Szintillationszähler auf der Basis eines Channel-Photomultipliers

Bernd Laquai 2.4.16

Channel-Photomultiplier (CPMs) bieten die Möglichkeit ohne großen Aufwand einen kompakten Szintillationszähler aufzubauen. Ältere Modelle bekommt man recht günstig über Ebay und man kombiniert sie aufgrund ihrer kleiner Fenstergröße am bestem mit kleinen Kristallen und erhält so auch eine recht kostengünstige Lösung. Das vom Hersteller bereitgestellte Hochspannungsmodul sollte man aber passend dazukaufen, denn die meist 3 Hochspannungsquellen selbst herzustellen, wovon zwei im Bereich von 2...3kV einstellbar sein sollten, wäre extrem aufwändig. Diese Hochspannungsmodule werden primärseitig meist mit 5V betrieben, was in Verbindung mit einer Mikrocontrollersteuerung sehr günstig ist. Man sollte beim Kauf von CPMs auch beachten, dass das Photokathodenmaterial und das Material des Eintrittsfensters in etwa auch zum Szintillationskristall passt, denn es gibt doch recht starke Unterschiede im Bereich der spektralen Empfindlichkeit der CPMs.

Anders als normale Photomultiplier-Röhren (PMTs) verwenden CPMs keine Dynoden zur Sekundärelektronen-Vervielfachung sondern einen halbleitenden Kanal wobei die Vervielfachung an der Wand des gekrümmten Kanals erreicht wird. Er ist in einer mehrfach wellenförmig gebogen Glasröhre untergebracht, die ihrerseits in ein lichtdichtes Kunststoffrohr eingebettet ist. CPMs können in sehr kleinen Bauformen hergestellt werden und haben meist nur 10-20mm Durchmesser bei einer Länge von 10-20cm.

Abb. 1: 10x40mm NaI-Kristall und CPM von Perkin-Elmer

Abb. 2: Auf das CPM lichtdicht aufgesetzter Kristall

Abb. 3: Alu-Schutzhülle aus einer Zigarillo-Verpackung

CPMs erreichen eine extrem hohe Verstärkung, so dass eine zusätzliche elektronische Verstärkung meist nicht mehr nötig ist, wenn man den Ausgang nicht zu stark belastet. Verwendet man einen Lastwiderstand von mehreren 10 bis 100kOhm zur Konversion des Ausgangsstroms in ein Spannungssignal entstehen Pulse von 50-200us Dauer, die man nach Justage des Vorspannungs-(Bias-) Pegels direkt auf den Interrupteingang eines Mirkocontrollers geben kann.

