

# **Komplexpraktikumsbericht**

Latchups in der Raumfahrt: Eine Datenanalyse basierend  
auf Ergebnissen von Experimenten mit  
Teilchenbeschleunigern und Lasern

Verfasst von

Thomas Schmidt, B.Eng.

Betreuer:

Prof. Dr.-Ing. Burkart Voß

Dr.-Ing. Hannes Zöllner

Vorgelegt am: 22.09.2023

# Inhaltsverzeichnis

Abkürzungen.....	3
1. Einleitung.....	4
2. Allgemeines.....	5
2.1. Übersicht über den Aufbau der Daten und die Dokumentation des Beschleunigerexperiments .....	5
2.2. Übersicht über den Aufbau der Daten und die Dokumentation des Laser-Fullscan- Experiments.....	6
2.3. Vorbereitungen.....	6
3. Übertragen der SEL auf Einzelbilder.....	8
3.1. Extrahieren der SEL-Koordinaten aus den *.dat.txt Files.....	8
3.2. Einzeichnen der SEL auf den Bildern.....	9
4. Ansätze zum Zusammenfügen zu einem Panorama.....	15
4.1. Hugin.....	15
4.2. Python – Ansatz über Homographiematrizen.....	16
4.3. Manuelles Zusammenfügen.....	17
5. Anpassen der Daten des Laserexperiments.....	18
5.1. Panoramas zusammensetzen.....	18
5.2. Größe ändern.....	18
5.3. Überlappende Pixel löschen.....	21
5.4. CSV-Dateien zusammenfügen.....	21
6. Berechnung der Wirkungsquerschnitte.....	22
6.1. Ionen – Daten.....	22
6.2. Laser – Daten.....	22
7. Auswertung.....	23
7.1. Plausibilitätsprüfung.....	23
7.2. Auswertung der Wirkungsquerschnitte.....	25
8. Zusammenfassung.....	26

## Abkürzungen

ADC	–	Analog-/Digitalwandler
CMOS	–	Complimentary Metal Oxide Semiconductor
CSV	–	Comma Seperated Values
CV	–	Computer Vision
DUT	–	Device Under Test
FOSS	–	Free Open Source Software
GeV	–	Giga Elektronenvolt
GSI	–	Gesellschaft für Schwerionenforschung
LUNTE	–	Latchup UNTERSuchungen an Elektronikkomponenten
PNPN	–	beschreibt Aufbau und dotierung von Halbleiterbauelementen
SEE	–	Single-Event-Effects
SEL	–	Single Event Latchup
SF	–	Skalierungsfaktor

# 1. Einleitung

Luft- und Raumfahrtanwendungen sind nehmen in den letzten Jahren eine immer wichtigere Rolle ein. Sie ermöglichen Technologien wie globale Navigationssatellitensysteme (z. B. GPS), Satellitenfernsehen oder Satellitentelefone sowie Forschungsanwendungen wie Erdbeobachtungen oder Weltraumerkundungen z. B. durch das James-Webb-Weltraumteleskop.

Ein integraler Bestandteil vieler Satelliten sind Mikrocontroller zum Steuern verschiedener Prozesse. Diese sind in der Regel in Standard-CMOS-Halbleitertechnik aufgebaut, in denen PNPN-Strukturen, so genannte Thyristoren, auftreten. Diese haben unter normalen Umständen keine Funktion, da kein Mechanismus zum Zünden vorhanden ist. In strahlungsreichen Umgebungen wie z. B. im Weltall kann ein hochenergetisches Partikel genügend freie Ladungsträger erzeugen um einen solchen parasitären Thyristor zu zünden und einen dauerhaften Stromfluss hervorzurufen, der zur thermischen Zerstörung des Bauteils führen kann. Dieser Effekt wird Single Event Latchup (SEL) genannt. [1]

Um dem Problem zuvorzukommen, werden Mikrocontroller für Weltraumanwendungen auf ihre Empfindlichkeit gegenüber SEL getestet. Dies findet bisher in teuren, aufwendig herzustellenden Teilchenbeschleunigern statt. Ein alternativer Ansatz wurde an der Ernst-Abbe-Hochschule im Rahmen des Projekts LUNTE (Latchup UNTERSuchungen an Elektronikkomponenten) entwickelt. Hier werden SEL mithilfe des Lasers eines handelsüblichen DVD-Laufwerks erzeugt. Um die Ergebnisse für die Raumfahrt besser nutzbar zu machen, muss ermittelt werden, wie die Pulsenergie des Lasers mit der Energie der Ionen in einem Teilchenbeschleuniger korreliert. Hierfür wurde ein Experiment mit einem Teilchenbeschleuniger am GSI Helmholtzzentrum für Schwerionenforschung durchgeführt. In ihm wurde ein Mikrocontroller mit Goldionen der Energie 1,2 GeV beschossen und erzeugte SEL detektiert. Dieser Bericht beschreibt die Auswertung der bei diesem Experiment generierten Daten, und deren Vergleich mit den Daten aus dem LUNTE-Experiment.

## 2. Allgemeines

Der Datensatz besteht aus Bildern und Textdateien, die in mehrere Scanbereiche aufgeteilt sind.

### 2.1. Übersicht über den Aufbau der Daten und die Dokumentation des Beschleunigerexperiments

#### 2.1.1. Mikroskopbilder:

Die Bilder sind nach dem Schema "Jenamicro18\_fs<Spalte><Zeile>.tif" (z.B. Jenamicro18\_fs12.tif) benannt. Sie bestehen aus einer Serie von Mikroskopbildern, die zu den jeweiligen Scanbereichen korrespondieren.

#### 2.1.2. Textdateien:

Die Textdateien sind nach dem Schema "micfs<Spalte><Zeile>.dat.txt" (z.B. "micfs11.dat.txt" ) benannt. Sie enthalten Koordinaten, die mit Ionen beschossen wurden. Das Format beinhaltet:

- a) Die ersten 6 Zeilen bilden das Kopfsegment (Metadaten zum Experiment).
- b) Jede Zeile repräsentiert die spezifischen Koordinaten, die beschossen werden, mit 5 Werten:
  - 1. x-Koordinate in Bezug auf den entsprechenden Scanbereich basierend auf einem 12-Bit-ADC (abweichend von den Bildkoordinaten).
  - 2. y-Koordinate in Bezug auf den entsprechenden Scanbereich basierend auf einem 12-Bit-ADC (abweichend von den Bildkoordinaten).
  - 3. Latchup (0 - kein SEL, 4 - SEL).
  - 4. möglicher ADC-Eingang (in diesem Fall nicht zutreffend).
  - 5. Der Channeltron-Wert (korreliert zu der Anzahl der emittierten Elektronen).

