## Ein Arduino-basierter Szintillationszähler mit LCD-Display, Real-Time-Clock und Datalogging

Bernd Laquai, 27.1.16 Update 6.2.16

Die Tatsache, dass ein Handheld GPS-Empfänger zu den Geokoordinaten auch die Zeit in einem sogenannten GPX-Track mitaufzeichnet, ermöglicht georeferenzierte Messungen, wenn gleichzeitig auch die Messgröße zusammen mit einem Zeitstempel aufgezeichnet wird. Dann lässt sich nämlich programmgesteuert einem Messwert auch eine Geokoordinate zuordnen, wenn beide Datensätze später über die Zeit synchronisiert werden. Das lässt sich insbesondere für Messungen der Umweltradioaktivität einsetzen, wo der Messort später rekonstruiert werden soll, oder Karten mit Messwerten erstellt werden sollen. Um dieses Prinzip der Georeferenzierung nutzen können, muss das Messgerät über eine interne Uhr und eine Messdaten-Aufzeichnungs-Einheit (z.B. Speicherung auf SD-Karte) besitzen. Gesteuert wird dies am einfachsten durch einen Mikrocontroller. Eine ideale Plattform dafür ist der Arduino Mikrocontroller, für den es sowohl eine SD-Karten Aufsteckplatine wie auch Real-Time-Clock Module und LCD-Displays gibt, die über fertig entwickelte Bibliotheksroutinen angesprochen werden können. Das reduziert den Entwicklungsaufwand massiv und ermöglicht es in kürzester Zeit ein funktionsfähiges System aufzubauen. Ein Szintillations-Detektor hat in dieser Anwendung schließlich den Vorteil, dass er eine hohe Empfindlichkeit für Gamma-Strahlung hat und durch seine hohe Zählraten stabile Messwerte liefert.

