Arduino Mikrocontroller Interface für Geigerzähler

Bernd Laquai, 18.2.2013

Die Arduino Platform

Mikrocontroller sind aus dem heutigen Leben nicht mehr wegzudenken. Selbst in einer Spülmaschine werden heute die über die Tasten anwählbaren Spülprogramme von einem Mikrocontroller gesteuert. Die heutigen Mikrocontroller sind sogenannte System-On-a-Chip Architekturen, die neben dem eigentlichen Mikrocontroller-Kern auch noch diverse andere Funktionen auf dem Siliziumchip integriert haben, wie zum Beispiel Analog/Digitalwandler und Timer/Counter. In der Regel ist auch der Speicher mitintegriert, ob das nun der nicht-flüchtige Programmspeicher im Flashmemory ist oder ein schneller Arbeitsspeicher als SRAM. Die heutigen Mikrocontroller sind auch im Gegensatz zu den legendären Vorfahren wie dem 8051 oder 6502 um Größenordnungen kleiner, so dass heute deutlich leistungsfähigere Controller in Gehäusen verfügbar sind, die gerade mal 5x5mm groß sind.

Dieser Trend hat aber einen Nachteil. Während sich in den frühen Jahren noch Heerscharen an Hobbyisten mit Mikrocontrollen beschäftigt haben, ist heute die Komplexität so hoch, dass die Latte für den Einstieg schon relativ hoch liegt und nur noch wenige sich in dieses spannende Gebiet der Elektronik trauen.

Es gab allerding vor nicht allzulanger Zeit einen erdrutschartigen Umbruch was den Einstiegsaufwand anbelangt. Massimo Banzi und David Cuarttielles initiierten im Jahre 2005 in Ivreat in Italien das Arduino Projekt (benannta nach Arduin von Ivrea, einem italienischen König). Dieses Lern-Projekt hat zum Ziel eine Mikrocontroller Platform zu schaffen, die aus einer fertigen, professionell hergestellten, scheckkarten-großen Platine und einer modernen Entwicklungsumgebung für die Sprache C++ besteht. Dabei wurden sowohl die Schaltpläne wie der Source-Code der Entwicklungsumgebung als Open-Source Projekt offengelegt. Die Platine hat Steckverbinder, so dass andere Anwendungsplatinen (sogenannte Shields) aufgesteckt werden können wie z.B. LCD-Anzeigen, GPS-Module oder SD-Kartenleser. Zu der Entwicklungsumgebung wurde auch eine kostenlose webseiten-basierte Platform mit diversen Bibliotheken aufgebaut (www.arduino.cc), die Wert darauflegt, den Einstieg so einfach zu machen, dass man als Student oder Hobbyist mit nur wenigen informatischen und elektronischen Grundkenntnissen in wenigen Minuten eine funktionierendes Applikationsprogramm (einen sogenannten Sketch) zusammenstricken kann. Ausserdem ermöglicht die professionelle Herstellung und Bestückung der Hardware-Platine sehr günstige Preise im Bereich von einigen 10 Euro, die man mit eigenen Mitteln im Hobbybereich nicht erreichen kann und von anderen Plattformen auch nicht erreicht wurde.

In der Zwischenzeit gibt es etliche unterschiedliche Arduino-Platinen mit zusätzlichen Features und deutlich höherer Leistung als die Arduino-Uno Basisplatine. Für den Einstieg aber empfiehlt sich immer noch die Uno Platine, die etwa 25Euro kostet.

Geigerzähler und Mikrocontroller

Ein Geigerzähler in seiner ursprünglichen Form zeigt die Radioaktivität dadurch an, dass er die vom Detektor erfasste Rate an Zerfallsakten durch Knackgeräusche in einem Lautsprecher hörbar macht. Bei relativ intensiver radioaktiver Strahlung bekommt man so einen guten Eindruck der Strahlungsintensität. Bei niedrigerer Intensität aber wird der quantitative Eindruck durch das statistisch zufällige Auftreten der Knackimpulse immer schlechter, so dass man im Niedrigdosisbereich allein vom Höreindruck eigentlich nicht mehr sagen kann, welche Probe die höhere Radioaktivität aufweist. Der Weg hier noch zu einer quantitativen Aussage zu kommen ist daher nur die Mittelung über die Zahl der Knackimpulse in einem langen Zählintervall.

Da aber das manuelle Auszählen von Impulsen über längere Zeit mühsam ist, wurden schon früh automatische Zähler verwendet, die ursprünglich auf diskreter Logik beruhten, die mit einer Digital-Anzeige verbunden waren. Heute werden für diesen Zweck Mikrocontroller eingesetzt, die für diesen Zweck ideal geeignet sind.

