Arduino mit Teviso Sensormodul als Strahlungsdetektor für Radioaktivität

Bernd Laquai 18.5.2014

Jeder, der sich beruflich mit der Entwicklung elektronischer Produkte befasst, kennt das Problem. Heute besteht ein elektronisches Produkt in der Regel immer aus Hardware und Software. Nur wollen die „Hardis“ nichts von der Software wissen und die „Softis“ nichts von der Hardware. Klar, der wahre Könner versteht von Beidem etwas, aber selten gibt er es dann wirklich zu. Nichts desto trotz tun die Komponentenhersteller das Ihre und machen die jeweilig andere Disziplin „transparent“ für den Anwender, wie man so schön sagt. Das heißt, man muss diese Disziplin nun nicht mehr verstehen und kann (fast) ohne Kenntnisse genau das tun, worin man seine „Kernkompetenz“ sieht.

Von diesem Sachverhalt ließ sich auch der schweizerische Komponenten-Hersteller für Strahlungssensoren Teviso (www.teviso.com) leiten und bietet nun einen digitalen Strahlungssensor für Beta- und Gamma Strahlung an, der direkt an den Mikrocontroller angeschlossen werden. Man muss also nichts mehr von der analogen Hardware im Sensor-Modul verstehen, man kann gleich loslegen und in kurzer Zeit ein geeignetes Prgrämmchen schreiben um Radioaktivität bzw. Strahlung ganz komfortabel und einfach detektieren zu können. Das einzige was man noch wissen muss ist, wie man den Detektor an den Mikrocontroller anschließen muss. Von der Strahlungsphysik sollte man allerdings schon auch ein wenig verstehen oder sich zum mindest etwas belesen.

Bei einer Arduino-Mikrocontroller Platine wie z.B. bei dem Uno sind Stiftleisten vorhanden an denen man sowohl die Versorgungsspannung von 5V sowie Masse (Gnd) findet. Diese kann man direkt zur Versorgung des Teviso Moduls mit 5V benutzen. Den digitalen Ausgang des Teviso Moduls legt man zweckmäßigerweise auf den Interrupt-Eingang und zählt die Zählimpulse des Sensormoduls Interrupt-gesteuert, also nicht synchron, da man ja nie weiß wann sie eintreffen. Der Anschluss erfolgt also etwa wie in Abb. 1 .

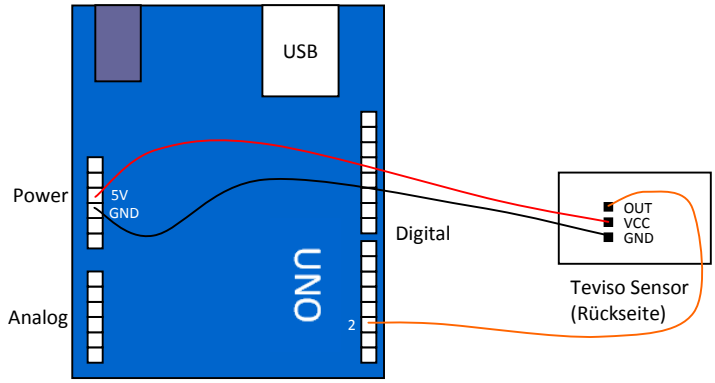

Abb. 1: Anschluss des Teviso Sensor Moduls an den Arduino Uno

Ein ganz einfaches Zählprogramm mit Ausgabe der Dosisleistung in µSv/h auf dem seriellen Monitor könnte dann etwa wie in Abb. 2 aussehen.

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 es ganz sinnvoll man muss nicht noch mal eine Stunde warten um mitzubekommen, dass die Rate auf 10 Pulse pro Minute hochgegangen ist.

```
#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;


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

  attachInterrupt(0, count, RISING);
}

void loop() {
  float sensorValue = rate;
  Serial.println(rate);
  delay(1000);
}

void count()
{
  detachInterrupt(0);
  counter++;
  if (counter == MAXCNT) {
    oldTime = time;
    time = millis();
    dt = time-oldTime;
    rate = (float)MAXCNT*60.0*1000.0/(float)dt/CalFactor;
    Serial.println(rate);
    counter = 0;
  }
  attachInterrupt(0, count, RISING);
}
```

Abb.2: Einfaches Zählprogramm für die Ausgabe der Dosisleistung

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. Diese Variable 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 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 Dosisleistung. Danach muss die Interruptvereinbarung 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.

Man kann die korrekte Funktion dieses Zählprogramms auch ganz leicht mit einem Impulsgenerator testen, der periodische Impulse der Dauer von ca. 200µsec ausgibt und das bei einer Frequenz von zum Beispiel 5.6Hz. Dann benötigen 10 Impulse (MAXCNT=10) nämlich 10*1/5.6Hz = 1.79 Sekunden. Gemessen wird dass vom Arduino in Millisekunden, also ist dt 1790ms. Das Prograämmchen rechnet dann aus dieser Zeit aus, dass die Dosisleistung 10*60*1000/1790/3.4 also ungefähr 100µSv/h = 0.1mSv beträgt. Trägt man für MAXCNT 1 ein hat man also einen kleinen Frequenzmesser, der allerdings die Frequenz (bzw. Impulsrate) in Impulsen pro Minute anzeigt.