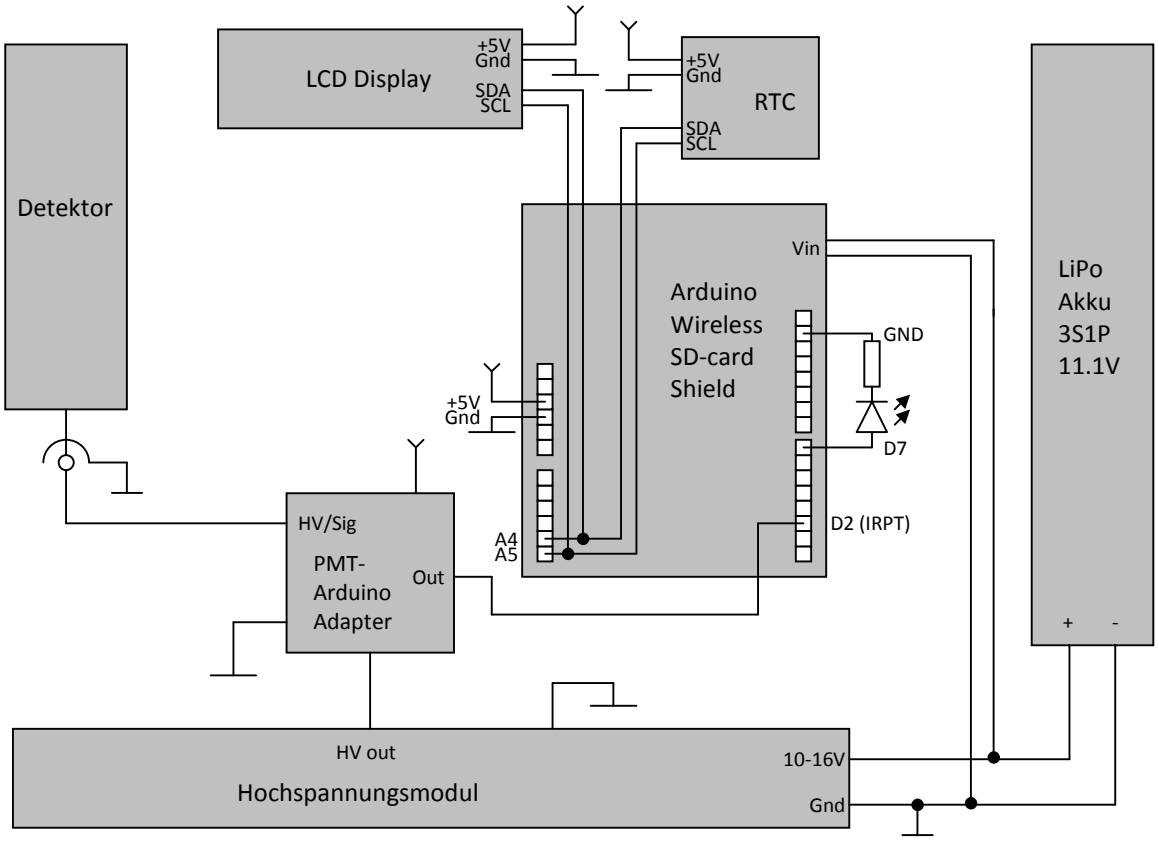


Abb. 1: Verschaltung der Systemkomponenten des Szintillationszählers

Um ein derartiges System zu implementieren, wurde hier als Detektor ein selbstgebauter Szintillations-Detektor mit 1 Zoll Photomultiplier verwendet, der nach dem Theremino-Schema für positive Hochspannung verdrahtet ist. Die Hochspannung wird von einem fertigen Hochspannungsmodul erzeugt, wie es beispielsweise von der Firma Volgen erhältlich ist. Es erzeugt 1000V Hochspannung, bei einem Maximalstrom von 2mA. Eingansseitig können 10.8-16.5V zur Versorgung angelegt werden.

Für die Anpassung der Ausgangspulse des Photomultipliers wurde ein PMT-zu-Arduino Adapter entwickelt, der über ein passives Anpassnetzwerk und einen Komparator digitale Zählpulse abliefert, die auf den Interrupt-Eingang des Arduino geführt werden. Die Real-Time-Clock (Adafruit) und das LCD-Display (YwRobot) besitzen eine serielle $I^2C$-Bus Schnittstelle, die Daten über die SDA und SCL Leitungen des $I^2C$-Bus austauschen. Der Arduino nutzt die Pins A4 nd A5 für den Anschluss eines $I^2C$-Bus. Auf ein Arduino Uno, der das System steurt, wurde zusätzlich noch ein Wireless SD-Card Shield aufgesteckt, welches das Abspeichern von Daten auf eine Micro-SD Speicherkarte ermöglicht. Der Wireless-Teil dieses Shields wurde nicht genutzt. As einem hier nicht relevanten Grund wurde bei dem Shield die Chip-Select Leitung der SD-Karte vom Pin 4 auf den Pin 10 verlegt. Die Nutzung des Pin 4 wäre aber genauso möglich gewesen.

Schließlich wurde noch zur Signalisierung Im Falle eines Fehlers bei der Initialisierung der SD-Karte eine Kontroll-LED an den digitalen Pin 7 angeschlossen.

Gespeist werden Arduino und das Hochspannungsmodul aus einem 3-zelligen Lithium-Polymer Modellbau-Akku. Die übrigen peripheren Komponenten, die am Arduino angeschlossen sind, werden über dessen interne 5V-Versorgung gespeist.

```
//Szintillations-Detektor mit Real Time Clock und SD-Card Datalogging
#include <Wire.h>
#include "RTClib.h"
#include <SD.h>
#include <SPI.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

// Define variables
#define I2C_ADDR    0x27  // Define I2C Address where the PCF8574A is
#define BACKLIGHT_PIN     3
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7

#define MAXCNT 500
#define CalFactor 1

RTC_DS1307 rtc;

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

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

//Initialise the LCD
LiquidCrystal_I2C   lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
```

```
void setup() {

// Define LCD as 16 column x 2 rows
  lcd.begin (16,2);

// Switch on the backlight
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);

// Goto first column (0 not 1!), first line (0 not 1!),
  lcd.setCursor ( 0, 0 );

// Print at cursor location
  lcd.print("opengeiger.de");

  pinMode(10, OUTPUT);
  pinMode(redLed, OUTPUT);
  digitalWrite(redLed, LOW);
  if (!SD.begin(10)) {
    digitalWrite(redLed, HIGH); //SD card not ready
    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();
  // 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));

  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)counter*1000.0/(float)dt/CalFactor;

    DateTime now = rtc.now();

    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();

    lcd.clear();
    lcd.setCursor (0, 0);
    lcd.print(now.hour());
    lcd.print(':');
    lcd.print(now.minute());
    lcd.print(':');
    lcd.print(now.second());

    lcd.setCursor ( 0, 1 );
    lcd.print(rate);
```