Da Mikrocontroller normalerweise ihre Programme in einem festen Takt abarbeiten, die Zählimpulse vom Detektor aber zeitlich zufällig eintreffen, bietet es sich an, eine spezielle Zusatzfunktion im Mikrocontroller zu verwenden: die Interrupt-Steuerung. Unter einem Interrupt versteht man eine temporäre Programunterbrechung mit der der Mikrocontroller auf ein unvorhersehbares Signal (ein Interrupt-Request, IRQ) reagieren kann. Sobald die Reaktion auf den Interrupt in einer speziellen Interrupt-Behandlungsroutine (Interrupt Service Routine, ISR) abgearbeitet ist, fährt der Controller mit der normalen Programm-Abarbeitung fort. Man kann also für eine Geigerzähler Anwendung in der Interrupt Service Routine einfach einen Zähler hochzählen und den Mikrocontroller im Hauptprogramm in einer Schleife laufen lassen, bis ein bestimmtes Messintervall abgelaufen ist. Nach Ablauf dieses Intervalls gibt man den Quotienten aus Zählimpulsen und Messintervalldauer als momentane Zählrate auf ein LCD Display aus oder speichert den Messwert auf einem SD-Kartenleser-Shield ab um die Daten später mit dem PC auszuwerten.

Eine andere Variante eines Geigerzähler-Interfaces wäre, einen vom Controller-Kern unabhängigen Zähler ohne Software Kontrolle direkt vom Detektor her anzusteuern und diesen in regelmäßigen Intervallen vom Mikrocontroller her auszulesen und zurückzusetzen. Das erfordert zwar einen zusätzlichen Aufwand um den separaten Zähler zu kontrollieren, könnte aber einerseits stromsparender sein, da man den Mikrocontroller-Kern dann während den Zählintervallen schlafen legen kann und andererseits könnte man höhere Zählraten verarbeiten, da ein separater Hardwarezähler mit digitaler Logik natürlich schneller zählen kann als der software-kontrollierte Mikrocontroller-Kern, der mit einer speziellen Routine auf die Interrupts reagieren muss. Inwieweit dazu der in den Arduino Controllern vorhandenen interne Counter verwendbar wäre, muss noch geprüft werden. Hier sei zunächst einmal nur die einfache Interruptsteuerung beschrieben, da sie sehr wenig Aufwand bedeutet und für den Einstieg leichter zu verstehen ist.

Die Interface Hardware

Viele Geigerzähler geben genau wie das Stuttgarter Geigerle ein akustisches Signal aus. Typischerweise ist dies für einen Lautsprecher gedacht und kapazitiv abgekoppelt, d.h. es enthält keinen Gleichanteil. Daher ist dies eine Schnittstelle die man gut einschätzen kann und auf die man ein Mikrocontroller-Interface zuschneiden kann. Da sich auch die Interrupt Eingänge der Mikrocontroller stark ähneln, kann das Interface sehr universell gestaltet werden. So zeigte sich, dass das hier vorgestellte Interface, das ursprünglich für die C-Control Mikrocontroller Plattform der Firma Conrad (HC908 Mikrocontroller) entwickelt wurde, ohne Änderung genauso auch für den Arduino benutzt werden kann.

Der Atmel Atmega328 auf dem Arduino Uno Board arbeitet ebenfalls mit einer 5V Versorgungsspannung und die Interrupt-Logik kann auf verschiedene Signalflanke eingestellt werden. Da beim Stuttgarter Geigerle die erste Flanke eine fallende Flanke ist, wurde beim Arduino der Flankentyp „FALLING“ benutzt. Allerdings ist die Signalstärke am Ohrhörer-Ausgang noch etwas zu schwach um den Interrupt-Eingang sicher zu triggern. Deswegen muss ein weiterer Verstärker benutzt werden, der auch die Aufgabe hat, die unterschiedlichen Betriebsspannungs-Niveaus auszugleichen.

