

MASTER THESIS

zur Erlangung des akademischen Grades
„Master of Science in Engineering“
im Studiengang Industrielle Elektronik

Aufbau eines automatisierten Mess- und Auswertesystems zur Bestimmung der Bestrahlungsstärkeverteilung in einem stationären Sonnensimulator

Ausgeführt von: Thomas Schmatz BSc

Personenkennzeichen: 1010300002

1. BegutachterIn: DI Bernhard Kubicek
2. BegutachterIn: DI (FH) Thomas Krametz

Wien, 24. Mai 2012

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht.“

Ort, Datum

Unterschrift

Kurzfassung

Die Arbeit umfasst die Planung und Realisierung eines selbstfahrenden Messroboters, der die Bestrahlungsstärkeverteilung in der Prüfebene eines stationären Sonnensimulators erfasst. Neben den Einstrahlungsdaten werden weitere zusätzliche Informationen über die Umgebungsbedingungen im Prüfkanal aufgezeichnet. Bei der Umsetzung wurden Rapid-Prototyping Techniken (3D-Druck, Platinenfräse und Lasercutter) eingesetzt. Behandelt werden theoretische Grundlagen und normative Anforderungen an stationäre Sonnensimulatoren, sowie Messunsicherheitsberechnungen und Validierung des Gesamtsystems.

Schlagwörter: Sonnensimulator, Messroboter, Bestrahlungsstärke

Abstract

This paper discripes the planning and the realisation of a autonomous measurement robot. The robot measures the irradiance in a test plane a a continous solar simulator.

Keywords: Keyword 1, Keyword 2, Keyword 3, Keyword 4, Keyword 5

Danksagung

Ich danke meinen Eltern für die Unterstützung und Geduld, die sie während des Studiums aufgebracht haben. Ich danke meinen Hochschulbetreuer für die umfangreiche Betreuung. Ich danke meinen Firmenbetreuer Thomas Krametz für die Zeit die er sich genommen hat. ... (Rohfassung!!!!)

Inhaltsverzeichnis

1. Aufgabenstellung	1
1.1. Stationäre Photovoltaik Sonnensimulatoren	1
1.2. Sonnensimulator Aufbau	1
1.3. Anforderungen	1
1.4. Theorie Referenzzelle	3
2. Entwicklungsprozess	6
2.1. Hardwaredesign und Komponenten	6
2.1.1. Mecanum-Plattform	6
2.1.2. Chassis	9
2.2. Steuerungselektronik	9
2.2.1. Motorensteuerung	9
2.2.2. Optische Sensorik	12
2.2.3. Spannungversorgung	12
2.2.4. Mikrokontroller	15
2.2.5. Temperatursensoren	16
2.3. Softwareentwicklung	16
2.3.1. Entwicklungsumgebung	16
2.3.2. Auswertung optische sensoren	16
2.3.3. Auswertung ADCs	18
2.3.4. Programmablauf	18
3. Kalibration	20
3.1. Temperatursensoren	20
3.2. Messzelle	20
3.3. Strommessung	24
3.4. Thermische Stabilität der Temperaturmessung	24
4. Messung	27
4.1. Messaufbau	27
4.1.1. Platten	27
4.1.2. Ablauf	27
4.2. Auswertung	28
4.3. Vermessung der Ausleuchtung einzelner Lampen	29
4.4. Schlussfolgerung	29
Literaturverzeichnis	37
Abbildungsverzeichnis	39

Tabellenverzeichnis	40
Abkürzungsverzeichnis	41
A. Sourcecode Arduino	42
B. Sourcecode Auswertung	57

1. Aufgabenstellung

Dieses Kapitel beschreibt die Notwendigkeit von Messungen in stationären Sonnensimulatoren, den Aufbau eines solchen Simulators, sowie dessen normativen Anforderungen. Weiters wird die Funktion eines Photovoltaik-Zelle beschrieben.

1.1. Stationäre Photovoltaik Sonnensimulatoren

Photovoltaik-Sonnensimulatoren werden zur elektrischen Charakterisierung und Prüfung von Photovoltaik-Modulen verwendet. Der wesentliche Vorteil liegt darin, dass die Durchführung unter reproduzierbarem Umweltbedingungen (Bestrahlungsstärke, spektrale Zusammensetzung des einfallenden Lichtes und die Prüfguttemperatur) ermöglicht wird. Ein Nachteil ist die nicht 100% Übereinstimmung mit dem Spektrum des natürlichen Sonnenlichts. Am Markt erhältliche Sonnensimulatoren für Modulgröße ($1,6 \text{ m}^2$) können das Spektrum nicht verändern, sind also für einen Sonnenstand beschränkt. Im Allgemeinen werden Messergebnisse auf Standard-Prüfbedingungen (STC) bezogen, die durch 1000W/m^2 , 25°C und Am 1.5 definiert ist. Die Air Mass (AM) beschreibt das Verhältnis des Weges des einfallenden Sonnenlichtes durch die Atmosphäre bezogen auf senkrechten Einfall. Je länger der Weg des Lichtes durch die Atmosphäre desto größer ist der Einfluss durch Streuung, Reflexion und Absorption, was sowohl die Stärke der Einstrahlung als auch die spektrale Zusammensetzung des Sonnenlichtes vom Sonnenstand abhängig macht.

$$AM = \frac{L}{L_0} \approx \frac{1}{\cos z} \quad (1.1)$$

Wobei L die Länge des Weges des Sonnenlichtes durch die Atmosphäre, L_0 die Weglänge bei senkrechten Einfall und z der Zenitwinkel des Sonnenstandes in Grad ist. Die spektrale Verteilung des Referenzsonnenspektrums Am 1.5 ist durch die Norm IEC 60904-3 festgelegt.

1.2. Sonnensimulator Aufbau

Als Lichtquellen dienen 10 Metallhalogenoid Strahler mit je 4 kW elektrischer Leistung. Für eine spektral unabhängige Reduzierung der Beleuchtungsstärke sind die Strahler an einer höhenverstellbaren Aufhängung befestigt. Die Testobjekte werden mittels Prüfgut-Einschub in einem Windkanal positioniert, dessen obere Abdeckung aus einem geeigneten Solarglas besteht. Die geneigte Glaspalte bewirkt eine Querschnittsreduktion des Luftkanals und somit eine Erhöhung der Strömungsgeschwindigkeit zwischen Zuluft- und Abluftseite. Diese stetige Erhöhung der Geschwindigkeit ist notwendig, damit die sich erwärmende Zuluft eine konstante Kühlwirkung hat, und somit die Testobjekte eine konstante Temperatur über die gesamte Fläche haben.

1.3. Anforderungen

Sonnensimulatoren werden nach IEC 60904-9 in 3 Klassen eingeteilt: A, B, und C.

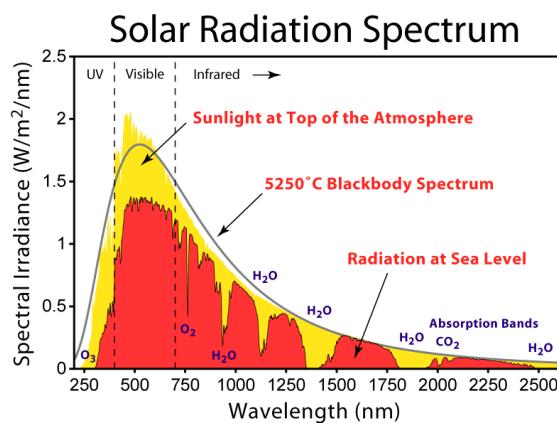


Abbildung 1.1.: Der Vergleich von Spektrum des Sonnenlichtes im Weltall (AM 0), der Strahlung eines schwarzen Körpers von 5250°C und dem AM1.5 Spektrum



Abbildung 1.2.: Der Aufbau des Sonnensimulators: Die 10 Metallhalogenoid Strahler, der Windkanal und die Prüfebene sind zu erkennen

	Wellenlängenbereich in nm	% der totalen Einstrahlung
1	400 - 500	18,4
2	500 - 600	19,9
3	600 - 700	18,4
4	700 - 800	14,9
5	900 - 900	12,5
6	900 - 1100	15,9

Tabelle 1.1.: Spektrale Strahlungsverteilung nach IEC 60904-9

Klassifikation	Spektrale Übereinstimmung	Örtliche Homogenität	Kurzzeit-stabilität	Langzeit-stabilität
A	0,75 - 1,25	2 %	0,5 %	2 %
B	0,6 - 1,4	5 %	2 %	5 %
C	0,4 - 2,0	10 %	10 %	10 %

Tabelle 1.2.: Anforderungen an die 3 verschiedenen Simulatorklassen

1.4. Theorie Referenzzelle

Es gibt verschiedene Solarzelltechnologien. Wirtschaftlich bedeutend sind im Moment nur kristalline Siliziumzellen und Dünnschichtzellen. Die kristallinen Siliziumzellen werden in monokristalline Zellen und multikristalline Zellen unterschieden. Beide zusammen machen über 80% des weltweiten Photovoltaikmarktes aus [1]. Als Referenzzelle für Messungen werden ausschließlich kristalline Zellen verwendet. Eine Referenzzelle ist eine genau vermessene Solarzelle, die als Strahlungsmessgerät dient. Wichtig ist, dass die Referenzzelle eine ähnliche spektrale Empfindlichkeit hat wie die zu messende Zelle bzw. das zu messende Modul. Siliziumsolarzellen bestehen aus einer großflächigen Diode. Die Sperrsicht ist dabei dem Sonnenlicht ausgesetzt. Gelangt ein Lichtquanten in die Sperrsicht kann aufgrund des inneren Photoeffektes ein Elektron/Loch Paar erzeugt werden. Durch das elektrische Feld in der Sperrsicht werden die Ladungsträger getrennt bevor sie kombinieren können. Elektronen bewegen aufgrund ihrer negativen Ladung entgegen der Feldrichtung in die n-Zone. Löcher wandern in Feldrichtung zur raumladungsfreien p-Zone. Die Leerlaufspannung einer Solarzelle ist kleiner als die Diffusionsspannung, der Spannung über die Raumladungszone, die der Diffusion von Ladungsträgern entgegenwirkt.