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

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

Abb. 2: Listing des Arduino Zähl- und Datalogging-Programms

Das Arduino-Programm nutzt die für die verwendeten Module vorhandenen Bibliotheken und baut auf den dort vorhandenen Beispielen auf. Das Programm beginnt mit der Initialisierung des LCD-Displays und der Real-time-Clock sowie der SD-Karte ganz nach den mitgelieferten Beispielen. Wenn die SD-Karte korrekt initialisiert wurde, wird eine Datalogging-Datei angelegt mit dem Namen „datalog.txt“. Existiert diese bereits, wird lediglich ein Trennzeichen „-----„ geschrieben, ansonsten „###“ um den Beginn der Aufzeichnung zu markieren. Wird die Uhr nach einem Batteriewechsel neu initialisiert werden muss in der Zeile:
// rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
das Kommentar entfernt werden, und dann der Code übertragen und ausgeführt werden. Danach muss das Kommentar wieder angebracht und der Code erneut übertragen werden, damit nun die Uhr nicht immer neu initialisiert wird.

Nach der Initialisierung wird schließlich noch die Interrupt-Verarbeitung aktiviert. Bei Auftreten eines Interrupts springt die Programmausführung in die Routine count(), die lediglich einen Zähler erhöht.

Tritt kein Interrupt auf, läuft das Programm in der Schleife loop() endlos. In dieser Schleife wird der Zählerstand abgefragt und wenn MAXCNT erreicht ist, dann wird zu allererst die Interrupt-Verarbeitung deaktiviert, damit sich keine neuen Interrupts aufstauen. Dann wird die Zählrate aus der Systemzeit in Counts pro Sekunde berechnet und die Uhr ausgelesen. Die Zeitinformation und die Zählrate werden in eine Zeile der Datalogging-Datei geschrieben. Danach erfolgt die Anzeige der Zeit und der Zählrate auf dem LCD-Display. Am Ende der Schleife wird das Zählregister und die Speichervariable für die Zeitberechnung wieder zurückgesetzt. Schließlich wird die Interrupt-Verarbeitung wieder neu aktiviert.

Dieses Zähl-, Speicher- und Anzeigeprogramm ist der Verständlichkeit halber maximal einfach gehalten. Es erlaubt aber dennoch einen erstaunlich leistungsfähigen und durchaus ergonomischen Betrieb. Ausgewertet werden die aufgezeichneten Daten nach der Messung am PC. Dazu wird die Speicherkarte mit dem PC ausgelesen und damit ein Auswerteprogramm (ganz nach Bedarf und Anwendung) gefüttert. Die für georeferenzierte Messungen notwendige Synchronisation der Daten mit den Daten eines GPS-Handheld-Empfängers können somit bequem am PC mit einer speziell entwickelten Software durchgeführt werden und die Daten auf einer digitalen Kartensoftware angezeigt werden.

Das System wurde als Prototyp auf einer Lochrasterplatine aufgebaut und provisorisch in einem Schuhkarton verstaut. Detektor und Akku wurden mit Kabelbindern befestigt, die Platine und das LCD-Display wurden mit Kunststoffschrauben fixiert. Der Schuhkarton hat gut in einer Umhängetasche Platz, die sich mit der Länge des Trageriemens auf Gonadenhöhe einstellen lässt.

