„Tino“ – Ein Messgerät für radioaktive Strahlung als Arduino-Shield auf Basis eines Teviso Sensormoduls

Bernd Laquai, 11.4.2014

Es ist schon erstaunlich was man durch Miniaturisierung und Kapselung erreichen kann. Die schweizerische Firma Teviso stellt mit ihren miniaturisierten Sensormodulen für Röntgen-, Gamma- und Betastrahlung einen PIN-Dioden Sensor mit digitalem Zählimpulsausgang zur Verfügung, so dass man sich um den analogen Signalpfad wirklich keine Gedanken mehr machen muss. Ein einfaches Verbinden des Sensors mit einem Mikrocontroller und einem Display ist ausreichend um ein relativ genaues Messgerät für radioaktive Strahlung selbst zu bauen. Das Teviso Sensormodul hat gerade mal die Größe eines kleinen Domino-Steins, enthält die notwendige EMV-Schirmung und beschränkt sich auf drei Anschlusspins, die 5V Stromversorgung, den digitalen Zählausgang und einen Masse-Anschluss. Ein enormer Vorteil gegenüber Geiger-Müller Zählrohren ist die Tatsache, dass man zum Betrieb des keine Hochspannung benötigt. Damit wird man gegen Feuchte deutlich unempfindlicher, die bei Zählrohr-Messgeräten leicht dazu führen kann, dass hohe Spannungen auf die empfindlichen CMOS-Eingänge des Mikrocontrollers oder Zählimpulsverstärkers gelangen können und ihn zerstören. Gänzlich wasserdicht ist das mit einer Vergussmasse auf der Rückseite verschlossene Sensormodul allerdings trotzdem nicht, bei permanenter Installation im Freien ist also zusätzlich eine wirklich wasserdichte Umhüllung nötig.

Ein unschätzbarer Vorteil der Teviso Module ist aber, dass werksseitig ein Kalibrierfaktor für die sogenannte Umgebungs-Äquivalentdosis H*(10) mit 15% Genauigkeit angegeben wird (auf Wunsch auch genauer). Diese Dosisangabe (auch kurz Ortsdosis bzw. bei Bezug auf ein Zeitintervall auch Ortsdosisleistung genannt) ist ein Schätzwert für die Körperdosis, die eine Person erhielte, die sich an diesem Ort im Strahlungsfeld befindet. Gemessen wird diese Dosis in einem aufgeweiteten und ungerichteten Strahlungsfeld an einem ICRU-Phantom in 10cm Tiefe. Das ICRU-Phantom ist eine Kugel mit 30cm Durchmesser aus einem vom ICRU definierten Weichteilgewebeersatz, welcher die Dichte 1 g/cm3 hat und aus 76,2% Sauerstoff, 11,1% Kohlenstoff, 10,1% Wasserstoff und 2,6% Stickstoff besteht. Das Material hat dann eine dem menschlichen Weichteilgewebe vergleichbare Absorptions- und Streueigenschaft für ionisierende Strahlung.

Diese Angabe eines Kalibrierfaktors aber bedeutet, dass man im Eigenbau mit praktisch geringstem Aufwand ein werksseitig kalibriertes Messgerät herstellen kann, indem man sich einfach auf diese Angabe des Sensorherstellers bezieht. Das unterscheidet die Lösung dann doch deutlich von einem selbstgebauten Zählrohr-Gerät, das man wegen der fehlenden Kalibrierung eigentlich nur als Strahlungsdetektor bezeichnen kann und nicht als echtes und verlässliches Messgerät.

Nimmt man sich als Ziel vor, ein Messgerät auf einem Arduino-Shield (auf die Arduino Mikrocontroller Platine aufsteckbare Zusatzplatine) unterzubringen, wird schnell klar, dass man sich in der Funktionalität etwas beschränken muss. Als minimal nötige Komponenten ist außer dem Sensormodul noch ein numerisches Anzeigemodul erforderlich. Fast unumgänglich ist auch ein kleiner akustischer Signalgeber, sei es für die Signalisierung der vom Sensor registrierten Strahlungsquanten (Geigerzähler-Knacken) als auch als Möglichkeit das Überschreiten einer Alarmschwelle zu signalisieren. Allerdings sollte man diesen Signalgeber über einen Jumper abschaltbar machen, da dieses Geräusch manche Mitmenschen unnötig nervös machen kann und auch beim Betrieb über Nacht sich unangenehm auswirken kann. Außerdem sollte man darauf achten, dass das Shield auch auf andere darunterliegende Shields (z.B. das Ethernet-Shield) aufsteckbar ist ohne mechanische Konflikte oder Pin-Belegungskonflikte auszulösen, so dass diese in einer erweiterten Lösung mitbenutzt werden können.

