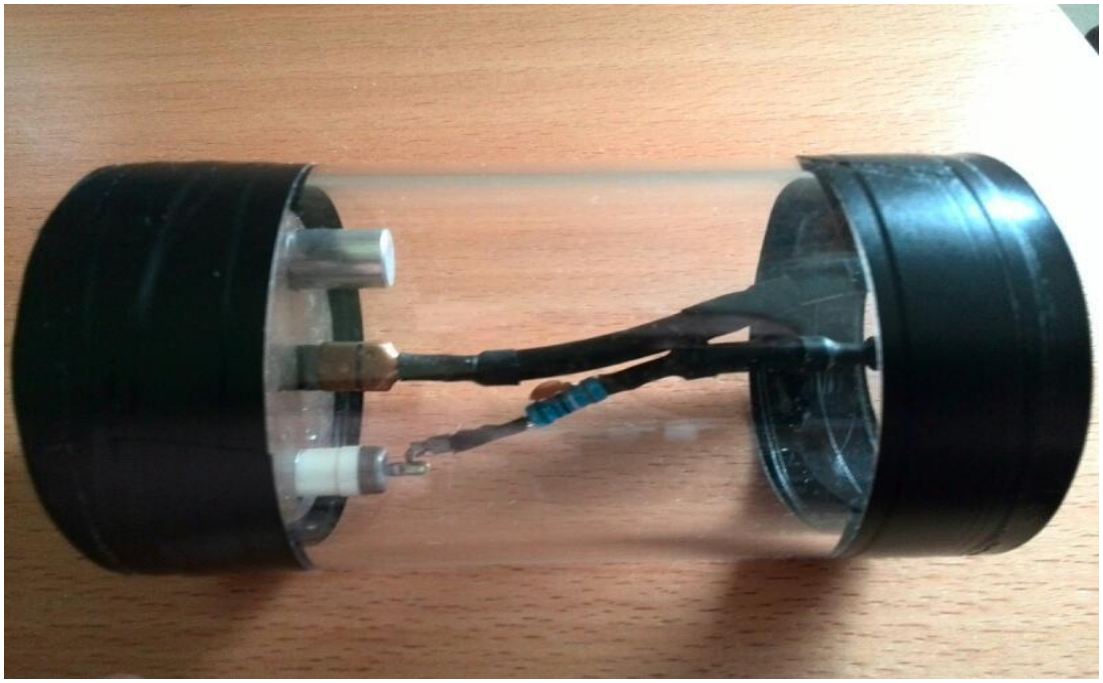
Gewerbliche Berufsbildende Schulen

des Landkreises Grafschaft Bentheim

- Berufliches Gymnasium Technik -

Projektarbeit in der Klasse 12 mit dem Thema

Bau eines Geiger-Müller-Zählers mit einem selbstgeschriebenen Auswertprogramm



vorgelegt von: Daniel Berning, Robin Stöhrmann

Fach:	Praxis der Technik
Fachlehrer:	Herr M. Kantner
Klasse:	BGT2-2
Ort:	Nordhorn
Datum:	24.04.2014

1 Inhaltsverzeichnis

2 Einleitung

Die überallvorkommende Radioaktivität in vielen Elementen des Periodensystems und Stoffen auf der Erde hat unser Interesse geweckt. Vor allem der Reiz mit einem selbstgebauten Geiger-Müller-Zähler diese Strahlung zu erkennen und auch in einem selbstentwickelten Programm zu visualisieren hat uns dazu angeregt, sich für dieses Projekt zu entscheiden. Außerdem ist es eine große Herausforderung um seine eigenen Kompetenzen in dem Bereich der Programmierung und dem eigenen Wissen über Radioaktivität zu erweitern. Der Schwierigkeitsgrad dieses Projekts ist hoch, weil alles selbst erarbeitet werden muss und es das erste Projekt dieser Art überhaupt ist und somit alles von Null aufgebaut werden muss. Wir wollten unsere Grenzen testen und sie erweitern, gerade deshalb haben wirdieses Pionierprojekt ausgewählt, um es zu realisieren.

3 Projektplanung

3.1 Projekt Beschreibung

Das vorgefertigte Geiger-Müller-Zählrohr befindet sich in einem selbst gebauten Gehäuse.Angeschlossen mit einer BNC Verbindung wird das Signal bis zur Sound Karte geleitet. Dazu werden ein Kondensator und ein Widerstand parallel gelötet, um ein Mikrofonsignal zu imitieren. Somit fungiert die Soundkarte als Interface zwischen dem Zählrohr und dem PC. Deshalb können die Daten per selbstprogrammierten C# Programm mit benutzerfreundlicher Oberfläche ausgelesen werden. Das Programm bietet noch ein Diagramm in verschieden Zeiträumen um die Strahlung zusätzlich darstellen zu können und gleichzeitig einen aktuellen Strahlenwert (Wird sekündlich gemessen) anzuzeigen. Die Strahlung wird in Clicks pro Sekunde angezeigt.

3.2 Projektziel und Gegebenheiten

Die folgende Dokumentation hat das Ziel unser Projekt „Geiger-Müller-Zähler“ zu verstehen in welcher Weise diese Technologie funktioniert und wie das selbst entwickelte Programm in C# abläuft. Das Hauptthemaist die Programmierung, auf welche wir noch detaillierter eingehen werden. Außerdem beschreiben wir die allgemeine Funktionsweise eines Geiger-Zählers und den Aufbauanhand unseres Ausstellungsstücks, und anschließend die Inbetriebnahme dieses Gerätes. In diesemProjekt greifen wir auf eigenes Wissen, das Wissen unserer betreuenden Lehrer und auf das Internet zurück. Besonderer Anreiz und Hilfe bat uns eine Anleitung der Universität Esslingen.[1] Da dies das erste Projekt ist, welches als Ziel

[11]Autor: Unbekannt
URL: www.strahlenschutzkurse.de/fileadmin/strahlenschutzkurse/dokumente/lehrer/grflzaro.pdf Letzter Abruf 25.04.2014
Seminar Esslingen Strahlenschutz

ein funktionierendes Ausleseprogrammm und einen selbst gebauten Geiger-Müller Zähler beinhaltet, ist das Wissen um die Ausführung begrenzt.

3.3 Zeitliche Einteilung

Wir haben uns vorab mit der wichtigen Frage beschäftigt, wie wir unser Projektplanen und in welchen Zeitabschnitten. Im groben haben wir es in fünf Phasen bzw. Meilensteine eingeteilt. Wobei die Präsentation wegfallen wird.

1. 18.11.2013-6.12.2013 Materialbeschaffung+ Projektplanung
   Hier werden die Zählköpfe, sowie die Kondensatoren, Widerstände, die BNC-Stecker und Buchsen besorgt. Die Verwendeten Leitungen und Schrumpfschlauch sind bereits vorhanden.