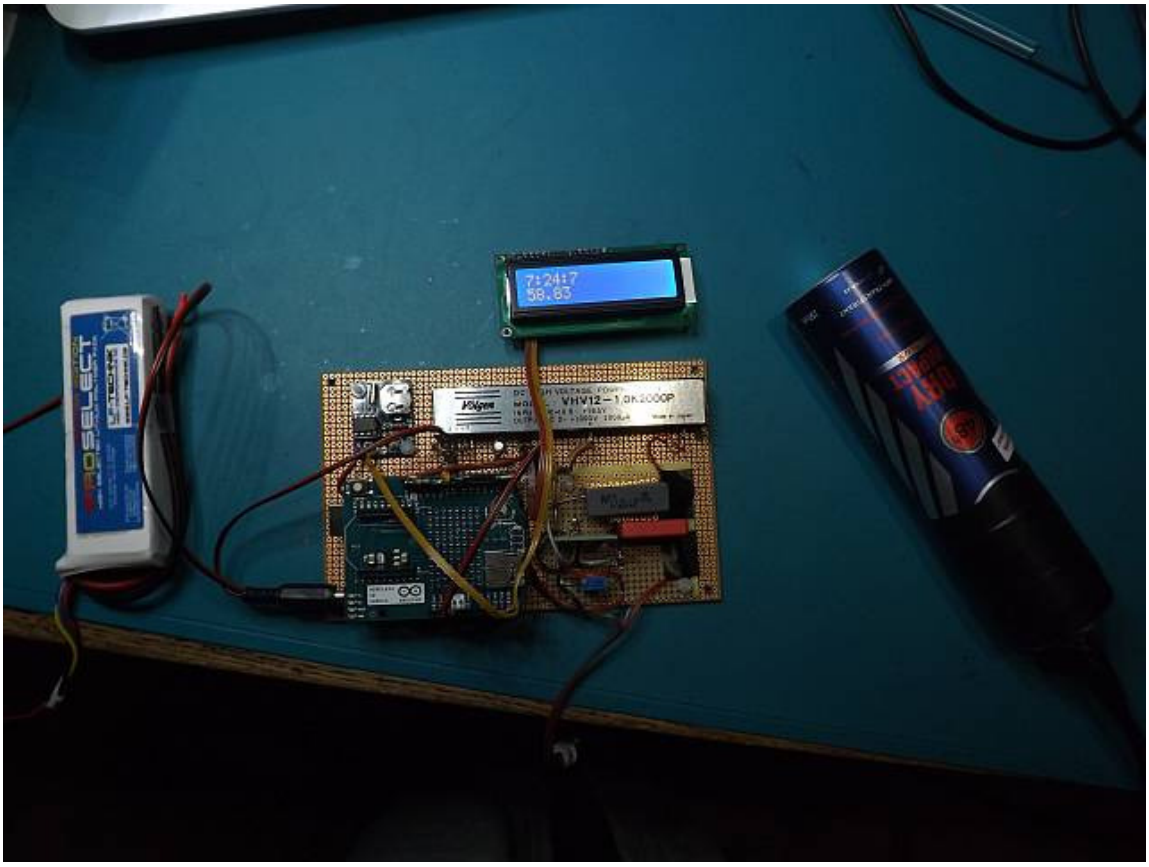

Abb. 3: Test der Systemkomponenten, die Nullrate im Wohnraum beträgt etwa 60cps

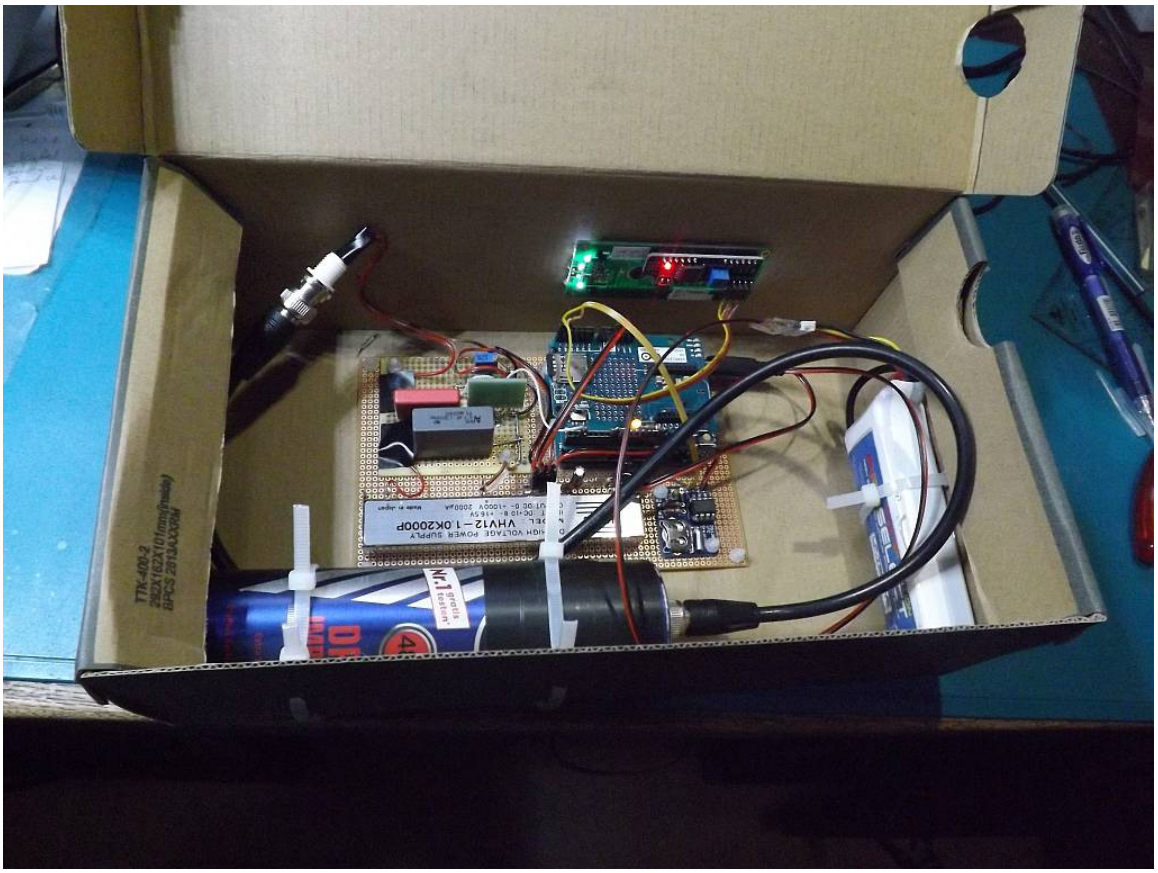

Abb. 4: Provisorisches Verstauen der Systemkomponenten in einem Schuhkarton

Abb. 5: Das beleuchtete LCD-Display wird an der Seitenwand montiert und kommt später in der Umhängetasche oben zu liegen.

Zur Demonstration und Kalibrierung des Szintillationszählers wurde der berühmte „Königstrassen-Achter“ in Stuttgart abgelaufen. Die Königstrasse ist in ihrer ganzen Länge mit Flossenbürger-Granit belegt, der einen deutlichen Urangehalt aufweist. Dieser erhöht die Ortsdosisleistung ziemlich genau um den Faktor 3. Dies wurde vom Landesamt für Umwelt in Baden-Württemberg (LUBW) nachgemessen und bestätigt. Die Dosisleistung ist entlang des breiten Pflasters sehr konstant, daher hat man einen idealen flächenhaften Prüfstrahler für Niedrigdosisstrahlung aus natürlichem Uran.

Das Profil der Messdaten zeigt entlang der Teile des Messwegs, die auf der Königstrasse liegen, eine mittlere Zählrate von etwa 105cps und im Schlossgarten bzw. der asphaltierten Theodor-Heuss-Strasse von ungefähr 35cps. Diese Zählraten wurden aus 500Pulsen (MAXCNT=500 ) berechnet. Ordnet man nun der Zählrate von 105cps die Ortsdosisleistung von 0.3uSv/h zu, wie vom LUBW ermittelt, dann ergeben sich für die 35cps grob 100uSv/h was fast perfekt mit den Werten, die von der ODL-Messstation des Bundesamts für Strahlenschutz im Schlossgarten gemessen werden, übereinstimmt. Auf diese Weise lässt sich der Detektor im Niedrigdosisbereich hervorragend auf die Gammastrahlung des natürlichen Uran kalibrieren.

