# Anhang

## A. Dokumentation der entwickelten Programme

Die im Rahmen dieser Arbeit entwickelten Programme sind über das Github-Repository LINK zugänglich. Zur besseren Übersicht sind die Programme in zwei Ordner aufgeteilt. Im Ordner *Measurement* befindet sich das Programm zur Steuerung der Messungen mit der DAQ-Karte, sowie alle Skripte, welche dadurch aufgerufen werden. Im Ordner *Analysis* befinden sich die Programme zur Verarbeitung und graphischen Dartstellung der gemessenen Daten. Alle Programm sind in der Programmiersprache Python geschrieben und verwenden die in Tab. A.1 aufgelisteten Bibliotheken.

Tabelle A.1.: Liste der in den entwickelten Programmen verwendeten Bibliotheken.

| Bibliothek | Verwendung |
|---|---|
| math | Mathematische Funktionen |
| time | Zeitoperationen |
| sys | Systemspezifische Parameter und Funktionen |
| os | Betriebssystem-Funktionen |
| serial | Ansteuern der seriellen Schnittstelle |
| argparse | Parser für Kommandozeilenargumente |
| numpy | Numerische Funktionen |
| matplotlib | Graphische Darstellung |
| kafe | KArlsruher Fitting Environment |

### A.1. Skripte zur Steuerung der einzelnen Messungen

In Tab. A.2 sind die einzelnen Skripte im *Measurement*-Ordner beschrieben. Zur Durchführung einer Messung ruft man jeweils *daq.py* mit den in Tab. A.3 erläuterten Parameter auf.

Die Ausgabe der Messergebnisse unterscheidet sich je nach Art der Messung.

**Ratenmessung:** Wird anhand der Parameter die graphische Ausgabe aktiviert, so werden die Messergebnisse laufend in einem Graphen aufgetragen und aktualisiert. Andernfalls wird die Anzahl der Ereignisse sowie am Ende der Messung die Rate auf der Kommandozeile ausgegeben.

**Messung des Pulshöhenspektrums:** Bei der Pulshöhenspektrumsmessung wird für jede Messung die eingestellte Schwellenspannung sowie die Messzeit in Sekunden und die Anzahl der registrierten Ereignisse in eine Textdatei geschrieben.

**Lebensdauermessung:** Bei der Lebensdauermessung wird die gesamte erweiterte Ausgabe der DAQ-Karte in eine Textdatei geschrieben.

Tabelle A.2.: Programme im *Measurement*-Ornder.

| Programm | Aufgabe |
|---|---|
| daq.py | Ausführbares Programm, welches anhand der übergebenen Parameter eine Messung durchführt |
| daqclass.py | Klassenimplementierung einer DAQ-Karte mit Steuerungsfunktionen |
| rate.py | Steuerung einer Ratenmessung mit der Möglichkeit zur Visualisierung der Daten während der Laufzeit |
| schwellenmessung.py | Steuerung einer Messung des Pulshöhenspektrums |
| lebensdauer.py | Steuerung einer Lebensdauermessung |

Wenn das Skript die gemessenen Daten in eine Textdatei schreibt, so werden diese in den angegebenen Ordner gespeichert. Um die Textdateien anschließend zuordnen zu können, besteht deren Name aus der Art der Messung und dem Zeitpunkt an dem die Messung gestartet wurde. Die Nomenklatur dabei ist Folgende: `Messart_YYYY_MM_DD_HH_MM.dat`

Zur Durchführung einer Ratenmessung von einer Stunde, bei der Dreifachkoinzidenzen gezählt und graphisch dargestellt werden, benutzt man folgenden Kommandozeilenbefehl:

```
python daq.py -M rate -t 3600 -T 3 -g
```

