transistor T1, der Induktivität L1, der Diode D1 und der Kapazität C2.

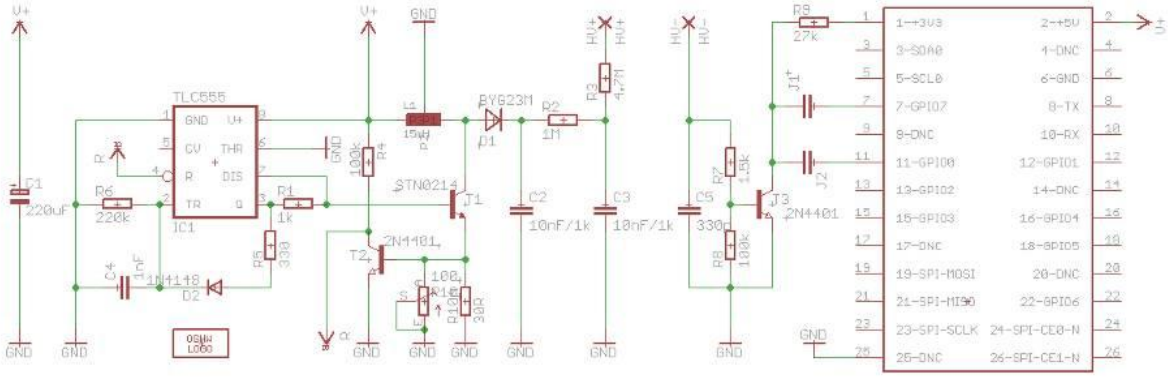

Abb. 2: Schaltplan des PiGI-Aufsteckmoduls

Der Hochspannungskapazität C2 ist noch ein weiteres RC-Siebglied, bestehend aus R2 und C3, nachgeschaltet um etwaige Spikes in der Hochspannung zu unterdrücken. R3 ist ein Anodenvorwiderstand mit 4.7MΩ, der den Strom durch das Zählrohr begrenzt und für viele gängige Zählrohre passend ist. Das Geiger-Müller Zählrohr wird an HV+ mit der Anode und an HV- mit der Kathode angeschlossen.

Bei diesem Modul wird das Zählimpulssignal kathodenseitig abgenommen. Es muss also beachtet werden, dass die Kathode nicht auf Masse liegt, sondern das schwache Zählimpulssignal trägt. Daher sollte das Zählrohr nicht in die Nähe von Quellen kommen, die Störstrahlung emittieren wie z.B. Handy, Motoren o.ä., und die gesamte Außenseite des Zählrohrs sollte auch mit keinem Leiter in Berührung kommen. Die Zählimpulse erzeugen am Widerstand R8 einen Spannungsabfall und werden vom

Transistor T3 verstärkt. Der Kollektorwiderstand von T3 muss auf der Spannung, mit dem das IO-Interface des Mikrocontrollers betrieben werden soll, liegen. Haben die IO‘s wie beim RasPi einen High-Pegel von 3.3V muss R8 auf 3.3V gelegt sein. Beim Arduino Uno oder Leonardo haben die IO‘s einen High Pegel von 5V, daher muss in diesem Fall R8 an 5V angeschlossen werden. Die eigentlichen Zählimpulse sind beim PiGI-Board „low-aktiv“ und werden über den Jumper J1 (das Symbol im Schaltplan dafür ist keine Kapazität!) und Pin 7 der Steckerleiste an den Mikrocontroller angekoppelt (der Jumper J2 an Pin 11 der Steckerleiste ist in der Regel auf dem Board nicht bestückt).