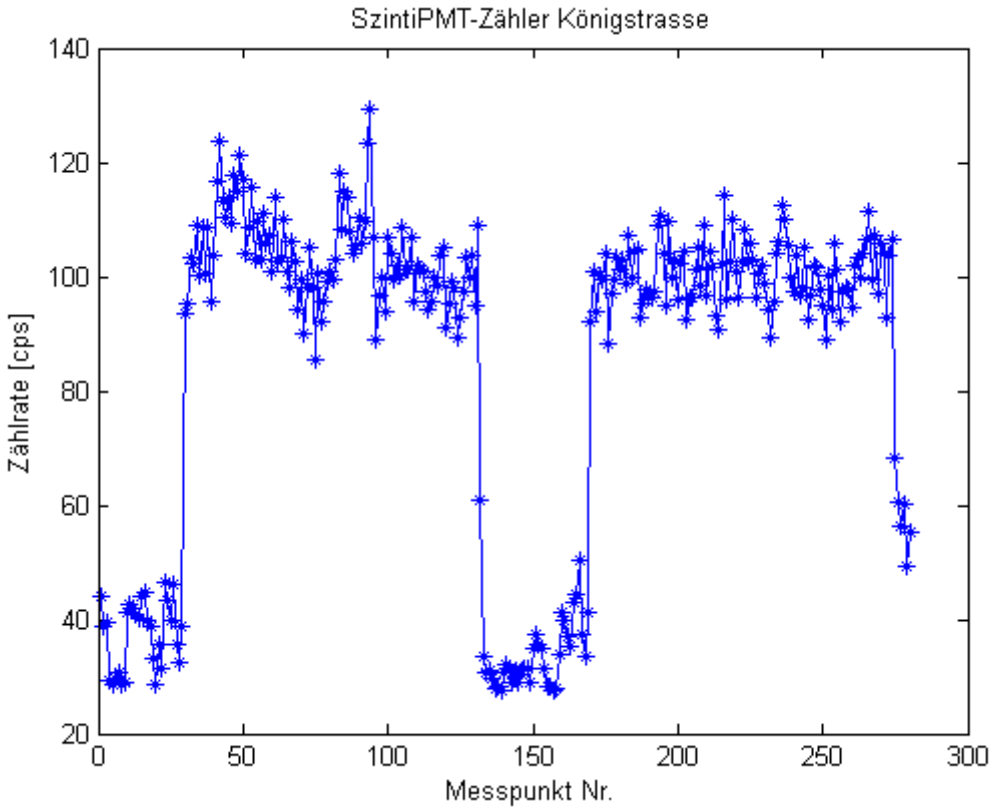


Abb. 6: Profil der Messstrecke, es zeigt die Zählrate über den Messpunkten an denen auch Koordinaten aufgezeichnet wurden

Ordnet man nun dem Messweg Farben entsprechend der Intensität der Ortsdosisleistng zu, dann lässt sich das Intensitätsprofil zur Visualisierung sehr gut auf einer Kartensoftware wie z.B. Google Maps oder anderen darstellen. Die um den Faktor 3 erhöhte Gamma-Ortsdosisleistung bildet sich so ganz deutlich entlang der Königstrasse in Rot-Tönen ab, im Schlossgarten und entlang der Thodor-Heuss-Strasse zeigt die Färbung die normale Ortsdosisleistung in Blau an.

Nach dieser Kalibrierung kann der Szintillations-Zähler zur quantitativen Messung von Böden mit vergleichbarem Gehalt an natürlichem Uran eingesetzt werden. Der Vorteil des Szintillationszählers dabei ist die hohe Zählrate, die relative stabile Messwerte (Streuung < 4%) trotz normaler Gehgeschwindigkeit von 4-6km/h für eine Ortsauflösung von wenigen Metern liefert.

