Zählrohrbasierter Kontaminationsdetektor mit hoher Empfindlichkeit und Data-Logging für geo-referenzierende Messungen mit hoher Ortsauflösung

Bernd Laquai, 8.11.14

Wenn eine möglicherweise radioaktive Probe zur Messung vorliegt, ist es oft möglich Detektoren mit niedrigerer Zählrate, die z.B. eine besonders geringe Abhängigkeit der Zählrate von der Energie aufweisen und deswegen auch genauer sind, zu verwenden und die niedrige Zählrate mit einer längeren Messzeit zu kompensieren. Dabei sind durchaus Messungen, die sich über Stunden erstrecken möglich, da bei konstanter Messanordnung und langlebigen Radionukliden nicht mit dynamisch veränderlichen Vorgängen zu rechnen ist. Wenn es dagegen gilt, Kontaminationen oder natürliche Strahlung in der Natur aufzuspüren, die im Niedrigdosisbereich liegen, ist ein Ansatz mit solchen Detektoren wenig hilfreich. In so einem Fall will man mit möglichst hoher Geschwindigkeit ein großes Gebiet ausmessen und ist dabei in der Regel durch die zur Verfügung stehende Messzeit beschränkt. Der ermittelte Messwert für die Gamma-Ortsdosisleistung, der von der Zahl der Impulse abhängt, die zur statistischen Mittelung verwendet wird, weist bekanntlich eine Streuung auf, die sich aus dem Quotienten der tatsächlichen Zählrate und der Wurzel aus der verwendeten Anzahl an Pulsen ergibt. Aus diesem Grund sollte der Detektor für diesen Einsatz eine möglichst hohe Zählrate aufweisen. Ist dies der Fall, dann ergibt sich auch eine höhere Ortsauflösung, denn bei einer gegebenen Geschwindigkeit, entfallen die verwendeten Zählimpulse auf eine kürzere räumliche Distanz entlang des zurückgelegten Weges. Gibt man umgekehrt die Zahl der Zählimpulse vor, und damit die akzeptierte Streuung, bedeutet die höhere Zählrate eine höhere mögliche Geschwindigkeit mit der ein Gebiet vermessen werden kann. Mathematisch ausgedrückt gilt:

$v=\Delta s/\Delta t$

$r=counts/\Delta t$

$\sigma(r) = r/\sqrt{(counts)}$

Unter der Annahme man will die Rate mit einer gewissen zulässigen Streuung σ bestimmen, liegt bei einer von der Kontamination abhängigen Zählrate r die Zahl für counts fest. Für eine Streuung von 10% muss also der Wert für counts 100 betragen, d.h. man sollte 100 Zählimpulse für die Bestimmung der Zählrate abwarten. Wenn also der Detektor die Zählrate r hat liegt somit auch Δt fest:

$\Delta t = counts/r$

und damit ergibt sich die mögliche Geschwindigkeit aus der gewünschten Ortsauflösung oder eben die Ortsauflösung Δs aus einer vorgegeben Geschwindigkeit v gemäß:

$\Delta s = v * \Delta t = v * counts/r$

In die Praxis umgesetzt bedeutet das beispielsweise, dass wenn eine Gebiet zu Fuß vermessen werden soll, man eine Bewegungsgeschwindigkeit von 5km/h = 1.39m/s annehmen kann. Ein PIN-Dioden basierter Detektor wie das Teviso Modul RD2007 erreicht

bei einer Gamma-Ortsdosisleistung von 1µSv/h eine Zählrate von 3.4cpm = 0.0567counts/s. Damit ergibt sich für eine noch tolerierte 10% Streuung eine Ortsauflösung von:
Δs = 1.39m/s*100counts / 0.0567 counts/s = 2451m

Das wäre also eine äußerst schlechte Ortsauflösung. Ein SBM-20 Zählrohr dagegen liefert bei 1 µSv/h etwa 190cpm = 3.17counts/s. Damit ergäbe sich bei Schrittgeschwindigkeit eine Ortsauflösung nach der gleichen Rechnung von 43.4m, was schon deutlich besser wäre. Um nun in den Bereich von wenigen Metern zu kommen könnte man also beispielsweise die Zählpulse von 4 SBM-20 Zählrohren auswerten, dann läge man bei knapp 769cpm und etwa 10.8m Ortsauflösung, was dann schon recht brauchbar ist.

