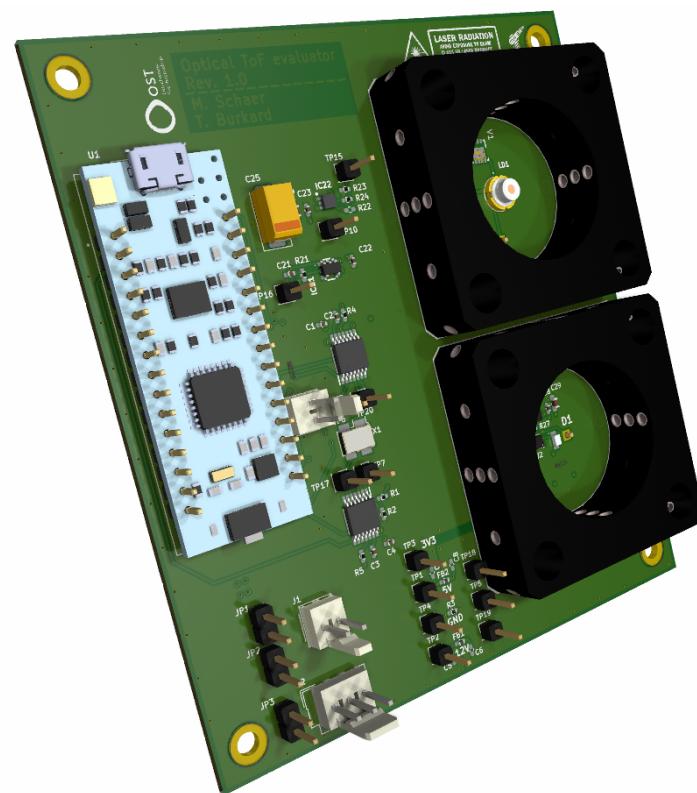


# DIY Optische ToF Distanzmessung

CAS Sensorik und Sensor Signal Conditioning

**Matthias Schär, Timon Burkard**  
OST – Ostschweizer Fachhochschule

23. Januar 2025



# **CAS Sensorik und Sensor Signal Conditioning an der OST – Ostschweizer Fachhochschule**

<b>Titel</b>	<b>DIY Optische ToF Distanzmessung</b>
<b>Diplomandin/Diplomand</b>	<b>Matthias Schär, Timon Burkard</b>
<b>Studiengang</b>	<b>CAS Sensorik und Sensor Signal Conditioning</b>
<b>Semester</b>	<b>HS24</b>
<b>Dozentin/Dozent</b>	<b>Prof. Guido Keel, Michael Lehmann</b>

## **Abstract**

Diese Projektarbeit behandelt die Entwicklung eines Do It Yourself (DIY) Systems zur optischen Time of Flight (ToF) Distanzmessung.

Um die Signallaufzeit präzise zu messen, wird ein Time to Digital Converter (TDC) eingesetzt. Zu Testzwecken wird ein zweiter TDC verwendet, der rein elektrisch betrieben wird, um die Länge unterschiedlicher Kupferkabel zu bestimmen.

Die Arbeit umfasst die Entwicklung der Hardware, einschließlich Lasertreiber, Photodetektor und Transimpedanzverstärker, sowie die Implementierung der Firmware zur Signalauswertung. Simulationen und Messungen wurden durchgeführt, um die Genauigkeit zu optimieren und Störfaktoren zu minimieren.

Die elektrischen Messungen zeigen, dass der TDC eine zeitliche Auflösung im Bereich einiger Dutzend Pikosekunden erreicht. Damit lassen sich Kabellängen auf wenige cm genau bestimmen.

Die optischen Messungen zeigen, dass eine grobe Distanzmessung bis ca. 30 cm möglich ist, wenn ein Spiegel als Target eingesetzt wird. Um die Distanz zu einer Wand messen zu können, sind weiterführende Verbesserungen des Demonstrators notwendig.

Rapperswil, 23. Januar 2025

© Matthias Schär, Timon Burkard, OST – Ostschweizer Fachhochschule

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>11</b>
1.1 Thema . . . . .	11
1.2 Zielsetzung . . . . .	11
1.3 Gliederung . . . . .	11
<b>2 Theorie</b>	<b>13</b>
2.1 Time of Flight . . . . .	13
2.2 Photostrom . . . . .	14
2.2.1 Berechnung mit RLD94PZJ5 und BPV23NF . . . . .	14
2.2.2 Berechnung mit RLD65NZX1 und NJL6401R-3 . . . . .	15
2.2.3 Berechnung mit Empfangs-Linse . . . . .	15
2.2.4 Erwarteter Lichtstrom . . . . .	16
2.3 Transimpedanzverstärker . . . . .	18
<b>3 Konzept</b>	<b>19</b>
3.1 Signalkette . . . . .	19
3.2 Zeitmessung . . . . .	20
3.3 Vorgehensweise . . . . .	21
<b>4 Simulationen</b>	<b>22</b>
4.1 Laser Treiber . . . . .	22
4.2 Transimpedanzverstärker . . . . .	23
<b>5 Umsetzung</b>	<b>26</b>
5.1 Schaltungen . . . . .	26
5.1.1 Selective Input Voltage . . . . .	26
5.1.2 Nucleo Board . . . . .	27
5.1.3 TDC Electrical Signal . . . . .	28
5.1.4 TDC Optical Signal . . . . .	28
5.1.5 Oscillator For TDCs . . . . .	29
5.1.6 Power Supply Separation . . . . .	30
5.1.7 Laser Driver . . . . .	31
5.1.8 Photo Receiver . . . . .	31
5.1.9 Decoupling Capacitors . . . . .	32
5.2 Layout . . . . .	33
5.3 3D View . . . . .	34
5.4 Fabrikation . . . . .	34
5.5 Firmware . . . . .	35
<b>6 Resultate</b>	<b>38</b>
6.1 Elektrische Messungen . . . . .	38
6.1.1 GPIO Toggle mit HAL . . . . .	38
6.1.2 GPIO Toggle ohne HAL . . . . .	41
6.1.3 Unterschiedliche Kabellängen . . . . .	44
6.1.4 Mode 1 vs. Mode 2 . . . . .	46
6.1.5 Timer Output . . . . .	48
6.1.6 Unterschiedliche Clock-Konfigurationen . . . . .	50
6.1.7 Unterschiedliche Kabellängen mit DSO . . . . .	52

---

6.2	Optische Messungen . . . . .	57
6.2.1	Signalpfad Sender . . . . .	57
6.2.2	ToF-Messungen via Spiegel . . . . .	59
6.2.3	ToF-Messungen via Wand . . . . .	65
<b>7</b>	<b>Fazit</b>	<b>66</b>
7.1	Ausblick . . . . .	66
7.2	Lessons Learned . . . . .	67
7.3	Danksagung . . . . .	67
<b>8</b>	<b>Anhang</b>	<b>68</b>
8.1	Outline . . . . .	68
8.2	TDC Treiber . . . . .	69
8.3	Python Skripte . . . . .	77
8.4	Schema . . . . .	82
8.5	Stückliste . . . . .	85
8.6	Layout . . . . .	86
8.7	Komponenten-Platzierung . . . . .	90
8.8	3D View . . . . .	92
8.9	Fotos Demonstrator . . . . .	93

# Abkürzungsverzeichnis

**ADC** Analog to Digital Converter. 18

**AMS** Analog/Mixed Signal. 20

**CPU** Central Processing Unit. 36, 37, 39, 40, 45, 46

**DIY** Do It Yourself. 2

**DSO** Digital Storage Oscilloscope. 36, 38, 39, 41, 42, 50, 51, 53–55, 58–60

**FW** Firmware. 33

**GPIO** General Purpose Input/Output. 26, 34, 36–43, 45–48, 50, 55, 64

**HAL** Hardware Abstraction Layer. 36–42, 45

**HSI** High Speed Internal. 47–49

**IC** Integrated Circuit. 9, 18, 25, 34, 65

**MCU** Microcontroller Unit. 18, 19, 25–27, 34, 36, 39, 45, 46, 55, 58, 63, 64

**PCB** Printed Circuit Board. 23, 31, 32, 57, 64, 65

**PLL** Phase-Locked Loop. 48–50

**SPI** Serial Peripheral Interface. 25, 26, 34

**TDC** Time to Digital Converter. 2, 9, 19, 25–29, 34, 37, 38, 40, 41, 44–47, 50–53, 55, 59–61, 64, 67, 71, 75

**TIA** Trans-Impedance Amplifier. 16, 21–23, 28, 29, 31, 62–65

**ToF** Time of Flight. 2, 9, 11, 17, 18, 21, 24, 32, 33, 64, 65, 76–78

**UART** Universal Asynchronous Receiver/Transmitter. 37, 40

**USB** Universal Serial Bus. 25

# Abbildungsverzeichnis

1	Visualisierung von a: direct ToF und b: indirect ToF . . . . .	13
2	Linse von QIOPTIQ . . . . .	16
3	Beschaltung Operationsverstärker als TIA . . . . .	18
4	Blockdiagramm . . . . .	19
5	Konzeptioneller Signalpfad . . . . .	20
6	Konzept Betrieb TDC rein elektrisch . . . . .	21
7	Simulation Laser-Treiber - Schema . . . . .	22
8	Simulation Laser-Treiber - Plot . . . . .	23
9	Simulation TIA - Schema . . . . .	23
10	Simulation TIA - Plot . . . . .	24
11	Simulation TIA - Plot ohne CF . . . . .	25
12	Selective Input Voltage . . . . .	26
13	Nucleo Board . . . . .	27
14	TDC Electrical Signal . . . . .	28
15	TDC Optical Signal . . . . .	29
16	Oscillator for TDCs . . . . .	30
17	Power Supply Separation . . . . .	30
18	Laser Driver . . . . .	31
19	Photo Receiver . . . . .	32
20	Decoupling Capacitors . . . . .	32
21	Innerer Layer 1 mit Vermerken zum Speisungskonzept . . . . .	33
22	3D View Top . . . . .	34
23	Demonstrator von vorne ohne Linse . . . . .	35
24	Aufbau Firmware-Projekt . . . . .	35
25	Pin-Konfiguration des STM32F042K6 . . . . .	37
26	GPIO Toggle mit HAL - Histogramm . . . . .	39
27	GPIO Toggle mit HAL - Boxplot . . . . .	40
28	GPIO Toggle mit HAL - DSO . . . . .	40
29	GPIO Toggle mit HAL - DSO (Zoom) . . . . .	41
30	GPIO Toggle ohne HAL - Histogramm . . . . .	42
31	GPIO Toggle ohne HAL - Boxplot . . . . .	43
32	GPIO Toggle ohne HAL - DSO . . . . .	43
33	GPIO Toggle ohne HAL - DSO (Zoom) . . . . .	44
34	Unterschiedliche Kabellängen . . . . .	45
35	TDC Mode 1 . . . . .	46
36	TDC Mode 2 . . . . .	47
37	Default Clock-Configuration . . . . .	49
38	Clock-Configuration HSI mit PLL 16 MHz . . . . .	51
39	Clock-Configuration HSI48 48 MHz . . . . .	51
40	PLL-Jitter . . . . .	52
41	Unterschiedliche Kabellängen - Messwerte TDC . . . . .	53
42	DSO-Aufzeichnung bei Kabellänge 1 m . . . . .	55
43	DSO-Aufzeichnung bei Kabellänge 6 m . . . . .	56
44	Messung Signalpfad Sender: <code>start_op</code> , <code>LD_pulse</code> , FET-Gate . . . . .	57
45	Messung Signalpfad Sender: Strom durch Laser-Diode . . . . .	58
46	Mess-Setup mit Spiegel . . . . .	59
47	Laser Abstrahl-Charakteristik . . . . .	59

48	Laser-Ausrichtung . . . . .	60
49	Laser-Ausrichtung angepasst . . . . .	60
50	Initiale Spiegelmessung mit DSO . . . . .	61
51	Unterschiedliche Distanzen zum Spiegel - Plot . . . . .	62
52	Unterschiedliche Distanzen zum Spiegel - Plot mit linearer Interpolation . . . . .	63
53	TIA-Ausgang und Komparator-Schaltschwelle bei 12 cm . . . . .	64
54	TIA-Ausgang und Komparator-Schaltschwelle bei 30 cm . . . . .	65
55	Outline . . . . .	68
56	Schema S.1/3 . . . . .	82
57	Schema S.2/3 . . . . .	83
58	Schema S.3/3 . . . . .	84
59	PCB Layout Top . . . . .	86
60	PCB Layout Innen 1 . . . . .	87
61	PCB Layout Innen 2 . . . . .	88
62	PCB Layout Bottom . . . . .	89
63	PCB Komponenten-Platzierung Top . . . . .	90
64	PCB Komponenten-Platzierung Bottom . . . . .	91
65	3D View Top . . . . .	92
66	3D View Bottom . . . . .	92
67	Demonstrator von oben . . . . .	93
68	Demonstrator von oben ohne Linse . . . . .	93
69	Demonstrator von vorne . . . . .	94
70	Demonstrator von vorne ohne Linse . . . . .	94
71	Demonstrator von unten . . . . .	95

# Formelverzeichnis

1	Eintreffende Lichtleistung . . . . .	14
2	Strahlungsintensität . . . . .	14
3	Raumwinkel . . . . .	14
4	Photostrom . . . . .	14
5	Werte des RLD94PZJ5 . . . . .	14
6	Werte des BPV23NF . . . . .	14
7	Nummerische Resultate mit RLD94PZJ5 und BPV23NF . . . . .	15
8	Werte des RLD65NZX1 . . . . .	15
9	Werte des NJL6401R-3 . . . . .	15
10	Nummerische Resultate mit RLD65NZX1 and NJL6401R-3 . . . . .	15
11	Vergrösserung der Empfangsfläche durch Linse . . . . .	16
12	Erhöhte Sendeleistung bei der Laser-Diode . . . . .	16
13	Photostrom Szenario 1 . . . . .	17
14	Modell Abstrahlung Laserdiode . . . . .	17
15	Photostrom Szenario 2 . . . . .	17
16	Übertragungsfunktion TIA . . . . .	18
17	Maximale räumliche Auflösung des TDC7200 . . . . .	20
18	TIA Zeitkonstante . . . . .	24
19	Vorwiderstand Laser-Diode . . . . .	31
20	GPIO Toggle mit HAL . . . . .	39
21	GPIO Toggle ohne HAL . . . . .	42
22	Zurückrechnen auf Kabellänge . . . . .	45
23	Mode 1 vs. Mode 2 . . . . .	48
24	GPIO-Schalten Verzögerungszeit . . . . .	49
25	GPIO Toggle mit Timer-Output . . . . .	50
26	GPIO-Schalten Verzögerungszeiten . . . . .	52
27	Berechnung Strom durch Laser-Diode . . . . .	58
28	Berechnung Strom durch Laser-Diode (tatsächlich) . . . . .	58
29	Theoretisch erwartete Laufzeit-Änderung . . . . .	62

# Tabellenverzeichnis

1	Unterschiedliche Kabellängen . . . . .	45
2	Kabellängen zurückgerechnet . . . . .	46
3	Mode 1 vs. Mode 2 . . . . .	48
4	Unterschiedliche Clock-Quellen . . . . .	51
5	Unterschiedliche Kabellängen - Resultate TDC . . . . .	53
6	Unterschiedliche Kabellängen - Resultate DSO . . . . .	53
7	Unterschiedliche Kabellängen - Theoretische Erwartung . . . . .	54
8	Unterschiedliche Kabellängen - Unterschied TDC [ns] zu Theorie . . . . .	54
9	Unterschiedliche Kabellängen - TDC [m] gem. Theorie . . . . .	54
10	Unterschiedliche Kabellängen - TDC [m] gem. Theorie 0.9 c . . . . .	55
11	Unterschiedliche Distanzen zum Spiegel - Statistische Grössen . . . . .	62
12	Bill of Material . . . . .	85

# Codeverzeichnis

1	TDC Init und Enable mit Treiber . . . . .	36
2	TDC Read/Write mit Treiber . . . . .	36
3	TDC Messung mit Treiber . . . . .	36
4	GPIO Toggle mit HAL . . . . .	38
5	GPIO Toggle mit HAL . . . . .	39
6	GPIO Toggle ohne HAL . . . . .	41
7	GPIO Toggle ohne HAL . . . . .	42
8	Mode 1 . . . . .	47
9	GPIO Toggle mit Timer-Output . . . . .	49
10	TDC Driver (Header) . . . . .	69
11	TDC Driver (Source) . . . . .	73
12	Python Analyse . . . . .	77
13	Python Analyse (Multi ToFs) . . . . .	78
14	Python Analyse (Multi Logs) . . . . .	79
15	Python Live Plot . . . . .	80

# 1 Einleitung

In diesem Kapitel wird eine Einleitung in die Thematik der vorliegenden Projektarbeit gegeben.

Verfasst wird diese Arbeit von Timon Burkard und Matthias Schär. Betreuend stehen die Dozenten Prof. Guido Keel und Michael Lehman zur Seite.

## 1.1 Thema

Die Arbeit befasst sich mit der Entwicklung einer Demonstrator-Elektronik für die Veranschaulichung von optischer Time of Flight (ToF) zur Distanzmessung.

Es wird ein Mess-IC, der TDC7200 von der Firma Texas Instruments, in Betrieb genommen. Dies geschieht in zwei Schritten: Zuerst sollen rein elektrische Messungen ermöglicht werden, um beispielsweise die Länge von angeschlossenen Kabeln zu bestimmen. Weiter soll sich auf dem Demonstrator die Optoelektronik befinden, welche für eine optische Distanzmessung benötigt wird.

## 1.2 Zielsetzung

Die Studierenden haben sich zum Ziel gesetzt, eine Elektronik zu entwickeln, mit der sich die Versuche durchführen lassen. Dabei soll die Ansteuerung der Mess-ICs sowie das Auslegen der Optoelektronik ebenfalls weitgehend selbstständig geschehen.

Es soll ein Demonstrator entstehen, mit welchem es möglich ist, unter Einsatz des TDC7200 unterschiedliche Kabellängen von Kupferkabeln zu bestimmen. Nach dem dieses Ziel erreicht wird, soll es weiter möglich sein, optische Messungen mit dem selben Demonstrator durchzuführen. Die Studierenden haben sich hierbei zum Ziel gesetzt, dass die Messungen mit einer Wand als Target bis zu einer Distanz von 10 m funktionieren. Weiter sollte eine räumliche Auflösung von etwa 30 cm erreicht werden.

## 1.3 Gliederung

Nachfolgend sei ein kurzer Überblick über die Kapitel im restlichen Teil dieser Arbeit gegeben:

- **Kapitel 2 - Theorie:** Befasst sich mit den theoretischen Grundlagen, welche für die Entwicklung des Demonstrators erarbeitet wurden.
- **Kapitel 3 - Konzept:** Erklärt die groben Konzepte und Herausforderungen, welche es zu bewältigen gibt. Hierbei werden insbesondere die zu entwickelnde Elektronik und die zeitlichen Anforderungen an den TDC erfasst.
- **Kapitel 4 - Simulationen:** Beinhaltet Informationen zu den Simulationen, welche vor dem Design aber auch während der Inbetriebnahme des Demonstrators durchgeführt werden.
- **Kapitel 5 - Umsetzung:** Erläutert die Umsetzung in die Realität. Hier werden Schaltungen sowie der Herstellungsprozess präsentiert. Zudem wird erklärt, was an Software-Entwicklung notwendig ist, um einen solchen Demonstrator zu realisieren.
- **Kapitel 6 - Resultate:** Dokumentiert sämtliche durchgeführte Messungen sowie deren Resultate. Hier ist ersichtlich, inwiefern die initiale Zielsetzung erreicht wurde.

- **Kapitel 7 - Fazit:** Erfasst abschliessende Gedanken der Studierenden und bildet ein Fazit der Arbeit. Es wird eine Aussicht auf mögliche Erweiterungen und Verbesserungen gegeben.

## 2 Theorie

In den nachfolgenden Unterkapitel werden die theoretischen Grundlagen erarbeitet, welche für das Verständnis sowie das Design eines Demonstrators für das ToF-Verfahren notwendig sind.

### 2.1 Time of Flight

Beim Messverfahren, welches als "Time of Flight" bekannt ist, geht es darum, die Laufzeit eines Signales zu messen. Das Signal kann hierbei optischer oder akustischer Natur sein. Eine Anwendung davon ist beispielsweise die Distanzmessung.

Beim hier behandelten optischen Prinzip wird mit einem Sender Licht ausgesandt, welches von einem Empfänger detektiert wird. Zwischen dem Senden und Empfangen des Signals kann eine Zeitmessung gemacht werden. Weil sich Licht mit einer definierten Geschwindigkeit ausbreitet, kann über diese Laufzeitmessung ein Rückschluss darüber gewonnen werden, welche Distanz das Signal innerhalb der gemessenen Zeit zurückgelegt hat.

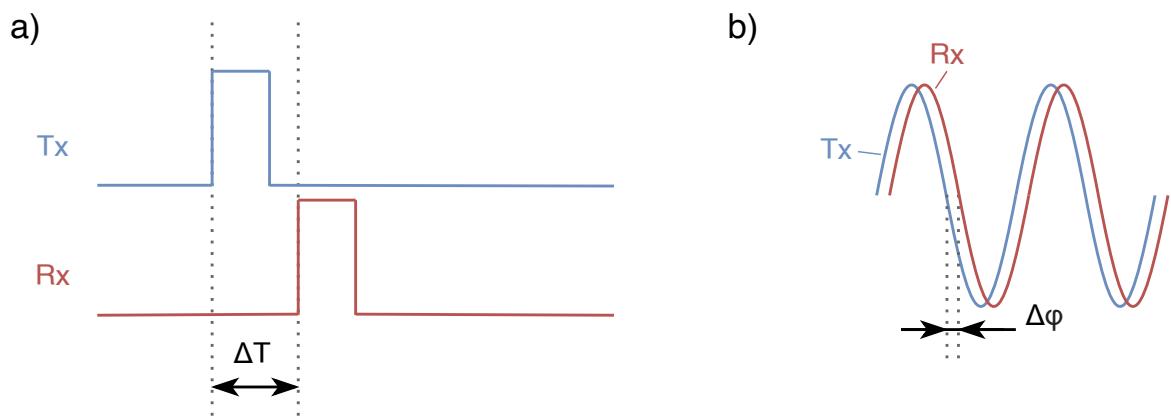


Abbildung 1: Visualisierung von a: direct ToF und b: indirect ToF

Man unterscheidet zwei Arten von ToF: Direct ToF (dTOf) und Indirect ToF (iTOf). Bei dToF wird einzig und alleine die Laufzeit zwischen Sende- und Empfangspuls ermittelt. In der Abbildung 1 sind bei a) beispielhaft solche Pulse dargestellt.

Bei iTOf wird (teilweise zusätzlich) zur Laufzeitmessung auch die Phasenverschiebung eines modulierten Lichtsignals gemessen. Dies sei bei Abbildung 1 im Teil b) dargestellt. Diese Methode erlaubt es, höhere Auflösungen zu erreichen, ist jedoch in der Auswertung und Signalaufbereitung ungleich komplexer.

Diese Arbeit beschränkt sich auf das direct ToF Verfahren.

Nachfolgend werden die theoretischen Grundlagen, vor allem in Hinblick auf das Auslegen einer geeigneten Optik, diskutiert.

## 2.2 Photostrom

Zur Berechnung des theoretisch zu erwartenden Photostroms wird von einer Distanz von 10 m zur Wand ausgegangen.

In einer vereinfachten Betrachtung treffe ein Laserstrahl mit einer Divergenz von 0° rechtwinklig auf eine Wand und werde dort uniform Halbkugel-förmig gestreut. In der Realität wird der Laser nicht mit 0° zur Wand gehen und die Streuung wird sich nicht uniform verteilen, sondern in der Mitte stärker konzentriert sein.

Die Berechnung der empfangenen Strahlungsleistung, der Strahlungsintensität, dem Raumwinkel einer Halbkugel und dem Photostrom sind in Formel 1, 2, 3 bzw. 4 gezeigt.

$$P_{in} = E_e \cdot A = \frac{I_e}{r^2} \cdot A \quad (1)$$

$$I_e = \frac{P_{out}}{\Omega} \quad (2)$$

$$\Omega = 4 \cdot \pi \cdot 0.5 \quad (3)$$

$$I_{ph} = S \cdot P_{in} \quad (4)$$

### 2.2.1 Berechnung mit RLD94PZJ5 und BPV23NF

Erste Berechnungen werden mit der Laserdiode RLD94PZJ5 (ROHM, 2020) und der Photodiode BPV23NF (Vishay Semic., 2024) durchgeführt.

Die relevanten Werte aus den Datenblättern sind in Formel 5 und 6 aufgelistet.

$$P_{out} = 285 \text{ mW} \quad (5)$$

$$A = 4.4 \text{ mm}^2$$

$$S = 0.6 \frac{A}{W} \quad (6)$$

Diese Werte eingesetzt in Formel 2, 1 und 4 ergibt die Resultate in Formel 7.

$$\begin{aligned}
 I_e &= \frac{P_{out}}{\Omega} = \frac{285 \text{ mW}}{4 \cdot \pi \cdot 0.5 \text{ sr}} = 45.4 \frac{\text{mW}}{\text{sr}} \\
 P_{in} &= \frac{I_e}{r^2} \cdot A = \frac{45.4 \frac{\text{mW}}{\text{sr}}}{(10 \text{ m})^2} \cdot 4.4 \text{ mm}^2 = 2 \text{ nW} \\
 I_{ph} &= S \cdot P_{in} = 0.6 \frac{A}{W} \cdot 2 \text{ nW} = 1.2 \text{ nA}
 \end{aligned} \tag{7}$$

## 2.2.2 Berechnung mit RLD65NZX1 and NJL6401R-3

Die Laserdiode RLD94PZJ5 hat im Bezug auf diese Projektarbeit zwei Nachteile: Einerseits besitzt sie eine so hohe Leistung, dass sie für das menschliche Auge schädigend wirken kann. Andererseits strahlt sie in einem Wellenlängenbereich ab, der für das menschliche Auge nicht sichtbar ist.

Aus diesen Gründen wird eine zweite Laserdiode evaluiert: RLD65NZX1 (ROHM, 2019). Gepaart wird sie mit der Photodiode NJL6401R-3 (JRC, 2014). Die folgenden Berechnungen werden basierend auf diesen beiden Komponenten durchgeführt.

Die relevanten Werte aus den Datenblättern sind in Formel 8 und 9 aufgelistet.

$$P_{out} = 10 \text{ mW} \tag{8}$$

$$\begin{aligned}
 A &= 0.7 \text{ mm} \cdot 0.7 \text{ mm} = 0.49 \text{ mm}^2 \\
 S &= 0.42 \frac{A}{W}
 \end{aligned} \tag{9}$$

Eingesetzt in die Formeln 2, 1 und 4 ergeben sich die neuen Resultate in Formel 10.

$$\begin{aligned}
 I_e &= \frac{P_{out}}{\Omega} = \frac{10 \text{ mW}}{4 \cdot \pi \cdot 0.5 \text{ sr}} = 1.59 \frac{\text{mW}}{\text{sr}} \\
 P_{in} &= \frac{I_e}{r^2} \cdot A = \frac{1.59 \frac{\text{mW}}{\text{sr}}}{(10 \text{ m})^2} \cdot 0.49 \text{ mm}^2 = 7.8 \text{ pW} \\
 I_{ph} &= S \cdot P_{in} = 0.42 \frac{A}{W} \cdot 7.8 \text{ pW} = 3.28 \text{ pA}
 \end{aligned} \tag{10}$$

## 2.2.3 Berechnung mit Empfangs-Linse

Auf Vorschlag der Dozenten werden verschiedene Optiken aus einem Baukasten-System von QI-OPTIQ ausprobiert. Besonders vielversprechend erscheint hierbei eine Linse mit einem Durchmesser von 17 mm bei einer Brennweite von 40 mm. Die Linse ist in Abbildung 2 dargestellt.



Abbildung 2: Linse von QIOPTIQ

Eine solche Linse vergrössert die effektive Fläche, auf welcher der Lichtstrom empfangen werden kann. Dies hat eine höhere Empfangsleistung, sprich einen höheren Lichtstrom, zur Folge.

Die Formel 11 zeigt die beim Einsatz einer solchen Optik zu erwartende Flächenvergrösserung.

$$A' = \frac{A_L}{A_{PD}} = \frac{\left(\frac{17 \text{ mm}}{2}\right)^2 \cdot \pi}{0.49 \text{ mm}^2} = 463.2 \quad (11)$$

## 2.2.4 Erwarteter Lichtstrom

Nachfolgend werden zwei Szenarien betrachtet: Einmal wird wie bis anhin davon ausgegangen, dass sich das Licht ab dem Laser in einem gebündelten Strahl ausbreitet, um danach halbkugelförmig und diffus am Target zu reflektieren.

Im zweiten Szenario wird versucht, die Abstrahlcharakteristik der Laser-Diode zu modellieren. Laut Datenblatt hat die Diode in der Abstrahlebene zwei unterschiedliche Öffnungswinkel, was mit dem Raumwinkel einer rechteckigen Pyramide annäherungsweise berechnet werden kann. Die Streuung wird verhindert, da an einem Spiegel reflektiert wird.