Man sieht also, die „Transparenz“ der Hardware hat den großen Vorteil, dass dann die Schnittstelle klar ist und man als „Softi“ in kürzester Zeit einen recht brauchbaren und dank der „schweizerischen Präzision“ zuverlässigen und recht genauen Strahlungsdetektor für Radioaktivität „zusammenklicken“ kann. Diese Transparenz gilt natürlich genauso für das Arduino Board, die Softwarebibliotheken und den Compiler. Denn zugegeben, was sich darin abspielt, das will man ja eigentlich auch nicht unbedingt wissen müssen. Müsste man das in Assembler programmieren würde es nämlich eine geraume Zeit mehr kosten und man müsste sich mit den Registern und der Funktion eines Atmel Mikrocontrollers sehr intensiv auseinandersetzen und das will normalerweise auch der „Softi“ nicht.

So gesehen, wenn also wie in diesem Fall die Software- und Hardwarekomponenten gut funktionieren, dann ist das Teviso Sensor Modul und der Arduino ein wirklich echtes Plug and Play und man freut sich daran.

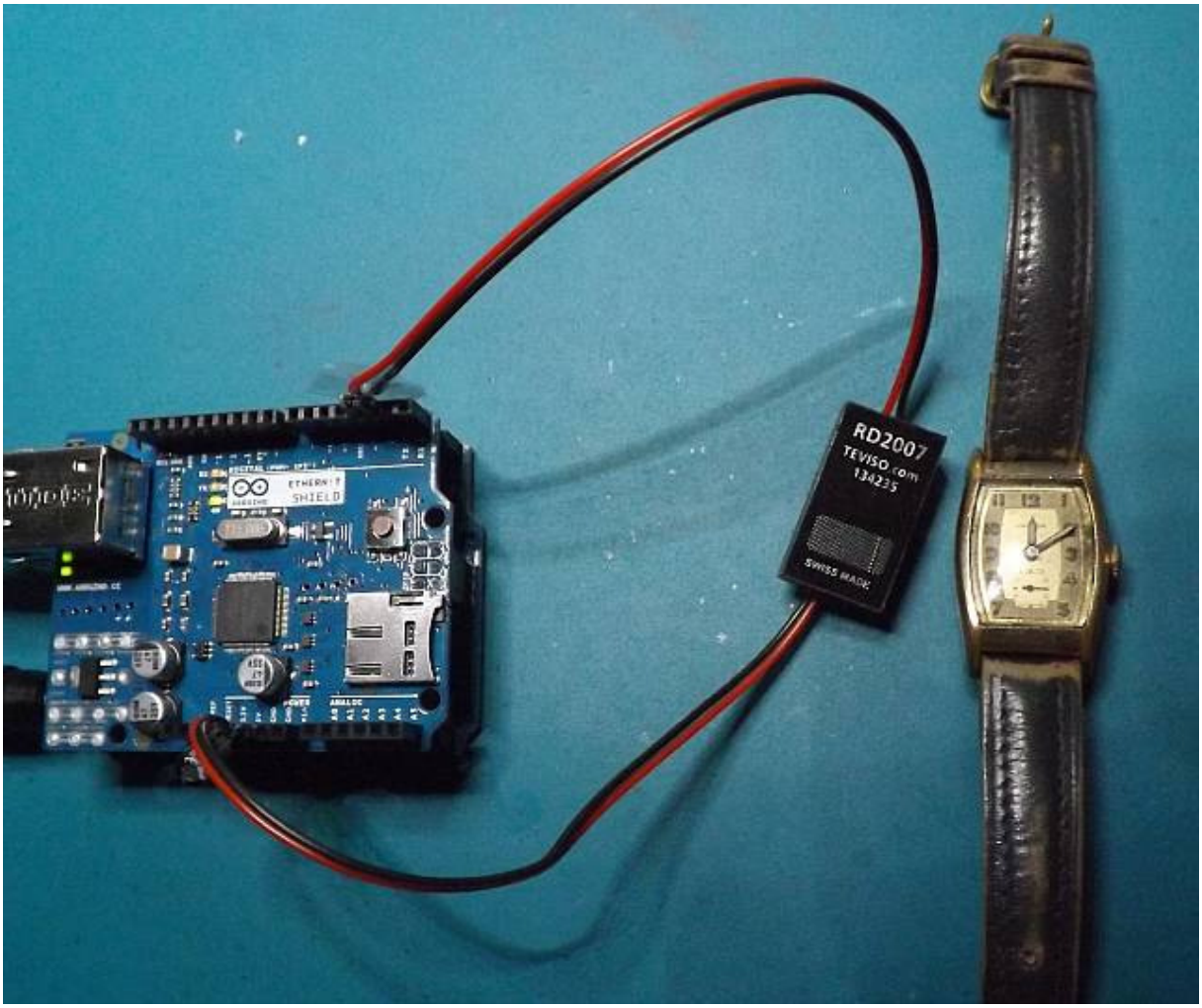

Abb. 3: Der Arduino Uno misst mit einem Teviso Strahlungssensor RD2007 die Radioaktivität einer alten Armbanduhr mit Radium Leuchtziffern

Weitere Literatur

Webseite von Teviso: http://www.teviso.com

Details über das RD2007 Sensormodul: http://opengeiger.de/TevisoModul.pdf

Hanno Krieger
Grundlagen der Strahlungsphysik und des Strahlungsschutzes
Springer Verlag