#### 2.1.3. Datenorganisation:

Die Daten sind auf 25 Scanbereiche verteilt, die jeweils ein Bild und eine \*.txt-Datei enthalten.

#### 2.1.4. Dokumentation

Für die Bilder mit 5-facher Vergrößerung sind folgende Größeninformationen bereitgestellt:

- $x: 597 \text{ px} = 900 \text{ }\mu\text{m}$
- $y: 461 \text{ px} = 700 \text{ }\mu\text{m}$

Größeninformationen zur Ionenablenkung:

- $x = 9 * 33,7 \text{ }\mu\text{m} = 303,3 \text{ }\mu\text{m}$
- $y = 9 * 35,7 \text{ }\mu\text{m} = 321,3 \text{ }\mu\text{m}$

Diese Übersicht soll ein besseres Verständnis der Datenstruktur für die Mikroskopbilder und Textdateien vermitteln, die nach den oben aufgeführten Spezifikationen organisiert sind, und so eine effiziente Nutzung und Analyse der Daten ermöglichen.

## 2.2. Übersicht über den Aufbau der Daten und die Dokumentation des Laser-Fullscan-Experiments

Der Datensatz besteht aus Bildern und CSV-Dateien, die jeweils einen von drei sich überlappende vertikalen Streifen darstellen. Diese Daten sind im Gegensatz zu den Ionendaten schon aufeinander abgestimmt. Der Mikrochip wird in vertikalen Streifen gescannt. Diese Streifen müssen an die Größe der Ionendaten angeglichen und danach zusammengesetzt werden.

## 2.3. Vorbereitungen

Um die bereitgestellten \*.dat-Dateien des Ionenexperiment in Textdateien umzuwandeln, wird das Perlskript "extract\_dat2txt.pl" verwendet, das vom GSI Helmholtzzentrum bereitgestellt wurde. Die Benutzung des Skripts erfolgt in einem ordnungsgemäß definierten Verzeichnis, um die korrekte Funktion des Programms zu gewährleisten.

Die Ordnerstruktur für die Daten des Ionenexperimentes ist wie folgt:

1. Hauptordner, in dem sich die Pythonskripte befinden:

- *fun3.py*
- *LatchUpMap\_v2\_3.py*
- *EmptyLatchUpMap\_v0\_0.py*
- *removeStripes.py*

## 2. Unterordner innerhalb des Hauptordners:

### a) "pics" für die Bilder:

Mikroskopbilder des DUT

### b) "docs" für die umgewandelten Textdateien:

Hier werden die Textdateien gespeichert, die aus den ursprünglichen \*.dat Dateien mithilfe von "extract\_dat2txt.pl" generiert wurden.

### c) "img" für die fertigen Bilder:

Hier werden die Ergebnisse des Skriptes "*LatchUpMap\_v2\_3.py*" gespeichert.

## Die Ordnerstruktur für die Daten des Laserexperiments ist wie folgt:

### 1. Hauptordner, in dem sich die Pythonskripte befinden:

- *binningFun\_v0\_6.py*
- *CombineCSV.py*
- *pixelBinning\_v0-6.py*
- *Gsi2csv\_v0-0.py*

### 2. Unterordner innerhalb des Hauptordners:

#### a) "csv" für die Laser CSV-Daten

#### b) "panos" für die Panoramas in original Größe

#### c) "pics" für die Bilder aus den Laser Daten

Um die volle Funktionsfähigkeit des Programms zu garantieren, ist diese Ordnerstruktur strikt einzuhalten. Durch die richtige Organisation können die verschiedenen Skripte und Dateien effizient verarbeitet und genutzt werden.

## 3. Übertragen der SEL auf Einzelbilder für das Ionenexperiment

### 3.1. Extrahieren der SEL-Koordinaten aus den \*.dat.txt-Files

Um die SEL auf die Mikroskopbilder zu projizieren, müssen die Koordinaten eingelesen und sortiert werden. Dazu werden durch die Software folgende Schritte ausgeführt:

#### 1. Erstellung der Listen mit relevanten Dokumenten:

Es werden zwei separate Listen für die in den Verzeichnissen "docs" und "pics" gefundenen relevanten Dokumente erstellt. Diese haben die Dateiendungen ".dat.txt" oder ".tif".

#### 2. Ablauf für jedes relevante Dokument in "docs":

Für jedes im Schritt 1 identifizierte relevante Dokument im Verzeichnis "docs" wird folgender Ablauf abgearbeitet:

##### *a) Zeilenweise einlesen:*

Das Dokument wird zeilenweise eingelesen. Jede Zeile wird anschließend einzeln verarbeitet.

##### *b) Kopfsegment löschen:*

Das Kopfsegment, also die ersten 6 Zeilen mit Metadaten, wird in diesem Schritt entfernt.

##### *c) Zeile in 5 Elemente aufteilen:*

Die verbleibenden Zeilen des Dokuments werden nun entsprechend dem Aufbau der Daten in fünf separate Elemente aufgeteilt. Damit wird eine Arraystruktur geschaffen, die es ermöglicht, die Informationen später leichter weiterzuverarbeiten.

##### *d) Prüfung und Speicherung von Latchups:*

In diesem Schritt wird überprüft, ob ein SEL für die jeweilige Zeile vorliegt. Wenn dies der Fall ist, wird die Zeile in der Liste "LatchUps" gespeichert. Diese Liste dient als Sammlung der Latchupevents, die in den verschiedenen Dokumenten gefunden wurden.

Zusammengefasst besteht der Prozess also aus der Erstellung zweier Listen mit relevanten Dokumenten und der Durchführung bestimmter Operationen wie dem Einlesen, Löschen von Headern und Aufteilung von Zeilen für jedes relevante Dokument im Verzeichnis "docs". Ziel ist es, Latchup-Events extrahieren und in einer separaten Liste zur Weiterverarbeitung zu speichern.

## 3.2. Einzeichnen der SEL auf den Bildern

### 3.2.1. Erster Ansatz

Nachdem die SEL aus den Textdateien extrahiert wurden, können sie nun auf die Mikroskopbilder projiziert werden. Dazu führt die Software folgende Schritte aus:

1. Finden des korrespondierenden Bildes:

Basierend auf dem Namen des \*.dat.txt-Dokuments wird das passende Bild ausgewählt.