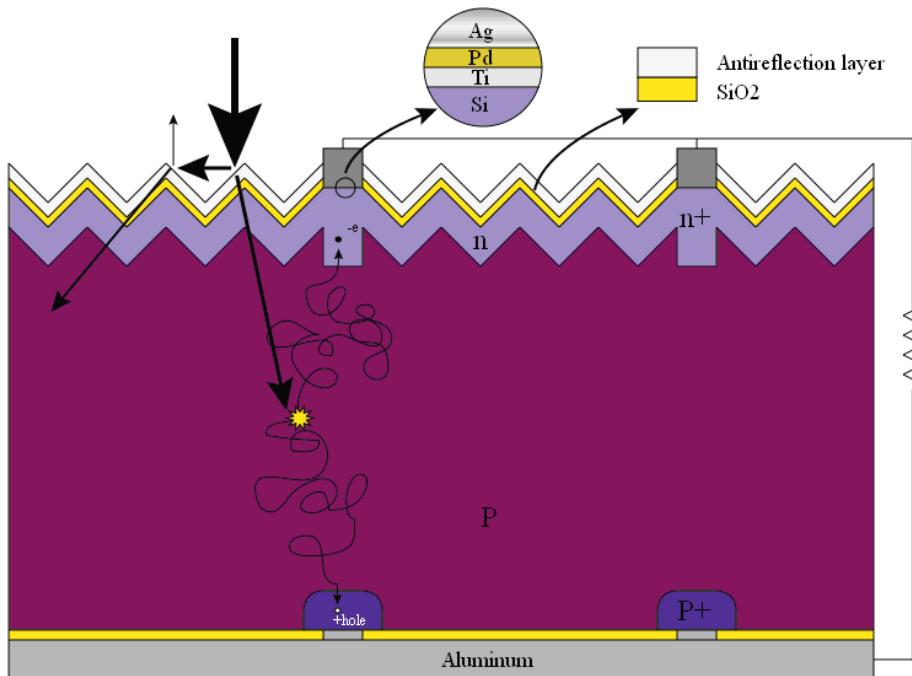


Abbildung 1.3.: Schematischer Aufbau einer Solarzelle

Die unbeleuchtete Solarzelle ist funktioniert wie eine normale Halbleiterdiode, die einen Durchlassstrom von p- nach n-Seite fließen lässt, falls eine Spannung von p nach n anliegt. Bei Beleuchtung wird zusätzlich ein Photostrom erzeugt, welcher proportional zur Bestrahlungsstärke und der Zellfläche ist. Das Eindiodenmodell (siehe Abbildung 1.4) besteht daher aus einer Stromquelle, dazu parallel einer Diode, einen Parallelwiderstand und einem Serienwiderstand. Der Parallelwiderstand fasst Kurzschlüsse zusammen, die realen Solarzellen am Rand oder an den Korngrenzen auftreten können. Mit dem Serienwiderstand werden alle Spannungsabfälle in der Solarzelle erfasst. Eine ideale Solarzelle hat einen Serienwiderstand von null und einen Parallelwiderstand von unendlich.

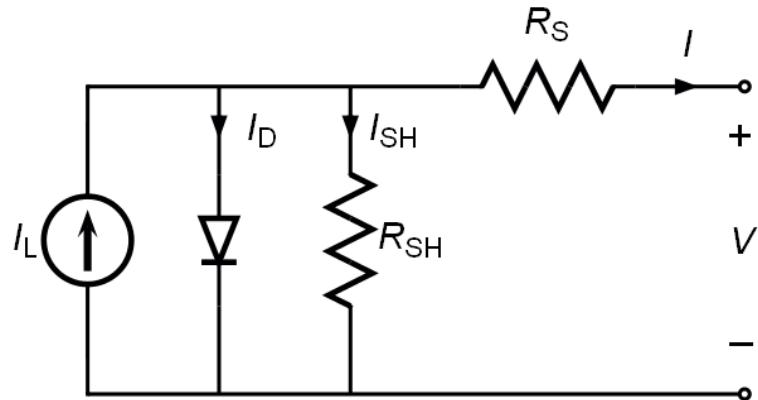


Abbildung 1.4.: Das Eindiodenmodell einer Solarzelle

Durch den zusätzlichen Photostrom verschiebt (siehe Abbildung 1.5) sich die Diodenkennlinie. Im Leerlauf bzw. Kurzschlussbetrieb gibt die Solarzelle keine Leistung ab. Der Punkt der Kennlinie mit der maximalen Leistung wird als Maximum Power Point (MPP) bezeichnet.

Der Füllfaktor berechnet sich aus Strom im maximalen Leistungspunkt I_{mp} , Spannung im maximalen Leistungspunkt U_{mp} , Kurzschlusstrom I_{sc} und Leerlaufspannung U_{oc} zu:

$$FF = \frac{I_{mp}U_{mp}}{I_{sc}U_{oc}} \quad (1.2)$$

Die Leerlaufspannung hängt von der Temperatur ab. Mit steigender Temperatur wird der Bandabstand kleiner. Dadurch können Photonen kleinerer Energie absorbiert werden und somit steigt der Kurzschlusstrom. Die Leistung im MP-Punkt sinkt mit steigender Temperatur. Zusammenhang von Kurzschlusstrom und Leerlaufspannung.

$$I_{sc} = I_0(e^{\frac{qU_{oc}}{kT}} - 1) \quad (1.3)$$

Hierbei ist I_0 der Sättigungsstrom der Diode (im Ersatzschaltbild 1.4):

$$I_0 = A \left(\frac{qD_e n_i^2}{L_e N_A} + \frac{qD_h n_i^2}{L_h N_D} \right) \quad (1.4)$$

N_A und N_D sind die Dichten von Akzeptoren sowie Donatoren, D_e und D_h sind die Diffusionskoeffizienten von Elektronen und Löchern, n_i ist die intrinsische Ladungsträgerdichte, L_e und L_h sind Diffusionslängen von Elektronen und Löchern. Alles materialabhängige Konstanten.

$$U_{sc} = \frac{kT}{q} \ln \left(\frac{I_L}{I_0} + 1 \right) \quad (1.5)$$

Hier ist I_L der Photostrom, und berechnet sich :

$$I_L = qAG(L_e + W + L_h) \quad (1.6)$$

Bei A handelt es sich um die Zellfläche, bei G um die Generationsrate von Elektron-Löcher-Paaren und bei W um die Breite der Raumladungszone.

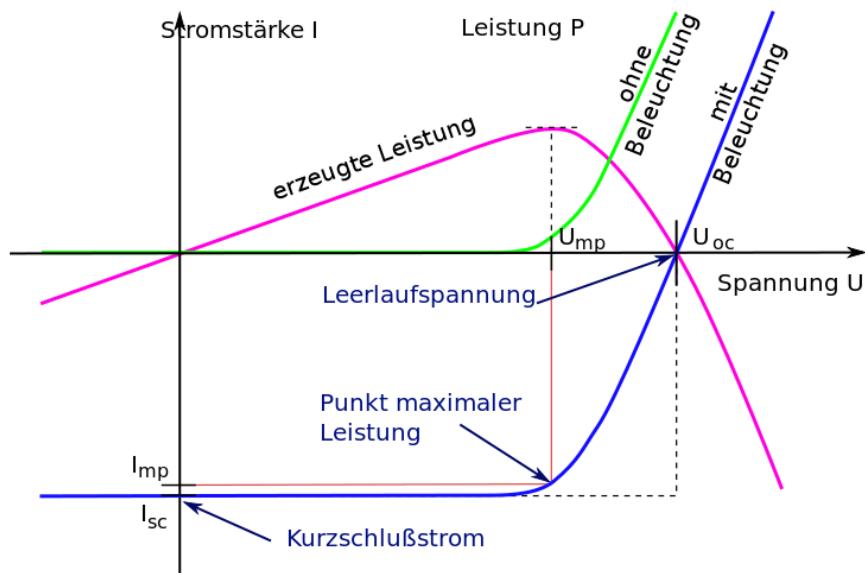


Abbildung 1.5.: Dunkel- und Hellkennlinien

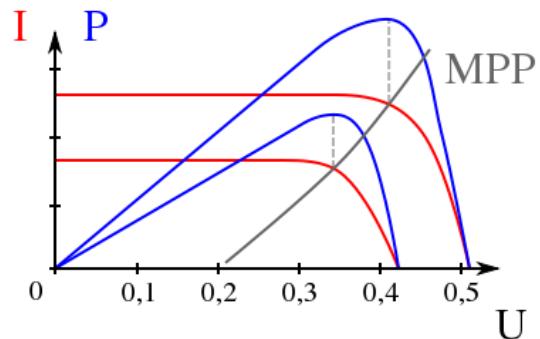


Abbildung 1.6.: Abhängigkeit des MPP von der Einstrahlung

2. Entwicklungsprozess

Diese Kapitel beschreibt die Entwicklung des Roboters und ist in 3 Unterabschnitte gegliedert: das mechanische Design, die Elektronik und die Software. Einige Designelemente waren zu Beginn der Entwicklung vorgegeben. Der Roboter sollte mit Mecanum-Rädern ausgestattet sein. Er sollte eine photovoltaische Messzelle aufnehmen können. Er muss sich irgendwie im Raum orientieren können. Die Materialien sollten UV beständig sein, da der Roboter in einer UV-verstrahlten Umgebung arbeitet. Die Mecanum-Räder erlauben es dem Roboter sich auch seitwärts fortzubewegen, die Längsachse des Roboters, und somit auch die Messzelle, während des gesamten Messvorganges die Orientierung im Raum beibehält.

2.1. Hardwaredesign und Komponenten

Dieser Abschnitt behandelt die Entwicklung der mechanischen Komponenten, diese sind die 4 Räder und die Chassis des Roboters. Die mögliche Bauhöhe ist durch den Windkanal im Sonnensimulator beschränkt, zusätzlich soll die Messzelle möglichst in der gleichen Höhe wie zu messende Module liegen. Die Größe der Messzelle, zusammen mit dem Raddurchmesser und der Breite der Räder gibt eine Minimalabmessung vor die der Roboter nicht unterschreiten kann. Zusätzlich müssen noch die Motoren, der Akku und die Elektronik im Inneren des Roboters Platz finden. Designprozess: - Mecanumräder sind festgelegt - Platz für Messzelle, oben, darunter Elektronik und Motoren - Mecanum Räder innerhalb des Chassis, Zelle am Dach - Optische Sensoren zum Verfolgen einer Linie, da auch seitwärts gefahren werden soll, in der Mitte des Roboters an der Unterseite. - Platz für den Mikrocontroller, die Elektronik, den Akku im Inneren. - Da wichtige Komponenten wie der Akku später gewechselt wurden, war es gut Reserveplatz zu haben.

Insgesamt ist der Roboter ein wenig zu groß und sehr breit geworden. Die Breite ist aber kein Problem, da es kein Bedürfnis den Roboter bis an den seitlichen Rand der Messebene fahren zu lassen. Das ist für Messungen kein interessanter Bereich mehr.

