Geigerzähler-Shield für den Arduino von Libelium

Bernd Laquai 17.1.2014

Wenn es auch ungemein Spaß machen kann, eine Schaltung selbst zu entwickeln, nicht jeder hat die Zeit, die Hardware-Kenntnisse oder die Infrastruktur an Hilfsmitteln dazu einen Geigerzähler selbst zu bauen. Aber manchmal sind genügend Informatikkenntnisse vorhanden bzw. man hat einfach mehr Spaß an der Software-Entwicklung, so dass bei gegebener Hardware immer noch genug Spaß bleibt, die Signale zur Anzeige zu bringen. Auch dafür bietet der Arduino eine sehr günstige Platform, denn es gibt auch sogenannte Geigerzähler Shields, also Platinen mit der nötigen Hardware, die man auf die Prozessor-Platine einfach aufstecken kann. Programmiert man sich den Arduino dann in der geeigneten Weise, oder nimmt gar eines der vorgefertigten Beispielprogramme, dann hat man in weniger als einer Stunde mit ein paar Mausklicks einen funktionsfähigen Geigerzähler zusammengestrickt und hat am Ende nicht mehr als etwa 160Euro ausgegeben.

Das wirklich gute daran ist aber, dass man ein äußerst flexibles Grundgerät hat, welches man dann beliebig abwandeln kann und durch die Spezialisierung auf ein bestimmtes Problem hin dann eine deutliche Wertsteigerung erreichen kann. So wäre es beispielsweise ein einfaches so einen Arduino basierten Geigerzähler mit einem GPS Modul zu erweitern und dann an einen Quadrokopter zu einer Messdrohne zu machen, welche geo-referenzierte Messungen aus der Luft von Orten macht, die man zu Fuß besser nicht erkunden will.

Eines dieser Geiger-Shields wurde unmittelbar nach der Katastrophe von Fukushima von der Firma Libelium entwickelt und Technik-orientierten Leuten in Tokio zur Verfügung gestellt um dort Nahrungsmittel und anderes auf Strahlung zu prüfen bzw. in Verbindung mit GPS Empfängern Strahlungskarten für die Bevölkerung zu erstellen. Nach dem Unglück war nämlich in Fukushima kein Geiger-Zähler mehr zu bekommen obwohl der Bedarf gerade wegen der Nahrungsmittelproblematik ins unermessliche wuchs.

Das von Libelium entwickelte Shield ist in allen Details veröffentlicht und erklärt. Die Dokumetation findet sich unter auf der Webseite von Libeliums Bastler-Webseite „Cooking-Hacks“. Es ist eine solide und robust entwickelte Platine, vielleicht mit der Ausnahme der Anschlüsse für das Zählrohr. Das ist etwas filigran geraten, hat aber den Vorteil dass man die unterschiedlichsten Zählrohre anklemmen kann bzw. unter Umständen auch zwei Zählrohre parallel schalten kann.

Geliefert wird das Shield mit einem chinesischen Glas-Zählrohr J305β, das mit seinen 11cm Länge schon enorm empfindlich ist. Angeblich soll es Beta-empfindlich sein, allerdings kann man nur eine Sensitivität für harte Beta-Strahlung erkennen. Ein Tritium-Gaslicht erkennt es nicht. Daher ist die Empfindlichkeit für soft-Beta Strahlung wohl nicht gegeben. Für Gamma-Strahlung zeigt es allerdings eine sehr gute Empfindlichkeit.

Das Shield wird auch mit einer aufsteckbaren, 2-zeiligen LCD-Anzeige geliefert, die dem Hitachi-Format (4-bit parallel) folgt. Das Beispielprogramm benutz das LCD-Display zur Anzeige der Zählrate und eine Dosisleistungswerts in uSv/h der durch eine einfache Umrechnung der Zählrate bestimmt wird. Sie belegt natürlich einige der Digitalpins, deswegen mag es fälle geben, wo man sie zugunsten eines anderen Shields weglässt. Wenn man die Anzeigefunktion weglässt, braucht man zum Betrieb des Zählers außer der

Stromversorgung nur den Digital-Pin 2, welcher die Zählpulse des Shields als Interrupt-Signale auswertet. Dann hat man also für ein weiteres Shield alle anderen Pins noch zur verfügung.