Im Hinblick auf die Pin-Belegung des Shields ist es vorteilhaft, insbesondere die für die Anzeige benötigte Zahl an Pins gering zu halten. In dieser Hinsicht empfiehlt sich ein Anzeigemodul mit serieller Schnittstelle. Da entsprechende LCD Anzeigen von der Bauform her sehr groß sind, wurde hier ein 4-stelliges 7-Segment Modul mit seriellem Interface verwendet. In Anbetracht der Tasache, dass es für gewisse sehr gebräuchliche Module schon vorgefertigte Bibliotheken gibt, wurde für das Tino-Shield die Anzeige vom Typ LTM-8328PKR-04 der Firma LiteOn (erhältlich z.B. über Digikey) verwendet. Hierzu gibt es von Doë Goebish eine gut dokumentierte und einfach zu nutzende Bibliothek. Das erspart dann doch das Schreiben etlicher Codezeilen, im Gegensatz dazu ist eine vorhandene Bibliothek mit einigen Mausklicks installiert. Die Verwendung dieser Anzeige und des fertigen Sensormoduls macht die notwendige Verdrahtung auf dem Shield sehr einfach und übersichtlich. Allerdings ist die Anzeige nicht allzu leuchtstark. Den Einstellpoti für die Helligkeit kann man auch gut durch einen 10kΩ Festwiderstand ersetzen, so dass die Stromaufnahme nicht zu hoch wird. Wer auf gute Lesbarkeit bei hellem Licht besonderen Wert legt, kann natürlich auch eine etwas teurere aber dafür lichtstärkere Anzeige mit einer vergleichbaren seriellen Schnittstelle verwenden. Bei dem Tino Prototyp wurde das Display Modul noch zusätzlich in eine Buchsenleiste gesteckt um es austauschbar bzw. auch ganz entfernbar zu machen.

Als Sensormodul wurde das günstigste Modul von Teviso mit der Bezeichnung RD2007 benutzt. Spezifiziert ist es mit einem Kalibrierfaktor von 3.4cpm pro µSv/h. Es gibt auch noch ein Modul mit höherer Zählrate, welches aber auch teurer ist. Die Module sind auf der Webseite des Herstellers Online bestellbar. Das Modul ist für liegende Montage gedacht, was für ein Arduino-Shield etwas ungünstig ist, wenn man gleichzeitig auch ein Display auf der Oberseite haben möchte und dieses gut ablesen können will. Deswegen wurde das Sensormodul beim Tino der Breitseite entlang aufgestellt und über 3 gewinkelte Verbinder aus Silberdraht elektrisch verbunden und mechanisch fixiert. Damit kann man das Arduino-Board mit Shield auch auf eine zu prüfende Oberfläche stellen (entlang der Seitenkante) und trotzdem noch das Display gut ablesen. Die Stromversorgung des Strahlungssensors wurde auf der Vdd Seite über ein LC Siebglied geführt, da es laut Hersteller empfindlich gegen allzu kräftige Störungen auf der Stromversorgung ist. Diese Vorkehrung wurde getroffen, da die Qualität der 5V Versorgung auf der Stiftleiste stark von der primären Versorgung (evtl. Schaltnetzteil!) und anderen schaltenden Komponenten auf den Platinen abhängig ist. Die Induktivität sollte einen nennenswerten Serienwiderstand > 10 Ohm haben, damit der Schwingkreis, den sie mit dem 100uF Siebkondensator bildet auch sicher überkritisch bedämpft ist und keine eigene Instabilität durch Resonanz erzeugt. Die verwendete Spule hat einen Serienwiderstand von etwa 28Ohm. Ein so gebautes RLC Filter ist effektiver (steilerer Dämpfungsverlauf) als nur ein einfaches RC Filter.

Als akustischer Signalgeber wurde ein kleiner Piezo Schallgeber mit eingebauter Ansteuerung von Euklit (Typ AL-60SP05) ausgewählt. Er lässt sich mit 50ms kurzen 5V Pulsen gut ansteuern und gibt dabei einen sehr kurzen Beep ab, der viel leichter hörbar ist als ein reiner Click eines Miniaturlautsprechers. Für diese kurze Zeit benötigt der verwendete Signalgeber etwa 20mA. Dieser Signalgeber wurde zusammen mit einem Jumper, der ihn abschaltbar macht, auf dem noch verfügbaren Platz auf dem Shield untergebracht.

Abb. 1 zeigt einen Prototyp des Tino-Shields, aufgebaut auf einer einfachen Lochrasterplatine. Da die Arduino-Mikrocontroller Platine 4 Stiftreihen hat, wovon eine (die mit den höheren Digitalpins) um 50mil gegenüber dem 100mil (2.54mm) Raster als „Verdreh-Schutz“ versetzt ist, wurde diese Stiftleise einfach weggelassen und es wurden nur Digitalpins mit niedrigen Nummern < 8 verwendet. D.h. gegenüber dem Bibliotheksbeispiel, welches die Pins D8 und D9 für Daten und Clock der Anzeige verwendet, wurde beim Tino-Shield der Pin D6 und D7 verwendet. Der Signalgeber wurde an Pin D5 angeschlossen.