2.1.1. Mecanum-Plattform

Das Mecanum-Rad ist ein Rad, das Fahrmanöver in jede Richtung erlaubt, ohne dass das Fahrzeug mit einer mechanischen Lenkung ausgestattet ist. Benannt ist es nach dem schwedischen Unternehmen Macanum, welches dieses Rad 1971 entwickelt hat. Erreicht wird die Wendigkeit der Fahrzeuge durch den Einsatz von Mecanum Rädern, die einzeln angetrieben werden. Diese Räder bestehen aus einer Felge, auf der unter einem Winkel von 45 Grad befestigte, ballige Rollen so angebracht sind, dass sie über den Abrollumfang einen exakten Kreis bilden. Durch die Schräganordnung der Rollen entstehen beim Antreiben des Rades 2 Kraftkomponenten. Gegeneinander gerichtete Kräfte der einzelnen Räder werden über die Achsen und den Rahmen kompensiert. Die übrigen Kräfte addieren sich zur resultierenden Fahrtrichtung. Auf diese Weise sind durch entsprechendes Ansteuern der einzelnen Räder omnidirektionale Fahrmanöver möglich (siehe Abbildung 2.1).

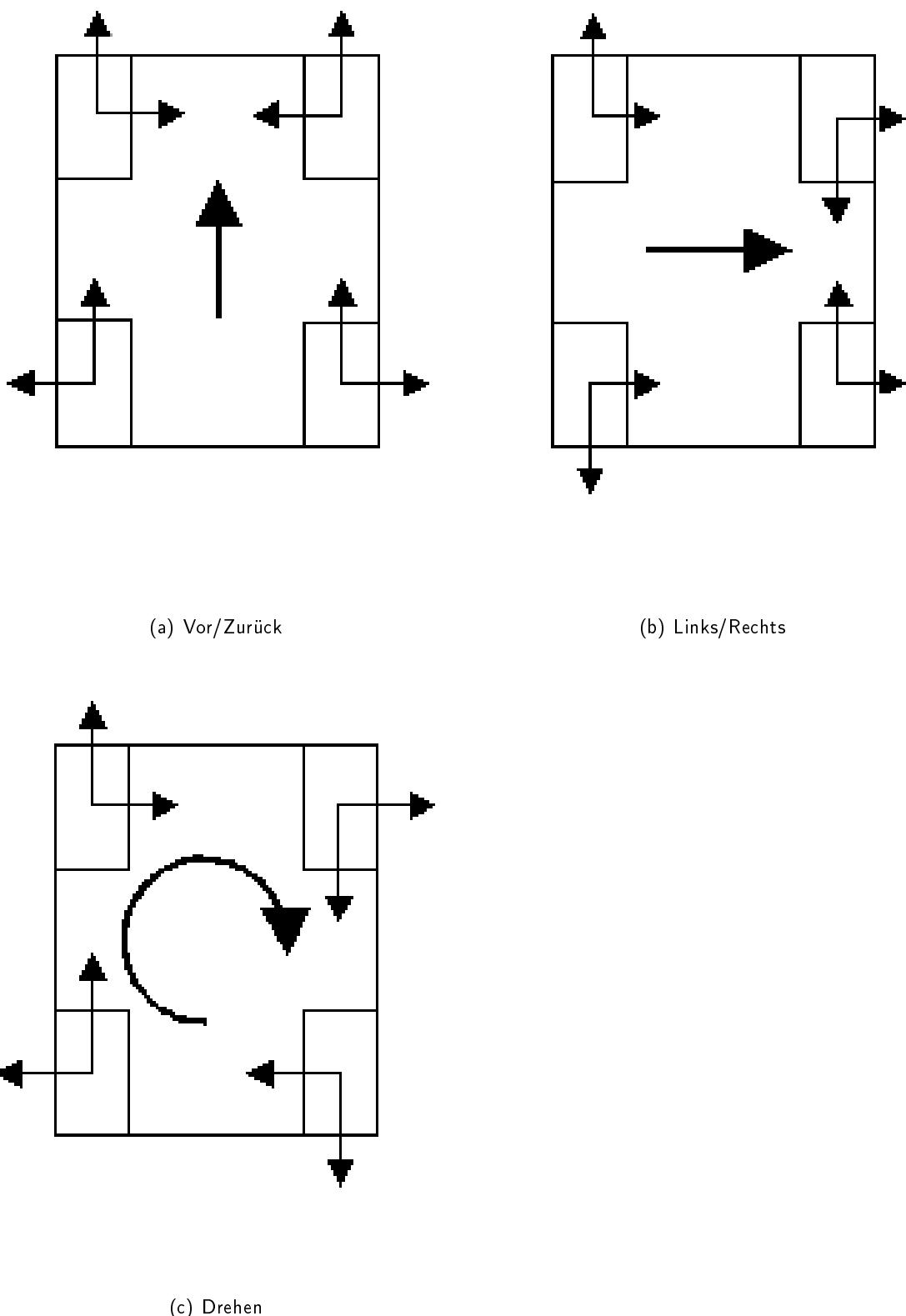


Abbildung 2.1.: Einige mögliche Fahrmanöver eines Mecanum-Rad-Roboters

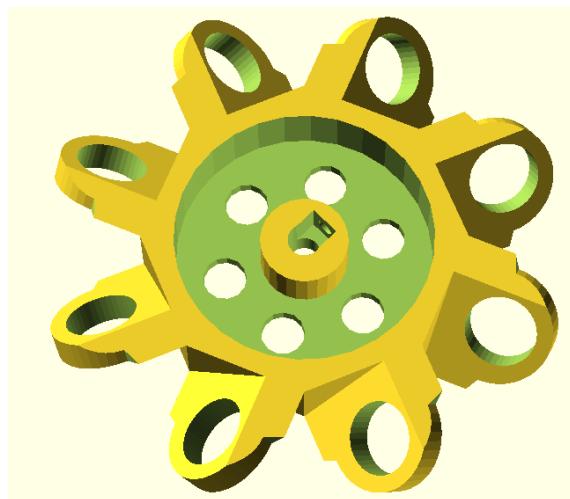


Abbildung 2.2.: Die Felge des Mecanum Rades

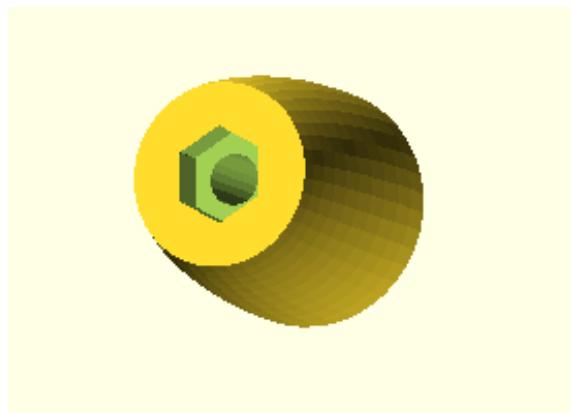


Abbildung 2.3.: Eine der 16 Rollen pro Reifen

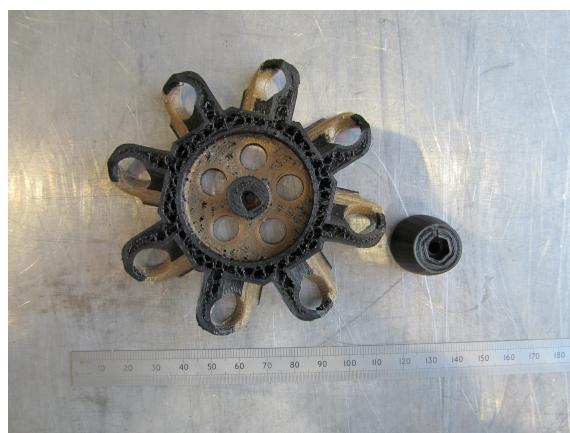


Abbildung 2.4.: Ein gedrucktes Rad samt Rolle

2.1.2. Chassis

Das Chassis wurde aus Sperrholz gefertigt. Dieses Material wurde gewählt weil es leicht zu bearbeiten, robust und UV-beständig ist. Das Design wurde mit QCAD entwickelt. QCAD ist ein 2-dimensionales CAD Programm. Die Abmessungen des Chassis wurde im Wesentlichen abgestimmt auf:

- Den Durchmesser und der Breite der Mecanum Räder, die innerhalb des Chassis liegen, damit die Räder vor der UV Strahlung geschützt sind, kein Einfluss (Beschattung oder Reflexionen) auf die Messzelle besteht.
- Die Größe der Motoren.
- Den Arduino-Mikrocontroller , die Motorsteuerelektronik und die Messelektronik
- Den Akku zur Stromversorgung, die dazugehörige Akku Spannungsüberwachung
- Die Abmessung der eingekapselten Solarzelle.

Die Einzelteile wurden mit einem Lasercutter aus einer Sperrholzplatte geschnitten. Die Einzelteile wurden teilweise miteinander verleimt. Damit eine zukünftige Wartung problemlos möglich ist, wurde der Aufbau verschraubt ausgeführt. Die Motoren sind verschraubt, damit ein Austausch möglich ist. Einzelne Räder als auch die Rollen lassen sich bei Bedarf austauschen. Das Chassis besteht aus der Bodenplatte, den Seitenwänden und der oberen Abdeckung mit der Aufnahme für die Messzelle.

2.2. Steuerungselektronik

Dieser Abschnitt beschreibt die Entwicklung der Elektronik. Die Funktionalität wurde auf verschiedene Module aufgeteilt. Das erleichtert die Platzierung der Elektronik innerhalb des Messroboters, und den Austausch von Komponenten im Fehlerfall. Das Gehirn des Roboters ist ein Arduino Mega 2560 Mikrocontroller Board. Darauf aufgesteckt ist ein sogenanntes Shield, eine selbst entwickelte Platine, das die $\pm 5V$ Spannungsversorgung, die Schnittstellen zu den anderen Platinen und einen SD-Karten Einschub beheimatet. Es gibt 3 Messplatinen, eine für den Kurzschlussstrom der Messzelle, zwei für Temperaturmessungen. Eine Platine überwacht den Ladezustand des Akkus. Die Motorensteuerung ist auf einer eigenen Platine beheimatet.

2.2.1. Motorensteuerung

Eine H-Brücke 2.7 erlaubt es einem Gleichstrommotor sich in zwei Richtungen zu drehen. Dafür müssen immer zwei schräg gegenüber liegende Schalter eingeschaltet sein. Zum Beispiel die Schalter S1 und S4 für die Drehrichtung im Uhrzeigersinn, S2 und S3 für die Drehrichtung gegen den Uhrzeigersinn. Zwei untereinander liegende Schalter dürfen nicht gleichzeitig eingeschaltet werden. Es muss einen hardwaretechnischen Schutz gegen diese Art von Kurzschlüssen geben. Ein softwaretechnischer Schutz reichte dem Autor nicht, nachdem ein früher Prototyp der Motortreibplatine sich in Rauch aufgelöst hatte. Deshalb wurde für den Roboter eine integrierte H-Brücke (NJM2670) verwendet, mit einer logischen Verriegelung, die keinen Kurzschluss produzieren kann. Die Drehzahl bei konstanter Versorgungsspannung lässt sich mit Pulsweitenmodulation variieren.