2. 6.12.2013- 31.1.2014 Bau des Zählrohres und Anfänge der Programmierung
   Hier wird die endgültige Entscheidung über das Material des Gehäuses Gefällt und die Teile zusammengesetzt. Die Programmierung wird im Groben Angefangen(Oberfläche mit Grafik)

3. 1.2.2014-1.4.2014 Hauptprogrammierung des Programms
   Der Hauptteil und das Fein-Tuning werden an unserem Programm durchgeführt, dazu zählt vor allem die Stimmigkeit der Werte der Messungen

4. 1.4.2014-20.6.2014 Dokumentation
   Die mit bestem Wissen und Gewissem erstellte Dokumentation wird vollendet

5 4.7.2014Präsentation des Projekts

3.4 Ressourcenplanung

Für unser Projekt benötigen wir einiges an Material: Teilweise sind die Materialien vorhanden, aber die Sachen aus Plexiglas, sowie der Leybold-Stecker sind schwer zu bekommen. Und auch die Widerstände und der Kondensator sind in 500V nicht üblich und deshalb umständlich zu bekommen.

- Kondensatoren 47pF 500v
- Widerstände 5,6*10^6Ω 500v
- Plexiglasscheibe 5mm Dick 10x10 cm
- Plexiglaszylinder54mm Durchmesser 3mm Dick
- BNC-Stecker 9mm
- BNC-Buchse 9mm
- Zuleitung Koaxialkabel
- Schrumpfschlauch (verschiedene Größen)
- Spezieller Leybold-Stecker (Extra Anfrage an Leybold geschickt)
- AUX Stecker für das Audiosignal
- Schutzgitter für das Empfindliche Fenster (Aus Boxen herausgebaut)
- Zählkopf (LND 7313“Pancacke“ Detektor)

4 Inhalt

4.1 Programm

4.1.1 Ablauf

Beim Starten des Programms erscheint das Fenster aus Abb.2. Hierbei muss der Anwender zunächst einen Speicherpfad auswählen, für die Wave-Dateien, die das Programm später erstellt.

[2]

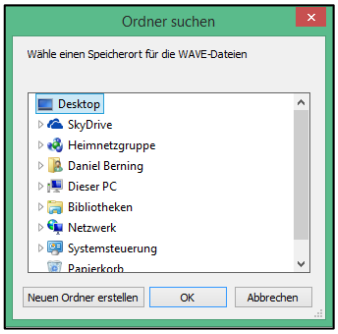


Abb.2 Auswahl des Speicherpfades der Wave-Dateien

Nach Auswahl des Pfades startet das eigentliche Programm.

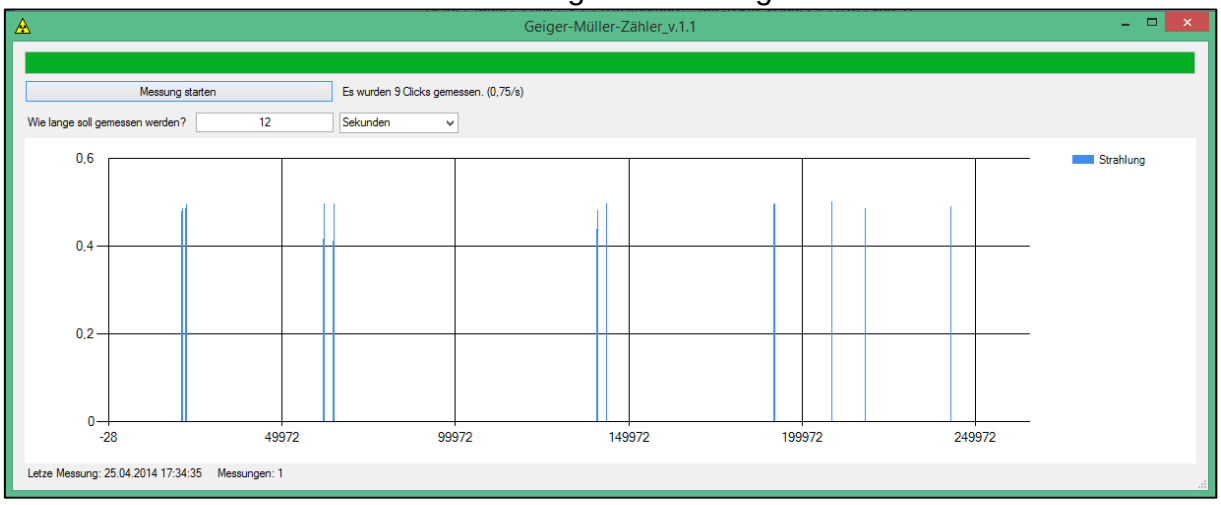


Abb.3 Eingabemaske des Programms. Hier wurde eine Messung über 12 Sekunden gemacht mit 9Clicks

In Abb.3 ist die Eingabemaske zu sehen. Die Eingabe erfolgt wie folgt:
Man gibt eine Zahl in das Eingabefeld ein und wählt in welcher Größenordnung gemessen werden soll. Dabei kann man zwischen Millisekunden und Sekunden wählen. Mit einem Klick auf den Button „Messung starten“ wird die Klacken[3] des Geigermüllerzählers über den Mikrofoneingang[4] aufgenommen. Der Fortschritt der Aufnahme wird durch den Balken dargestellt. Nach Beendigung der Aufnahme wird diese als Wave-Datei an dem zuvor gewählten Speicherort gespeichert. Daraufhin

[2] Screenshots: Daniel Berning
[3] Der Geigermüllerzähler nimmt bei Strahlung ein Klacken war.
[4] Dabei ist darauf zu achten, dass in den Soundeinstellungen des Pc‘s die Rauschunterdrückung ausgeschaltet ist.

wird diese Wave-Datei ausgelesen und dadurch die Strahlung ermittelt, welche im Ausgabetext ausgegeben wird und zusätzlich als Grafik, in der man ablesen kann wann die Clicks waren. Mit dem erneuten starten einer Messung wird automatisch die alte Wave-Datei gelöscht um den Speicher vor Redundanz zu schützen.
Als Zusatzfunktion zeigt das Programm noch die Anzahl der Messungen und zu welchem Zeitpunkt die letzte Messung war. Außerdem passen sich die Größe und Position der Elemente automatisch der Größe des Formulars[5]an. Außerdem wird per Sprachausgabe die Höhe der Strahlung ausgegeben. Während das Formular geöffnet ist werden im Hintergrund Messungen im Ein-Sekunden-Takt gemacht, die ebenfalls ausgegeben werden. Sowohl die Langzeitmessungen, als auch die kontinuierlichen Messungen werden in Protokollen im csv-Dateiformat festgehalten.