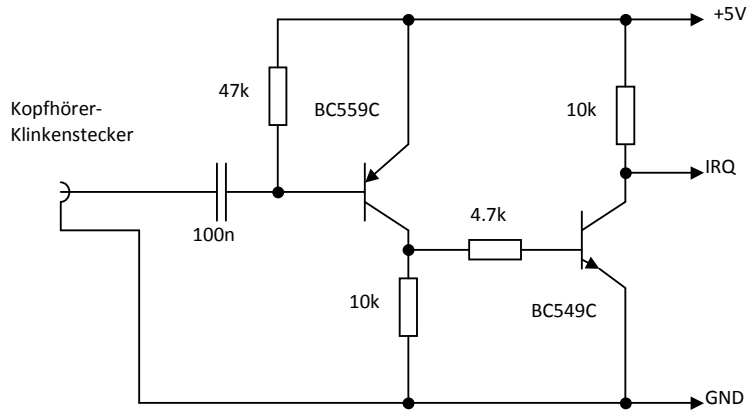

Abb. 1a: Schaltplan Mikrocontroller Interface

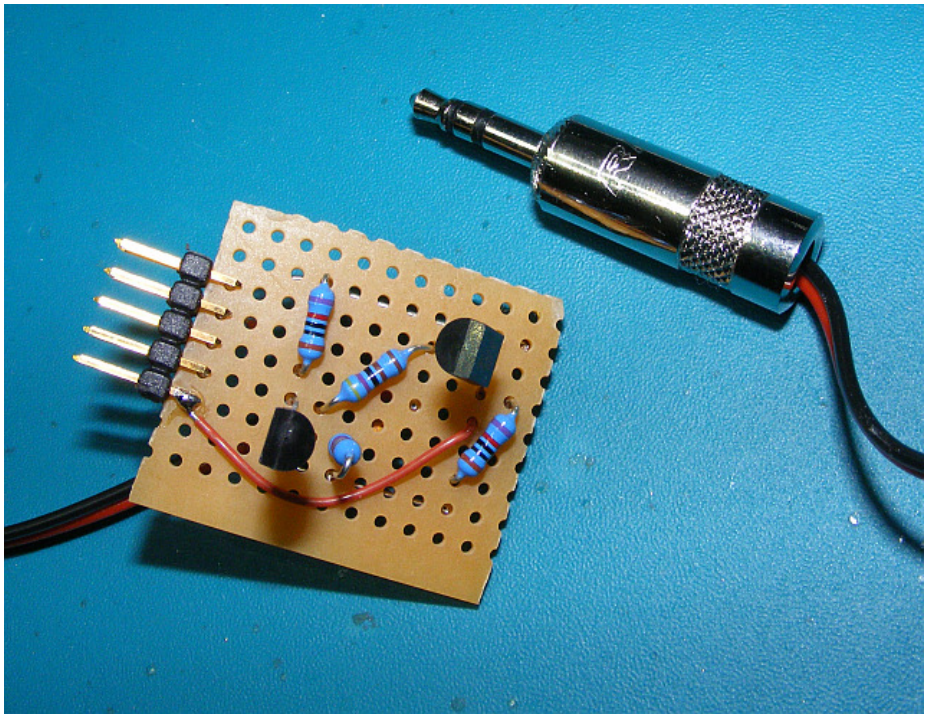

Abb. 1b: Platine für das Mikrocontroller Interface

Das Eingangssignal, welches ein negativer Puls ist, wird zunächst kapazitiv angekoppelt, wobei die Größe der Kapazität die Stärke der Basisstromänderung am ersten PNP-Transistor einstellt. Im Ruhezustand ist die Basis-Emitter Spannung 0 und der PNP-Transistor sperrt. Im Falle eines negativen Pulses fließt ein Basisstrom, der einen verstärkten Kollektorstrom durch den 10kOhm Widerstand zur Folge hat. Damit steigt die Spannung am Kollektor an, was wiederum einen Basisstrom im NPN-Transistor zur Folge hat, der normalerweise gesperrt ist. Der NPN-Transistor verstärkt diesen bereits verstärkten Basisstrom nochmals, so dass nun der NPN-Transistor voll durchschaltet und in die Sättigung geht. Damit bewegt sich die Kollektorspannung am NPN Transistor von +5V auf nahe 0V und erzeugt damit eine steile, fallende Flanke am Interrupt-Eingang und löst den Interrupt-Request sicher aus.

Diese Schaltung kann bequem auf eine Lochraster-Platine aufgelötet werden, welche mit einem Pfostenstecker versehen werden kann und als Break-Out auf das Arduino Board aufgesteckt werden kann. Es muss dabei beachtet werden, dass als Interrupt Eingang 1 beim Uno Board der Digital Pin 2 verwendet wird. Die 5V Spannungsversorgung liegt auf dem Pfostenstecker auf der anderen Seite des Boards (daneben liegt Ground), so dass im Falle eines Break-Outs (im Gegensatz zu einem Shield) noch ein kurzes Kabel auf die andere Seite gelegt werden muss.