Die Gleichspannungsmotoren sind Getriebemotoren mit der Bezeichnung RB 35, und einem Übersetzungsverhältnis von 1:200. Die maximale Versorgungsspannung wurde mit 12V angegeben.

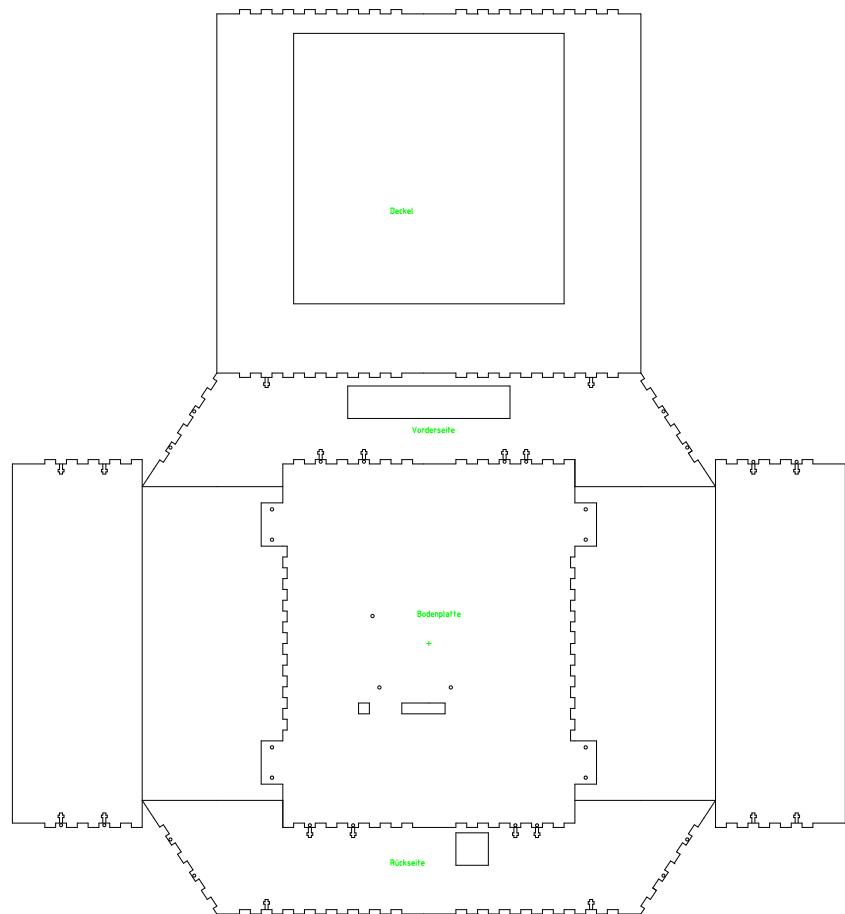


Abbildung 2.5.: Schittvorlage der Chassis

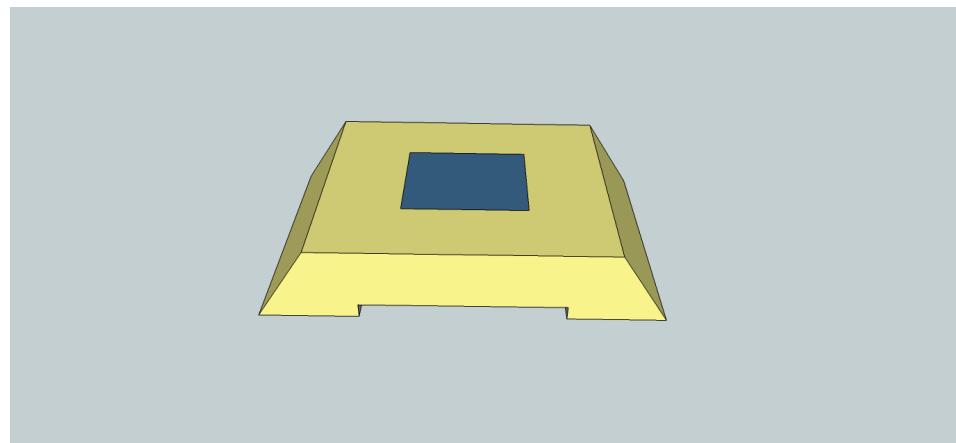


Abbildung 2.6.: 3D-Modell der Chassis

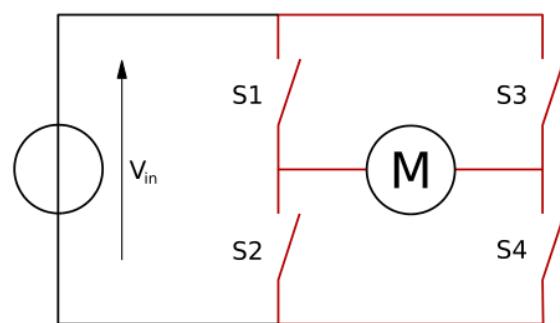


Abbildung 2.7.: H-Brücke

Die Drehzahl beträgt bei 12V Versorgungsspannung 20 Umdrehungen pro Minute. Der Roboter wird mit einem Lithium-Ionen-Akku versorgt. Die Spannung beträgt, abhängig vom Ladezustand zwischen 16,4V und 15,0V. Damit es zu keiner Überlastung der 12V-Motoren, kommt wurde bei der Pulswidtemodulation ein maximales Tastverhältnis von 50% eingehalten.

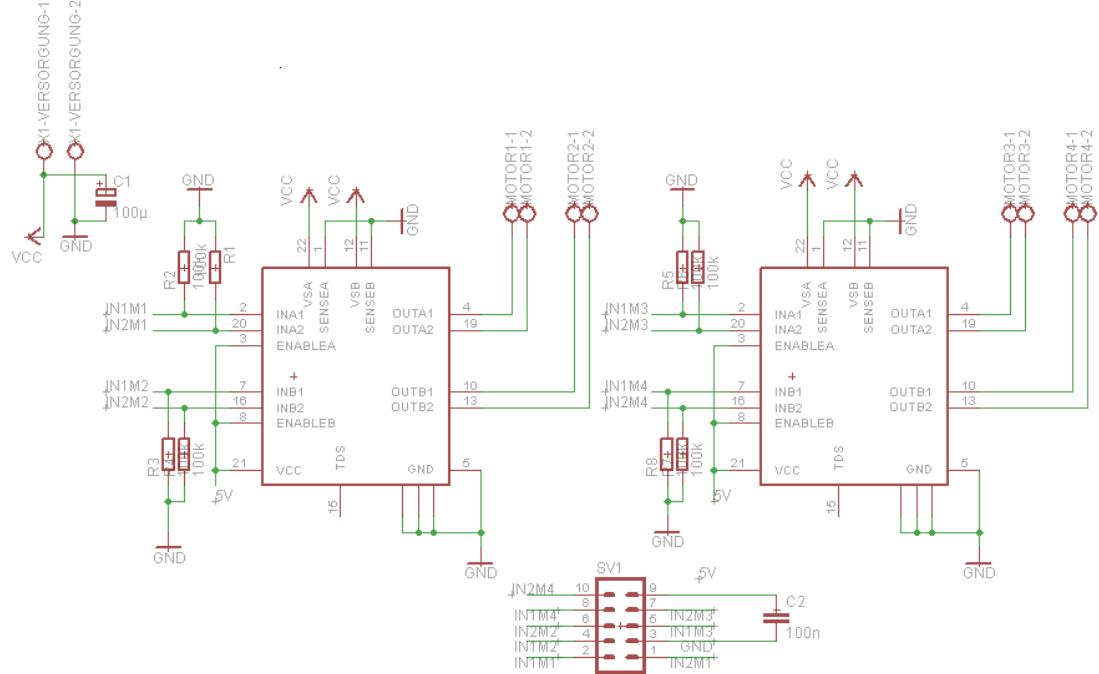


Abbildung 2.8.: H-Brücke Schaltung

2.2.2. Optische Sensorik

Der Roboter ist ein Linienfolger. Damit er einer Linie folgen kann, muss diese zuerst zuverlässig erkannt werden. Als Sensor wurde ein CNY70 verwendet. Der CNY 70 ist ein Reflexionsensor. Der Sensor wird in einem fixen Abstand zu einer Ebene montiert. Eine LED sendet Licht mit 950nm aus, welches an der Ebene reflektiert wird. Damit können Unterschiede des Reflektionkoeffizienten, umgangssprachlich der Farbe, gemessen werden. Die Anordnung 9 solcher Sensoren in einem 3 mal 3 Array kann sowohl vertikale als auch horizontale Linien erkennen, sowie Ecken.

2.2.3. Spannungversorgung

Aku-handling, Überwachung der Akkuspannung, Alarm und Abschaltung bei Unterspannung, Es wird empfohlen vor jeden Messzyklus den Akku zu laden. Ein voller Akku schafft ohne Probleme 10 Messfahrten. Bauteile: Akku, Ladegerät (extern), bequemes und schnelles Laden des Akkus Eine Schmelzsicherung ist direkt im Anschlusskabel des Akkus eingebaut. Der Akku schafft Ströme bis 200A, da er aus dem Modellbausektor kommt. Was für den Roboter völlig überdimensioniert ist. Der maximale Strom aller 4 Motoren und der restlichen Elektronik liegt bei einem Ampere. Die Motoren werden direkt mit der Akkuspannung versorgt. Da realistischerweise die Versorgungs- spannung zwischen 16,4V (Akku voll) und 15,5V (nach vielen Fahrten) liegt, ist der Einfluss auf

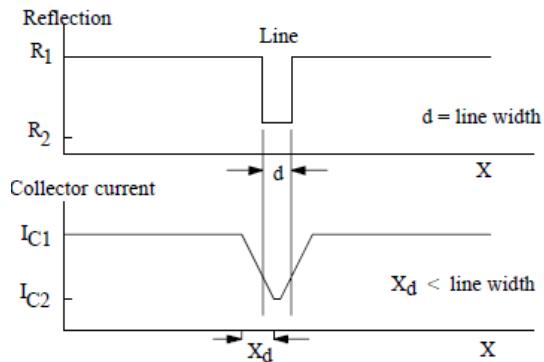


Abbildung 2.9.: Der Strom durch den Phototransistor ist abhängig von der Reflektivität des Untergrundes