Nun findet man im Internet durchaus Schaltungsbeispiele, wo 4 SBM-20 Zählrohre einfach parallel an eine Hochspannungsquelle angeklemmt sind und von einem Zählrohrverstärker ausgewertet werden. Das funktioniert durchaus und ergibt sicher auch grob die Vierfache Rate eines einzelnen Zählrohrs. Wenn nun aber stromsparende Geigerzählermodule eingesetzt werden, muss man damit rechnen, dass die Hochspannungsquelle „in die Knie“ geht, wenn man z.B. statt einem Zählrohr gleich 4 Rohre anschließt, die dann viermal soviel Strom entnehmen würden.

Wenn fertige Geigerzählermodule vorliegen und die Zählrohrverstärker digitale Ausgänge haben, dann lassen sich separat erzeugten Pulse aus mehreren Modulen aber auch über eine digitale Logik aufsummieren. Verwendet man beispielsweise die SBM-20 Driver Module von 4N-Galaxy, dann hat man bei so einer Lösung eine separierte Hochspannung und separate Zählverstärker, so dass die Logik die Zählimpulse von 4 voneinander völlig unabhängigen Modulen aufsummiert, was natürlich die 4-fachen Kosten erzeugt, aber eine schnelle Lösung ist um sicherzustellen, dass die Zählrohre sicher in den gegebenen Spezifikationen betrieben werden.

Solange die digitalen Zählpulse sehr kurz sind, d.h. in der Größenordnung kleiner 100µs liegen, ist es möglich eine einfache Summation dadurch zu erreichen, dass man die digitalen Signale im logischen Sinne „ODER“ verknüpft. Dann entsteht ein Ausgangspuls wenn einer der Zähleingänge logisch „wahr“ wird. Das ist aber nur solange richtig, wenn man eine positive Logik vorliegen hat, d.h. logisch „wahr“ ist die 1 oder elektrisch 5V und logisch „falsch“ ist die 0 mit 0V Pegel. Wenn ein Zählmodul wie das SBM-20 Driver Modul eine negative Logik hat, d.h. normalerweise auf 5V Pegel liegt und für einen Zählimpuls kurz den 0V Pegel annimmt, dann benötigt man eine AND oder NAND Funktion des logischen Gatters. D.h. der Ausgangspegel bei einem AND ist nur dann auf 5V wenn keiner der Module auf 0V liegt. Das NAND dreht dann lediglich den logischen Sinn des Ausgangs um. Es wird also ein NAND-Gatter mit 4 Eingängen benötigt um die digitalen Zählpulse von 4 Zählmodulen zu summieren und die Summe als positive Pulse auszugeben.

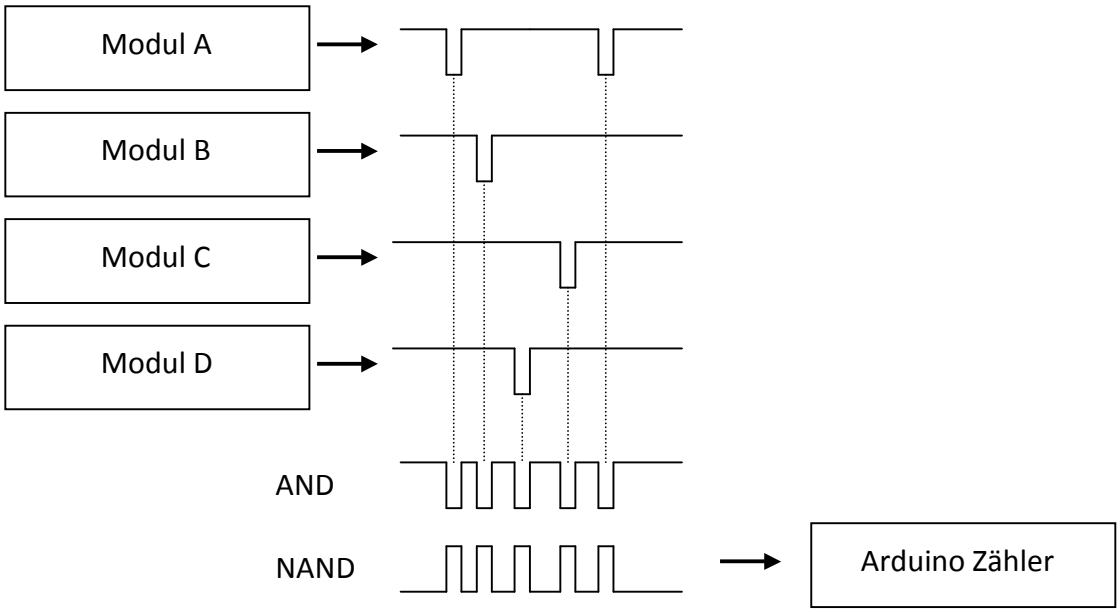

Abb. 1: Konzept der digitalen Summation von Zählpulsen aus 4 unabhängigen Geigerzähler Modulen vom Typ SBM-20 Driver