Die Schaltung ist im Prinzip sehr einfach aber doch sehr stromsparend aufgebaut. Als Hochspannungsquelle wird ein Boost- (Step Up) Konverter mit diskreten Transistoren verwendet, die Spannung wird aber noch zusätzlich über einen Transformator hochgespannt. Was an der Schaltung sehr gut gemacht ist, ist die Regulierung der Hochspannung. Über eine Serienschaltung dreier Zenerdioden und der Basis-Emitter Strecke eines Regeltransistors wir der Oszillator so beeinflusst, dass die Hochspannung nicht über einen gewissen Wert ansteigt. Da auf dieses Weise auch der Oszillator nicht ständig unter Volllast läuft, sinkt dadurch der Stromverbrauch erheblich.

Der Zählimpulsverstärker ist eine doppelte Darlington-Stufe aus zwei diskreten Transistoren, die ihr Eingangssignal an einem Spannungsteiler des Anodenvorwiderstands abnehmen. Daher ist eine kleine Hochspannungsfeste Kapazität nötig (C7, 12pF) um die Darlington-Stufe gleichspannungsfrei abzukoppeln. Die Zählimpulse werden nach der Verstärkung noch mit einer 4.7uF Kapazität etwas verlängert und auf einen aktiven Signalgeber mit integrierter Tonerzeugung gegeben. Gleichzeitig wird dieses Signal auf den Digitalpin 2 des Arduino geführt. Im Beispielprogramm wird durch die auf verschiedenen Zeilen verteilten Statements:

```
int geiger_input = 2;
pinMode(geiger_input, INPUT);
digitalWrite(geiger_input,HIGH);
attachInterrupt(0,countPulse,FALLING);
```

der Pin 2 zunächst als Input mit Pull-Up auf High aufgesetzt und dann als Interrupt-Eingang 0 benutzt, der eine Interrupt-Anforderung für eine fallende fallende Signalflanke auslöst. Die Interrupt-Behandlungsroutine heißt countPulse() und tut nichts anderes als den Impulszähler hochzählen. Sie wartet dann bis das Signal wieder hochgeht und gibt dann die Kontrolle in die Hauptprogramm-Schleife zurück. In der Hauptprogramm-Schleife wird die Funktion millis() benutzt um die Uhr auszulesen und ein Zeitintervall von 10 Sekunden zu definieren nach dessen Ablauf die Rate berechnet und ausgegeben wird. Zusätzlich wird anhand von Schwellwerten (TH1-5) und einer Routine ledVar() eine LED-Zeile aus 3 grünen und 2 roten noch eine Strahlungs-Intensitätsanzeige implementiert.

Das mitgelieferte Beispielprogramm ist für den einfachen Hausgebrauch bereits schon recht gut geeignet. Selbst der voreingestellte Umrechnungsfaktor von Zählrate in uSv/h passt gar nicht schlecht, sollte aber auf jeden Fall an Hand einiger Prüfstrahler überprüft werden. Die Größe des Zählrohrs macht dabei allerdings etwas Probleme, da es natürlich eine deutlich größere effektive Fläche hat als ein kleines Fensterzählrohr. So sieht man einen krassen Unterschied, wenn man einfach nur die Nullrate auf dem Boden eines Wohnraums misst und danach das Zählrohr in eine Keramikschale mit Uranglasur in Stellung bringt. In dieser Geometrie ergeben sich natürlich astronomisch hohe Dosisleistungen. Korrekterweise muss man schon durch einen gewissen Abstand sicherstellen, dass man trotz der Größe des Zählrohrs noch von einem punktförmigen Detektor ausgehen kann. In 1m Abstand zur Strahlungsquelle ist das sicher deutlich besser gewährleistet. Allerdings ist doch erstaunlich, dass überhaupt eine so hohe Zählrate entstehen kann in Anbetracht von Totzeiten durch die

analoge Tiefpassfilterung im analogen Signalpfad und der zusätzlichen Verarbeitungszeit in der Interrupt-Behandlungsroutine. Hier leistet der Arduino offenbar eine durchaus flotte Abarbeitung der Interrupt Requests.

Insgesamt kann man sagen hat das Geigerzähler Shield zwar einen stolzen Preis von zwischen 120 und 150Euro je nachdem wo man kauft, aber es bietet eine sehr hohe Flexibilität für individuelle Erweiterungen. Das größte Problem wird vermutlich das Bereitstellen eines passenden und robusten Gehäuses sein, besonders wenn man das Gerät auch im Outdoor Bereich einsetzen möchte. Wofür sich der Zähler recht gut eignen wird ist als Kontaminationsmonitor und im Niedrigdosisbereich, da das große Zählrohr eine relativ hohe Rate pro Dosis hat und man sicherlich auch zwei Rohre parallel schalten kann, sofern sie einigermaßen identisch sind bzw. wenn man ein Pancake-Rohr mit großem Durchmesser anschließt.