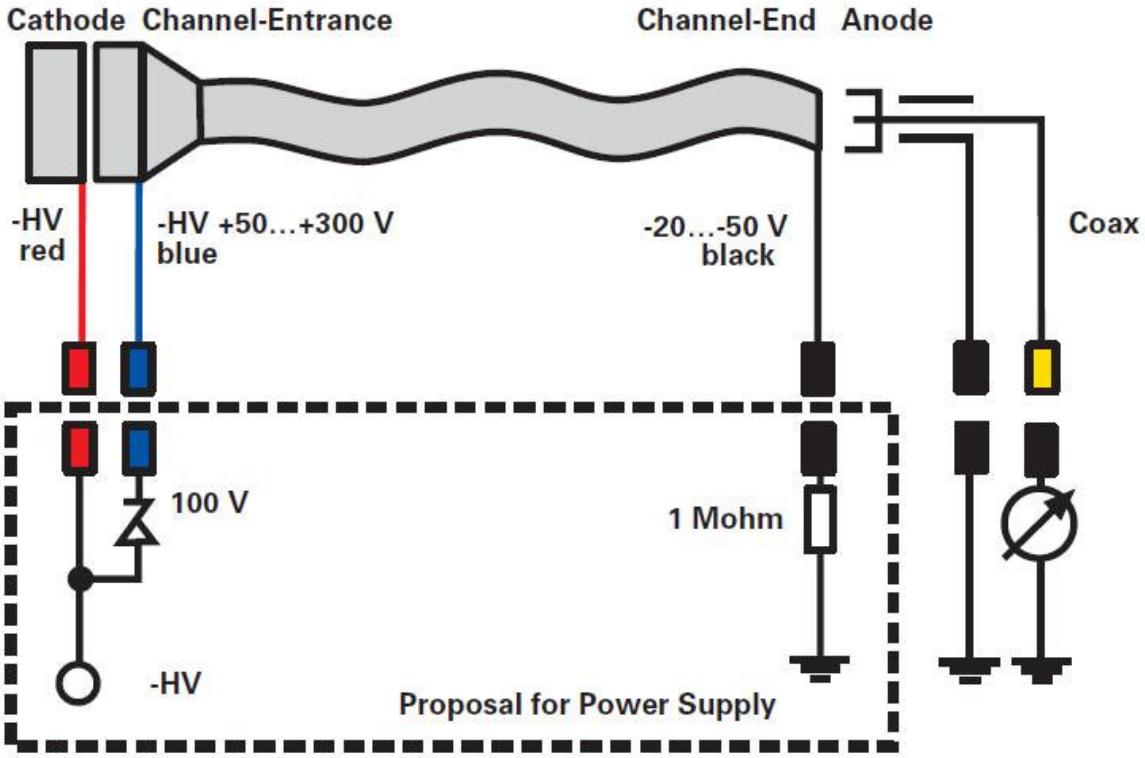

Abb. 4: Anschlussschema des zum CPM gehörigen negativen Hochspannungsmoduls (CHV30N, Perkin-Elmer). Nicht gezeigt sind die Steuer-Anschlusse, wovon nur der Vset-Eingang an ein Poti angeschlossen werden muss, so dass er zwischen 0 und 3V eingestellt werden kann.

Anwendungen mit negativer Hochspannungsversorgung (Kathode auf –HV) sind häufiger, deswegen sind negative Hochspannungsmodule leichter zu bekommen als positive. Das bedeutet aber, dass die Pulse des Spannungssignals am Lastwiderstand ins Negative gehen. Verwendet man als Mikrocontroller beispielsweise einen 5V Typ, wie z.B. den Arduino Uno oder Nano, dann hat der Interrupteingang meist jedoch eine Schaltschwelle in der Gegend um 2.5V. Das lässt sich aber durch eine einfache Pegelverschiebung mit wenigen passiven Bauelementen aneinander anpassen. Zunächst braucht man den Widerstand von ca. 100k gegen Ground, der sicherstellt, dass sich die Anode nicht statisch auflädt. Er wird direkt am Koaxialkabel des CPMs angeschlossen. Dieses Signal wird dann über einen Koppelkondensator auf einen zweiten Widerstand der ebenfalls als mit ca. 100k gewählt werden kann, geführt, der nun aber auf eine einstellbare positive Spannung zwischen 2.5V und 5V vorgespannt wird, so dass der DC-Pegel über der digitalen Schaltschwelle des Interrupteingangs zu liegen kommt. Die Vorspannung wird mit Hilfe eines niederohmigen Potentiometers erzeugt. Der eigentliche Lastwiderstand besteht nun also aus der Parallelschaltung der beiden 100k Widerstände. Dadurch aber, dass der zweite beispielsweise auf 4V gelegt werden kann, gehen nun die Pulse von 4V aus ins negative und überschreiten daher sicher die Schaltschwelle des Interrupteingangs. Würde man es dabei belassen, könnte es allerdings passieren, dass bei starken Impulsen die deutlich unter Null gehen würden, die Schutzdioden des Digitaleingangs leitend werden. Um das zu verhindern und den Eingang zu schonen, kann man eine schnelle Schottky-Diode in Sperrrichtung parallel zum Eingang legen. Sie übernimmt dann den Strom für den Fall dass ein Impuls stark negativ werden würden, da sie schneller leitend wird, als die Si-Schutzdiode. Durch die entsprechende Rückwirkung auf den CPM-Ausgang, der wie eine Stromquelle wirkt,

erscheinen nun alle Pulse skaliert zwischen 0 bis 4V. Wählt man nun über das Potentiometer eine Vorspannung, die geringer ist als 5V, dann schiebt man so den Startpegel der Pulse näher an die Schaltschwelle des Interrupteingangs und kann so auch Pulse detektieren, deren Amplitude kleiner ist als 2.5V. D.h. der Aufwand für die Anpassung des CPM Ausgangs an eine Mikrocontroller beschränkt sich auf 2 Widerstände, die Koppelkapazität, die Diode und das Potentiometer.