Getaktet wird der Hochspannungstransistor T1 vom CMOS Timer-Baustein TLC555. Allerdings ist in die Pulserzeugung auch der Emitterwiderstand R10 involviert (nur der Poti, der Festwiderstand ist nicht bestückt). An diesem entsteht ein Spannungsabfall, der zum Strom durch die Spule und durch den Hochspannungstransistor T1 proportional ist. Dieser steigt von Null aus gemäß $i(t)=-1/L*U_0*t$ linear an, wenn T1 durchgeschaltet ist. Sobald dieser Spannungsabfall an R10 groß genug ist, schaltet auch T2 durch, welcher den Reset auslöst, so dass der Ausgang am TLC555 von bisher High auf Low geht. Nach einer gewissen Zeit, springt der Timer dann wieder zurück auf High und der nächste Schaltzyklus beginnt. Da die Größe des Spannungsabfalls den Rücksetz-Zeitpunkt bestimmt, kann über den Poti die Dauer des Stromflusses durch die Spule sowie die Frequenz der Pulse und damit auch die Spannung, welche der Step-Up Konverter erzeugt, eingestellt werden. Wenn die Hochspannung eingestellt und nachgemessen werden soll, ist es angebracht wenigstens einen 100:1 Spannungsteiler (100MΩ:1MΩ) zu verwenden, da das Modul durch den herkömmlichen Innenwiderstand eines Multimeters (typ. 10MΩ) zu sehr belastet werden würde, so dass die Spannung dann zusammenbricht.

In der Originalanwendung, zusammen mit dem RasPi, passt der Steckverbinder direkt zu seinem Gegenstück auf dem Controller-Board und kann daher ganz einfach aufgesteckt werden. Setzt man das PiGI-Board für den Arduino ein, muss man den Steckverbinder per Kabel an die Stiftleiste eines Arduino anschließen. Wenn man einen älteren Arduino mit 5V IO‘s nutzen möchte, muss der Pin1 am PiGI-Board mit dem 5V Anschluss des Arduino-Boards verbunden werden, bei einem Arduino aus der Maker Familie mit 3.3V IO‘s muss der Pin 1 am PiGI-Board mit dem 3.3V Anschluss des Arduino-Board verbunden werden, damit der High-Pegel des Zählimpulsausgangs zum Controller passt. Der Zählimpulsausgang selbst liegt am PiGI-Board auf Pin Nr. 7 und muss beim Arduino-Board auf den passenden Interrupteingang für den Interrupt 0, wie er auch im gezeigten Arduino-Sketch verwendet wird, gelegt werden. In der unten gezeigten Abbildung ist zu beachten, dass der Stecker auf der Rückseite des Boards sitzt, das Board aber von der Top Seite dargestellt ist (Pin 1 oben links am Stecker).