2. Öffnen des Bildes mit OpenCV:

Nachdem das korrespondierende Bild gefunden wurde, wird es mit der OpenCV-Bibliothek geöffnet. OpenCV ermöglicht die Berechnung der Dimensionen des Bildes und konvertiert es in ein Numpyarray zur weiteren Verarbeitung. Numpyarrays sind Datenstrukturen in der Programmiersprache Python.[6][5]

3. Berechnung der SEL-Koordinaten:

Für jeden SEL im Datensatz werden die genauen Koordinaten im Bild berechnet. Dies bezieht sich auf den Scanbereich des Bildes und liefert die Positionsdaten für die anstehenden Transformationen. Dazu werden die folgenden Formeln verwendet:

$$P_x = \frac{d_x}{ADC} x_{ADC}$$

$$P_y = \frac{d_y}{ADC} y_{ADC}$$

P – Position im Scanbereich in Pixeln  
d – Ablenkung

$x_{ADC}/y_{ADC}$  – Position im Scanbereich  
bezogen auf ADC  
ADC – 12 Bit (4096)

4. Umrechnung der Positionen in Bildkoordinaten:

Die zuvor berechneten Positionen werden in Bildkoordinaten umgerechnet. Dazu werden folgende Schritte durchgeführt:

1. Spiegelung an der y-Achse: Die Koordinaten werden so angepasst, dass sie symmetrisch bezüglich der y-Achse gespiegelt werden
2. Drehung um 90° gegen den Uhrzeigersinn: Der Koordinatenvektor wird um 90° gegen den Uhrzeigersinn gedreht.
3. Koordinatentransformation: Transformation von Scanbereichskordinaten in Bildkoordinaten

$$x_{Bild} = P_{x, neu} + \frac{Höhe}{2} - \frac{d_y}{2} \quad y_{Bild} = P_{y, neu} + \frac{Breite}{2} - \frac{d_x}{2}$$

5. Ändern der Pixel im Numpyarray:

Innerhalb des Bildes (Numpyarray) werden die Farbwerte der Pixel für SEL auf den Farbcode [0, 0, 255] (rot) gesetzt. Optional können auch die Farbwerte der Pixel für den Rahmen auf [255, 0, 0] (blau) gesetzt werden, um die Analyse zu vereinfachen.

6. Speichern des modifizierten Bildes:

Schließlich wird das veränderte Bild mithilfe von OpenCV im Ordner "LatchUps" gespeichert. Dadurch ist das Bild für die spätere Analyse und Verwendung verfügbar.

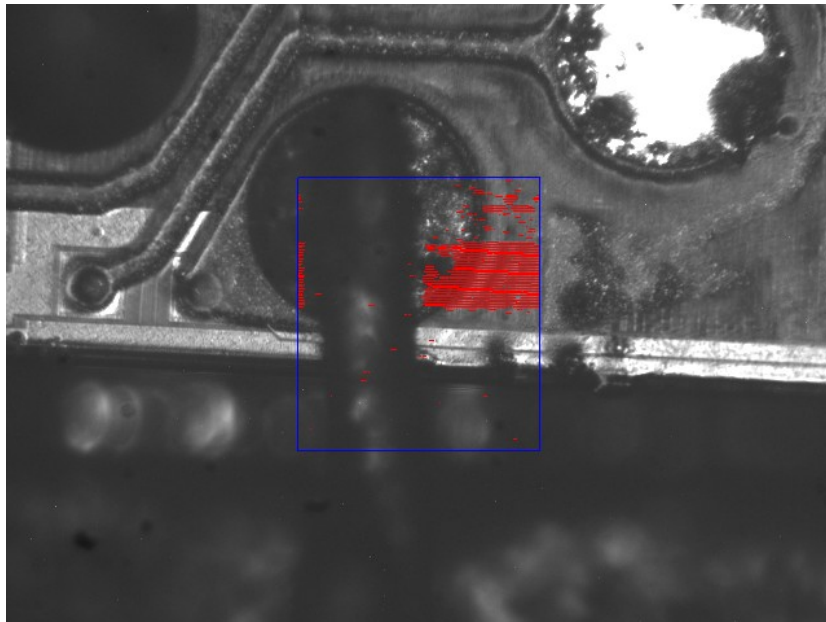


Abbildung 1: Jenamicro18\_fs35\_LatchUpMap.png, Beispiel Zeilenbildung

Hier fällt auf, dass Bereiche, die komplett empfindlich sein sollten, Zeilen ausbilden.

### 3.2.2. Zweiter korrigierter Ansatz

Die Zeilenbildung beim Eintragen der SEL ist darauf zurückzuführen, dass die SEL-Daten an die Größe der Bilder angepasst werden. Die bei dem Experiment abgetasteten Zeilen sind nicht horizontal, sondern haben eine kleine vertikale Komponente. Um diesem Effekt entgegenzuwirken, müssen die Koordinaten durch die durchschnittliche Anzahl der leeren Zeilen zwischen zwei besetzten Zeilen (34,4) geteilt werden. Dieser Faktor wird durch Analyse der Rohdaten bestimmt.

Dafür wurde ein neuer Ansatz entwickelt. Die ersten zwei Schritte sind identisch zum oben beschriebenen Vorgehen.

1. analog erster Ansatz
2. analog erster Ansatz
3. Bildgröße ändern:

Um die Größe des Bildes anpassen zu können, muss zuerst ein Skalierungsfaktor (SF) berechnet werden. Dieser wird aus der Größe des aktiven Fläches auf den Originalbildern und der Größe des aktiven Fläches auf den neuen Bildern berechnet.

$$SF_{Bild,x} = \frac{x \text{ Ausdehnung Aktiver Bereich auf Originalbildern}}{x \text{ Ausdehnung Aktiver Bereich auf neuen Bildern}} = \frac{201}{120} \approx 0.6$$

$$SF_{Bild,y} = \frac{y \text{ Ausdehnung Aktiver Bereich auf Originalbildern}}{y \text{ Ausdehnung Aktiver Bereich auf neuen Bildern}} = \frac{211}{120} \approx 0.57$$

Mit diesen Faktoren kann nun die neue Größe des Bildes berechnet werden und das Bild mithilfe der Pythonfunktion `cv2.resize()` verkleinert. Als Interpolationsmethode wird hierbei `cv2.INTER_AREA` verwendet, da diese die bevorzugte Methode zum Verkleinern von Bildern ist.[6]

4. Erstellen einer SEL-Map:

Als Grundlage dient ein Array mit den Dimensionen der aktiven Fläche (120x120 Pixel). Dieses wird mit schwarzen Pixeln gefüllt und ein blauer Rand wird eingefügt.