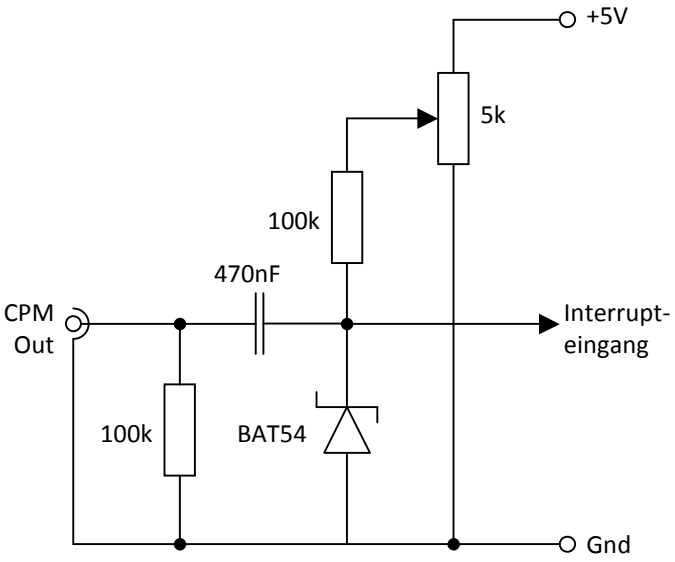

Abb. 5: Schaltung zur Anpassung des CPM Spannungspegel an den Interrupt-Digitaleingang des Mikrocontrollers

Der Rest der Schaltung ist trivial und beschränkt sich auf eine Mikrocontroller-Platine mit Anzeigemodul. Da die CPM-Kristall-Kombination schon relativ klein ist, bietet es sich natürlich an auch eine kleine Anzeigemodul und eine kleine Mikrocontroller Platine einzusetzen. OLED Displays werden als sehr kleine Displays mit seriellem Interface angeboten und haben auch den Vorteil, dass sie selbstleuchtende Pixel haben, die zu allen Tageszeiten gut lesbar sind. In der Arduino-Familie ist hat der Nano derzeit die kleinste Platine. Er hat genug Pins um ein kleines, serielles OLED-Display anzusteuern und der verwendete Spannungsregler auf der Platine hat noch genug Luft für den Betrieb des OLEDs und des Hochspannungsmoduls. Alles zusammen erzeugt im Falle des Perkin-Elmer CPMs und des 128x32 Pixel OLED einen Stromverbrauch von ca. 120mA, was gut mit einem 11.1V Modellbauakku gedeckt werden kann.

Für das 128x32 OLED von Adafruit gibt es eine umfangreiche Bibliothek mit der man selbst Grafiken erzeugen könnte. Woran man sich allerdings gewöhnen muss ist die Tatsache, dass die weiße Pixel stets mit dem bereits vorhandenen Bitmuster verodert werden. Um einen neuen Text zu schreiben muss man daher alten Text mit schwarzen Buchstaben löschen wenn man davor nicht gerade das gesamte Display löschen will.

Für das einbinden der OLED Bibliothek in das Arduino Programm werden Beispiele mitgeliefert, die man direkt übernehmen kann. Ansonsten gestaltet sich das Programm für den Arduino Nano äußerst einfach. Da das CPM mit sehr hohen Zählraten aufwartet wurde hier ein festets Zeitintervall von 5 Sekunden gewählt um die Zählrate zu berechnen und

anzuzeigen. Darüberhinaus würde die normale Interruptsteuerung wie in vielen anderen Arduino-Beispielen auf der opengeiger Webseite verwendet.