Sobald das Summensignal als digitales Signal vorliegt, kann man für die Impulszählung und die Umrechnung der Zählrate in eine Gamma-Ortsdosisleistung wie bei der universellen Zählerschaltung mit dem Arduino vorgehen. D.h. man legt das Summensignal auf den Interrupteingang und misst die Zeit, die benötigt wird bis eine gewisse Anzahl Zählpulse eingetroffen ist. Daraus berechnet sich die Rate, die dann durch Multiplikation mit einem Konversionsfaktor in eine Dosisleistung in µSv/h umgerechnet werden kann.

Wenn große Gebiete untersucht werden sollen, dann ist das Data-Logging eine sehr wesentliche Eigenschaft, die ein solcher Kontaminationsdetektor haben sollte. Wenn allerdings nur die registrierten Zählraten oder die berechneten Dosisleistungen auf ein Speichermedium geschrieben werden, dann ist das ebenfalls noch wenig hilfreich, da es bei der Auswertung schwer ist einen Ortsbezug herzustellen. Für die Herstellung eines Bezugs zu Geokoordinaten gibt es zwei Lösungsmöglichkeiten. Der erste ist der Einsatz eines GPS-Moduls das ermöglicht zusammen mit den gemessenen Werten gleichzeitig auch die GPS-Koordinaten auf das Speichermedium z.B. eine SD-Karte zu schreiben. Falls ein separates GPS-Gerät (z.B. ein Garmin Handheld Gerät) parallel eine Track-Aufzeichnung macht, ist auch ein zweiter Ansatz möglich, indem ein sehr stromsparendes Real-Time-Clock (RTC) Modul hinzugefügt wird, das mit jeder Messung ausgelesen wird. Der zurückgelieferte Zeitstempel wird dann zusammen mit den Messdaten abgespeichert. Da bei der GPS-Trackaufzeichnung die GPS-Koordinaten ebenfalls zusammen mit einem Zeitstempel aufzeichnet werden, können beide Aufzeichnungen später anhand des Zeitstempels synchronisiert werden. Auch Fotos, die einen Zeitstempel enthalten können so den Messdaten eines bestimmten Ortes im Nachhinein noch zugeordnet werden. Die Verwendung einer Real-Time-Clock soll deswegen hier als Lösungsmöglichkeit vorgestellt werden.

RTC's gibt es heute als single Chip Lösungen, die nur wenig Beschaltung benötigen und mit einem seriellen I2C-Bus Interface ausgestattet sind. Ein sehr viel verwendeter Baustein dieser Art ist der DS1307 von Dallas. Für ihn gibt es beispielsweise von Adafruit einen kleinen

sogenannten „Breakout“ Bausatz für den Arduino, den man in kurzer Zeit zusammenlöten kann. Das Breakout Board wird dann mit nur 4 Pins am Arduino Board angeschlossen. Außerdem gibt es für den DS1307 eine fertige Bibliothek (RTClib) , so dass man sich um das Kommunikationsprotokoll nicht kümmern muss.

Sobald die kleine Knopfzelle, die dem Erhalt der internen Clock Daten dient, eingesetzt wird, kann mit einem einmaligen Aufruf einer Adjust-Routine die Uhr gestellt werden. Die Uhr läuft dann auch ohne Stromversorgung des Breakout-Boards mit der richtigen Uhrzeit weiter, bis die Knopfzelle wieder entfernt wird. Um das Modul auslesen zu können, muss allerdings das Modul von außen zusätzlich zur Knopfzelle mit Strom versorgt werden.