Bei beiden Szenarien sei die abgestrahlte Leistung der Laser-Diode höher als in der ersten Rechnung. Im gepulsten Betrieb kann diese problemlos über das maximale Rating erhöht werden. Thermisch stellt dies kein Problem für die Diode dar, da der Duty-Cycle sehr klein ist. Die Formel 12 zeigt die neu zu erwartete Abstrahl-Leistung eines solchen Pulses. Die Effizienz der Laser-Diode wird dem Datenblatt des RLD65NZX1 entnommen (ROHM, 2019).

$$\begin{aligned} \eta &= \frac{P_{25 \text{ mA}}}{I_{ld}} = \frac{7 \text{ mW}}{25 \text{ mA}} = 0.28 \frac{\text{W}}{\text{A}} \\ I'_{ld} &= \frac{5 \text{ V} - V_{LD}}{R_v} = \frac{5 \text{ V} - 2.8 \text{ V}}{5 \Omega} = 440 \text{ mA} \\ P'_{out} &= \eta \cdot I'_{ld} = 0.28 \frac{\text{W}}{\text{A}} \cdot 440 \text{ mA} = 123 \text{ mW} \end{aligned} \quad (12)$$

Bei einem Target, welches sich in 2 m Distanz befindet, kann nach Formel 13 folgender Lichtstrom erwartet werden:

$$P_{in} = \frac{P_{out}}{\Omega \cdot r^2} \cdot A_L = \frac{123 \text{ mW}}{4\pi \cdot 0.5 \text{ sr} \cdot (2 \text{ m})^2} \cdot 227 \text{ mm}^2 = 1.11 \mu\text{W}$$

$$I_{ph} = S \cdot P_{in} = 0.42 \frac{A}{W} \cdot 1.11 \mu\text{W} = 467 \text{ nA} \quad (13)$$

Ohne Optik strahlt die Laserdiode keinen gebündelten Strahl. Diese Abstrahlcharakteristik sei im zweiten Szenario über den Raumwinkel einer Pyramide, wie er in Formel 14 definiert ist, modelliert. Die typischen Abstrahlwinkel  $\theta$  werden ebenfalls dem Datenblatt der Laser-Diode entnommen. Hierbei ist zu beachten, dass im Datenblatt der Vollwinkel angegeben wird. Der Wert in der Formel wird deswegen halbiert.

$$\Omega_P = 4 \cdot \arcsin(\sin(\theta_x) \cdot \sin(\theta_y)) = 4 \cdot \arcsin\left(\sin\left(\frac{9^\circ}{2}\right) \cdot \sin\left(\frac{29^\circ}{2}\right)\right) = 4.2 \text{ sr} \quad (14)$$

Mit der Formel 15 wird erneut die zu erwartende Empfangsleistung sowie der dazugehörige Photostrom berechnet. Der Unterschied zur vorherigen Berechnung liegt nun aber darin, dass der Raumwinkel auch bereits vor der Reflexion berücksichtigt werden muss. Deswegen wird die Distanz zum Target mit zwei multipliziert.

$$P_{in} = \frac{P_{out}}{\Omega \cdot (2 \cdot r)^2} \cdot A_L = \frac{123 \text{ mW}}{4.2 \text{ sr} \cdot (2 \cdot 2 \text{ m})^2} \cdot 227 \text{ mm}^2 = 416 \text{ nW}$$

$$I_{ph} = S \cdot P_{in} = 0.42 \frac{A}{W} \cdot 416 \text{ nW} = 175 \text{ nA} \quad (15)$$

## 2.3 Transimpedanzverstärker

Wie die Berechnungen in den vorgängigen Kapiteln aufzeigen, besitzen die Photoströme, welche zu detektieren sind, sehr kleine Stromstärken. Weiter sind die Lichtimpulse typischerweise oft nur von sehr kurzer Dauer, nämlich im Bereich einiger Nanosekunden. Dies hat zur Folge, dass die Signale dementsprechend kurze Anstiegszeiten haben, welche es zu detektierten gilt.

Die Anforderungen deuten darauf hin, dass ein Verstärker mit sehr hohem Gain bei gleichzeitig sehr hoher Bandbreite eingesetzt werden muss. Die Verstärkerschaltung, welche eingesetzt wird, ist ein Strom-Spannungswandler. Eine solche Beschaltung des Operationsverstärkers, wie sie in Abbildung 3 zu sehen ist, wird auch Transimpedanzverstärker (TIA) genannt.

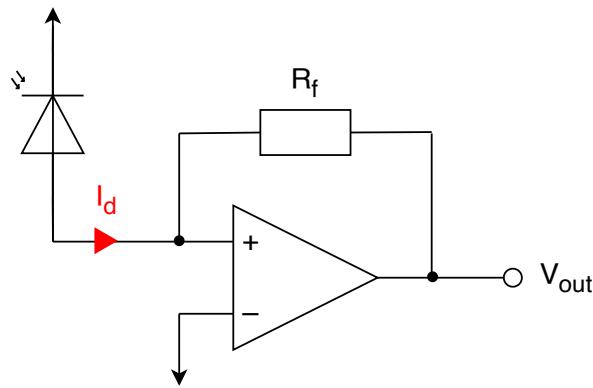


Abbildung 3: Beschaltung Operationsverstärker als TIA

Wie der Name der Schaltung vermuten lässt, wandelt der Transimpedanzwandler den Photostrom  $I_d$  in eine Spannung  $V_{out}$  um. Dabei gibt der Feedback-Widerstand  $R_f$  die Verstärkung vor, wie die Formel 16 zeigt:

$$V_{out} = -R_f \cdot I_d \quad (16)$$

## 3 Konzept

Die folgenden Unterkapitel klären über die Entwicklungsarbeiten auf, welche für das Umsetzen des ToF-Demonstrators nötig sind. Hierbei wird vor allem darauf eingegangen, wie die Elektronik aufgebaut sein kann. Zudem werden weitere Herausforderungen eines solchen Systems und die mögliche Bewältigung ebendieser aufgezeigt.

### 3.1 Signalkette

Nachfolgend sind einige Erklärungen dazu zu finden, wie die ToF-Messung realisiert wird.

Ein Blockdiagramm des Gesamtprojektes, bestehend aus dem elektrischen und dem optischen Teil, ist in Abbildung 4 dargestellt.

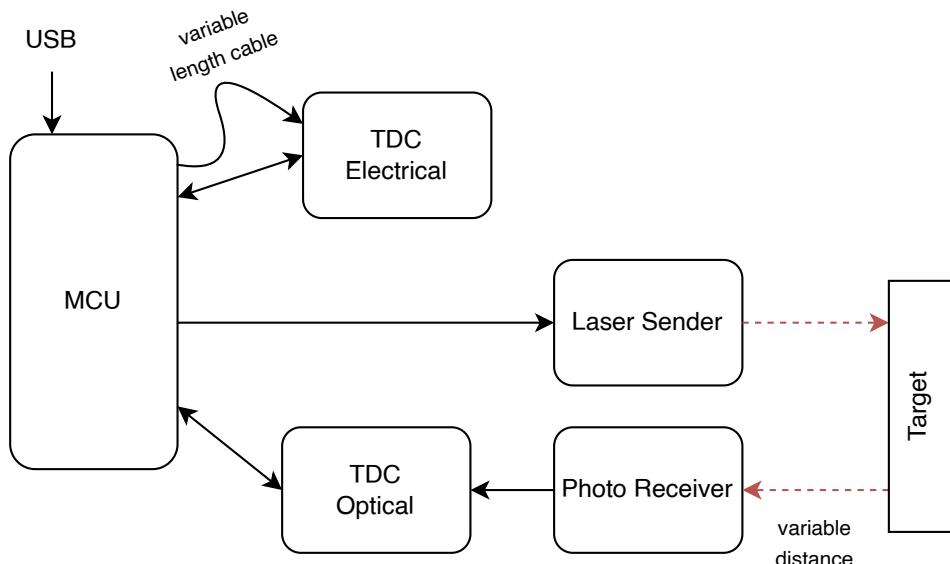


Abbildung 4: Blockdiagramm

Eine detailliertere Skizze der optischen Signalkette ist in der Abbildung 5 ersichtlich.

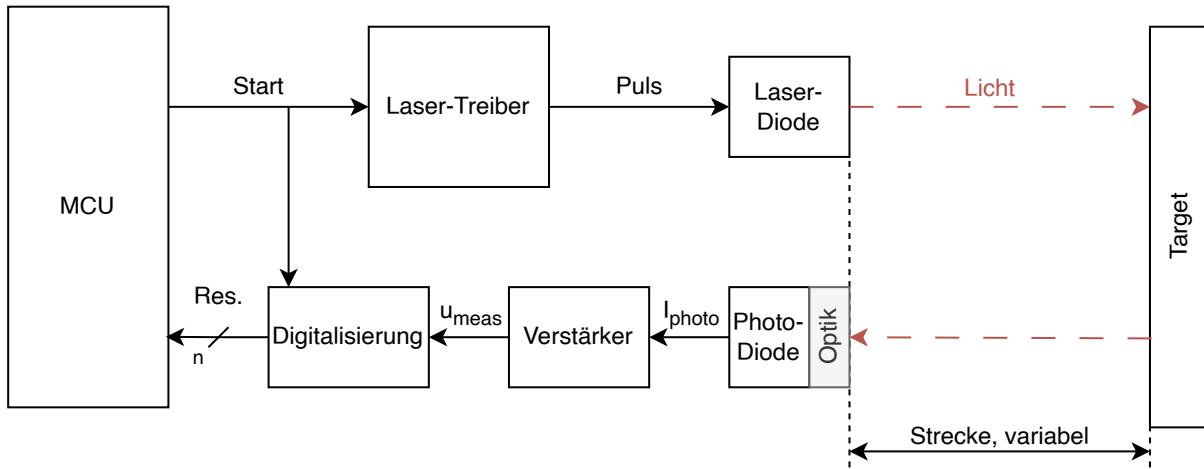


Abbildung 5: Konzeptioneller Signalfpfad

Eine MCU kümmert sich um das Starten und Auswerten der Messresultate. Das Generieren eines Startsignals hat zur Folge, dass über einen Laser-Treiber eine Laser-Diode mit einem Puls angesteuert wird. Dieser Puls, nach der Laser-Diode in Form eines Lichtpulses, wird dann von einem Target reflektiert, welches in einer variablen Distanz zur Elektronik steht.

Der reflektierte Lichtpuls wird über eine Optik fokussiert und anschliessend von einer Photo-Diode in einen Lichtstrom umgewandelt. Dieser Lichtstrom wird von einem Transimpedanzverstärker in eine Spannung umgewandelt. Bevor die MCU diese Spannung auswerten kann, muss diese erst digitalisiert werden. Dies passiert unter Verwendung eines Komparators. Dieser kann als 1 bit ADC verstanden werden, da bloss eine Unterscheidung der beiden Zustände "Licht" und "kein Licht" notwendig ist. Es ergibt sich also ein Empfangspuls.

Wird nun die Zeit zwischen Startsignal und empfangenem Puls gemessen, ist es möglich einen Rückschluss über die variable Strecke zwischen Elektronik und Target zu ziehen. Die Zeitmessung unterliegt höchsten Anforderungen. Als Versinnbildlichung: Innert einer Nanosekunde legt Licht eine Distanz von 30 cm zurück. Wie eine solche Zeitmessung realisiert werden kann, erklärt das nächste Unterkapitel.

## 3.2 Zeitmessung

Damit ein direct ToF-Signal ausgewertet werden kann, benötigt es eine hochauflöste Zeitmessung. Die Firma Texas Instruments bietet dafür mit dem TDC7200 eine gute Lösung an. Laut Datenblatt (TI, 2016b) kann dieser bis zu 55 ps auflösen, mit einer Standardabweichung von 35 ps. Setzt man nun die Lichtgeschwindigkeit sowie diese minimale zeitliche Auflösung in eine Bewegungsgleichung ein, sollte der Baustein gemäss der Formel 17 folgende räumliche Auflösung erreichen:

$$s_{min} = c \cdot t = 299.8 \cdot 10^6 \frac{m}{s} \cdot 55 \text{ ps} = 16.5 \text{ mm} \quad (17)$$

Prinzipiell soll die ToF-Funktionalität in zwei Schritten in Betrieb genommen werden. In einem ersten Teil soll es rein darum gehen, den TDC7200 IC genauer kennenzulernen.

Bemerkung: In der Realität wird eine solche Auflösung nicht erreichbar sein. Die Herausforderungen bestehen beispielsweise darin, dass in der gesamten Signalkette diverse Verzögerungen vorhanden sind. So spielen zum Beispiel das Einschalten der Laser-Diode, die Bandbreite des Verstärkers sowie die Schaltzeit des Komparators eine grosse Rolle. Auch der Jitter der MCU beim Ein- und Ausschalten ihrer Ausgänge wird bei der minimalen Auflösung ins Gewicht fallen. Im Kapitel 6.1 wird mittels verschiedener Messungen dargestellt, wie nah an die vom TDC7200 vorgegebene Grenze herangekommen wird.

### 3.3 Vorgehensweise

Wie bereits erläutert, ist davon auszugehen, dass der gesamte Signalpfad eine Ungenauigkeit in die Messung einbringt, welche weit ausserhalb derjenigen des TDC7200 liegt. Es bietet sich also an, dass diese im Vorfeld untersucht wird. Dazu wird ein zweiter TDC7200 auf der Elektronik eingesetzt, welcher zur Messung eines rein elektrischen Signals zuständig ist.

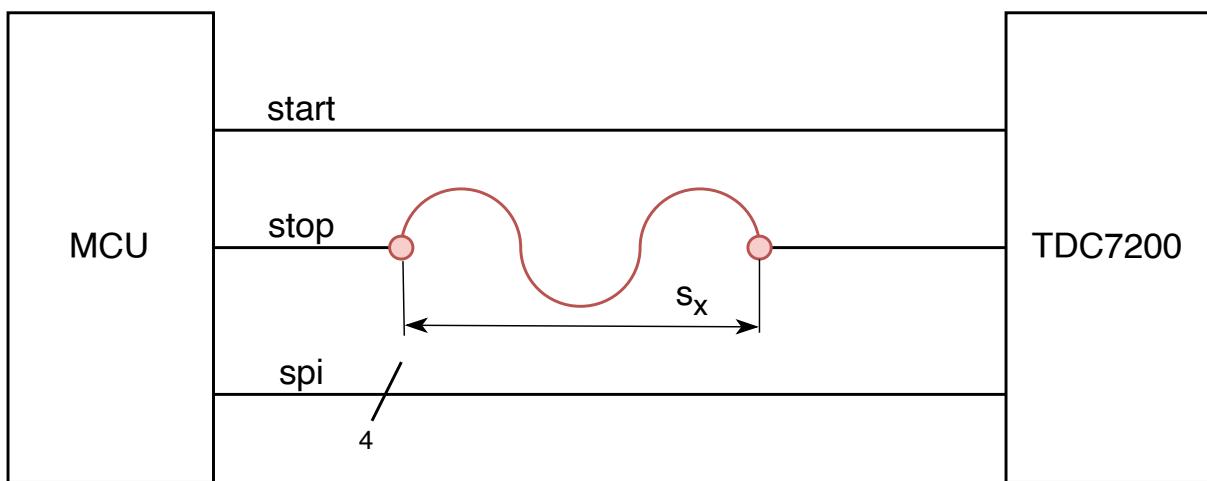


Abbildung 6: Konzept Betrieb TDC rein elektrisch

In Abbildung 6 sei skizziert, wie eine solche Schaltung aussehen könnte. Die gewellte, rot markierte Linie versinnbildlicht dabei einen variablen Messpfad. Über einen Stecker können hier unterschiedlich lange Kabel angeschlossen werden. Somit ist es möglich, die Messgenauigkeit des TDC7200 rein elektrisch nachzuweisen.

Folgende Punkte können mit einer solchen Erweiterung überprüft werden:

- Messung Jitter der Ausgänge der verwendeten MCU
- Konfiguration und Inbetriebnahme TDC7200 (z.B. verschiedene Messmodi)
- Auswertung der Resultate des TDC7200

## 4 Simulationen

In diesem Kapitel werden die durchgeführten Simulationen dokumentiert.

Zur Simulation wird das Tool "Xpedition AMS" von Siemens verwendet. Wie der Name sagt, unterstützt dieses Tool Analog/Mixed Signal (AMS) Analyse. (Siemens, 2025)

### 4.1 Laser Treiber

In Abbildung 7 ist das Schema für die Simulation des Laser-Treibers dargestellt. Dies entspricht der Schaltung aus Kapitel 5.1.7.

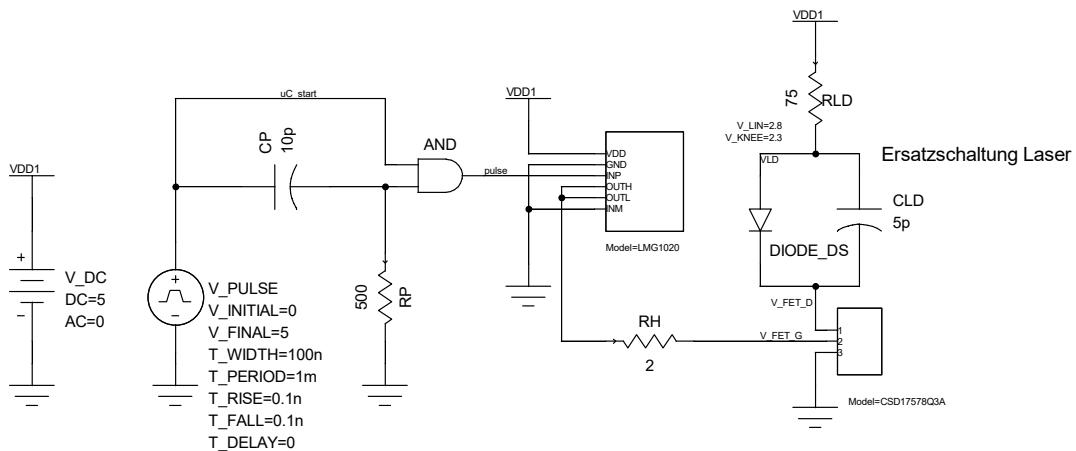


Abbildung 7: Simulation Laser-Treiber - Schema

Das SPICE-Modell des Gate-Treibers LMG1025-Q1 ist dasselbe wie für LMG1020 und wurde von der Website von TI heruntergeladen (TI, 2024b).

Das SPICE-Modell des NexFET CSD17578Q3A wurde ebenfalls von der Website von TI heruntergeladen (TI, 2024a).

Für die Laser-Diode RLD65NZX1 konnte kein SPICE-Modell gefunden werden, es wurde deshalb eine Ersatzschaltung aus Diode und paralleler, parasitärer Kapazität eingefügt.

Das Resultat der Simulation ist in Abbildung 8 dargestellt.

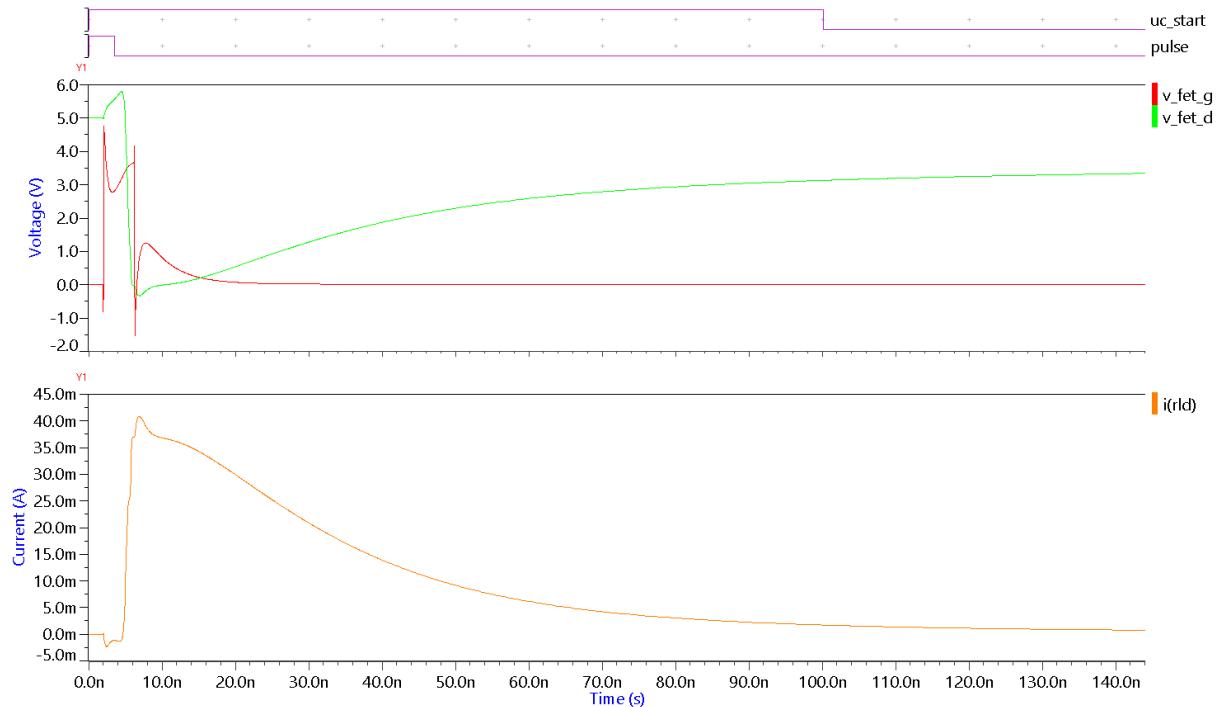


Abbildung 8: Simulation Laser-Treiber - Plot

Es ist zu erkennen, dass der Laser sehr schnell einschalten kann (2 ... 3 ns). Das Ausschalten dauert etwas länger (ca. 100 ns). Da die ToF bis zur ersten Flanke gemessen wird, ist eine längere Ausschaltzeit für diese Anwendung kein Problem.

## 4.2 Transimpedanzverstärker

In Abbildung 9 ist das Schema für die Simulation des Transimpedanzverstärkers dargestellt. Dies entspricht der Schaltung aus Kapitel 5.1.8, ohne den Komparator.

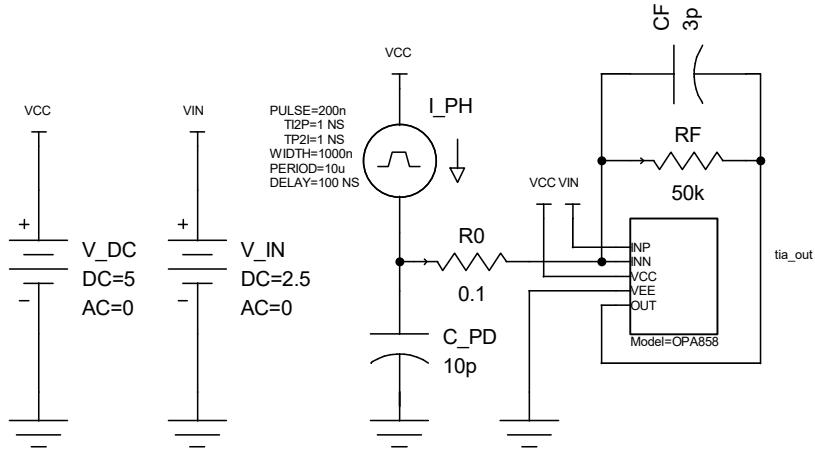


Abbildung 9: Simulation TIA - Schema

Das SPICE-Modell des Operationsverstärkers OPA858 wurde von der Website von TI heruntergeladen (TI, 2024d).

Für die Photo-Diode NJL6401R konnte kein SPICE-Modell gefunden werden, es wurde deshalb eine Ersatzschaltung aus Strompuls-Quelle und paralleler, parasitärer Kapazität eingefügt.

Das Resultat der Simulation ist in Abbildung 10 dargestellt.

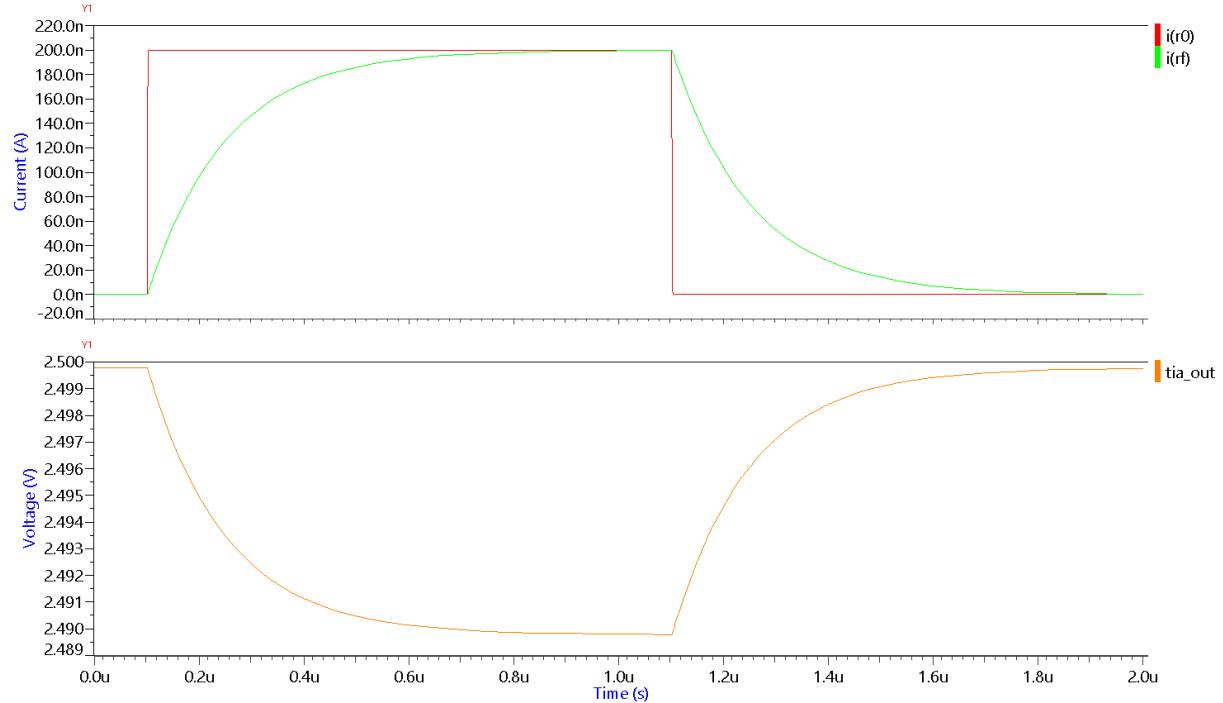


Abbildung 10: Simulation TIA - Plot

Es ist zu erkennen, dass die Zeitkonstante des Stroms durch den Feedback-Widerstand  $R_F$  gemäss Formel 18 bestimmt werden kann.

$$\tau = R_F \cdot C_F = 50 \text{ } k\Omega \cdot 3 \text{ } pF = 150 \text{ } ns \quad (18)$$

Beliebig klein kann der Feedback-Kondensator  $C_F$  jedoch nicht gewählt werden, da die Schaltung sonst schwingt.

In Abbildung 11 ist das Simulations-Resultat ohne den Feedback-Kondensator  $C_F$  dargestellt.

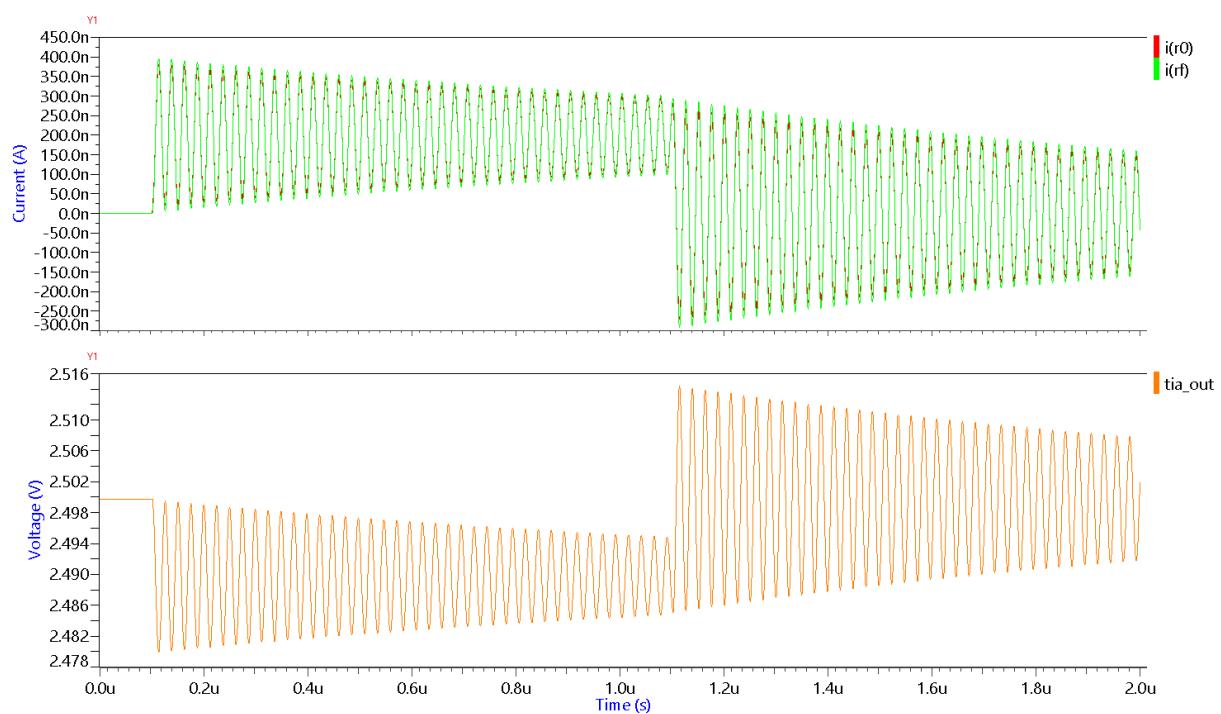


Abbildung 11: Simulation TIA - Plot ohne  $c_F$

Die genaue Dimensionierung von  $c_F$  wird durch Tests mit dem PCB bestimmt werden. Eventuell, je nach parasitären Kapazitäten auf dem PCB, wird  $c_F$  auch gar nicht bestückt werden müssen.

## 5 Umsetzung

In diesem Kapitel ist das Umsetzen der Hardware sowie Firmware des ToF-Demonstrators dokumentiert.

Im Anhang 8.4 bis und mit 8.9 sind sämtliche technischen Dokumente und einige Fotos des fabrizierten Demonstrators abgelegt.

### 5.1 Schaltungen

Nachfolgend werden sämtliche Teil-Schaltungen thematisiert, welche für das entwickelte ToF-Evaluationsmodul nötig sind. Ein vollständiges Schema kann dem Anhang 8.4 entnommen werden. Die kompletten Projekt-Dateien sind im elektronischen Anhang dieses Projektes verfügbar.

Gezeichnet wird das Schema mit Hilfe des Open-Source Tools "KiCad EDA 8.0" (KiCad, 2025).

#### 5.1.1 Selective Input Voltage

Abbildung 12 zeigt die Beschaltung zur Selektion der Speisung.