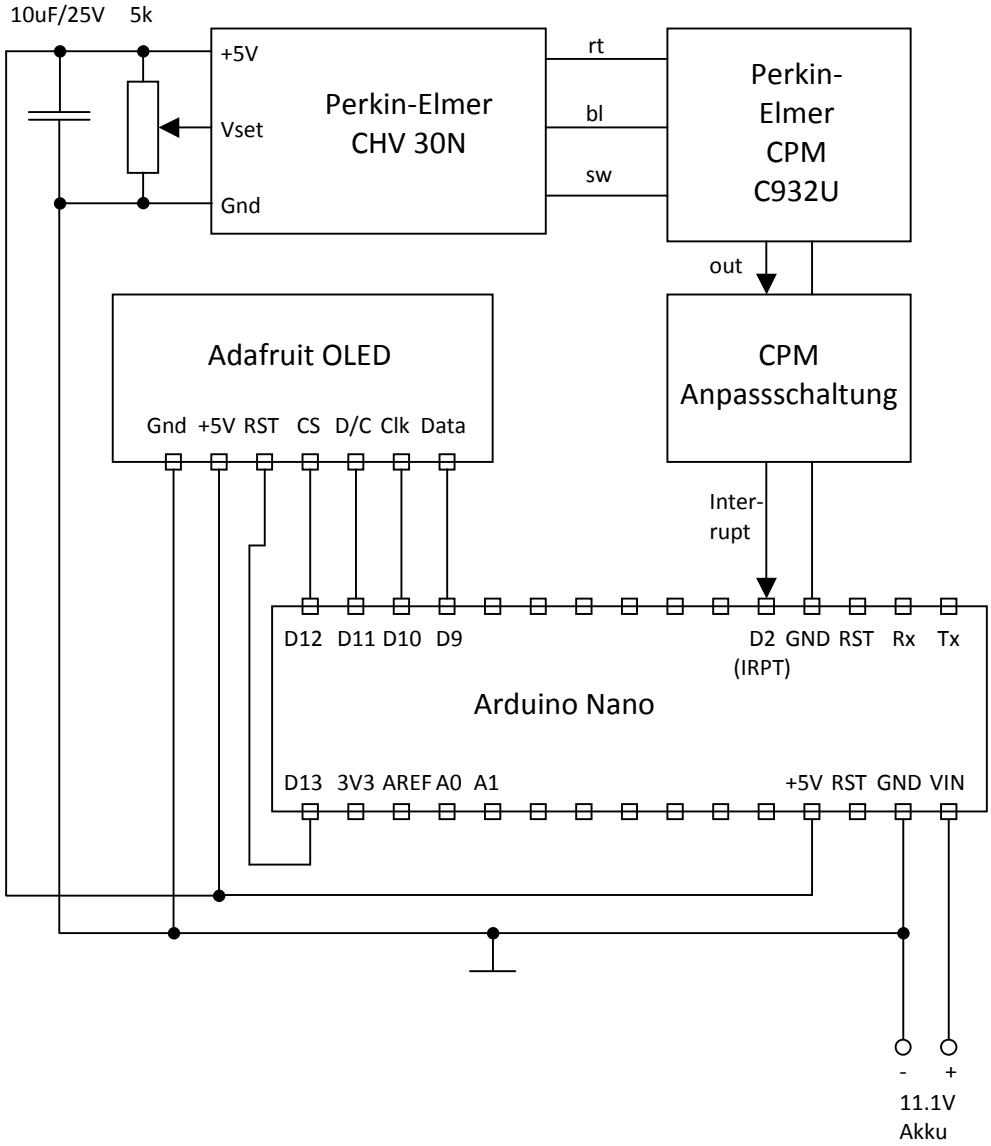

Abb. 6: Gesamtschaltung des Szintillations-Zählers mit Channel Multiplier, Arduino-Nano und OLED-Display

Bei der Inbetriebnahme stellt man fest, dass bereits bei einer Hochspannung von 2kV (Vset = 2.0V) bereits beachtliche Zählraten entstehen. Das hat aber wenig mit Radioaktivität zu tun, die Pulse sind Dunkelstrom-Pulse und entstehen durch thermische Anregung an den Kanalwänden so wie an einem PMT auch Pulse an den Dynoden ganz ohne Licht entstehen. Man stellt die Hochspannung am besten so ein, dass das Verhältnis zwischen Pulsrate mit Probe und ohne Probe noch deutlich genug ist. Dabei sollte man den Pegel in der Anpassschaltung zunächst bei 4V belassen und prüfen, ob man die Pulse eines Rauchmelders (Am241 bei 59keV) deutlich genug sieht. Schiebt man den Pegel näher gegen 3V nimmt die Pulsrate enorm zu, wobei dann aber die Zahl an Dunkelsstrom-Pulsen meist den größten Teil ausmacht.