Um die SEL in das Array einzufügen, müssen die aus dem Textdokument extrahierten Koordinaten an die Dimensionen des neuen Arrays angepasst werden. Dies erfolgt durch eine Division mit dem Skalierungsfaktor. Die Daten sind im Bezug auf die Bilder gespiegelt und verschoben. Um sie kombinierbar zu machen, müssen die Daten also noch an der y-Achse gespiegelt werden und um 90° in mathematisch positive Richtung gedreht werden.

Mit den angepassten Koordinaten können die SEL als rote Pixel in das Array eingefügt werden.

5. Zusammenfügen von SEL-Map und Bildern

Das Zentrum von Bild und zugehöriger SEL-Map müssen übereinander liegen. Um diese zusammenfügen zu können, muss also zuerst ein Versatz in x und y Richtung berechnet werden um die Zentren aufeinander auszurichten.

$$x_0 = \text{trunc}\left(\frac{B_{Bild} - B_{SEL-Map}}{2}\right), y_0 = \text{trunc}\left(\frac{H_{Bild} - H_{SEL-Map}}{2}\right)$$

B – Breite

H – Höhe

$x_0, y_0$  – Versatz

Nun erfolgt eine Iteration durch die Elemente der SEL Map. Wenn das Pixel nicht schwarz ist, wird das korrespondierende Pixel im Bild mit dem Wert überschrieben.

#### 6. Speichern des modifizierten Bildes

Schließlich wird das veränderte Bild mithilfe von OpenCV im Ordner "img" gespeichert. Dadurch ist das Bild für die spätere Analyse und Verwendung verfügbar.

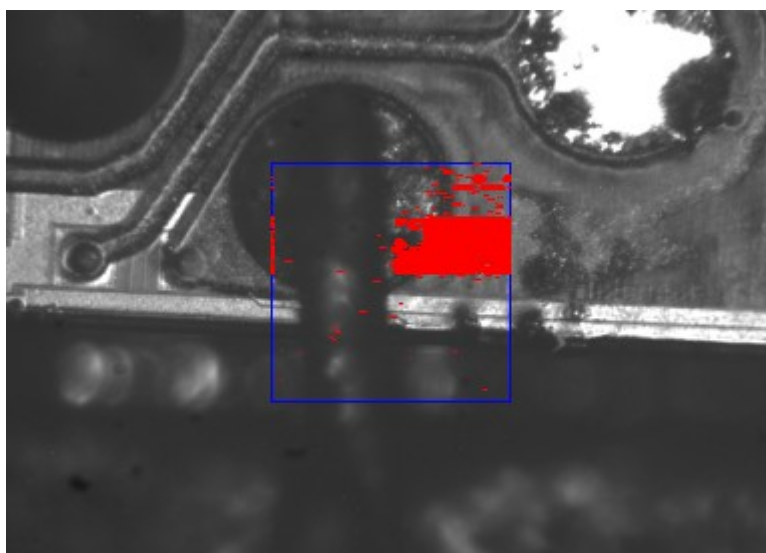


Abbildung 2: micfs35.dat.txt\_overlay.png

Bei diesen Bildern kommt es nun nicht mehr zur Zeilenbildung wie beim vorherigen Ansatz.

Es fällt in den Bildern allerdings ein Hang zur "Streifenbildung" auf.

### 3.2.3. Dritter korrigierter Ansatz

Die Streifenbildung im Bild ist durch einen systematischen Fehler im Experiment zu erklären. Wenn ein SEL detektiert wird, wird der Mikrocontroller kurz ausgeschaltet (1 ms). Es können in dieser Zeit also keine weiteren SEL auftreten. In dem Experiment wurden jedoch alle Ionen, die in diesem Zeitraum auf den Chip treffen, als SEL detektiert. Dadurch werden Ereignisse, die eigentlich nur auf einem Pixel stattfinden, über mehrere Pixel "verschmiert".

Um diesen Fehler zu beseitigen, müssen die Daten bearbeitet werden. Dazu wurde das Skript "removeStripes.py" entwickelt. In ihm werden die Dokumente im Ordner "docs" eingelesen. Wenn ein SEL erkannt wurde, werden die folgenden 10 Zeilen auf "0\t0\t0\t0\t0\n" gesetzt und die fälschlicherweise identifizierten SEL so gelöscht. Die Zeile mit dem erkannten SEL wird jetzt in einer anderen Liste im Unterordner "remStrips" gespeichert. Dies muss vor allen anderen Schritten ausgeführt werden.

Mit diesem bearbeiteten Dokument können die Bilder mit den eingezeichneten SEL generiert werden.

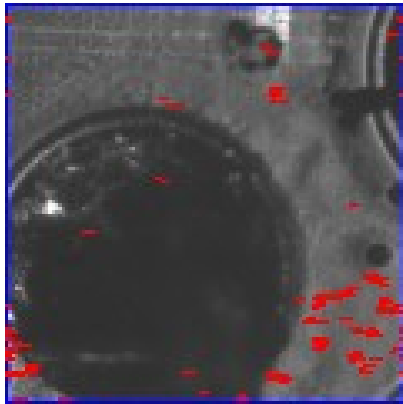


Abbildung 3:  
*micfs33.dat.txt\_overlay.png*  
ohne Streifenkorrektur

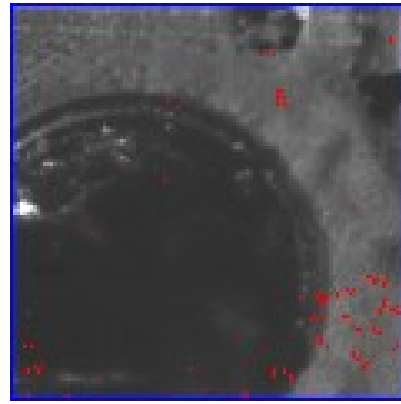


Abbildung 4:  
*micfs33.dat.txt\_overlay.png*  
mit Streifenkorrektur

Auf den korrigierten Bildern treten keine Streifen auf. Das Entfernen der Zeilen führt aber dazu, dass Bereiche, die in ihrer gesamten Ausdehnung empfindlich sein müssten, nicht mehr kontinuierlich als empfindlich dargestellt werden.