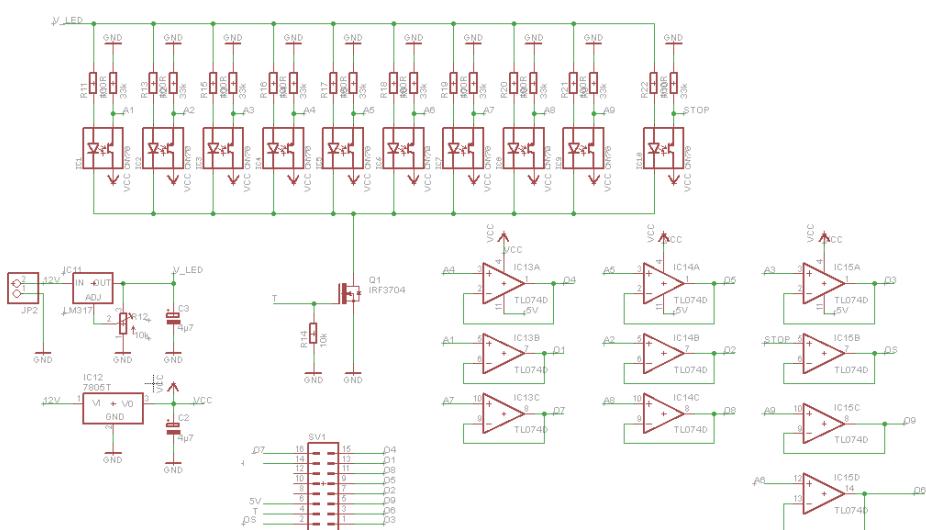


Abbildung 2.10.: Der Schaltplan des Sensorarrays

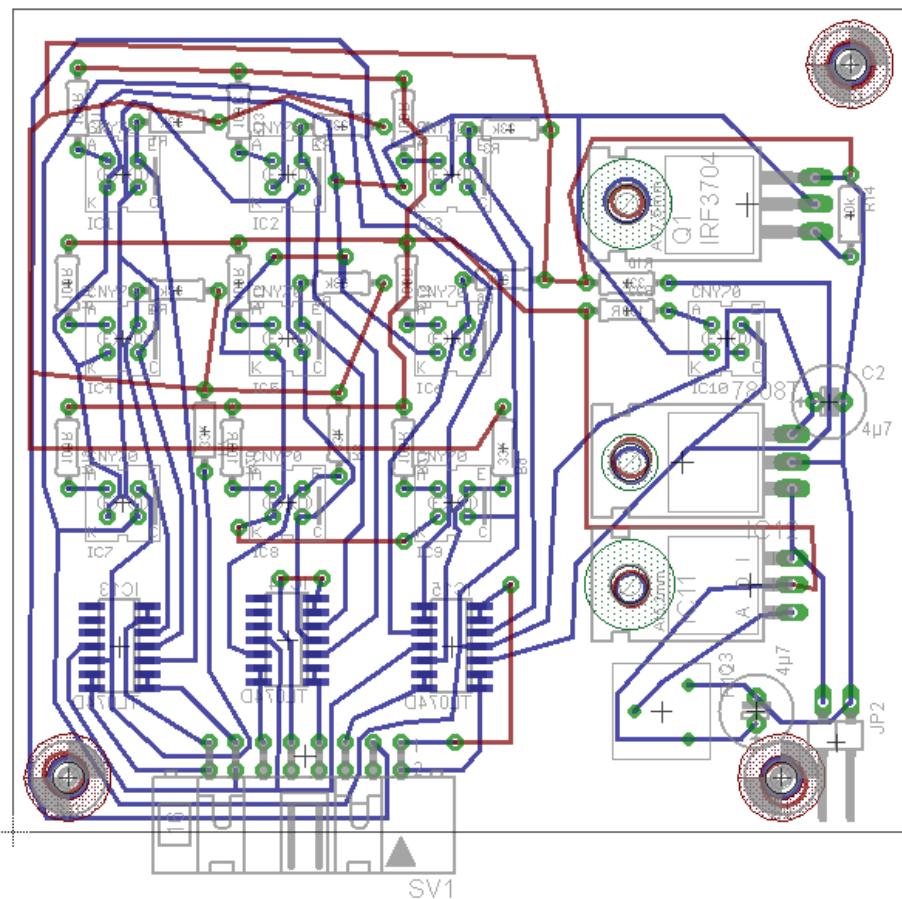


Abbildung 2.11.: Das Layout des Sensorarrays

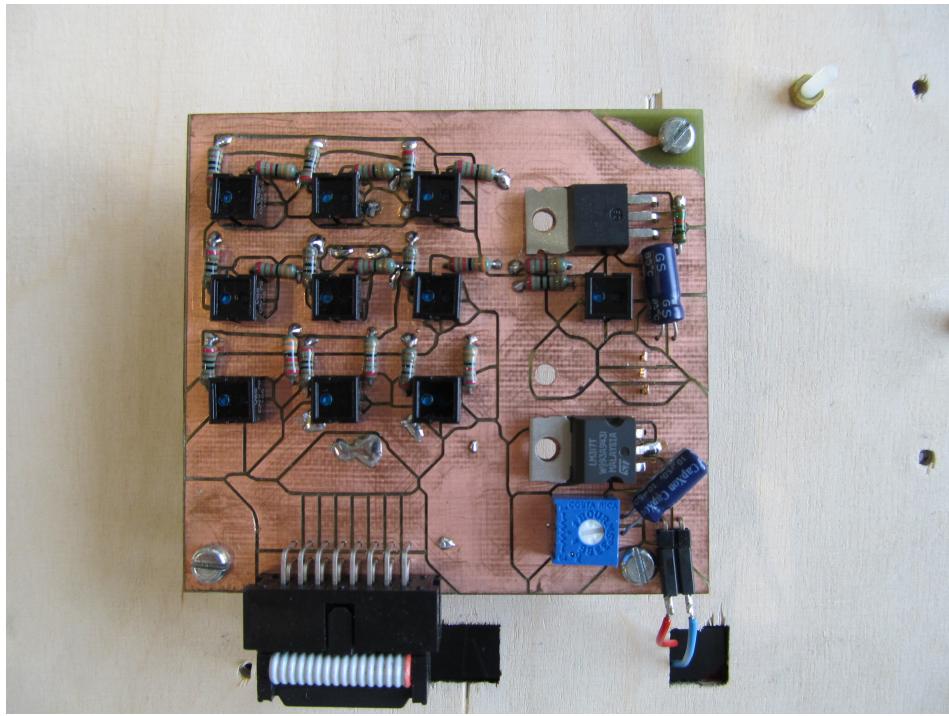


Abbildung 2.12.: Das unter dem Roboter montierte Sensorarrayt

die Drehzahl der Motoren zu vernachlässigen. Die 5 Volt Versorgung des Mikrocontrollers und der OPVs wurde mit einem RECOM R-785.0-1.0 DC-DC Konverter gelöst. Er ist pinkompatibel zu einem 7805, und bietet die Vorteile einer geringen Verlustleistung und einer sehr genauen Ausgangsspannung. Die Analog-Digitalwandlung des Arduino benötigt eine Referenzspannung. Das kann die Versorgungsspannung sein, was aber doch zu ungenau ist. The LT1021 is a precision reference with ultralow drift and noise, extremely good long term stability and almost total immunity to input voltage variations. The reference output will both source and sink up to 10mA. Three voltages are available: 5V, 7V and 10V. The 7V and 10V units can be used as shunt regulators (two-terminal zeners) with the same precision characteristics as the three-terminal connection. Special care has been taken to minimize thermal regulation effects and temperature induced hysteresis.

2.2.4. Mikrokontroller

Arduino (was ist das), Shield, SD, Aref,

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

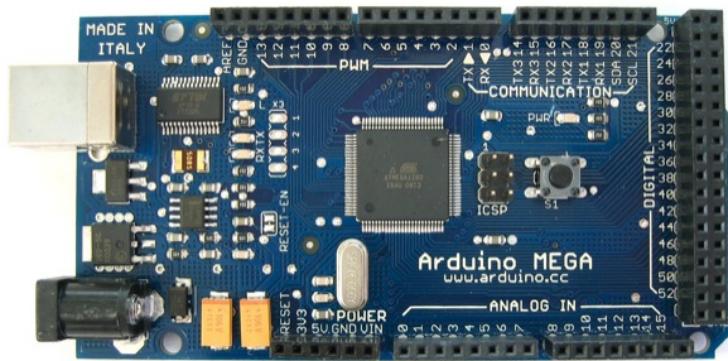


Abbildung 2.13.: Arduino Mega 2560

2.2.5. Temperatursensoren

Vergleich des Stromes durch eine 100 Ohm Präzisionswiderstand (Temperaturstabilität!) mit dem Strom durch einen PT-100. Als Stromquelle diente eine REF200-Stromquelle, welche 2 mal 100 Mikroampere (+- 0.5 Prozent) liefert.

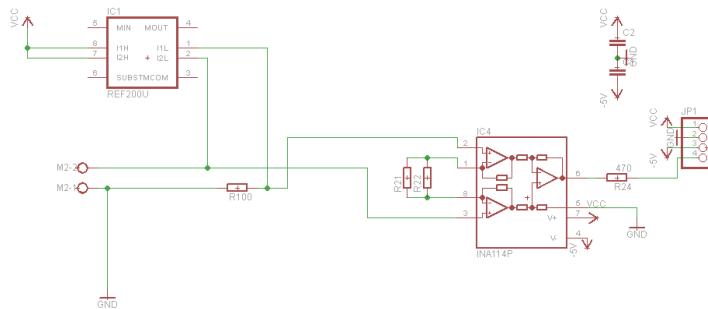


Abbildung 2.14.: Schaltung der Temperaturmessung

2.3. Softwareentwicklung

2.3.1. Entwicklungsumgebung

The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, MaxMSP).