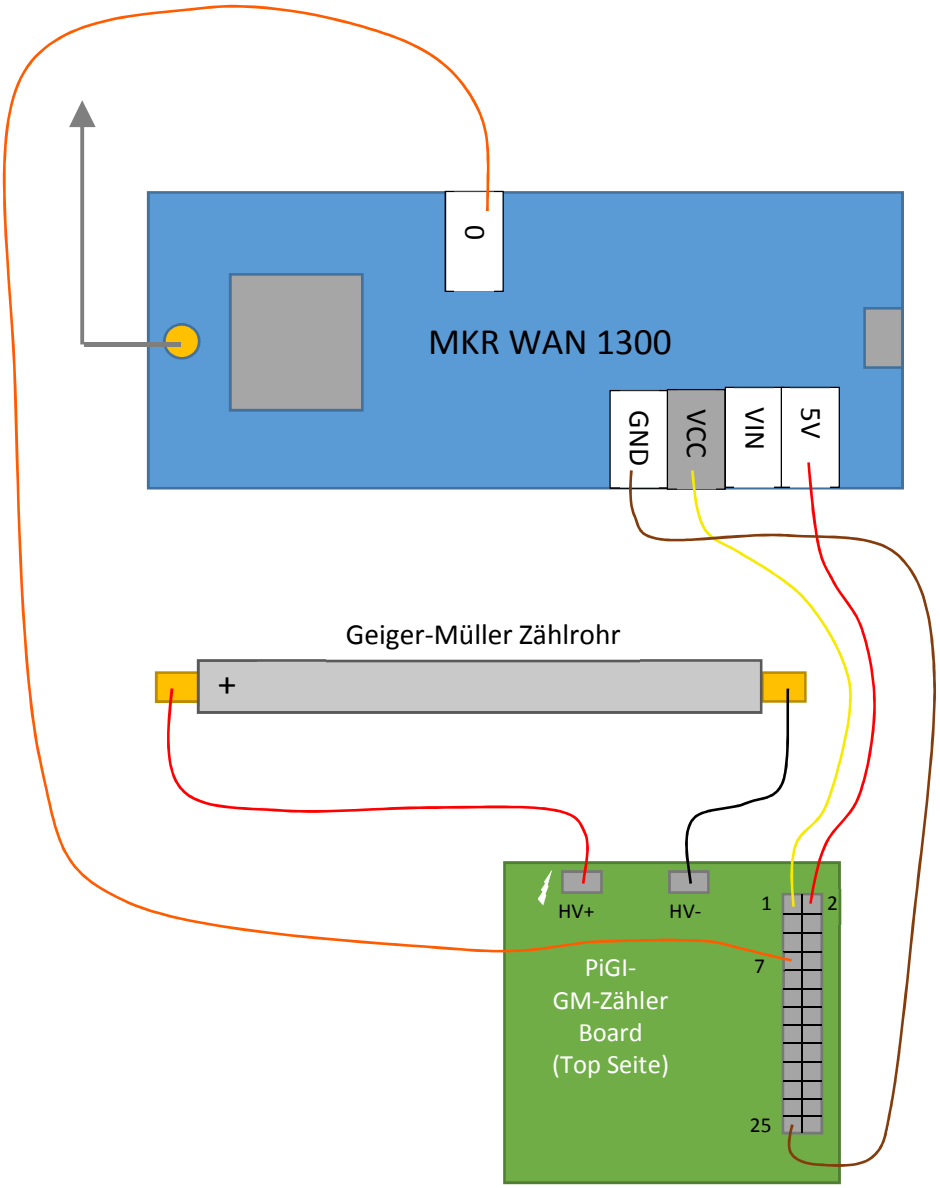

Abb. 3: Anschluss des PiGI Boards an einen Controller der Arduino Maker Familie (hier am Beispiel MKR WAN 1300) und an das Zählrohr. Sicht auf den Stecker von der Oberseite des PiGI-Boards!

Wenn man zunächst einmal lediglich die einfache Funktion eines Geigerzählers implementieren möchte, dann spielt es keine Rolle, welchen speziellen Controller man aus der Arduino Maker Familie benutzt. Man kann also z.B. den Arduino MKRZERO benutzen, welcher sonst eher ein Fokus auf Audio-Anwendungen hat. Das folgende Skript müsste auf jedem Ardino der Maker Familie laufen (getestet wurde es auf dem MKRZERO):

```
#define MAXCNT 100
#define CalFactor 1

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

void setup()
{
  Serial.begin(9600);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  digitalWrite(4, HIGH);
```

```
  digitalWrite(5, LOW);
  speaker=0;
  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*60.0*1000.0/(float)dt/CalFactor;
    Serial.println(round(rate));
    oldTime = millis();
    counter = 0;
    attachInterrupt(digitalPinToInterrupt(0), count, FALLING);
  }
}

void count()
{
  if (speaker) {
    digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
    speaker = 0;
  }
  else {
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    speaker = 1;
  }
  counter++;
}
```

Dieser Sketch gibt die aus 100 gezählten Impulsen berechnete Zählrate auf dem seriellen Monitor aus. Zusätzlich wurde noch am Pin 4 und Pin 5 ein Piezo-Lautsprecher angeschlossen um die Pulse mit dem berühmten Knacken hörbar zu machen.