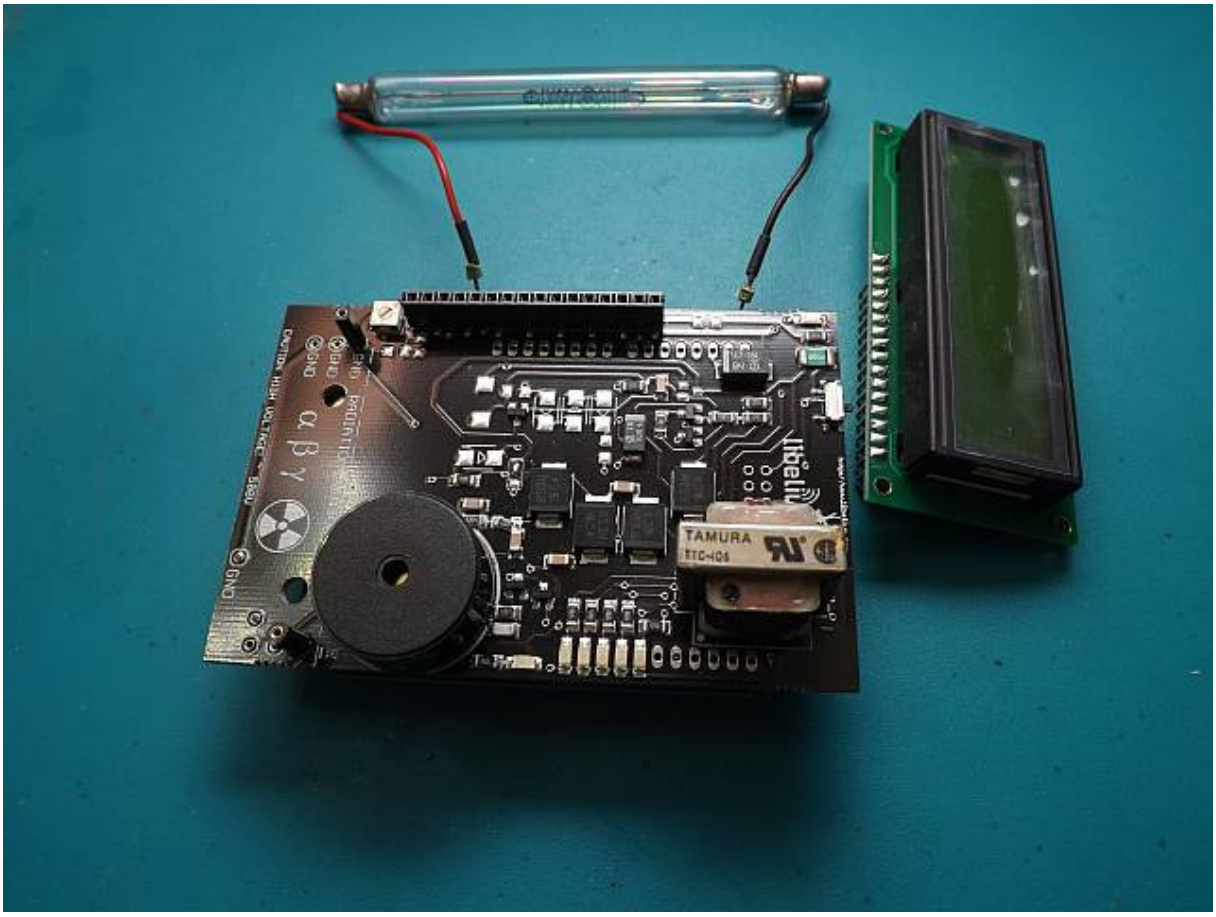

Abb. 1: Das Libelium Geiger-Shield mit Glas-Zählrohr und LCD-Anzeige

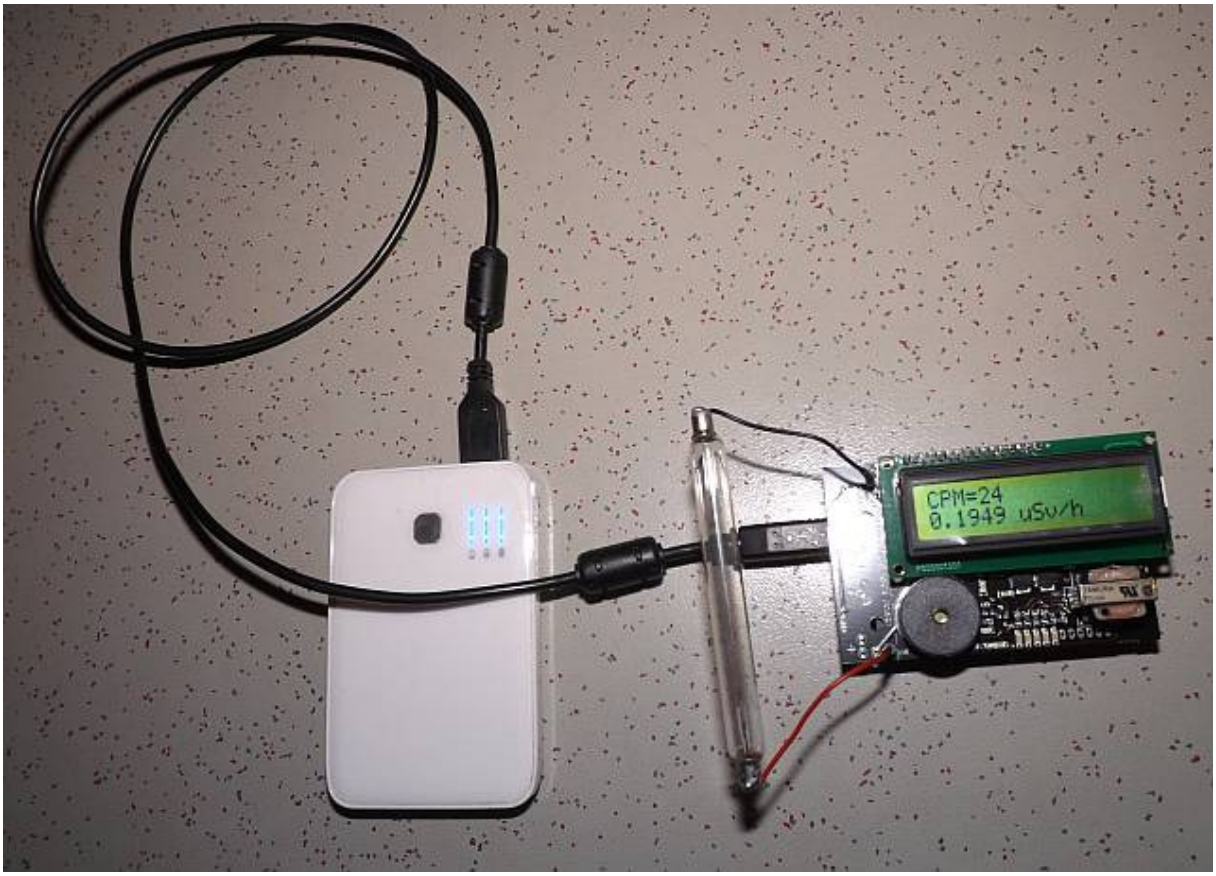

Abb. 2: Nullraten-Messung auf dem Boden

Abb. 3: Messung einer hohen Zählrate in einer Keramikchale mit Uranglasur

Spezifikation bei Exp-Tech.de:

Detect Alpha, Beta and Gamma radiation integrating any Geiger Tube which works in the range 400V - 1000V and read this levels using Arduino. As well as from the terminal, the radiation levels can be shown using different actuators:

Piezo: it allows us to hear the typical "chirp" common in the radioactivity counters
Leds: 3 green and 2 red let easily to show low, medium and high levels
LCD: it displays the counts per minute (cpm) and the equivalent absorbed energy levels in Servants (µSV/h).

The current version of the pack comes with the J305ß Geiger tube which detectes Beta and Gamma radiation.

Specifications:

Manufacturer: North Optic
Radiation Detection: β, γ
Length: 111mm
Diameter: 11mm
Recommended Voltage: 350V
Plateau Voltage: 360-440V
Sensitivy γ (60Co): 65cps/(µR/s)
Sensitivy γ (equivalent Sievert): 108cpm / (µSv/h)
Max cpm: 30000
cps/mR/h: 18
cpm/m/h: 1080
cpm/µSv/h: 123.147092360319
Factor: 0.00812037037037

NOTES:

There is a little switch below the LCD you must switch on to use the board.
the Geiger tube should not be exposed to sunlight because the CPM may not be as accurate as indoors. We advise to use an opaque enclosure when working outdoors to cover this sensor.