10526	1000	ms	43	43	/s	11.06.2014 18:34
10527	1000	ms	39	39	/s	11.06.2014 18:34
10528	1000	ms	44	44	/s	11.06.2014 18:34
10529	1000	ms	46	46	/s	11.06.2014 18:34
10530	1000	ms	45	45	/s	11.06.2014 18:34
10531	1000	ms	47	47	/s	11.06.2014 18:34
10532	1000	ms	52	52	/s	11.06.2014 18:34
10533	1000	ms	50	50	/s	11.06.2014 18:34
10534	1000	ms	45	45	/s	11.06.2014 18:34
10535	1000	ms	49	49	/s	11.06.2014 18:34
10536	1000	ms	45	45	/s	11.06.2014 18:34
10537	1000	ms	39	39	/s	11.06.2014 18:34
10538	1000	ms	49	49	/s	11.06.2014 18:34
10539	1000	ms	45	45	/s	11.06.2014 18:34
10540	1000	ms	39	39	/s	11.06.2014 18:34
10541	1000	ms	45	45	/s	11.06.2014 18:34
10542	1000	ms	45	45	/s	11.06.2014 18:34
10543	1000	ms	47	47	/s	11.06.2014 18:34
10544	1000	ms	46	46	/s	11.06.2014 18:34
10545	1000	ms	46	46	/s	11.06.2014 18:34

[6]

Abb.4 Protokoll

[5] Als Formular bezeichnet man das Fenster.
[6]Screenshot: Daniel Berning

4.1.2 Code

Für die Programmierung wurde die Sprache Microsoft C# verwendet.
Hier haben wir die Klasse „Meine Klasse“ erstellt um den Programmiercode im Hauptprogramm einfacher und übersichtlicher zu gestalten. Um einen reibungslosen Ablauf des Programms zu gewährleisten, haben wir zwei Instanzen der KlasseBackgroundWorker erstellt, einen für die Intervallmessungen und ein für die kontinuierlichen Messungen.
Diese werden mit dem Form1_Load-Ereignis deklariert(Abb.5).

```
private void Form1_Load(object sender, EventArgs e)//Auswahl des Speicherpfades
{
    chartStrahlung.Series[0].Name = "Strahlung";//Name der Grafik
    chartStrahlung.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
    main = new BackgroundWorker();
    main.WorkerReportsProgress = true;
    main.WorkerSupportsCancellation = true;
    main.RunWorkerCompleted += new RunWorkerCompletedEventHandler(main_RunWorkerCompleted);
    main.ProgressChanged += new ProgressChangedEventHandler(main_ProgressChanged);
    main.DoWork += new DoWorkEventHandler(main_DoWork);
    worker = new BackgroundWorker();//Kontinuierlich
    worker.WorkerReportsProgress = true;
    worker.WorkerSupportsCancellation = true;
    worker.DoWork += new DoWorkEventHandler(worker_DoWork);
    worker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(worker_RunWorkerCompleted);
    DialogResult objResult;
    do
    {
        System.Windows.Forms.FolderBrowserDialog objDialog = new FolderBrowserDialog();
        objDialog.Description = "Wähle einen Speicherort für die WAVE-Dateien";
        objResult = objDialog.ShowDialog(this);
        zaehler.Pfad = objDialog.SelectedPath + @"\";
        if (objResult == DialogResult.OK)
        { MessageBox.Show("Neuer Pfad : " + objDialog.SelectedPath); }
        else
        { MessageBox.Show("Bitte einen Pfad auswählen!"); }
    } while (!(objResult == DialogResult.OK));
    iL1 = zaehler.GetID();
    labelMessungen.Text = "Messungen: " + Convert.ToString(iL1);
    labelLetzteMessung.Text = Convert.ToString(zaehler.GetDateTime());
    Konti.Datei = "Konti";
    Konti.Protokoll = "Kontinuierliche_Messung";
    Konti.Pfad = zaehler.Pfad;
    Konti.Zeit("1");
    i = Konti.GetID();
    worker.RunWorkerAsync(labelKonti.Text);
```

[7]

Abb.5Form1_Load

Ebenfalls wird mit diesem Ereignis der Pfad ausgewählt, an dem der Speicherort für die Wave-Aufnahmen und die Protokolle festgelegt. Die Schleife läuft so lange, bis ein Pfad ausgewählt, sonst startet das eigentliche Programm nicht. Mit dem Beenden dieses Ereignisses wird der BackgroundWorker für die kontinuierlichen Messungen mit *worker.RunWorkerAsync()* gestartet.
Mit dem Ereignis buttonMessen_Click wird das Formular zurückgesetzt und der zweite BackgroundWorker gestartet(Abb.6).

```
private void buttonMessen_Click(object sender, EventArgs e)
{
    try
    {
        chartStrahlung.Series[0].Points.Clear();//Alle Punkte werden aus der Grafik gelöscht
        Zeit = zaehler.Zeit(textBoxZeitintervall.Text);//
        iZaiet = Convert.ToInt32(textBoxZeitintervall.Text);
        progressBarAufnahmeFortschritt.Maximum = Zeit;
        progressBarAufnahmeFortschritt.Value = 0;//Fortschritt auf 0 setzen
        main.RunWorkerAsync(progressBarAufnahmeFortschritt.Value);
    }
    catch (Exception Fehler)
    { MessageBox.Show(Fehler.Message); }
}
```

Abb.6 buttonMessen_Click

Die Funktionen aus Abb.7 werden die Aufgaben des BackgroundWorkers ausgeführt. Mit DoWork wird die letzte Aufnahme gelöscht und eine neue über eine Sekunde aufgenommen. Mit dem Beenden von DoWork beginnt automatisch RunWorkerCompleted. In dieser Funktion wird die Aufnahme ausgewertet und die Anzahl der Clicks wird in dem Formular Form1 mit einem Label ausgegeben.

[7]Screenshot: Daniel Berning

Außerdem wird der Wert in das Protokoll geschrieben. Am Ende wird der BackgroundWorker nochmal aktiviert, was zu einer Endlosschleife führt.

```
void worker_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    try
    {
        i++;
        double Clicks;
        Clicks = Konti.Messen();
        labelKonti.Text = "Aktueller Wert: " + Convert.ToString(Clicks);
        Konti.Schreiben(i, Clicks, Clicks);
        worker.RunWorkerAsync();
    }
    catch (Exception)
    { i = 0; }
}
void worker_DoWork(object sender, DoWorkEventArgs e)
{
    try
    {
        Konti.AufnahmeLoeschen();
        Konti.AufnahmeStart();
        System.Threading.Thread.Sleep(1000);
        Konti.AufnahmeStop();
    }
    catch (Exception)
    { }
}
```

[8]