Zur Berechnung der Zählrate ordnet das Skript dem Zählimpulseingang die fallende Flanke des Interrupt von Pin 0 zu und definiert, dass falls ein Zählimpuls eintrifft, die Interrupt-Handler-Routine count() ausgeführt wird. Diese zählt die Pulse in der Variablen counter hoch. Ein deutliches Knackgeräusch ohne Audioverstärker mit einem Arduino der Maker Familie zu erzeugen ist gar nicht so ganz einfach, da ein Pin, der als digitaler Output erklärt wurde, nur eine Spannung von 3.3V mit max. 7mA liefern kann. Daher kann an einen Output-Pin nicht einfach ein 8 Ohm Lautsprecher angeschlossen werden. Ein Miniatur-Piezo-Lautsprecher (mit Helmholtz-Resonator-Gehäuse) ist jedoch sehr hochohmig, braucht daher auch sehr wenig Strom und ist deshalb für diesen Zweck geeignet. Um eine maximal hohe Signal-Amplitude ohne weitere externe Bauteile zu erreichen, kann man anstatt einfach nur den Logikpegel an einem einzelnen digitalen Output-Pin umzuschalten, auch zwei Output Pins benutzen und die Logikpegel der Pins gegensinnig umschalten. Damit wird der Knacks dann deutlich lauter. Diese Knackerzeugung muss ebenfalls in der Interrupt-Handler-Routine ablaufen. Viel mehr an Befehlen sollte die Interrupt-Handler Routine jedoch nicht abarbeiten um die Totzeit des Zählers nicht unnötig zu erhöhen.

Ansonsten dreht sich der Sketch in der Funktion loop() endlos. Wenn die maximale Zahl an Pulsen, die für die Berechnung der Zählrate verwendet wird (MAXCNT) erreicht ist, findet die eigentliche Rechnung statt. Dazu wird die vergangene Zeit mit millis() abgefragt und die Differenz zur letzten Zeitabfrage gebildet. Die Rate ergibt sich dann mit einer Division von MAXCNT durch die Zeitdifferenz unter Berücksichtigung der Zeiteinheit (millis liefert die Zeit in Millisekunden). Als Ausgabe entsteht so die Zählrate in Counts per Minute (cpm), die über den seriellen Monitor ausgegeben wird.