2.3.2. Auswertung optische sensoren

Differentielle Messung: Zwar liegen die optischen Sensoren auf der Unterseite des Roboters, aber da es im Sonnensimulator eine Einstrahlung von 1000 Watt pro Quadratmeter gibt, mit einem Anteil im spektralen Arbeitsbereich der optischen Sensoren CNY70, ist es notwendig den Einfluss des

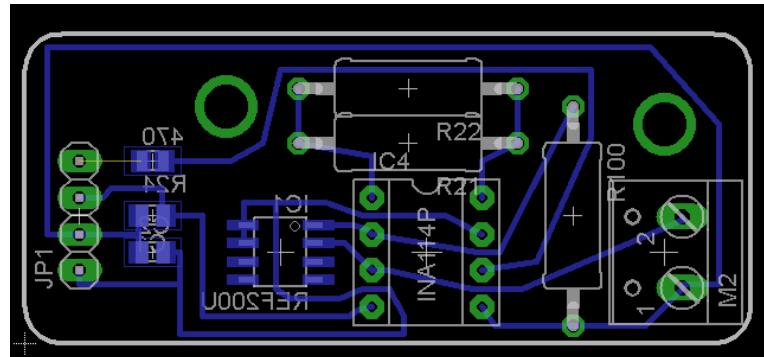


Abbildung 2.15.: Layout der Temperaturmessung

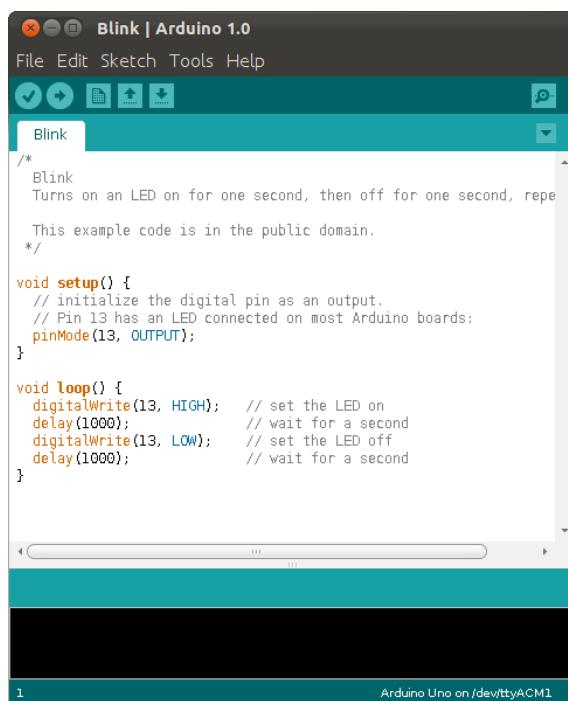


Abbildung 2.16.: Arduino Entwicklungsumgebung

Streulichtes zu unterdrücken. Dazu wird einmal mit und einmal ohne eingeschalteter Infrarot-LED gemessen, dann der Dunkelwert vom Hellwert abgezogen.

Mathematik Auswertung, Anhand der Werte der einzelnen optischen Sensoren wird die Lage der Linie relativ zum Mittelpunktes des Arrays bestimmt. Rohefassung: Die Methode der kleinsten Quadrate (engl.: method of least squares) ist das mathematische Standardverfahren zur Ausgleichungsrechnung. Dabei wird zu einer Datenpunktewolke eine Kurve gesucht, die möglichst nahe an den Datenpunkten verläuft. Die Daten können physikalische Messwerte, wirtschaftliche Größen oder Ähnliches repräsentieren, während die Kurve aus einer parameterabhängigen problemangepassten Familie von Funktionen stammt. Die Methode der kleinsten Quadrate besteht darin, die Kurvenparameter so zu bestimmen, dass die Summe der quadratischen Abweichungen der Kurve von den beobachteten Punkten minimiert wird. Die Abweichungen werden Residuen genannt. Folge der Linie: Der Roboter fährt standardmäßig gerade aus. Wird ein Versatz zur Linie erkannt, wird dieser Versatz durch eine Bewegung normal zur Linie minimiert. Eine Verdrehung zur Linie wird durch Verdrehen des Roboters korrigiert. Ein minimaler Versatz, bzw. eine minimale Verdrehung ist dabei zulässig. Das vorwärts- und rückwärts-Fahren funktioniert dabei besser als das nach rechts-Fahren. Mecanum-Räder haben eine Vorzugsrichtung.

Behandlung der Ecken:

Ecken müssen als solche erkannt werden, bevor sie die normale Fahrregelung stören. Wird eine Ecke erkannt wird der normale Fahrmodus unterbrochen, und in einem speziellen Eckenmodus gewechselt. Es gibt zwei Arten von Ecken: - einmal die klassische Ecke beim Wechsel vom Vor- bzw. Rückwärtsfahren nach rechts. - Die nach Rechts-Fahren Bahn endet, anstatt in einer Ecke in die Vor- bzw. Rückwärtsbahn zu enden. Damit ist auch das um die Ecke fahren auch unterschiedlich. Im Vor- bzw. Rückwärts-Fahren wird die Fahrt ohne Regelung solange fortgesetzt bis die obere bzw. untere Zeile der Sensormatrix schwarz sieht, dann fährt der Roboter ohne Regelung einige hundert Millisekunden nach rechts. Erst dann wird in den nach Rechts-Fahr-Modus gewechselt. Für den Rechts-fahr Modus endet die Linie abrupt. Der Roboter fährt einige hundert Millisekunden vor bzw. zurück. Dann steht er auf der Führungslinie. Bevor in den eigentlichen Vor- bzw. Rückwärtsfahrmodus gewechselt wird, wird der Roboter zur Linie hin ausgerichtet. Die Unterschiedliche Behandlung der Ecken ist notwendig, weil im Rechtsfahren größere Abweichungen von der Ideallinie auftreten als im Vor- und Zurückfahren.

Haltepunkte: Haltepunkte werden mit dem Haltepunktsensor erkannt, wenn dieser den Schwellwert für weiß überschreitet. Damit ein Haltepunkt nicht mehrfach erkannt werden kann, gibt es eine Verriegelung in der Software, die einen neuen Haltepunkt erst wieder zulässt wenn der Haltepunktsensor in der zwischen Zeit schwarz gesehen hat.

2.3.3. Auswertung ADCs

Aufgrund von Rauschen der ADC-Werte, wurden alle Messwerte über 500 Einzelmessungen gemittelt. Der notwendige Messzeit dafür beträgt unter einer halben Sekunde.

2.3.4. Programmablauf

Das Programm startet im Modus Start, der Roboter steht, und wartet 60 Sekunden, eine Pufferzeit damit nach dem Einschalten des Roboters die Lade des Sonnensimulator hineingeschoben und die Klappe geschlossen werden kann. Nach Ablauf dieser 60 Sekunden wechselt das Programm in den Modus Vorwärts. In diesem Modus fährt der Roboter in Vorwärtsrichtung. Abweichungen von der Idealspur werden erkannt und korrigiert. Wird mit dem Stopsensor eine Stopmarkierung erkannt,

wechselt das Programm in den Messmodus.

3. Kalibration

Dieses Kapitel beschreibt die Kalibrierung der Messsensoren. Es gibt drei Messplatinen, eine für die Messung des Kurzschlussstromes der Messzelle, und zwei identisch aufgebaute Messplatinen zur Messung der Zelltemperatur beziehungsweise der Umgebungstemperatur. Die Strom-Messplatine wandelt den Kurzschlussstrom der Messzelle in eine Spannung um, die in einem für den ADC des Arduino auswertbaren Bereich (0 bis 5 Volt) ist. Die beiden Messschaltungen zur Temperatormessung wandeln die Größe der temperaturabhängigen Messwiderstände, jeweils ein Pt-100, in eine Spannung um. Die ADC-Messwerte wurden über die USB-Schnittstelle des Roboters ausgelesen. Dafür benötigt der Roboter ein einfaches Kalibrierprogramm.

3.1. Temperatursensoren

Die Temperatur der Messzelle wird mit einem von unten aufgeklebten flächigen Folienmesswiderstand gemessen. Ein zweiter Pt100, in kompakter Dünnschichtbauweise ausgeführt, misst die Temperatur der Umgebung. Beide Messkreise sind identisch aufgebaut, dennoch führen Bauteiltoleranzen zu leicht unterschiedlichen Verstärkungen. Für die Kalibration der Temperaturmesselektronik wurden die Pt-100 Widerstände durch einen hoch genauen, einstellbares und kalibriertes Messnormal ersetzt. Der Widerstand wurde im Bereich von 100 bis 122 Ω in 1 Ω Schritten verändert, was einer Temperatur von 0 bis 55 °C entspricht. Die ADC-Werte wurden über die USB-Schnittstelle des Arduino ausgelesen. Es zeigte sich, dass die beiden Schaltungen leicht unterschiedliche Verstärkungen haben. Daher mussten jede Temperatormessschaltung separat kalibriert werden.

Obwohl der 10-Bit ADC des Arduino einen Maximalwert von 1023 hat, ist bedingt durch den Einsatz des Instrumentenverstärkers INA114, der maximale ADC-Wert bei 857, was einer Spannung von 4,18V entspricht.

Die mit Excel gewonnenen Gleichungen der Ausgleichsgerade für Messschaltung A (siehe Abbildung 3.1) und B (siehe Abbildung 3.1) werden verwendet um aus den ADC-Werten die dazugehörigen Widerstandswerte zu berechnen.

$$R_A(ADC) = \frac{ADC_A + 4039,9}{40,107} \quad (3.1)$$

$$R_B(ADC) = \frac{ADC_B + 4023,6}{40,027} \quad (3.2)$$

3.2. Messzelle

Der Kurzschlussstrom der Messzelle wurde im Flasher (siehe Abbildung 3.3) über einen weiten Temperaturbereich gemessen (siehe Abbildung 3.4). Es wurde mit der Berger Messlast [2] gemessen. Bei der Messung wurden die Leitungswiderstände der Messkabel minimiert, indem Kabel mit

R / Ω	ADC Wert
101	12
102	53
103	93
104	133
105	173
106	212
107	253
108	293
109	332
110	374
111	412
112	454
113	493
114	533
115	573
116	613
117	653
118	694
119	734
120	775
121	815
122	857

Tabelle 3.1.: Die Messwerte der Kalibrierung Messschaltung A

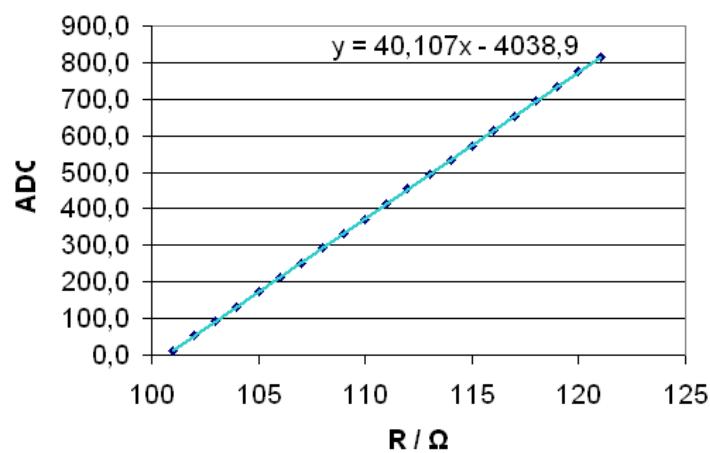


Abbildung 3.1.: Ausgleichsgerade Temperaturmessung A

R / Ω	ADC Wert
100	0
101	20
102	60
103	99
104	139
105	179
106	219
107	259
108	299
109	339
110	379
111	420
112	460
113	499
114	539
115	579
116	619
117	659
118	700
119	740
120	780
121	820
122	857