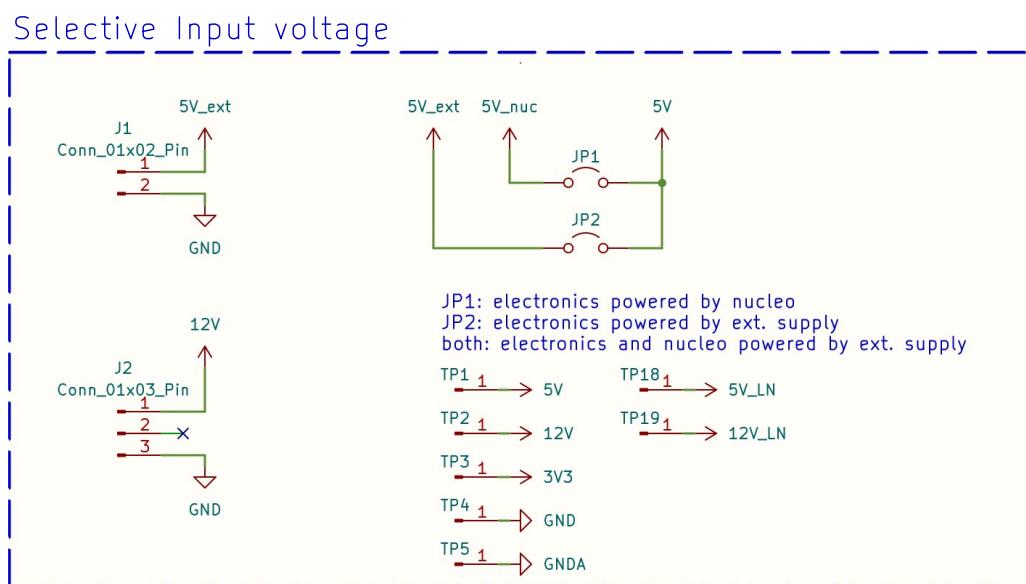


Abbildung 12: Selective Input Voltage

Für die Speisung des Nucleo-Boards bestehen die folgenden Möglichkeiten:

- 5 V von USB-Buchse
- 5 V von externem Power-Supply (JP1 + JP2)
- 12 V von externem Power-Supply (JP3)

Siehe dazu auch Kapitel 5.1.2.

Für die Speisung der 5 V Elektronik bestehen die folgenden Möglichkeiten:

- 5 V von Nucleo-Board (JP1)
- 5 V von externem Power-Supply (JP2)
- 12 V von externem Power-Supply via Nucleo-Board (JP1 + JP3)

Für die Speisung der Photodiode bestehen die folgenden Möglichkeiten:

- 5 V von 5 V-Elektronik (sw2 Position 3)
- 12 V von externem Power-Supply (sw2 Position 1)

Siehe dazu auch Kapitel 5.1.8.

### 5.1.2 Nucleo Board

Die Beschaltung des NUCLEO-F042K6 Boards (ST, 2024) ist in Abbildung 13 gezeigt.

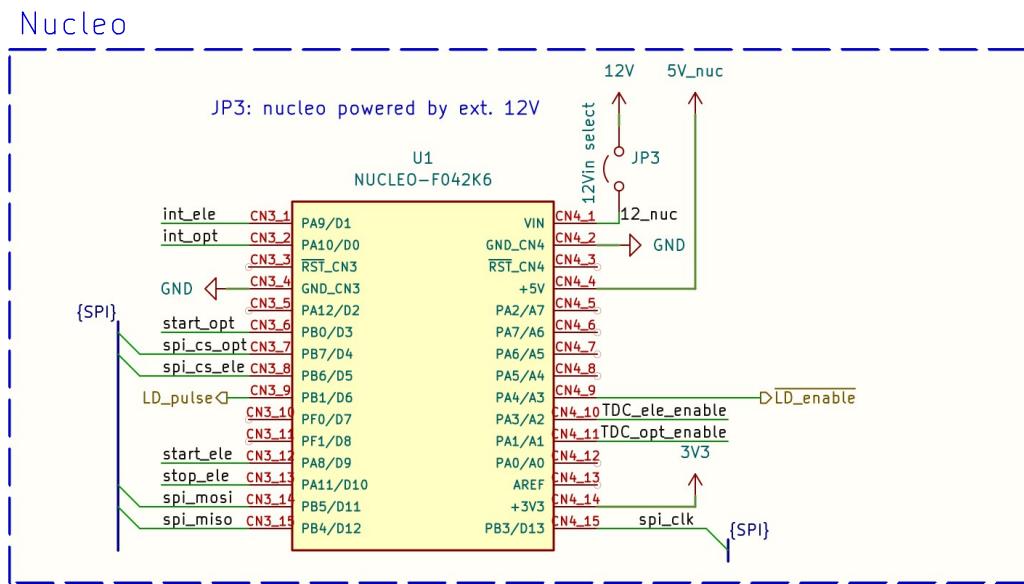


Abbildung 13: Nucleo Board

Das NUCLEO-Board ist ein sogenanntes "Development-Kit", welches eine STM32F042K6 MCU beinhaltet. Am Rand des Boards werden diverse Pins der Microcontroller Unit (MCU) via Pin-Header zur Verfügung gestellt. Dies erleichtert die Integration in eine eigene Elektronik enorm. Geflasht wird die MCU über eine USB-Schnittstelle.

In diesem Design wird das Entwicklungs-Board dazu benötigt, die TDC7200 ICs zu bedienen. Dazu wird ein Serial Peripheral Interface (SPI) verwendet, um die Mess-ICs zu konfigurieren und auszulesen (in der Abbildung als SPI-Bus zusammengefasst). Mit den Signalen `start_ele` bzw. `start_opt` können auf den TDCs Messungen gestartet werden. Für den TDC, welcher für die elektrischen Signale zuständig ist, kann zudem mit `stop_ele` ein Stopp-Puls generiert werden. Zu guter Letzt ist das NUCLEO dafür zuständig, die Laser-Diode anzusteuern, was mit den Signalen `LD_pulse` sowie `LD_enable` geschieht.

### 5.1.3 TDC Electrical Signal

Die Beschaltung des TDC7200 (TI, 2016b) für den elektrischen Teil ist in Abbildung 14 gezeigt.

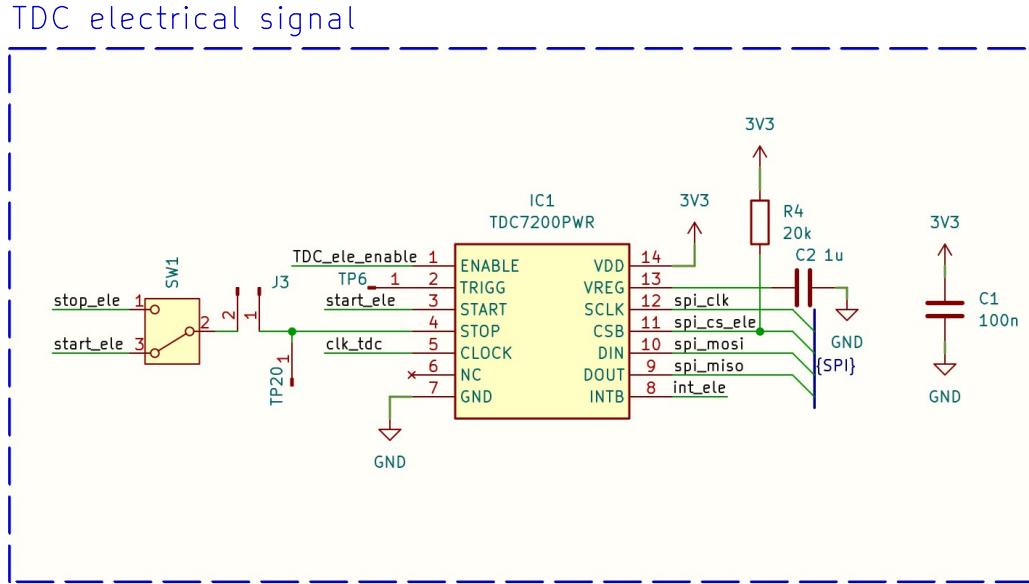


Abbildung 14: TDC Electrical Signal

Der TDC7200 ist über die Start- und Stopp-Leitungen, ein Enable-Signal sowie die SPI-Leitungen mit der MCU verbunden.

Wie bereits im Kapitel 3.3 angesprochen, wird der elektrische TDC dazu verwendet, den Chip erstmalig in Betrieb zu nehmen und sich damit vertraut zu machen. Am Anschluss J3 kann in einem nächsten Schritt ein beliebig langes Kabel angeschlossen werden. Dies ermöglicht es, erste Messresultate, natürlich rein elektrisch, vom TDC7200 auszulesen.

Mittels Schalter SW1 kann das STOP-Signal wahlweise via stop\_ele oder start\_ele generiert werden. Dies bietet zum einen die Möglichkeit, die Zeit zwischen dem Schalten von zwei GPIOs zu messen. Zum anderen kann derselbe GPIO-Pin zum Generieren des START- und STOP-Signals verwendet werden, wodurch von der Verzögerungszeit direkt auf die Kabellänge an J3 geschlossen werden kann.

### 5.1.4 TDC Optical Signal

Die Beschaltung des TDC7200 (TI, 2016b) für den optischen Teil ist in Abbildung 15 gezeigt.

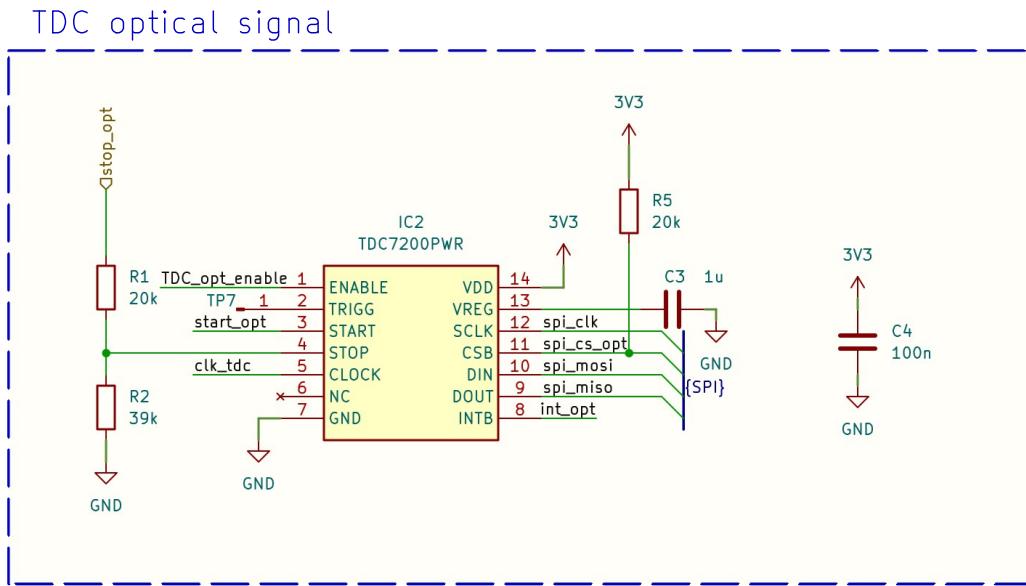


Abbildung 15: TDC Optical Signal

Die Beschaltung des TDCs für die optische Messung gestaltet sich praktisch gleich wie beim elektrischen Gegenstück. Der Hauptunterschied ist hier aber, dass dessen STOP-Signal nicht von der MCU selber generiert wird, sondern von einem Komparator, welcher am Ende des optischen Messpfades steht. Da der Komparator mit einem 5 V Pegel arbeitet, ist der Spannungsteiler  $R_1 / R_2$  vonnöten, welcher den 5 V Puls auf die geeigneten 3.3 V herunterteilt.

### 5.1.5 Oscillator For TDCs

Die Beschaltung des Oszillators für die beiden TDC ist in Abbildung 16 gezeigt. Es handelt sich hierbei um einen normalen Quartz-Oszillatator mit integriertem Schwingkreis. Praktischerweise ist bei diesem also keine weitere Beschaltung notwendig.

Oscillator for TDCs

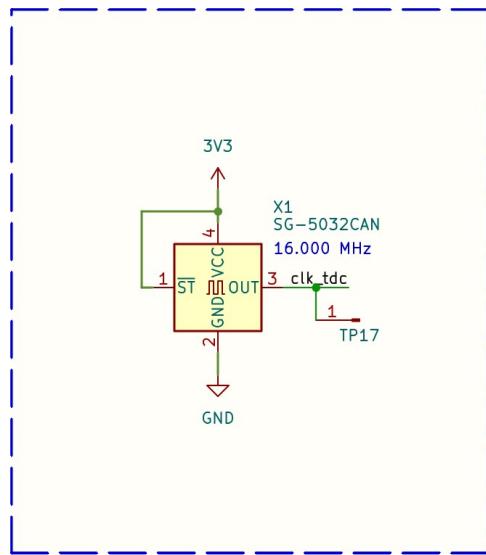


Abbildung 16: Oscillator for TDCs

### 5.1.6 Power Supply Separation

Für die Beschaltung der Photodiode, inkl. TIA und Komparator, macht es Sinn eine Spannungsversorgung mit möglichst wenig Rauschen zu haben.

Dazu wurde die Separierung vorgenommen, welche in Abbildung 17 dargestellt ist.

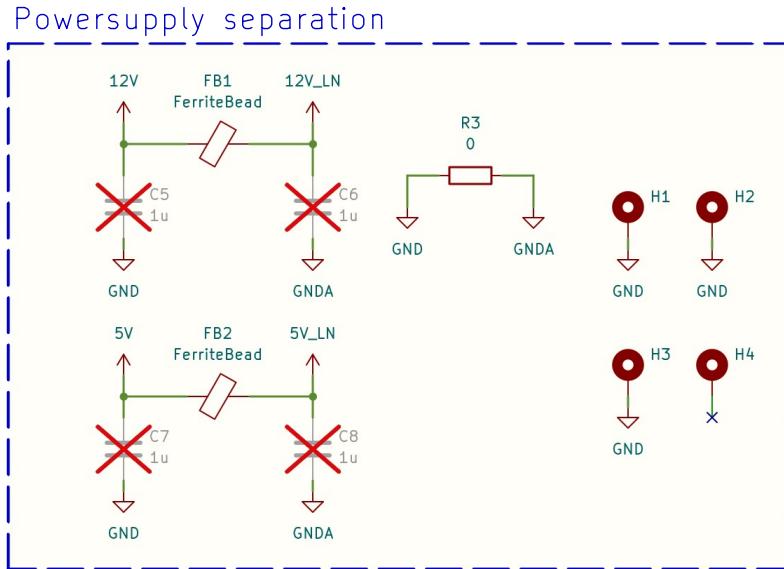


Abbildung 17: Power Supply Separation

Prinzipiell sollen die Speisungen über eine Ferrit-Perle etwas entstört werden. Massgeblich ist hier natürlich auch der physikalische Verlauf des Speisungspfades auf dem Layout. Dazu mehr im entsprechenden Kapitel 5.2. Wahlweise besteht nebst den Ferrit-Perlen die Möglichkeit, mittels

Kondensatoren die Speisung weiter zu entkoppeln. Es wird jedoch davon ausgegangen, dass dies in einem ersten Schritt nicht notwendig ist.

### 5.1.7 Laser Driver

Die Laser Diode RLD65NZX1 (ROHM, 2019) wird mittels Lasertreiber LMG1025-Q1 (TI, 2024c) und NexFET (TI, 2016a) angesteuert. Das AND-Gatter IC21 (Diodes Inc., 2020) implementiert zusammen mit dem Hochpass (C21, R21 und RV21) einen Pulsverkürzer. Über diesen Pfad lassen sich für den Laser Pulse im Bereich von 0.5 ... 100 ns generieren. Siehe dazu Abbildung 18.

Der Widerstand R25 bildet den Vorwiderstand für die Laser-Diode, welcher sich gemäss der Formel 19 berechnet.

$$R_v = \frac{V_{cc} - V_{fld}}{I_{ld}} = \frac{5 \text{ V} - 2 \text{ V}}{40 \text{ mA}} = 75 \Omega \quad (19)$$

Im Schema eingezeichnet ist aktuell ein Platzhalterwert. Während der Inbetriebnahme wird der Widerstand je nach Bedarf verändert. Wird dieser verkleinert, so vergrössert sich der Strom durch die Laser-Diode und somit die ausgesandte Lichtleistung.

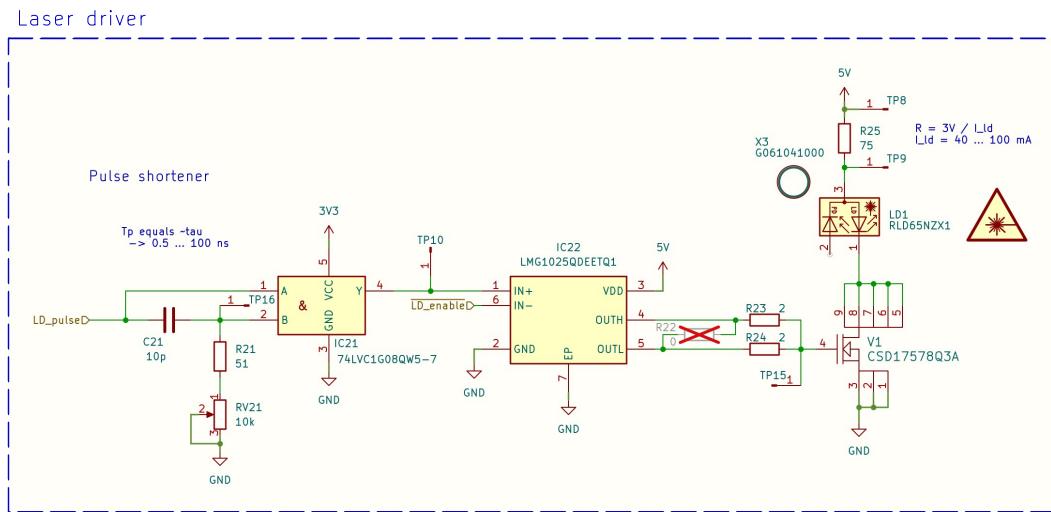


Abbildung 18: Laser Driver

### 5.1.8 Photo Receiver

Um den Photostrom der Photodiode NJL6401R (JRC, 2014) zu verstärken und in eine Spannung umzuwandeln, wird mit dem Operationsverstärker OPA858 (TI, 2018) ein Transimpedanzverstärker aufgebaut. Der Ausgang des Transimpedanzverstärkers geht auf den Komparator TLV3501 (TI, 2016c), um das stop-Signal für den TDC zu generieren. Siehe dazu Abbildung 19.

Der Feedback-Widerstand R26 kann je nach Bedarf verändert werden. Wird dieser vergrössert, so steigt auch der negative Puls am Ausgang des TIAs. Mit C20 kann zudem bei Bedarf eine kleine Kapazität hinzugefügt werden. Diese hat zum Ziel, ein Schwingen des Verstärkers zu unterdrücken, falls dieser eine Neigung zum Schwingen haben sollte.

Der Offset des OPA858 und die Schaltschwelle des TLV3501 werden mit Hilfe der beiden Trimmer RV22 und RV23 einstellbar gemacht. Dies hilft bei der Inbetriebnahme, da im Voraus noch nicht ganz klar ist, wo der Ausgang des Operationsverstärkers aufgrund des gewandelten Dunkelstroms der Photodiode zu liegen kommt.

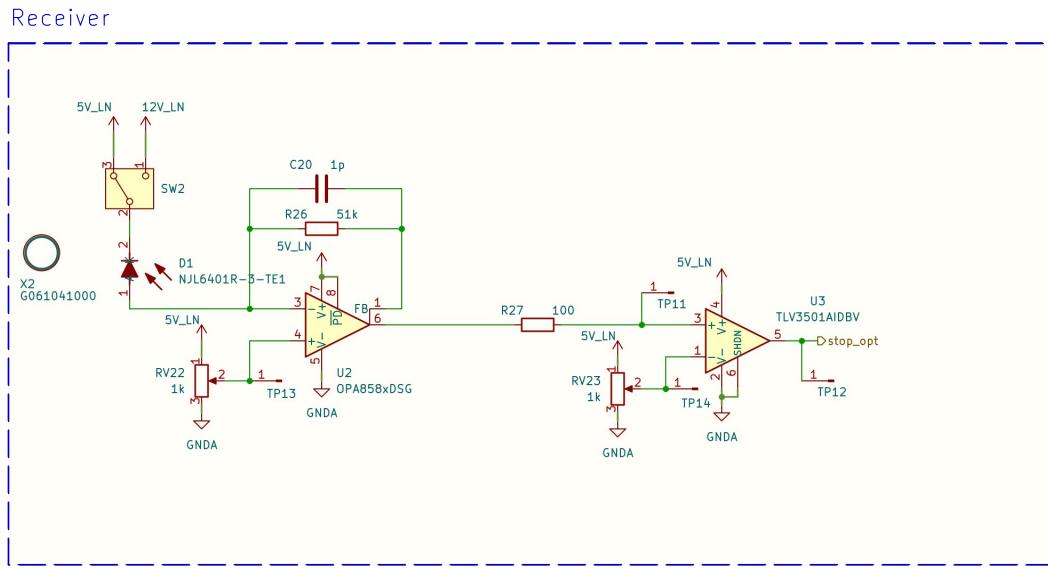


Abbildung 19: Photo Receiver

### 5.1.9 Decoupling Capacitors

Die Beschaltung der Entkopplungs-Kondensatoren ist in Abbildung 20 dargestellt. Die Kondensatoren für den Operationsverstärker OPA858 und für den Komparator TLV3501 werden dem Vorschlag im Datenblatt entnommen. Zusätzlich wird auch der Gate-Treiber beim Laser-Treiber mit einer zusätzlichen, grösseren Kapazität gestützt, was sich positiv auf die Stabilität der Speisung auswirken sollte.

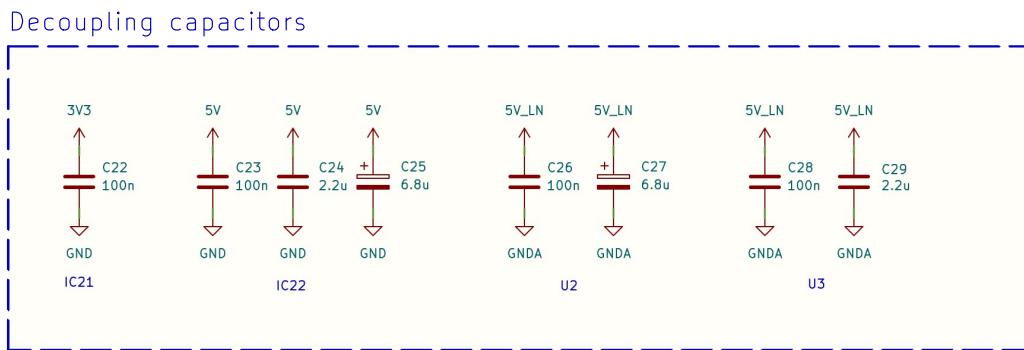


Abbildung 20: Decoupling Capacitors

## 5.2 Layout

Beim Layout wird ein Augenmerk darauf gelegt, dass die verschiedenen Schaltungen sich auf dem PCB nicht gegenseitig stören. Dafür ist ein sauberes Speisungskonzept im Einsatz, welches die Ground- und Power-Planes nach Layer definiert. Prinzipiell sind die vier Lagen der Leiterplatte wie folgt aufgetrennt:

- Top-Layer: Signal und Ground
- Innerer Layer 1: Stromversorgungsebene
- Innerer Layer 2: Signal und Ground
- Bottom-Layer: Signal und Ground

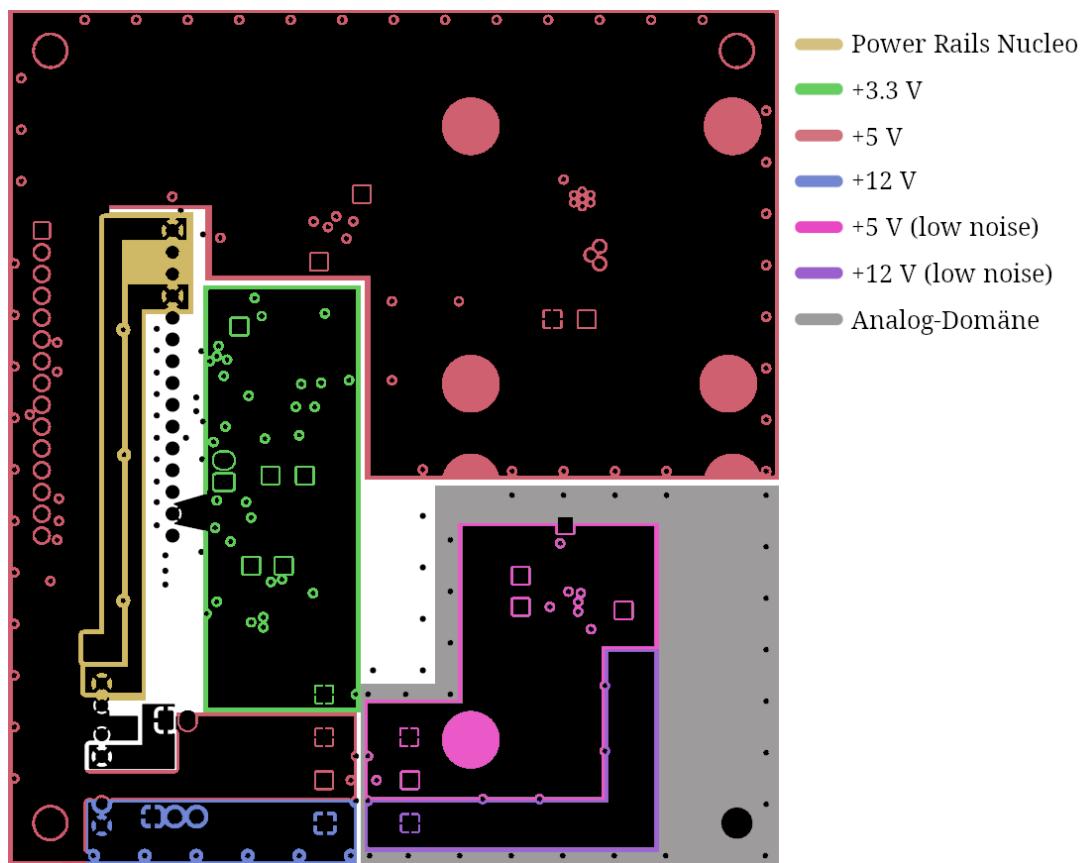


Abbildung 21: Innerer Layer 1 mit Vermerken zum Speisungskonzept

In Abbildung 21 ist die Umsetzung des Speisungskonzeptes ersichtlich. In der unteren, rechten Ecke befindet sich die Analog-Domäne, welche die Photodiode, den TIA und den Komparator beinhaltet. Dieser Abschnitt des PCBs besitzt eigene Ground-Planes sowie entstörte Speisungen. Mit Hilfe von einigen Vias an der Aussengrenze dieses Abschnittes sollte die Analog-Domäne zudem weniger Anfällig auf extern induzierte Störfelder sein.

Solche sogenannten "Stitching-Vias" sind zudem rund um das ganze PCB zur Abschirmung platziert.

Das komplette Layout kann dem Anhang 8.6 entnommen werden.

### 5.3 3D View

Beim Design des Layouts und vor allem bei der Komponentenplatzierung kann eine 3D-Ansicht unterstützend wirken. Dies, weil ersichtlich gemacht wird, wo sich mechanische und elektronische Komponenten in den Weg kommen könnten. Die Abbildung 22 zeigt eine 3D-Wiedergabe der Aufsicht auf den ToF-Demonstrator.

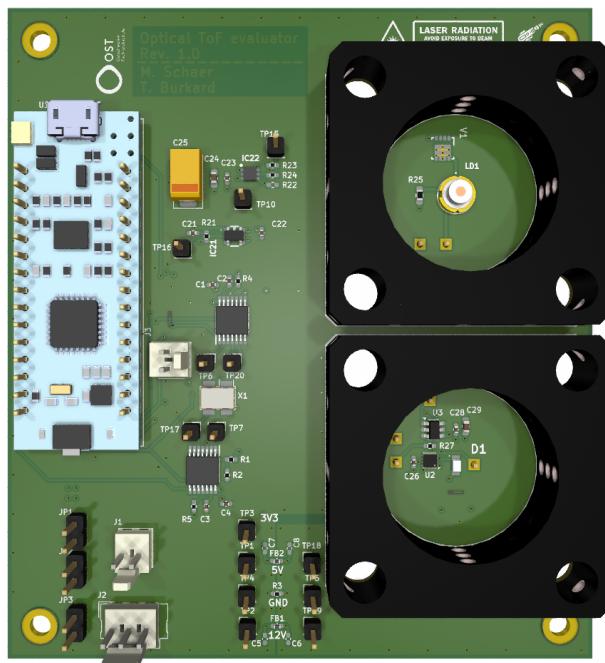


Abbildung 22: 3D View Top

Die Linsenhalterungen sind mit Hilfe von einer Step-Datei, die von QIOPTIQ zur Verfügung gestellt wird in die Ansicht integriert (Qioptiq, 2024b). In der Abbildung 22 werden diese als schwarze Quader mit zylinderförmigen Ausschnitten dargestellt. Direkt unter diesen Halterungen dürfen sich keine Komponenten befinden, da sich die Halterungen ansonsten nicht montieren lassen.

Bei der Komponentenplatzierung wird zusätzlich auch darauf geachtet, dass sämtliche Testpunkte sowie die Trimmer und Schalter zur Konfiguration gut zugänglich sind. Wenn sich diese Komponenten nahe an den optoelektronischen Komponenten befinden, findet sich deren Platzierung auf der Unterseite. Somit sind sie auch mit montierter Optik verfügbar.

## 5.4 Fabrikation

KiCad 8.0 ermöglicht es, aus dem Layout-Design diverse Fabrikations-Dateien für die verschiedenen Layer zu erzeugen. Diese sogenannten Gerber und NC-Drill Dateien können von einem Leiterplattenhersteller zur Produktion einer Leiterplatte verwendet werden. Wahlweise kann eine Leiterplatte auch direkt bestückt werden, was jedoch zusätzliche Zeit in der Produktion beansprucht.

Da sich die meisten Komponenten auf dem PCB relativ gut bestücken lassen, wird auf eine automatisierte Bestückung verzichtet und die Komponenten werden von Hand bestückt.

