

# **Geschwindigkeitserfassung Ballwurfmaschine**

Projektarbeit HS18

**Autor**

Luca Mazzoleni

**Dozent**

Markus Kottmann

**Betreuer**

Daniel Raillard

**Modul**

Einführung Programmierung

HSR Hochschule für Technik Rapperswil

4. Dezember 2018

# Inhaltsverzeichnis

<b>1 Aufgabenstellung</b>	<b>2</b>
1.1 Vorgehen . . . . .	2
<b>I Hardware</b>	<b>3</b>
<b>2 Komponenten</b>	<b>3</b>
2.1 Technische Daten der Lichtschranke . . . . .	4
2.2 Arduino . . . . .	5
<b>II Software</b>	<b>6</b>
<b>3 Programm</b>	<b>6</b>
3.1 Modi . . . . .	7
3.2 Globale Variablen . . . . .	7
3.3 Hilfsfunktionen . . . . .	7
3.3.1 calculate_velocity_ms() . . . . .	7
3.3.2 printlog() . . . . .	7
3.4 Polling-Modus . . . . .	8
3.5 Interrupt-Modus . . . . .	8
3.5.1 Hilfsfunktionen . . . . .	9
3.5.1.1 ISRLB1() . . . . .	9
3.5.1.2 ISRLB2() . . . . .	9
3.6 Hauptfunktionen . . . . .	9
3.6.1 setup() . . . . .	9
3.6.2 loop() . . . . .	9
<b>4 Python</b>	<b>10</b>
4.1 Randbedingungen . . . . .	10
4.2 Messdaten . . . . .	10
4.3 Ablauf . . . . .	10
<b>III Versuch</b>	<b>11</b>
<b>5 Aufbau</b>	<b>11</b>
<b>6 Auswertung</b>	<b>12</b>
6.1 Fehlerquellen . . . . .	15
<b>IV Fazit</b>	<b>16</b>
<b>7 Empfehlung</b>	<b>16</b>
<b>Anhang</b>	<b>17</b>

# 1 | Aufgabenstellung

Für die Ballschussmaschiene Unihockey soll zusätzlich eine Geschwindigkeitserfassung entworfen werden.

Dazu gibt es folgende Randbedingungen:

Randbedingungen

- Messung der Ballgeschwindigkeit (max. 180km/h).
- Als Zusatz vor die Maschine zu hängen, Befestigung z.B. magnetisch.
- Eigenständige Einheit mit Arduino und Anzeige.
- Bei Bedarf serielle Verbindung zum Master-Arduino möglich, z.B. um Richtungswinkel zu berücksichtigen.
- Sensorik: Z.B. zwei Lichtschranken in einem fixen Abstand, welche vom Ball unterbrochen werden. Evtl. führt man den Lichtstrahl mittels Spiegel mehrmals durch den Flugbereich, um den Einfluss der Flugbahnrichtung und der Kugelform des Balles zu reduzieren.

## 1.1 | Vorgehen

Es wurde folgendermassen vorgegangen:

Vorgehen

- Überblick über die bestehende Anlage verschafft.
- Mögliche Anordnung der Lichtschranken skizziert.
  - Flugzeiten berechnet.
- Kritische Parameter und mögliche Fehlerquellen identifiziert.
- Passende Lichtschranke ausgewählt und beschafft.
- Versuchsaufbau erstellt.
- Benötigte Software erstellt.
- Resultate validiert.

## Teil I

# Hardware

## 2 | Komponenten

Die Abb. 1 zeigt das Zusammenspiel der unterschiedlichen Komponenten vereinfacht dargestellt. Die Hauptkomponenten bilden dabei die Energieversorgung, der Antrieb zur Erzeugung der Drehbewegung, das Embedded System zur Erfassung der Geschwindigkeit und der PC zur Programmierung und Auswertung der generierten Daten.

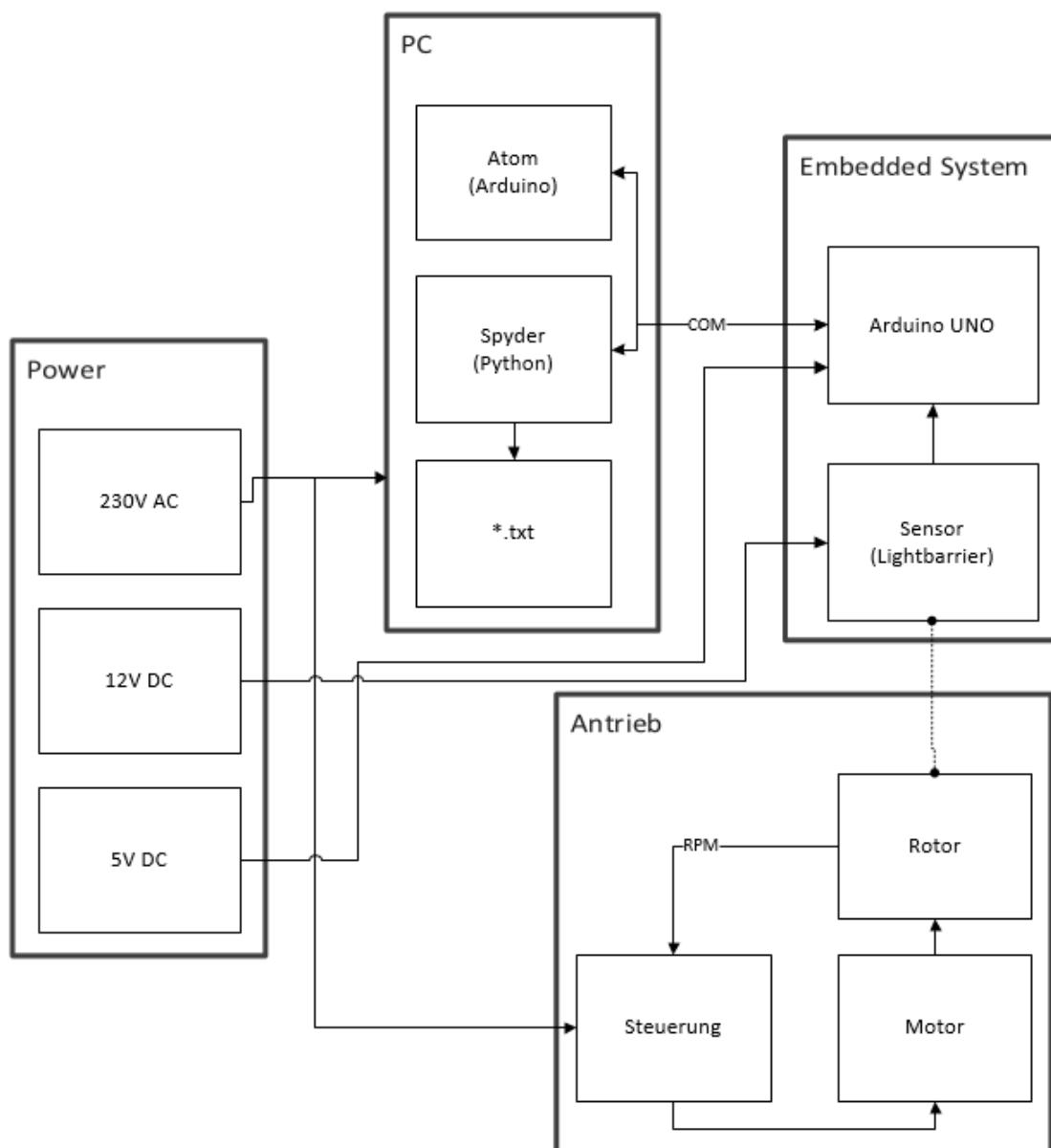


Abb. 1: Übersicht der Hardware

## 2.1 | Technische Daten der Lichtschanke

### Lichtschanke

Hersteller	Panasonic
Typ	Einweglichtschanke
Modellnummer	EX-21A-PN
Schalttyp	Hell-EIN
Reichweite	1m
Wiederholpräzision	max. 0.05 mm
Ansprechzeit	max. 0.5 ms
Spannung	12 bis 24V DC ± 10%
Stromaufnahme	max. 10 mA

Die kompletten Technischen Daten findet man in Anhang A.

Bei der Auswahl der Lichtschanke wurde vor allem drauf geachtet, dass sie eine schnelle Reaktionszeit hat sowie eine gute Wiederholpräzision. Weiter war es wichtig, dass sie Aktiv-Low ist, da nicht alle Arduinos auf ein HIGH-Signal einen Interrupt auslösen können.

## 2.2 | Arduino

Als Embedded System wird ein Arduino Uno verwendet. Er wird wie in Abb. 2 gezeigt angeschlossen. Um die Ausgangsspannung der Lichtschranke von 12V auf das 5V-Level des Arduinos zu reduzieren wird ein Spannungsteiler verwendet. Die Berechnung dazu findet sich im Anhang B.

Für den Anschluss der Peripherie werden folgende Pins benötigt:

**Anschluss**

**Pin Bezeichnung**

- 2 Lichtschranke 1 (LB1)
- 3 Lichtschranke 2 (LB2)

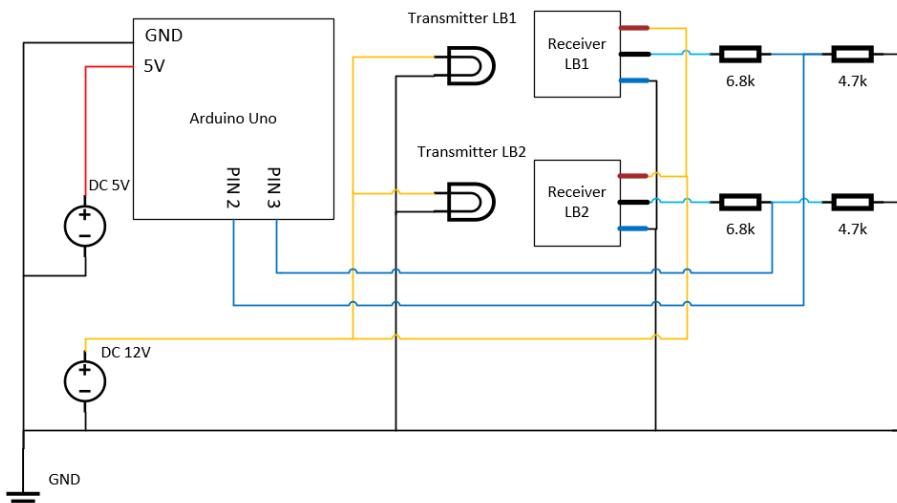
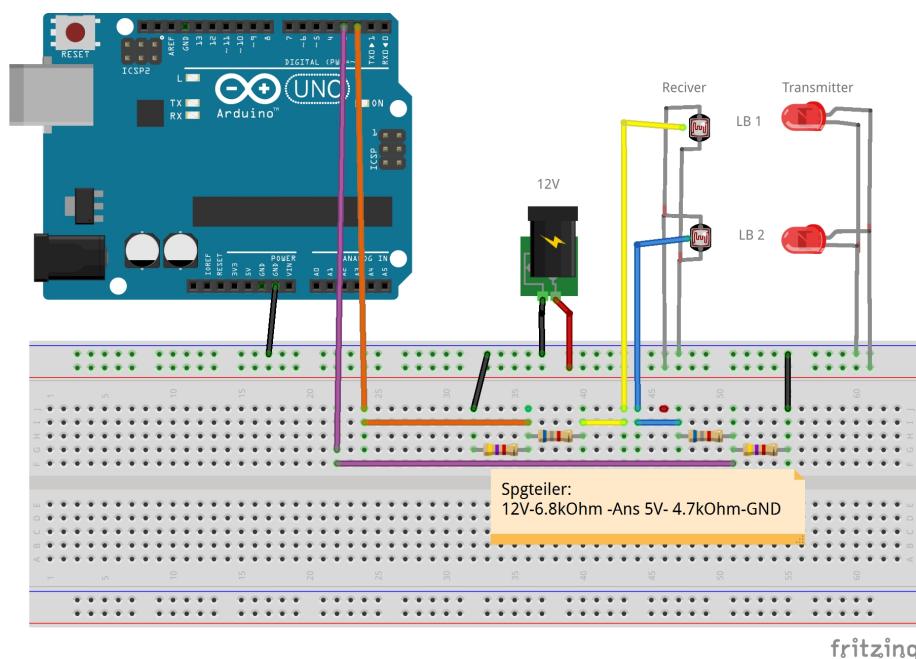


Abb. 2: Anschluss des Arduino Uno

## Teil II

# Software

Softwareseitig wurde mit dem Plugin PlatformIO für Atom die Programmierung des Arduinos erstellt. Für Python wurde die Spyder IDE verwendet. Zur Versionierung wird Git verwendet. Die Auswertung der Daten erfolgt mit Matlab und Excel.

## 3 | Programm

Das Programm *BWMvelocity.cpp* welches man in Anhang C findet, ermöglicht es mit Hilfe von Polling oder Interrupts die Unterbrechungszeiten der Lichtschranke zu erfassen und so auf Grund der Grösse des unterbrechenden Objekts oder auf Grund des Abstands der Lichtschranken auf die Geschwindigkeit zu schliessen.

$$v = \frac{s}{\Delta t} \quad (\text{Formel 1})$$

Parameter	Bemerkung	Einheit
$v$	Geschwindigkeit	[m/s]
$s$	Strecke	[m]
$\Delta t$	Zeit	[s]

Tab. 1: Übersicht der Parameter in Formel 1

Ein Druchlauf eines Objektes erzeugt dabei die in Abb. 3 gezeigten Signale.

Sensor-Signale

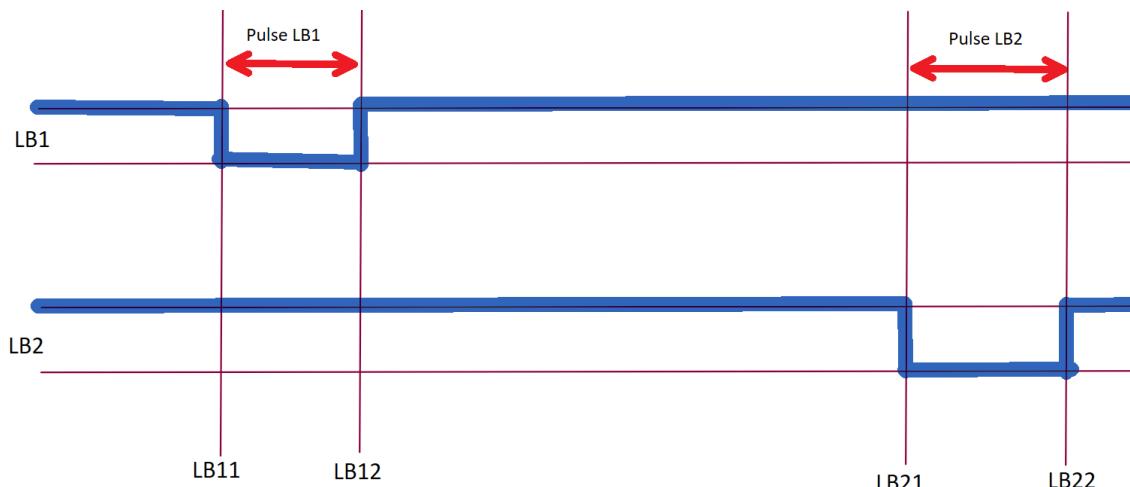


Abb. 3: Generierte Signale der Lichtschranke beim Durchflug eines Objekts abhängig der Zeit

## 3.1 | Modi

Mit Hilfe der #define Anweisung können drei verschiedene Modi ausgewählt werden. **Preprocessor** Dafür müssen die entsprechenden Befehle ein-kommentiert werden.