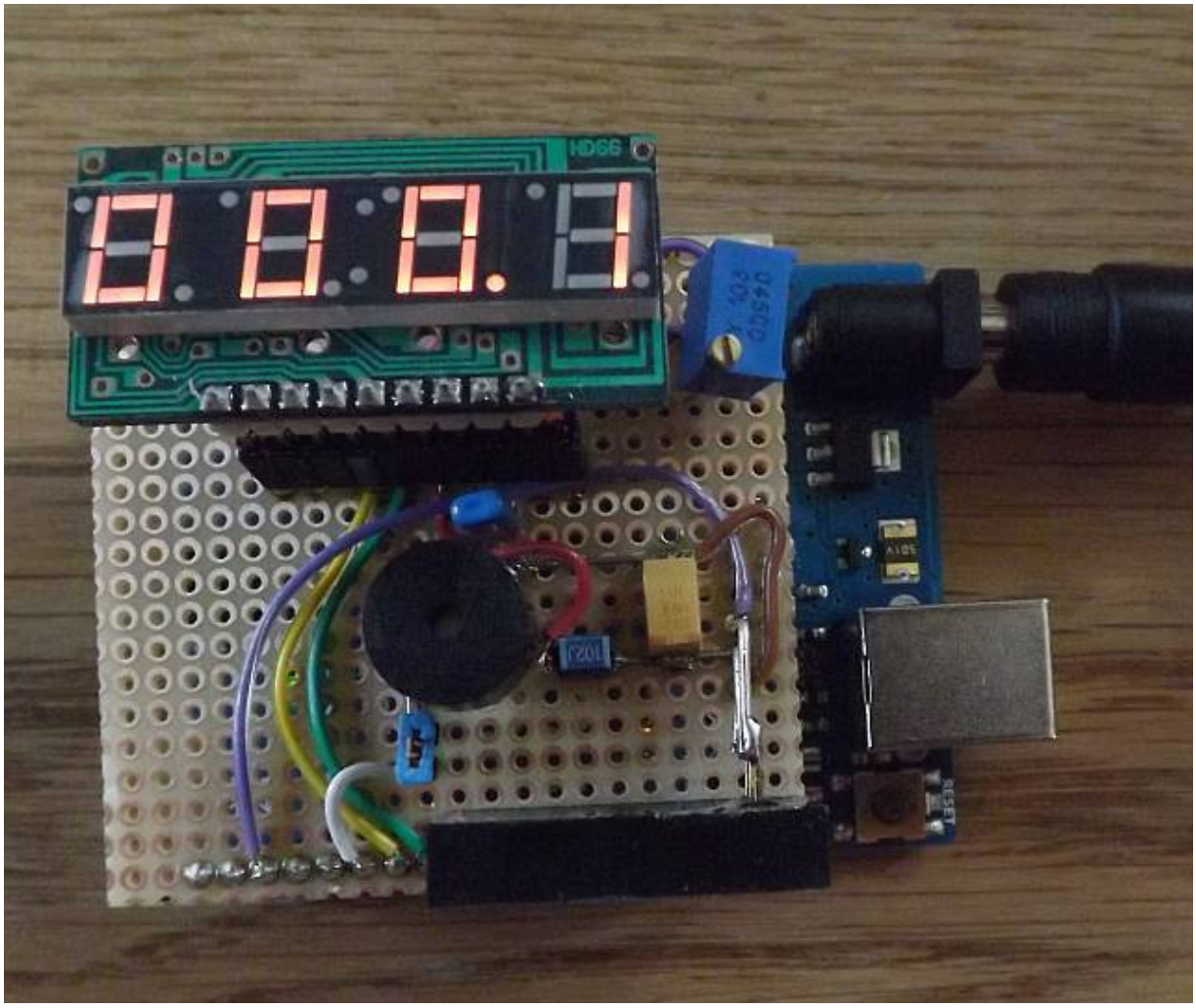

Abb. 1a: Das „Tino 1“ Prototypen-Shield auf dem Arduino UNO board, angezeigt wird gerade die Nullrate