Die Verwendung der RTC Bibliothek erfordert zunächst, dass man eine Objektvariable vom Typ RTC_DS1307 generiert. Über diese kann man nun auf die zugehörigen Funktionen (Methoden) zugreifen. In dem Sketch folgt zunächst die Deklaration einiger globaler Variablen. So wird zum Beispiel auf der SD Karte eine Datei mit dem Namen „datalog.txt“ angelegt, sofern sie noch nicht existiert. Die Variable, unter der auf die Datei zugegriffen wird, ist vom Typ File. Es wurde zusätzlich noch eine rote LED für den Digitalpin D7 vorgesehen, die anzeigt, wenn es zu Schreibfehlern auf die SD Karte kommt (z.B. wenn sie nicht richtig eingesteckt ist oder die Betriebsspannung nicht stimmt). Für den Fall dass ein serieller Monitor angeschlossen ist gibt der Sketch die Logdaten auch auf den Monitor aus. Dazu wird in der Setup Routine zunächst der Serial Monitor mit 9600 Baud gestartet. Danach wird die rote LED auf 0V initialisiert. Dann folgt ein Block in dem getestet wird, ob die SD-Karte schreibbereit ist und ob die Datei mit dem Namen datalog.txt bereits existiert. Falls ja, wird eine Trennmarkierung ausgegeben und die Daten werden einfach nur angehängt. Andernfalls wird die Datei neu angelegt. Die Kommunikation mit der Real Time Clock erfolgt mit Hilfe der Wire Bibliothek für das I2C-Interface. Es wird auch zunächst getestet, ob die Clock schon gestellt ist und läuft. Wenn nicht, geht auch für diesen Fall die rote LED an. Für diesen Fall muss die Code-Zeile durch Entfernen des Kommentars aktiviert werden, mit der die Clock gestellt wird (rtc.adjust ...).

Die periodisch wiederholte Loop() Routine läuft wie beim universellen Zählprogramm ab, außer, dass bei jedem Durchlauf ein Objekt vom Type DateTime generiert wird, aus dem durch Aufruf der Stunden-, Minuten- und Sekunden-Funktion ein Zeitstring zusammengebaut wird, der als Zeitstempel zusammen mit dem Wert für die Dosisrate sowohl auf den seriellen Monitor wie auch auf die SD Karte ausgegeben wird. Das ist dann schon alles wesentliche an dem Sketch.

```
#include <Wire.h>
#include "RTClib.h"
#include <SD.h>
#include <SPI.h>

#define MAXCNT 10
#define CalFactor 700
// =4*1/0.0057 für SBM-20

RTC_DS1307 rtc;

char fileName[15] = "datalog.txt";
File myFile;
int redLed = 7;

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

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

  pinMode(10, OUTPUT);
  pinMode(redLed, OUTPUT);
  digitalWrite(redLed, LOW);
  if (!SD.begin(10)) {
    Serial.println("SDcard not ready\n");
    digitalWrite(redLed, HIGH);
    return;
  }
  if (!SD.exists(fileName)) {
    myFile = SD.open(fileName, FILE_WRITE);
    myFile.println("###");
    myFile.flush();
  }
  else {
    myFile = SD.open(fileName, FILE_WRITE);
    myFile.println("-----");
    myFile.flush();
  }

  Wire.begin();
  rtc.begin();
  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    digitalWrite(redLed, HIGH);
    // following line sets the RTC to the date & time this sketch was
compiled
    //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to
set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }

  Serial.println("Starting...");
  attachInterrupt(0, count, FALLING);
}

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

```
  if (counter == MAXCNT) {
    detachInterrupt(0);
    time = millis();
    dt = time-oldTime;
    rate = (float)MAXCNT*60.0*1000.0/(float)dt/CalFactor;

    DateTime now = rtc.now();
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.print(' ');
    Serial.println(rate);

    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    myFile.print(now.second(), DEC);
    myFile.print(' ');
    myFile.println(rate);
    myFile.flush();

    oldTime = time;
    counter = 0;
    attachInterrupt(0, count, FALLING);
  }
}

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

Listing 1: Zählprogramm für den Arduino mit Auslesen des Real Time Clock Moduls und Schreiben (Loggen) auf SD-Karte

Abb. 2 zeigt einen Prototypen des Kontaminationsdetektors, der in einer einfachen unauffälligen Pappkiste untergebracht ist (um bei der Messung kein Aufsehen zu erregen) und von einem Lithium-Polymer Akku mit 5Ah gespeist wird. Die Zählmodule zusammen mit dem Arduino, dem RTC-Modul und dem SD-Karten-Shield haben zusammen eine Stromaufnahme von etwa 58mA, so dass man auch längere Strecken aufzeichnen kann (mehrere Tage Akkulaufzeit).