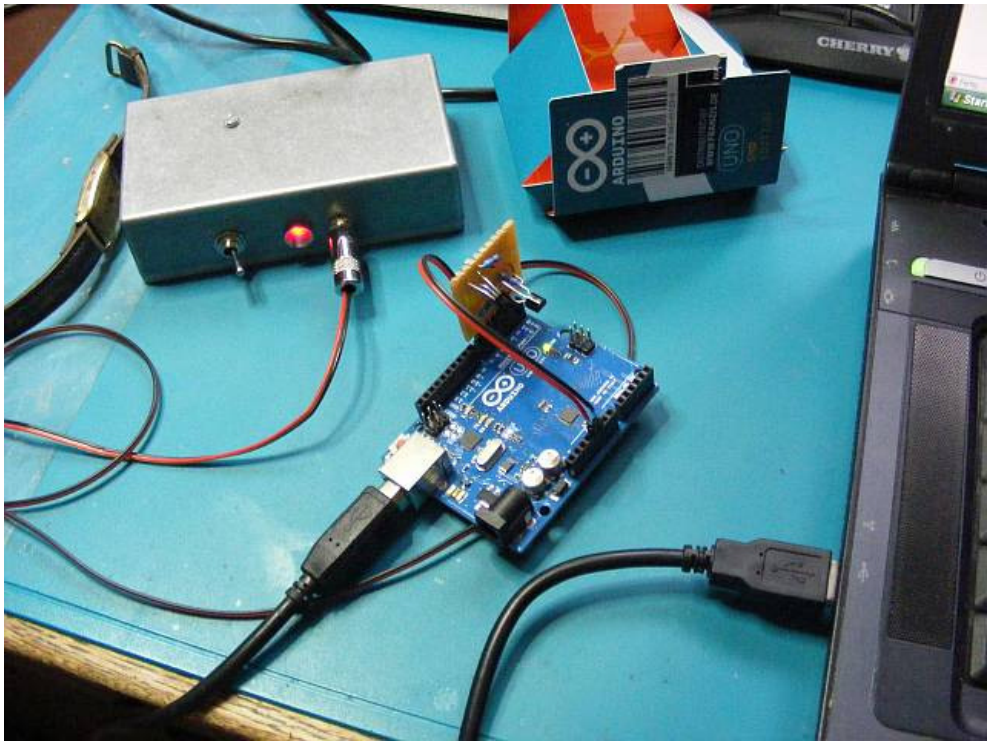

Abb. 2: Stuttgarter Geigerle PIN-Dioden Zähler mit Arduino Interface

Software

Die einfachste Software um das Interface für einen Geigerzähler in Betrieb zu nehmen ist das Blink-Beispiel zu der Funktion attachInterrupt() aus der Referenz Webseite:

http://arduino.cc/en/Reference/AttachInterrupt

Das Einzige was an dieser Routine noch verändert werden muss ist der Flankentyp. Er muss von CHANGE auf FALLING geändert werden, da die Zählimpulse so kurz sind und eine steigende Flanke in weniger als 1µs auf die fallende Flanke folgt, dass man die LED nicht blinken sehen würde. Damit sieht dann ein Gegerzähler-Sketch etwa so aus:

```
int pin = 13;
volatile int state = LOW;

void setup()
{
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, FALLING);
}

void loop()
```

```
{
  digitalWrite(pin, state);
}

void blink()
{
  state = !state;
}
```

Die Variable state muss als volatile erklärt werden, da sie global vereinbart wird und in der Interrupt Service Routine verändert werden muss. Mit attachInterrupt() wird dem Compiler erklärt, welches die Interrupt Service Routine ist, auf die reagiert werden soll, wenn an pin 2 eine fallende Flanke auftritt. Das Hauptprogramm arbeitet lediglich eine kurze Endlosschleife ab, in der laufend der Pegel der LED entsprechend der Variablen state ausgegeben wird. Tritt der Interrupt auf, dann kehrt sich der logische Wert der state Variablen um und dementsprechend wechselt auch die LED am Pin 13 ihren Zustand. Damit entspricht das Blinken der LED auf dem UNO Board dem Blinken der LED Anzeige am Geigerle mit dem Unterschied, dass die UNO LED an- und wieder ausschaltet während die Geigerle LED bei jedem Zustandswechsel die Farbe ändert. Daher ist der Vergleich beider LEDs ein guter Test ob das Interface richtig funktioniert.

Wenn dieser einfache Sketch richtig läuft können entsprechend komplexere Sketches entwickelt werden, um bestimmte Zähl- und Auswertefunktionen zu implementieren oder um z.B. die Anzeige der gemessenen Radioaktivität auf einem LCD-Display zu erreichen. Das unterscheidet sich dann nicht mehr von normalen Arduino-Anwendungen für die es unzählige Beispiele in der inzwischen riesigen Arduino Open-Source Gemeinde gibt.