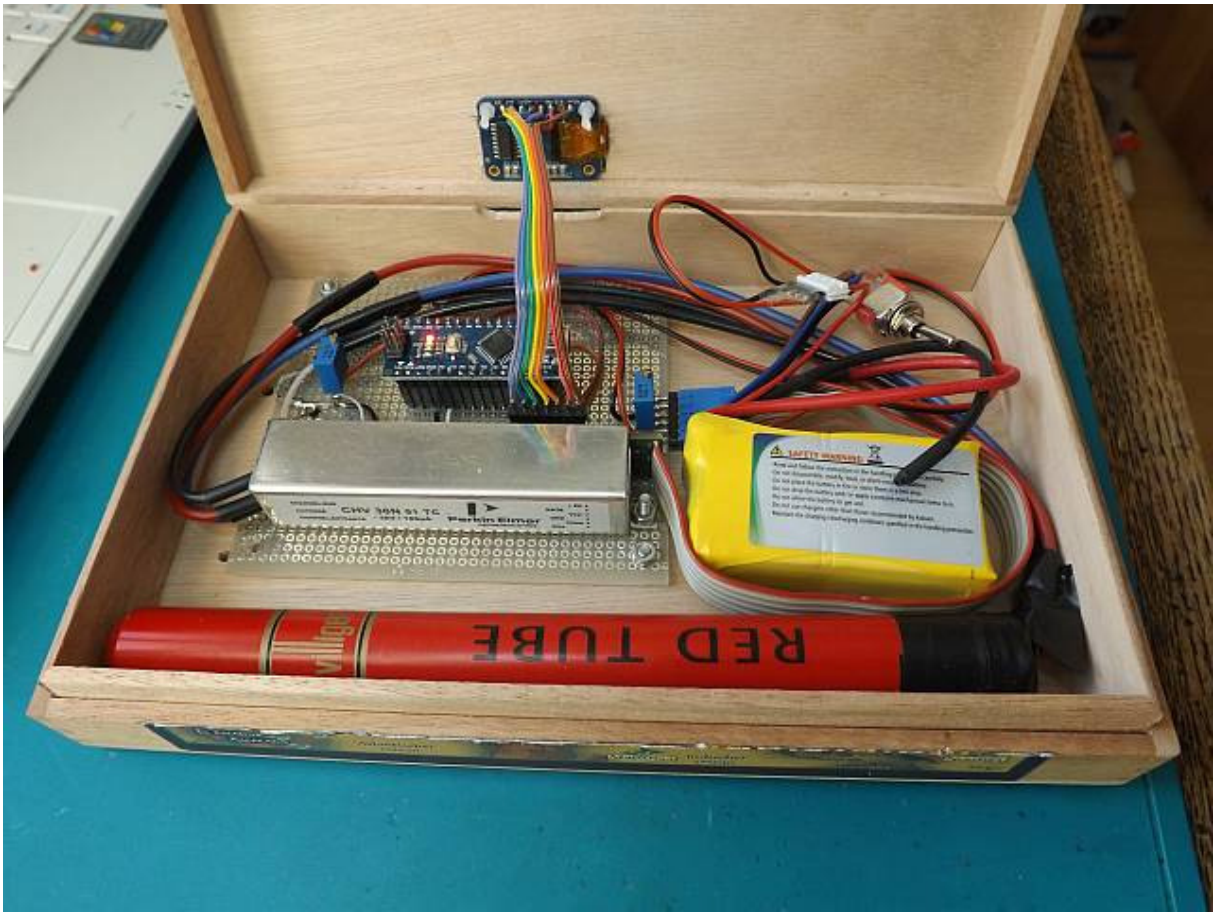

Abb. 5: Holzkiste als Prototypengehäuse, hinter dem Detektor-Rohr links das Hochspannungsmodul und der Arduino Nano, rechts der 3S1P-Flugmodell Akku (11.1V 910mAh)