Abb.7 BackgroundWorkerworker mit den Ereignissen die bei seinem Aufruf ablaufen

Der BackgroundWorker aus Abb.8 macht das gleiche wie der andere. Er löscht zunächst die alte Aufnahme und beginnt mit dem Aufnehmen. Nach je einer Millisekunde wird ProgressChanged aktiviert, bei dem der Betrag der Progressbar um 1 erhöht. Nach beendigen der Aufnahme wird die Anzahl der Clicks ermittelt und in einer Grafik angezeigt. Auch hier wird der Wert in einem Protokollfestgehalt.

```
void main_DoWork(object sender, DoWorkEventArgs e)
{
    try
    {
        int percent = (int)e.Argument;
        zaehler.AufnahmeLoeschen();//Löschen der letzten Aufnahme
        zaehler.AufnahmeStart();//Starten der Aufnahme
        for (int i = 0; i < Zeit; i++)
        {
            main.ReportProgress(percent);
            System.Threading.Thread.Sleep(1);//Eine Millisekunde pausieren
        }
        zaehler.AufnahmeStop();//Beenden und speichern der Aufnahme
        e.Result = percent;
    }
    catch (Exception)
    { }
}

void main_ProgressChanged(object sender, ProgressChangedEventArgs e)
{
    progressBarAufnahmeFortschritt.Value++;
}

void main_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    try
    {
        double Clicks;
        Clicks = zaehler.Messen();//Übergabe des Strahlungswertes
        double ClicksProZeit = Clicks / iZaiet; ;//
        for (int j = 0; j != zaehler.X.Length; j++)
        {
            chartStrahlung.Series[0].Points.AddXY(zaehler.X[j], zaehler.Y[j]);//Zeichnen der Grafik
        }
        labelAusgabe.Text = "Es wurden " + Cogvert.ToString(Clicks) + " Clicks gemessen. (" + Convert.ToString(ClicksProZeit) + "/s)";/
        labelMessungen.Text = "Messungen: " + Convert.ToString(iL1);//Ausgabe von Anzahl der Messungen
        labelLetzteMessung.Text = "Letze Messung: " + Convert.ToString(DateTime.Now);//Ausgabe des Zeitpunkts der letzten Messung
        zaehler.Schreiben(iL1, Clicks, ClicksProZeit);
    }
    catch (Exception)
    { }
```

[9]

Abb.8 BackgroundWorkermain mit den Ereignissen die bei seinem Aufruf ablaufen

Nun werden wir die selbsterstellte Klasse „Meine Klasse“ näher erläutern.
Als erstes werdenwir die Befehle, die das Aufnehmen [10] betreffen(siehe Abb.9) erläutern.

[8]Screenshot: Daniel Berning
[9]Screenshot: Daniel Berning
[10]Autor: Maximilian Müller
URL: http://dotnet-snippets.de/snippet/wave-dateien-ueber-mci-aufnehmen/1594
LetzterAufruf 22.03.2014

```
[DllImport("winmm.dll")]
private static extern int mciSendString(string lpstrCommand, string lpstrReturnString, int uReturnLength, int hwndCallback);
public void AufnahmeStart()
{
    mciSendString("open new Type waveaudio Alias recsound", "", 0, 0);

    mciSendString("set recsound CHANNELS 1", "", 0, 0);
    mciSendString("record recsound", "", 0, 0);
}
public void AufnahmeStop()
{
    mciSendString("save recsound " + Pfad + "Strahlung.wav", "", 0, 0);
    mciSendString("close recsound", "", 0, 0);
}
public void AufnahmeLoeschen()
{
    System.IO.File.Delete(Pfad + "Strahlung.wav");
}
```

[11]

Abb.9 Aufnahme-Befehle

Dabei muss zunächst die DLL „winmm.dll“ importiert werden. Damit kann man auf die Soundkarte des Pc‘s zugreifen und die entsprechenden MCI-Befehle zugreifen.
Mit „AufnahmeStart“ wird die Aufnahme gestartet. Dabei wird die Anzahl der Kanäle, über die aufgenommen wird auf 1 gesetzt.
Mit „AufnahmeStop“ wird die Aufnahme beendet und an dem gewählten Pfad unter den Namen „Strahlung“ als Wave-Datei gespeichert.
„AufnahmeLoeschen“ löscht die Datei. Wenn keine Datei vorhanden ist wird keine Aktion durchgeführt.
Als nächstes wird das Herzstück, das Messen erläutert. Dabei sollte man aber auch wissen, was die hier verwendete Klasse „WaveFile[12]“ für eine Funktion hat. Mit der Klasse „WaveFile“ lassen sich Wave-Dateien auslesen. Dabei werden Dateigröße, Anzahl der Kanäle und andere Eigenschaften der eingelesenen Datei, unter anderem auch die Amplitudenwerte der einzelnen Samples, die in dem folgenden Abschnitt zur Auswertung gebraucht werden, ermittelt.
In Abb.10 wird der Befehl „Messen“ gezeigt.

[11]Screenshot: Daniel Berning
[12] Die Klasse stammt aus dem Internet
Autor: Oliver S
URL: http://csharp-tricks.blogspot.de/2011/03/wave-dateien-einlesen.html
LetzterAufruf 23.03.2014

```
public double[] X;//Array für Punkte in der Grafik
public double[] Y;//Array für Punkte in der Grafik
public int Messen()
{
    int iL1 = 1;
    int Strahlung;
    int Attacks = 0;
    double[] Normalisierung;
    double Maximal;
    bool Attack = false;
    WaveFile WF1 = new WaveFile();
    WF1.LoadWave(Pfad + Datei + ".wav");
    Maximal = WF1.Data[0].Max();
    Normalisierung = new Double[WF1.Data[0].Length];
    for (int i = 0; i < WF1.Data[0].Length; i++)
    {
        Normalisierung[i] = WF1.Data[0][i] / Maximal;
    }
    for (int j = 1; j < Normalisierung.Length; j++)
    {
        if (Normalisierung[j] >= 0.75 && Normalisierung[j - 1] < 0.75)
        { Attack = true; }
        if (Attack && Normalisierung[j] < 0.75 && Normalisierung[j] != 0)//Attacken zählen
        {
            Attacks++;
            Attack = false;
        }
    }
    X = new double[Attacks + 2];
    Y = new double[Attacks + 2];
    X[0] = 0;
    Y[0] = 0;
    for (int k = 1; k < Normalisierung.Length; k++)
    {
        if (Normalisierung[k] >= 0.75 && Normalisierung[k - 1] < 0.75)
        { Attack = true; }
        if (Attack && Normalisierung[k] < 0.75 && Normalisierung[k] != 0)
        {
            X[iL1] = k;
            Y[iL1] = Normalisierung[k];
            iL1++;
            Attack = false;
        }
    }
    X[X.Length - 1] = Normalisierung.Length;
    Y[Y.Length - 1] = 0;
    Strahlung = Runden(Attacks, 3);
    return Strahlung;
```

13

Die Arrays X und Y beinhalten die Werte für die Grafik die im Hauptprogramm gezeichnet wird.
Zunächst wird die lokale Variable WF1, die eine WaveFile ist erstellt. Diese lädt die Aufnahme und schreibt die Daten in ein Array. Da es bei verschiedenen Aufnahmen verschiedene Maximalwerte innerhalb des Arrays vorkommen können werden die Werte in der ersten for-Schleife normalisiert[14]
und in das Array Normalisierung geschrieben. In der darauffolgenden for-Schleife wird das Array Normalisierung dann ausgelesen. Wenn ein Wert 0,75 übersteigt und wieder abfällt, wird dies als Attack[15] gewertet und gezählt.