Zudem sind am linken Rand des aktiven Fläches einige Artefakte zu beobachten. Im Skript "*LatchUpMap\_v2\_3.py*" kann durch Setzen der Variable "D4Col" auf "True" sichergestellt werden, dass diese beim Übertragen der SEL in das Array nicht berücksichtigt werden.

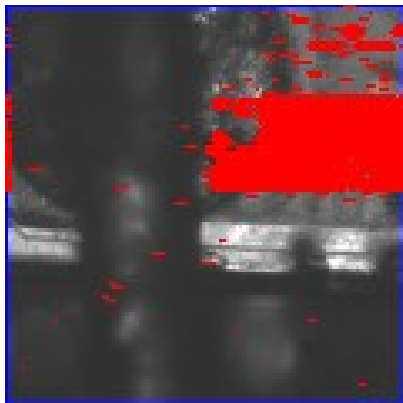


Abbildung 5:  
*micfs35.dat.txt\_overlay.png*  
ohne Streifenkorrektur: Es  
wird die gesamte Fläche als  
empfindlich angezeigt.

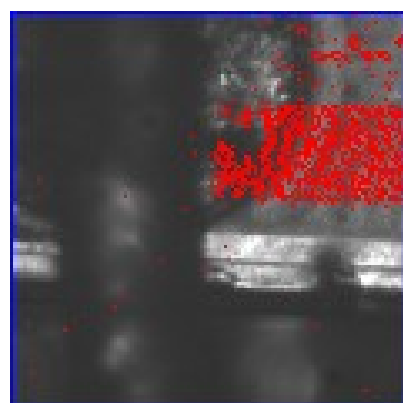


Abbildung 6:  
*micfs35.dat.txt\_overlay.png*  
mit Streifenkorrektur: Es wird  
nicht die gesamte Fläche als  
empfindlich angezeigt.

## 4. Ansätze zum Zusammenfügen zu einem Panorama

Die neuen Bilder sollen miteinander kombiniert werden, um den Vergleich mit den Ergebnissen Laser-Fullscan-Experimentes zu erleichtern. Außerdem soll geprüft werden, ob der gesamte Mikrocontroller wirklich gescannt wurde oder ob einige Bereiche ausgelassen wurden.

### 4.1. Hugin

Hugin ist ein FOSS (Free Open Source Software) Stitchingprogramm zur Erstellung von Panoramen.

Die neu generierten Bilder werden in drei Schritten zusammengefügt:

1. *Einlesen der Bilder:* Die Bilder können per Drag-and-drop in Hugin eingelesen werden.
2. *Korrespondierende Punkte finden:* Korrespondierende Punkte werden zuerst automatisch gefunden und können anschließend manuell bearbeitet werden. Punkte können gelöscht oder hinzugefügt werden. Es wird eine Projektdatei angelegt.
3. *Stitchen:* Die Bilder werden automatisiert zusammengefügt. Das resultierende Panorama wird im angegebenen Pfad gespeichert.

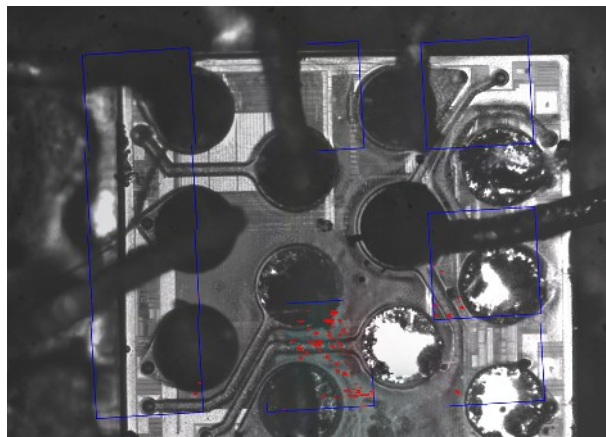


Abbildung 7: Mit Hugin generiertes Panorama

Bei der Verwendung von Hugin treten folgende Probleme auf:

1. Da die Scanbereiche jeweils nur einen Teil der Mikroskopaufnahme einnehmen, gibt es erhebliche Überschneidungen. Da die SEL nur auf einem der Bilder eingezeichnet sind, kommt es zum "Verschlucken" der eingezeichneten SEL und das resultierende Panorama ist nicht zu gebrauchen.
2. Aus dem generierten Projektfile können keine relevanten Informationen zur Transformation der Bilder rekonstruiert werden. Dadurch ist es nicht möglich, die SEL nach dem Erstellen des Panoramas einzuzeichnen.

Dementsprechend ist Hugin für diese Anwendung nicht geeignet und muss als Methode verworfen werden.[2]

## 4.2. Ansatz über Homographiematrizen mit Python

Python ermöglicht es mithilfe von Homographiematrizen Koordinatentransformationen zwischen zwei sich überlappenden Bildern auszuführen. Um zwischen den einzelnen Bildern und dem erstellten Panorama Homographiematrizen zu berechnen, müssen folgende Schritte durchgeführt werden:

1. Zuerst müssen korrespondierende Punkte in beiden Bildern gefunden und die "guten" Korrespondenzen gespeichert werden.
2. Im nächsten Schritt werden Komponenten der Programmbibliothek OpenCV verwendet, um die Homographiematrix zu erstellen.[6]
3. Anschließend werden die SEL-Koordinaten für das Panorama berechnet.

Es gilt:  $x' = Hx$ , wobei  $H$  die Homographiematrix,  $x$  die Koordinaten im Originalbild (Einzelbild) und  $x'$  die transformierten Koordinaten im Panorama sind.