#define MYDEBUG	Aktiviert den Debuggmodus und somit die <i>DEBUG_PRINT</i> -Funktionen.
#define MYLOG	Aktiviert den Datenloggingmodus und somit die <i>LOG_PRINT</i> -Funktionen.
#define USEINTERRUPT	Aktiviert den Interruptmodus und deaktiviert den Pollingmodus.

Der Debugg- und Logging-Modus ist nur für die Auswertung und/oder Fehlersuche über die serielle Schnittstelle relevant. Im Normalbetrieb (Polling- oder Interrupt-Modus) können sie auskommentiert bleiben.

## 3.2 | Globale Variablen

Es gibt folgende globale Variablen:

- const double sensordistance = 101.0\*1000
- const double ballwidth = 72.0\*1000
- unsigned long passingtime[4]
- double velocitykmh[4]
- int countingvar

**globale  
Variablen**

*sensordistance* und *ballwidth* beschreiben die geometrischen Randbedingungen in  $\mu m$ .

*passingtime* beinhaltet die Zeitstempel in  $\mu s$  und *velocitykmh* beinhaltet die gemessenen Geschwindigkeiten in  $km/h$ .

Bei *countingvar* handelt es sich um eine Lauf-variabel zur Indexierung der SerialPrint-Ausgaben.

## 3.3 | Hilfsfunktionen

### 3.3.1 | calculate\_velocity\_ms()

Geschwindigkeit [ $m/s$ ] = calculate\_velocity\_ms( Zeit [ $\mu S$ ], Distanz [ $\mu m$ ])

Die Funktion *double calculate\_velocity\_ms(unsigned long passingduration,double distance)* übernimmt eine Zeitdauer in  $\mu S$  und eine Distanz in  $\mu m$ . Sie giebt eine Geschwindigkeit mit Hilfe der Formel 1 in  $m/s$  zurück vom Typ double.

### 3.3.2 | printlog()

Wenn der Datenlogging-Modus aktiv ist printet die Funktion in die serielle Schnittstelle.  
Im Pollingmodus:

Iteration Bezeichnung Puls LB1 Puls LB2

Im Interruptmodus:

Iteration Bezeichnung Int. LB1 Int. LB2 Int. LB11 - LB21 Int. LB12 - LB22

## 3.4 | Polling-Modus

Im Polling-Modus wird mit der Funktion `pulseIn(pin, value, timeout)` die Länge des Unterbruches in  $\mu\text{s}$  angegeben. Die minimale detektierbare Pulslänge ist dabei  $10 \mu\text{s}$ . Es werden folgende Zeiten erfasst und im Array `passingtime` abgelegt:

Index	Zeit [ $\mu\text{s}$ ]	relevante Distanz
0	Pulse LB1	Objektgrösse
1	Pulse LB2	Objektgrösse

Die Werte im Index 2 und 3 sind im Pollingmodus immer 0.

Mit der Funktion `calculate_velocity_ms()`(Abschnitt 3.3.1) kann die Geschwindigkeit in  $\text{m/s}$  berechnet werden. Danach wird das Resultat mit dem Faktor 3.6 multipliziert und im Array `velocitykmh` abgelegt.

Die Werte im Index 2 und 3 sind im Pollingmodus immer 0.

## 3.5 | Interrupt-Modus

Im Interrupt-Modus wird bei jedem Flankenwechsel ein Interrupt ausgelöst und ein Zeitstempel in  $\mu\text{s}$  gespeichert. **Interrupt**

**Beim Arduino UNO müssen die Lichtschranken dafür zwingend auf Pin 2 und 3 angeschlossen sein. Weiter muss zuerst die Lichtschranke 1 danach Lichtschranke 2 ausgelöst werden.** Dies liegt daran, dass beim Durchlauf von LB22 das Flag `LBinterrupted` gesetzt wird. Werden die Lichtschranken vertauscht, erfolgt die Auslösung an der Position LB12 (vgl. Abb. 3)

Dies ergibt gesamt 4 Messwerte für LB11, LB12, LB21 und LB22 wie in Abb. 3 gezeigt. Diese werden im Array `passingtime` gespeichert.

Das Array `passingtime` beinhaltet also im Interruptmodus die folgenden Einträge:

Index	Zeit [ $\mu\text{s}$ ]
0	LB11
1	LB12
2	LB21
3	LB22

Mit der Funktion `calculate_velocity_ms()`(3.3.1) kann die Geschwindigkeit in  $\text{m/s}$  berechnet werden. Danach wird das Resultat mit dem Faktor 3.6 multipliziert und im Array `velocitykmh` abgelegt.

Index	Zeit [ $\mu\text{s}$ ]	relevante Distanz [ $\mu\text{m}$ ]
0	LB12 - LB11	Objektgrösse
1	LB22 - LB21	Objektgrösse
2	LB21 - LB11	Sensorabstand
3	LB22 - LB12	Sensorabstand

### 3.5.1 | Hilfsfunktionen

Der Interruptmodus benötigt noch weitere Hilfsfunktionen, nämlich die Interrupt-Service-Routine. Weiter wird noch das Flag *LBinterrupted* benötigt, welches auf True wechselt, **Interrupt-Flag** sobald das Objekt den Punkt LB22 durchlaufen hat.

#### 3.5.1.1 | ISRLB1()

Diese Funktion ist die Interrupt-Service-Routine für die Lichtschranke LB1. Sie erkennt den Signalwechsel und schreibt den aktuellen Zeitstempel LB11 oder LB12 je nach dem ob das Signal HIGH oder LOW ist. Weiter wird das Flag *LBinterrupted* auf False gesetzt.

#### 3.5.1.2 | ISRLB2()

Diese Funktion ist die Interrupt-Service-Routine für die Lichtschranke LB1. Sie erkennt den Signalwechsel und schreibt den aktuellen Zeitstempel LB11 oder LB12 je nach dem ob das Signal HIGH oder LOW ist. Bei LB21 wird das Flag *LBinterrupted* auf False gesetzt. Bei LB22 auf True.

## 3.6 | Hauptfunktionen

### 3.6.1 | setup()

Die *setup()*-Funktion läuft nur ein mal beim Einschalten oder dem Reset des Arduinos durch. Sie initialisiert die serielle Kommunikation und erstellt die Interrupt-Zuweisung der Pins im Interrupt-Modus. Ausserdem erstellt sie die erste Zeile des Log-Files im Logging-Modus.

### 3.6.2 | loop()

Die *loop()*-Funktion wird nach der *setup()*-Funktion dauernd ausgeführt. In ihr läuft das Hauptprogramm. Je nach aktivem Modus entweder im Polling- oder im Interrupt-Modus.

# 4 | Python

Das Pythonskript aus Anhang D dient dazu, die gesendeten Daten über den Serialport auszulesen und in einem .txt File abzuspeichern. Dies erleichtert die Datenanalyse mit Excel oder Matlab. Eine direkte Auswertung in Phyton wäre ebenfalls möglich.

## 4.1 | Randbedingungen

Die Baudrate im Python-Skript (Zeile 29) muss die selbe sein wie im Arduino-Code angegeben.

Der COM-Port sollte unter Windows automatisch gefunden werden. Ist dies nicht der Fall müssen die Zeilen 31 bis 38 angepasst werden. Das Argument *serial\_port* kann dabei einfach mit dem gewünschten Port ersetzt werden.

## 4.2 | Messdaten

Der Name des Ausgabefiles kann in Zeile 48 beliebig angepasst werden.

## 4.3 | Ablauf

Wird das Skript gestartet erstellt es ein \*.txt-File und schreibt eine Kopfzeile. Danach wird alles, was über den gewählten Serialport empfangen wird in dieses .txt-File gespeichert. Sobald ein *KeyboardInterrupt* erzeugt wird, beispielsweise mit Ctrl + C oder dem Abbruch-Knopf wird das File gespeichert und das Skript beendet.

## Teil III

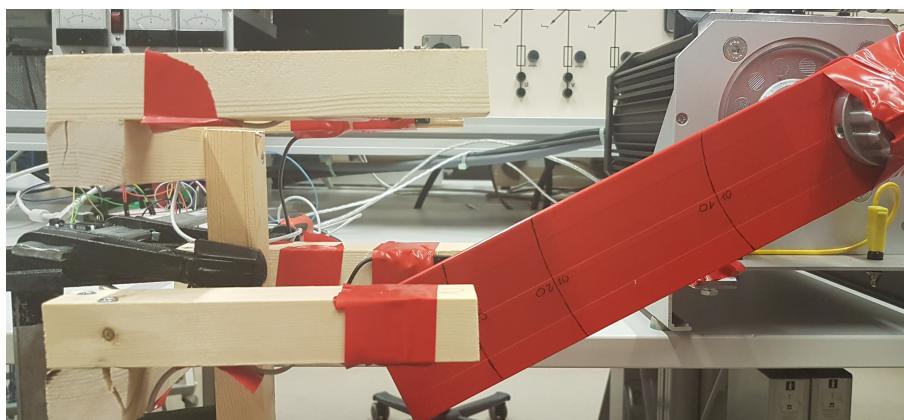
# Versuch

## 5 | Aufbau

Der Versuchsaufbau wurde im Zimmer 2.001b mit einem Servomotor aus den Praktikas durchgeführt. Er kann zwischen -2500 bis +2500  $U/min$  betrieben werden und im *Synchronisation Mode* auf 1  $U/min$  genau eingestellt werden. Im *Speed-Control Mode* liegt die Schrittweite bei 10  $U/min$ .



(a) Übersicht



(b) Detail

*Abb. 4: Versuchsaufbau*

Die Lichtschranken wurden mit Hilfe einer Holzkonstruktion montiert.

Sie haben einen Abstand von 100 mm.

Der Rotor ist 72 mm breit und somit die Dimension eines Unihockeyballs.

**Rotor**

Bei einem Durchmesser von 50 cm ergibt sich mit 2000  $U/min$  eine Umfangsgeschwindigkeit von 188.5  $km/h$  wie man dem Anhang B entnehmen kann.

# 6 | Auswertung

Mit dem Aufbau ist es möglich Geschwindigkeiten von über 200 km/h zu erzeugen.

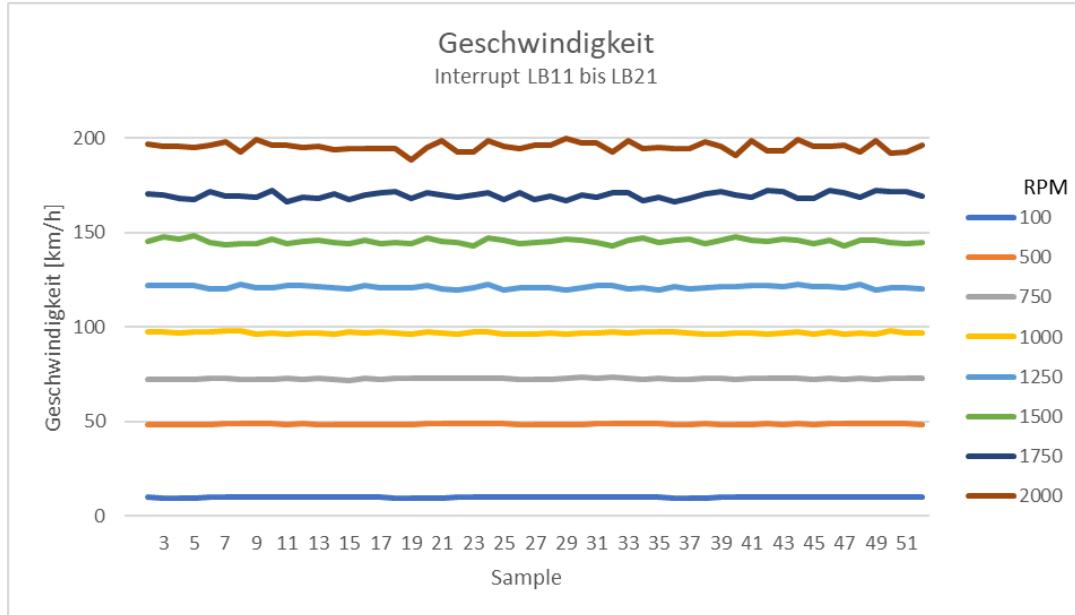


Abb. 5: Vergleich unterschiedliche Geschwindigkeiten

In Abb. 5 werden die verschiedenen gemessenen Geschwindigkeit abhängig von der Drehzahl dargestellt. Erwartet sind folgende Geschwindigkeiten:

RPM	Geschwindigkeit [km/h]
100	9.4
500	47.1
750	70.7
1000	94.2
1250	117.8
1500	141.4
1750	164.9
2000	188.5

Die Präzision nimmt mit höherer Geschwindigkeit immer weiter ab, während die Genauigkeit etwa gleich bleibt (vgl. Abb. 5, Abb. 6).

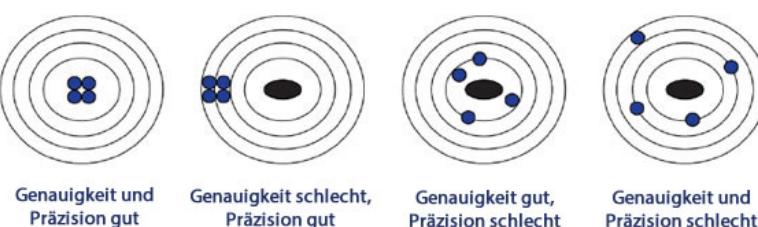
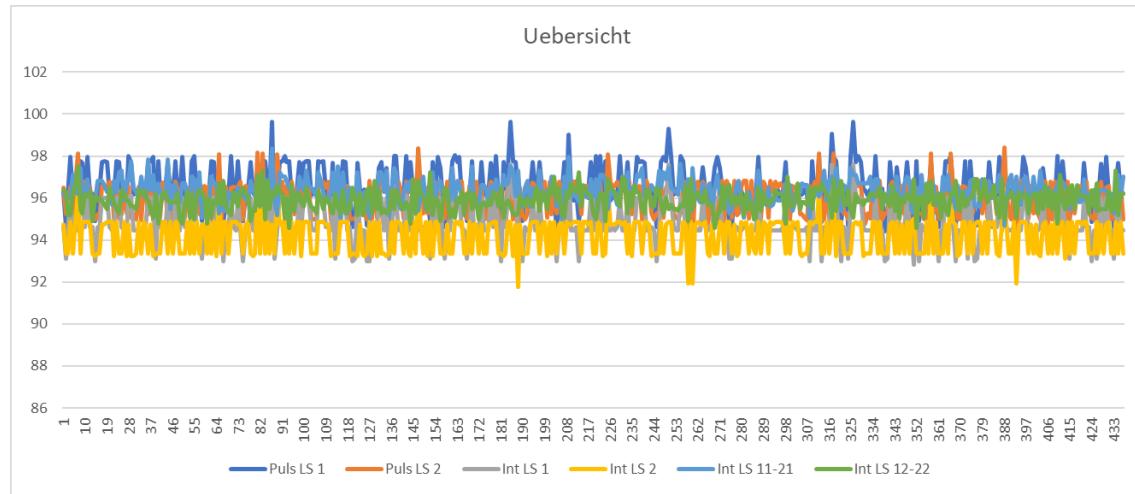


Abb. 6: Genauigkeit und Präzision

In Abb. 7 werden die verschiedene Implementationsmethoden verglichen.

### Vergleich Polling - Interrupt

Die Daten für Puls LS1 und Puls LS2 wurden durch Polling gewonnen während die restlichen durch Interrupts generiert wurden. Detailliertere Grafiken dazu finden sich in Anhang E.