## Literatur

opengeiger.de:

Eigenbau eines Szintillations-Detektors mit NaI-Szintillator und Photomultiplier als empfindlicher, mobiler Sensor für Umwelt-Radioaktivität

PMT-zu-Arduino Adapter zum Betrieb eines Szintillations-Detektors mit Photomultiplier an einem Arduino Mikrocontroller

Abb. 7: Beispiel der Messwegdarstellung auf Google Maps

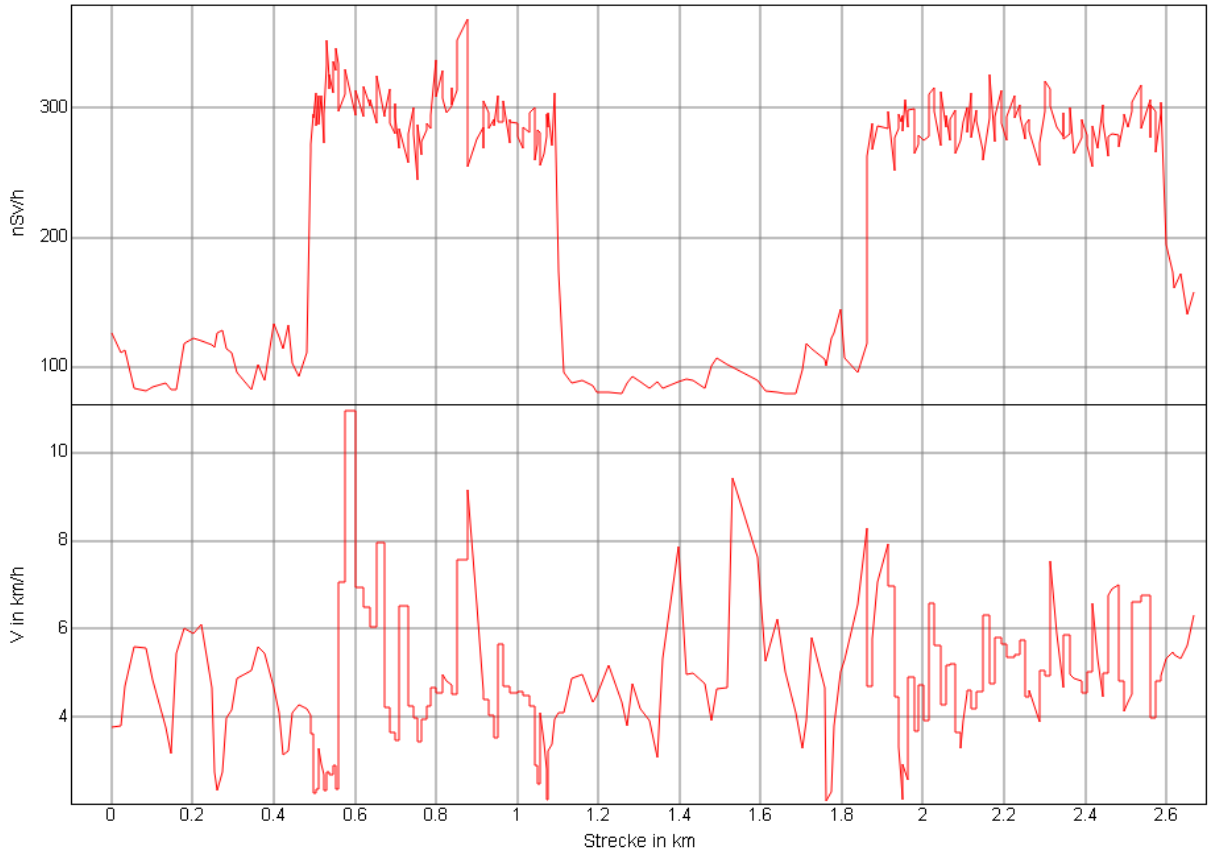


Abb. 8: Darstellung der Messdaten, oben auf nSv/h umgerechnete Ortsdosisleistung unten Geschwindigkeit entlang des Messwegs