4. Schließlich werden die SELs im Panorama eingezeichnet.

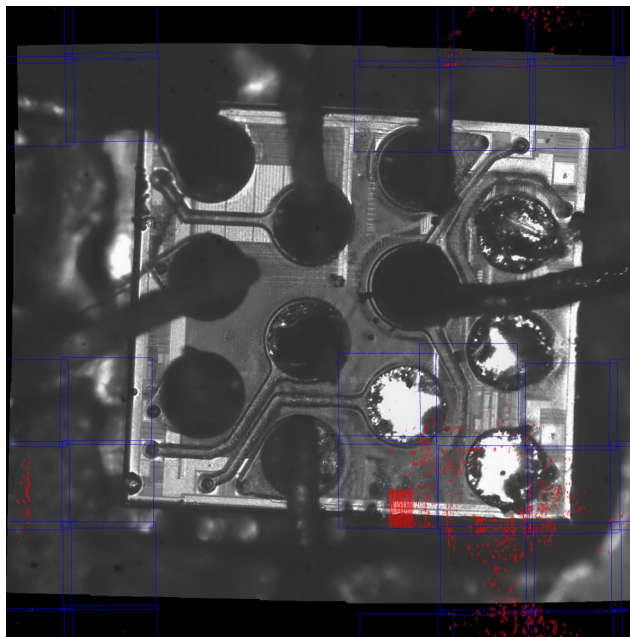


Abbildung 8: Mit Python über Homographiematrizen erstelltes Panorama

Trotz der vielversprechenden Idee gibt es einige Probleme. Zum Beispiel sind die Scanbereiche an der y-Achse gespiegelt und um 90° gegen den Uhrzeigersinn gedreht. Obwohl das Potenzial der Idee besteht, ist der Aufwand im Vergleich zum Nutzen zu hoch.

### 4.3. Manuelles Zusammenfügen

Bei der manuellen Verarbeitung der erzeugten Bilder müssen diese zunächst auf den Scanbereich zugeschnitten werden. Anschließend müssen die Scanbereiche per Hand so ausgerichtet werden, dass sie nahtlos zusammengefügt werden können. Dies erfordert möglicherweise einige Anpassungen und eine sorgfältige Handhabung, um das beste Ergebnis zu erzielen. Diese Methode wurde für das Projekt verwendet.



*Abbildung 9: manuell zusammengefügtes Panorama*

## 5. Anpassen der Daten des Laserexperiments

Um die Wirkungsquerschnitte beider Experimente vergleichen zu können, müssen die Panoramen die selben Dimensionen haben. Dafür müssen die Daten (Bilder und CSV-Dokumente) des Laser-Fullscan-Experiments verkleinert werden. Dabei ist darauf zu achten, dass die totale Pixelfläche, auf der SEL detektiert wurden, relativ zur Gesamtfläche konstant bleibt. Es dürfen also keine Pixel verloren gehen, wodurch Standardlösungen aus Pythonbibliotheken wie *scipy* oder *numpy* ungeeignet sind.[3][5]

In Fällen, in denen mehrere Pixel mit detektierten SEL auf ein identisches Pixel projiziert werden, muss der Minimalwert größer 0 der originalen Pixel eingesetzt werden. Dies dient dazu, potenzielle Verfälschungen der Latch-Up-Schwelle zu vermeiden.

Dafür müssen folgende Schritte ausgeführt werden:

1. Panoramas zusammensetzen
2. Größe ändern (Pixel Binning)
3. Überlappende Pixel löschen
4. Streifen zusammenfügen

### 5.1. Panoramas zusammensetzen

Die Panoramas werden aus den Bildern genau wie die Bilder nach dem Ionenbeschuss manuell zusammen gesetzt und so eng wie möglich auf die Größe des Chips zugeschnitten.

### 5.2. Größe ändern

#### 5.2.1. Lösungsalgorithmus

Nachdem die Skalierungsfaktoren berechnet und die CSV-Datei eingelesen wurde, werden sie an die Pythonfunktion *custom\_binning\_smallest\_value()* zur weiteren Verarbeitung gegeben. Dabei werden 3 verschiedene Schritte ausgeführt:

1. Berechnen der neuen Dimensionen und Erstellen eines entsprechenden Arrays.

Für jedes Pixel im neuen Array:

2. Berechnen, welche Pixel des originalen Arrays mit dem aktuellen Pixel korrespondieren
3. Extraktion der Minimalwerte größer 0 aus dem Bereich und Speichern im neuen Array

#### Berechnen der neuen Dimensionen und Erstellen eines entsprechenden Arrays

Um die neuen Dimensionen zu berechnen, werden die Dimensionen des Originalarrays mit dem jeweiligen Skalierungsfaktor multipliziert und in einer Liste *new\_shape* gespeichert. Damit kann

dann mithilfe der Pythonfunktion `numpy.zeros(new_shape)` ein neuer Array mit den neuen Dimensionen erstellt und mit Nullen gefüllt werden.

### Berechnen, welche Pixel des originalen Arrays mit dem aktuellen Pixel korrespondieren

Für jedes Pixel in dem neuen Array wird berechnet, mit welchen Pixeln des originalen Arrays es korrespondiert. Dazu wird der Index entlang jeder Achse mit dem entsprechenden Skalierungsfaktor dividiert. Dadurch wird die Skalierung rückgängig gemacht und die Position im originalen Array wird zurückgegeben.

$$P_{original,Start} = trunc\left(\frac{P_{neu}}{Skalierungsfaktor}\right)$$

Um den Bereich einzugrenzen, fehlt noch ein Endwert. Dieser wird mit der Formel

$$P_{original,Ende} = trunc\left(\frac{P_{neu} + 1}{Skalierungsfaktor}\right)$$

berechnet.

Mit den Start- und Endpixeln kann nun ein Ausschnitt des Originalarrays extrahiert werden, der in dem neuen Array zu einem Pixel zusammengesetzt wird.

### Extraktion der Minimalwerte größer Null aus dem Bereich und Speichern im neuen Array

In dem Ausschnitt kann nun der Minimalwert größer Null ( $\min\{P \in \text{Ausschnitt} | P > 0\}$ ) gefunden werden und in das neue Array eingesetzt werden. Wenn das auf keinen der Werte zutrifft, bleibt das Pixel Null.

## 5.2.2. Auswertung und Bewertung des Algorithmus

Um bewerten zu können, ob die Funktion die Vorgaben, dass keine Pixel verloren gehen und die minimale SEL-Schwelle gewählt wird, erfüllt, wird die CSV-Datei 20230606-124422\_t.csv als Array eingelesen und dann um die Faktoren

x: 0.8266033254156769

y: 0.7913486005089059

(original Skalierungsfaktoren bei Verwendung des manuell erstellten Ionenpanoramas vor Korrektur der Zeilenbildung) verkleinert.