[13] Screenshot: Daniel Berning
[14] Alle Werte werden entsprechend des Maximalwertes geändert, so dass von allen Aufnahmen der Maximalwert 1 ist.
[15] Als Attack bezeichnet man den Anstieg eines Wertes über eine bestimmte Grenze. Es gibt auch Release den Abfall unter eine bestimmte Grenze. Anzahl von Attacks und Releases ist gleich.

```
public double[] X;
public double[] Y;
public int Messen()
{
    int iL1 = 1;
    int Strahlung;
    int Attacks = 0;
    double[] Normalisierung;
    double Maximal;
    bool Attack = false;
    WaveFile WF1 = new WaveFile();
    WF1.LoadWave(Pfad + Datei + ".wav");
    Maximal = WF1.Data[0].Max();
    Normalisierung = new Double[WF1.Data[0].Length];
    for (int i = 0; i < WF1.Data[0].Length; i++)
    {
        Normalisierung[i] = WF1.Data[0][i] / Maximal;
    }
    for (int j = 1; j < Normalisierung.Length; j++)
    {
        if (Normalisierung[j] >= 0.75 && Normalisierung[j - 1] < 0.75)
        { Attack = true; }
        if (Attack && Normalisierung[j] < 0.75 && Normalisierung[j] != 0)//Attacken zählen
        {
            Attacks++;
            Attack = false;
        }
    }
    X = new double[Attacks + 2];
    Y = new double[Attacks + 2];
    X[0] = 0;
    Y[0] = 0;
    for (int k = 1; k < Normalisierung.Length; k++)
    {
        if (Normalisierung[k] >= 0.75 && Normalisierung[k - 1] < 0.75)
        { Attack = true; }
        if (Attack && Normalisierung[k] < 0.75 && Normalisierung[k] != 0)
        {
            X[iL1] = k;
            Y[iL1] = Normalisierung[k];
            iL1++;
            Attack = false;
        }
    }
    X[X.Length - 1] = Normalisierung.Length;
    Y[Y.Length - 1] = 0;
    Strahlung = Runden(Attacks, 3);
    return Strahlung;
}
```

[16]

Abb.6 Der Befehl Messen

Die Arraylänge der Arrays X und Y ist die Anzahl der Attacken +2, weil man den ersten Wert und den letzten Wert des Arrays Normalisierung übernimmt damit man später in der Grafik die Zeitpunkte der Clicks ablesen kann. Dafür ist die dritte for-Schleife zuständig. Man kann es nicht in der gleichen Schleife, wie die für das Zählen der Attacken ausführen, da man zuerst die feste Arraylänge benötigt, die man mithilfe der Anzahl der Attacken ermittelt, um es zu beschreiben. Die Anzahl der Clicks lässt sich einfach durch das Teilen der Anzahl der Attacken durch 3 ermitteln. Auf den Faktor 3sind wir über verschiedene Messungen gekommen. Da haben wir manuell mithilfe des Programms „Audacity[17]“ die Clicks gezählt und mir von dem noch unfertigen Programm die Anzahl der Attacken ausgeben lassen.
Dabei konnte man erkennen, dass die Attacken fast immer das Dreifache von den Clicks ergab mit einer Abweichung von eins. Daher wird am Ende das Ergebnis noch einmal gerundet. Der gerundete Wert wird dann zurückgegeben.
Um die Anzahl der Messungen bei einem Neustart des Programms zu ermitteln, liest die Funktion GetID aus Abb.11 die letzte Zeile des Protokolls und wandelt es in einen Zahlenwert um.

[16]Screenshot: Daniel Berning
[17] Dieses Programm kann die Struktur von Audiodateien anzeigen.

```
public long GetID()
{
    long ID = 0;
    int i = 0;
    char[] Dezi = new char[10] { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };
    if (File.Exists(Pfad + Protokoll + ".csv"))
    {
        string Protokollw = "";
        FileStream fs = new FileStream(Pfad + Protokoll + ".csv", FileMode.Open);
        StreamReader sr = new StreamReader(fs);
        while (sr.Peek() != -1)
        {
            Protokollw = sr.ReadLine();
        }
        char[] cID = Protokollw.ToCharArray();
        while (Dezi.Contains(cID[i]))
        {
            i++;
        }
        for (int j = 0; j != i; j++)//12 hat 2 Ziffern ==> 1*10^1 +2*10^0
        {
            string sid = Convert.ToString(cID[j]);
            ID += Convert.ToInt64(sid) * Convert.ToInt64(Math.Pow(10, (i - j - 1)));
        }
        fs.Close();
        sr.Close();
    }
    else
    {
        StreamWriter sw = File.AppendText(Pfad + Protokoll + ".csv");
        sw.WriteLine("Nr;Zeit;;Clicks;Clicks pro Sekunde;;Datum/Zeit");
        sw.Close();
    }
    return ID;
```

[18]

Abb.11 Ermitteln der Nummer der Messungen

Um dem Protokoll Werte hinzuzufügen wird die Funktion Schreiben benutzt. Hier bei wird in eine neue Zeile die Nummer, die Dauer, der Durchschnittswert und der Zeitpunkt der Messung geschrieben(Abb.12).

```
public void Schreiben(long iL, double C, double CPS)
{
    string text;
    text = Convert.ToString(iL) + ";" + Convert.ToString(zeit * 2) + ";" + "ms;" + Convert.ToString(C) + ";" + Convert.ToString(CPS) + ";" + "/s;" + Convert.ToString(DateTime.Now);
    StreamWriter sw = File.AppendText(Pfad + Protokoll + ".csv");
    sw.WriteLine(text);
    sw.Close();
}
```

[19]

Abb.12 Protokoll schreiben

[18]Screenshot: Daniel Berning

[19]Screenshot: Daniel Berning

4.1.3 Probleme bei der Programmierung

Ein großes Problem war, dass ein Array in der Klasse „WaveFile“ die falsche Länge hatte und deshalb nur einen Wert speichern konnte. Nach genauem Einarbeiten in diesem Programmteil konnte nach langer Zeit dieser einfache Fehler behoben werden. Der Fehler kam wahrscheinlich dadurch zustande, dass wir die diesen Teil eins zu eins aus dem Internet kopiert haben. Anschließend haben wir im gleichen Programmteil noch ein weiteren kleinen, dennoch auf das gesamte Programm ein großen Fehler entdeckt. Dabei wurde die aufgenommene Datei zwar geladen, aber nie geschlossen und konnte daher nicht gelöscht werden, was zum Absturz des Programms geführt hat.
Nachdem diese Fehler behoben waren, stellte sich das Problem, wie man die Clicks zählt. Dabei war bei den Aufnahmen, die von einem anderen Pc kamen ein ungewöhnlicher Verlauf zu erkennen, der auf ein exponentielles Wachstum schließen lässt und haben dann eine Funktionsgleichung erstellt und diese dann umgestellt damit wir die Clicks als Ergebnis haben. Dabei ergab sich eine komplizierte Funktion mit langen Kommazahlen und Logarithmen. Als das an einen anderen Pc angeschlossen wurde, hatten wir einen anderen Verlauf. Diesmal war er linear und daher die Rechnung auch einfacher. Jetzt muss nur noch die Zahl der Attacken durch drei geteilt werden (siehe Code).