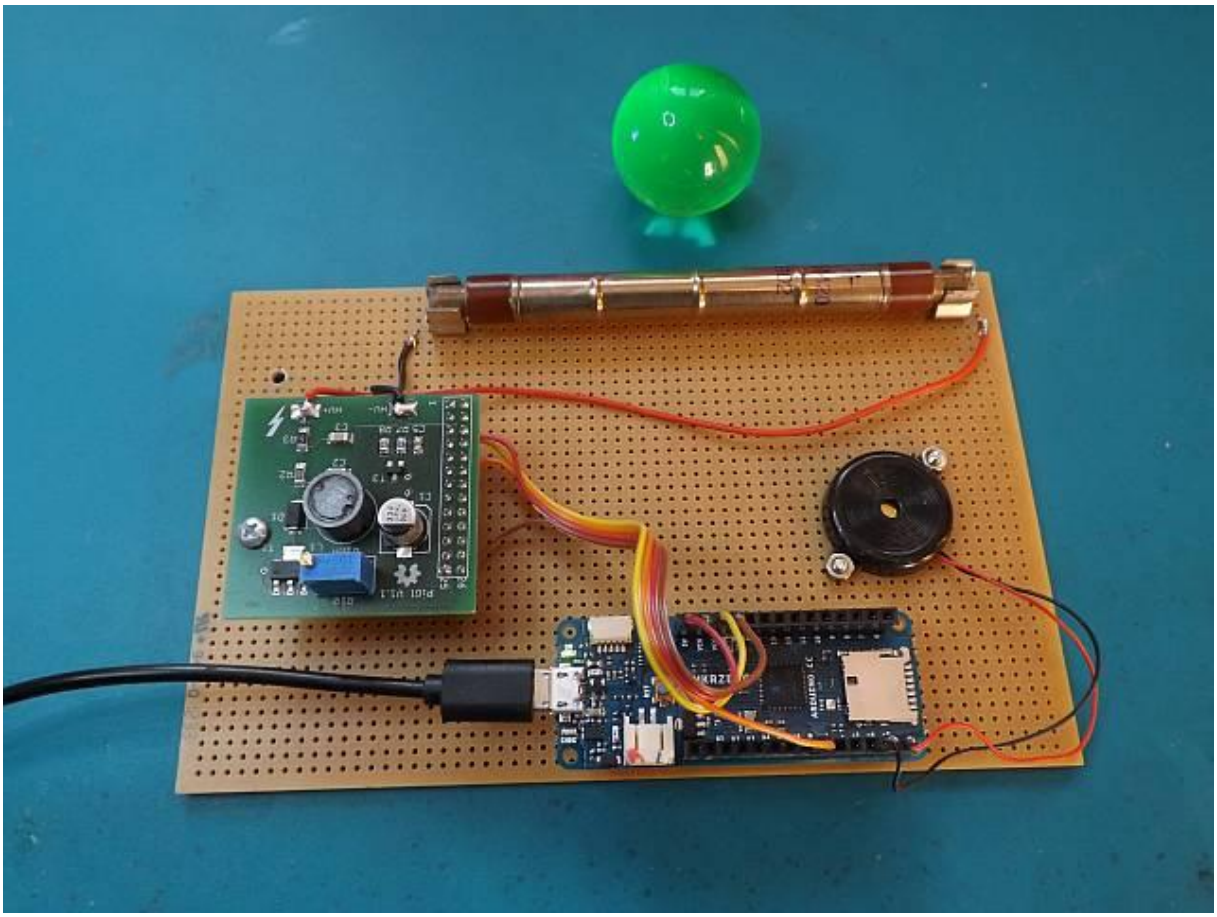

Abb. 4: Der MKRZERO in der Konfiguration als Geigerzähler mit Knacksignalisierung. Vor dem Zählrohr liegt eine Murmel aus fluoreszierendem Uranglas (mit UV-Licht angeleuchtet), welche eine gegenüber der Nullrate deutlich erhöhte Knackrate erzeugt.

Angesichts der Möglichkeiten, welche die IoT-Controller der Arduino Maker Serie bieten, ist es natürlich naheliegend speziell die Kommunikationsmöglichkeiten dieser Controller zu nutzen um die Messdaten ins Internet zu übertragen und die Geigerzähler Anwendung zu einem Messknoten für die Messung der Umweltradioaktivität im Außenbereich, auch außerhalb der heimischen WLAN-Reichweite, zu erweitern. Wenn man dazu den MKRWAN 1300 nutzt, dann hat man derzeit noch eine kostenfreie Möglichkeit die Daten über ein drahtloses LP-WAN bis max. 30km zu übertragen. Der MKRWAN 1300 enthält dazu ein LoRaWAN Funkmodul und es wird eine Bibliothek bereitgestellt, mit der die Daten direkt über LoRaWAN in das freie IoT-Netz des Providers „The Things Network“ (TTN) übertragen werden können. Wie man dazu vorgehen muss, ist in /1/ ausführlich beschrieben. Voraussetzung dazu ist aber, dass in dem Gebiet, in dem die Messstation eigesetzt werden soll, auch eine LoRaWAN Funknetzabdeckung vorhanden ist. Dies kann sehr einfach über die Webseite ttnmapper.org überprüft werden (siehe auch /2/).