Die Komponenten werden bei Mouser Electronics bestellt, während die Leiterplatten vom chinesischen Hersteller JLCPCB (JLC, 2025) bezogen werden. Mouser erfüllt die erwartete Lieferzeit von einigen Tagen. Die Bestellung von JLCPCB kommt mit genau sieben Tagen ebenfalls pünktlich an. Die Ingenieure von JLCPCB müssen keine Rückfragen zum Design stellen, was sich wohl positiv auf die Lieferzeit auswirkt.

Die fertig bestückte Leiterplatte ist in der Abbildung 23 ersichtlich. Weitere Fotografien des ToF-Demonstrators können dem Anhang 8.9 entnommen werden.



Abbildung 23: Demonstrator von vorne ohne Linse

## 5.5 Firmware

Bei der Firmware wird darauf geachtet, dass eine saubere Trennung zwischen Applikation, Treiber und Peripherie vorhanden ist. Dies sorgt für ein möglichst modulares Design. Die FW-Hierarchie ist in der Abbildung 24 nachvollziehbar.

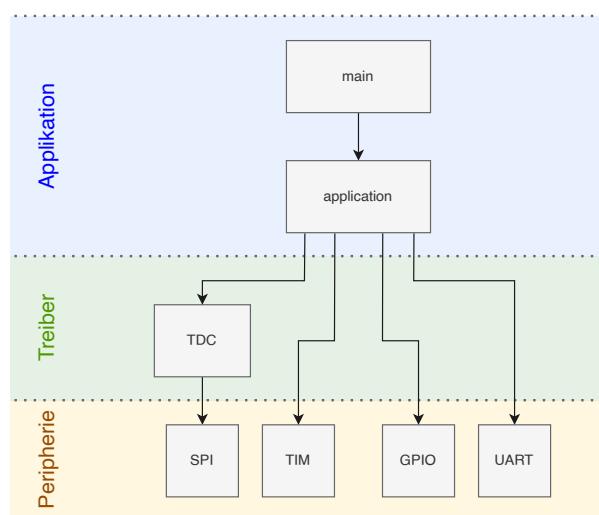


Abbildung 24: Aufbau Firmware-Projekt

Die Applikation ist für das Timing der Messungen verantwortlich. Der Code für das auslösen wird während der Messphase schrittweise ausgebaut, erst über die Verwendung einfacher GPIOs, bis hin zu optimalem zeitlichen Verhalten unter Einsatz verschiedener Hardware-Timer für das Generieren der Trigger-Pulse für die TDCs. Im Kapitel 6.1 sind die einzelnen Erweiterungsschritte detaillierter erklärt.

Der TDC-Treiber stellt Funktionen zur Initialisierung und zum Aktivieren des TDC zur Verfügung. Siehe dazu Code 1.

```
1 TDC_init(&tdc_opt, &hspi1, SPI_CS_OPT_GPIO_Port, SPI_CS_OPT_Pin,  
          TDC_OPT_ENABLE_GPIO_Port, TDC_OPT_ENABLE_Pin);  
  
2  
3 TDC_enable(&tdc_opt);
```

Code 1: TDC Init und Enable mit Treiber

Der TDC-Treiber beinhaltet die komplette Registerdefinition des TDC7200. Der Treiber stellt zudem einfache Lese- und Schreib-Routinen für die Kommunikation via SPI zur Verfügung. Damit kann das Mess- IC mit Leichtigkeit umkonfiguriert, aufgesetzt und ausgelesen werden kann. Ein Beispiel ist in Code 2 dargestellt.

```
1 TDC_read(&tdc_ele, TDC_ADR_CONFIG1, data);  
2  
3 data[0] |= TDC_CONFIG1_MEAS_MODE_2;  
4  
5 TDC_write(&tdc_ele, TDC_ADR_CONFIG1, data);
```

Code 2: TDC Read/Write mit Treiber

Weiter kann der Treiber die Messresultate direkt umrechnen, anhand der im Datenblatt vorgegebenen Formeln. Im Code 3 ist der Beispiel-Code für eine einzelne Messung ersichtlich.

```
1 uint64_t tof_fs = 0;  
2  
3 TDC_start(&tdc_opt);  
4  
5 /* ... generate start and stop signals */  
6  
7 if (TDC_read_result(&tdc_opt, &tof_fs) != TDC_OK) {  
8     for (;;) {} // Error handling...  
9 }  
10  
11 /* tof_fs now holds the measurement result, in [fs] */
```

Code 3: TDC Messung mit Treiber

Der selbst entwickelte Firmware-Treiber für den TDC befindet im Anhang 8.2.

Mit der Entwicklungsumgebung STM32CubeIDE / CubeMX (ST, 2025) ist es möglich, das Pinout der MCU grafisch zu konfigurieren. Mit Hilfe eines Code-Generators können die Initialisierungs-Routinen für sämtliche Peripherien automatisiert erzeugt werden. Somit kann während der Entwicklung der Fokus auf den acrshorttdc-Treiber sowie die eigentliche Applikation gelegt werden.

Die Pin-Konfiguration ist in der Abbildung 25 ersichtlich.

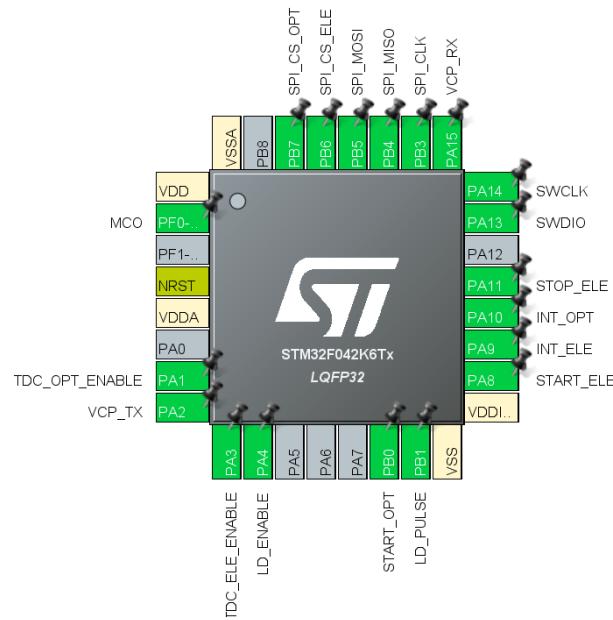


Abbildung 25: Pin-Konfiguration des STM32F042K6

# 6 Resultate

In diesem Kapitel werden die Resultate dokumentiert.

Die verwendeten Python-Skripte zur Berechnung der statistischen Größen und zum Plotten der Diagramme befinden sich im Anhang 8.3.

## 6.1 Elektrische Messungen

In diesem Teilkapitel werden die Messresultate dokumentiert, welche rein elektrisch (also ohne optischen Teil) erfasst wurde.

Das verwendete Digital Storage Oscilloscope (DSO) ist ein Rohde & Schwarz RTB2004 1.25 GSa/s.

Die Zeitmessenungen werden von `IC1` (siehe Abbildung 14) durchgeführt und von der Firmware, welche auf dem Nucleo Board `U1` (siehe Abbildung 13) getriggert und ausgelesen.

### 6.1.1 GPIO Toggle mit HAL

Als erstes wird gemessen, wie lange es für die CPU der MCU dauert mittels Hardware Abstraction Layer (HAL) - Library (ST, 2020) zwei GPIO-Pins zu schalten.

In Code 4 ist die Firmware-Implementation dazu gezeigt.

```
1 // Configure TDC
2
3 TDC_init(&tdc_ele, &hspi1, SPI_CS_ELE_GPIO_Port, SPI_CS_ELE_Pin,
4           TDC_ELE_ENABLE_GPIO_Port, TDC_ELE_ENABLE_Pin);
5 TDC_enable(&tdc_ele);
6
7 TDC_read(&tdc_ele, TDC_ADR_CONFIG1, data);
8 data[0] |= TDC_CONFIG1_MEAS_MODE_2;
9
10 TDC_write(&tdc_ele, TDC_ADR_CONFIG1, data);
11
12 while (1) {
13     TDC_start(&tdc_ele);
14
15     // Set Pins to High with HAL
16     HAL_GPIO_WritePin(START_ELE_GPIO_Port, START_ELE_Pin, GPIO_PIN_SET);
17     HAL_GPIO_WritePin(STOP_ELE_GPIO_Port, STOP_ELE_Pin, GPIO_PIN_SET);
18
19     TDC_read_result(&tdc_ele, &tof_fs);
20
21     sprintf(string, "ToF = %lu [ps]\n", tof_fs / 1000);
22     HAL_UART_Transmit(&huart2, (uint8_t *)string, strlen(string), 10000);
23
24     // Set Pins to Low with HAL (preparation for next iteration)
25     HAL_GPIO_WritePin(START_ELE_GPIO_Port, START_ELE_Pin, GPIO_PIN_RESET);
26     HAL_GPIO_WritePin(STOP_ELE_GPIO_Port, STOP_ELE_Pin, GPIO_PIN_RESET);
27
28     HAL_Delay(10); // 10 ms
29 }
```

Code 4: GPIO Toggle mit HAL

Der TDC misst also die Zeit zwischen Zeile 15 und 16 in Code 4. Dazu wird der STOP-Pin des TDC via `sw1` mit `stop_ele` verbunden. Der Kabelanschluss `j3` wird kurzgeschlossen. Siehe Abbildung 14.

Via UART wurden 2000 Messwerte erfasst. Ein Ausschnitt davon ist in Code 5 gezeigt. Die restlichen Daten befinden sich im elektronischen Anhang.

```

1 ToF = 6375779 [ps]
2 ToF = 6374666 [ps]
3 ToF = 6377003 [ps]
4 ToF = 6375333 [ps]
5 ToF = 6377061 [ps]
6 ToF = 6374721 [ps]
7 ToF = 6377504 [ps]
8 ToF = 6374888 [ps]
9 ToF = 6376725 [ps]
10 ToF = 6377005 [ps]
11 ...

```

Code 5: GPIO Toggle mit HAL

Arithmetischer Mittelwert und Standardabweichung sind in Formel 20 aufgeführt.

$$\overline{ToF} = 6'375'888 \text{ ps} \quad (20)$$

$$\sigma = 1'059 \text{ ps}$$

Da die CPU mit 8 MHz läuft, lässt sich daraus schliessen, dass ein Pin-Toggle mit HAL ca. 50 CPU-Cycles benötigt. Dies erscheint plausibel.

Histogramm und Boxplot sind in Abbildung 26 bzw. 27 dargestellt.

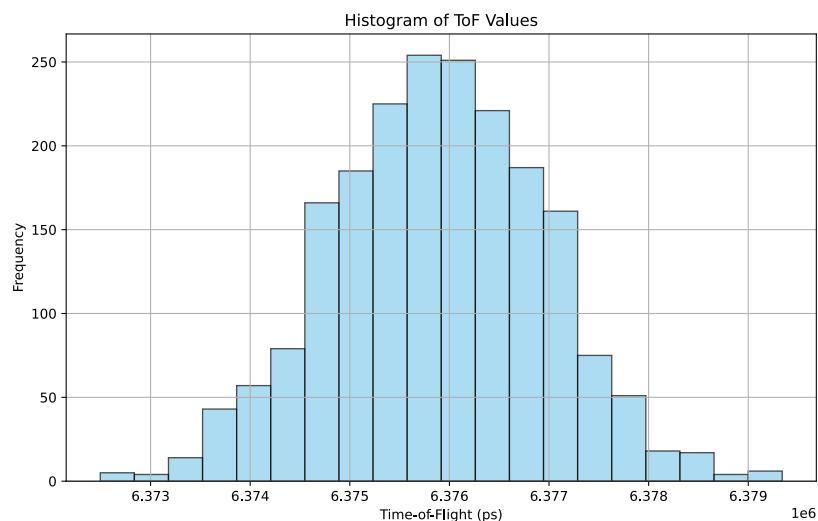


Abbildung 26: GPIO Toggle mit HAL - Histogramm

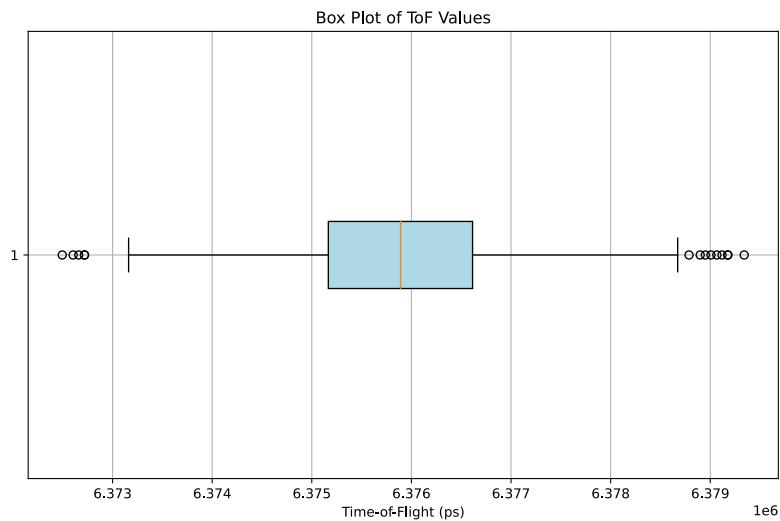


Abbildung 27: GPIO Toggle mit HAL - Boxplot

Um die Resultate des TDC zu validieren, wurde dieselbe Messung auch mittels Digital Storage Oscilloscope (DSO) durchgeführt. Die Messungen sind in Abbildung 28 und 29 dargestellt.



Abbildung 28: GPIO Toggle mit HAL - DSO



Abbildung 29: GPIO Toggle mit HAL - DSO (Zoom)

### 6.1.2 GPIO Toggle ohne HAL

Als nächstes wird gemessen wie lange es für die CPU der MCU dauert mit direktem Register-Zugriff (via Pointer; ohne HAL-Library) zwei GPIO-Pins zu schalten.

In Code 6 ist die Firmware-Implementation dazu gezeigt.

```

1 // Configure TDC
2
3 TDC_init(&tdc_ele, &hspi1, SPI_CS_ELE_GPIO_Port, SPI_CS_ELE_Pin,
4   TDC_ELE_ENABLE_GPIO_Port, TDC_ELE_ENABLE_Pin);
5 TDC_enable(&tdc_ele);
6
7 TDC_read(&tdc_ele, TDC_ADR_CONFIG1, data);
8 data[0] |= TDC_CONFIG1_MEAS_MODE_2;
9
10 TDC_write(&tdc_ele, TDC_ADR_CONFIG1, data);
11
12 while (1) {
13   TDC_start(&tdc_ele);
14
15   // Set Pins to High with direct register access
16   *((volatile uint32_t*)(GPIOA_BASE + 0x14)) |= (1 << 8); // START_ELE
17   *((volatile uint32_t*)(GPIOA_BASE + 0x14)) |= (1 << 11); // STOP_ELE
18
19   TDC_read_result(&tdc_ele, &tof_fs);
20
21   sprintf(string, "ToF = %lu [ps]\n", tof_fs / 1000);
22   HAL_UART_Transmit(&huart2, (uint8_t *)string, strlen(string), 10000);
23
24   // Set Pins to Low with direct register access (preparation for next
25   // iteration)
26   *((volatile uint32_t*)(GPIOA_BASE + 0x14)) &= ~(1 << 8); // START_ELE
27   *((volatile uint32_t*)(GPIOA_BASE + 0x14)) &= ~(1 << 11); // STOP_ELE
28
29   HAL_Delay(10); // 10 ms

```

### Code 6: GPIO Toggle ohne HAL

Der TDC misst also die Zeit zwischen Zeile 15 und 16 in Code 6. Dazu wird, wie in Kapitel 6.1.1, der STOP-Pin des TDC via `sw1` mit `stop_ele` verbunden. Der Kabelanschluss `j3` wird kurzgeschlossen. Siehe Abbildung 14.

Via UART wurden 2000 Messwerte erfasst. Ein Ausschnitt davon ist in Code 7 gezeigt. Die restlichen Daten befinden sich im elektronischen Anhang.

```

1 ToF = 1377894 [ps]
2 ToF = 1378450 [ps]
3 ToF = 1377615 [ps]
4 ToF = 1377840 [ps]
5 ToF = 1377729 [ps]
6 ToF = 1377615 [ps]
7 ToF = 1377337 [ps]
8 ToF = 1378063 [ps]
9 ToF = 1377949 [ps]
10 ToF = 1377615 [ps]
11 ...

```

### Code 7: GPIO Toggle ohne HAL

Arithmetischer Mittelwert und Standardabweichung sind in Formel 21 aufgeführt.

$$\overline{ToF} = 1'377'773 \text{ ps} \quad (21)$$

$$\sigma = 402 \text{ ps}$$

Da die CPU mit 8 MHz läuft, lässt sich daraus schliessen, dass ein Pin-Toggle ohne HAL ca. 10 CPU-Cycles benötigt. Dies erscheint plausibel.

Histogramm und Boxplot sind in Abbildung 30 bzw. 31 dargestellt.

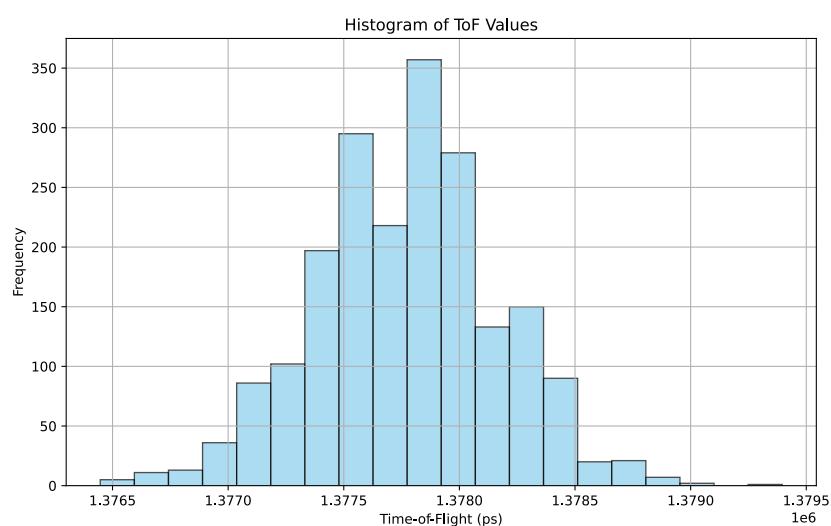


Abbildung 30: GPIO Toggle ohne HAL - Histogramm

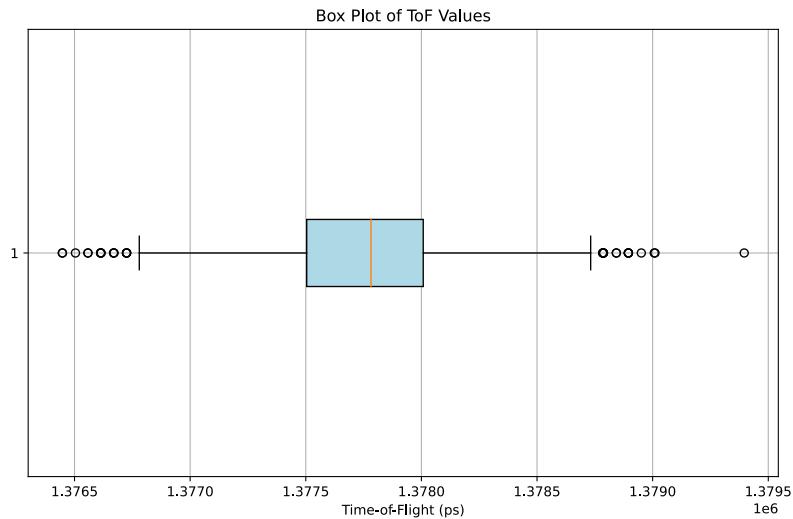


Abbildung 31: GPIO Toggle ohne HAL - Boxplot

Um die Resultate des TDC zu validieren, wurde dieselbe Messung auch mittels Digital Storage Oscilloscope (DSO) durchgeführt. Die Messungen sind in Abbildung 32 und 33 dargestellt.

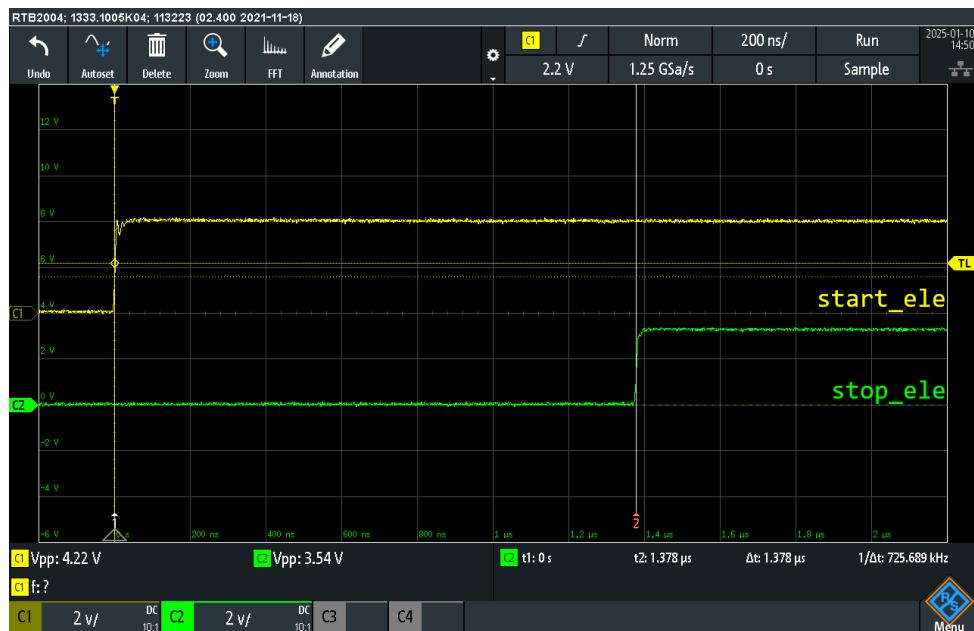


Abbildung 32: GPIO Toggle ohne HAL - DSO



Abbildung 33: GPIO Toggle ohne HAL - DSO (Zoom)

### 6.1.3 Unterschiedliche Kabellängen

Für diese Messung wird dasselbe Setup wie in Kapitel 6.1.2 verwendet.

Anstelle eines Kurzschlusses von J3 (siehe Abbildung 14) werden nun verschiedene Kabellängen angeschlossen.

Als Kabel wird ein einzelner, isolierter Kupferdraht ( $\varnothing_{cu} = 0.8 \text{ mm}$ ,  $\varnothing_{ges} = 1.6 \text{ mm}$ ) verwendet.

Es hat sich herausgestellt, dass eine kreisförmige Anordnung des Kabels wichtig ist. Denn bei einer Überlappung der beiden Kabelenden werden kurze Zeiten gemessen. Dies hat mit der kapazitiven Kopplung zwischen den Leitern zu tun.

Die Resultate sind in Abbildung 34 dargestellt. Die Liste mit den Datenpunkten befindet sich im elektronischen Anhang.

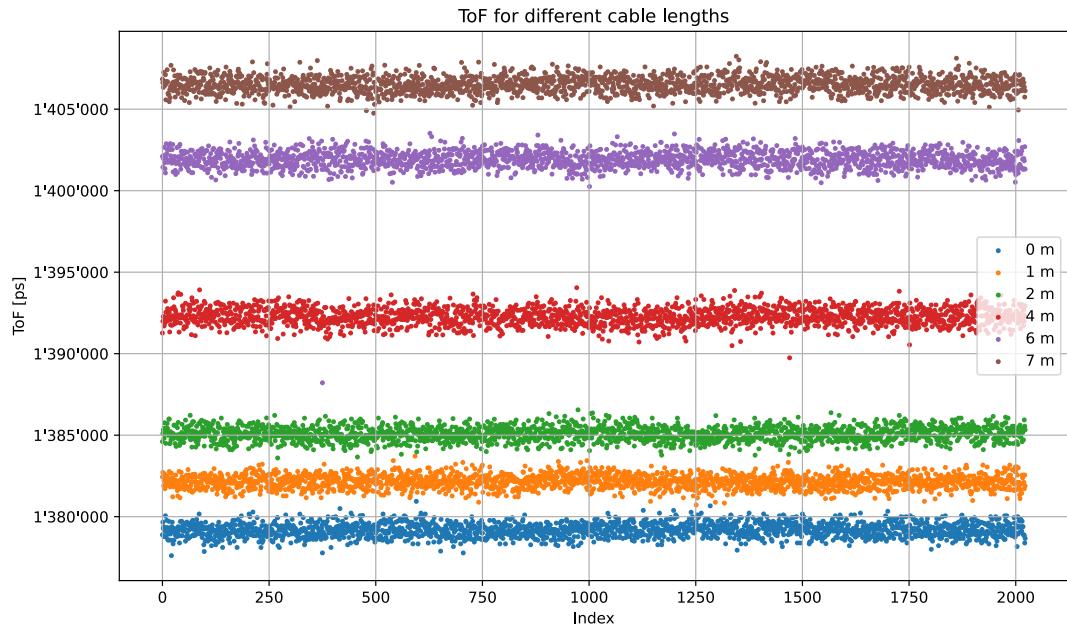


Abbildung 34: Unterschiedliche Kabellängen

Die arithmetischen Mittelwerte und Standardabweichungen sind in Tabelle 1 aufgeführt.

Länge	Mittelwert	Standardabweichung
0 m	1'379'188 ps	421 ps
1 m	1'382'152 ps	413 ps
2 m	1'385'074 ps	437 ps
4 m	1'392'275 ps	519 ps
6 m	1'401'903 ps	577 ps
7 m	1'406'512 ps	491 ps

Tabelle 1: Unterschiedliche Kabellängen

Die Signal-Ausbreitungs-Geschwindigkeit in Kupfer beträgt ca.  $2/3$  der Lichtgeschwindigkeit (firewall.xc, 2025). Um die Resultate in Tabelle 1 zu validieren, rechnen wir, wie in Formel 22 gezeigt, auf die Kabellänge zurück. Die Laufzeit bei 0 m wird dabei abgezogen, um die Verzögerung zu kompensieren, welche durch das Schalten der GPIOs entsteht.

$$c_{cu} \approx \frac{2}{3} \cdot c_0 = \frac{2}{3} \cdot 299'792'458 \frac{m}{s} \approx 200'000'000 \frac{m}{s}$$

$$ToF_n = ToF_{n_{abs}} - ToF_0$$

$$l_n = ToF_n \cdot c_{cu}$$
(22)

Die Resultate sind in Tabelle 2 dargestellt.

Tatsächliche Länge	ToF <sub>n</sub>	Zurückgerechnete Länge
0 m	0 ps	0.6 m
1 m	2'964 ps	0.6 m
2 m	5'886 ps	1.2 m
4 m	13'087 ps	2.6 m
6 m	22'715 ps	4.5 m
7 m	27'324 ps	5.5 m

Tabelle 2: Kabellängen zurückgerechnet

Es fällt auf, dass die Resultate nicht genau übereinstimmen. Es ist jedoch eine klare Korrelation zu erkennen.

Der Unterschied könnte verschiedene Ursachen haben. Eine mögliche Erklärung wäre, dass die Ausbreitungs-Geschwindigkeit beim verwendeten Kabel näher an der Lichtgeschwindigkeit ist als angenommen. Weitere Messresultate und Auswertungen dazu in Kapitel 6.1.7.

#### 6.1.4 Mode 1 vs. Mode 2

Der TDC7200 unterstützt zwei Modi mit unterschiedlichen Messbereichen (TI, 2016b):

- Mode 1: 12 ns bis 500 ns
- Mode 2: 250 ns bis 8 ms

Im Mode 1 wird nur der interne Ring-Oszillator des TDC verwendet. Siehe dazu Abbildung 35.

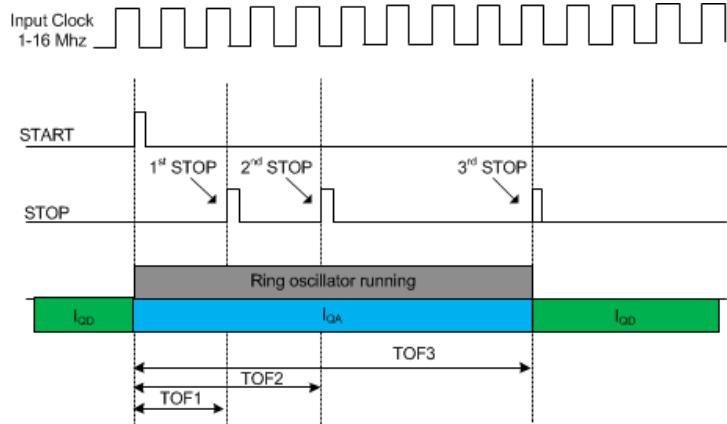


Abbildung 35: TDC Mode 1 (TI, 2016b)

Im Mode 2, um längere Zeiten messen zu können, wird zusätzlich der externe Clock des TDC verwendet. Siehe dazu Abbildung 36.

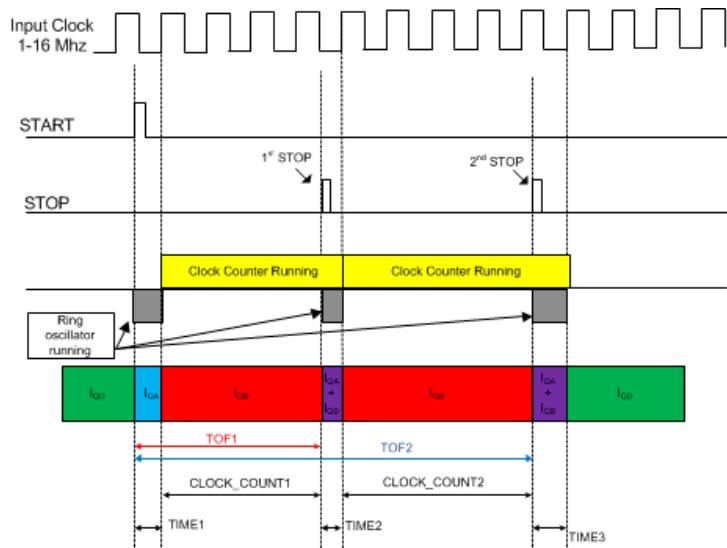


Abbildung 36: TDC Mode 2 (TI, 2016b)

Die bisherigen Messungen (Kapitel 6.1.1, 6.1.2 und 6.1.3) wurden im Mode 2 durchgeführt, da das Schalten der GPIOs mit der CPU mehr als 500 ns brauchte.