4.2 Geiger- Müller-Zähler

4.2.1 Was ist Radioaktivität[20]?

Bei Radioaktivität handelt es sich um eine Eigenschaft, bei der die Atomkerne ohne Außeneinwirkung zerfallen. Dabei wird Energie freigesetzt, die man als Strahlung bezeichnet. Man unterscheidet dabei zwischen Alpha-, Beta- und Gammastrahlung.

Bei der Alphastrahlung werden Heliumkerne ausgesandt. Diese Strahlung wird bereits durch ein einfaches Blatt Papier abgeschirmt.

Bei der Betastrahlung werden Elektronen ausgesandt, die durch ein etwa 5 Zentimeter dickes Buch bereits abgeschirmt werden.

Bei der Gammastrahlung werden hingegen der Alpha- und Betastrahlung keine Teilchen ausgesandt, sondern elektromagnetische Wellen, für dessen Abschirmung eine ein Meter dicke Betonwand benötigt wird.

Für die Strahlung gibt es drei gängige Einheiten: Becquerel, Gray und Sievert.

Becquerel ist die Aktivität und gibt an wie viele Atome pro Zeiteinheit zerfallen. Gemessen in $Bq = \frac{1}{s}$

Gray gibt die durch ionisierende Strahlung verursachte Energiedosis an und beschreibt die pro Masse absorbierte Energie. Gemessen in $Gy = \frac{J}{kg}$

Sievert ist die Äquivalentdosis und ist ähnlich wie Gray. Da aber die verschiedenen Strahlungsarten verschiedenstarke Einwirkungen haben und die Strahlung, in Gray angegeben, mit einem Faktor multipliziert(siehe Tabelle). Also wären 1 Gray Alphastrahlung 20 Sievert. Gemessen in $Sv = \frac{J}{kg}$

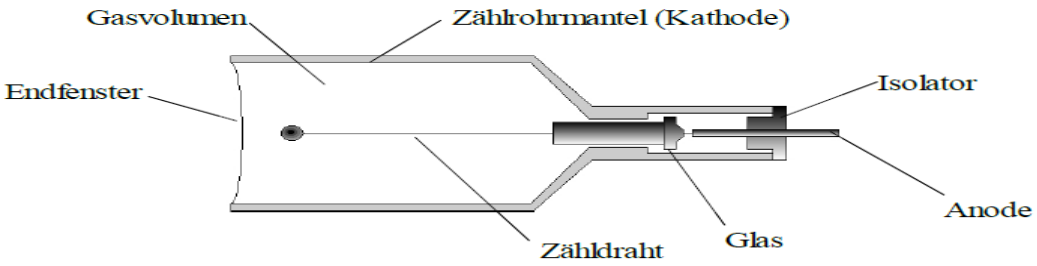
Strahlungsart	Faktor
Alpha	20
Beta	1
Gamma	1

[20]Autor: BayerischesLandesamtfürUmwelt (LfU)
URL: http://www.lfu.bayern.de/umweltwissen/doc/uw_56_radioaktivitaet_strahlung_grundbegriffe.pdf
LetzterAbruf 23.04.2014

4.2.2 Allgemeines Funktionsprinzip des Geiger-Müller-Zählers

Das allgemeine Prinzip eines Geiger- Müller- Zählrohres ist mit relativ einfacher Technik (hier vereinfacht dargestellt) konstruiert. Wir beziehen uns dabei genauer auf unser Zählrohr und dessen spezifischen Aufbautyp.

Ein Zählrohrgehäuse kann aus Glas oder Metall bestehen. Wichtig ist dabei, ob es sich um ein empfindlichen Fensterzähler, welcher auch Alphastrahlung messen kann, da die Membran dünn genug ist um die Heliumkerne nicht aufzuhalten), handelt, oder um ein weniger empfindlichen ohne Fenster. In der unteren Skizze ist ein Fenster zu erkennen, auch unser Zählkopf (LND 7313“Pancacke“ Detektor) besitzt so eins. In der großen Kammer hinter der dem Fenster befindet sich ein Gasgemisch aus Neon und Halogen, welches in dem Ursprungszustand nicht leitend ist.Wenn dann aber eine Radioaktive Strahlung auf dieses Gas trifft, trennt es andere Atome und löst eine Kettenreaktion von Zerfällen aus. Nun sind frei bewegliche Elektronen (e-) und Atomkerne(+) vorhanden, somit ist das Gas ionisiert und kann danndie außenanliegende Spannung von 540V leiten. Die Anode, an welcher sich derparallelgeschalteter Widerstand von 5,6 ^10 Ohm und der 47 pF Kondensator befinden, ist in der unteren Abbildung Mittig angeordnet, in unserem Geiger-Müller-Zähler ist diese außen angebracht. Die Kathode welche das gesamte Gehäuse (mit Aufnahme der Anode umfasst) liegt mittig angeordnet an dem aus Edelstahlgefertigtem Kopf.



[21]

Es entsteht ein Kurzschluss aufgrund dessen, da das Gas plötzlich ionisiert und keinen unüberwindbaren Wiederstand mehr darstellt. Nach dem Kurzschluss versucht das Gas sich wieder in den Ursprungszustand zu versetzen. Dadurch, dass sich in der Gaskammer ein Unterdruck befindet, kann das Neon -Halogengemisch schneller in den Ausgangszustand und hat deswegen eine geringe Tot -Zeit von 40µ Sekunden. Ein Zählkopf ist nicht unendlich lange einsetzbar, nach relativ vielen Messungen ist das Gas nicht mehr gut ionisierbar und verliert anEmpfindlichkeit bis er nicht mehr zu gebrauchen ist.