Um das Gerät auszutesten und das Konzept zu validieren wurde mit dem Sportplatz in Kleinnaundorf bei Dresden (N50 59.871 E13 41.080) ein nahezu ideales Testgelände gefunden. Beim Bau dieses Fussballplatzes wurde teilweise Haldenmaterial aus dem lokalen Uranbergbau verwendet. Am Südwestlichen Eck ist dort eine deutliche Kontamination in Form eines Hotspot mit einer Gammadosisleistung von ca. 1µSv/h zu sehen. Abb. 3 zeigt den Messweg, der vom GPS-Empfänger (Garmin) aufgezeichnet wurde und dessen Ortsauflösung zu dem Zeitpunkt etwa 5m betrug. Abb. 4 zeigt eine georeferenzierung der Messdaten anhand des Zeitstempels zum Gammascout, der eine so niedere Zählrate hat, dass erst ab einem Mittelungsintervall von 2min, das statistische Rauschen bei der normalen Nullrate von 0.15uSv/h einigermaßen erträglich bleibt. Es wird klar erkennbar, dass anhand des Zeitstempels die geringe Zahl an Messpunkten auch nur zu wenigen Trackpunkten

zugeordnet werden kann. Damit wird die Ortsauflösung schlecht und man kann den Mäanderförmigen Weg im Prinzip nicht mehr erkennen. Mann kann auch im Profil entlang des Messwegs, das vom Gammascout aufgezeichnet wurde erkennen, dass bei der normalen Gehgeschwindigkeit deutlich weniger Dosisleistungs-Messpunkte aufgezeichnet wurden als vom Garmin Trackpunkte mit den Koordinaten.

Ganz anders sieht das nun mit dem Kontaminationszähler. Dieser erzeugt nun deutlich mehr Dosisleistungs-Messwerte als das GPS Trackpunkte. Damit ist nun die Ortsauflösung der geo-referenzierten Messung identisch mit der Auflösung des GPS-Geräts und liegt bei etwa 5m und ist nicht mehr durch das Strahlungsdetektor bestimmt. Die hohe Messwertdichte wie auch im Profil entlang des Messwegs deutlich. Damit bilden sich nun auch die gelaufenen Mäandern sehr präzise ab (Abb. 6 und 7).

Abb. 2: Der Kontaminationszähler untergebracht in einer einfachen Pappkiste als Prototyp