*Abb. 7: Vergleich Interrupt und Polling bei 1000 RPM (94.2 km/h)*

Bei 2000 RPM wurden in beiden Modi jeweils 500 Sample aufgezeichnet und ausgewertet. Dabei wurde eine Präzision von unter 2% und eine Genauigkeit von unter 5% erreicht. Die Histogramme aus Abb. 9 wurde mit Hilfe eines Matlabskripts (Anhang E.1) erstellt und ausgewertet.

Berechnung Genauigkeit [%]:

### Genauigkeit

$$\frac{|Sollgeschwindigkeit - Mittelwert der Samples|}{Sollgeschwindigkeit} \cdot 100$$

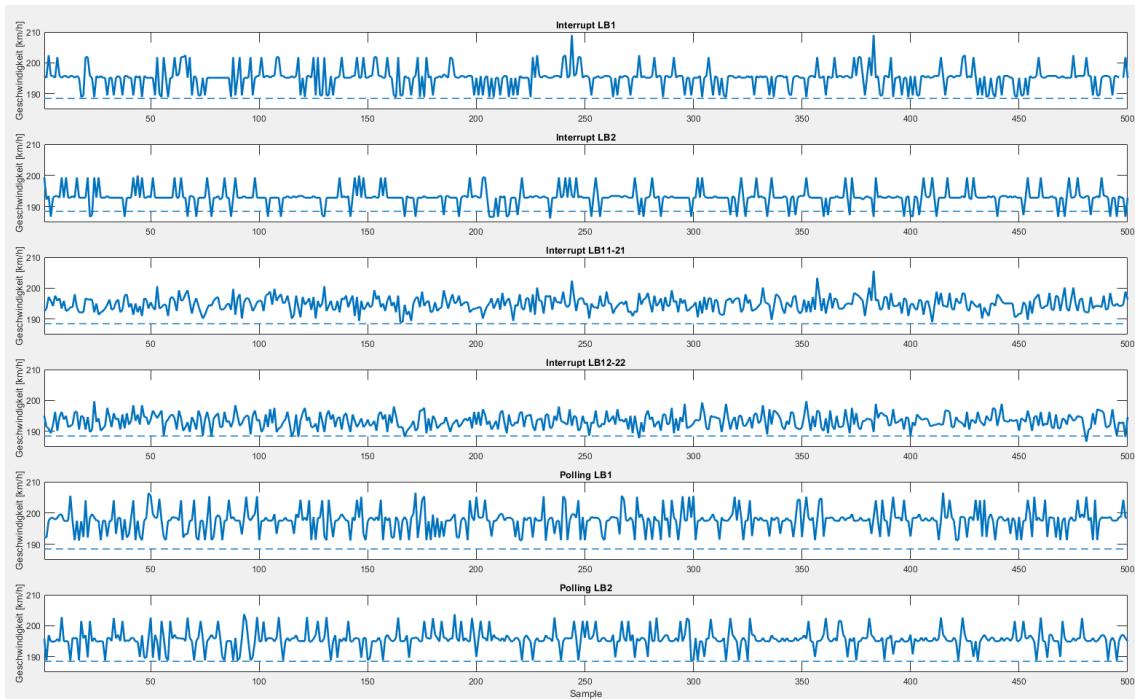
Berechnung Präzision [%]:

### Präzision

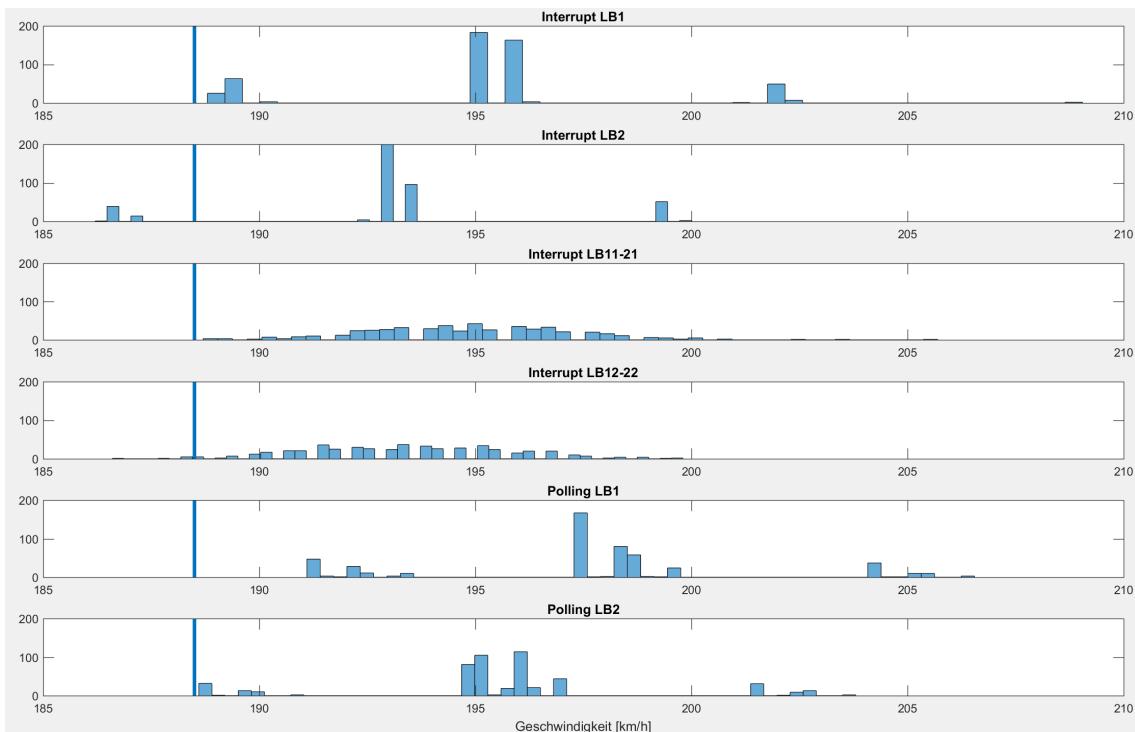
$$\frac{\text{Standardabweichung der Samples}}{\text{Mittelwert der Samples}} \cdot 100$$

Mode	Messung	Genauigkeit [%]	Präzision [%]
Polling	LB1	4.9	1.9
	LB2	3.8	1.6
Interrupt	LB11 - LB12	3.5	1.8
	LB21 - LB22	2.4	1.5
	LB11 - LB21	3.4	1.2
	LB12 - LB22	2.6	1.2

## Auswertung



*Abb. 8: Gemessene Werte bei 2000 RPM*



*Abb. 9: Histogramm*

## 6.1 Fehlerquellen

Die Abb. 10 zeigt den maximalen Geschwindigkeitsfehler wenn mit der maximalen Ansprechzeit der Lichtschranke von 0.5 ms gerechnet wird. Diese Abbildung soll vor allem verdeutlichen, dass Zeitfehler bei hohen Geschwindigkeiten stärker Gewichtet werden.

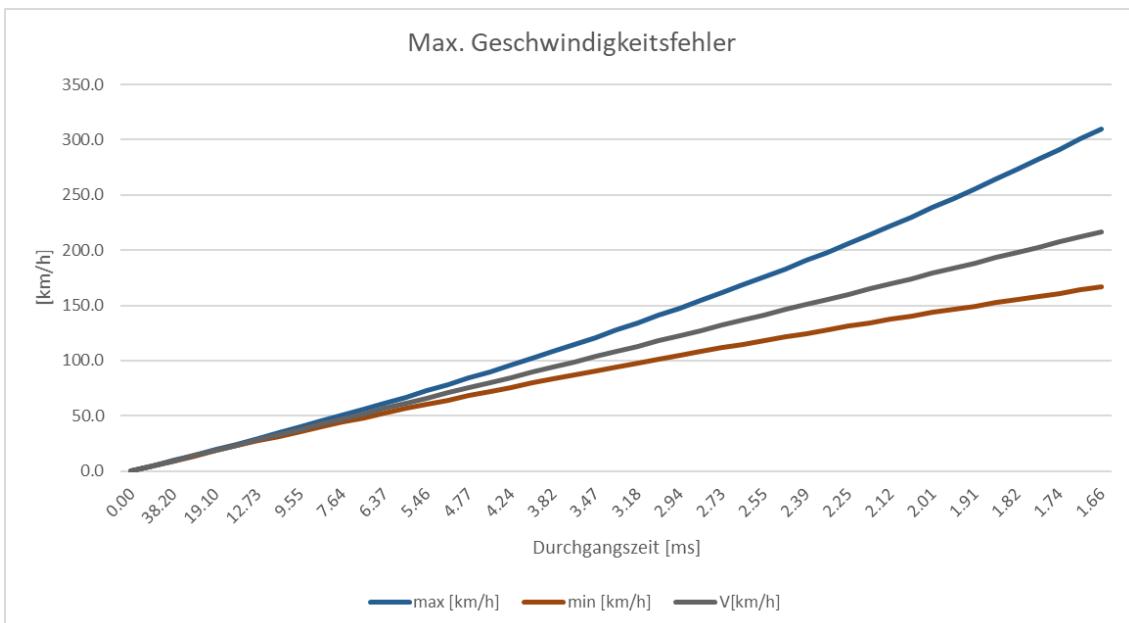


Abb. 10: Max. Geschwindigkeitsfehler maximaler Ansprechverzögerung.

Die für diesen Versuchsaufbau relevanten Fehlerquellen sind im Polling-Modus:

### Fehlerquellen

- Ungenauer Abstand der Lichtschranken zum Drehpunkt des Rotors.
- Ungenauigkeit infolge unterschiedlicher Reaktionszeiten der Lichtschranke.
  - Wiederholpräzision
  - Hysterese
- Ungenauigkeit infolge Regelabweichung des Motors.
- Ungenauigkeit infolge der Reaktionszeit des Arduinos.

Beim Interrupt-Modus kommt wegen dem Vergleich zwischen den Lichtschranken noch weitere Ungenauigkeiten dazu:

- Ungenauer Abstand der Lichtschranken zueinander.

Die Ungenauigkeit infolge unterschiedlicher Reaktionszeiten der beiden Lichtschranken kann evtl. mit dem Aufsetzen einer Schlitzmaske, welche es als Zubehör gibt, auf kosten der Reichweite, noch verbessert werden. Dies würde sich positiv auf die Präzision auswirken. Für eine Verbesserung der Genauigkeit ist vor allem ein geometrisch genauerer Aufbau nötig.

Ein weiteres Problem ist, dass auf Grund des rotierenden Aufbaus nicht abschliessend gesagt werden kann, ob alle Durchläufe korrekt detektiert werden. Einzelne Durchläufe könnten nicht detektiert worden sein. Mit diesem Versuchsaufbau ist es jedoch nicht möglich einen einzelnen schnellen Durchgang zu erzeugen.

## Teil IV

## Fazit

# 7 | Empfehlung

Einige Fehler könnten mit einem genaueren Messaufbau eliminiert werden. Die Genauigkeit von 5% sowie eine Präzision von 2% ist in den Augen des Verfassers ausreichend. Eine Verbesserung kann erzielt werden wenn die momentane Ausführung mit 2 Sensoren beibehalten wird und die erfassten Messwerte noch weiterverarbeitet werden. Vorstellbar wäre dabei die Bildung eines Medians um Ausreisser zu minimieren.

Jedoch zeigte der Vergleich der Implementationsmethoden, dass auch nur eine Lichtschranke ausreichend sein kann. Es empfiehlt sich, dies weiter zu untersuchen.

Für die genauere Identifikation der Fehlerquellen sind jedoch weitere Abklärungen und Experimente nötig.

Der Verfasser empfiehlt die Nutzung des Interrupt-Modes, da er die bessere Präzision und Genauigkeit aufweist. Außerdem erlaubt der Interrupt-Modus das Ausführen anderer Programme bis ein Event auftritt, während im Polling-Modus aktiv gewartet werden muss.

Weiter könnte der Code in ein Definitions- (\*.cpp) und Deklarations-File (\*.h) aufgeteilt werden. So kann er einfacher in bestehende Programme integriert werden.

# Anhang

<b>A Datenblatt Panasonic EX-20</b>	<b>18</b>
<b>B Berechnungen</b>	<b>24</b>
<b>C Arduinoprogramm</b>	<b>29</b>
<b>D Python</b>	<b>33</b>
<b>E Auswertung</b>	<b>35</b>
E.1 Matlab-skript . . . . .	35
E.2 Plots . . . . .	40

# A | Datenblatt Panasonic EX-20

## Panasonic® BETRIEBSANLEITUNG

Optosensor

Ultra-kompakter Optosensor

### Serie EX-20

MEUDE-EX20 V2.0

Danke, dass Sie sich für ein Produkt von Panasonic Electric Works SUNX Co., Ltd. entschieden haben. Bitte lesen Sie diese Bedienungsanleitung für die bestimmungsgemäße Verwendung dieses Produkts sorgfältig durch. Heben Sie diese Bedienungsanleitung zum Nachlesen griffbereit auf.

#### ⚠️ WARNUNG

- Benutzen Sie dieses Produkt nicht zum Schutz von Personen.
- Wenn Sie Sensorelemente zum Schutz von Personen verwenden, sollten Sie Produkte benutzen, die den jeweiligen Landesgesetzen und Standards entsprechen, wie etwa OSHA, ANSI oder IEC.

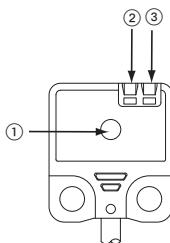
## 1 VORSICHTSMASSNAHMEN

- Dieses Produkt wurde ausschließlich zur industriellen Verwendung entwickelt/hergestellt.
- Für das Produkt wird ein dünnes, 0,1mm<sup>2</sup> starkes Kabel verwendet. Beim Herausziehen des Kabels keine übermäßige Kraft aufwenden: Es kann sonst zu einem Kabelbruch kommen.
- Der Sensortyp EX-24□(-PN) ist nicht mit einem Empfindlichkeitspotenziometer ausgestattet. Es sollte ein angemessener Abstand von reflektierenden Objekten im Hintergrund eingehalten werden, z.B. von Förderbändern etc., da diese die Erkennung beeinträchtigen können.
- Wenn im Hintergrund ein reflektierendes Objekt vorhanden ist, kann die Erkennungsleistung des X-28□A(-PN) beeinträchtigt werden. Prüfen Sie bei der Einrichtung des Sensors, dass das reflektierende Objekt keine Beeinträchtigung hervorruft. Wenn das reflektierende Objekt die Erkennung beeinträchtigt, entfernen Sie es, schwärzen Sie es oder ergreifen Sie andere Gegenmaßnahmen.
- Wenn die Sensoren nah aneinander montiert sind, und die Umgebungstemperatur nahe dem maximalen Nennwert ist, sorgen Sie für eine ausreichende Wärmeabstrahlung und Ventilation.
- Die Verdrahtung muss bei ausgeschalteter Spannungsversorgung erfolgen.
- Falsche Verdrahtungen können den Sensor beschädigen.
- Die Spannungsversorgung muss innerhalb der angegebenen Werte inklusive Restwelligkeit liegen. Beachten Sie, dass die Betriebsspannung innerhalb der angegebenen Werte liegt.
- Wird der Strom von einem handelsüblichen Schaltregler bereitgestellt, stellen Sie sicher, dass die Geräteerde (F.G.) der Spannungsversorgung an eine Schutzerde angeschlossen ist.
- Falls elektrische Bauteile (Schaltregler, Frequenzumrichter, etc.) in der Nähe des Produkts verwendet werden, die Störstrahlungen erzeugen, müssen Sie den Erdungsanschluss der Bauteile an eine vorhandene Schutzerde anschließen.
- Verlegen Sie die Kabel nicht zusammen mit Starkstromkabeln oder Hochspannungsleitungen in demselben Kabelkanal. Dies kann zu Fehlfunktionen führen.
- Das Kabel für den Sender und Empfänger der Einweg-Lichtschranke mit einer Stärke von 0,3mm<sup>2</sup> oder mehr lässt sich bis zu max. 50m verlängern. Um Störstrahlungen zu vermeiden, sollte das Kabel jedoch möglichst kurz gehalten werden.
- Während des Initialisierungsvorgangs (0,5s nach dem Einschalten der Spannungsversorgung) dürfen keine Einstellungen erfolgen.
- Bei starken äußeren Lichteinflüssen, kann die Objekterkennung behindert werden. Stellen Sie deshalb sicher, dass der Sensor nicht direkt einer der folgenden Lichtquellen ausgesetzt ist: fluoreszierendem Licht von Leuchtstoffröhren mit Schnellstarter, Hochfrequenz-Licht, Sonnenlicht etc.
- Beanspruchen Sie die Kabelverbindungsstelle des Sensors nicht durch gewaltsames Verbiegen oder Ziehen.