Folgender Kommandozeilenbefehl startet eine Messung des Pulshöhenspektrums, bei der die Schwellenspannung von 10mV − 100mV jeweils in 10mV-Schritten durchiteriert wird. Dabei wird bei jeder Schwellenspannung entweder solange gemessen, bis 50000 Ereignisse registriert worden sind oder bis zu maximal einer Stunde Messzeit.

```
python daq.py -M threshold -t 3600 -n 50000 -s 10 -e 100 -i 10
```

Um eine zehnstündige Lebensdauermessung durchzuführen, welche die Daten im Ordner *Lebensdauer* speichert und am Eingang 0 als Schwellenspannung 85mV und 250mV an den Eingängen 1 und 2 benutzt, verwendet man folgenden Befehl:

```
python daq.py -M lifetime -p ./Lebensdauer -t 36000 -S 85 250 250
```

## A.2. Skripte zur Analyse der gemessenen Daten

Im *Analysis*-Ordner befinden sich die Skripte zur Analyse und Darstellung der Daten, welche mit dem oben beschriebenen Skript und der DAQ-Karte gemessenen wurden.

### A.2.1. Analyse des Pulshöhenspektrums

Das Skript *plot_threshold.py* benötigt als Eingabeparameter die Dateinamen von einer Pulshöhenspektrumsmessung mit voller und einer mit leerer Kanne. Optional kann man auch Start- und Endpunkt der x-Achse angeben:

```
python plot_threshold.py -f volle_kanne.dat -e leere_kanne.dat -x 0 500
```

Dabei gilt zu beachten, dass beide Messungen bei gleichen Schwellenspannungen stattgefunden haben, da das Skript sonst den Rauschanteil bei einer bestimmten Schwellenspannung nicht bestimmen kann. Nachdem das Skript die Raten für die einzelnen Messpunkte der beiden Messungen bestimmt hat, werden, zur Vergleichbarkeit der beiden Messreihen,

Tabelle A.3.: Erklärung der Parameter und deren Default-Werte von *daq.py*.

| Parameter | Datentyp | Default | Art der Messung | Erklärung |
|---|---|---|---|---|
| -d | String | "/dev/ttyUSB0 " | alle | Name der Schnittstelle, an der die DAQ-Karte angeschlossen wird. |
| -M | String | "threshold" | alle | Art der durchzuführenden Messung (Ratenmessung = "rate", Pulshöhenspektrumsmessung="threshold", Lebensdauermessung="lifetime"). |
| -p | String | "./Data" | alle | Name des Ordners in dem die Daten gespeichert werden sollen. |
| -t | Integer | 10 | alle | Laufzeit in Sekunden für eine Messung, bzw. pro Messintervall beim Pulshöhenspektrum |
| -g | - | - | Ratenmessung | Aktiviert die graphische Darstellung während der Messung. |
| -T | Integer | 3 | Ratenmessung | Trigger der verwendet werden soll. (Einfach-, Zweifach- oder Dreifach-Trigger) |
| -S | Integer | [100,100,100] | Ratenmessung, Lebensdauer | Schwellenspannungen für die einzelnen Eingänge der DAQ-Karte |
| -n | Integer | 10000 | Pulshöhenspektrum | Anzahl der maximal zu messenden Ereignisse bei einem Intervall der Pulshöhenspektrumsmessung. |
| -s | Integer | 500 | Pulshöhenspektrum | Startwert für die Schwellenspannung bei der Pulshöhenspektrumsmessung. |
| -e | Integer | 1000 | Pulshöhenspektrum | Endwert für die Schwellenspannung bei der Pulshöhenspektrumsmessung. |
| -i | Integer | 100 | Pulshöhenspektrum | Intervall zwischen zwei Schwellenspannungen bei der Pulshöhenspektrumsmessung. |

die Raten normiert. Anschließend wird der Rauschanteil für jeden Datenpunkt bestimmt und graphisch ausgegeben.

Nach der Bestimmung des integralen Pulshöhenspektrums wird das differentiale Pulshöhenspektrum bestimmt und ebenfalls ausgegeben.