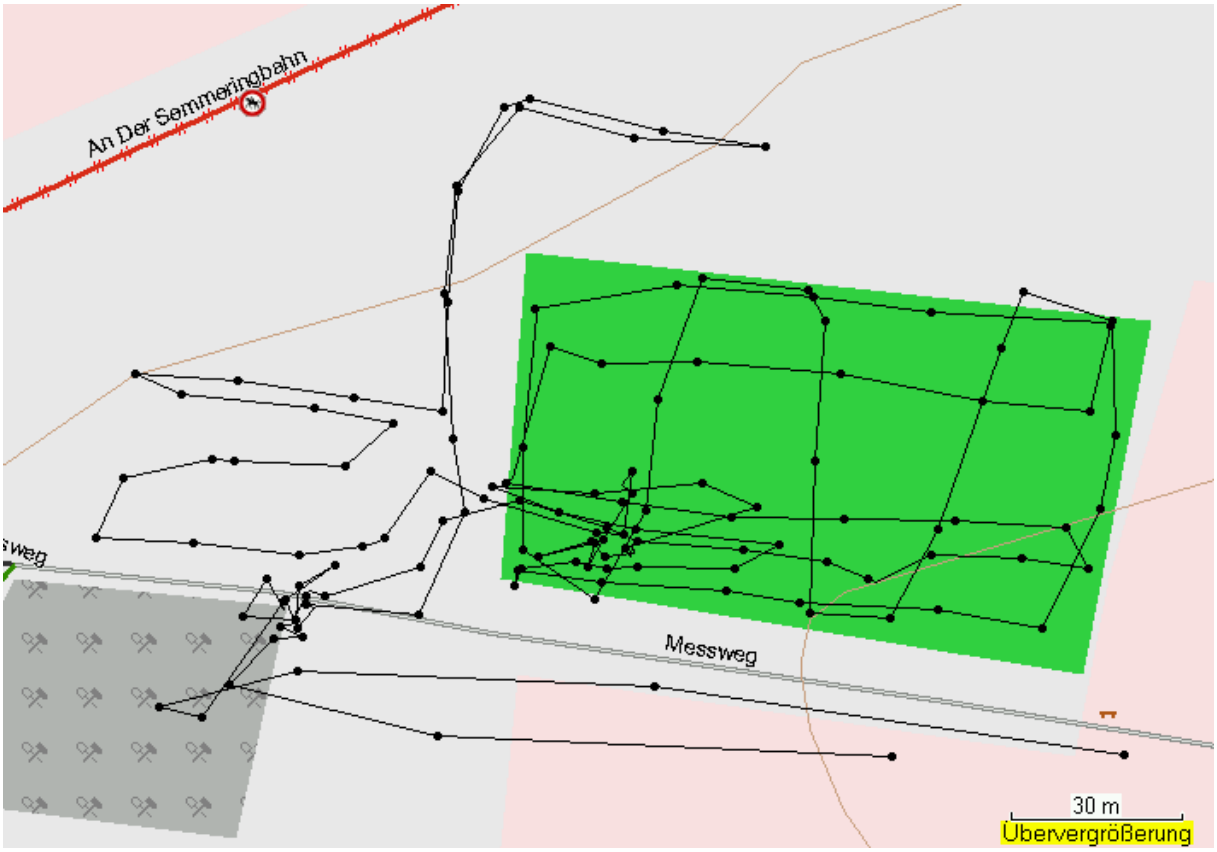

Abb. 3: Der vom GPS-Empfänger aufgezeichnete Messweg (Track), deutlich sind die anfangs gelaufenen Mäandern zu erkennen, später wurde der Messweg um den Hotspot verdichtet

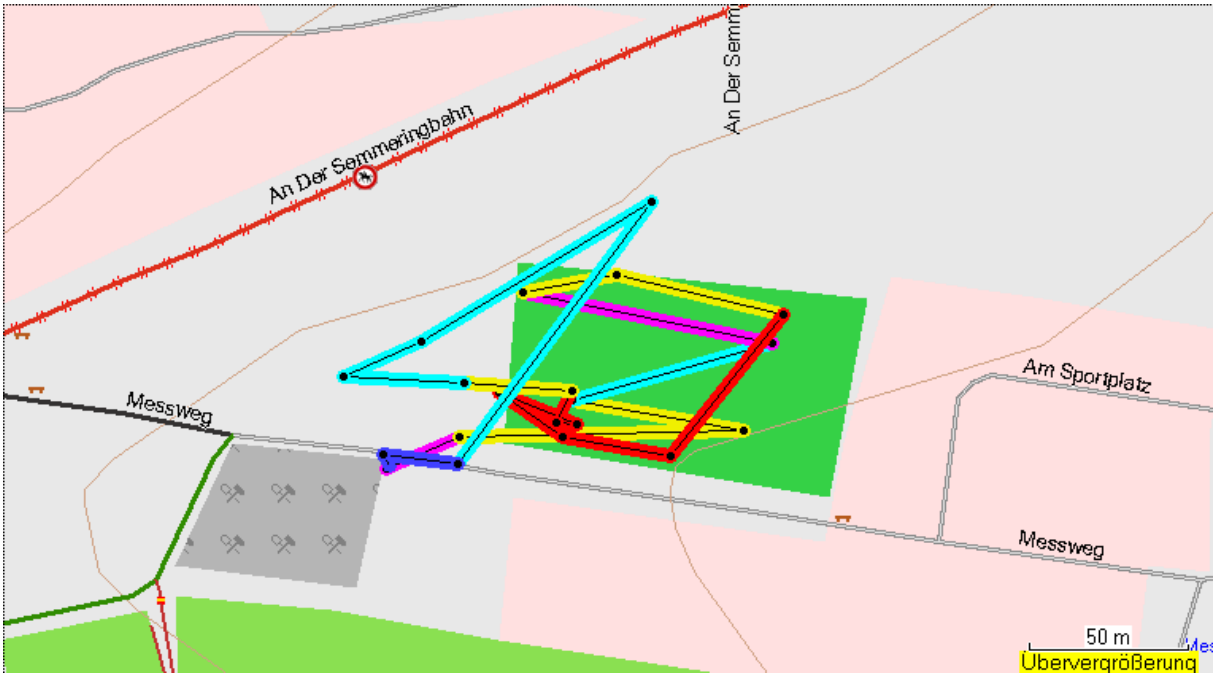

Abb. 4: Konventionelle geo-referenzierte Messung mit dem Gammascout und einem Mittelungsintervall von 2 Minuten. Dabei bedeutet rot >0.5uSv/h, magenta>0.4uSv/h, gelb>0.3uSv/h, türkis>0.2uSv/h, blau>0.1uSv/h