- Dieser Sensor darf nur in Innenräumen verwendet werden.
- Vermeiden Sie Staub, Schmutz und Dampf. Montageorte mit übermäßig viel Dampf, Staub, etc. sind ungeeignet. Vermeiden Sie auch, dass der Sensor korrodierenden Dämpfen ausgesetzt wird.
- Der Sensor darf nicht mit Wasser, Öl, Fett oder organischen Lösungsmitteln, wie Verdünner, in Berührung kommen.

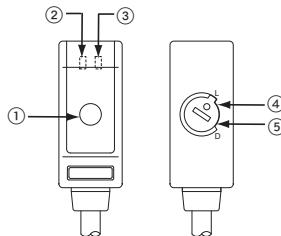
## 2 ANZEIGE- UND BEDIENELEMENTE

EX-21□



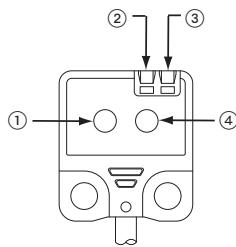
Nr.	Element	Beschreibung
1	Strahlachse	
2	Stabilitätsanzeige (grün)	<b>Nur Empfänger.</b> Leuchtet, wenn die Erkennung entsprechend den eingestellten Parametern stabil ist.
3	Betriebsanzeige (orange)	<b>Nur Empfänger.</b> Leuchtet, wenn der Ausgang aktiv ist.

EX-23□



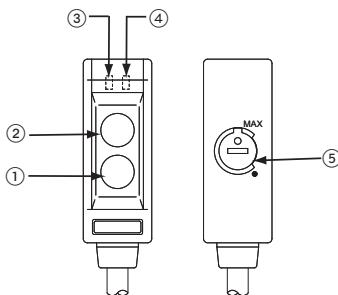
Nr.	Element	Beschreibung
1	Strahlachse	
2	Stabilitätsanzeige (grün)	<b>Nur Empfänger.</b> Leuchtet, wenn die Erkennung entsprechend den eingestellten Parametern stabil ist.
3	Betriebsanzeige (orange)	<b>Nur Empfänger.</b> Leuchtet, wenn der Ausgang aktiv ist.
4	Hell-Dunkel-Schalter	<b>Nur Empfänger.</b> <ul style="list-style-type: none"> <li>• L: Hell-EIN Hell-Dunkel-Schalter im Uhrzeigersinn bis zum Anschlag drehen.</li> <li>• D: Dunkel-EIN Hell-Dunkel-Schalter entgegen dem Uhrzeigersinn bis zum Anschlag drehen.</li> </ul>
5	Empfindlichkeitspotenziometer	<b>Nur Sender.</b> Drehen im Uhrzeigersinn erhöht die Reichweite. Siehe "EMPFINDLICHKEIT ANPASSEN" auf Seite 2.

#### EX-24□



Nr.	Element	Beschreibung
1	Empfänger	
2	Stabilitätsanzeige (grün)	Leuchtet, wenn die Erkennung entsprechend den eingestellten Parametern stabil ist.
3	Betriebsanzeige (orange)	Leuchtet, wenn der Ausgang aktiv ist.
4	Sender	

#### EX-22□, EX-26□, EX-28□, EX-29□



Nr.	Element	Beschreibung
1	Sender	
2	Empfänger	
3	Stabilitätsanzeige (grün)	Leuchtet, wenn die Erkennung entsprechend den eingestellten Parametern stabil ist.
4	Betriebsanzeige (orange)	Leuchtet, wenn der Ausgang aktiv ist.
5	Empfindlichkeitspotenziometer	Drehen im Uhrzeigersinn erhöht die Reichweite. Siehe "EMPFINDLICHKEIT ANPASSEN" auf Seite 2.

### 3 EMPFINDLICHKEIT ANPASSEN

Zur Anpassung der Empfindlichkeit ist es wichtig, den Unterschied zwischen dem Status "Hell" und dem Status "Dunkel" zu verstehen. Verwechseln Sie die Status "Hell" und "Dunkel" nicht mit der Betriebsanzeige "Hell-EIN" und "Dunkel-EIN"!

Sensortyp	Status "Hell"	Status "Dunkel"
Einweg-Lichtschranke	Sender → Empfänger	Sender → Empfänger Zu erkennendes Objekt
Reflexions-Lichtschranke	Sensor ← Reflektor	Sensor → Reflektor Zu erkennendes Objekt
Standard-Lichttaster	Sensor ← Zu erkennendes Objekt	Sensor →
Konvergenter Lichttaster, Typ mit kleinem Lichtfleck	Sensor ← Zu erkennendes Objekt	Sensor →

#### Verhältnis zwischen Ausgang und Anzeigen

Hell-EIN			Dunkel-EIN			
Stabilitätsanzeige	Betriebsanzeige	Ausgang	Messbedingung	Ausgang	Betriebsanzeige	Stabilitätsanzeige
○	○	EIN	Stabiles Licht	AUS	●	○
●			Instabiles Licht			●
	●	AUS	Instabile Dunkelheit	EIN	○	
			Stabile Dunkelheit			○

○ = LED leuchtet, ● = LED leuchtet nicht

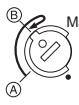
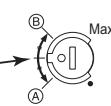
#### Vorgehensweise

Diese Vorgehensweise setzt voraus, dass der Betriebsmodus "Hell-EIN" aktiv ist. Wenn der Betriebsmodus "Dunkel-EIN" aktiv ist, verhält sich der Ausgang umgekehrt!

☞ Verwenden Sie den mitgelieferten Schraubendreher und drehen Sie den Potenziometer vorsichtig. Bei zu großer Krafteinwirkung kann er beschädigt werden.

☞ Wenn der EX-22□(-PN) für eine Reichweite von max. 50mm eingesetzt wird, ist der Bereich für die Anpassung der Empfindlichkeit extrem klein.

Schritt	Empfindlichkeitspotenziometer	Beschreibung
1	Max	Empfindlichkeitspotenziometer entgegen dem Uhrzeigersinn bis zum Anschlag in die Position der geringsten Empfindlichkeit drehen (●).
2	Max	Im Status "Hell" drehen Sie den Empfindlichkeitspotenziometer langsam im Uhrzeigersinn bis Punkt A gefunden ist, an dem der Ausgang auf EIN schaltet.*1

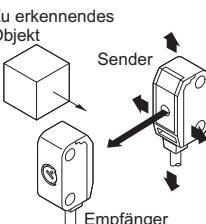
Schritt	Empfindlichkeitspotenziometer	Beschreibung
3		Im Status "Dunkel" den Empfindlichkeitspotenziometer langsam im Uhrzeigersinn drehen, bis der Ausgang auf EIN schaltet.* <sup>1</sup> Dann langsam zurück bis Punkt B drehen, d.h. jenem Punkt, an dem der Ausgang auf AUS schaltet.* <sup>1</sup> Wenn der Sensor nicht EIN schaltet, obwohl der Potenziometer im Uhrzeigersinn bis zum Anschlag gedreht ist, dann entspricht Punkt B der Position MAX.
4		Die Position, die genau zwischen den ermittelten Punkten A und B liegt, ist die optimale Erkennungsposition.

\*<sup>1</sup> Beachten Sie, dass dies nur für den Betriebsmodus Hell-EIN gilt.

## 4 STRAHLAUSRICHTUNG

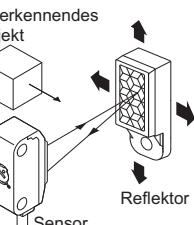
### Einweg-Lichtschanke

- ① Beim EX-23(-PN) den Hell-/Dunkel-Schalter in die Position (L) drehen, um den Modus Hell-EIN zu aktivieren.
- ② Sender und Empfänger so platzieren, dass sie in einer geraden Linie gegenüber liegen. Sender nach oben, unten, links und rechts bewegen, um mit Hilfe der Betriebsanzeige (orange) des Empfängers festzustellen, in welcher Position das Licht empfangen wird. Sender in der Mitte des Empfangsbereichs einrichten.
- ③ Winkel des Senders anpassen. Dazu Sender nach oben, unten, links und rechts drehen.
- ④ Ausrichtungswinkel des Empfängers ebenso anpassen.
- ⑤ Sicherstellen, dass die Stabilitätsanzeige grün leuchtet.
- ⑥ Beim EX-23(-PN) den benötigten Betriebsmodus (Hell-EIN oder Dunkel-EIN) mit dem Hell-Dunkel-Schalter auswählen.



### Reflexions-Lichtschanke

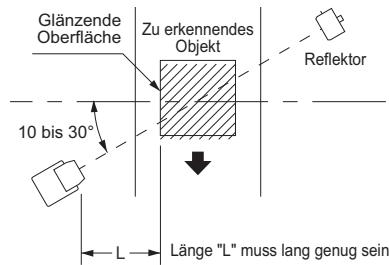
- ① Empfindlichkeitspotenziometer im Uhrzeigersinn bis zum Anschlag in die Position der höchsten Empfindlichkeit (MAX) drehen.
- ② Sensor und Reflektor so platzieren, dass sie in einer geraden Linie gegenüber liegen. Reflektor nach oben, unten, links und rechts bewegen, um mit Hilfe der Betriebsanzeige (orange) festzustellen, in welcher Position das Licht empfangen wird. Reflektor in der Mitte des Empfangsbereichs einrichten.
- ③ Ausrichtungswinkel des Reflektors anpassen, indem Sie ihn nach oben, unten, links und rechts drehen.
- ④ Ausrichtungswinkel des Sensors ebenso anpassen.
- ⑤ Sicherstellen, dass die Stabilitätsanzeige grün leuchtet.



## 5 REFLEXIONS-LICHTSCHANKE

Prüfen Sie bei der Einrichtung des EX-29□(-PN) folgende Punkte, wenn Objekte aus glänzenden Materialien erkannt werden sollen.

- Der im Diagramm als "L" bezeichnete Abstand sollte lang genug sein.
- Sensor in einem Winkel von 10 bis 30 Grad zum Objekt installieren.

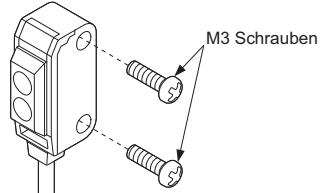


## 6 MONTAGE

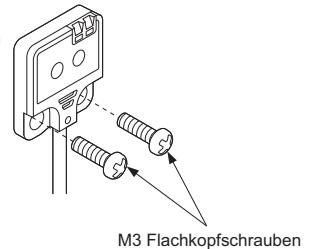
Sensor mit M3-Schrauben montieren. Verwenden Sie beim Fronttyp die M3 Flachkopfschrauben ohne Beilagscheiben.

- Das Anzugsdrehmoment sollte maximal 0,5N·m sein.

Seiten-Typ



Front-Typ

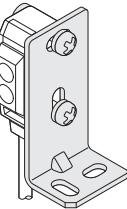
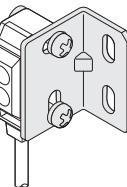
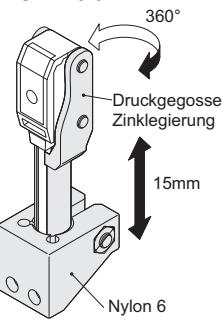


### Montagewinkel

Die Montagewinkel sind als Zubehör erhältlich.

- Das Anzugsdrehmoment sollte maximal 0,5N·m sein.

	Modellnr.	Beschreibung
Fronttyp	<b>MS-EX20-1</b>	Zwei M3-Flachkopfschrauben à 5mm werden mitgeliefert. Material: korrosionsbeständiger Stahl (SUS304).
	<b>MS-EX20-3</b>	

Modellnr.	Beschreibung
<b>MS-EX20-2</b> 	Zwei M3 Schrauben à 14mm mit Beilagscheiben werden mitgeliefert. Material: korrosionsbeständiger Stahl (SUS304).
<b>MS-EX20-4</b> 	
<b>MS-EX20-5</b> 	Für EX-23(-PN): Zwei M3 Schrauben à 12mm (korrosionsbeständiger Stahl), eine M3 Innensechskantschraube und eine M3 Sechskantmutter. Material: korrosionsbeständiger Stahl (SUS304).

## 7 SCHLITZMASKEN

☞ Die Schlitzmaske steht nur für die Einweg-Lichtschranke zur Verfügung.

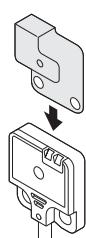
Die als Zubehör erhältlichen Schlitzmasken unterstützen die Erkennung kleiner Objekte. Die Genauigkeit der Positionsbestimmung wird ebenfalls erhöht. Die Reichweite wird jedoch reduziert.

Sensortyp	Schlitzmaske Modellnr.	Beschreibung
EX-21□	<b>OS-EX20-05</b>	Schlitz-Durchmesser: 0,5mm.
	<b>OS-EX20-05X3</b>	Schlitz: 0,5 x 3,0mm.
EX-23□	<b>OS-EX20E-05</b>	Schlitz-Durchmesser: 0,5mm.
	<b>OS-EX20E-05X3</b>	Schlitz: 0,5 x 3,0mm.

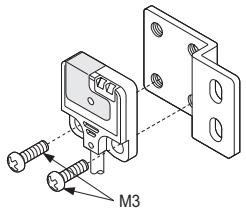
### Montage

☞ Das Anzugsdrehmoment sollte maximal 0,5N•m sein.

- Schlitzmaske über den Sensor legen.



- Löcher aneinander ausrichten und das Distanzstück mit den mitgelieferten M3-Schrauben befestigen.



## 8 DISTANZSTÜCK MONTIEREN

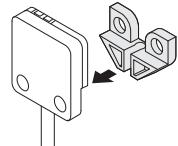
Das als Zubehör erhältliche Distanzstück steht nur für den Fronttyp zur Auswahl.

**Bauteilnr.:** MS-EX20-FS.

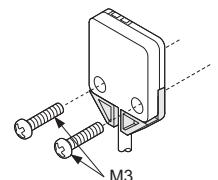
### Montage

☞ Das Anzugsdrehmoment sollte maximal 0,5N•m sein.

- Distanzstück an den Sensor anlegen.



- Löcher aneinander ausrichten und das Distanzstück mit den mitgelieferten M3-Schrauben befestigen.

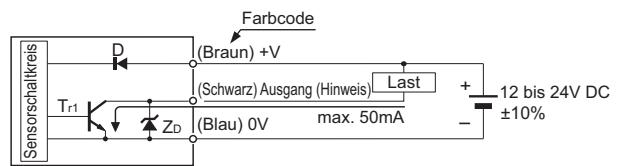


## 9 E/A SCHALTPLÄNE

In diesem Abschnitt werden folgende Symbole verwendet.