Für die Implementierung einer einfachen Messstation kann nun folgendes Skript verwendet werden, welches das oben genannte reine Geigerzähler Skript um die LoRaWAN Funkkommunikation erweitert.

```
#include <MKRWAN.h>
#include <CayenneLPP.h>
#define MAXCNT 100
#define CalFactor 1

LoRaModem modem;

//#include "arduino_secrets.h"
// Please enter your sensitive data in the Secret tab or arduino_secrets.h
String appEui = "70B3D57ED000C115";
String appKey = "E75E6938CCE6999BC6D69A16B244534D";
//String appKey = "FC1A8D5692A52362EA911BC9A514AC41";

CayenneLPP lpp(51);

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

void setup()
{
  Serial.begin(9600);

  pinMode(LED_BUILTIN, OUTPUT);        //sign of live
  for (int i=1; i<=10; i++) {
    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);                       // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);
    // wait for a second
    Serial.println(10-i);
  }

  // change this to your regional band (eg. US915, AS923, ...)
  if (!modem.begin(EU868)) {
    Serial.println("Failed to start module");
    while (1) {}
  };
  Serial.print("Your module version is: ");
  Serial.println(modem.version());
  Serial.print("Your device EUI is: ");
  Serial.println(modem.deviceEUI());

  int connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    Serial.println("Something went wrong; are you indoor? Move near a window and retry");
    while (1) {}
  }

  // Set poll interval to 60 secs.
  modem.minPollInterval(60);
  // NOTE: independently by this setting the modem will
  // not allow to send more than one message every 2 minutes,
  // this is enforced by firmware and can not be changed.

  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*60.0*1000.0/(float)dt/CalFactor;
    Serial.println(round(rate));
    lpp.reset();
    lpp.addLuminosity(1, round(rate));
    modem.beginPacket();
    modem.write(lpp.getBuffer(), lpp.getSize());
```

```
    err = modem.endPacket(true);
    if (err > 0) {
      Serial.println("Message sent correctly!");
    } else {
      Serial.println("Error sending message :(");
      Serial.println("(you may send a limited amount of messages per minute, depending on the
signal strength");
      Serial.println("it may vary from 1 message every couple of seconds to 1 message every
minute)");
    }
    delay(1000);
    if (!modem.available()) {
      Serial.println("No downlink message received at this time.");
    }
    else {
      String rcv;
      rcv.reserve(64);
      while (modem.available()) {
        rcv += (char)modem.read();
      }
      Serial.print("Received: " + rcv + " - ");
      for (unsigned int i = 0; i < rcv.length(); i++) {
        Serial.print(rcv[i] >> 4, HEX);
        Serial.print(rcv[i] & 0xF, HEX);
        Serial.print(" ");
      }
      Serial.println();
    }
    Serial.println("waiting 60 seconds");
    delay(60000);

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

void count()
{
  counter++;
}
```

Dieser Sketch ist nun mit Aufrufen der MKWAN-Bibliothek und der Cayenne-Bibliothek von mydevices.com zur späteren Visualisierung der Daten (z.B. mit der Cayenne App) erweitert. Sobald die Zählrate wie oben beschrieben berechnet und auf dem Serial Monitor ausgegeben ist, wird zunächst die Formatierung der Zählrate für die webbasierte Visualisierungsplattform Cayenne vorgenommen, an die der TTN-Gateway die Daten weiterreicht. Dazu dient die Cayenne lpp-Methode. Da lpp bisher noch keine Zählraten in cpm und auch keine Geiger-Müller Zählrohre kennt, wird hierzu die Helligkeit (Luminosity) als Größe in Cayenne verwendet. Später, bei der Einrichtung des Cayenne Dashboards kann „Helligkeit“ wieder mit „Count Rate in CPM“ überschrieben werden. Die Funktion modem.write() schickt die mit lpp als Helligkeit formatierte Zählrate mit den Parametern lpp.getBuffer() und lpp.getSize() schließlich über das LoRaWAN an das TTN Backend, das die Daten an mydevices.com weiterreicht, wo die Daten im Cayenne Dashboard des Anwenders entsprechend visualisiert werden. Dazu muss auch die entsprechende Cayenne Integration im TTN User Bereich unter der verwendeten Application aufgesetzt sein (für die detaillierte Konfiguration siehe /1/).