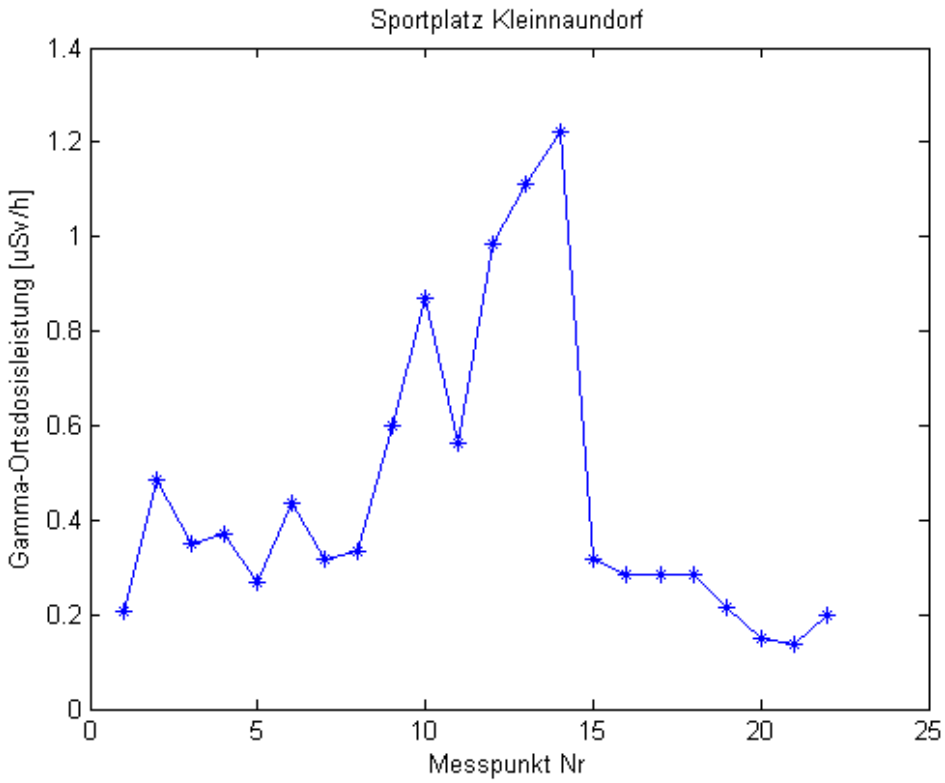

Abb. 5: Profil des Messwegs aufgezeichnet vom Gammascout im 2 Minuten Raster

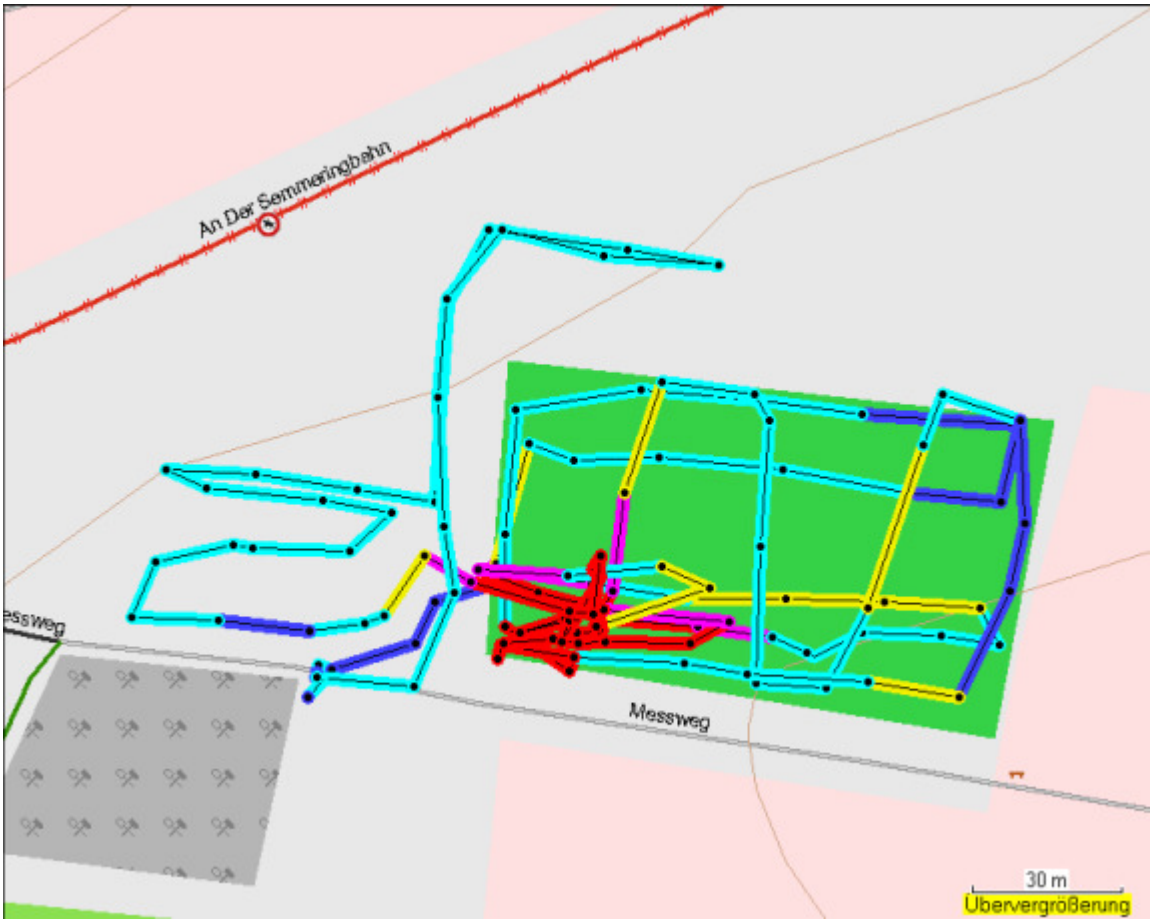

Abb. 6: Geo-referenzierte Messung mit dem Kontaminationsdetektor. Dabei