Symbol	Bedeutung
D	Verpolungsschutzdiode
ZD	Zenerdiode (Spannungsspitzenenschutz)
Tr1	NPN-Ausgangstransistor
Tr2	PNP-Ausgangstransistor

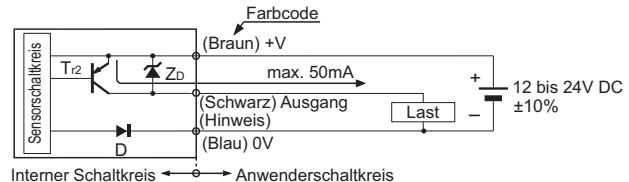
### Typ mit NPN-Ausgang



Interner Schaltkreis → Anwenderschaltkreis

☞ Nur der Empfänger der Einweg-Lichtschranke besitzt einen Ausgang.

### Typ mit PNP-Ausgang



☞ Nur der Empfänger der Einweg-Lichtschranke besitzt einen Ausgang.

## 10 TECHNISCHE DATEN

Typ		Einweg-Lichtschranke		Reflexions-Lichtschranke	Standard-Lichttaster	Konvergenter Lichttaster		Typ mit kleinem Lichtfleck				
		Fronttyp	Seitentyp			Seitentyp	Fronttyp					
Modellnr. <sup>*1</sup>	Hell-EIN	EX-21A(-PN)	EX-23(-PN) <sup>*2</sup>	EX-29A(-PN)	EX-22A(-PN)	EX-24A(-PN)	EX-26A(-PN)	EX-28A(-PN)				
	Dunkel-EIN	EX-21B(-PN)		EX-29B(-PN)	EX-22B(-PN)	EX-24B(-PN)	EX-26B(-PN)	EX-28B(-PN)				
<b>Reichweite</b>		1m	2m	30 bis 300mm <sup>*3</sup>	5 bis 160mm <sup>*4</sup>	2 bis 25mm <sup>*5</sup>	6 bis 14mm <sup>*6</sup>	45 bis 115mm <sup>*7</sup>				
<b>Zu erkennendes Objekt</b>		Mind. Ø2,6mm undurchsichtiges Objekt <sup>*8</sup>	Mind. Ø3mm undurchsichtiges Objekt <sup>*9</sup>	Mind. Ø 15mm undurchsichtig oder halbdurchsichtiges Objekt <sup>*3</sup>	Undurchsichtiges, halbdurchsichtiges oder transparentes Objekt	Mind. Ø0,1mm Kupferdraht <sup>*10</sup>	Mind. Ø0,1mm Kupferdraht <sup>*10</sup>	Undurchsichtiges, halbdurchsichtiges oder transparentes Objekt <sup>*11</sup>				
<b>Hysterese</b>		–		max. 15% der Tastweite								
<b>Wiederholpräzision (bei rechtem Winkel zu Strahlachse)</b>		max. 0,05mm		max. 0,5mm	max. 0,3mm	max. 0,1mm <sup>*10</sup>	max. 0,05mm <sup>*10</sup>	max. 0,3mm				
<b>Betriebsnennspannung</b>		12 bis 24V DC±10%, Restwelligkeit Spitze-Spitze max. 10%										
<b>Stromaufnahme</b>		Sender: max. 10mA Empfänger: max. 10mA	max. 13mA									
<b>Ausgang</b>		<b>EX-□A, EX-□B, EX-23</b> NPN-Transistor mit offenem Kollektor <ul style="list-style-type: none"> <li>Maximale Senke: 50mA</li> <li>Anliegende Spannung: Max. 30V DC (zwischen Ausgang und 0V)</li> <li>Restspannung: max. 1V (bei 50mA Laststrom), max. 0,4V (bei 16mA Laststrom)</li> </ul>			<b>EX-□A-PN, EX-□B-PN, EX-23-PN</b> PNP-Transistor mit offenem Kollektor <ul style="list-style-type: none"> <li>Maximale Quelle: 50mA</li> <li>Anliegende Spannung: Max. 30V DC (zwischen Ausgang und +V)</li> <li>Restspannung: max. 1V (bei 50mA Laststrom), max. 0,4V (bei 16mA Laststrom)</li> </ul>							
	Kurzschluss-schutz	Integriert										
<b>Ansprechzeit</b>		max. 0,5ms										
<b>Betriebsanzeige</b>		Orange LED; leuchtet, wenn Ausgang auf EIN schaltet. Einweg-Lichtschranke: befindet sich am Empfänger.										
<b>Stabilitätsanzeige</b>		Grüne LED leuchtet bei stabilem Lichtempfang oder stabiler Dunkelheit. Einweg-Lichtschranke: befindet sich am Empfänger.										
<b>Empfindlichkeitspotenzimeter</b>		–	Stufenlos regelbarer Poti	Stufenlos regelbarer Poti	–	Stufenlos regelbarer Poti						
<b>Schutzart</b>		IP67 (IEC)										
<b>Umgebungstemperatur</b>		-25 bis +55°C (Kondensation oder Eisbildung ist nicht zulässig), Lagerung: -30 bis +70°C										
<b>Luftfeuchtigkeit</b>		35 bis 85% relative Feuchte, Lagerung: 35 bis 85% relative Feuchte										
<b>Sendediode</b>		Rote LED (moduliert)										
<b>Material</b>		Gehäuse: PET, Linsen: Polyalylat										
<b>Kabel</b>		0,1mm <sup>2</sup> 3-adriges Kabel mit Kappe (Sender der Einweg-Lichtschranke: 2-adrig), Länge: 2m										
<b>Gewicht</b>		Sender, Empfänger: jeweils ca. 20g	ca. 20g									

Typ		Einweg-Lichtschranke		Reflexions-Lichtschranke	Standard-Lichttaster	Konvergenter Lichttaster		Typ mit kleinem Lichtfleck
						Diffuser Strahl	Fokussierter Strahl	
	Fronttyp	Seitentyp	Seitentyp	Seitentyp	Fronttyp	Seitentyp	Seitentyp	
Modellnr.* <sup>1</sup>	Hell-EIN	EX-21A(-PN)	EX-23(-PN) <sup>*2</sup>	EX-29A(-PN)	EX-22A(-PN)	EX-24A(-PN)	EX-26A(-PN)	EX-28A(-PN)
	Dunkel-EIN	EX-21B(-PN)		EX-29B(-PN)	EX-22B(-PN)	EX-24B(-PN)	EX-26B(-PN)	EX-28B(-PN)
Zubehör	–	Schraubendreher für Poti: 1 Stück	<ul style="list-style-type: none"> <li>• RF-200 (Reflektor): 1 Stück</li> <li>• Schraubendreher für Poti: 1 Stück</li> </ul>	Schraubendreher für Poti: 1 Stück	–	Schraubendreher für Poti: 1 Stück		

\*<sup>1</sup> Modellnummern mit der Endung -PN besitzen einen PNP-Ausgang. Die Endung P auf dem Etikett der Einweg-Lichtschranken, z.B. EX-□P, kennzeichnet einen Sender; die Endung D, z.B. EX-□D, kennzeichnet einen Empfänger. Die Reflexions-Lichtschranke mit der Endung -Y enthält den Reflektor RF-200 nicht.

\*<sup>2</sup> Die Betriebsarten Hell-EIN oder Dunkel-EIN können über den Hell-/Dunkel-Schalter, der sich am Empfänger befindet, ausgewählt werden.

\*<sup>3</sup> Die Reichweite und die Objektgröße für die Reflexions-Lichtschranke beziehen sich auf den Einsatz des Reflektors RF-200. Die angegebene Reichweite bezieht sich auf den Einstellungsbereich des Sensors in Verbindung mit diesem Reflektor; der Sensor selbst kann auch Objekte in einer geringeren Entfernung als 30mm erkennen. Wenn der Abstand zum Reflektor max. 100mm beträgt, sollte das zu erkennende Objekt undurchsichtig sein.

\*<sup>4</sup> Mit weißem Büropapier (200 x 200m). Wenn Sie dieses Produkt bei einer maximalen Reichweite von 50mm einsetzen, wird der Anpassungsbereich für die Empfindlichkeit extrem klein.

\*<sup>5</sup> Konvergenzpunkt: 10mm. Mit weißem Büropapier (50 x 50m).

\*<sup>6</sup> Konvergenzpunkt: 10mm. Mit weißem Büropapier (50 x 50m). Fleckdurchmesser 1mm mit einem einstellbaren Abstand von 10mm.

\*<sup>7</sup> Mit weißem Büropapier (100 x 100m). Fleckdurchmesser 5mm mit einem einstellbaren Abstand von 80mm.

\*<sup>8</sup> Abstand zwischen Sender und Empfänger: 1m.

\*<sup>9</sup> Abstand zwischen Sender und Empfänger: 2m.

\*<sup>10</sup> Abstand: 10mm.

\*<sup>11</sup> Mind. ø1mm Kupferdraht. Abstand: 80mm.

## Panasonic Electric Works SUNX Co., Ltd.

URL : <http://panasonic-electric-works.net/sunx>

Overseas Sales Division (Head Office)

2431-1 Ushiyama-cho, Kasugai-shi, Aichi, 486-0901, Japan

Phone: +81-568-33-7861 FAX: +81-568-33-8591

Europe Headquarter: Panasonic Electric Works Europe AG

Rudolf-Diesel-Ring 2, D-83607 Holzkirchen, Germany

Phone: +49-8024-648-0

# B | Berechnungen

## Uebersicht

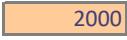
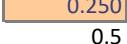
### Schussmaschiene

V-Unihockeyball  km/h  
50 m/s mm/ms

### Unihockeyball

Durchmesser 72 mm  
Durchgangszeit 1.44 ms

### Motor

Rpm  1/min  
2x 4000 1/min  
33.33 1/sec  
Radius  m  
Durchmesser 0.5 m  
Umfang 1.57 m  
V\_umf 52.36 m/s  
188.50 km/h

-> da Stab 2 unterbrüche pro Umdrehungen generiert

0.785398 Abstand Sensoren 0.79 m  
delta\_t 0.015708 s  
15.71 ms

### Lichtschranke

Reaktion  ms

### Teststrecke

Distanz  m  
Durchgangszeit 0.002 s  
2 ms

### Arduino

Clockspeed  MHz  
16000 Hz  
0.0000625 s  
0.0625 ms

## Lichtschranke

High            kein Hinderniss  
Low            Hinderniss            Wichtig zur Auslösung des Interrupts beim Arduino  
Reaktionszeit unter 1.44ms

Auswahl       EX-21A-PN - Einweglichtschranke 1 m PNP

### Fehler

Reaktionszeit        0.5 ms  
Abstand Sensor      0.1 m

### Spannungsteiler

12V - 5V

$$I = U/R$$

U                  12

U1                7

U2                5

R1                6800

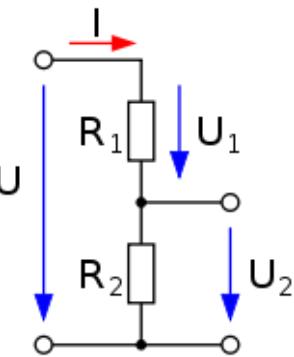
R2                4857.142857 ->

I                0.001029412

$$U_2 = U / (R_1 + R_2) * R_2$$

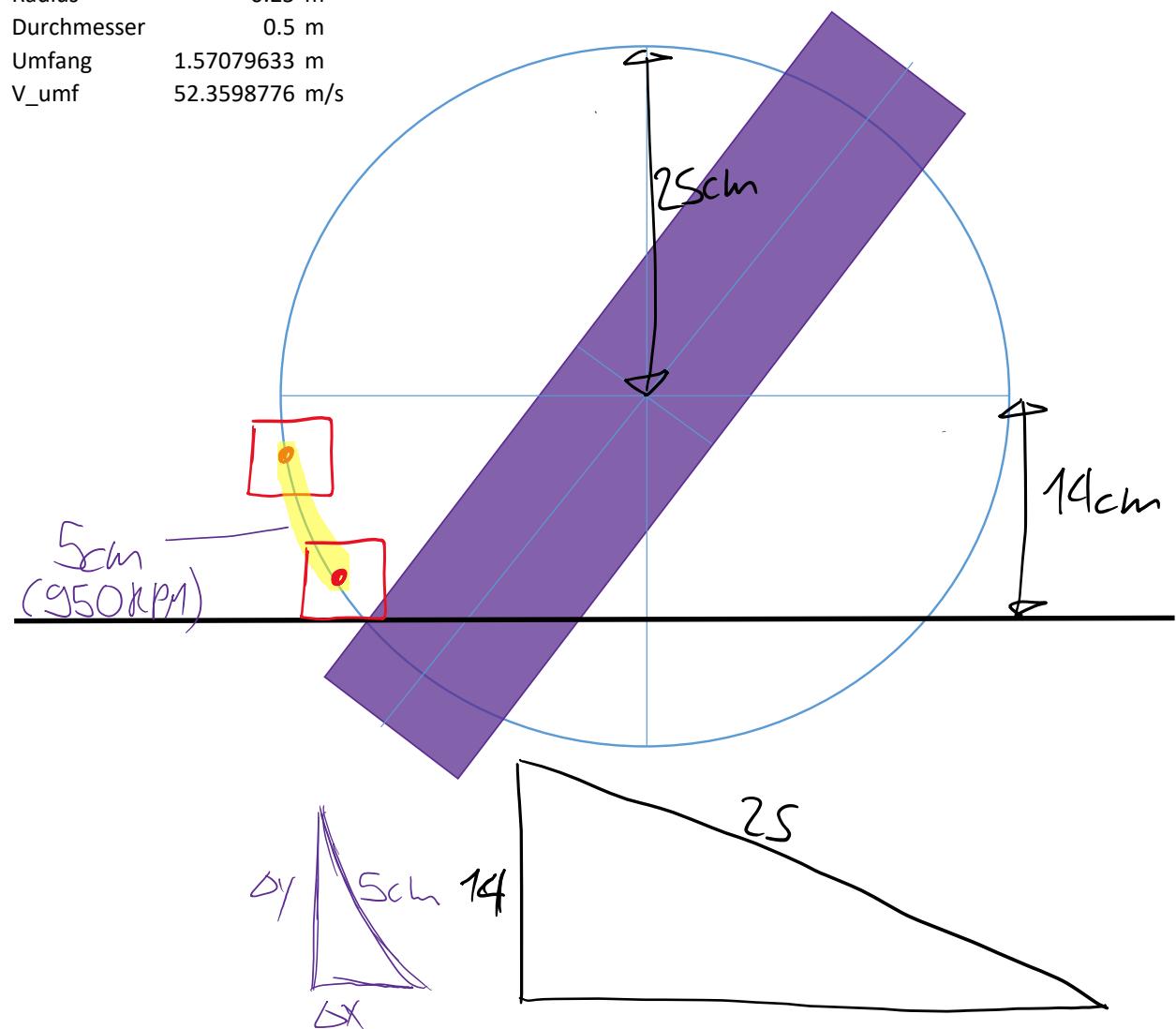
$$4700 \quad 4.90434783$$

$$0.00104348$$



## Versuchsaufbau

Uebersicht	0.785 m
delta_t	0.01571 s
	15.71 ms
Rpm	2000 1/min
Radius	0.25 m
Durchmesser	0.5 m
Umfang	1.57079633 m
V_umf	52.3598776 m/s