Tabelle 3.2.: Die Messwerte der Kalibrierung Messschaltung B

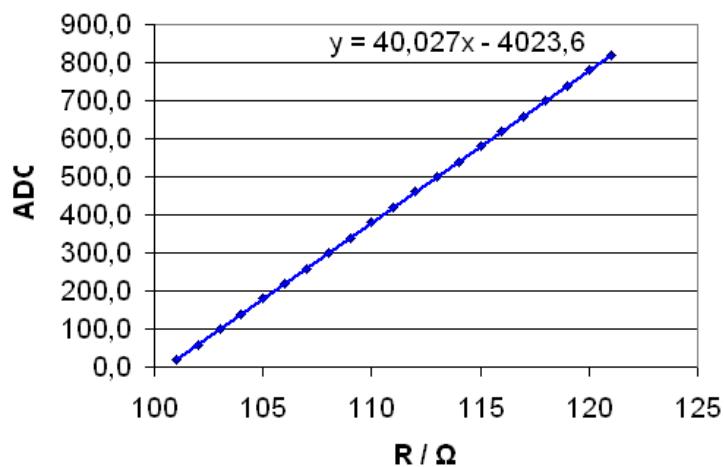


Abbildung 3.2.: Ausgleichsgerade Temperaturmessung B

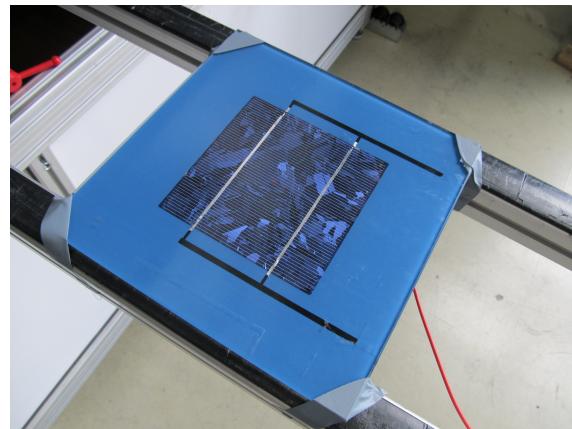


Abbildung 3.3.: Ein Photo der Messzelle

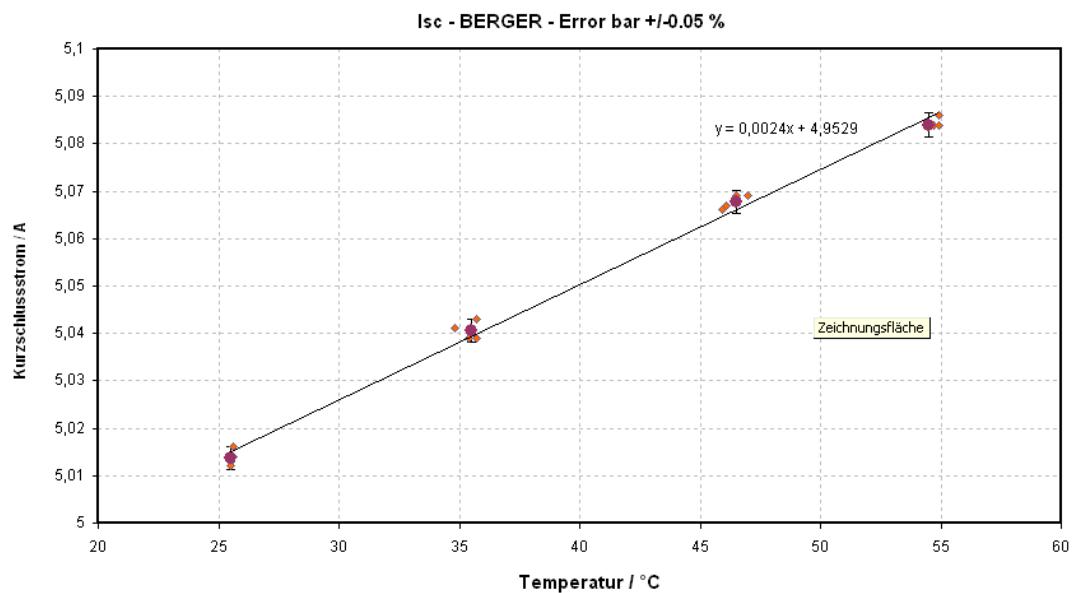


Abbildung 3.4.: Abhangigkeit des Kurzschlussstromes von der Temperatur

hohem Querschnitt (6mm^2) verwendet wurden, und die Kabel möglichst kurz gehalten wurden ($<1,5\text{m}$).

$$I(T) = 4,9529 + 0,0024 * T \quad (3.3)$$

$$I(T) = 4,9529 * (1 + 0,00048 * T) \quad (3.4)$$

3.3. Strommessung

Der Messroboter wurde für die Kalibrierung der Strommessung in eine Klimazelle platziert. Die Temperatur der Klimazelle wurde im Bereich von 8 bis 35°C variiert. Durch den Messwiderstand der Strommessung wurde ein Strom im Bereich von 3 bis 8 A geschickt.

T/ $^\circ\text{C}$	I/mA	ADC Wert
24,4	3997	472
	6001	712
	8002	854
35,3	3999	473
	6001	712
	7999	859
35,3	3999	472
	6001	710
	8001	852
16,4	2004	234
	3005	353
	4003	473
	5003	591
	6005	711
8	1993	233
	2995	352
	4001	472
	5000	590
	6005	710

Tabelle 3.3.: Kalibrierung Strommessung

$$I(ADC) = \frac{ADC + 4,5766}{119,2} \quad (3.5)$$

3.4. Thermische Stabilität der Temperaturmessung

Zur Bestimmung der Temperaturstabilität wurde der gesamte Roboter in ein Klimakammer gestellt. Der Pt100 wurde durch einen temperaturstabilen 110Ω simuliert. Es gab zwei Durchgänge dieses Versuches. Bei der ersten Messung zeigte sich, dass eine der beiden Messplatinen nicht temperaturstabil war. Durch das Tauschen der Ref200 Stromquelle dieser Platine konnte das Problem gelöst werden.

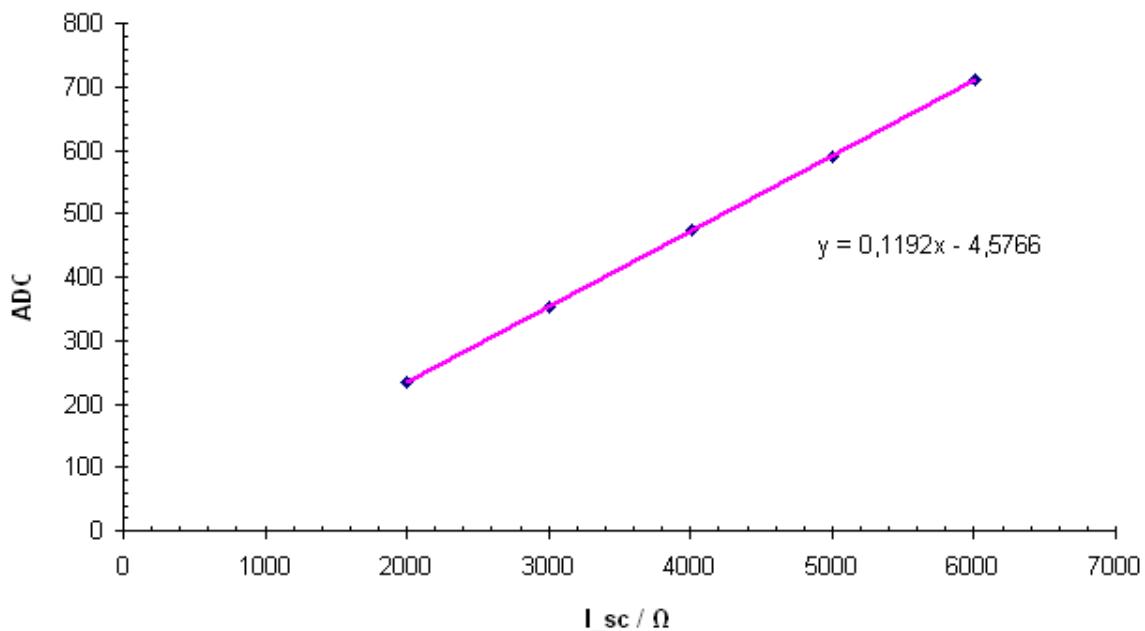


Abbildung 3.5.: Lineraer Zusammenhang zwischen Strom und dem ADC-Wert

T / °C	ADC Platine A	ADC Platine B
11,7	368	407
13,0	368	406
13,7	368	406
14,5	368	405
15,5	368	405
16,6	368	405
17,5	368	404
23,5	368	403
27,5	368	402

Tabelle 3.4.: Temperaturabhängigkeit der Temperaturmessung

T in °C	ADC Platine A	ADC Platine B
30,7	368	380
20,1	368	379
9,2	368	378

Tabelle 3.5.: Temperaturabhängigkeit der Temperaturmessung

Durch den Austausch der Referenzstromquelle der Platine B hat sich auch der ADC-Wert gegenüber der ersten Messung geändert. Die Kalibrierwerte der Platine B 3.2 sind nach diesem Austausch aufgenommen worden. Da die Platine A temperaturstabilier als die Platine B ist, wurde Platine A zur Messung der Zelltemperatur ausgewählt, die Platine B wurde für die Messung der Umgebungstemperatur verwendet.

4. Messung

Diese Kapitel beschreibt die Messvorbereitung, den Messaufbau, die eigentliche Messung und die Auswertung der Messdaten.

4.1. Messaufbau

Dieser Abschnitt behandelt den Messaufbau. Die Messung der Bestrahlungsstärkeverteilung findet im Sonnensimulator statt. Zuerst muss eine Messebene geschaffen werden. Die, im normalen Messalltag vermessenen, Module liegen auf verschiebbaren Metallschienen auf. Damit der Roboter fahren kann, braucht er eine ebene Fläche. Die Messebene wird aus 3 Holzplatten gebildet, diese sind auf der Oberseite schwarz gefärbt und mit einer weißen mäanderförmigen Spur versehen. Die Platten hängen durch, dadurch ist die Messebene nicht eben. Schlimmer ist der nicht gleiche Durchhang verschiedener Platten. So kommt es zu Stufen zwischen den Platten. Um diese Stufen zu verringern wurde auf der Unterseite der Platten mit dem geringeren Durchhang eine kleine Aluplatte befestigt um eine Aufflage für die andere Platte zu schaffen. Eine zu große Stufe behindert den Roboter beim Fahren.