In künftigen Messungen soll das Schalten der GPIOs von einem Hardware-Timer der MCU erledigt werden. Damit werden Schaltzeiten von 125 ns bei 8 MHz bzw. 20.8 ns bei 48 MHz möglich sein. Es soll deshalb in diesem Kapitel ein Vergleich der Messresultate der beiden Modi gemacht werden.

Dazu wurden drei Messungen gemacht:

1. GPIO Toggle ohne HAL im Mode 2 mit Kabellänge = 0 m (wie in Kapitel 6.1.2 und 6.1.3)
2. GPIO Toggle ohne HAL im Mode 2 mit Kabellänge = 6 m (wie in Kapitel 6.1.3)
3. Ohne GPIO Toggle Im Mode 1 mit Kabellänge = 6 m

Für die Messung im Mode 1 soll auf die Verzögerung durch das Schalten der GPIOs verzichtet werden. Dazu wird das **START**- und **STOP**-Signal vom selben GPIO-Pin, **start\_ele**, generiert. Dazu wird **SW1** mit **start\_ele** verbunden (siehe Abbildung 14).

In Code 8 ist die Firmware-Implementation für eine Messung im Mode 1 gezeigt.

```

1 // Configure TDC
2
3 TDC_init(&tdc_ele, &hspi1, SPI_CS_ELE_GPIO_Port, SPI_CS_ELE_Pin,
           TDC_ELE_ENABLE_GPIO_Port, TDC_ELE_ENABLE_Pin);
4 TDC_enable(&tdc_ele);
5
6 TDC_read(&tdc_ele, TDC_ADR_CONFIG1, data);
7 data[0] |= TDC_CONFIG1_MEAS_MODE_1;
8
9 TDC_write(&tdc_ele, TDC_ADR_CONFIG1, data);
10
11 while (1) {
12     TDC_start(&tdc_ele);
13
14     // Set Pin to High
15     *((volatile uint32_t*)(GPIOA_BASE + 0x14)) |= (1 << 8); // START_ELE

```

```

16
17     TDC_read_result(&tdc_ele, &tof_fs);
18
19     sprintf(string, "ToF = %lu [ps]\n", tof_fs / 1000);
20     HAL_UART_Transmit(&huart2, (uint8_t *)string, strlen(string), 10000);
21
22     // Set Pin to Low (preparation for next iteration)
23     *((volatile uint32_t*)(GPIOA_BASE + 0x14)) &= ~(1 << 8); // START_ELE
24
25     HAL_Delay(10); // 10 ms
26 }
```

Code 8: Mode 1

Die Unterschiede im Vergleich zu den Messungen in Kapitel 6.1.2 und 6.1.3 sind:

- Zeile 7: TDC wird im Mode 1 konfiguriert (anstatt Mode 2)
- Zeile 15 und 23: Es wird nur der `start_ele` Pin getoggelt (anstatt `start_ele` und `stop_ele` Pin)

Die Erwartung ist, dass die Differenz aus Messung 1 und Messungen 2 ungefähr dem Resultat aus Messung 3 entspricht.

Die Resultate dieser drei Messungen sind in Tabelle 3 aufgeführt. Die restlichen Daten befinden sich im elektronischen Anhang.

Messung	Mittelwert	Standardabweichung
1	1'378'222 ps	407 ps
2	1'403'224 ps	541 ps
3	25'145 ps	331 ps

Tabelle 3: Mode 1 vs. Mode 2

Die Berechnung in Formel 23 zeigt, dass die Messresultate nahe beieinander liegen.

$$\begin{aligned} \Delta \text{Mode 2} &= 1'403'224 \text{ ps} - 1'378'222 \text{ ps} = 25'001 \text{ ps} \\ \text{Mode 1} &= 25'145 \text{ ps} \end{aligned} \quad (23)$$

Es kann also davon ausgegangen werden, dass Mode 1 und Mode 2 im Firmware-Treiber korrekt implementiert wurden.

Bemerkung: Im Firmware-Treiber (siehe Anhang 8.2) kann für beide Modi dieselbe Funktion `TDC_read_result()` verwendet werden. Für Mode 1 vereinfacht sich die Berechnung, weil `TIME2 = 0` und `CLOCK_COUNT1 = 0`.

### 6.1.5 Timer Output

In Kapitel 6.1.1 und 6.1.2 wurde gezeigt, dass es einige CPU-Cycles dauert, um mittels Software, also mit CPU-Instruktionen, GPIOs zu toggeln.

Eine schnellere Methode ist es, das Toggeln der GPIOs von der Hardware-Peripherie der MCU erledigen zu lassen. Dazu eignen sich Hardware-Timer.

In Abbildung 37 ist die default Clock-Configuration des Nucleo-Boards gezeigt.

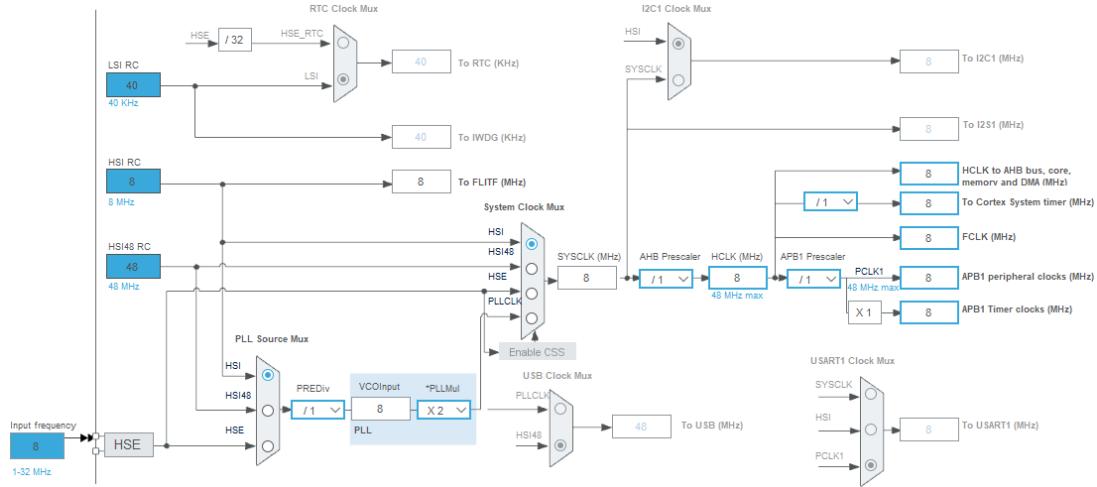


Abbildung 37: Default Clock-Configuration

Per default wird der High Speed Internal (HSI) Clock mit 8 MHz verwendet. Damit sollte es mit Hardware-Timer möglich sein, zwei GPIOs zu schalten mit einer Verzögerung gemäss Formel 24.

$$t = \frac{1}{8 \text{ MHz}} = 125 \text{ ns} \quad (24)$$

Dazu werden Kanal `CH1` und `CH4` des Hardware-Timers `TIM1` im Output Compare Match Modus verwendet. Die zugeordneten GPIO-Pins sind `start_ele` bzw. `stop_ele`.

Erreicht der 16 bit Timer den Wert 32'000 wird `start_ele` auf High geschalten. Beim Erreichen des Werts 32'001 wird `stop_ele` auf High geschalten.

Der TDC wird in Mode 1 konfiguriert. Der STOP-Pin des TDC wird via `sw1` mit `stop_ele` verbunden. Der Kabelanschluss `J3` wird kurzgeschlossen. Siehe Abbildung 14.

In Code 9 ist die Firmware-Implementation dazu gezeigt.

```

1 // Configure TDC
2
3 TDC_init(&tdc_ele, &hspi1, SPI_CS_ELE_GPIO_Port, SPI_CS_ELE_Pin,
           TDC_ELE_ENABLE_GPIO_Port, TDC_ELE_ENABLE_Pin);
4 TDC_enable(&tdc_ele);
5
6 TDC_read(&tdc_ele, TDC_ADR_CONFIG1, data);
7 data[0] |= TDC_CONFIG1_MEAS_MODE_1;
8
9 TDC_write(&tdc_ele, TDC_ADR_CONFIG1, data);
10
11 while (1) {
12     TDC_start(&tdc_ele);
13
14     // Start Timer to toggle Pins from Low to High
15     MX_TIM1_Init();
16     HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_1); // START_ELE
17     HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_4); // STOP_ELE
18 }
```

```

19     HAL_Delay(10); // 10 ms
20
21     TDC_read_result(&tdc_ele, &tof_fs);
22
23     sprintf(string, "ToF = %lu [ps]\n", (uint32_t)(tof_fs / 1000));
24     HAL_UART_Transmit(&huart2, (uint8_t*)string, strlen(string), 10000);
25
26     // Start Timer one more time to toggle Pins from High to Low (preparation
27     // for next iteration)
28     MX_TIM1_Init();
29     HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_1); // START_ELE
30     HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_4); // STOP_ELE
31
32     HAL_Delay(10); // 10 ms
33 }
```

Code 9: GPIO Toggle mit Timer-Output

Der gemessene arithmetische Mittelwert und die Standardabweichung sind in Formel 25 aufgeführt. Die Liste mit den Datenpunkten befindet sich im elektronischen Anhang.

$$\overline{ToF} = 128'108 \text{ ps} \quad (25)$$

$$\sigma = 81 \text{ ps}$$

Erwartungsgemäss dauert das Schalten der GPIOs ca. 125 ns. Die Standardabweichung ist aufgrund der kleineren Anzahl an Clock-Cycles tiefer als in den bisherigen Kapiteln.

### 6.1.6 Unterschiedliche Clock-Konfigurationen

In diesem Teilkapitel werden Messungen mit demselben Setup wie in Kapitel 6.1.5 durchgeführt.

Es werden die arithmetischen Mittelwerte und die Standardabweichungen verschiedener Clock-Quellen des Nucleo-Boards verglichen. Es werden drei Messungen gemacht:

1. HSI: 8 MHz (wie in Kapitel 6.1.5)
2. HSI mit PLL: 16 MHz
3. HSI48: 48 MHz

Die Clock-Konfigurationen für Messung 2 und 3 sind in Abbildung 38 bzw. 39 dargestellt.

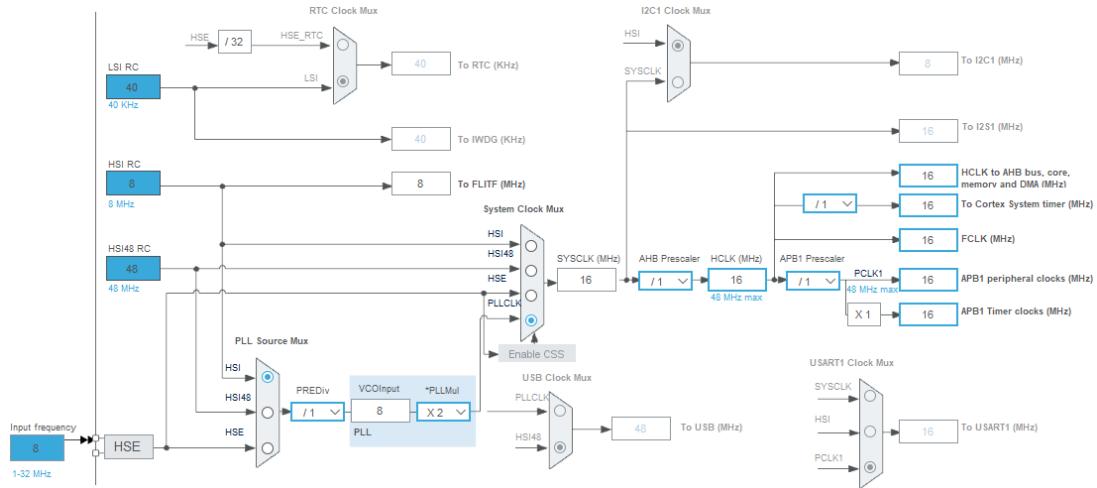


Abbildung 38: Clock-Configuration HSI mit PLL 16 MHz

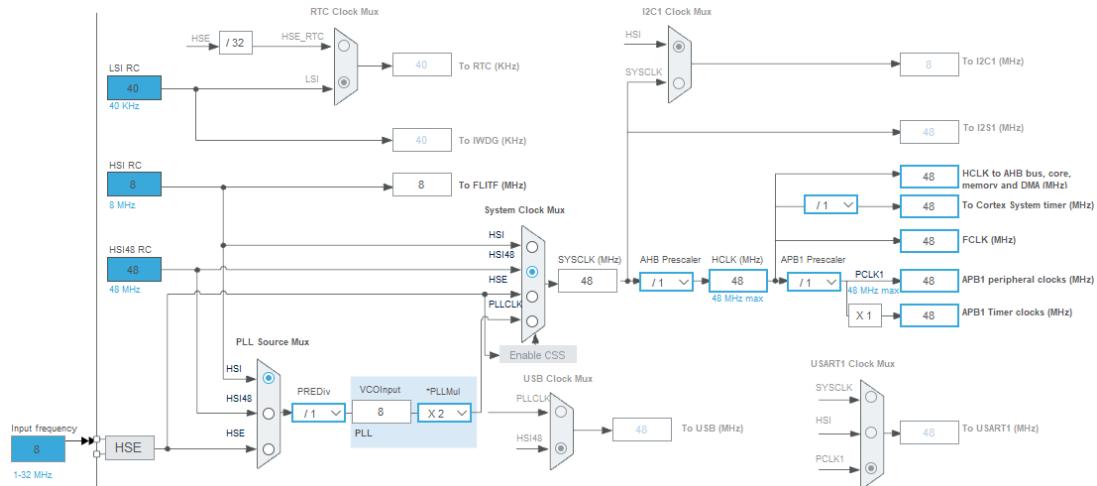


Abbildung 39: Clock-Configuration HSI48 48 MHz

Die arithmetischen Mittelwerte und Standardabweichungen sind in Tabelle 4 dargestellt. Die Liste mit den Datenpunkten befindet sich im elektronischen Anhang.

Messung	Mittelwert	Standardabweichung
1	128'108 ps	81 ps
2	65'694 ps	83 ps
3	24'275 ps	42 ps

Tabelle 4: Unterschiedliche Clock-Quellen

Die Resultate stimmen ungefähr mit den erwarteten Werten aus Formel 26 überein.

$$\begin{aligned} t_1 &= \frac{1}{8 \text{ MHz}} = 125 \text{ ns} \\ t_2 &= \frac{1}{16 \text{ MHz}} = 62.5 \text{ ns} \\ t_3 &= \frac{1}{48 \text{ MHz}} = 20.8 \text{ ns} \end{aligned} \quad (26)$$

Im Datenblatt konnte eine Angabe zum PLL-Jitter gefunden werden, siehe Abbildung 40.

**Table 42. PLL characteristics**

<b>Symbol</b>	<b>Parameter</b>	<b>Value</b>			<b>Unit</b>
		<b>Min</b>	<b>Typ</b>	<b>Max</b>	
$f_{\text{PLL\_IN}}$	PLL input clock <sup>(1)</sup>	1 <sup>(2)</sup>	8.0	24 <sup>(2)</sup>	MHz
	PLL input clock duty cycle	40 <sup>(2)</sup>	-	60 <sup>(2)</sup>	%
$f_{\text{PLL\_OUT}}$	PLL multiplier output clock	16 <sup>(2)</sup>	-	48	MHz
$t_{\text{LOCK}}$	PLL lock time	-	-	200 <sup>(2)</sup>	$\mu\text{s}$
Jitter <sub>PLL</sub>	Cycle-to-cycle jitter	-	-	300 <sup>(2)</sup>	ps

1. Take care to use the appropriate multiplier factors to obtain PLL input clock values compatible with the range defined by  $f_{\text{PLL\_OUT}}$ .
2. Guaranteed by design, not tested in production.

Abbildung 40: PLL-Jitter (ST, 2017)

Diese Angabe ist ungefähr in derselben Größenordnung wie die Messresultate.

### 6.1.7 Unterschiedliche Kabellängen mit DSO

In diesem Unterkapitel werden nochmals dieselben Kabellängen wie in Kapitel 6.1.3 ausgemessen.

Es wird die folgende Firmware-Konfiguration verwendet, um den Jitter zu reduzieren:

- TDC: Modus 1
- Clock: 48 MHz
- Schalten der GPIOs: Timer Output

Zusätzlich werden die verschiedenen Kabellängen gleichzeitig auch via DSO (Rohde & Schwarz RTB2004 2.5 GSa/s.) ausgemessen, um die Resultate des TDC damit zu vergleichen.

In Abbildung 41 sind die erfassten Messresultate des TDC dargestellt. Während dem Erfassen der Messwerte war das DSO an den Pins angeschlossen. Die Liste mit den Datenpunkten befindet sich im elektronischen Anhang.

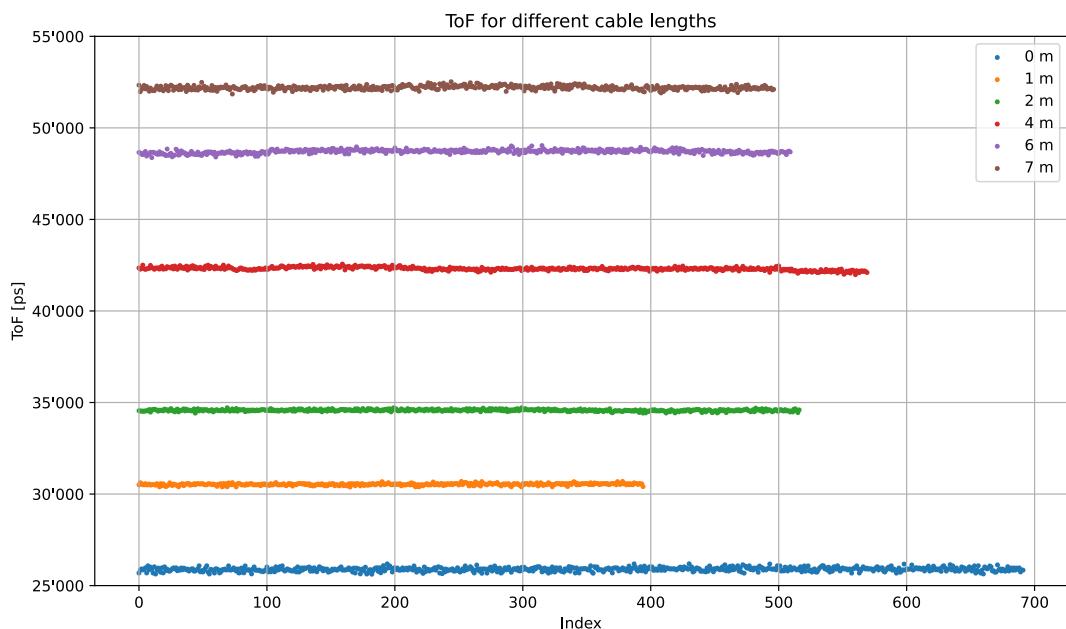


Abbildung 41: Unterschiedliche Kabellängen - Messwerte TDC

In Tabelle 5 sind die Mittelwerte und die Standardabweichungen der erfassten Messresultate des TDC aufgelistet.

Länge	Mittelwert	Standardabweichung	Δ zu 0 m
0 m	25'890 ps	104 ps	0 ns
1 m	30'531 ps	53 ps	4.6 ns
2 m	34'575 ps	50 ps	8.7 ns
4 m	42'308 ps	88 ps	16.4 ns
6 m	48'710 ps	98 ps	22.8 ns
7 m	52'194 ps	104 ps	26.3 ns

Tabelle 5: Unterschiedliche Kabellängen - Resultate TDC

In Tabelle 6 sind die erfassten Messresultate des DSO aufgelistet. Die letzte Spalte zeigt den Unterschied zwischen DSO und TDC. Es ist zu erkennen, dass der Unterschied sehr klein ist.

Länge	Wert	Δ zu 0 m	Δ zu TDC
0 m	1.5 ns	0 ns	0.0 ns
1 m	6.2 ns	4.7 ns	+0.1 ns
2 m	10.2 ns	8.7 ns	0.0 ns
4 m	18.1 ns	16.6 ns	+0.2 ns
6 m	24.7 ns	23.2 ns	+0.4 ns
7 m	28.1 ns	26.6 ns	+0.3 ns

Tabelle 6: Unterschiedliche Kabellängen - Resultate DSO

Tabelle 7 zeigt die theoretisch zu erwartenden Laufzeiten bei den unterschiedlichen Annahmen für die Signalausbreitungs-Geschwindigkeiten von  $\frac{2}{3} c$  und  $c$ .

Länge	Theorie $2/3 c$	Theorie $c$
0 m	0.0 ns	0 ns
1 m	5.0 ns	3.3 ns
2 m	10.0 ns	6.7 ns
4 m	20.0 ns	13.3 ns
6 m	30.0 ns	20.0 ns
7 m	35.0 ns	23.3 ns

Tabelle 7: Unterschiedliche Kabellängen - Theoretische Erwartung

Die Unterschiede der TDC-Resultate zu den theoretischen Erwartungen sind in den Tabellen 8 und 9 dargestellt. Zuerst im Bezug auf den Laufzeit-Unterschied, dann die zurückgerechnete Kabellänge.

Länge	$\Delta$ TDC zu Theorie $2/3 c$	$\Delta$ TDC Theorie zu $c$
0 m	0.0 ns	0.0 ns
1 m	-0.4 ns	+1.3 ns
2 m	-1.3 ns	+2.0 ns
4 m	-3.6 ns	+3.1 ns
6 m	-7.2 ns	+2.8 ns
7 m	-8.7 ns	+3.0 ns

Tabelle 8: Unterschiedliche Kabellängen - Unterschied TDC [ns] zu Theorie

Länge	TDC gem. Theorie $2/3 c$	TDC gem. Theorie $c$
0 m	0.0 m	0.0 m
1 m	0.9 m	1.4 m
2 m	1.7 m	2.6 m
4 m	3.3 m	4.9 m
6 m	4.6 m	6.8 m
7 m	5.3 m	7.9 m

Tabelle 9: Unterschiedliche Kabellängen - TDC [m] gem. Theorie

In Tabelle 9 ist zu erkennen, dass keine der beiden theoretischen Werte für  $c$  zu einem perfekten TDC-Resultat für die zurückgerechnete Distanz [m] führt.

In Tabelle 8 ist zu erkennen, dass die Differenz der TDC-Resultate zur theoretisch zu erwartenden Laufzeit bei  $c$  etwas kleiner ist als bei  $2/3 c$ . Es scheint, dass  $c$  eine etwas bessere Näherung darstellt.

Falls sich der gemessene Unterschied tatsächlich durch eine falsche Annahme der Ausbreitungs-Geschwindigkeit erklären lässt, dann müsste die Ausbreitungs-Geschwindigkeit im gewählten Kabel bei ca. 0.9  $c$  liegen. Die zurückrechneten Resultate dazu sind in Tabelle 10 aufgelistet.

Länge	TDC gem. Theorie 0.9 c
0 m	0.0 m
1 m	1.2 m
2 m	2.3 m
4 m	4.4 m
6 m	6.2 m
7 m	7.1 m

Tabelle 10: Unterschiedliche Kabellängen - TDC [m] gem. Theorie 0.9 c

In den Abbildungen 42 und 43 sind die DSO-Aufzeichnungen bei den Kabellängen 1 m bzw. 6 m gezeigt. Die restlichen DSO-Aufzeichnungen befinden sich im elektronischen Anhang.



Abbildung 42: DSO-Aufzeichnung bei Kabellänge 1 m



Abbildung 43: DSO-Aufzeichnung bei Kabellänge 6 m

Es ist zu erkennen, dass die Flanke des **STOP**-Signals bei grösserer Kabellänge etwas abflacht. Dies hat einen Einfluss auf die Laufzeit-Messung. Steilere Flanken würden helfen diesen Einfluss zu minimieren. Mehr dazu in Kapitel 7.1.

## 6.2 Optische Messungen

In diesem Teilkapitel werden die Messresultate dokumentiert, welche sich auf den optischen Pfad des Demonstrators beziehen.

Das verwendete DSO ist ein Rohde & Schwarz RTB2004 2.5 GSa/s.

Für die Messungen in diesem Kapitel werden die folgenden Firmware-Konfigurationen verwendet:

- TDC: Modus 1
- Clock: 48 MHz
- Schalten der GPIOs: Timer Output

### 6.2.1 Signalpfad Sender

In diesem Unterkapitel wird der Signalpfad des optischen Sendeteils ausgemessen.

Die Verzögerungszeiten zwischen den MCU-Signalen `start_op` und `LD_pulse` sowie dem Potential am Gate des NexFET (v1 Pin 4 in Abbildung 18) sind in Abbildung 44 dargestellt.



Abbildung 44: Messung Signalpfad Sender: `start_op`, `LD_pulse`, FET-Gate

Die gemessene Verzögerungszeit von 20.4 ns (20.8 ns entsprechen einem Clock-Cycle bei 48 MHz) zwischen `start_op` und `LD_pulse` wurde bewusst eingefügt, um sicherzustellen, dass die minimale Messdauer von 12 ns (TI, 2016b) im Mode 1 garantiert immer eingehalten wird.

Die gemessene Verzögerungszeit von 8.4 ns zwischen **LD\_pulse** und dem FET-Gate (dies entspricht dem Ausgang des Gate-Treibers **IC22**) ist klein und zeigt eine steile Flanke.

In Abbildung 45 ist der Strom durch die Laser-Diode (**LD1** in Abbildung 18) aufgezeigt.



Abbildung 45: Messung Signalpfad Sender: Strom durch Laser-Diode

Der Strom wurde gemäss Formel 27 berechnet.

$$I_{LD} = \frac{V(TP8) - V(TP9)}{R_V} = \frac{V(TP8) - V(TP9)}{5 \Omega} \quad (27)$$

Damit die Verzögerungszeit ersichtlich ist, wurde zusätzlich, wie in Abbildung 44, das Signal **LD\_pulse** und das Potential am FET-Gate aufgezeichnet.

Der gemessene Strom während dem Senden beträgt ca. 174 mA.

Dies ist um einiges weniger als der erwartete Strom von 440 mA aus Formel 12. Dies hat mit der erhöhten Dioden-Spannung zu tun. Während dem Senden erhöht sich die Spannung über der Laser-Diode auf ca. 4 V, die Spannung über Drain-Source des FET ist vernachlässigbar klein. Damit ergibt sich der Strom durch die Diode gemäss Formel 28.

$$I_{LD} = \frac{5 V - V_{LD} - V_{DS}}{R_V} \approx \frac{5 V - 4 V}{5 \Omega} \approx 200 mA \quad (28)$$

### 6.2.2 ToF-Messungen via Spiegel

Damit die Photo-Diode ein möglichst starkes Signal erhält, wurde anstelle einer Wand für diese Messungen ein Spiegel verwendet. Das Setup ist in Abbildung 46 dargestellt.

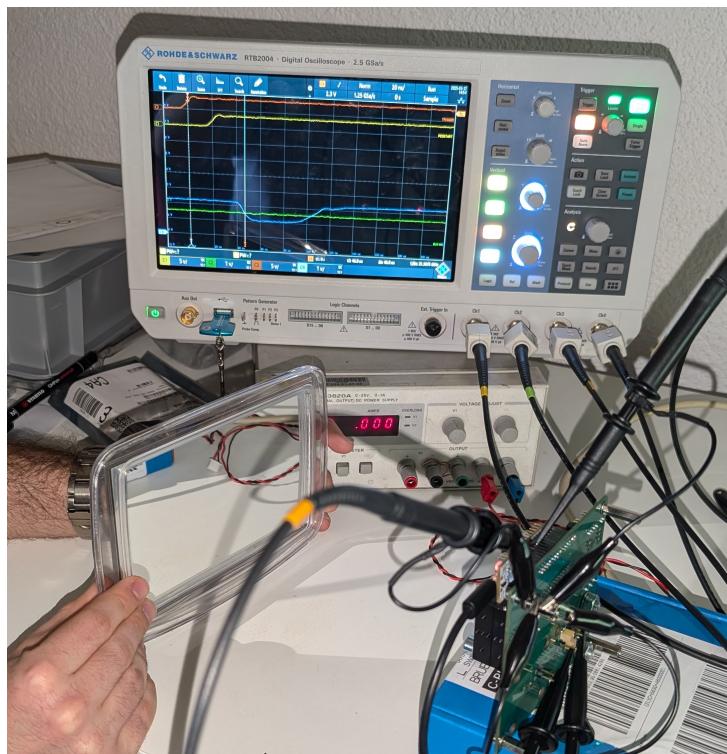


Abbildung 46: Mess-Setup mit Spiegel

Gemäss Datenblatt hat die Laser-Diode eine nicht ideal punktförmige Abstrahl-Charakteristik, siehe Abbildung 47.

Beam divergence	$\theta_{\parallel}$	Po=7mW	7	9	13	deg.
	$\theta_{\perp}$		20	27	35	deg.

Abbildung 47: Laser Abstrahl-Charakteristik (ROHM, 2019)

Es ist aufgefallen, dass mit dem gewählten PCB-Layout die Orientierung der Laser-Diode für eine Spiegel- Messung nicht ganz ideal ist. Der breitere Abstrahlwinkel ist senkrecht zur Linie zwischen Laser- und Photodiode. Siehe dazu Abbildung 48.



Abbildung 48: Laser-Ausrichtung

Die Laser-Diode wurde deshalb manuell um 90° gedreht, wie in Abbildung 49 dargestellt.



Abbildung 49: Laser-Ausrichtung angepasst

Das DSO-Diagramm einer initialen Spiegel-Messung mit ca. 20 cm Distanz zum Spiegel ist in Abbildung 50 dargestellt. Es zeigt den Ausgang des Transimpedanzverstärkers. Zusätzlich, um die Verzögerungszeit zu sehen, wurden auch die MCU-Signale `start_op` und `LD_pulse` sowie das Spannungspotential am Gate des FET aufgezeichnet.



Abbildung 50: Initiale Spiegelmessung mit DSO

Mit dem Demonstrator wurden Messwerte für drei verschiedene Distanzen aufgezeichnet: 19 cm, 24 cm und 28.5 cm.

Die mit dem optischen TDC erfassten Datenpunkte sind in Abbildung 51 dargestellt. Es wurden je ca. 300 Messwerte erfasst, die Liste mit den Datenpunkten befindet sich im elektronischen Anhang.