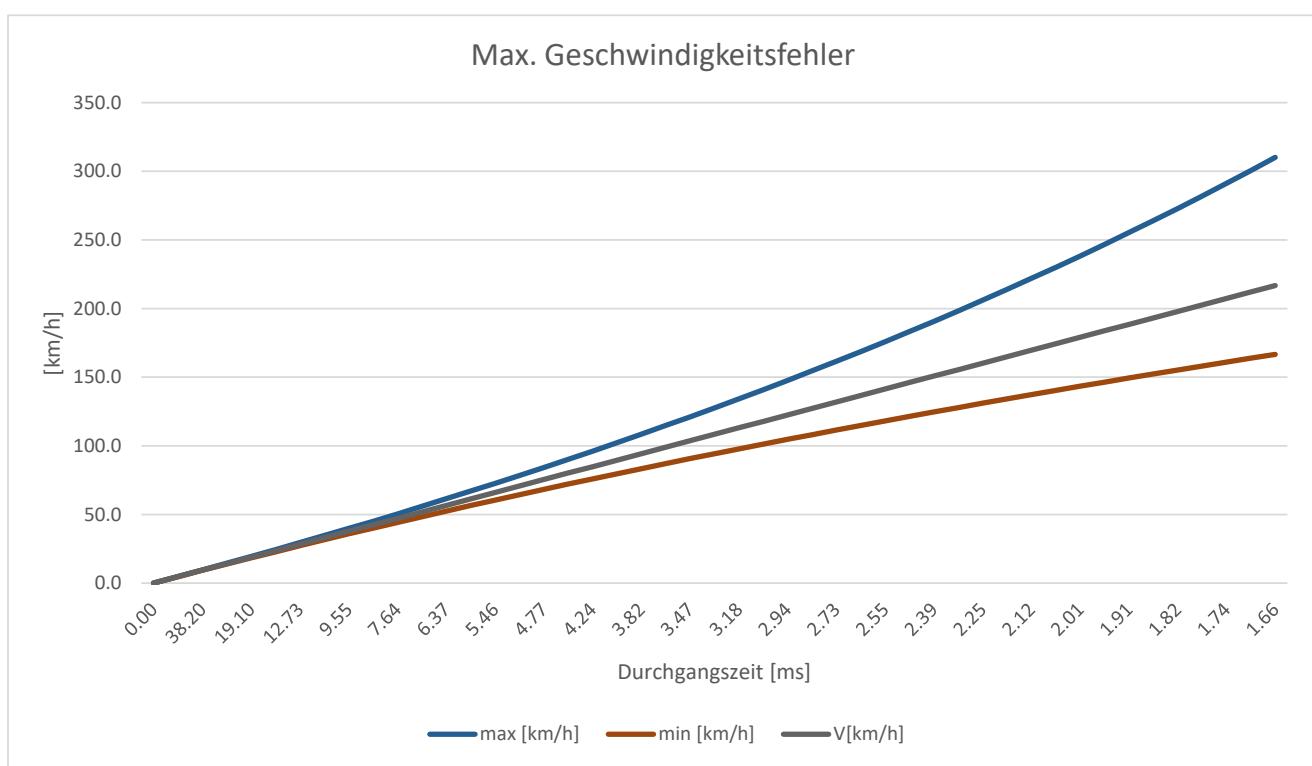
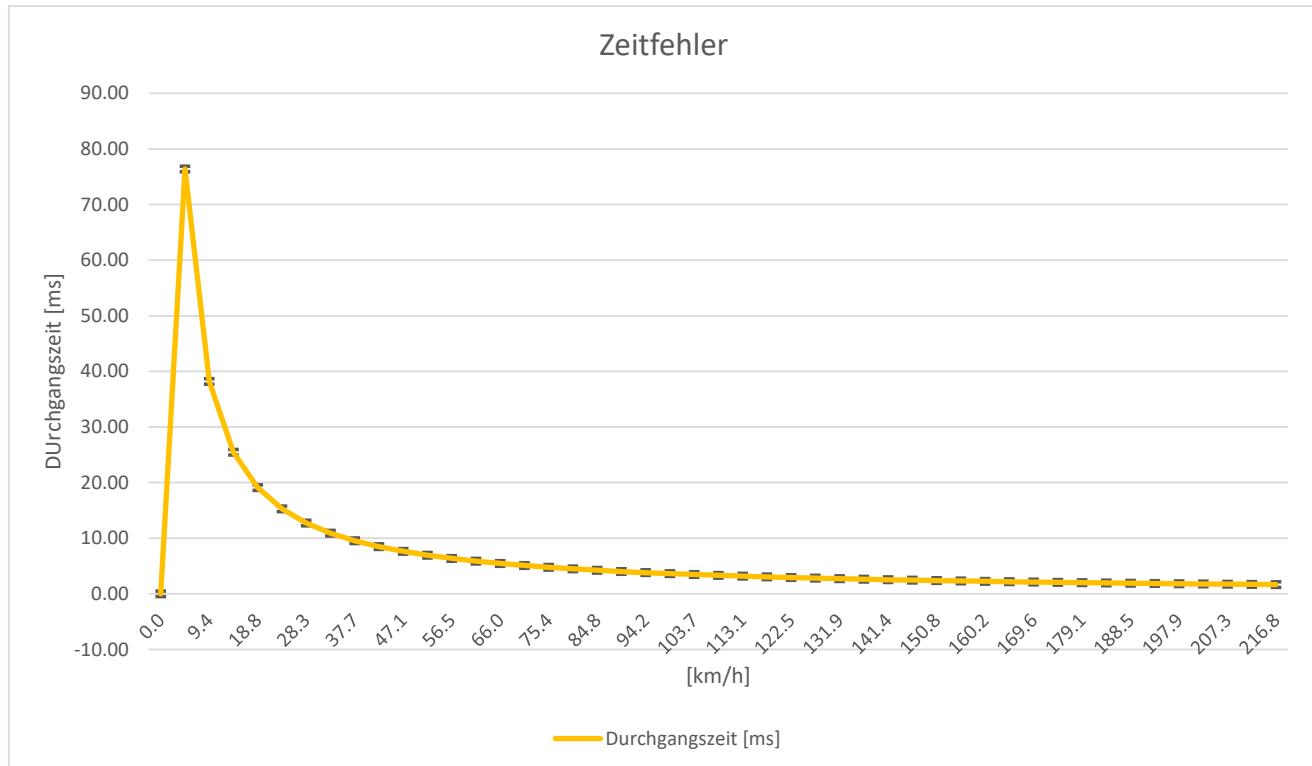


Für die Geschwindigkeit wird die Formel  $Umfang \cdot RPM / 60$  angewendet. Die min/max Geschwindigkeit ergibt sich aus der Durchgangszeit  $\pm$  des max. Sensorfehlers.

## Motor

Luca Mazzoleni

Umfang	1.57 m	Abstand Sens	0.10 m	Fehler Sensor	0.50 ms			
RPM [1/min]	V [m/s]	V[km/h]	Durchgangszeit [ms]	min [m/s]	min [km/h]	max [km/h]	- Fehler km/h	+ Fehler km/h
0	0.0	0.0	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!
50	1.3	4.7	76.39	1.3	4.7	4.7	0.0	0.0
100	2.6	9.4	38.20	2.6	9.3	9.5	0.1	0.1
150	3.9	14.1	25.46	3.9	13.9	14.4	0.3	0.3
200	5.2	18.8	19.10	5.1	18.4	19.4	0.5	0.5
250	6.5	23.6	15.28	6.3	22.8	24.4	0.7	0.8
300	7.9	28.3	12.73	7.6	27.2	29.4	1.1	1.2
350	9.2	33.0	10.91	8.8	31.5	34.6	1.4	1.6
400	10.5	37.7	9.55	10.0	35.8	39.8	1.9	2.1
450	11.8	42.4	8.49	11.1	40.1	45.1	2.4	2.7
500	13.1	47.1	7.64	12.3	44.2	50.4	2.9	3.3
550	14.4	51.8	6.94	13.4	48.4	55.9	3.5	4.0
600	15.7	56.5	6.37	14.6	52.4	61.4	4.1	4.8
650	17.0	61.3	5.88	15.7	56.5	67.0	4.8	5.7
700	18.3	66.0	5.46	16.8	60.4	72.6	5.5	6.7
750	19.6	70.7	5.09	17.9	64.4	78.4	6.3	7.7
800	20.9	75.4	4.77	19.0	68.3	84.2	7.1	8.8
850	22.3	80.1	4.49	20.0	72.1	90.1	8.0	10.0
900	23.6	84.8	4.24	21.1	75.9	96.2	8.9	11.3
950	24.9	89.5	4.02	22.1	79.6	102.3	9.9	12.7
1000	26.2	94.2	3.82	23.1	83.3	108.4	10.9	14.2
1050	27.5	99.0	3.64	24.2	87.0	114.7	12.0	15.8
1100	28.8	103.7	3.47	25.2	90.6	121.1	13.0	17.4
1150	30.1	108.4	3.32	26.2	94.2	127.6	14.2	19.2
1200	31.4	113.1	3.18	27.2	97.7	134.2	15.4	21.1
1250	32.7	117.8	3.06	28.1	101.2	140.9	16.6	23.0
1300	34.0	122.5	2.94	29.1	104.7	147.6	17.8	25.1
1350	35.3	127.2	2.83	30.0	108.1	154.5	19.1	27.3
1400	36.7	131.9	2.73	31.0	111.5	161.6	20.4	29.6
1450	38.0	136.7	2.63	31.9	114.9	168.7	21.8	32.0
1500	39.3	141.4	2.55	32.8	118.2	175.9	23.2	34.5
1550	40.6	146.1	2.46	33.7	121.4	183.3	24.6	37.2
1600	41.9	150.8	2.39	34.6	124.7	190.7	26.1	39.9
1650	43.2	155.5	2.31	35.5	127.9	198.3	27.6	42.8
1700	44.5	160.2	2.25	36.4	131.1	206.1	29.2	45.9
1750	45.8	164.9	2.18	37.3	134.2	213.9	30.7	49.0
1800	47.1	169.6	2.12	38.1	137.3	221.9	32.3	52.3
1850	48.4	174.4	2.06	39.0	140.4	230.1	34.0	55.7
1900	49.7	179.1	2.01	39.8	143.4	238.4	35.7	59.3
1950	51.1	183.8	1.96	40.7	146.4	246.8	37.4	63.0
2000	52.4	188.5	1.91	41.5	149.4	255.3	39.1	66.8
2050	53.7	193.2	1.86	42.3	152.3	264.1	40.9	70.9
2100	55.0	197.9	1.82	43.1	155.2	273.0	42.7	75.0
2150	56.3	202.6	1.78	43.9	158.1	282.0	44.5	79.4
2200	57.6	207.3	1.74	44.7	161.0	291.2	46.4	83.9
2250	58.9	212.1	1.70	45.5	163.8	300.6	48.2	88.5
2300	60.2	216.8	1.66	46.3	166.6	310.1	50.2	93.4



# C | Arduinoprogramm

```
/*
2 File: BWMvelocity.cpp
Author: Luca Mazzoleni
4 Date: 27.11.2018

6 HSR Hochschule Rapperswil

8 Description:
Programm which detects the Speed of an Object that passes two Lightbarrier.
10 Either with interrupt or polling.

12 Pythonscript for Datalogging:
https://github.com/LMazzole/ArdPyt-ReadSerial
14 */
16 #include <Arduino.h>

18 //==== Defines ====
19 /*
20 To switch between the mode Debug, Datalog and Interrupt comment
or uncomment the corresponde define below. All modes are possible at the same time.
*/
22
// #define MYDEBUG // Uncomment to see all DEBUG- Messages
24 #define MYLOG //Uncomment to see all Log-Message. Check the Pythonfile for
    Serialdataloggin.
#define USEINTERRUPT //Uncomment so use interrupts inseatd of polling

26
//Define Debug-Function
28 #ifdef MYDEBUG
    #define DEBUG_PRINT(x)           Serial.print (x)
29    #define DEBUG_PRINTDEC(x)       Serial.print (x, DEC)
30    #define DEBUG_PRINTLN(x)        Serial.println (x)
32 #else
    #define DEBUG_PRINT(x)
33    #define DEBUG_PRINTDEC(x)
34    #define DEBUG_PRINTLN(x)
36 #endif

38 //Define Log-Function for Python-Script
39 #ifdef MYLOG
40    #define LOG_PRINT(x)           Serial.print (x)
41    #define LOG_PRINTDEC(x)         Serial.print (x, DEC)
42    #define LOG_PRINTLN(x)         Serial.println (x)
44 #else
45    #define LOG_PRINT(x)
46    #define LOG_PRINTDEC(x)
47    #define LOG_PRINTLN(x)
49 #endif

51 //=====Global Variabel=====
```

```

50 const double sensordistance= 101.0*1000; //um -> 101mm
51 const double ballwidth= 72.0*1000; //um -> 72mm
52
53 double velocitykmh[4];
54 unsigned long passingtime[4];
55 int countingvar = 0;
56
57 #ifdef USEINTERRUPT
58     volatile unsigned long passingtimeLB1 = 0;
59     volatile unsigned long passingtimeLB11 = 0;
60     volatile unsigned long passingtimeLB12 = 0;
61     volatile unsigned long passingtimeLB2 = 0;
62     volatile unsigned long passingtimeLB21 = 0;
63     volatile unsigned long passingtimeLB22 = 0;
64     volatile bool LBinterrupted=false;
65 #else
66     unsigned long passingdurationLB1 = 0;
67     unsigned long passingdurationLB2 = 0;
68 #endif
69
70 //=====Pindefine=====
71 // Interrupt Pins for Uno, Nano, Mini: 2, 3
72 const byte lightbarrier1 = 2;
73 const byte lightbarrier2 = 3;
74
75 //=====Functionprototype=====
76 double calculate_velocity_ms(unsigned long passingduration,double distance);
77 void printlog(double speedinkmh[4]);
78 #ifdef USEINTERRUPT
79     void ISRLB1();
80     void ISRLB2();
81 #endif
82
83 void setup() {
84     Serial.begin(57600); // Set serial monitor baud rate. For Datalogging
85     with python the baudrates need to be consistent
86     #ifdef USEINTERRUPT
87         attachInterrupt(digitalPinToInterrupt(lightbarrier1), ISRLB1, CHANGE);
88         attachInterrupt(digitalPinToInterrupt(lightbarrier2), ISRLB2, CHANGE);
89     #endif
90
91     DEBUG_PRINTLN("Arduino Initialise");
92     LOG_PRINTLN("");
93     LOG_PRINT("Iter"); LOG_PRINT("\t");
94     LOG_PRINT("Bez"); LOG_PRINT("\t");
95     #ifdef USEINTERRUPT
96         LOG_PRINT("Int LB 1"); LOG_PRINT("\t");
97         LOG_PRINT("Int LB 2"); LOG_PRINT("\t");
98         LOG_PRINT("Int LB 11-21"); LOG_PRINT("\t");
99         LOG_PRINT("Int LB 12-22");
100    #else
101        LOG_PRINT("Puls LB 1"); LOG_PRINT("\t");