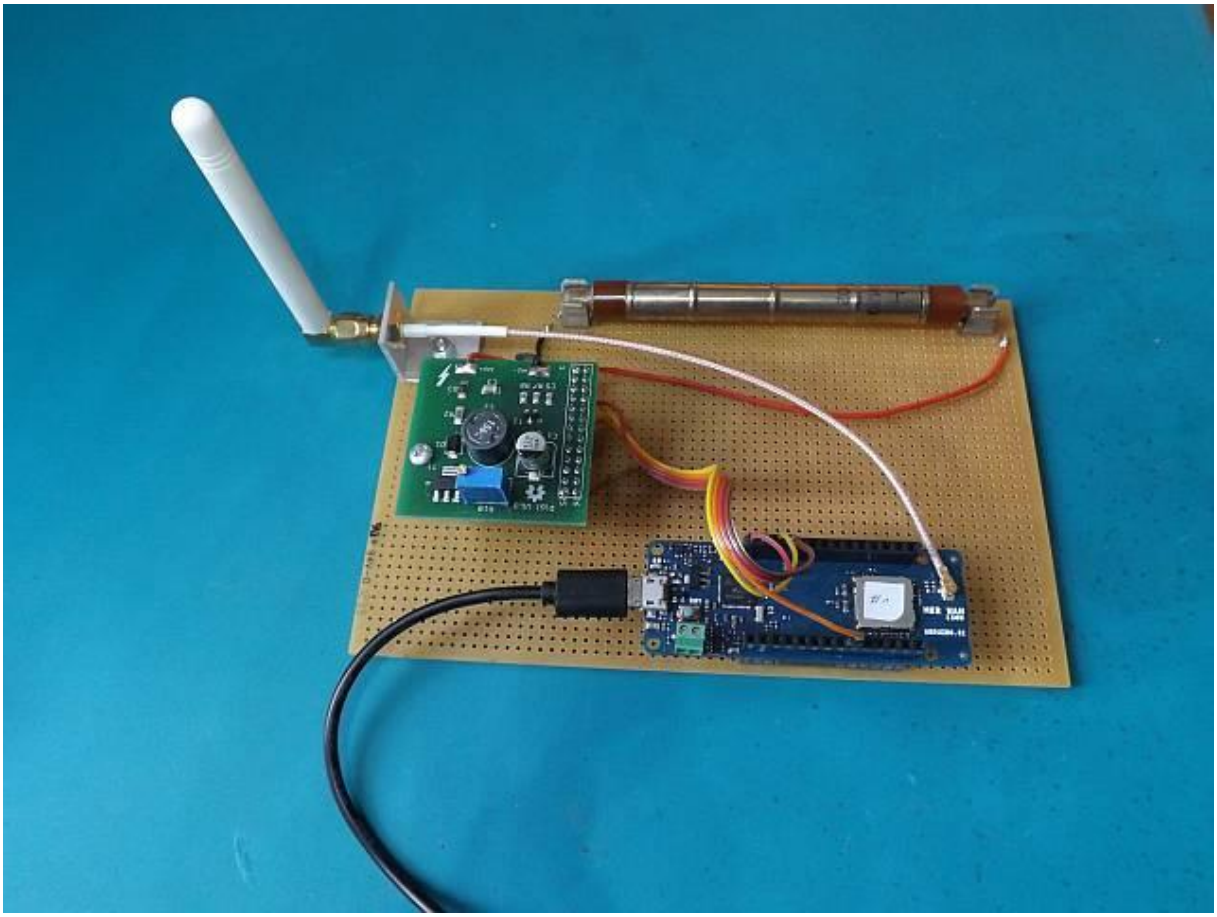

Abb. 5: Der Arduino Maker-Geiger als Applikation zur Erfassung der Umweltradioaktivität mit dem Arduino LoRaWAN Controller MKRWAN 1300

Um die Funktion der Messstation für Dosisleistungen, wie sie in der Umwelt vorkommen können, zu prüfen, ohne sich irgendwelchen Gefahren im Sinne des Strahlenschutzes auszusetzen, eignet sich unter anderem kaliumhaltige Pottasche (Kaliumcarbonat), die es als Backtriebmittel vor allem vor Weihnachten sehr günstig zu kaufen gibt. Zudem ist sie auch gleich recht praktisch als leicht handzuhabender Flächenstrahler verpackt und die Verpackung muss bei Verwendung nicht geöffnet werden. Das natürliche Kalium in der Pottasche enthält nämlich in geringen Mengen auch das radioaktive Kalium-40 Isotop, dessen sehr schwache und in dieser Dosierung völlig ungefährliche radioaktive Strahlung (Gamma- und Betastrahlung) auch mit einem SBM-20 Zählrohr noch sehr leicht nachweisbar ist.