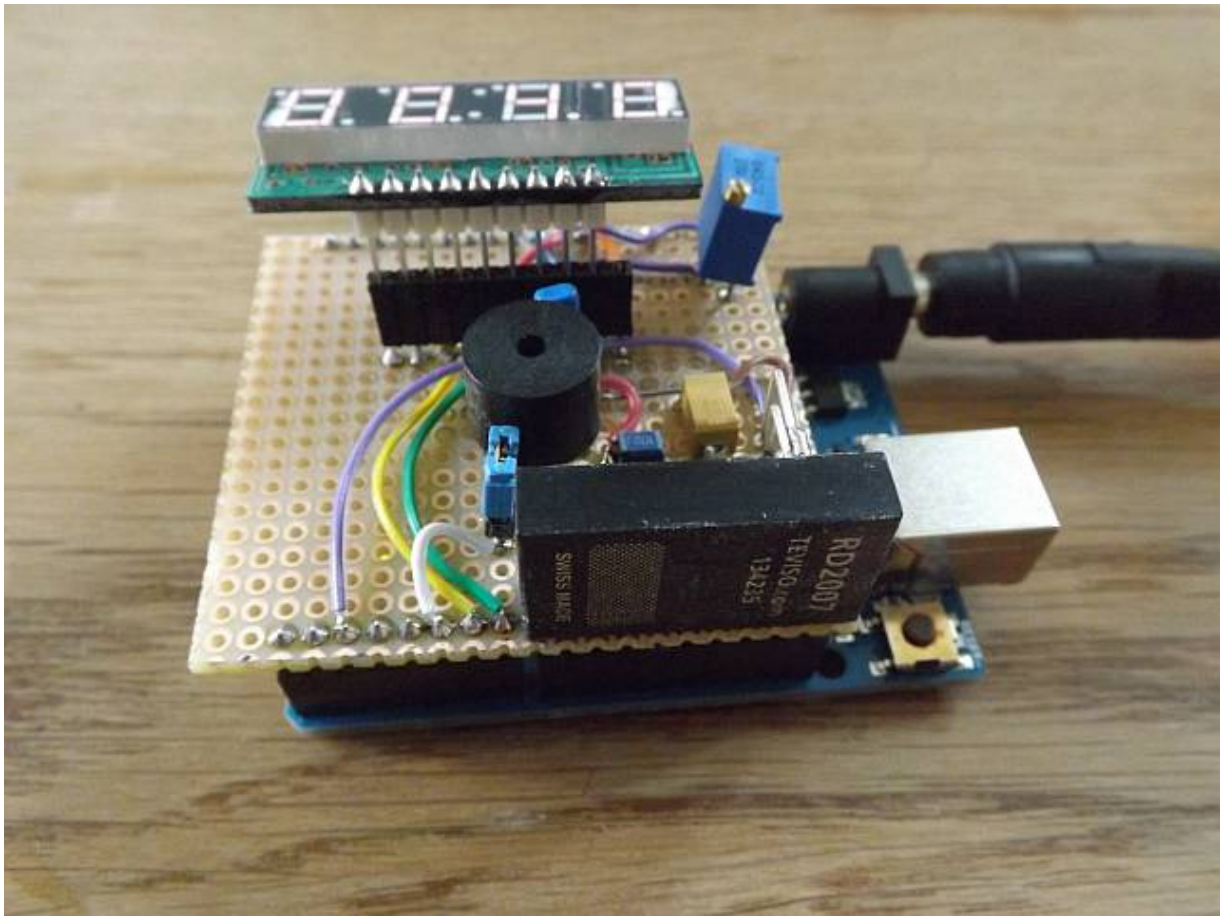

Abb. 1b: Das Tino 1 Prototypenshield von der Seite, vorne das RD2007 Sensormodul von Teviso

Als Zählprogramm wurde das bereits für den einfachen Modulanschluss entwickelte Programm übernommen und lediglich die Ausgabe durch den entsprechenden Bibliotheksaufruf für die 7-Segmentanzeige ersetzt. Um die Bibliothek nutzen zu können, muss die heruntergeladene zip-Datei entpackt werden, das entstandene Verzeichnis umbenannt und in das Standard-Bibliotheksverzeichnis der verwendeten Arduino Software verschoben werden (<Pfad zur Arduinosoftware>/libraries).