```

```

    LOG_PRINT("Puls LB 2");
102 #endif
    LOG_PRINTLN("");
104 }

106 void loop() {
107     #ifdef USEINTERRUPT
108     noInterrupts(); //eliminates racecondition
109     if (LBinterrupted){
110         LBinterrupted=false;
111         passingtime[0]=passingtimeLB11;
112         passingtime[1]=passingtimeLB12;
113         passingtime[2]=passingtimeLB21;
114         passingtime[3]=passingtimeLB22;
115         interrupts();
116         velocitykmh[0]=calculate_velocity_ms(passingtime[1]-passingtime[0],ballwidth)*3.6;
117         velocitykmh[1]=calculate_velocity_ms(passingtime[3]-passingtime[2],ballwidth)*3.6;
118         velocitykmh[2]=calculate_velocity_ms(passingtime[2]-passingtime[0],sensordistance)
119             *3.6;
120         velocitykmh[3]=calculate_velocity_ms(passingtime[3]-passingtime[1],sensordistance)
121             *3.6;
122         printlog(velocitykmh);
123     }
124     interrupts();
125     /*do something else*/
126     #else //Polling-Mode
127         passingdurationLB1 = pulseIn(lightbarrier1, LOW);
128         DEBUG_PRINTLN("passingdurationLB1: ");
129         DEBUG_PRINTLN(passingdurationLB1);
130         if(passingdurationLB1!=0){
131             passingdurationLB2 = pulseIn(lightbarrier2, LOW,500000);
132             DEBUG_PRINTLN("passingdurationLB2: ");
133             DEBUG_PRINTLN(passingdurationLB2);
134             if(passingdurationLB2!=0){
135                 passingtime[0]=passingdurationLB1;
136                 passingtime[1]=passingdurationLB2;
137                 velocitykmh[0]=calculate_velocity_ms(passingtime[0],ballwidth)*3.6;
138                 velocitykmh[1]=calculate_velocity_ms(passingtime[1],ballwidth)*3.6;
139                 printlog(velocitykmh);
140             }
141             passingdurationLB1 = 0;
142             passingdurationLB2 = 0;
143         }
144     #endif
145     DEBUG_PRINTLN("=====END-LOOP=====");
146 }

147 double calculate_velocity_ms(unsigned long duration,double distance){ //us, um
148     DEBUG_PRINTLN("calculate_velocity_ms(unsigned long passingduration,double distance) ");
149     double velocityms=0.0;
150     velocityms=distance/duration;

```

```

150 DEBUG_PRINT("Geschwindigkeit [m/s]: ");
151 DEBUG_PRINTLN(velocityms);
152 return velocityms;
153 }
154
155 void printlog(double speedinkmh[4]){
156     DEBUG_PRINTLN("printlog(double speedinkmh[4])");
157     countingvar+=1;
158     LOG_PRINT(countingvar); LOG_PRINT("\t");
159     LOG_PRINT("Geschwindigkeit [km/h]:"); LOG_PRINT("\t");
160     LOG_PRINT(velocitykmh[0]); LOG_PRINT("\t");
161     LOG_PRINT(velocitykmh[1]);
162     #ifdef USEINTERRUPT
163         LOG_PRINT("\t");
164         LOG_PRINT(velocitykmh[2]); LOG_PRINT("\t");
165         LOG_PRINT(velocitykmh[3]);
166     #endif
167     LOG_PRINTLN("");
168 }
169
170 #ifdef USEINTERRUPT
171     void ISRLB1() {
172         DEBUG_PRINT("ISRLB1() - ");
173         passingtimeLB1=micros(); //Resolution 4 micros
174         if(digitalRead(lightbarrier1)){ //Direct Port Acces would be faster
175             passingtimeLB12=passingtimeLB1;
176             DEBUG_PRINTLN(passingtimeLB12);
177             LBinterrupted=false;
178         }
179         else{
180             passingtimeLB11=passingtimeLB1;
181             DEBUG_PRINTLN(passingtimeLB11);
182             LBinterrupted=false;
183         }
184     }
185
186     void ISRLB2() {
187         DEBUG_PRINT("ISRLB2() - ");
188         passingtimeLB2=micros(); //Resolution 4 micros
189         if(digitalRead(lightbarrier2)){
190             passingtimeLB22=passingtimeLB2;
191             DEBUG_PRINTLN(passingtimeLB22);
192             LBinterrupted=true;
193         }
194         else{
195             passingtimeLB21=passingtimeLB2;
196             DEBUG_PRINTLN(passingtimeLB21);
197             LBinterrupted=false;
198         }
199     }
200 #endif

```

# D | Python

```
# -*- coding: utf-8 -*-
"""
Created on Fri Nov  2 16:35:09 2018

@author: Luca
"""

#####
## Script listens to serial port and writes contents into a file
#####
## requires pySerial to be installed
import serial
from datetime import datetime

#def AskForPortnumber():
#    port = str(input("Choose Serial Port COM: "))
#    port = "COM" + port
#    return port

locations=['/dev/ttyACM0', '/dev/ttyUSB0', '/dev/ttyUSB1', '/dev/ttyUSB2', '/dev/ttyUSB3', 'COM10', 'COM11']

COM_Number_min=8
COM_Number_max=20
locations=[None] * (COM_Number_max-COM_Number_min)

for x in range(COM_Number_min, COM_Number_max):
    locations[x-COM_Number_min]='COM'+ str(x)

baud_rate = 9600 #In arduino, Serial.begin(baud_rate)

for serial_port in locations:
    try:
        print("Trying...",serial_port)
        ser = serial.Serial(serial_port, baud_rate)
        ser.timeout = None
        break
    except:
        print("Failed to connect on",serial_port)

## loop until the arduino tells us it is ready
connected = False
while not connected:
    serin = ser.read()
    ser.flushInput()
    ser.flushOutput()
    connected = True

write_to_file_path = datetime.now().strftime('%Y%m%d-%H%M%S')+"Messung_Geschwindigkeit.txt"
```

```

print("Write to: ",write_to_file_path)
50 output_file = open(write_to_file_path, 'w')

52 output_file.write(write_to_file_path+'\n')
output_file.write("Messungen Arduino"+ '\t' + "Luca Mazzoleni"+'\n')
54

print("== Start Serial read ==")
56 try:
    while True:
        line = ser.readline()
        line = line.decode("utf-8") #ser.readline returns a binary, convert to string
        line = line.rstrip()
        print(line)
        line=datetime.now().strftime('%Y/%m/%d - %H:%M:%S')+'\t'+line+'\n'
        output_file.write(line)
64 except KeyboardInterrupt:
    ser.close()
    output_file.close()
    pass
68

print("End Script")

```

# E | Auswertung

## E.1 | Matlab-skript

```
%Ballwurfmaschine
2 %Auswertung - Histogrammplot
%Luca Mazzoleni
4 %HSR Hochschule Rapperswil
%02.12.2018
6
clc;
8 clear;
close all;
10 %% Read Data
DataTable= readtable('Data.csv', 'HeaderLines', 1);
12 data=table2array(DataTable(:, [4:9]));
IntLB1=data(:, 1);
14 IntLB2=data(:, 2);
IntLB1121=data(:, 3);
16 IntLB1122=data(:, 4);
PulsLB1=data(:, 5);
18 PulsLB2=data(:, 6);
%% Plot Histogram
20
nbins=50;
22 xlim=[185 210];
ylim=[0 200];
24 linewidth=3;
26 figure(1)
hold on
28 subplot(6,1,1);
histogram(IntLB1,nbins)
% histfit(IntLB1,nbins)
title('Interrupt LB1');
32 line([188.5 188.5], [0 300], 'LineWidth', linewidth);
xlim(xlimit)
34 ylim(ylim)
36
38 subplot(6,1,2);
histogram(IntLB2,nbins)
% histfit(IntLB2,nbins)
40 title('Interrupt LB2');
line([188.5 188.5], [0 300], 'LineWidth', linewidth);
42 xlim(xlimit)
44 ylim(ylim)
46 subplot(6,1,3);
histogram(IntLB1121,nbins)
% histfit(IntLB1121,nbins)
```

```

48 title('Interrupt LB11-21');
49 line([188.5 188.5], [0 300], 'LineWidth', linewidth);
50 xlim(xlimit)
51 ylim(ylim)
52
53 subplot(6,1,4);
54 histogram(IntLB1222,nbins)
% histfit(IntLB1222,nbins)
55 title('Interrupt LB12-22');
56 line([188.5 188.5], [0 300], 'LineWidth', linewidth);
57 xlim(xlimit)
58 ylim(ylim)
59
60 subplot(6,1,5);
61 histogram(PulsLB1,nbins)
% histfit(PulsLB1,nbins)
62 title('Polling LB1');
63 line([188.5 188.5], [0 300], 'LineWidth', linewidth);
64 xlim(xlimit)
65 ylim(ylim)
66
67 subplot(6,1,6);
68 histogram(PulsLB2,nbins)
% histfit(PulsLB2,nbins)
69 title('Polling LB2');
70 line([188.5 188.5], [0 300], 'LineWidth', linewidth);
71 xlim(xlimit)
72 ylim(ylim)
73 xlabel('Geschwindigkeit [km/h]');
74 hold off
75 %% Plot Data
76 xaxislen=[1:length(IntLB1)];
77 xlim=[1 length(IntLB1)];
78 ylim=[185 210];
79 linewidth=1;
80 linewidth1=2;
81 figure(2)
82 hold on
83 subplot(6,1,1);
84 plot(xaxislen,IntLB1,'LineWidth', linewidth1);
85 title('Interrupt LB1');
86 ylabel('Geschwindigkeit [km/h]');
87 line(xaxislen,ones(1,length(IntLB1))*188.5, 'LineWidth', linewidth, 'LineStyle', '--');
88 xlim(xlimit)
89 ylim(ylim)
90
91 subplot(6,1,2);
92 plot(xaxislen,IntLB2,'LineWidth', linewidth1)
% histfit(IntLB2,nbins)
93 title('Interrupt LB2');
94 line(xaxislen,ones(1,length(IntLB1))*188.5, 'LineWidth', linewidth, 'LineStyle', '--');
95 ylim(ylim)

```

```

100 xlim(xlimit)
101 ylabel('Geschwindigkeit [km/h]');
102
103 subplot(6,1,3);
104 plot(xaxislen,IntLB1121,'LineWidth',linewidth1)
105 % histfit(IntLB1121,nbins)
106 title('Interrupt LB11-21');
107 line(xaxislen,ones(1,length(IntLB1))*188.5,'LineWidth', linewidth, 'LineStyle', '--');
108 ylim(ylim)
109 xlim(xlimit)
110 ylabel('Geschwindigkeit [km/h]');

111 subplot(6,1,4);
112 plot(xaxislen,IntLB1222,'LineWidth',linewidth1)
113 % histfit(IntLB1222,nbins)
114 title('Interrupt LB12-22');
115 line(xaxislen,ones(1,length(IntLB1))*188.5,'LineWidth', linewidth, 'LineStyle', '--');
116 ylim(ylim)
117 xlim(xlimit)
118 ylabel('Geschwindigkeit [km/h]');

119 subplot(6,1,5);
120 plot(xaxislen,PulsLB1,'LineWidth',linewidth1)
121 % histfit(PulsLB1,nbins)
122 title('Polling LB1');
123 line(xaxislen,ones(1,length(IntLB1))*188.5,'LineWidth', linewidth, 'LineStyle', '--');
124 ylim(ylim)
125 xlim(xlimit)
126 ylabel('Geschwindigkeit [km/h]');

127 subplot(6,1,6);
128 plot(xaxislen,PulsLB2,'LineWidth',linewidth1)
129 % histfit(PulsLB2,nbins)
130 title('Polling LB2');
131 line(xaxislen,ones(1,length(IntLB1))*188.5,'LineWidth', linewidth, 'LineStyle', '--');
132 ylim(ylim)
133 xlim(xlimit)
134 ylabel('Geschwindigkeit [km/h]');
135 xlabel('Sample');
136 hold off
137 %% Berechnung Verteilung
138 figure(99)
139 hold on
140 hist_IntLB1=histfit(IntLB1,nbins);
141 hist_IntLB1=hist_IntLB1(2);
142 hist_IntLB1X=hist_IntLB1.XData;
143 hist_IntLB1Y=hist_IntLB1.YData;

144 hist_IntLB2=histfit(IntLB2,nbins);
145 hist_IntLB2=hist_IntLB2(2);
146 hist_IntLB2X=hist_IntLB2.XData;
147 hist_IntLB2Y=hist_IntLB2.YData;