[21]URL: http://www.b-kainka.de/Geiger-M%FCller-Z%E4hler.pdf
Autor: M. Bindhammer
Geiger-Müller-Zähler

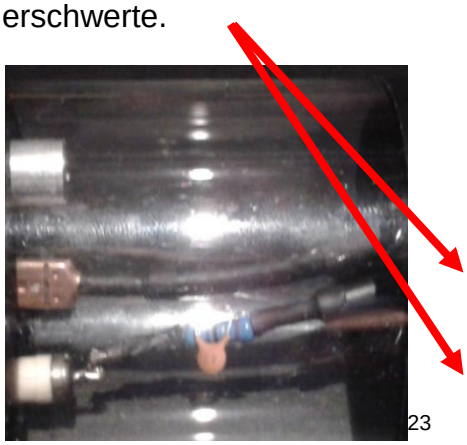
4.2.3 Bau des eigenen Zählers

Der Bau des GM- Zähler hat sich in zwei Phasen abgespielt. In der Ersten haben wir uns die Materialien sorgfältig ausgesucht und organisiert und in der Zweiten das Gerät zusammengebaut und den Kondensator, sowie den Widerstand angelötet.

Das Material des Gehäuses ist aus Plexiglas und hat einen Innendurchmesser von 54mm, in dieses passt der Fensterzählkopf mit 53,3mm sehr gut. Es musste beim reinschreiben immer darauf geachtet werden, dass das Fenster auf keinen Fall berührt oder verschmutzt wird, dieses hat den folge Schritt sehr schwer gemacht: Mithilfe von 5 kleinen Keilen haben wir das Zählrohr fixiert, zwei vorne bei dem empfindlichem Fenster und drei bei den Kontakten. Anschließend befestigten wir ein feinporiges Gitter aus Stahl vor jenes Glimmerfenster, um es vor Gegenständen und Fingern zu schützen. Unsere Wahl des dafür benötigten Materials viel auf die besonders stabilen Gitter von Audioboxen. Auf dem Folgendem Bild ist das Schutzgitter zu sehen. (bereits verbaut)

 [22]

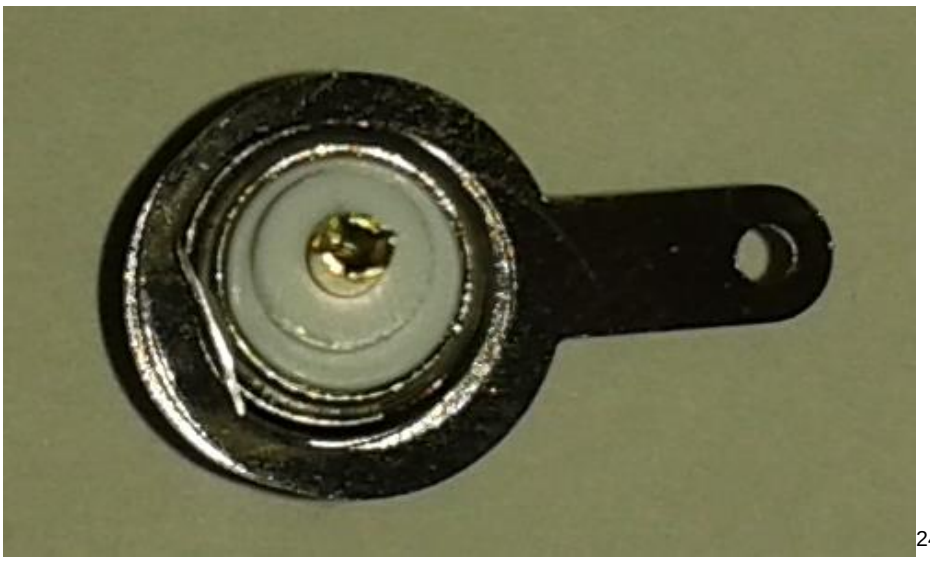
Die Kontakte der Anode, sowie der Kathode befinden sich jetzt in einem 10cm langen Rohr aus Plexiglas, was das Anschließen der Kabel mit der BNC-Buchse erschwerte.

 [23]

[22]Fotografen: Daniel Berning, Robin Stöhrmann
[23]Fotografen: Daniel Berning, Robin Stöhrmann

Im nächsten Schritt haben wir den Kondensator 47pF mit dem Widerstand 5,6*10^6Ω parallel gelötet und an die Anode angeschlossen. Das Massekabel wurde mit einem Flachstecker an die Kathode angeschlossen. (Oben im Bild gut zu erkennen)

Im unteren Bild ist die BNC-Buchse zu sehen welche mit dem Innenleben des Gerätes verbunden werden musste, zunächst aber in das vorgebohrte Loch des Plexiglasdeckels (9mm Durchmesser) eingeschraubt werden musste. Das Äußere wurde anschließend mit dem Massekabel verbunden und wie alle Kabel bei unserem Projekt sorgfältig mit Schrumpfschlauch isoliert, da es mit 540V betrieben wird und ein Kurzschluss nicht zulässig ist. Das goldfarbene ist der Anschluss für das Anodenkabel und musste auch gelötet werden.

 [24]

Nach dem zusammenbauen ist das Gerät fertig und einsatzbereit. In der Unteren Abbildung ist der Geiger- Müller- Zähler zusammengebaut zu sehen.

 [25]

[24]Fotografen: Daniel Berning, Robin Stöhrmann

4.2.4 Technische Daten des Herstellers

OCEANSIDE, NEW YORK, USA [26]

Gasfüllung	Ne + Halogen
Kathodenmaterial	Edelstahl 446
Max. Länge	31,7 mm
Effektive Länge	12,7 mm
Max. Durchmesser	53,3 mm
Effektiver Durchmesser	44,5 mm
Arbeitstemperatur	-40 °C bis +75 °C
Gewicht	125g
Massenbelegung	1,5 – 2,0 mg/cm²
Effektiver Durchmesser	44,5 mm
Material	Glimmer
Min. Anoden – Widerstand	3,3 MOhm
Empf. Anoden – Widerstand	4,7 MOhm
Min. Einsatzspannung	425 V
Empf. Arbeitsspannung	500 V
Spannungsbereich	475 – 675 V
Max. Plateausteigung	10%/100V
Min. Totzeit	40 µs
Gammaempfindlichkeit für Co60	60 Ips/mR/h
Zählrohrkapazität 3 pf	3 pf
Max. Nulleffekt, Abschirmung	50 mm Pb + 20 mm Al 30 Ipm[27]

[25]Fotografen: Daniel Berning, Robin Stöhrmann
[26]Designer: Unbekannt
URL: http://www.lndinc.com/products/391/LetzterAufruf 27.04.2014 LDN Inc Designers and Manufacturers of Nuclear Radiation Detectors
[27]Autor: Unbekannt
URL: http://www.lndinc.com/products/391/Letzter Aufruf 27.04.2014 LDN Inc Designers andManufacturersofNuclear Radiation Detectors Änderungen: Von Englisch in Deutsch, Formatierung

4.2.5 Inbetriebnahme des Geiger- Müller- Zählers

Um die Strahlung mit unserem Gerät am PC messen zu können, müssen verschiedene Zusatzkomponenten vorhanden sein. Zunächst haben wir ein selbst konstruiertes Kabel, welches einen BNC Stecker und Leyboldstecker besitzt(Auf dem Unterem Bild der größere Stecker ist der Leyboldstecker, der kleinere der BNC-Stecker.