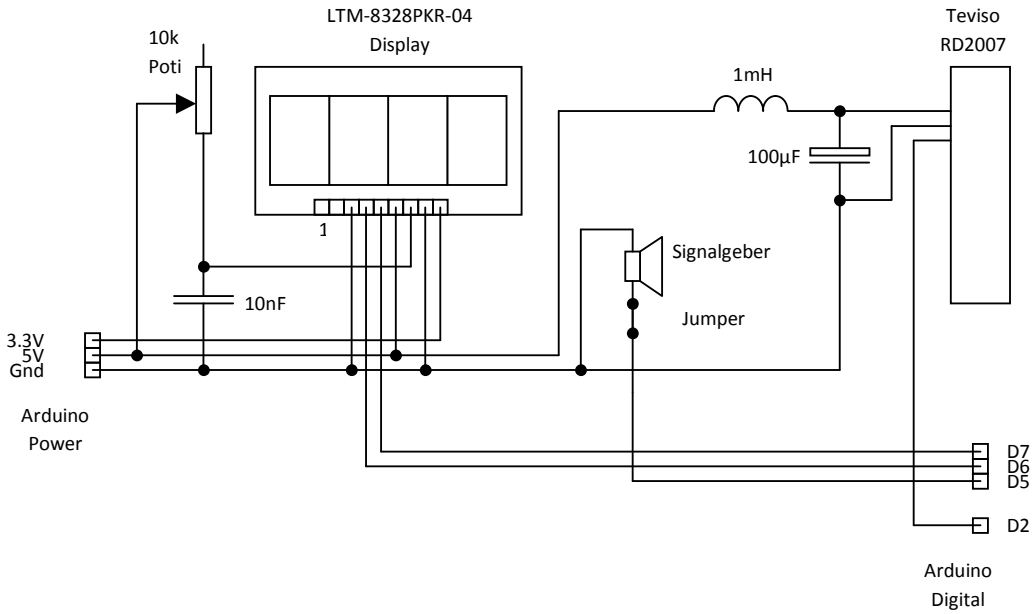

Abb. 2 Schaltplan des Tino-Shield

Das besondere an diesem Zählprogramm ist, dass nicht wie normal für ein vorgegebenes Zeitintervall die Anzahl an eingetroffenen Zählimpulsen gezählt wird, sondern es wird gewartet, bis eine vorgegebene Anzahl MAXCNT an Impulsen eingetroffen ist und dann wird bestimmt welche Zeit vergangen ist um die Zählrate zu bestimmen. Dieser Ansatz hat den Vorteil, dass es Proben gibt, für die man sehr lange zählen muss um eine gewisse Anzahl Impulse und damit eine gewisse statistische Sicherheit zu bekommen. Für andere Proben hat man schnell die Notwendige Anzahl beieinander. Wenn man nun schnell auf einen Anstieg der Rate von einem niedrigen Wert auf einen hohen Wert reagieren will, wäre es ungünstig wenn man wegen der niedrigen Rate ein langes Zählintervall vorgegeben hätte. Dann bekommt man nämlich den Anstieg auf die hohe Rate erst nach Ablauf des langen Intervalls mit. Mit diesem Ansatz bleibt also die Statistik (Streuung des Ergebnis-Wertes) unabhängig von der Zählrate. Da man für die natürliche Strahlung mit einem Teviso Sensor durchaus eine Stunde brauchen kann, bis man 10 Pulse gezählt hat, ist diese Strategie ganz sinnvoll, so dass man nicht noch mal eine Stunde warten muss um mitzubekommen, dass die Rate auf z.B. 3 Pulse pro Minute hochgegangen ist.

Neben den Standard Routinen setup() und loop() gibt es bei diesem Zählprogramm noch die Routine count(), welche das eigentliche Zählen und berechnen der Zählimpulsrate bewerkstelligt. Sie ist eine sogenannte Interrupt Service Routine (ISR) und wird dann ausgeführt, wenn auf dem Interrupt Eingang 0 (Digital Pin 2) eine steigende Flanke registriert wird. Das wird mit der Anweisung „attachInterrupt(0, count, RISING);“ so vereinbart. Das heißt, im Normalfall befindet sich der Mikrocontroller in der Schleifenroutine loop() und gibt dabei im Abstand von 1 Sekunde laufend die augenblickliche Rate aus. Sobald aber das Eintreffen eines Zählimpulses registriert wird, legt er diese Arbeit kurz zur Seite und bearbeitet das, was in der ISR count() vorgegeben ist. In der ISR wird zunächst die Interrupt Vereinbarung wieder gelöscht (detachInterrupt(0);). Danach wir der Impulszähler inkrementiert und für 50ms der Signalgeber angeschaltet. Die Variable counter muss außerhalb der Routine mit der Eigenschaft „volatile“ deklariert werden, genau wie die Variablen, welche der Zeitmessung dienen, da sie sich in der ISR ändern können müssen, obwohl sie eigentlich in anderen Routinen verwendet werden.

Nur für den Fall, dass die vorgegebene maximale Impulszahl MAXCNT erreicht ist, wird die Zählimpulsrate berechnet. Dazu wird die Zeitdifferenz dt gemessen, welche zwischen dem letzten Erreichen von MAXCNT und der augenblicklichen Zeit vergangen ist. In die Zählimpulsrate wird noch der Kalibrierfaktor hineingerechnet. Wenn er nicht auf 1 steht, sondern wie hier für den RD2007 Sensor auf 3.4, dann ist das Ergebnis die Ortsdosisleistung. Danach muss die Interrupt-Vereinbarung natürlich wieder neu gemacht werden, bevor die Ausführung wieder in der loop() Routine wie gehabt weitergeht und nun die neue Rate bzw. Dosisleistung ausgegeben wird.

Um die Ausgabe nun auf dem 7-Segment Display zu machen und nicht über den serial Monitor, muss ein Objekt (eine Variable) sevSeg vom Typ LTM8328PKR04 vereinbart werden. Das wird durch die Display-Bibliothek so vorgegeben. Bei der Objektvereinbarung werden die verwendeten Pinnummern für den Daten- und den Clockpin mitgegeben. Für dieses Objekt werden damit einige Methoden (Routinen) verfügbar, mit denen man auf das Display zugreifen kann. Auf das Display schreiben kann man mit der Methode .print(). Dabei wird der eigentliche Wert als Ganzzahl mit der Integer-Variablen i übergeben, gefolgt von einer Byte-Variablen, welche die Position des Dezimalpunkts angibt. Da die Rate als Float-Größe mit Fließkomma-Arithmetik berechnet wird, ist eine Konversion ins Ganzzahlformat notwendig, was das Abschneiden der Nachkommastellen zur Folge hat. Wenn man die Ortsdosis in µSv/h ausgeben möchte, dann hat man es aber mit mindestens einer Stelle nach dem Komma zu tun. Um diese nicht zu verlieren, muss die berechnete Größe (rate) vor der Konversion mit dem Faktor 10 multipliziert werden, was schließlich dadurch wieder kompensiert wird, dass man den Dezimalpunkt hinter die dritte Ziffer verschiebt.