Beim Einbau der Platine in ein dünnwandiges, wasserdichtes Gehäuse aus Kunststoff für den Einsatz im Außenbereich ist noch ein ganz wichtiger Aspekt zu erwähnen. In der Außenluft entstehen oft hohe Luftfeuchten und es kann beim Einbau in zu (luft-)dichten Gehäusen schnell zur Kondenswasser-Bildung kommen. Im Zusammenhang mit der Betriebsspannung von 450V würde das wiederum recht schnell zum Zusammenbrechen der Hochspannung führen. Das Gehäuse sollte daher zwar zum Schutz vor Regen so wasserdicht wie möglich sein, aber man sollte an der regengeschützten Unterseite (am tiefsten Punkt) ein sehr kleines Loch in das Gehäuse bohren (ca. 1mm). Das Loch sollte so klein sein, das keine Insekten eindringen können, aber dennoch ein Gasaustausch von innen nach außen möglich ist, so dass die Feuchte im Innern des Gehäuses nicht kondensiert, wenn das Gehäuse von außen ohne Luftaustausch abkühlt.

Aufgestellt sollte das Gehäuse etwa 1m über dem Boden (am besten Grasboden) werden, so dass man einer amtlich gemessenen Ortsdosisleistung auch möglichst nahekommt.

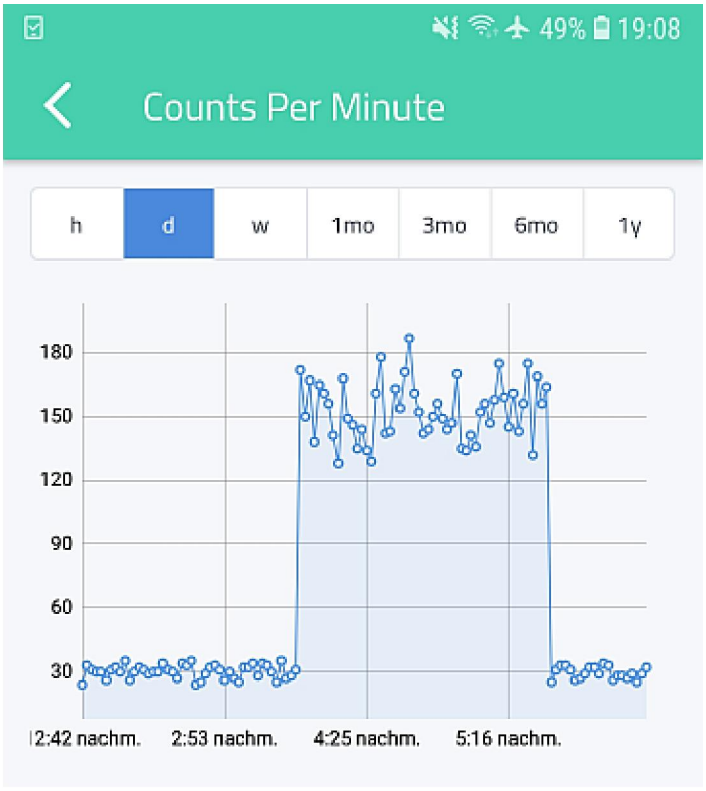

Abb. 6: Pottasche als Backtriebmittel eignet sich hervorragend als Prüfstrahler zur Funktionskontrolle. Die Cayenne App zeigt deutlich den Anstieg der Zählrate um den Faktor 5 gegenüber der Nullrate als Folge der Gamma- und Betastrahlung des Kalium-40 Isotops

Links und Literatur

/1/ Ein Umwelt-Radioaktivitäts-Messknoten für das LoRaWAN IOT-Netzwerk „The Things Network“

http://opengeiger.de/LoRaGeigerTTN.pdf

/2/ Ein einfaches Mapping Gerät für das LoRaWAN TTN-Netzwerk auf Basis eines Arduino MKR WAN 1300

www.opengeiger.de/LoRaWAN/MkrWan1300Mapper.pdf

/3/ TDRM Messnetz (Tihange Doel Radiation Monitoring)

https://tdrm.fiff.de/index.php

/4/ Apollo-NG PiGI-Projekt

https://apollo.open-resource.org/lab:pigi

/5/ Kaliumchlorid als Kochsalzersatz und als Prüfstrahler

https://de.wikipedia.org/wiki/Kaliumchlorid

/6/ Bezugsquelle für das fertige PiGI-Modul eMail: PeterKa@TreeDev.eu