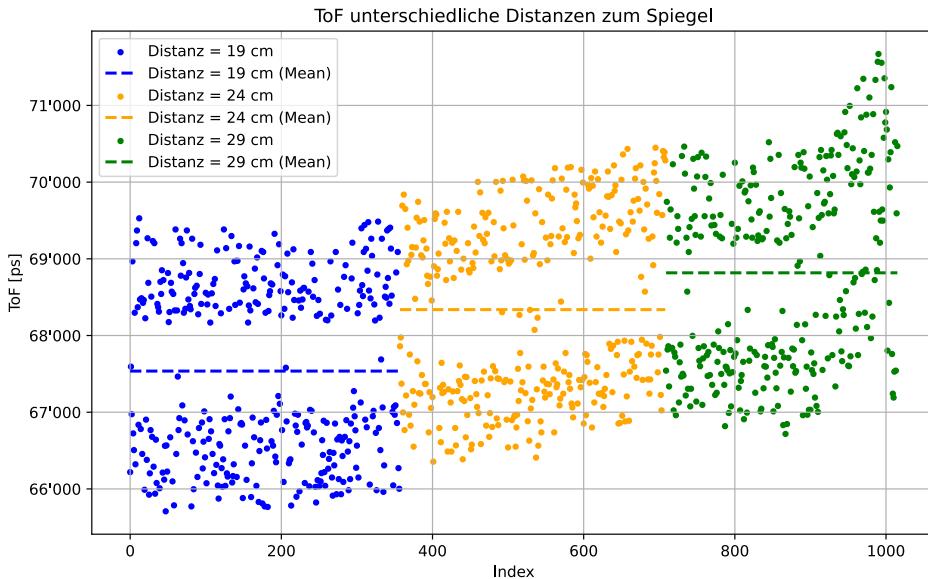


Abbildung 51: Unterschiedliche Distanzen zum Spiegel - Plot

Die arithmetischen Mittelwerte und die Standardabweichungen sind in Tabelle 11 aufgelistet.

Distanz zum Spiegel	Mittelwert	Standardabweichung
19 cm	67'536 ps	1'172 ps
24 cm	68'338 ps	1'191 ps
29 cm	68'817 ps	1'273 ps

Tabelle 11: Unterschiedliche Distanzen zum Spiegel - Statistische Grössen

Es fällt auf, dass die Standardabweichungen relativ gross sind, dies hat insbesondere mit den beiden Clustern zu tun, welche in Abbildung 51 zu sehen sind. Die Ursache dieser Cluster ist unklar. Eine Hypothese ist, dass diese mit dem DSO zusammenhängen. Es wurde festgestellt, dass das Anschliessen des DSO einen signifikanten Einfluss auf den Offset der TDC-Laufzeitwerte hat (ca. 10 ns). Eventuell hat das DSO durch Sample und Hold einen zeitlich-abhängigen Einfluss auf die Messresultate – je nach dem ob der Lichtpuls gerade in einer Sample- oder Hold-Phase auf die Photodiode trifft. Die Ursache dieser Cluster sollten in einer allfälligen weiterführenden Arbeit genauer analysiert werden.

Eine Spiegeldistanz-Änderung von 5 cm entspricht einer Längen-Änderung von 10 cm.

Die Messungen zeigen für diese Längen-Änderung von 10 cm eine Änderung der Laufzeit von durchschnittlich ca. 0.6 ns.

In Formel 29 wird der theoretisch zu erwartende Laufzeit-Unterschied berechnet.

$$\Delta ToF = \frac{\Delta L}{c} = \frac{10 \text{ cm}}{3 \cdot 10^8 \text{ m/s}} = 0.33 \text{ ns} \quad (29)$$

Dieser Unterschied (ca. Faktor 2) könnte mehrere Gründe haben:

- 1.) Eine mögliche Erklärung ist, dass die Flanke bei grösserer Distanz weniger steil wird und dadurch fälschlicherweise eine etwas längere Laufzeit gemessen wird.
- 2.) Etwas anderes, das in Abbildung 51 auffällt, ist ein Time-Drift während dem Aufzeichnen der Messdaten bei 24 cm und 29 cm. Abbildung 52 soll dies verdeutlichen.

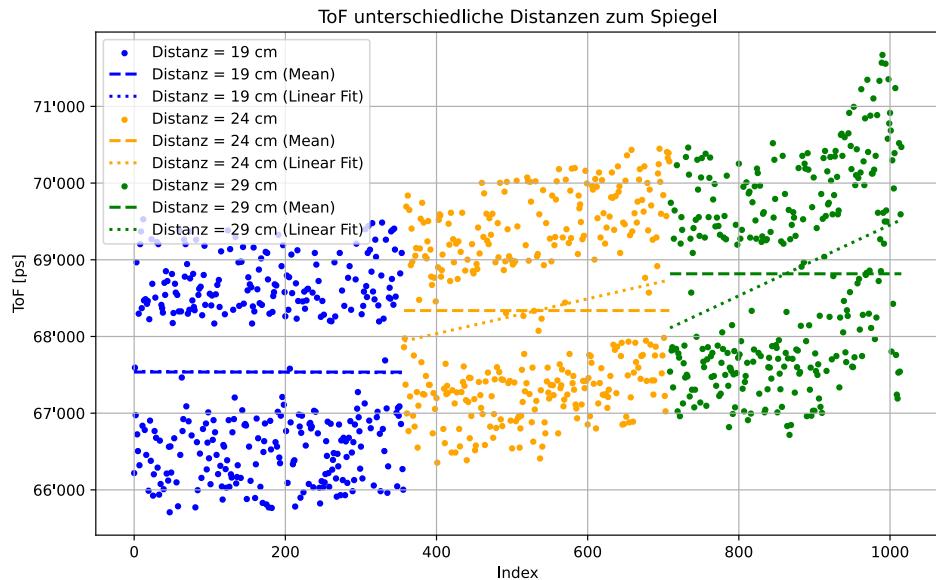


Abbildung 52: Unterschiedliche Distanzen zum Spiegel - Plot mit linearer Interpolation

Dieser Time-Drift kann aber den gemessenen Unterschied nicht schlüssig erklären.

Bemerkung: Der Spiegel musste sehr genau ausgerichtet werden und dann sehr stabil gehalten werden, um Resultate mit relativ kleiner Standardabweichung (ca. 1 ns) messen zu können.

Bei Distanzen von mehr als 30 cm wird das TDC-Signal unbrauchbar.

Dies hat damit zutun, dass die Photodiode ab dann nur noch wenig Licht empfängt und entsprechend einen kleinen Lichtstrom generiert. Dies führt zu einer kleineren Aussteuerung am Ausgang des Transimpedanzverstärkers. Somit wird die Schaltschwelle des Komparators nicht mehr oder nur noch ganz knapp erreicht.

Abbildungen 53 und 54 sollen dies verdeutlichen.



Abbildung 53: TIA-Ausgang und Komparator-Schaltschwelle bei 12 cm



Abbildung 54: TIA-Ausgang und Komparator-Schaltschwelle bei 30 cm

Die Abbildungen zeigen, neben den MCU-Signalen `start_op` und `LD_pulse`, den Ausgang des Transimpedanzverstärkers und die Komparator-Schaltschwelle.

Mögliche Verbesserungen dazu sind im Kapitel 7.1 aufgeführt.

### 6.2.3 ToF-Messungen via Wand

Anstelle des in Kapitel 6.2.2 verwendeten Spiegels, werden in diesem Kapitel Messresultate mit einer weißen Wand als Target aufgezeichnet. Es wurden Distanzen von ca. 10 bis 30 cm ausprobiert.

Es konnten mit diesem Setup in der kurzen zur Verfügung stehenden Zeit keine brauchbaren Messresultate erfasst werden.

Mögliche Verbesserungen dazu sind im Kapitel 7.1 aufgeführt.

## 7 Fazit

In diesem Kapitel werden abschliessende Gedanken gesammelt. Es soll dazu dienen, einen Ausblick für mögliche, weiterführende Arbeiten zu geben. Es sollen auch einige Punkte genannt werden, die gut gelaufen sind, aber auch jene bei denen Verbesserungspotential identifiziert wurde.

### 7.1 Ausblick

Grundsätzlich kann bestätigt werden, dass sich ein ToF-System, so wie es sich die Studierenden vorstellen, entwickeln und betreiben lässt. Nachfolgend werden ein paar mögliche Verbesserungen und weiterführende Messungen aufgelistet.

1. Die Optik besser aufeinander abstimmen. Dies würde es zum einen erlauben, über weitere Distanzen hinweg Messungen machen zu können. Des weiteren sollte es ebenfalls möglich sein, auch Targets mit nicht oder schwach reflektierenden Oberflächen-Beschaffenheiten zu detektieren.
2. Für die Messungen mit dem Spiegel würde es Sinn machen, es mit einer Linse mit grösserer Brennweite (oder mit derselben Linse mit etwas weniger Distanz zum PCB) auszuprobiieren. Dies hätte den Vorteil, dass der Spiegel nicht ganz so genau ausgerichtet werden muss. Der Nachteil wäre etwas weniger Photostrom bei parallel einfallendem Licht.
3. Bei der Ansteuerung der TDCs gibt es Verbesserungspotential. So könnte beispielsweise mit einer schneller getakteten MCU der Jitter bei **START**- und **STOP**-Signal minimiert werden, was zu Messresultaten mit noch weniger Streuung führen würde.
4. Eine andere Möglichkeit die Streuung zu minimieren, wäre eine externe Clock-Quelle zu verwenden.
5. Eventuell ist es möglich auf die künstlich eingefügte Verzögerungszeit von einem Taktzyklus zu verzichten. Dies wäre der Fall, falls im Messpfad bereits genug Verzögerung entsteht, um die minimale Messdauer von 12 ns zu überschreiten. Falls nicht, könnte die künstliche Verzögerung anstatt von der MCU auch mittels separater Schaltung (mit möglichst tiefem Jitter) umgesetzt werden.
6. Beim verwendeten Nucleo gibt es die Möglichkeit, die Anstiegszeit der GPIO zu konfigurieren. Mit schnellerer Anstiegszeit lassen sich evtl. die Messresultate verbessern.
7. Optischer Empfangspfad: Es sollten Messungen mit unterschiedlichen Verstärkungen des TIA durchgeführt werden, um die Distanz mit Spiegel zu erhöhen und Messungen gegen die Wand oder andere Targets zu ermöglichen. Damit zusammenhängend sollte die Schaltschwelle des Komparators genauer ausgetestet und feinjustiert werden.
8. Optischer Sendepfad: Es lohnt sich mit höherer Sendeleistung zu experimentieren. Aufgrund der 5 V Speisung der Laser-Diode sind wir mit dem  $5 \Omega$  Vorwiderstand schon fast am Limit. In einer nächsten PCB-Version würde es sich lohnen, mit einem Schalter die Möglichkeit zur 12 V Speisung vorzusehen, analog dazu wie dies für die Photodiode bereits umgesetzt ist.
9. Optische Messungen: Es lohnt sich zu analysieren, weshalb die Messresultate in zwei Clustern (ca. 2 ns auseinander liegend) verteilt sind.

10. Für weitere Messungen mit einem Spiegel lohnt es sich einen präzisen, mechanischen Versuchsaufbau zu konstruieren. Mit diesem könnte der Spiegel besser fixiert und eingestellt werden. Von Hand ist es schwierig reproduzierbare Ergebnisse zu erzielen.

Abschliessend kann gesagt werden, dass es durchaus möglich ist, mit einem TDC7200 eine ToF-Messung zu machen, bei welcher die zeitliche Auflösung des Mess-ICs selber die limitierende Grösse ist.

## 7.2 Lessons Learned

Wie die Messresultate und auch der Ausblick bereits vermuten lassen, hätte primär beim Auslegen der Optoelektronik etwas mehr Zeit investiert werden können. Die initialen, überschlagsmässigen Schätzungen und Rechnungen bezüglich der gewünschten Limitierungen des Systems hätten kritischer hinterfragt werden sollen. Initial war das Ziel der Studenten, Messungen auf Distanzen von bis zu 10 m mit einer Auflösung von etwa 10 cm zu machen. Herausgestellt hat sich am Ende jedoch, dass nicht die zeitliche Auflösung, sondern vielmehr die Distanz, resp. die Stärke des Messsignals problematisch ist.

Mit der Fabrikation des ToF-Demonstrators ist das Studenten-Team zufrieden. Beim erneuten Einsatz so kleiner SMD-Komponenten, wie beispielsweise beim TIA lohnt es sich jedoch darüber nachzudenken, ob nicht eine maschinelle Bestückung der händischen Bestückung vorzuziehen ist. Durch die manuelle Bestückung wurde vermutlich kaum Zeit eingespart, da diese auch einige Stunden, über mehrere Abende verteilt, in Anspruch nahm.

## 7.3 Danksagung

Wir möchten uns an dieser Stelle herzlich bei den beiden Dozenten Prof. Guido Keel und Michael Lehmann bedanken. Während der Laufzeit des Projekts sind sie uns jederzeit mit Rat und Tat zur Seite gestanden und haben das Projekt mit grossem Interesse verfolgt. Auch für das optische Leihmaterial bedanken wir uns. Dieses hat uns ermöglicht, bei der optischen Messung einen Teilerfolg erzielen zu können.

Weiter bedanken wir uns bei Robin Burkard, der für uns ein Layout-Review durchgeführt hat. Dank seinen wertvollen Tipps konnte das PCB so störfrei wie möglich in Betrieb genommen werden.

# 8 Anhang

## 8.1 Outline

### Projektarbeit: Distanzmessung mit TDC

#### Outline

Modul: CAS Sensorik und Sensor Signal Conditioning  
Institution: OST – Ostschweizer Fachhochschule  
Autoren: Matthias Schär, Timon Burkard

Wir möchten in einer ersten Phase den TDC7200 kennen lernen und austesten. In einer zweiten Phase soll ein optischer Pfad dazu kommen, Ziel ist es eine Distanzmessung mittels Time of Flight (ToF) durchzuführen.

#### Teil 1: Elektrisch

Um den TDC7200 elektrisch kennen zu lernen, sollen die folgenden Punkte getestet werden:

##### 1a: IO ein- und ausschalten

Ersten Pin schalten zum START, dann einen zweiten Pin zum STOP. Mit verschiedenen Varianten; z.B. mit CPU oder Timer – Jitter ausmessen und beste Variante finden.

##### 1b: Elektrische Laufzeitmessung

z.B. mit unterschiedlich langen Leitungen auf PCB und externe Anschlüsse für längere Kabel.

Ziel: Länge eines angeschlossenen Kabels bestimmen. Wie nahe kommen wir an das theoretische Minimum von 15mm?

#### Teil 2: Optisch

Schaltung wird mit optischen Komponenten ergänzt: Laserdiode und Photodiode/-transistor.  
Dazu braucht es Verstärkerschaltungen mit OP.

Wie können wir die Laufzeit des Verstärkersignal bestimmen? – z.B. optischen Teil überbrücken und vom Ende des Sende-Verstärkers direkt zum Anfang des Empfangsverstärkers.

Kalibrierung mit fixer Distanz. Wie gross ist die Standardabweichung?

Was passiert bei unterschiedlichen Oberflächen? – Kann man die Stärke des Messsignals kompensieren? – Evtl. führt z.B. ein stärkeres Messsignal zu einer steilere Flanke bei Signal der Photodiode?

Wie nahe kommen wir nun an das theoretische Minimum von 15mm?

#### PCB

Für Teil 1 möchten wir ein eigenentwickeltes PCB verwenden. Um später nicht ein zweites PCB bestellen zu müssen, sollte bereits dann klar sein, wie das PCB aufgebaut sein muss. um auch den Teil 2 damit erledigen zu können.

Ziel ist es das PCB vor Jahresende zu bestellen, sodass dieses Anfangs Januar geliefert wird.

Dazu möchten wir gerne in KW51 vom Dozenten ein Feedback zum PCB einholen.

Abbildung 55: Outline

## 8.2 TDC Treiber

In Code 10 und 11 ist der selbst entwickelte Firmware-Treiber für den TDC dargestellt.

```

1  /*
2   * TDC.h
3   *
4   * Driver for TI TDC7200 -- Header file
5   *
6   * (C) T. Burkard | M. Schaeer
7   *
8   */
9
10 #ifndef TDC_H_
11 #define TDC_H_
12
13 #include <stdint.h>
14 #include "stm32f0xx_hal.h"
15
16 #define TDC_CLOCK_PERIOD_FS 62500000 // 16 MHz in [fs]
17
18 /**
19  * Convert 24bit buffer received from SPI to integer
20 */
21 #define TDC_24BIT_BUF_TO_INT(buf) ((buf[0] << 16) + (buf[1] << 8) + buf[2])
22
23 /**
24  * @brief Register addresses of the TDC
25  *
26 */
27 typedef enum {
28     TDC_ADR_CONFIG1          = 0x00, // Configuration Register 1
29     TDC_ADR_CONFIG2          = 0x01, // Configuration Register 2
30     TDC_ADR_INT_STATUS       = 0x02, // Interrupt Status Register
31     TDC_ADR_INT_MASK         = 0x03, // Interrupt Mask Register
32     TDC_ADR_COARSE_CNTR_OVF_H = 0x04, // Coarse Counter Overflow Value High
33     TDC_ADR_COARSE_CNTR_OVF_L = 0x05, // Coarse Counter Overflow Value Low
34     TDC_ADR_CLOCK_CNTR_OVF_H = 0x06, // CLOCK Counter Overflow Value High
35     TDC_ADR_CLOCK_CNTR_OVF_L = 0x07, // CLOCK Counter Overflow Value Low
36     TDC_ADR_CLOCK_CNTR_STOP_MASK_H = 0x08, // CLOCK Counter STOP Mask High
37     TDC_ADR_CLOCK_CNTR_STOP_MASK_L = 0x09, // CLOCK Counter STOP Mask Low
38     TDC_ADR_TIME1             = 0x10, // Measured Time 1
39     TDC_ADR_CLOCK_COUNT1     = 0x11, // CLOCK Counter Value
40     TDC_ADR_TIME2             = 0x12, // Measured Time 2
41     TDC_ADR_CLOCK_COUNT2     = 0x13, // CLOCK Counter Value
42     TDC_ADR_TIME3             = 0x14, // Measured Time 3
43     TDC_ADR_CLOCK_COUNT3     = 0x15, // CLOCK Counter Value
44     TDC_ADR_TIME4             = 0x16, // Measured Time 4
45     TDC_ADR_CLOCK_COUNT4     = 0x17, // CLOCK Counter Value
46     TDC_ADR_TIME5             = 0x18, // Measured Time 5
47     TDC_ADR_CLOCK_COUNT5     = 0x19, // CLOCK Counter Value
48     TDC_ADR_TIME6             = 0x1A, // Measured Time 6
49     TDC_ADR_CALIBRATION1     = 0x1B, // Calibration 1, 1 CLOCK Period
50     TDC_ADR_CALIBRATION2     = 0x1C, // Calibration 2, 2/10/20/40 CLOCK
51     Periods
52     TDC_ADR_AMOUNT // Amount of registers
53 } TDC_adr_t;
54
55 /**
56  * @brief Register sizes of the TDC
57 */

```

```

57 * Index = Register address (TDC_adr_t)
58 * Value = Register size (in bytes)
59 *
60 */
61 extern const uint8_t TDC_REG_SIZE[TDC_ADR_AMOUNT];
62
63 /**
64 * @brief Bit description of the TDC CONFIG1 register
65 *
66 */
67 typedef enum {
68     TDC_CONFIG1_FORCE_CAL_OFF    = (0b0 << 7), // Calibration is not performed
69     after interrupted measurement (for example, due to counter overflow or
70     missing STOP signal)
71     TDC_CONFIG1_FORCE_CAL_ON     = (0b1 << 7), // Calibration is always
72     performed at the end (for example, after a counter overflow)
73     TDC_CONFIG1_PARITY_EN_OFF    = (0b0 << 6), // Parity bit for Measurement
74     Result Registers disabled (Parity Bit always 0)
75     TDC_CONFIG1_PARITY_EN_ON     = (0b1 << 6), // Parity bit for Measurement
76     Result Registers enabled (Even Parity)
77     TDC_CONFIG1_TRIGG_EDGE_RISE = (0b0 << 5), // TRIGG is output as a Rising
78     edge signal
79     TDC_CONFIG1_TRIGG_EDGE_FALL = (0b1 << 5), // TRIGG is output as a Falling
80     edge signal
81     TDC_CONFIG1_STOP_EDGE_RISE  = (0b0 << 4), // Measurement is stopped on
82     Rising edge of STOP signal
83     TDC_CONFIG1_STOP_EDGE_FALL  = (0b1 << 4), // Measurement is stopped on
84     Falling edge of STOP signal
85     TDC_CONFIG1_START_EDGE_RISE = (0b0 << 3), // Measurement is started on
86     Rising edge of START signal
87     TDC_CONFIG1_START_EDGE_FALL = (0b1 << 3), // Measurement is started on
88     Falling edge of START signal
89     TDC_CONFIG1_MEAS_MODE_1     = (0b00 << 1), // Measurement Mode 1 (for
90     expected time-of-flight < 500 ns).
91     TDC_CONFIG1_MEAS_MODE_2     = (0b01 << 1), // Measurement Mode 2
92     (recommended)
93     TDC_CONFIG1_START_MEAS      = (0b1 << 0), // Start New Measurement.
94     Writing a 1 will clear all bits in the Interrupt Status Register and Start
95     the measurement (by generating an TRIGG signal) and will reset the content
96     of all Measurement Results registers
81 } TDC_config1_t;
82
83 /**
84 * @brief Bit description of the TDC CONFIG2 register
85 *
86 */
87 typedef enum {
88     TDC_CONFIG2_CALIBRATION2_PERIODS_2   = (0b00 << 6), // Measuring 2 CLOCK
89     periods
90     TDC_CONFIG2_CALIBRATION2_PERIODS_10  = (0b01 << 6), // Measuring 10 CLOCK
91     periods
92     TDC_CONFIG2_CALIBRATION2_PERIODS_20  = (0b10 << 6), // Measuring 20 CLOCK
93     periods
94     TDC_CONFIG2_CALIBRATION2_PERIODS_40  = (0b11 << 6), // Measuring 40 CLOCK
95     periods
96     TDC_CONFIG2_AVG_CYCLES_1            = (0b000 << 3), // 1 Measurement Cycle
97     only (no Multi-Cycle Averaging Mode)
98     TDC_CONFIG2_AVG_CYCLES_2            = (0b001 << 3), // 2 Measurement Cycles
99     TDC_CONFIG2_AVG_CYCLES_4            = (0b010 << 3), // 4 Measurement Cycles
100    TDC_CONFIG2_AVG_CYCLES_8           = (0b011 << 3), // 8 Measurement Cycles
101    TDC_CONFIG2_AVG_CYCLES_16          = (0b100 << 3), // 16 Measurement Cycles

```

```

97     TDC_CONFIG2_AVG_CYCLES_32          = (0b101 << 3), // 32 Measurement Cycles
98     TDC_CONFIG2_AVG_CYCLES_64          = (0b110 << 3), // 64 Measurement Cycles
99     TDC_CONFIG2_AVG_CYCLES_128         = (0b111 << 3), // 128 Measurement
100    Cycles
101    TDC_CONFIG2_NUM_STOP_1           = (0b000 << 0), // Single Stop
102    TDC_CONFIG2_NUM_STOP_2           = (0b001 << 0), // Two Stops
103    TDC_CONFIG2_NUM_STOP_3           = (0b010 << 0), // Three Stops
104    TDC_CONFIG2_NUM_STOP_4           = (0b011 << 0), // Four Stops
105    TDC_CONFIG2_NUM_STOP_5           = (0b100 << 0), // Five Stops
106
107 /**
108 * @brief Bit description of the TDC INT_STATUS register
109 */
110 /**
111 typedef enum {
112     TDC_STATUS_MEAS_COMPLETE_FLAG   = (1 << 4), // Measurement has completed
113     TDC_STATUS_MEAS_STARTED_FLAG    = (1 << 3), // Measurement has started
114     (START signal received)
115     TDC_STATUS_CLOCK_CNTR_OVF_INT  = (1 << 2), // Clock overflow detected,
116     running measurement will be stopped immediately
117     TDC_STATUS_COARSE_CNTR_OVF_INT = (1 << 1), // Coarse overflow detected,
118     running measurement will be stopped immediately
119     TDC_STATUS_NEW_MEAS_INT        = (1 << 0), // Interrupt detected - New
120     Measurement has been completed
121 } TDC_status_t;
122
123 /**
124 * @brief Bit description of the TDC INT_MASK register
125 */
126 /**
127 typedef enum {
128     TDC_MASK_CLOCK_CNTR_OVF_MASK_OFF = (0 << 2), // CLOCK Counter Overflow
129     Interrupt disabled
130     TDC_MASK_CLOCK_CNTR_OVF_MASK_ON  = (1 << 2), // CLOCK Counter Overflow
131     Interrupt enabled
132     TDC_MASK_COARSE_CNTR_OVF_MASK_OFF = (0 << 1), // Coarse Counter Overflow
133     Interrupt disabled
134     TDC_MASK_COARSE_CNTR_OVF_MASK_ON  = (1 << 1), // Coarse Counter Overflow
135     Interrupt enabled
136     TDC_MASK_NEW_MEAS_MASK_OFF      = (0 << 0), // New Measurement Interrupt
137     disabled
138     TDC_MASK_NEW_MEAS_MASK_ON       = (1 << 0), // New Measurement Interrupt
139     enabled
140 } TDC_mask_t;
141
142 /**
143 * @brief Error codes of the TDC module
144 */
145 /**
146 typedef enum {
147     TDC_OK = 0,
148     TDC_ERROR,
149     TDC_WRONG_ADDRESS,
150     TDC_WRONG_SIZE,
151     TDC_COM_ERROR,
152 } TDC_error_t;
153
154 /**
155 * @brief Handle for a TDC instance
156 */

```

```
147 */
148 typedef struct {
149     SPI_HandleTypeDef* spi;
150     GPIO_TypeDef* cs_port;
151     uint16_t cs_pin;
152     GPIO_TypeDef* en_port;
153     uint16_t en_pin;
154 } TDC_t;
155
156 /**
157 * @brief Initialize a TDC instance
158 *
159 * Note, SPI and GPIOs need to be initialized manually before calling this
160 * function.
161 *
162 * @param[out] tdc -- Pointer to TDC handle
163 * @param[in] spi -- Pointer to SPI handle
164 * @param[in] cs_port -- GPIO port of Chip Select
165 * @param[in] cs_pin -- GPIO pin of Chip Select
166 * @param[in] en_port -- GPIO port of Enable
167 * @param[in] en_pin -- GPIO pin of Enable
168 *
169 * @return TDC_error -- Error code
170 */
171 TDC_error_t TDC_init(TDC_t* tdc, SPI_HandleTypeDef* spi, GPIO_TypeDef* cs_port,
172                      uint16_t cs_pin, GPIO_TypeDef* en_port, uint16_t en_pin);
173
174 /**
175 * @brief Enable the TDC
176 *
177 * @param[in] tdc -- Pointer to TDC handle
178 *
179 * @return TDC_error_t -- Error code
180 */
181 TDC_error_t TDC_enable(TDC_t* tdc);
182
183 /**
184 * @brief Disable the TDC
185 *
186 * @param[in] tdc -- Pointer to TDC handle
187 *
188 * @return TDC_error_t -- Error code
189 */
190 TDC_error_t TDC_disable(TDC_t* tdc);
191
192 /**
193 * @brief Read register data from TDC
194 *
195 * @param[in] tdc -- Pointer to TDC handle
196 * @param[in] address -- Address of the register(s) to read
197 * @param[out] rx_data -- Pointer to data
198 *
199 * @attention @p rx_data needs to provide space for at least TDC_REG_SIZE[ @p
200 * address ] bytes.
201 *
202 * @return TDC_error_t -- Error code
203 */
204 TDC_error_t TDC_read(TDC_t* tdc, TDC_addr_t address, uint8_t* rx_data);
205
206 /**
207 * @brief Write register data to TDC
208 *
```

```

205 *
206 * @param[in] tdc      -- Pointer to TDC handle
207 * @param[in] address -- Address of the register(s) to write
208 * @param[in] tx_data -- Pointer to data
209 *
210 * @attention @p tx_data needs to provide space for at least TDC_REG_SIZE[ @p
211 *           address ] bytes.
212 *
213 */
214 TDC_error_t TDC_write(TDC_t* tdc, TDC_addr_t address, uint8_t* tx_data);
215
216 /**
217 * @brief Start measurement of TDC
218 *
219 * @param[in] tdc -- Pointer to TDC handle
220 *
221 * @return TDC_error_t -- Error code
222 */
223 TDC_error_t TDC_start(TDC_t* tdc);
224
225 /**
226 * @brief Read ToF measurement result from TDC
227 *
228 * @param[in] tdc -- Pointer to TDC handle
229 * @param[out] tof_fs -- ToF Measurement result [fs]
230 *
231 * @return TDC_error_t -- Error code
232 */
233 TDC_error_t TDC_read_result(TDC_t* tdc, uint64_t* tof_fs);
234
235 #endif /* TDC_H_ */

```

Code 10: TDC Driver (Header)

```

1 /*
2  * TDC.c
3  *
4  * Driver for TI TDC7200 -- Source file
5  *
6  * (C) T. Burkard | M. Schaeer
7  *
8  */
9
10 #include "TDC/TDC.h"
11 #include <stdbool.h>
12
13 #define TDC_AUTO_INC_OFF (0 << 7)
14 #define TDC_AUTO_INC_ON (1 << 7)
15
16 #define TDC_RW_READ (0 << 6)
17 #define TDC_RW_WRITE (1 << 6)
18
19 #define TDC_CMD_READ (TDC_AUTO_INC_ON | TDC_RW_READ)
20 #define TDC_CMD_WRITE (TDC_AUTO_INC_ON | TDC_RW_WRITE)
21
22 const uint8_t TDC_REG_SIZE[TDC_ADR_AMOUNT] = {
23     // Register Address                  Register Size [bytes]
24     [TDC_ADR_CONFIG1]                 = 1,
25     [TDC_ADR_CONFIG2]                 = 1,
26     [TDC_ADR_INT_STATUS]              = 1,