Man kann die korrekte Funktion dieses Zählprogramms auch ganz leicht mit einem Impulsgenerator testen, der periodische Impulse mit einer Dauer von ca. 200µsec ausgibt und das bei einer Frequenz von ca. 5.6Hz. Dann benötigen 10 Impulse (MAXCNT=10) nämlich 10*1/5.6Hz = 1.79 Sekunden. Gemessen wird diese Zeit vom Arduino in Millisekunden, also ist dt = 1790ms. Das Programm rechnet dann aus dieser Zeit aus, dass die Dosisleistung 10*60*1000/1790/3.4 also ungefähr 100µSv/h = 0.1mSv beträgt.

Dieses Programm kann nun auch ganz leicht dahingehend abgeändert werden, dass z.B. der Signalgeber ab einer bestimmten Ortsdosisleistung einen konstanten Signalton als Warnung abgibt. Weitere Verbesserungen und Erweiterungen sind wegen der Übersichtlichkeit und Einfachheit des Konzepts ebenfalls ganz einfach möglich.

```
#include <LTM8328PKR04.h>
#define MAXCNT 10
#define CalFactor 3.4

volatile int counter = 0;
volatile unsigned long time;
volatile unsigned long oldTime = 0;
volatile unsigned long dt = 0;
volatile float rate = 0.0;

const byte dataPin = 6;
const byte clockPin = 7;
int speaker = 5;

LTM8328PKR04 sevSeg(dataPin,clockPin);

void setup()
{
  pinMode(speaker, OUTPUT);
  //Serial.begin(9600);
  sevSeg.setLeadingZeros(1);
  attachInterrupt(0, count, RISING);
}

void loop()
{
  int i = (int)(rate*10.0);
  //Serial.println(rate);
  sevSeg.print(i,3);
  delay(1000);
}

void count()
{
  detachInterrupt(0);
  counter++;
  digitalWrite(speaker, HIGH);
  delay(50);
  digitalWrite(speaker, LOW);
  if (counter == MAXCNT) {
    oldTime = time;
    time = millis();
    dt = time-oldTime;

    rate = (float)MAXCNT*60.0*1000.0/(float)dt/CalFactor;
    counter = 0;
  }
  attachInterrupt(0, count, RISING);
}
```