Das Ergebnis wird mit zwei Methoden überprüft:

1. Vergleich der relativen aktiven Fläche
2. Exportieren als Bild

## Vergleich der relativen aktiven Fläche

Um die relativen aktiven Flächen zu vergleichen, wurden für beide Arrays alle SEL (Werte >0) gezählt und dann durch die Gesamtzahl aller Werte dividiert. Das liefert folgende Ergebnisse

**original:**      **0,014004811633939492**

**verkleinert:**    **0,014512083097795364**

Im verkleinerten Bild zeigt sich, dass ein größerer Anteil der Fläche, nämlich ungefähr 0,05% der Gesamtfläche mehr aktiv ist. Das entspricht einem Zuwachs von ca. 3,6% bezogen auf die relative aktive Fläche im originalen Array.

## Exportieren als Bild

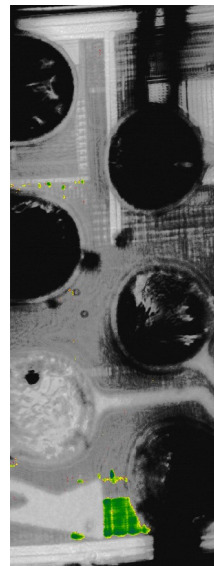
Es werden sowohl das originale als auch das neue Array mithilfe der OpenCV-Funktion `cv2.imwrite()` als Bilder exportiert. Dadurch können die Arrays besser visuell verglichen werden.[6]



*Abbildung 10:  
Originalversion*



*Abbildung 11:  
verkleinerte  
Version*



*Abbildung 12:  
Laserbilddaten*

Auf den ersten Blick sehen beide Bilder sehr ähnlich aus und weisen dieselben Strukturen auf. Das deutet darauf hin, dass die Funktion grundsätzlich funktioniert. Jetzt muss noch geprüft werden, ob einzelne Pixel auch bestehen bleiben oder "verschluckt" werden. Dazu wird ein Cluster von Pixeln im oberen Viertel des Originalbildes ausgewählt und mit dem korrespondierenden Bereich im verkleinerten Bild verglichen.



*Abbildung 13:  
verkleinerte  
Version*



*Abbildung 14:  
Originalversion*

Auch diese Ausschnitte stimmen überein. Zudem werden einzelne Pixel nicht "verschluckt" und bleiben im verkleinerten Datensatz als Datenpunkt bestehen.

### **5.3. Überlappende Pixel löschen**

Um herauszufinden, wie viele Pixelspalten sich überlappen, werden die skalierten Bilder zu einem Panorama zusammengefügt und dann so zugeschnitten, dass nur die sich überlappenden Bereiche übrig bleiben. Die sich überlappenden Pixel können gezählt werden. Anschließend können die korrespondierenden Spalten am rechten und linken Rand der CSV-Datei gelöscht werden. Eine eventuelle Verschiebung entlang der y-Achse hat keinen Einfluss auf die Berechnung des Wirkungsquerschnitts.

### **5.4. CSV-Dateien zusammenfügen**

Zum Zusammenfügen der CSV-Dateien werden diese im ersten Schritt eingelesen und die Dimensionen der einzelnen Arrays ermittelt. Die Dokumente werden reihenweise zusammengefügt. Wenn es zu Unterschieden in der Länge der Arrays kommt, werden diese mit Nullen (0) aufgefüllt. Der so entstandene Array wird in eine CSV-Datei geschrieben und im Unterordner "resized" gespeichert.

## 6. Berechnung der Wirkungsquerschnitte

### 6.1. Ionendaten

Zur Berechnung des Wirkungsquerschnitts der Ionendaten wird ein vorhandenes Skript von Dr.-Ing. Hannes Zöllner verwendet. Um das Skript anwenden zu können, muss das Panorama in eine CSV-Datei umgewandelt werden.

Dazu wird das Bild eingelesen und pixelweise ausgewertet. Wenn das Pixel rot ist, wird an seiner Position eine angenommene Impulsenergie der Goldionen als 12-Bit-ADC-Wert in eine Liste eingetragen, ansonsten eine 0. Diese Liste wird anschließend als CSV-Datei gespeichert.

Eine alternative Methode ist das Teilen der Anzahl der hervorgerufenen SEL durch die Anzahl der verschossenen Ionen.

### 6.2. Laserdaten

Die Daten aus dem Laser-Fullscan-Experiment wurden im vorherigen Schritt zur Verwendung mit dem Skript "*LatchUpMap\_v2\_3.py*" vorbereitet und können ohne weitere Zwischenschritte mit dem Skript verwendet werden.

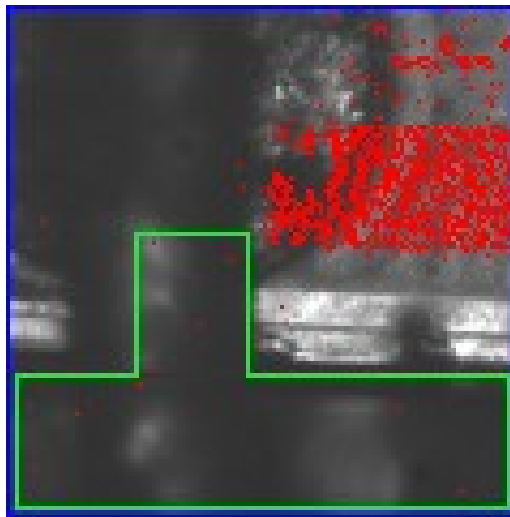
## 7. Auswertung

### 7.1. Plausibilitätsprüfung

#### 7.1.1. Vergleich der Strukturen auf den Einzelbildern

Die von den SEL gebildeten Muster stimmen größtenteils mit den Strukturen der Mikroskopbilder überein. Es gibt allerdings einige Artefakte, die nicht mit den Strukturen auf den Bildern übereinstimmen.

Eine Hypothese für diese Artefakte ist die Herbeiführung von SEL durch sogenannte Sekundäreffekte. Hier prallen Ionen an Hindernissen (z. B. Bonddrähten) ab, treffen auf empfindliche Stellen und lösen dort einen SEL aus. Dieser wird dann mit der Position des ursprünglichen Aufprallens assoziiert. Diese Hypothese lässt sich nur schwer überprüfen.