```

```
27     [TDC_ADR_INT_MASK]          = 1,
28     [TDC_ADR_COARSE_CNTR_OVF_H]  = 1,
29     [TDC_ADR_COARSE_CNTR_OVF_L]  = 1,
30     [TDC_ADR_CLOCK_CNTR_OVF_H]   = 1,
31     [TDC_ADR_CLOCK_CNTR_OVF_L]   = 1,
32     [TDC_ADR_CLOCK_CNTR_STOP_MASK_H] = 1,
33     [TDC_ADR_CLOCK_CNTR_STOP_MASK_L] = 1,
34     [TDC_ADR_TIME1]             = 3,
35     [TDC_ADR_CLOCK_COUNT1]       = 3,
36     [TDC_ADR_TIME2]             = 3,
37     [TDC_ADR_CLOCK_COUNT2]       = 3,
38     [TDC_ADR_TIME3]             = 3,
39     [TDC_ADR_CLOCK_COUNT3]       = 3,
40     [TDC_ADR_TIME4]             = 3,
41     [TDC_ADR_CLOCK_COUNT4]       = 3,
42     [TDC_ADR_TIME5]             = 3,
43     [TDC_ADR_CLOCK_COUNT5]       = 3,
44     [TDC_ADR_TIME6]             = 3,
45     [TDC_ADR_CALIBRATION1]      = 3,
46     [TDC_ADR_CALIBRATION2]      = 3,
47 };
48
49 static TDC_error_t send(TDC_t* tdc, TDC_addr_t address, uint8_t* data, bool
50                         write);
51
52 TDC_error_t TDC_init(TDC_t* tdc, SPI_HandleTypeDef* spi, GPIO_TypeDef* cs_port,
53                      uint16_t cs_pin, GPIO_TypeDef* en_port, uint16_t en_pin)
54 {
55     if ((tdc == NULL) || (spi == NULL) || (cs_port == NULL) || (en_port ==
56         NULL)) {
57         return TDC_ERROR;
58     }
59
60     tdc->spi      = spi;
61     tdc->cs_port  = cs_port;
62     tdc->cs_pin   = cs_pin;
63     tdc->en_port  = en_port;
64     tdc->en_pin   = en_pin;
65
66     return TDC_OK;
67 }
68
69 TDC_error_t TDC_enable(TDC_t* tdc)
70 {
71     if (tdc == NULL) {
72         return TDC_ERROR;
73     }
74
75     HAL_GPIO_WritePin(tdc->en_port, tdc->en_pin, GPIO_PIN_SET);
76
77     return TDC_OK;
78 }
79
80 TDC_error_t TDC_disable(TDC_t* tdc)
81 {
82     if (tdc == NULL) {
83         return TDC_ERROR;
84     }
85
86     HAL_GPIO_WritePin(tdc->en_port, tdc->en_pin, GPIO_PIN_RESET);
```

```
85     return TDC_OK;
86 }
87
88 TDC_error_t TDC_write(TDC_t* tdc, TDC_addr_t address, uint8_t* tx_data)
89 {
90     return send(tdc, address, tx_data, true);
91 }
92
93 TDC_error_t TDC_read(TDC_t* tdc, TDC_addr_t address, uint8_t* rx_data)
94 {
95     return send(tdc, address, rx_data, false);
96 }
97
98 TDC_error_t TDC_start(TDC_t* tdc)
99 {
100     uint8_t      data[TDC_REG_SIZE[TDC_ADR_CONFIG1]];
101     TDC_error_t error_code = TDC_OK;
102
103     if ((error_code = TDC_read(tdc, TDC_ADR_CONFIG1, data)) != TDC_OK) {
104         return error_code;
105     }
106
107     data[0] |= TDC_CONFIG1_START_MEAS;
108
109     return TDC_write(tdc, TDC_ADR_CONFIG1, data);
110 }
111
112 TDC_error_t TDC_read_result(TDC_t* tdc, uint64_t* tof_fs)
113 {
114     uint8_t time1_buf[TDC_REG_SIZE[TDC_ADR_TIME1]];
115     uint8_t time2_buf[TDC_REG_SIZE[TDC_ADR_TIME2]];
116     uint8_t clock_count1_buf[TDC_REG_SIZE[TDC_ADR_CLOCK_COUNT1]];
117     uint8_t calibration1_buf[TDC_REG_SIZE[TDC_ADR_CALIBRATION1]];
118     uint8_t calibration2_buf[TDC_REG_SIZE[TDC_ADR_CALIBRATION2]];
119     uint8_t config2_buf[TDC_REG_SIZE[TDC_ADR_CONFIG2]];
120
121     uint32_t time1;
122     uint32_t time2;
123     uint32_t clock_count1;
124     uint32_t calibration1;
125     uint32_t calibration2;
126     uint32_t calibration2_periods;
127
128     TDC_error_t error_code = TDC_OK;
129
130     // Read registers
131
132     if ((error_code = TDC_read(tdc, TDC_ADR_TIME1, time1_buf)) != TDC_OK) {
133         return error_code;
134     }
135
136     time1 = TDC_24BIT_BUF_TO_INT(time1_buf);
137
138     if ((error_code = TDC_read(tdc, TDC_ADR_TIME2, time2_buf)) != TDC_OK) {
139         return error_code;
140     }
141
142     time2 = TDC_24BIT_BUF_TO_INT(time2_buf);
143
144     if ((error_code = TDC_read(tdc, TDC_ADR_CLOCK_COUNT1, clock_count1_buf)) !=
```

```

145     return error_code;
146 }
147
148 clock_count1 = TDC_24BIT_BUF_TO_INT(clock_count1_buf);
149
150 if ((error_code = TDC_read(tdc, TDC_ADR_CALIBRATION1, calibration1_buf)) != TDC_OK) {
151     return error_code;
152 }
153
154 calibration1 = TDC_24BIT_BUF_TO_INT(calibration1_buf);
155
156 if ((error_code = TDC_read(tdc, TDC_ADR_CALIBRATION2, calibration2_buf)) != TDC_OK) {
157     return error_code;
158 }
159
160 calibration2 = TDC_24BIT_BUF_TO_INT(calibration2_buf);
161
162 if ((error_code = TDC_read(tdc, TDC_ADR_CONFIG2, config2_buf)) != TDC_OK) {
163     return error_code;
164 }
165
166 if (config2_buf[0] & TDC_CONFIG2_CALIBRATION2_PERIODS_10) {
167     calibration2_periods = 10;
168 } else if (config2_buf[0] & TDC_CONFIG2_CALIBRATION2_PERIODS_20) {
169     calibration2_periods = 20;
170 } else if (config2_buf[0] & TDC_CONFIG2_CALIBRATION2_PERIODS_40) {
171     calibration2_periods = 40;
172 } else { // TDC_CONFIG2_CALIBRATION2_PERIODS_2
173     calibration2_periods = 2;
174 }
175
176 // Calculations
177
178 double cal_count = (double)(calibration2 - calibration1) /
179 (calibration2_periods - 1);
180 uint64_t norm_lsb_fs = (uint64_t)(TDC_CLOCK_PERIOD_FS / cal_count);
181
182 *tof_fs = (int64_t)norm_lsb_fs * ((int64_t)time1 - (int64_t)time2) +
183 ((int64_t)clock_count1 * TDC_CLOCK_PERIOD_FS);
184
185 return TDC_OK;
186 }
187 /**
188 * @brief Read/write register data from/to TDC
189 *
190 * Function for both reading and writing register data, depending on @p write
191 * parameter.
192 *
193 * @param[in] tdc -- Pointer to TDC handle
194 * @param[in] address -- Address of the register(s) to read/write
195 * @param[in,out] data -- Pointer to data
196 * @param[in] write -- True = write, false = read
197 *
198 * @attention @p data needs to provide space for at least TDC_REG_SIZE[ @p
199 * address ] bytes.
200 *
201 * @return TDC_error_t -- Error code
202 */

```

```

200 static TDC_error_t send(TDC_t* tdc, TDC_addr_t address, uint8_t* data, bool
201   write)
202 {
203     uint8_t size; // number of data bytes
204     uint8_t cmd = address | (write ? TDC_CMD_WRITE : TDC_CMD_READ);
205
206     if ((tdc == NULL) || (data == NULL)) {
207         return TDC_ERROR;
208     }
209
210     if (address >= TDC_ADR_AMOUNT) {
211         return TDC_WRONG_ADDRESS;
212     }
213
214     size = TDC_REG_SIZE[address];
215
216     HAL_GPIO_WritePin(tdc->cs_port, tdc->cs_pin, GPIO_PIN_RESET);
217     HAL_Delay(1); // 1 ms
218
219     if (HAL_SPI_Transmit(tdc->spi, &cmd, 1, HAL_MAX_DELAY) != HAL_OK) {
220         return TDC_COM_ERROR;
221     }
222
223     if (write) {
224         if (HAL_SPI_Transmit(tdc->spi, data, size, HAL_MAX_DELAY) != HAL_OK) {
225             return TDC_COM_ERROR;
226         }
227     } else {
228         if (HAL_SPI_Receive(tdc->spi, data, size, HAL_MAX_DELAY) != HAL_OK) {
229             return TDC_COM_ERROR;
230         }
231     }
232
233     HAL_GPIO_WritePin(tdc->cs_port, tdc->cs_pin, GPIO_PIN_SET);
234     HAL_Delay(1); // 1 ms
235
236     return TDC_OK;
237 }
```

Code 11: TDC Driver (Source)

## 8.3 Python Skripte

In Code 12 ist das Python-Skript zur Berechnung des arithmetischen Mittelwerts und der Standardabweichung sowie zum Plotten des Histogramms und des Boxplots dargestellt.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Read the data from the log.txt file
5 with open('log.txt', 'r') as file:
6     tof_values = []
7     for line in file:
8         # Extract the ToF value using the pattern 'ToF = <value> [ps]'
9         if 'ToF' in line:
10             tof_value = int(line.split('=')[1].split('[')[0].strip())
11             tof_values.append(tof_value)
12
13 # Convert the list to a numpy array for easier calculation
14 tof_array = np.array(tof_values)
```

```

15
16 # Calculate the arithmetic mean and standard deviation
17 mean_tof = np.mean(tof_array)
18 std_tof = np.std(tof_array)
19
20 # Print the results
21 print(f'Arithmetic Mean of ToF: {mean_tof}')
22 print(f'Standard Deviation of ToF: {std_tof}')
23
24 # Plot a histogram
25 plt.figure(figsize=(10, 6))
26 plt.hist(tof_array, bins=20, color='skyblue', edgecolor='black', alpha=0.7)
27 plt.title('Histogram of ToF Values')
28 plt.xlabel('Time-of-Flight (ps)')
29 plt.ylabel('Frequency')
30 plt.grid(True)
31 plt.show()
32
33 # Plot a box plot
34 plt.figure(figsize=(10, 6))
35 plt.boxplot(tof_array, vert=False, patch_artist=True,
            boxprops=dict(facecolor='lightblue', color='black'))
36 plt.title('Box Plot of ToF Values')
37 plt.xlabel('Time-of-Flight (ps)')
38 plt.grid(True)
39 plt.show()

```

Code 12: Python Analyse

In Code 13 ist das Python-Skript zur Berechnung des arithmetischen Mittelwerts und der Standardabweichung sowie zum Plotten der Werte für mehrere ToF-Datensätze dargestellt.

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.ticker as mticker
4
5 # Load data from logs.txt, handling empty values
6 filename = 'logs.txt'
7 data = np.genfromtxt(filename, delimiter=';', dtype=float,
8                      filling_values=np.nan)
9
9 # Calculate arithmetic mean and standard deviation for each column, ignoring
10 # NaNs
11 means = np.nanmean(data, axis=0)
12 std_devs = np.nanstd(data, axis=0)
13
13 # Print results
14 for i, (mean, std_dev) in enumerate(zip(means, std_devs), start=1):
15     print(f"Column {i}: Mean = {mean:.2f}, Standard Deviation = {std_dev:.2f}")
16
17 # Plot all columns with points
18 plt.figure(figsize=(10, 6))
19 labels = ["0 m", "1 m", "2 m", "4 m", "6 m", "7 m"]
20 for i in range(data.shape[1]):
21     plt.scatter(range(len(data[:, i])), data[:, i], label=labels[i], s=5)
22
23 plt.title('ToF for different cable lengths')
24 plt.xlabel('Index')
25 plt.ylabel('ToF [ps]')
26 plt.gca().yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _:
f'{x:.0f}'.replace(',', '')))

```

```

27 plt.legend()
28 plt.grid()
29 plt.tight_layout()
30 plt.show()

```

Code 13: Python Analyse (Multi ToFs)

In Code 14 ist das Python-Skript zum Plotten der Werte aus mehreren Log-Files dargestellt.

```

1 import matplotlib.pyplot as plt
2 import matplotlib.ticker as mticker
3 import numpy as np
4
5 # Function to read ToF values from a log file
6 def read_tof_values(filename):
7     tof_values = []
8     with open(filename, 'r') as file:
9         for line in file:
10             if "ToF =" in line:
11                 try:
12                     value = int(line.split('=')[1].split()[0])
13                     tof_values.append(value)
14                 except (IndexError, ValueError):
15                     print(f"Skipping malformed line in {filename}:
{line.strip()}")
16     return tof_values
17
18 # Read ToF values from the three log files
19 log_files = ["log1.txt", "log2.txt", "log3.txt"]
20 datasets = []
21
22 for log_file in log_files:
23     datasets.append(read_tof_values(log_file))
24
25 # Combine all data for plotting and calculate indices
26 data_indices = []
27 offset = 0
28
29 for data in datasets:
30     indices = list(range(offset, offset + len(data)))
31     data_indices.append(indices)
32     offset += len(data)
33
34 # Plot the data
35 plt.figure(figsize=(10, 6))
36 colors = ['blue', 'orange', 'green'] # Colors for the datasets
37 labels = ['19 cm', '24 cm', '29 cm'] # Dataset labels
38
39 for i, (indices, data) in enumerate(zip(data_indices, datasets)):
40     # Scatter plot for the original data points
41     plt.scatter(indices, data, label=f'Distanz = {labels[i]}', color=colors[i],
s=10)
42
43     # Add mean value as a horizontal line within the range of the dataset
44     mean_value = np.mean(data)
45     plt.plot(indices, [mean_value] * len(indices), color=colors[i],
linestyle='--', linewidth=2, label=f'Distanz = {labels[i]} (Mean)')
46
47     # Linear fit for the dataset (polynomial of degree 1, i.e., a straight line)
48     slope, intercept = np.polyfit(indices, data, 1) # Fit a line to the data
49     fit_line = np.polyval([slope, intercept], indices) # Calculate the fitted
values

```

```

50     plt.plot(indices, fit_line, color=colors[i], linestyle=':', linewidth=2,
51               label=f'Distanz = {labels[i]} (Linear Fit)')
52
53 # Customize the plot
54 plt.title('ToF unterschiedliche Distanzen zum Spiegel')
55 plt.xlabel('Index')
56 plt.ylabel('ToF [ps]')
57 plt.gca().yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _:
58     f'{x:.0f}'.replace(',', '')))
59 plt.legend()
60 plt.grid(True)
61
62 # Show the plot
63 plt.show()

```

Code 14: Python Analyse (Multi Logs)

In Code 15 ist das Python-Skript zum live Plotten von ToF-Werte dargestellt. Dieses wird während der Präsentation für die Demonstration verwendet.

```

1 import serial
2 import matplotlib.pyplot as plt
3 import matplotlib.animation as animation
4 import matplotlib.ticker as mticker
5 from collections import deque
6 import re
7
8 # Configuration
9 COM_PORT      = 'COMX'
10 BAUD_RATE    = 38400
11 MAX_SAMPLES  = 20000
12
13 # Calibration (zero point)
14 TOF_0M_PS = 25800 # ToF at cable length 0m in [ps]
15
16 # Constants
17 TOF_MIN_PS      = 0                      # ToF min. value (sanity check) [ps]
18 TOF_MAX_PS      = 30000 + TOF_0M_PS       # ToF max. value (sanity check) [ps]
19 TOF_PS_TO_LEN_CM = 0.9 * 3e8 / 1e12 * 100 # from [ps] to [cm] via c
20
21 def parse_tof(line):
22     """Extract the ToF value from a line of text."""
23     match = re.search(r'ToF\s*=\s*(\d+)\s*[ps]', line)
24     if match:
25         return int(match.group(1))
26     return None
27
28 def update(frame):
29     """Update the plot with new data."""
30     global data, ser
31
32     try:
33         if ser.in_waiting > 0:
34             lines = ser.read(ser.in_waiting).decode('utf-8').splitlines()
35             for line_content in lines:
36                 tof_value = parse_tof(line_content)
37                 if tof_value is not None and tof_value > TOF_MIN_PS and
tof_value < TOF_MAX_PS: # Skip invalid values
38                     tof_value -= TOF_0M_PS # Calibration
39                     data.append(tof_value)
40

```

```

41             # Print latest value
42             tof_str_ps = f"{tof_value:.0f}".replace(","," ")    #
43             1000th separator
44             tof_cm      = round(tof_value * TOF_PS_TO_LEN_CM)      #
45             No decimal places
46             line_text.set_text(f"Latest ToF: {tof_str_ps} ps | {tof_cm}#
47             cm")
48
49             # Update the Line2D object
50             line.set_data(range(len(data)), data)
51             ax.relim()
52             ax.autoscale_view()
53             except Exception as e:
54                 print(f"Error reading from serial port: {e}")
55
56 # Initialize serial port
57 try:
58     ser = serial.Serial(COM_PORT, BAUD_RATE, timeout=1)
59     print(f"Connected to {COM_PORT} at {BAUD_RATE} baud.")
60 except Exception as e:
61     print(f"Failed to connect to serial port: {e}")
62     exit()
63
64 # Initialize plot
65 fig, ax = plt.subplots()
66 data = deque(maxlen=MAX_SAMPLES) # Increase buffer to hold up to 2000 samples
67 line, = ax.plot([], [], label="ToF [ps]") # Create a Line2D object
68 ax.set_ylabel("Time of Flight [ps]")
69 ax.set_xlabel("Index")
70 plt.gca().yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _:
71     f"{x:.0f}".replace(","," ")))
72
73 ax_m = ax.twinx() # Create secondary y-axis
74 ax_m.set_ylabel("Cable Length [cm]") # Label for secondary y-axis
75
76 line_text = plt.text(0.5, 0.95, "", transform=plt.gca().transAxes, ha="center")
77
78 def update_secondary_yaxis_limits(*args):
79     """Update the secondary y-axis limits whenever the primary y-axis
80     changes."""
81     primary_limits = ax.get ylim()
82     ax_m.set ylim(primary_limits[0] * TOF_PS_TO_LEN_CM, primary_limits[1] *
83     TOF_PS_TO_LEN_CM)
84
85 # Connect the limit update to the primary y-axis changes
86 ax.callbacks.connect("ylim_changed", update_secondary_yaxis_limits)
87
88 # Initial call to synchronize the limits
89 update_secondary_yaxis_limits()
90
91 ani = animation.FuncAnimation(fig, update, interval=50)
92
93 # Show plot
94 try:
95     plt.show()
96 except KeyboardInterrupt:
97     print("Exiting...")
98 finally:
99     ser.close()

```

Code 15: Python Live Plot

## 8.4 Schema

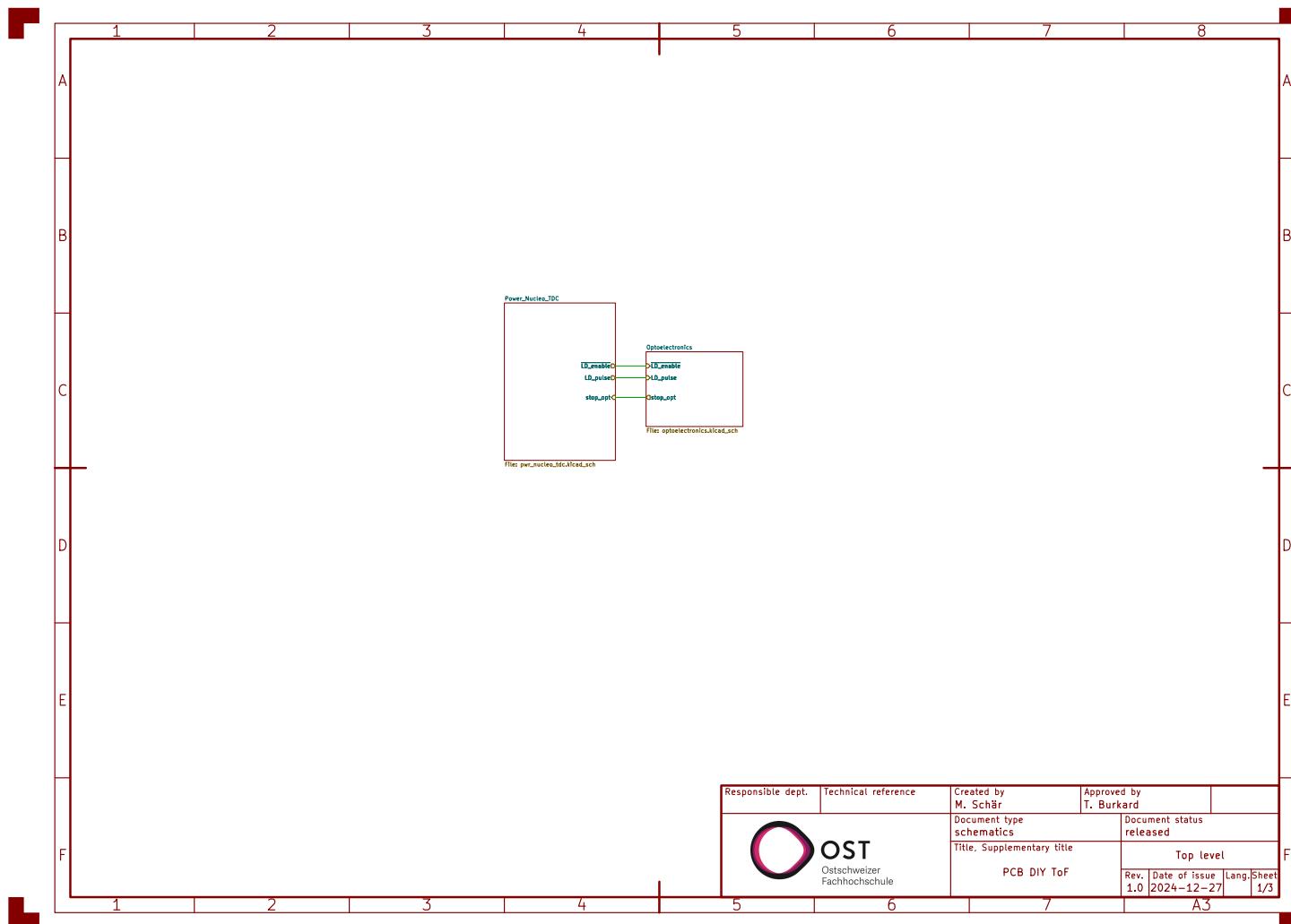


Abbildung 56: Schema S.1/3

# DIY Optische ToF Distanzmessung

## Projektarbeit

OST – Ostschweizer Fachhochschule  
CAS Sensorik und Sensor Signal Conditioning

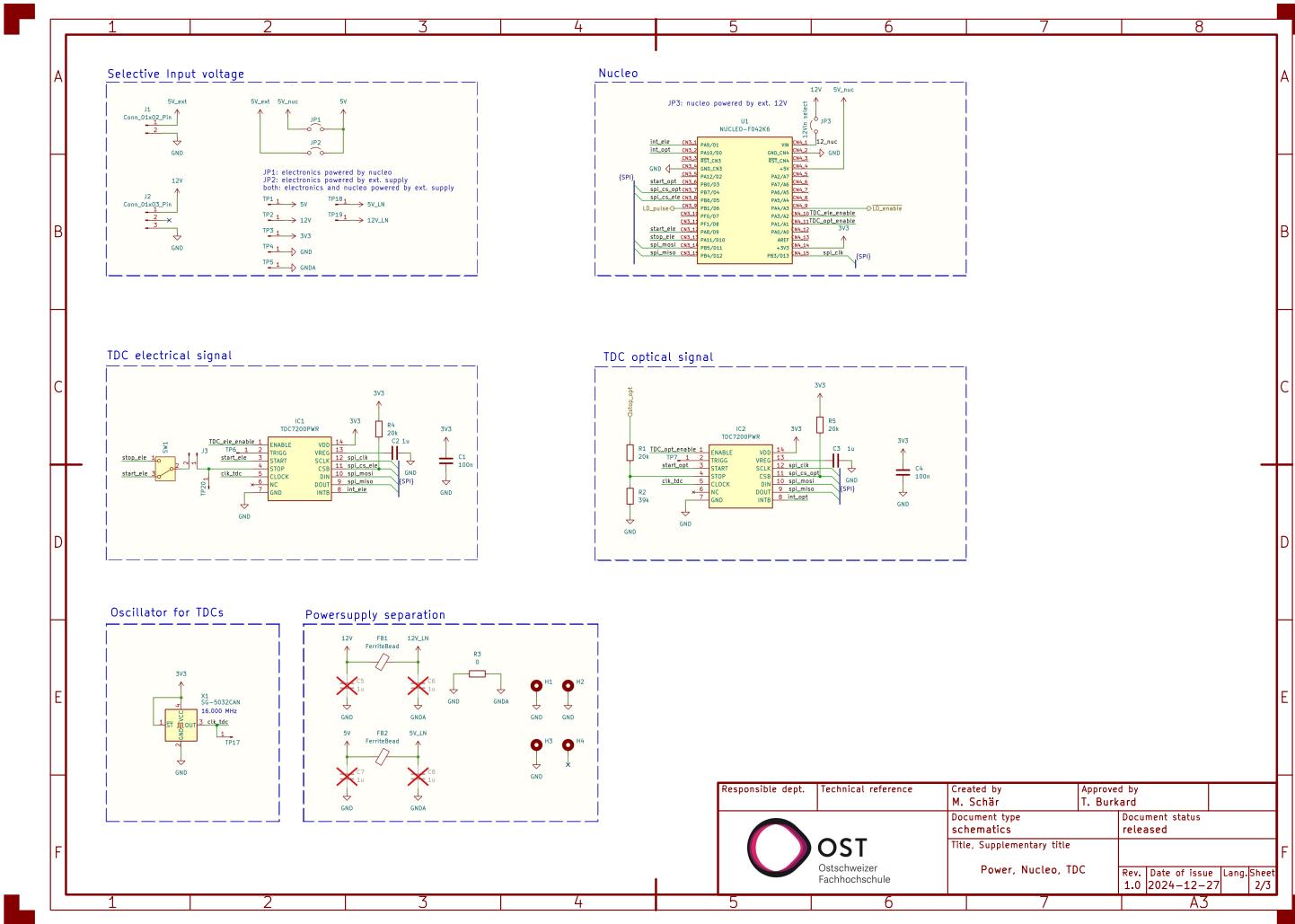


Abbildung 57: Schema S.2/3

# DIY Optische ToF Distanzmessung

Projektarbeit

**OST – Ostschweizer Fachhochschule**  
CAS Sensorik und Sensor Signal Conditioning

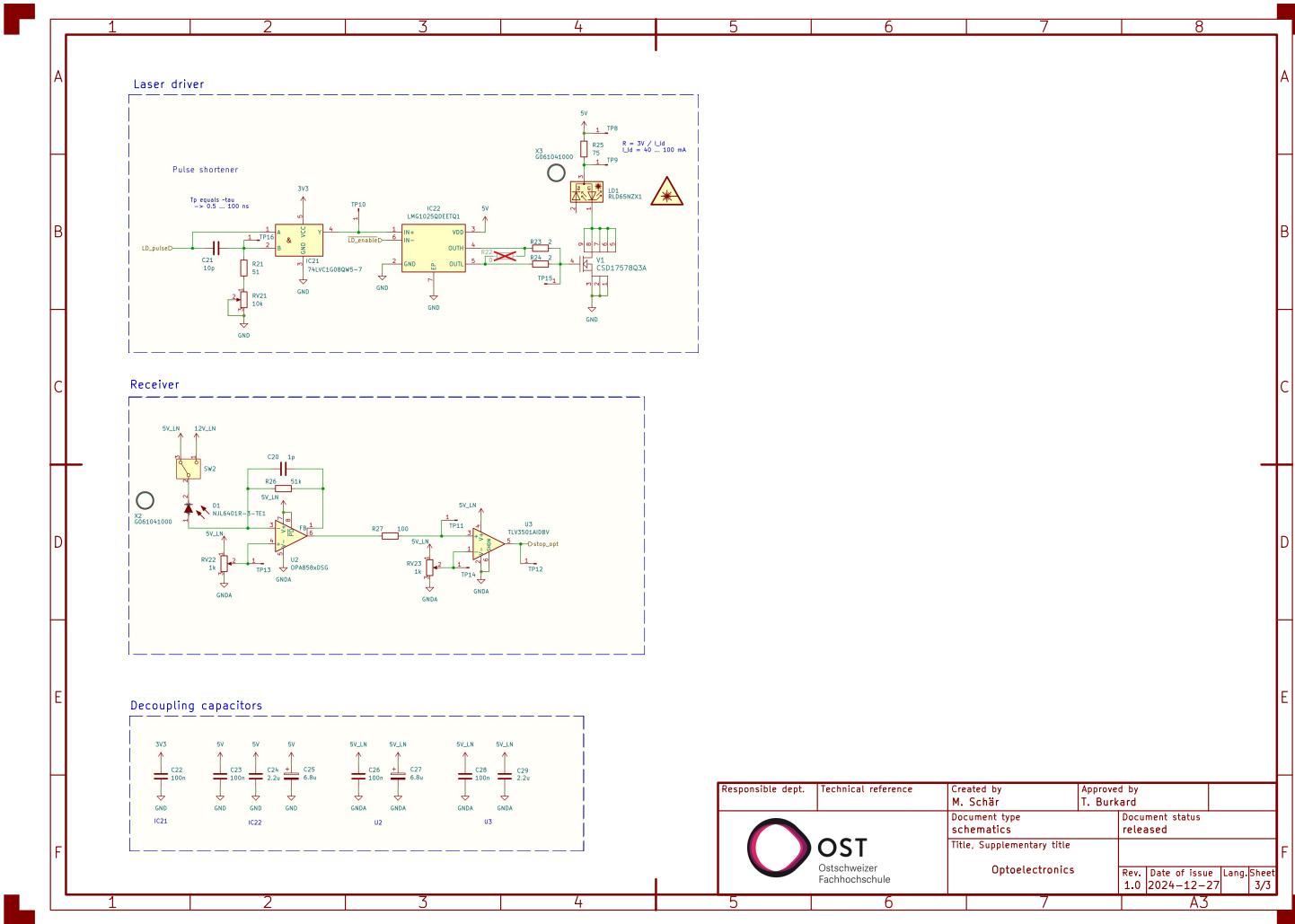


Abbildung 58: Schema S.3/3

## 8.5 Stückliste

Reference	Value	Datasheet	Footprint	Qty	DNP
C1,C4,C22,C23,C26,C28	100n		Capacitor_SMD:C_0402_1005Metric_Pad0.74x0.62mm	6	
C2,C3	1u		Capacitor_SMD:C_0402_1005Metric_Pad0.74x0.62mm	2	
C5,C6,C7,C8	1u		Capacitor_SMD:C_0402_1005Metric_Pad0.74x0.62mm	4	DNP
C20	1p		Capacitor_SMD:C_0402_1005Metric_Pad0.74x0.62mm	1	
C21	10p		Capacitor_SMD:C_0402_1005Metric_Pad0.74x0.62mm	1	
C24,C29	2.2u		Capacitor_SMD:C_0402_1005Metric_Pad0.74x0.62mm	2	
C25,C27	6.8u		Capacitor_SMD:C_0603_1608Metric_Pad1.08x0.95mm	2	
D1	NJL6401R-3-TE1	(JRC, 2014)	Capacitor_Tantal_SMD:CP_EIA-7343-43_Kemet-X_Pad2.25x2.55mm	2	
FB1,FB2	FerriteBead	(TDK, 2019)	NJL6401R3TE1	1	
IC1,IC2	TDC7200PWR	(TI, 2016b)	Inductor_SMD:L_0402_1005Metric_Pad0.77x0.64mm	2	
IC21	74LVC1G08QW5-7	(Diodes Inc., 2020)	SOP65P640X120-14N	2	
IC22	LMG1025QDEETQ1	(TI, 2024c)	74LVC1G08QW57	1	
J1,J3	Conn_01x02_Pin		SON65P200X200X80-7N	1	
J2	Conn_01x03_Pin		Connector:Molex_KK-254_AE-6410-02A_1x02_P2.54mm_Vertical	2	
JP1,JP2	Jumper_2_Open		Connector:Molex_SL_171971-0003_1x03_P2.54mm_Vertical	1	
JP3	12Vin select		TestPoint:TestPoint_2Pads_Pitch2.54mm_Drill0.8mm	2	
LD1	RLD65NZX1-00A	(ROHM, 2019)	TestPoint:TestPoint_2Pads_Pitch2.54mm_Drill0.8mm	1	
R1,R4,R5	20k		RLD65NZX1-00A:RLD85PZJ400A	1	
R2	39k		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	3	
R3	0		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	1	
R21	51		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	1	
R22	0		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	1	
R23,R24	2		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	2	DNP
R25	75		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	1	
R26	51k		Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm	1	
R27	100		Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	1	
RV21	10k	(Bourns, 2024)	Resistor_SMD:R_0402_1005Metric_Pad0.72x0.64mm	1	
RV22,RV23	1k	(Bourns, 2024)	Potentiometer_SMD:Potentiometer_Bourns_3224W_Vertical	1	
SW1,SW2	SW_Nidec_CAS-120A1	(Nidec Comp., 2024)	Potentiometer_SMD:Potentiometer_Bourns_3224W_Vertical	2	
TP1..TP20	Conn_01x01_Pin		Button_Switch_SMD:Nidec_Copal_CAS-120A	2	
U1	NUCLEO-F042K6	(ST, 2019)	Connector_2.54mm:PinHeader_1x01_P2.54mm_Vertical	20	
U2	OPA858xDSG	(TI, 2018)	NUCLEO_F042K6:MODULE_NUCLEO-F042K6	1	
U3	TLV3501AIDBV	(TI, 2016c)	Package SON:WSON-8-1EP_2x2mm_P0.5mm_EP0.9x1.6mm	1	
V1	CSD17578Q3A	(TI, 2016a)	Package TO_SOT_SMD:SOT-23-6	1	
X1	SG-5032CAN	(Epson Timing, 2024)	DNH0008A	1	
X2,X3	G061041000	(Qioptiq, 2024a)	Oscillator:Oscillator_SMD_SeikoEpson_SG8002LB-4Pin_5.0x3.2mm user:Qioptiq-Mount	1	
				2	