Bei einer Messung verschiedener Proben stellt man auch wieder den deutlichen Unterschied im Ansprechverhalten zu PIN-Dioden Zählern und auch zu Geiger-Müller-Zählrohren fest. Eine Uranglasur erhöht die Zählraten nur sehr moderat, da sie wenig Gamma-Strahlung emittiert im Vergleich zu einem Uran-Erzstück aus der Natur, das auch alle über die Jahrmillionen gebildeten Zerfallsprodukte enthält, die wie das Bi214 auch ordentliche Gammastrahler sind. Auch an der Entfernungsabhängigkeit merkt man den Unterschied. Befindet man sich beispielsweise auf dem Granitpflaster der Königstrasse sieht man die um etwa den Faktor 3 höhere Pulsrate nur wenig abhängig von der Höhe über dem Pflaster. Ein stark Beta-empfindliches Zählrohr zeigt dagegen im Kontakt mit der Granitoberfläche dagegen deutlich mehr an als in 1m Höhe.

Was am Szintillationszähler gegenüber anderen Instrumenten allerdings sehr angenehm auffällt ist die hohe Zählrate, die sehr stabile Messwerte in kürzester Zeit ergibt. Stellt man die Zählrate über Vset in der Wohnung auf einen Wert von ca. 20 cps ein, sind es draußen auf der grünen Wiese meist etwa nur 10 cps. Bei 10 Sekunden Messzeit hat man dann 100Pulse und bekommt alle 10sek einen Messwert, der nur etwa 10% streut. Das ist mit anderen Detektoren bei gleichem Investment kaum zu schaffen. Will man noch schneller messen bekommt man über ein höheres Vset auch durchaus Raten von über 100cps hin. Die Streuung nimmt jedoch nur mit der Wurzel aus der Zahl der Pulse in der Messzeit ab.