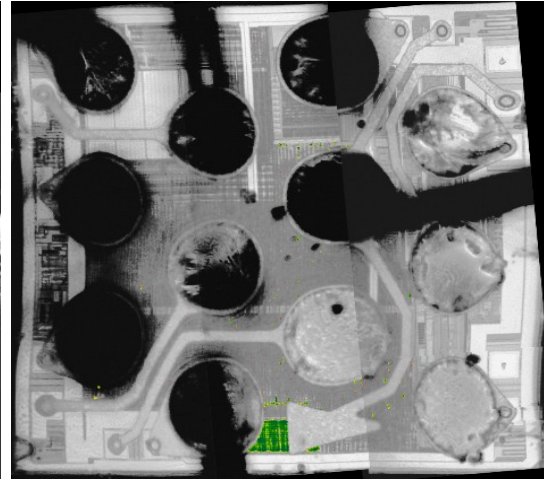
*Abbildung 15: Beispiel für Artefakte:  
Alle SEL im grün umrandeten Bereich  
sind Artefakte, da sie entweder auf  
Bonddrähten oder außerhalb des  
Mikrochips liegen*

### 7.1.2. Vergleich der Panoramen nach Ionen- und Laserbeschuss

Die Strukturen zwischen den Ionen- und Laserdaten stimmen zu großen Teilen überein. Die Unterschiede zwischen den beiden lassen sich dadurch erklären, dass der Laser beim Laser-Fullscan-Experiment metallisierte Oberflächen nicht durchdringen kann und dort auch keine SEL auslösen kann.



*Abbildung 16: Ionen-Panorama*



*Abbildung 17: Laser Panorama*

Bei dem Panorama nach Ionenbeschuss fällt auf, dass es Bereiche am rechten Rand des Mikrocontrollers gibt, die nicht mit Ionen beschossen wurden. So konnten dort demzufolge keine SEL hervorgerufen werden. Laut der Daten des Laserexperiments ist dieser Bereich allerdings nicht empfindlich.

## 7.2. Auswertung der Wirkungsquerschnitte

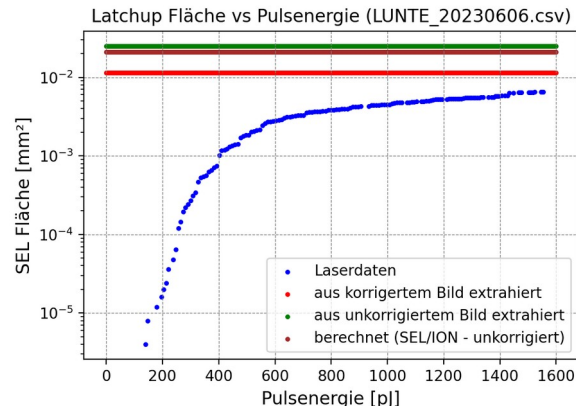


Abbildung 18: Berechnete SEL-Fläche der Laser- und Ionendaten.

Latchup Flächen im Ionenexperiment in mm <sup>2</sup>	
Bild (unkorrigiert)	0,025
Bild (korrigiert)	0,0115
SEL/Ion (unkorrigiert)	0,021

Die SEL-Fläche der korrigierten Daten des Ionenexperiments zirka doppelt so groß wie die SEL-Fläche der Laserdaten bei maximaler Pulsenergie. Da die Kurve der Daten des Laserexperiments ab einer Pulsenergie von ungefähr 700 pJ abflacht und der Laser auf metallisierten Flächen keine SEL hervorrufen kann, kann die Energie der Goldionen (1,2 GeV) zu einer Pulsenergie des Lasers zwischen etwa 700 pJ und mehr als 1600 pJ korrespondieren. Um diesen Wert zu präzisieren, muss die metallisierte Fläche, bei der die Ionen einen SEL auslösen konnten, ermittelt werden. Diese kann dann in die Berechnung des Wirkungsquerschnitts mit einbezogen werden.

## 8. Zusammenfassung

Um die korrespondierende Laserpulsenergie zu der Energie der Ionen ermitteln zu können, müssen zuerst folgende Schritte ausgeführt werden:

1. Generieren der Einzelbilder aus den Ionendaten
2. Zusammenfügen der Einzelbilder zu einem Panorama
3. Anpassen der Größe der Laserdatenn
4. Zusammenfügen der Laserdatenn zu einem Panorama
5. Berechnen der Wirkungsquerschnitte für Laser- und Ionendaten?

Die so aufbereiteten Daten müssen zuerst auf Plausibilität geprüft werden. Dazu wird die Topologie des Mikrocontrollers mit den Positionen der SEL verglichen und kontrolliert, ob die sich ergebenden Muster nachvollziehbar sind. Danach werden beide Panoramen gegenübergestellt. Dabei soll herausgefunden werden, ob bei beiden dieselben Bereiche empfindlich sind. Beide Plausibilitätsprüfungen waren erfolgreich. Obwohl die SEL-Koordinaten unterschiedlich sind, sind bei beiden dieselben Bereiche empfindlich, wodurch die Unterschiede plausibel werden.

Nachdem etabliert wurde, dass beide Datensätze korrekt aufgearbeitet wurden, kann für beide der Wirkungsquerschnitt bestimmt werden.

Durch den flachen Anstieg der Wirkungsquerschnittskurve und die Abschirmung metallisierter Flächen vor dem Laser kann kein eindeutiges Ergebnis ermittelt werden. Die korrespondierende Pulsenergie des Lasers kann zwischen 700 pJ und mehr als 1600 pJ liegen.

Um das Ergebnis zu präzisieren, kann die Anzahl der durch Ionen hervorgerufenen SEL hinter metallisierten Flächen bestimmt und in die Berechnung des Wirkungsquerschnitts einbezogen werden. Die erforderliche Methodologie muss in einem zukünftigen Projekt ermittelt werden.

## Literaturverzeichnis

1: Hannes Zöllner, Erzeugung und Untersuchung von Effekten kosmischer Strahlung mit einem Klasse-1-Laseraufbau, 2021, Verfügbar: <https://dx.doi.org/10.14279/depositonce-11862>

2: Hugin project, Hugin Documentation, 12.09.2023, Verfügbar: <https://hugin.sourceforge.io/>

3: The SciPy community, SciPy documentation, 17.08.2023, Verfügbar: <https://docs.scipy.org/doc/scipy/>

5: NumPy Developers, NumPy documentation, 12.09.2023, Verfügbar:

6: OpenCV: Open Source Computer Vision, OpenCV Documentation, 12.09.2023, Verfügbar: <https://docs.opencv.org/4.x/index.html>