Tabelle 12: Bill of Material

## 8.6 Layout

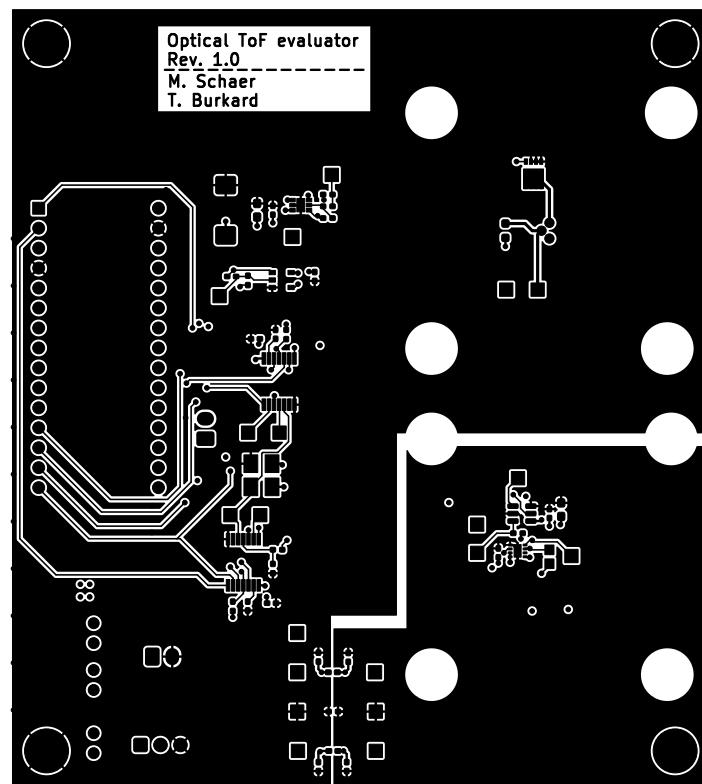


Abbildung 59: PCB Layout Top

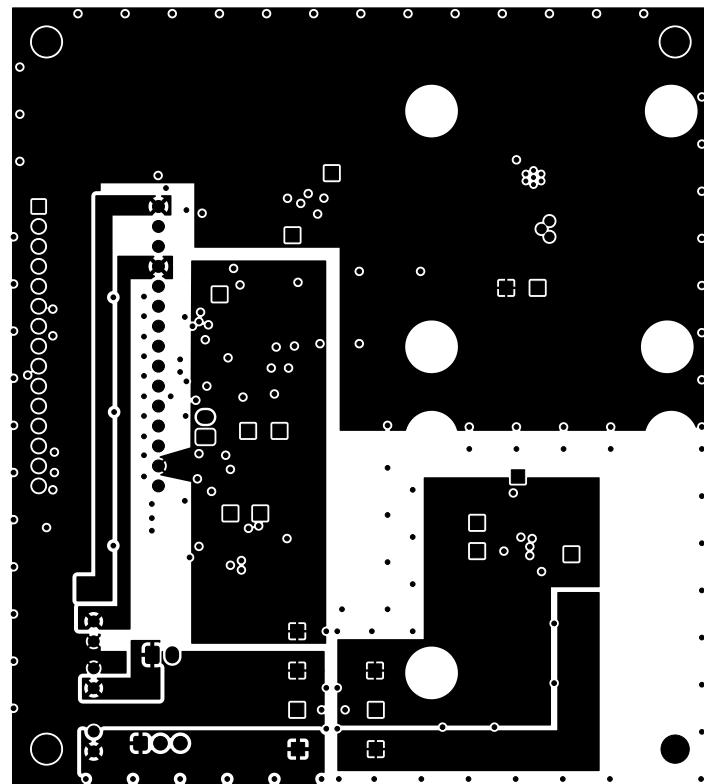


Abbildung 60: PCB Layout Innen 1

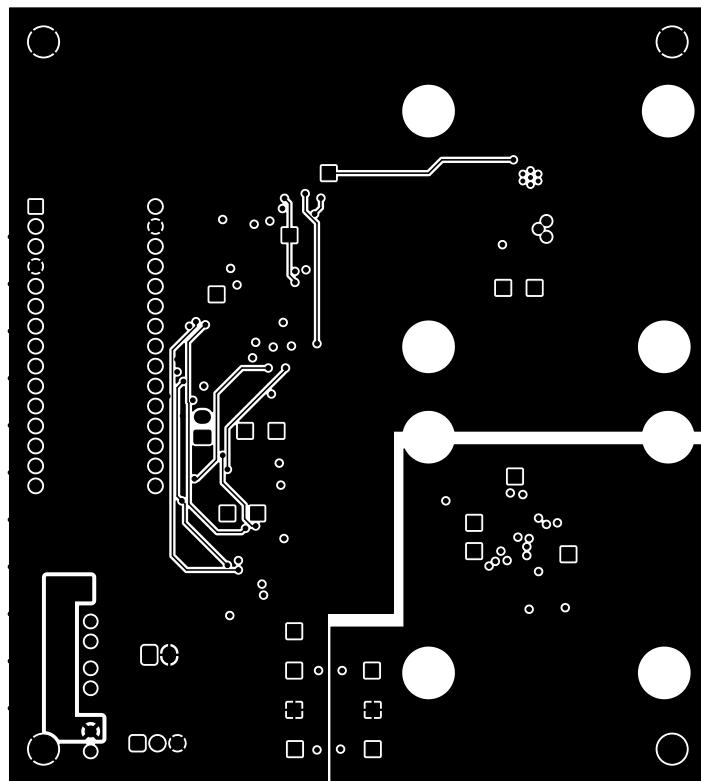


Abbildung 61: PCB Layout Innen 2

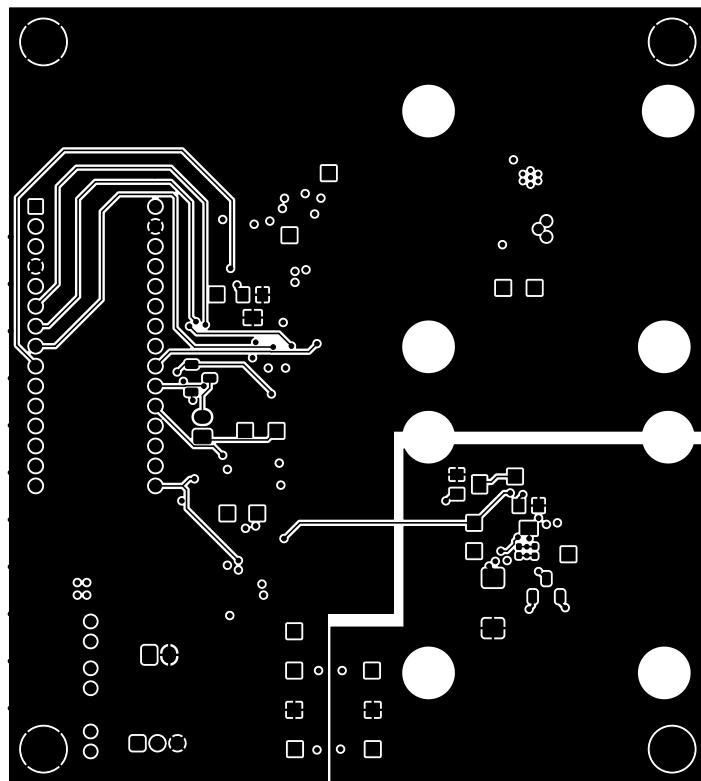


Abbildung 62: PCB Layout Bottom

## 8.7 Komponenten-Platzierung

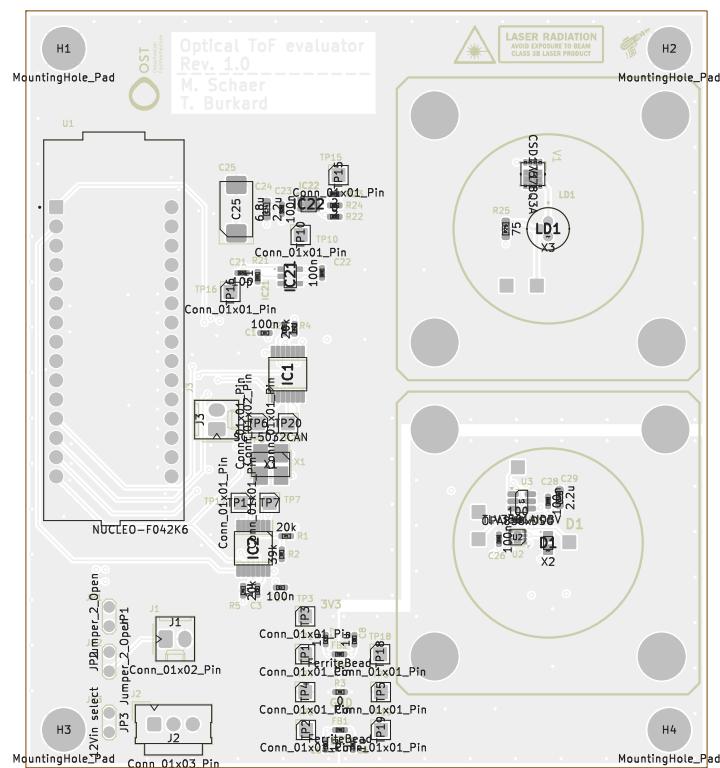


Abbildung 63: PCB Komponenten-Platzierung Top

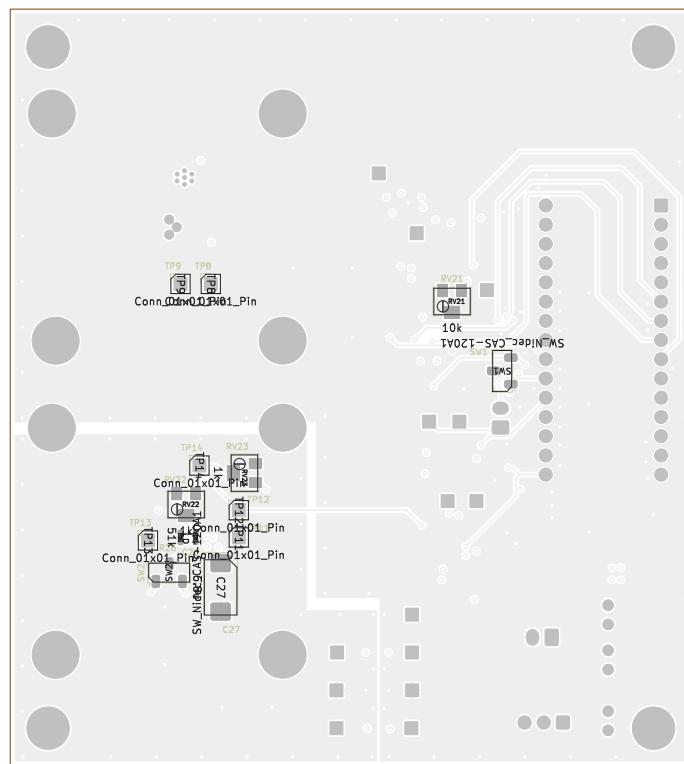


Abbildung 64: PCB Komponenten-Platzierung Bottom

## 8.8 3D View

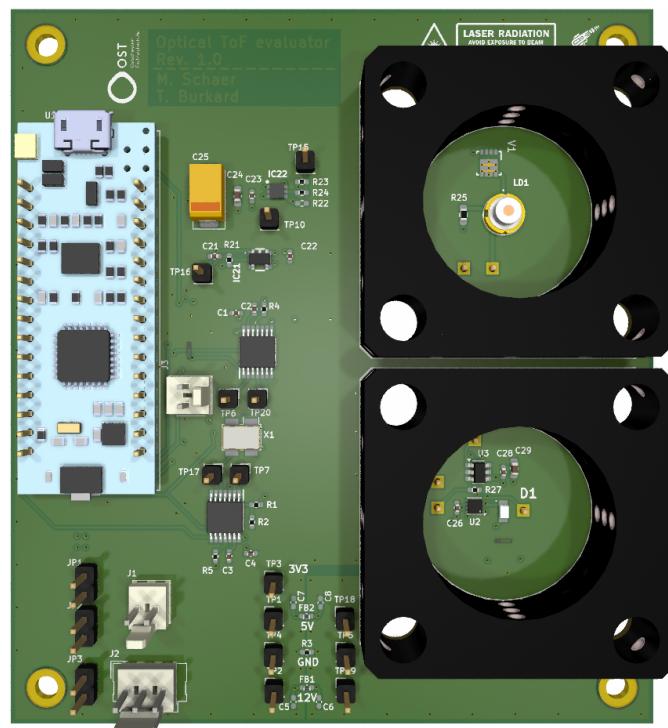


Abbildung 65: 3D View Top

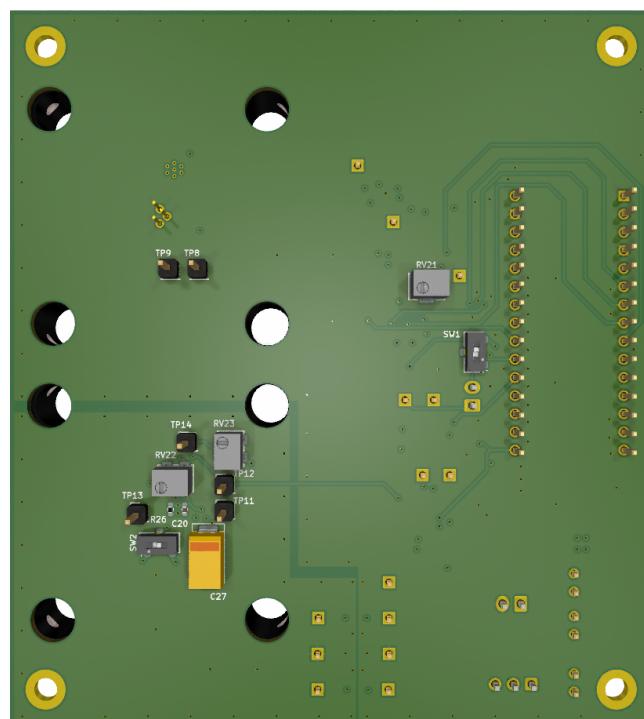


Abbildung 66: 3D View Bottom

## 8.9 Fotos Demonstrator



Abbildung 67: Demonstrator von oben

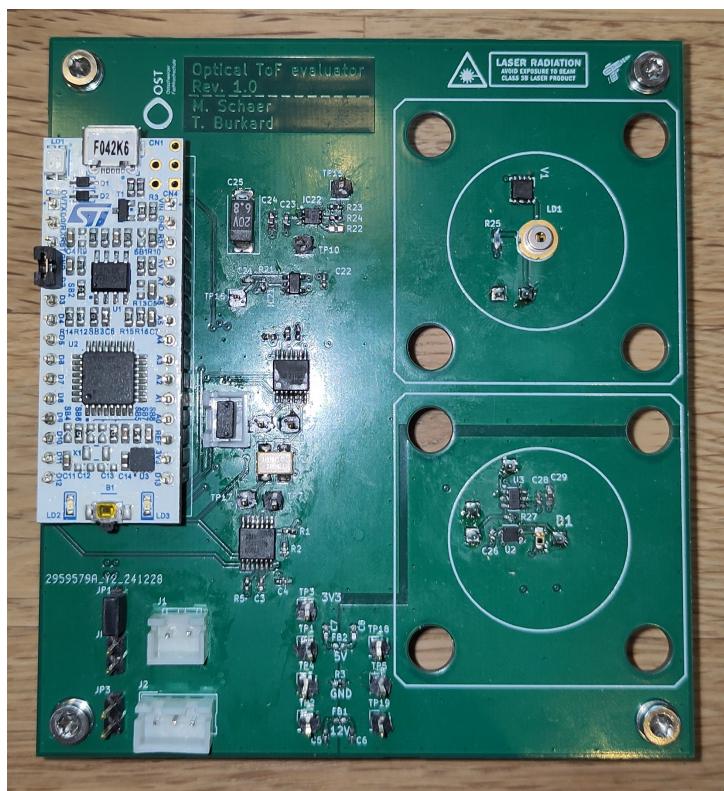


Abbildung 68: Demonstrator von oben ohne Linse

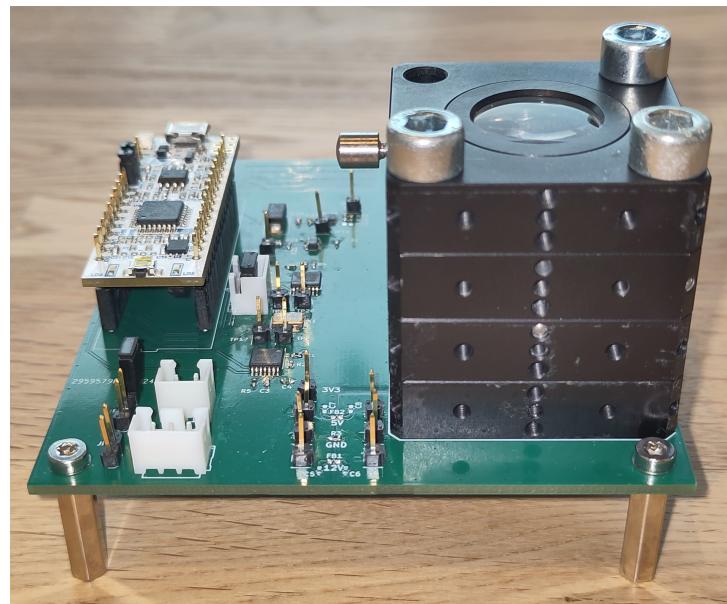


Abbildung 69: Demonstrator von vorne



Abbildung 70: Demonstrator von vorne ohne Linse

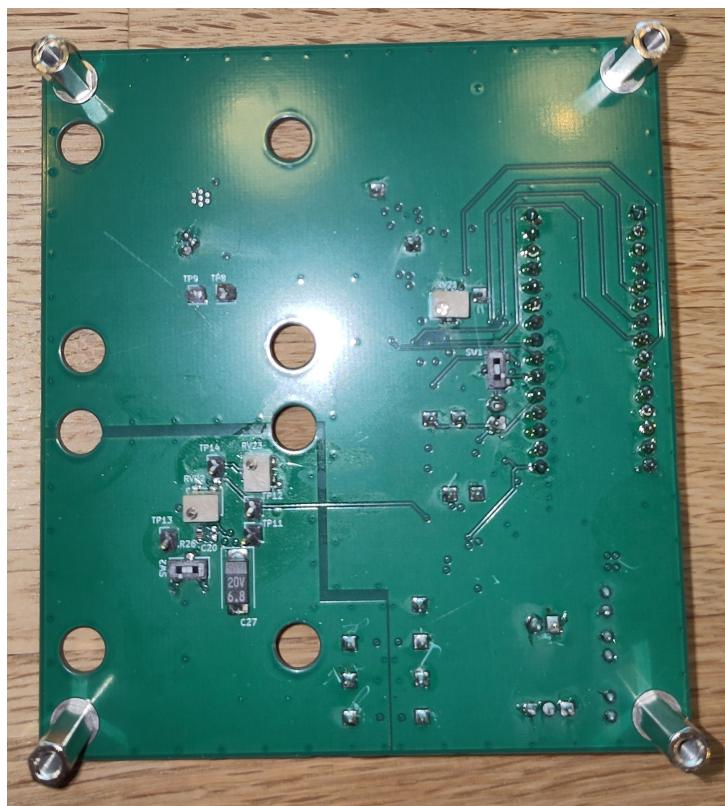


Abbildung 71: Demonstrator von unten

# Quellenverzeichnis

- Bourns. (2024). *3224W-2-10XE Datasheet*. Zugriff auf  
<https://www.mouser.ch/datasheet/2/54/3224-776900.pdf> (aufgerufen am 26.12.2024)
- Diodes Inc. (2020). *74LVC1G08Q Datasheet*. Zugriff auf  
[https://www.mouser.ch/datasheet/2/115/DIOD\\_S\\_A0010762531\\_1-2543394.pdf](https://www.mouser.ch/datasheet/2/115/DIOD_S_A0010762531_1-2543394.pdf) (aufgerufen am 26.12.2024)
- Epson Timing. (2024). *SG5032CAN Datasheet*. Zugriff auf  
[https://www.mouser.ch/datasheet/2/137/SG5032CAN\\_en-961596.pdf](https://www.mouser.ch/datasheet/2/137/SG5032CAN_en-961596.pdf) (aufgerufen am 26.12.2024)
- firewall.xc. (2025). *Propagation Delay*. Zugriff auf  
<https://www.firewall.cx/networking/ethernet/propagation-delay.html> (aufgerufen am 12.01.2025)
- JLC. (2025). *JLCPCB*. Zugriff auf <https://jlcpcb.com/> (aufgerufen am 18.01.2025)
- JRC. (2014). *NJL6401R-3 Datasheet*. Zugriff auf  
[https://www.mouser.ch/datasheet/2/294/NJL6401R\\_3\\_E-1019028.pdf](https://www.mouser.ch/datasheet/2/294/NJL6401R_3_E-1019028.pdf) (aufgerufen am 26.12.2024)
- KiCad. (2025). *KiCad EDA*. Zugriff auf <https://www.kicad.org/> (aufgerufen am 16.01.2025)
- Nidec Comp. (2024). *CAS-120A1 Datasheet*. Zugriff auf  
<https://www.mouser.ch/datasheet/2/972/cas-1628136.pdf> (aufgerufen am 26.12.2024)
- Qioptiq. (2024a). *G061041000 Datasheet*. Zugriff auf <https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/692/G061041000.pdf> (aufgerufen am 26.12.2024)
- Qioptiq. (2024b). *G061041000 Shop-Seite*. Zugriff auf  
<https://www.qioptiq-shop.com/en/Optomechanics/LINOS-Microbench-30-mm-Cage-System/Mounting-Plates/Mounting-Plate-25-with-Mounting-Holes.html> (aufgerufen am 26.12.2024)
- ROHM. (2019). *RLD65NZZ1 Datasheet*. Zugriff auf [https://fscdn.rohm.com/en/products/databook/datasheet/ opto/laser\\_diode/red/rld65nzz100a008-e.pdf](https://fscdn.rohm.com/en/products/databook/datasheet/ opto/laser_diode/red/rld65nzz100a008-e.pdf) (aufgerufen am 26.12.2024)
- ROHM. (2020). *RLD94PZJ5 Datasheet*. Zugriff auf [https://fscdn.rohm.com/en/products/databook/datasheet/ opto/laser\\_diode/infrared/rld94pzj5-e.pdf](https://fscdn.rohm.com/en/products/databook/datasheet/ opto/laser_diode/infrared/rld94pzj5-e.pdf) (aufgerufen am 26.12.2024)
- Siemens. (2025). *Xpedition AMS*. Zugriff auf  
<https://resources.sw.siemens.com/en-US/fact-sheet-xpedition-ams/> (aufgerufen am 16.01.2025)
- ST. (2017). *STM32F042K6 Datasheet*. Zugriff auf  
<https://www.st.com/resource/en/datasheet/stm32f042k6.pdf> (aufgerufen am 13.01.2025)
- ST. (2019). *NUCLEO-F042K6 Datasheet*. Zugriff auf  
<https://www.mouser.ch/datasheet/2/389/nucleo-f031k6-1484037.pdf> (aufgerufen am 26.12.2024)
- ST. (2020). *Description of STM32F0 HAL*. Zugriff auf  
[https://www.st.com/resource/en/user\\_manual/dm00122015-description-of-stm32f0-hal-and-low-layer-drivers-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/dm00122015-description-of-stm32f0-hal-and-low-layer-drivers-stmicroelectronics.pdf) (aufgerufen am 12.01.2025)

- ST. (2024). *NUCLEO-F042K6 Usermanual*. Zugriff auf [https://www.st.com/resource/en/user\\_manual/um1956-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/um1956-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf) (aufgerufen am 26.12.2024)
- ST. (2025). *STM32CubeIDE*. Zugriff auf  
<https://www.st.com/en/development-tools/stm32cubeide.html> (aufgerufen am 18.01.2025)
- TDK. (2019). *MPZ0402S100 Datasheet*. Zugriff auf  
[https://product.tdk.com/system/files/dam/doc/product/emc/emc/beads/catalog/beads\\_commercial\\_power\\_mpz0402\\_en.pdf](https://product.tdk.com/system/files/dam/doc/product/emc/emc/beads/catalog/beads_commercial_power_mpz0402_en.pdf) (aufgerufen am 26.12.2024)
- TI. (2016a). *CSD17578Q3A Datasheet*. Zugriff auf  
<https://www.ti.com/lit/ds/symlink/csd17578q3a.pdf?ts=1735200469410> (aufgerufen am 26.12.2024)
- TI. (2016b). *TDC7200 Datasheet*. Zugriff auf <http://www.ti.com/lit/gpn/tdc7200> (aufgerufen am 25.12.2024)
- TI. (2016c). *TLV3501 Datasheet*. Zugriff auf  
<https://www.ti.com/lit/ds/symlink/tlv3501.pdf?ts=1735168630258> (aufgerufen am 26.12.2024)
- TI. (2018). *OPA858 Datasheet*. Zugriff auf  
<https://www.ti.com/lit/ds/symlink/opa858.pdf?ts=1735178891996> (aufgerufen am 26.12.2024)
- TI. (2024a). *CSD17578Q3A*. Zugriff auf <https://www.ti.com/product/CSD17578Q3A> (aufgerufen am 28.12.2024)
- TI. (2024b). *LMG1025-Q1*. Zugriff auf <https://www.ti.com/product/LMG1025-Q1> (aufgerufen am 28.12.2024)
- TI. (2024c). *LMG1025-Q1 Datasheet*. Zugriff auf  
<https://www.ti.com/lit/ds/symlink/lmg1025-q1.pdf?ts=1735200095732> (aufgerufen am 26.12.2024)
- TI. (2024d). *OPA858*. Zugriff auf <https://www.ti.com/product/OPA858> (aufgerufen am 28.12.2024)
- Vishay Semic. (2024). *BPV23NF Datasheet*. Zugriff auf  
<https://www.vishay.com/docs/81513/bpv23nf.pdf> (aufgerufen am 26.12.2024)