### A.2.2. Analyse der Lebensdauermessung

Die Analyse der Lebensdauermessung besteht aus drei Schritten. Im ersten Schritt werden aus der erweiterten Ausgabe der DAQ-Karte sämtliche Informationen zu den Pulsen herausgeschrieben. Im zweiten Schritt werden die Daten auf Doppelpulse untersucht, deren Zeitdifferenz bestimmt und anschließend in eine vorgegebene Anzahl an Intervallen aufgeteilt, bevor sie im dritten Schritt graphisch dargestellt werden und die Lebensdauer ermittelt wird.

Das Skript *extract_pulses.py* untersucht die Ausgabe der DAQ-Karte auf registrierte Pulse. Zuerst werden alle Daten von einem Ereignis gesammelt und anschließend auf Pulse in diesem Ereignis untersucht. Dabei werden die Start- und Endzeitpunkte der Pulse, unter Angabe an welchem Eingang sie registriert worden sind, mit einer absoluten Zeitmarke[1] versehen. Das Skript wird mit dem Befehl `python extract_pulses.py -f messung.dat` aufgerufen und die Daten werden anschließend in folgendem Format in eine neue Datei geschrieben:

```
#EndEvent
0 145.07311828250 145.07311829625
1 145.07311827125 145.07311831375
#EndEvent
0 145.32484073625 145.32484074125
#EndEvent
0 145.90453236875 145.90453237875
1 145.90453235375 145.90453240500
0 145.90453836500 145.90453837750
#EndEvent
```

Dass die Weiterverarbeitung der Daten in ein separates Skript ausgelagert wurde, beruht auf der Tatsache, dass bisher keine notwendigen Informationen verloren gegangen sind. Will man eine Selektion der Daten (bspw. anhand der Anzahl der Pulse pro Ereignis) durchführen, so kann man anhand von einem eigens dafür geschrieben Skript mit diesem Datenformat wesentlich einfacher als mit der Ausgabe der DAQ-Karte arbeiten.

Als zweiter Schritt wird obiges Datenformat vom Skript *double_pulse_difference.py* eingelesen und auf Doppelpulse am Eingang 0 untersucht. Von denen wird jeweils die Zeitdifferenz bestimmt bevor sie anschließend in eine über einen Parameter angegebene Anzahl von Bins aufgeteilt werden (`python double_pulse_difference.py -f pulsdaten.dat -n 50`). Der Default-Wert sind 100 Bins. Anschließend wird die Anzahl der Doppelpulse pro Bin mitsamt Fehler in eine neue Datei geschrieben, sodass das Format von KaFE eingelesen werden kann.

Da dieses Skript, aufgrund der Analyse der großen Datenmengen eine längere Laufzeit hat und es im Gegensatz zu der anschließenden Analyse der Lebensdauer in der Regel nur einmal ausgeführt wird, wurde dieses Skript ebenfalls in eine separate Datei ausgelagert.

Mit `python plot_lifetime.py -f doppelpulsdaten.dat` werden die angegebenen Daten von KaFE eingelesen und geplottet. Anschließend wird der Anwender gefragt, ab welchem Punkt die Daten verwendet werden sollen, um die die Lebensdauer zu ermitteln. Daraufhin wird eine Exponentialfunktion an die angebenen Daten angepasst und die ermittelten Daten sowie ein Plot der Daten mitsamt der angepassten Funktion ausgegeben.

[1] Die hier verwendete Zeitmarke sind die Sekunden, welche seit dem letzten Zählerüberlauf vergangen sind. Aufgrund der Tatsache, dass die Pulse jeweils pro Ereignis abgespeichert werden und die Rate in der Regel nicht unter 1 Ereignis pro 170s fällt, stellen Zählerüberläufe bei der Ermittlung der Zeitdifferenzen der Doppelpulse in der Regel keine Probleme dar.