 [28]

Durch dieses Kabel wird die nötige Versorgungsspannung 540V und der zu erfassende Impuls des Zählrohres übertragen.

[28]Fotografen: Daniel Berning, Robin Stöhrmann

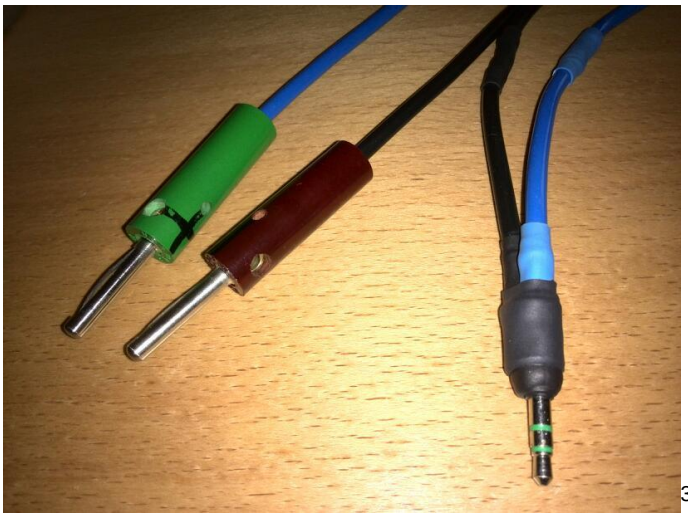
In der unteren Abbildung ist das Leyboldgerät zu sehen, dieses ist für die Versorgungsspannung und die Übertragung des zu verarbeitenden Signals zuständig.

29

Das BNC Stück wird an die BNC- Buchse des GM-Zählers angeschlossen und das andere an das Leybold Gerät (einfaches verkuppeln durch drücken und anschließend drehen).

[29]Fotografen: Daniel Berning, Robin Stöhrmann

Um das Signal von dem Gerät über die Soundkarte aufnehmen zu können, benötigt man außerdem noch einVerbindungskabel, welches durch seine Bauart aus einem Monosignal ein Stereosignal imitiert. Die Verbindung mit dem Zwischengeräterfolgt über den roten und grünen Stecker unten in der Abbildung. Wieoben im Bild bereits erwähnt werden diese Stecker in die Buchsen für die Signal Übertragung eingesteckt. Der rote Stecker ist für die Masse(-) und die grüne Steckverbindung für +, es ist auf die richtige Polung zu achten, da sonst die korrekte Auswertung des Programms nicht gegeben ist.

 [30]

Den 3,5 mm Klinkenstecker (rechts in obiger Abbildung) wird wie ein gewöhnliches aufnahmegerät in den Mikrophon-Port (rosa)der Soundkarte ggf. je nach Beschaffenheit des PCs, des Mainboards eingesteckt. Jetzt ist das Geiger- Müller-Zählrohr physikalisch verbunden.

[30]Fotografen: Daniel Berning, Robin Stöhrmann

5 Projektziele in der Rückbetrachtung

Nach der Fertigstellung unseres Projektes kann man sagen wir haben vieles erreicht, aber nicht alles. Wir konnten nicht in Becquerel (Aktivität Zerfall/Sekunde) messen, da dies eine Komplizierte Rechnung nach sich zieht und bei verändertem Abstand von der Strahlungsquelle zum Fenster eine andere Rechnung mit $1/r^2$ abnehmender Strahlungsmenge zu tun hat. Zudem breitet sich Strahlung Kugelförmig aus und wir müssten bis zum Zähler den Abstand messen die Oberfläche der Kugel berechnen und die Oberfläche des Zählrohrfensters abziehen (Die Fläche des Zählrohres selbst ist eine keine gekrümmte Ebene wie bei der Kugel, deshalb immer noch ungenau). Das Messen in Becquerel ist ein Projekt an sich, es müsste noch ein Abstandsmesser integriert werden und der müsste auch durch C# ausgelesen werden.

Das Zählrohr befindet sich in einem selbst gebauten Gehäuse, ist besser als selbst angenommen erreicht worden und die Beschaltung hat auch geklappt (Beim ersten Mal war die Polung verkehrt herum, wurde durch Herr Kantners Aufmerksamkeit aber wieder korrigiert, bevor es zum Einsatz kam).

Unser Signal wird durch einen Leiter zum Leybold- Gerät geleitet und dann direkt zum Mikrophon Eingang geswitcht. Somit ist auch dieses Vorhaben gelungen.

Die Messungen können mithilfe des C# Programms ausgelesen und als Diagramm angezeigt werden und man kann eine beliebige Länge messen lassen, aber gleichzeitig wird eine aktuelle Strahlenhöhe im Sekundentakt wiedergegeben. Außerdem wird zu jeder Messung ein automatischer Protokolleintrag eingetragen. Während unserer IT und Technikmesse wurden so genau 100593 Clicks festgestellt.

Auch unsere beiden Geigerzähler funktionieren, erst war es unser Auftrag nur ein Zähler zu bauen, jedoch wurden wir eines besseren belehrt und haben zwei gebaut, somit haben wir auch dieses Ziel erfüllt.

Alles in allem hat uns dieses Projekt Spaß gemacht, auch wenn es eine Herausforderung war, wir sind der Meinung, dass wir trotzt Schwierigkeiten unser Projekt geschafft haben. Während der gesamten Zeit, die wir an diesem Projekt gearbeitet haben, haben wir viel über den Geiger-Müller-Zähler, Radioaktivität und auch über die Programmiersprache Microsoft C# gelernt.

6 Quellenverzeichnis

[1]Autor: Unbekannt
URL:
www.strahlenschutzkurse.de/fileadmin/strahlenschutzkurse/dokumente/lehrer/grflzaro.pdf Letzter Abruf 25.04.2014 Seminar Esslingen Strahlenschutz
[5]Autor: Maximilian Müller
http://dotnet-snippets.de/snippet/wave-dateien-ueber-mci-aufnehmen/1594
Letzter Abruf 22.03.2014
[6]Autor: Oliver S
http://csharp-tricks.blogspot.de/2011/03/wave-dateien-einlesen.html
Letzter Abruf 23.03.2014
[10]Autor: Bayerisches Landesamt für Umwelt (LfU)
http://www.lfu.bayern.de/umweltwissen/doc/uw_56_radioaktivitaet_strahlung_grundbegriffe.pdf
Letzter Abruf 23.04.2014

Alle Bilder, welche keine gesonderte Quelle haben, sind von uns aufgenommen worden und die Rechte liegen bei Daniel Berning und Robin Stöhrmann