Abb. 3: Einfaches Beispiel-Programm für das Tino-Shield

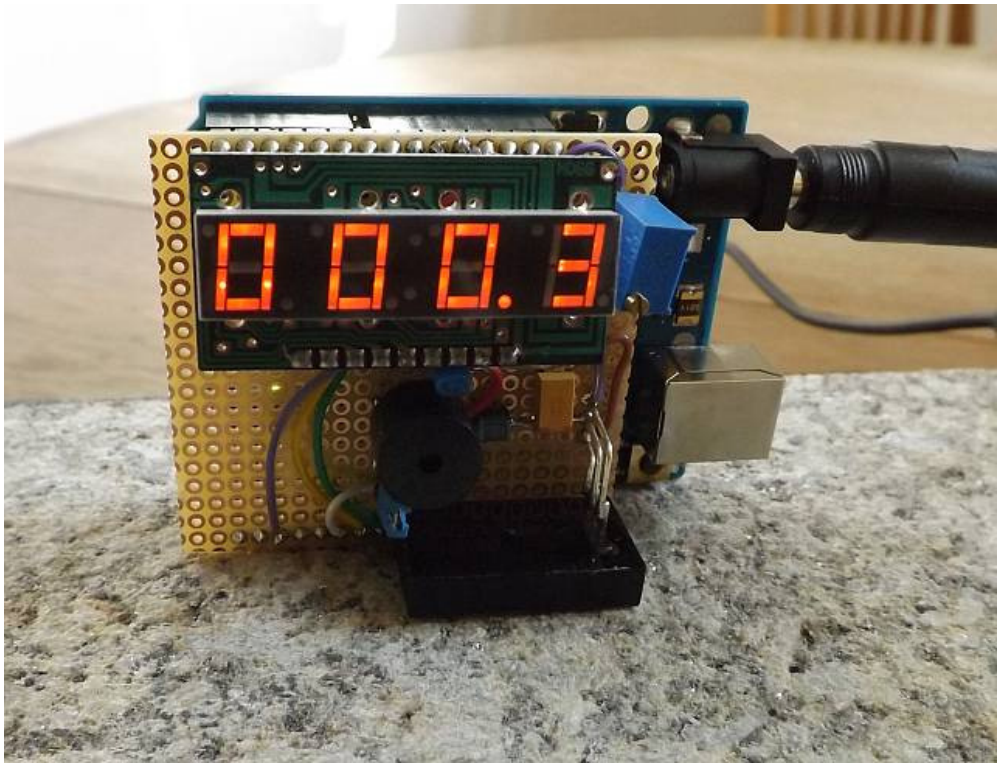

Abb. 4: Der Tino auf einer Bodenfliese aus Flossenbürger Granit

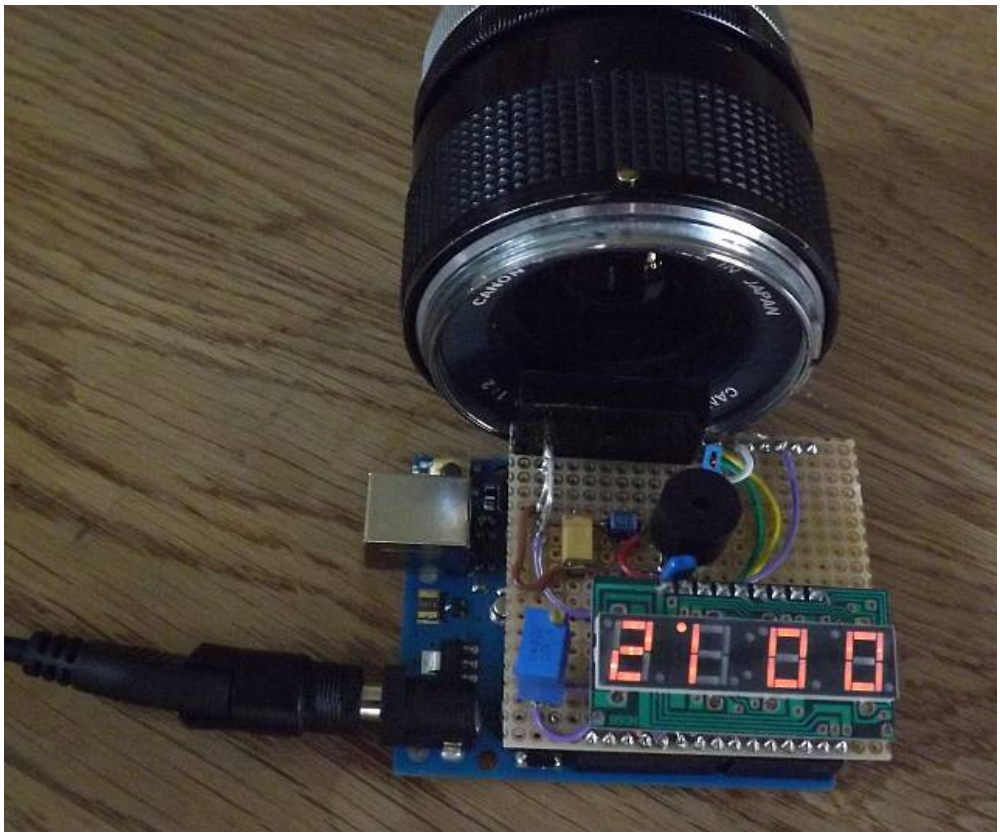

Abb. 5: Der Tino vor einem mit Thorium „veredelen“ Foto-Objektiv (Anzeige steht auf dem Kopf)