```

```

152 hist_IntLB1121=histfit(IntLB1121,nbins);
154 hist_IntLB1121=hist_IntLB1121(2);
156 hist_IntLB1121X=hist_IntLB1121.XData;
158 hist_IntLB1121Y=hist_IntLB1121.YData;

158 hist_IntLB1222=histfit(IntLB1222,nbins);
160 hist_IntLB1222=hist_IntLB1222(2);
162 hist_IntLB1222X=hist_IntLB1222.XData;
164 hist_IntLB1222Y=hist_IntLB1222.YData;

162 hist_PulsLB1=histfit(PulsLB1,nbins);
164 hist_PulsLB1=hist_PulsLB1(2);
166 hist_PulsLB1X=hist_PulsLB1.XData;
168 hist_PulsLB1Y=hist_PulsLB1.YData;

168 hist_PulsLB2=histfit(PulsLB2,nbins);
170 hist_PulsLB2=hist_PulsLB2(2);
172 hist_PulsLB2X=hist_PulsLB2.XData;
174 hist_PulsLB2Y=hist_PulsLB2.YData;
172 hold off
173 close(99)

174 figure(3)
175 hold on
176 plot(hist_IntLB1X,hist_IntLB1Y,'LineWidth',2)
177 plot(hist_IntLB2X,hist_IntLB2Y,'LineWidth',2)
178 plot(hist_PulsLB1X,hist_PulsLB1Y,'--','LineWidth',2)
179 plot(hist_PulsLB2X,hist_PulsLB2Y,'--','LineWidth',2)
180 plot(hist_IntLB1121X,hist_IntLB1121Y,'LineWidth',2)
181 plot(hist_IntLB1222X,hist_IntLB1222Y,'LineWidth',2)
182 % title('Histogram');
183 legend('Interrupt LB1', ...
184     'Interrupt LB2',...
185     'Polling LB1',...
186     'Polling LB2',...
187     'Interrupt LB11-21',...
188     'Interrupt LB12-22');
189 xlabel('Geschwindigkeit [km/h]');
190 line([188.5 188.5], [0 300],'LineWidth',2);
191 xlim([185 205])
192 ylim([0 30])
193 hold off

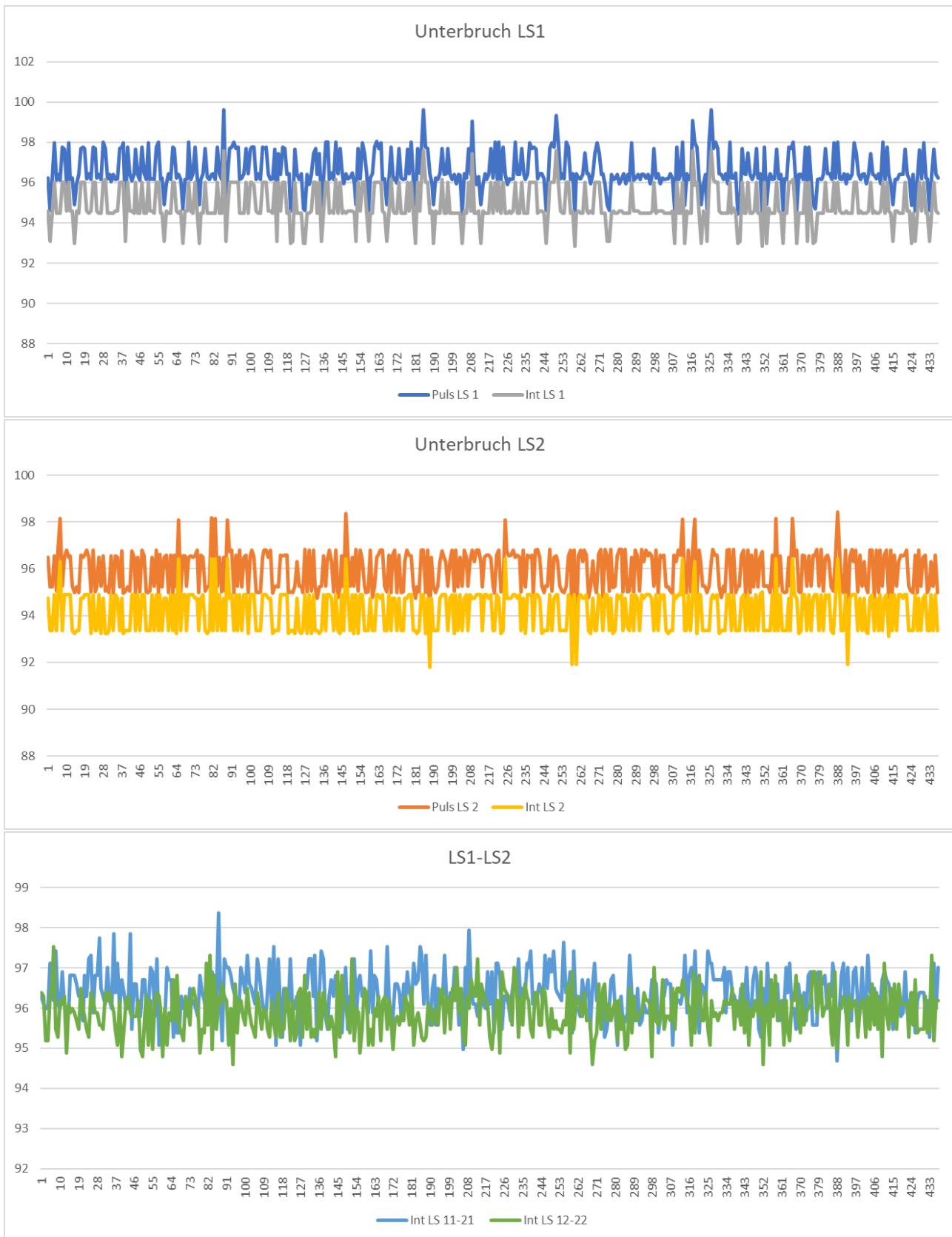
194 %% Ausgabe
195 mean_data=mean(data,'omitnan');
196 std_data=std(data,'omitnan');
197 speed = 188.5; %Sollwert

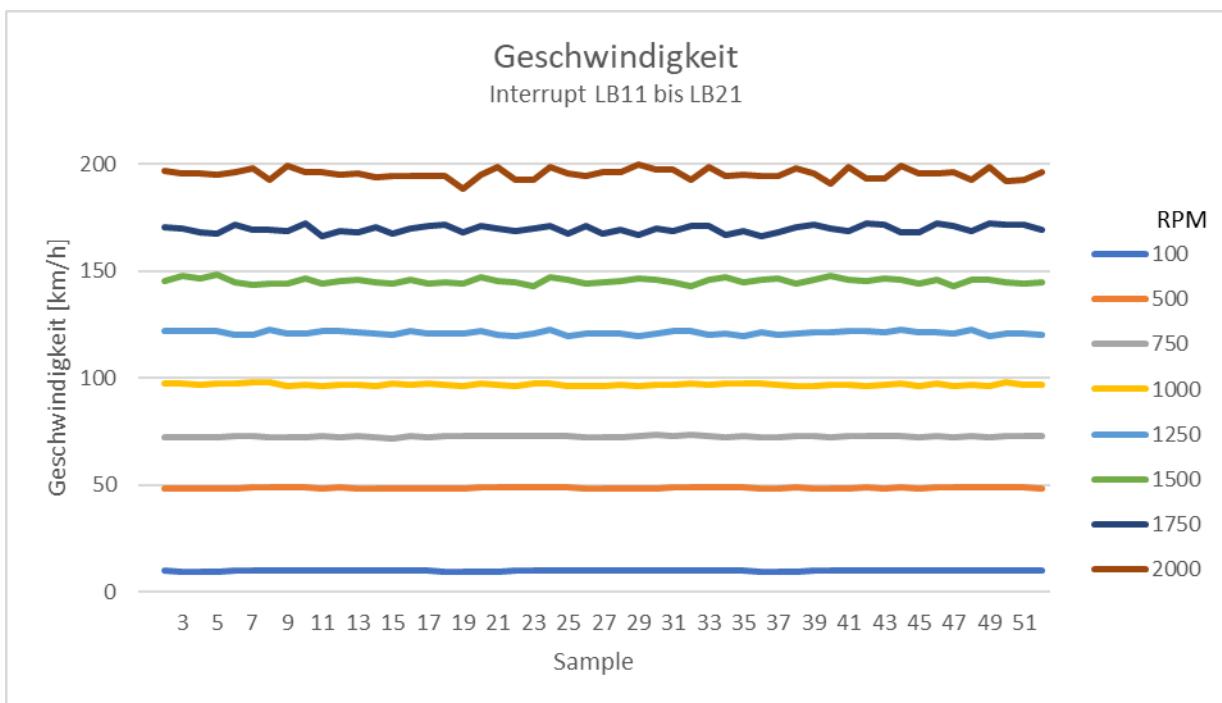
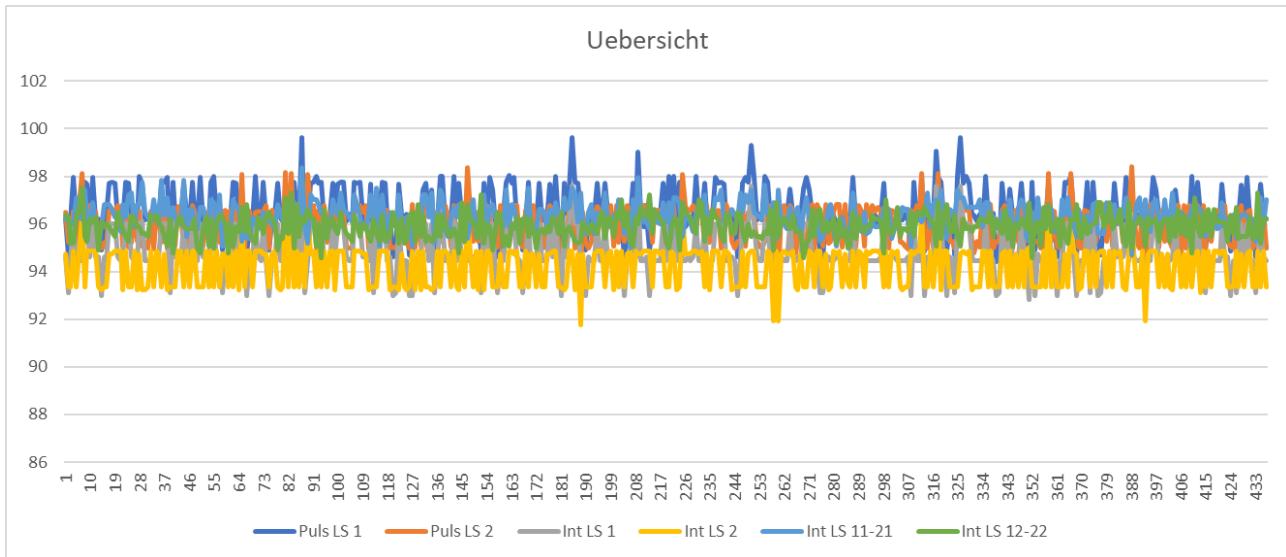
198 genau_data=abs(speed-mean_data);
199 genau_data_proz=genau_data/speed*100;
200 prez_data_proz=std_data./mean_data*100;

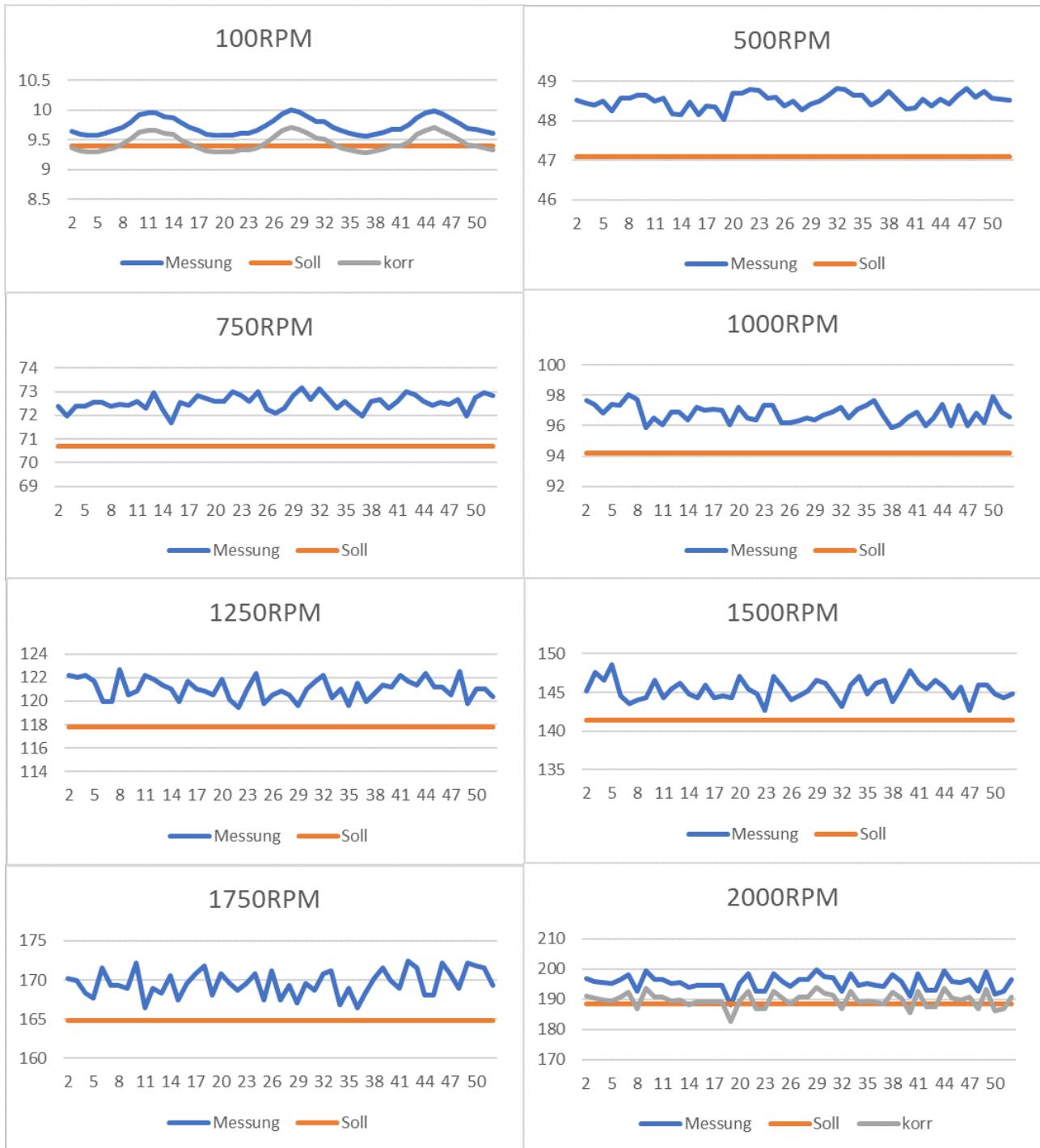
```

```
204 disp('====Auswertung====')
206 disp('Genauigkeit [km/h]')
207 disp('    IntLB1      IntLB2      IntLB1121      IntLB1222      PulsLB1      PulsLB2')
208 disp(genau_data)
209 disp('Prazision [km/h]')
210 disp('    IntLB1      IntLB2      IntLB1121      IntLB1222      PulsLB1      PulsLB2')
211 disp(std_data)
212 disp('Genauigkeit [%]')
213 disp('    IntLB1      IntLB2      IntLB1121      IntLB1222      PulsLB1      PulsLB2')
214 disp(genau_data_proz)
215 disp('Prazision [%]')
216 disp('    IntLB1      IntLB2      IntLB1121      IntLB1222      PulsLB1      PulsLB2')
217 disp(prez_data_proz)
```

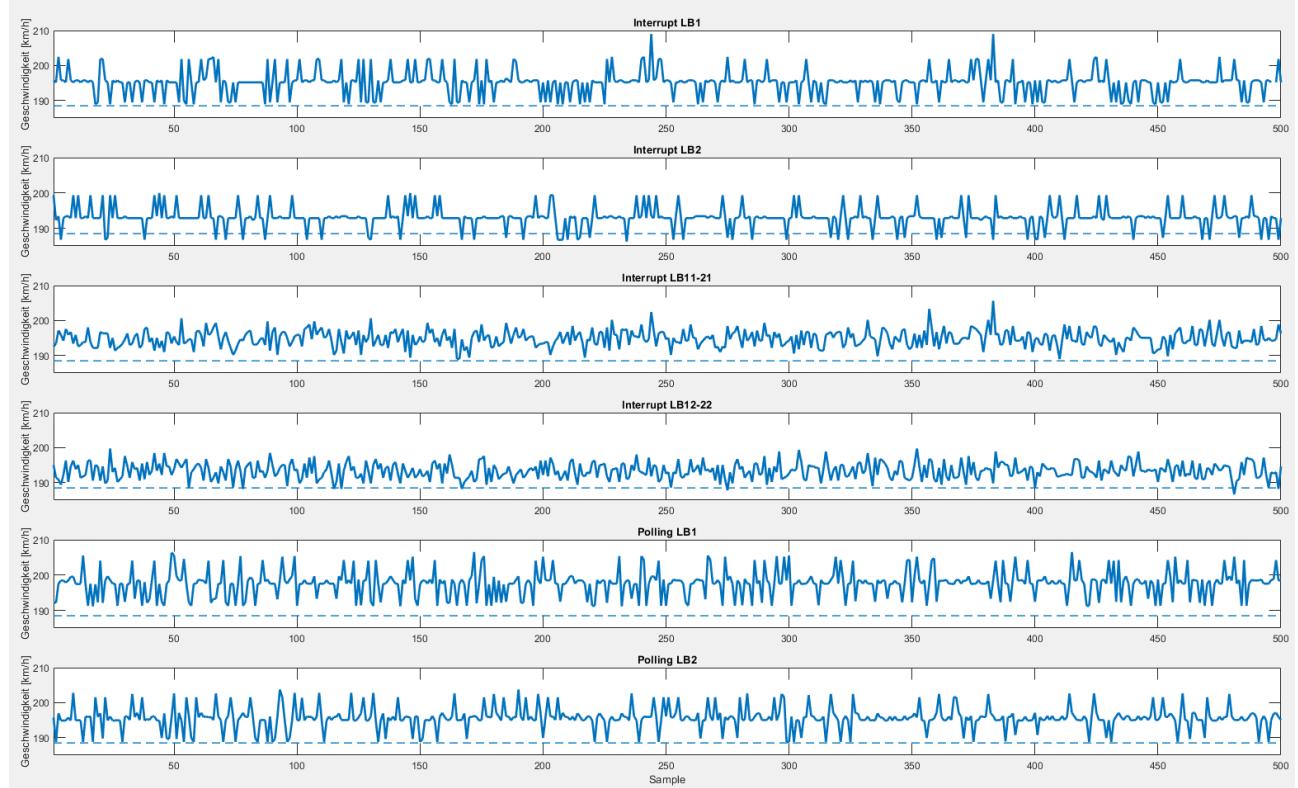
## E.2 | Plots







## Gemessenens Signal bei 2000RPM



## Histogramm des gemessenen Signals bei 2000RPM