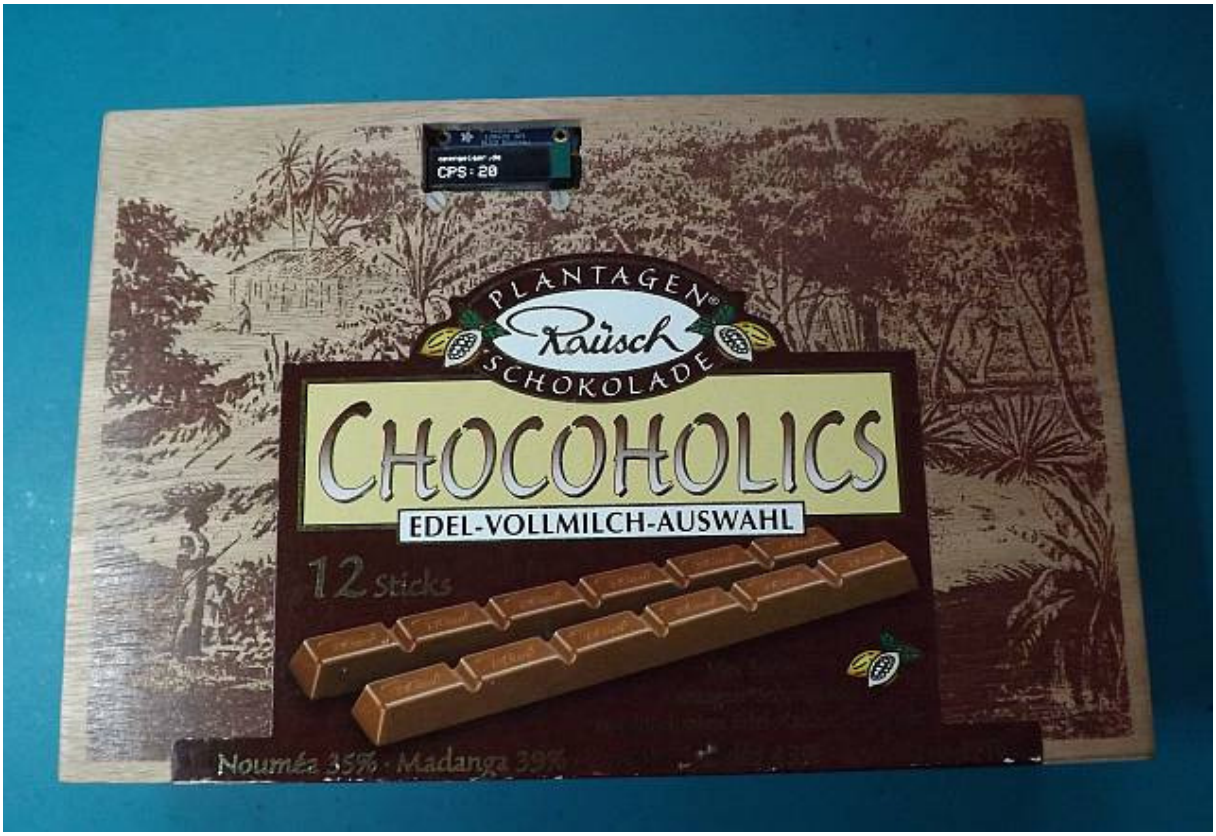

Abb. 6: Unauffälliger Einbau des kleinen OLED Displays in eine dünne Holzkiste

Abb. 7: OLED-Display Pixel mit hoher Leuchtkraft zeigen auch kleinste Zeichen gut lesbar an. Messung auf der Wiese im Schlossgarten (laut Gamma-ODL-Station ca. 0.12uSv/h in 1m Höhe)

Abb. 8: Messung auf dem uranhaltigen Pflasterbelag aus Flossenbürger Granit auf der Königstrasse, wie von anderen Instrumenten bekannt: ein Faktor 3 zum Schlossgarten, Ursache ist der doch merklich radioaktive Granit

Abb. 9: Der strahlungsärmste öffentliche Ort in Stuttgart: die steile Rolltreppe hinunter zur S-Bahn-Station unterm Bahnhof, die Höhenstrahlung ist gut abgeschirmt und man hat so gut wie keine Strahlung vom Boden her

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//OLED
// If using software SPI (the default case):
#define OLED_MOSI   9
#define OLED_CLK   10
#define OLED_DC    11
#define OLED_CS    12
#define OLED_RESET 13
Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);

/* Uncomment this block to use hardware SPI
#define OLED_DC     6
#define OLED_CS     7
#define OLED_RESET  8
Adafruit_SSD1306 display(OLED_DC, OLED_RESET, OLED_CS);
*/

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16
static const unsigned char PROGMEM logo16_glcd_bmp[] =
{ B00000000, B11000000,
  B00000001, B11000000,
  B00000001, B11000000,
  B00000011, B11100000,
  B11110011, B11100000,
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };

#if (SSD1306_LCDHEIGHT != 32)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

#define tInterv 5000
#define CalFactor 1

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

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

  // by default, we'll generate the high voltage from the 3.3v line internally!
(neat!)
  display.begin(SSD1306_SWITCHCAPVCC);
  // Clear the buffer.
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.println("opengeiger.de");
  display.setTextSize(2);
```

```
  display.setCursor(0,15);
  display.println("CPS:");
  display.display();
  attachInterrupt(0, count, FALLING);
}


void loop() {
  unsigned long time;
  unsigned long dt;
  unsigned long rate;

  time = millis();
  dt = time-oldTime;
  if (dt >= tInterv) {
    rate = counter*1000/tInterv;

    display.setTextColor(BLACK);
    display.setCursor(50,15);
    display.print(oldRate);
    display.display();
    display.setTextColor(WHITE);
    display.setCursor(50,15);
    display.print(rate);
    display.display();
    oldRate = rate;
    oldTime = time;
    counter = 0;
  }
}

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

Listing des Progamms für den Arduino Nano zur Bestimmung und Anzeige der Pulsrate (Display-Update Rate 5 Sek.)

Links

Don W. Lake
The Channel Photomultiplier and its Operation
http://archives.sensorsmag.com/articles/0300/38/index.htm

Datenblatt der Perkin Elmer CPM Serie
http://www.datasheetcatalog.com/info_redirect/datasheet/perkinelmer/C983P.pdf.shtml

Operating instructions Perkin Elmer CPMs
http://www.hofoo.com.cn/uploadfiles/cpmoperatinginstruction.pdf

CPM-Hochspannungsmodule Perkin Elmer
www.hofoo.com.cn/uploadfiles/chv30n.pdf

Adafruit Monochrome 128x32 SPI OLED Graphic Display
http://www.exp-tech.de/adafruit-monochrome-128x32-spi-oled-graphic-display? SID=U
https://www.adafruit.com/categories/98