bedeutet wieder rot >0.5uSv/h, magenta>0.4uSv/h, gelb>0.3uSv/h, türkis>0.2uSv/h, blau>0.1uSv/h

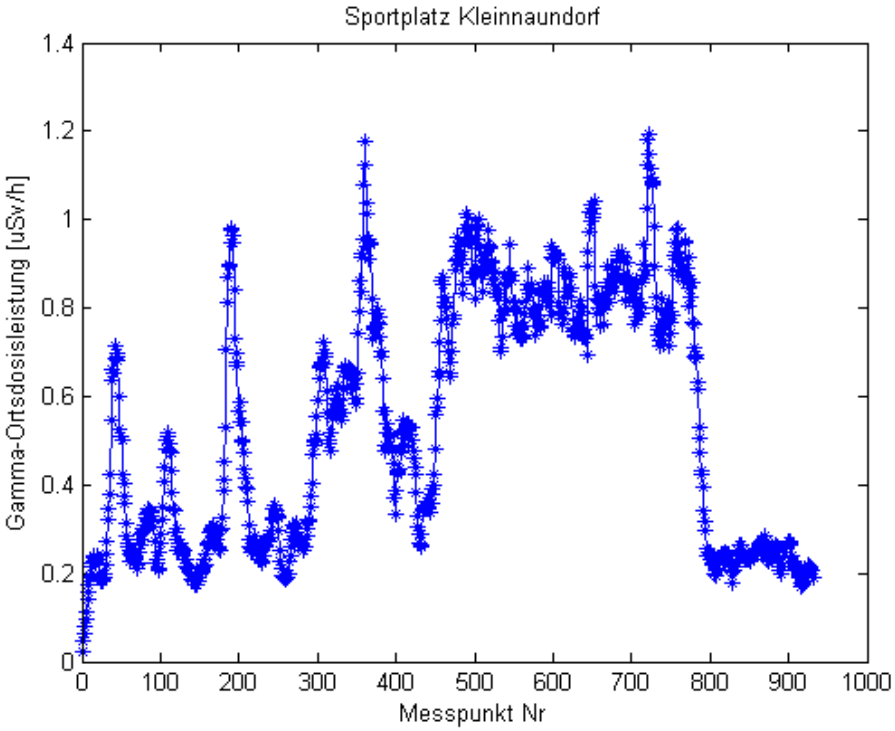

Abb 7: Profil des Messwegs aufgezeichnet vom Kantaminationsdetektor