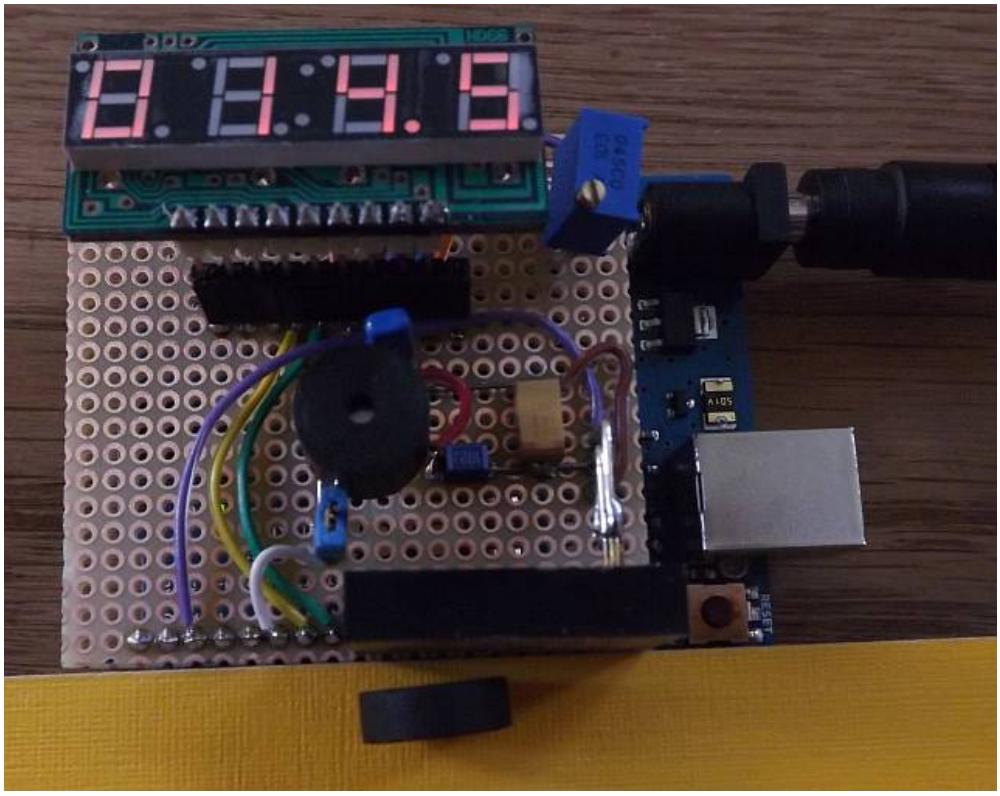

Abb. 6: Messung einer mit Radon beladenen Kohle-Tablette mit dem Tino

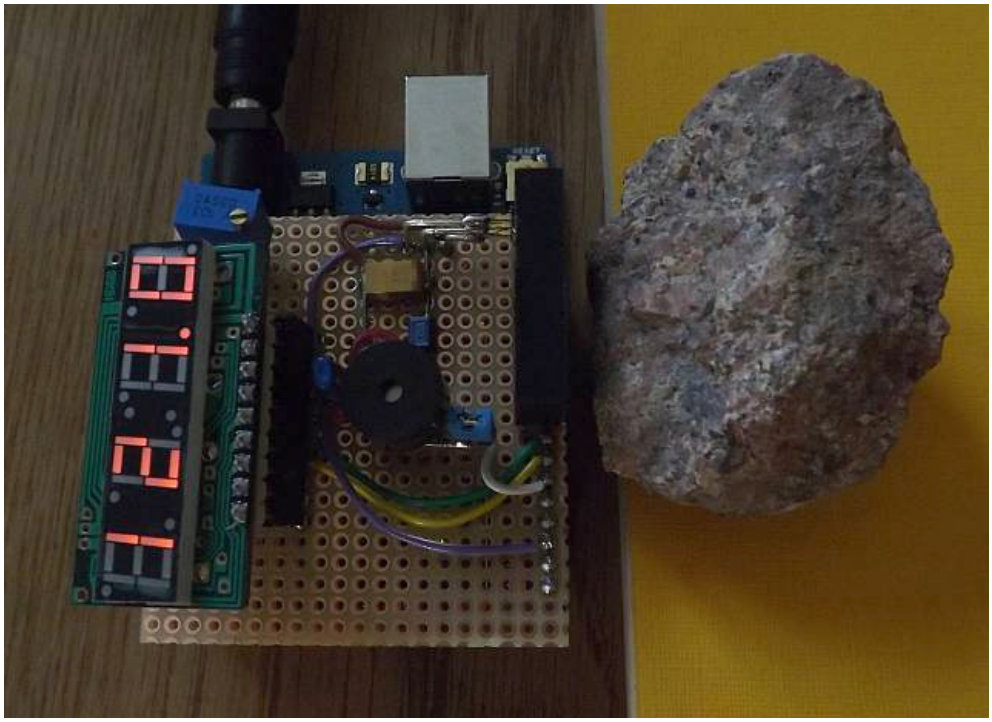

Abb. 7: Messung an einer ca. 1cm großen Uranoxid-Einlagerung in einem Granit aus Menzenschwand

Literatur und Quellen:

/1/ Teviso Sensormodule
http://www.teviso.com

/2/ Lite-On LTM-8328PKR-04 4x7 LED segments display library for Arduino
https://github.com/goebish/LTM8328PKR04/

/3/ Datasheet des LTM-8328PKR-04 bei Digikey:
http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTM-8328PKR-04.pdf

/4/ Ekulit Signalgeber mit Ansteuerung Typ: AL - 60 SP 05
http://www.ekulit.de/fileadmin/Bilder/produkte/signalgeber_mit/170050%20AL-60SP05.pdf

/5/ Physikalisch-Technische Bundesanstalt Braunschweig und Berlin
Was bedeuten die Dosismessgrößen der neuen Strahlenschutzverordnung für die Messtechnik?
Konzept der Dosisgrößen im Strahlenschutz
Jürgen Böhm
171. PTB-Seminar, 20. und 21. Juni 2002
http://www.ptb.de/de/org/6/63/information/vortrag1.pdf

/6/ Das Teviso Pin-Dioden Sensormodul RD2007 als Detektor für radioaktive Strahlung
http://opengeiger.de/TevisoModul.pdf

/7/ Einfaches Messgerät für die radioaktive Dosisleistung mit dem Arduino und dem RD2007 Sensormodul von Teviso
http://opengeiger.de/TevisoArduino.pdf