Bemerkung von Libelium zum Zählrohr:

Finally we have chosen the J305ß as the official tube for the radiation sensor board as it is fairly sensible to Beta and Gamma radiation at the same time it keeps a affordable price at a minimum size (11cm).

Link zur Dokumentation auf der Cooking-Hacks Webseite von Libelium:
http://www.cooking-hacks.com/documentation/tutorials/geiger-counter-arduino-radiation-sensor-board

Link zum Geiger-Shield bei Exp-Tech.de :
Exp-Tech: http://www.exp-tech.de/Shields/Pack-Radiation-Sensor-Board-for-Arduino-Geiger-Tube.html

Mitgelieferter Beispielcode:

```
/*
 *  ------Geiger Tube board (Arduino Code) Example--------
 *
 *  Explanation: This example shows how to get the signal from the Geiger
Tube
 *  in Arduino, we use one of the Arduino interrupt pins (PIN2).
 *  We count the time (ms) between two pulses of the Geiger tube.
 *
 *  Copyright (C) 2011 Libelium Comunicaciones Distribuidas S.L.
 *  http://www.libelium.com
 *
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 *  Version:           0.3
 *  Design:            Marcos Yarza, David Gascon
 *  Implementation:    Marcos Yarza
 */

// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(3,4,5,6,7,8);


// Threshold values for the led bar
#define TH1 45
#define TH2 95
#define TH3 200
#define TH4 400
#define TH5 600

// Conversion factor - CPM to uSV/h
#define CONV_FACTOR 0.00812

// Variables
int ledArray [] = {10,11,12,13,9};
int geiger_input = 2;
long count = 0;
long countPerMinute = 0;
long timePrevious = 0;
long timePreviousMeassure = 0;
long time = 0;
long countPrevious = 0;
float radiationValue = 0.0;

void setup(){
  pinMode(geiger_input, INPUT);
  digitalWrite(geiger_input,HIGH);
  for (int i=0;i<5;i++){
```

```
    pinMode(ledArray[i],OUTPUT);
  }

  Serial.begin(19200);

  //set up the LCD\'s number of columns and rows:
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Radiation Sensor");
  lcd.setCursor(0,1);
  lcd.print("Board - Arduino");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Cooking Hacks");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("www.cooking-hacks.com");
  delay(500);
  for (int i=0;i<5;i++){
    delay(200);
    lcd.scrollDisplayLeft();
  }
  delay(500);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("  - Libelium -");
  lcd.setCursor(0,1);
  lcd.print("www.libelium.com");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("CPM=");
  lcd.setCursor(4,0);
  lcd.print(6*count);
  lcd.setCursor(0,1);
  lcd.print(radiationValue);

  attachInterrupt(0,countPulse,FALLING);

}

void loop(){
  if (millis()-timePreviousMeassure > 10000){
    countPerMinute = 6*count;
    radiationValue = countPerMinute * CONV_FACTOR;
    timePreviousMeassure = millis();
    Serial.print("cpm = ");
    Serial.print(countPerMinute,DEC);
    Serial.print(" - ");
    Serial.print("uSv/h = ");
    Serial.println(radiationValue,4);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("CPM=");
    lcd.setCursor(4,0);
    lcd.print(countPerMinute);
```

```
    lcd.setCursor(0,1);
    lcd.print(radiationValue,4);
    lcd.setCursor(6,1);
    lcd.print(" uSv/h");

    //led var setting
    if(countPerMinute <= TH1) ledVar(0);
    if((countPerMinute <= TH2)&&(countPerMinute>TH1)) ledVar(1);
    if((countPerMinute <= TH3)&&(countPerMinute>TH2)) ledVar(2);
    if((countPerMinute <= TH4)&&(countPerMinute>TH3)) ledVar(3);
    if((countPerMinute <= TH5)&&(countPerMinute>TH4)) ledVar(4);
    if(countPerMinute>TH5) ledVar(5);

    count = 0;

  }

}

void countPulse(){
  detachInterrupt(0);
  count++;
  while(digitalRead(2)==0){
  }
  attachInterrupt(0,countPulse,FALLING);
}

void ledVar(int value){
  if (value > 0){
    for(int i=0;i<=value;i++){
      digitalWrite(ledArray[i],HIGH);
    }
    for(int i=5;i>value;i--){
      digitalWrite(ledArray[i],LOW);
    }
  }
  else {
    for(int i=5;i>=0;i--){
      digitalWrite(ledArray[i],LOW);
    }
  }
}
```