Foto Platten , Foto Auflage

4.1.1. Platten

Im Sonnensimulator 2 Platten. Auf den 3 Platten ist die Messbahn aufgemalt. Die Platten müssen eng aneinander anliegen. Damit keine zu groß Stufe wegen unschiedlicher Duschbiegung der Platten entsteht ist auf der Rückseite der Holzplatten Aluminiumbleche befestigt, auf einer Seite angeschaubt, die danebenliegende Platte liegt darauf. Da die Breite aller 3 Platten geringer ist als die Breite der Lade des Sonnensimulators, werden, damit die Messungen zu verschiedenen Zeitpunkten verglichen werden können, die Platten so weit wie möglich nach rechts geschoben. Am Rand liegen die Platten auf Aluschienen auf. Die Messpunkte liegen im Abstand von 15 cm, damit ist auch der Abstand zwischen den Messbahnen 15 cm, der Roboter muss nur 15 cm seitwärts fahren. Das seitwärts fahren funktioniert nicht so einwandfrei wie das Vor/Zurückfahren, die Bewegungsrichtung ist bezogen bezogen auf die Orientierung der Räder.

Die Übergänge der Messbahn von einer Platte zur nächsten sind mit weißem Isolierband der Breite 14mm zu überbrücken. Beschädigung der Messbahn, welche durch unsanftes Handling der Platten entstehen können, sind ebenfalls mit weißem Isolierband zu Überkleben.

4.1.2. Ablauf

Vor der Messung ist der Akku des Roboters voll zu laden. Der Roboter ist auf mechanische Beschädigung zu untersuchen, eventuell ist ein Rad locker, dann lässt es sich entlang der Achse verschieben. Zuallererst sind die Platten auf Beschädigung zu untersuchen. Beschädigungen der Messbahn sind mit einem weißen Isolierband zu überkleben. Beschädigungen des Holzes direkt neben der Messbahn sind mit schwärzer Farbe auszubessern. Stellen, die der Sensor nicht sehen

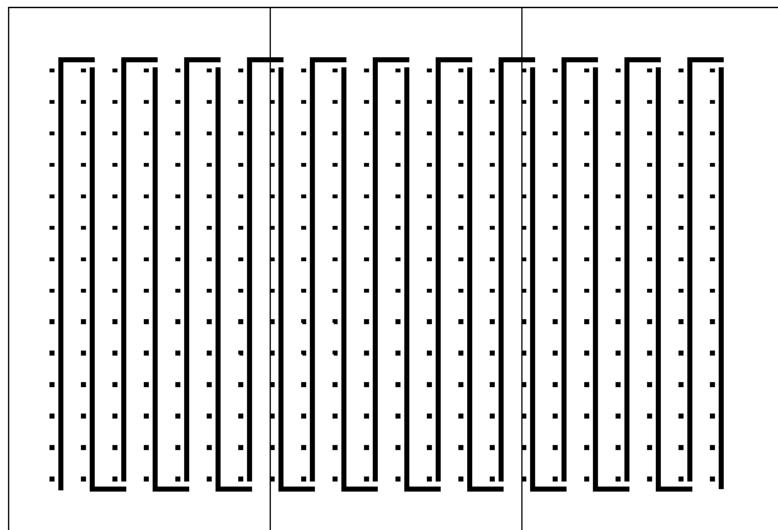


Abbildung 4.1.: Messbahn auf den Platten

kann, sind egal. Die Platten sind in den Einschub des Sonnensimulators zu legen. Das ist ein Puzzle und nicht schwer. Beginnend rechts. Die Übergänge der Messbahn von einer Platte zur nächsten sind mit weißem Isolierband der Breite 14mm zu überbrücken. Beschädigung der Messbahn, welche durch unsanftes Handling der Platten entstehen können, sind ebenfalls mit weißem Isolierband zu Überkleben.

Roboter wird auf das linke vordere Ende der Messbahn gestellt. Der Roboter muss dabei auf der Messbahn stehen. Kleinere Abweichungen der Idealposition werden beim losfahren korrigiert. Vor der eigentlichen Messfahrt ist eine Testfahrt mit geöffneter Lade des Sonnensimulators zu empfehlen. Besonders die Übergänge von einer Platte zur nächsten können Probleme schaffen. Nach der bestandenen Testfahrt wird der Sonnensimulator eingeschaltet. Bis zur Messung sind 20 bis 30 Minuten zu warten, bis die Temperaturen im Simulator stabil sind. Der Roboter kann während dieser Zeit im Sonnensimulator stehen. Die Messzelle sollte allerdings abgeschattet werden, damit diese sich nicht aufheizt. Zum Starten wird der Schalter von 0 auf 1 umgelegt. Damit wird der Mikrocontroller mit Spannung versorgt. Das Programm startet. Nach einer Pause von 60 Sekunden, die zum Schließen des Sonnensimulators notwendig ist, fährt der Roboter los. Eine Messfahrt dauert etwa 13 Minuten. Nach Ablauf dieser Zeit wird der Roboter entnommen und die Daten der SD-Karte entnommen, oder der Roboter wird auf die Startposition gestellt um weitere Messfahrten durchzuführen.

4.2. Auswertung

Die Messwerte werden in einer Textdatei auf einer SD-Karte gespeichert. Die einzelnen Messwerte sind dabei durch einen Tabulator getrennt. Die Tabelle 4.1 zeigt wie die Daten auf der SD-Karte gespeichert sind. Es sind nicht alle Werte für die Auswertung notwendig. Der Erste Wert ist die Zeit in Millisekunden, die seit dem Einschalten des Roboters vergangen ist. Die nächsten 3 Werte sind Werte des Analogdigitalwandlers im Bereich von Null bis 1023, diese Werte werden ohne

Umrechnung direkt auf die SD Karte gespeichert. Es sind 308 Messpunkte, 14 mal 22, es kann vorkommen, am Übergang von einer Platte zur anderen, dass ein Messpunkt ausgelassen wird. Das ist erkennbar wenn nicht 308 Messwerte im datalog-File sind. Um den Fehler leicht zu finden wurde Spalte und Reihe mit aufgezeichnet. Sind in einer Spalte nur 13 Messwerte, fehlt in dieser Spalte ein Wert. Anhand der mit aufgezeichneten Zeit kann der fehlende Messpunktes lokalisiert werden. Das geschieht nicht automatisch Der Messwert wird anhand der neben liegenden Werte geschätzt. Alternativ werden nur Messfahrten mit allen Punkten ausgewertet.

$\frac{t}{ms}$	I_{SC}	T_{Zelle}	$T_{Umgebung}$	Messpunkt	Messpunkt 2	Messreihe
165515	466	437	237	47	5	4
167523	479	437	235	48	6	4
169551	471	437	228	49	7	4

Tabelle 4.1.: Ein Ausschnitt der Messwertedatei

Im Zuge der Auswertung wird ein eindimensionales Array von 308 Messwerten in ein zweidimensionales Array von 14 mal 22 Zahlen umgewandelt. Dazu wird der erste bis zum 14. Wert des eindimensionalen Arrays zur ersten Reihe des zweidimensionalen Arrays. Der 15. bis zum 28. Wert des eindimensionalen Arrays bilden die zweite Reihe des zweidimensionalen Arrays, allerdings in umgekehrter Reihenfolge. So bildet die Reihenfolge der Messwerte des zweidimensionalen Arrays die Messfahrt des Roboters ab.

Aus der Matrix
$$\begin{pmatrix} 1 \\ \vdots \\ \vdots \\ 308 \end{pmatrix}$$
 wird die Matrix

$$\begin{pmatrix} 1 & \dots & \dots & 14 \\ 28 & \dots & \dots & 15 \\ 29 & \dots & \dots & 42 \\ \vdots & \dots & \dots & \vdots \\ 298 & \dots & \dots & 308 \end{pmatrix}$$

Der ADC Wert des Kurschlusstromes wird in Ampere umgerechnet. Verwendet wird die in der Kallibrierung gewonnene Formel.

4.3. Vermessung der Ausleuchtung einzelner Lampen

Die geringe Messzeit des Roboters, verglichen mit der mechanischen Methode, macht es möglich die Bestrahlungsstärkeverteilung der einzelnen Lampen in realistischer Zeit zu bestimmen. Veränderbar sind der Abstand zwischen Lampenfeld und Prüfebene. Die Leistungen der Lampen lassen sich einzeln steuern.

4.4. Schlussfolgerung

Der Roboter erleichtert die Messung der Bestrahlungsstärkemessung enorm. Durch die kurze Messdauer ist es möglich die Bestrahlungsstärke in periodischen Zeitabständen zu vermessen, um so

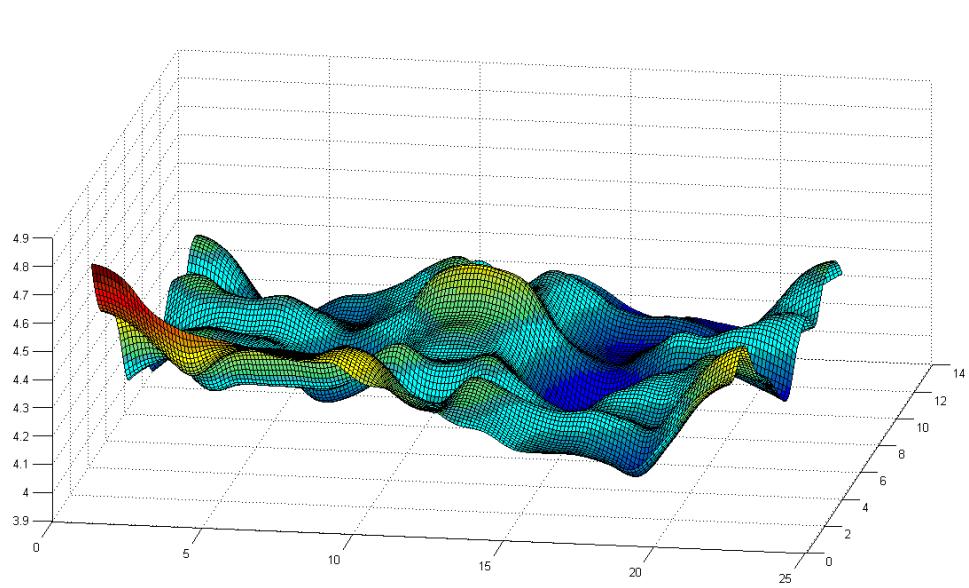


Abbildung 4.2.: Verteilung des Kurzschlusstromes in Ampere abhängig von der Position

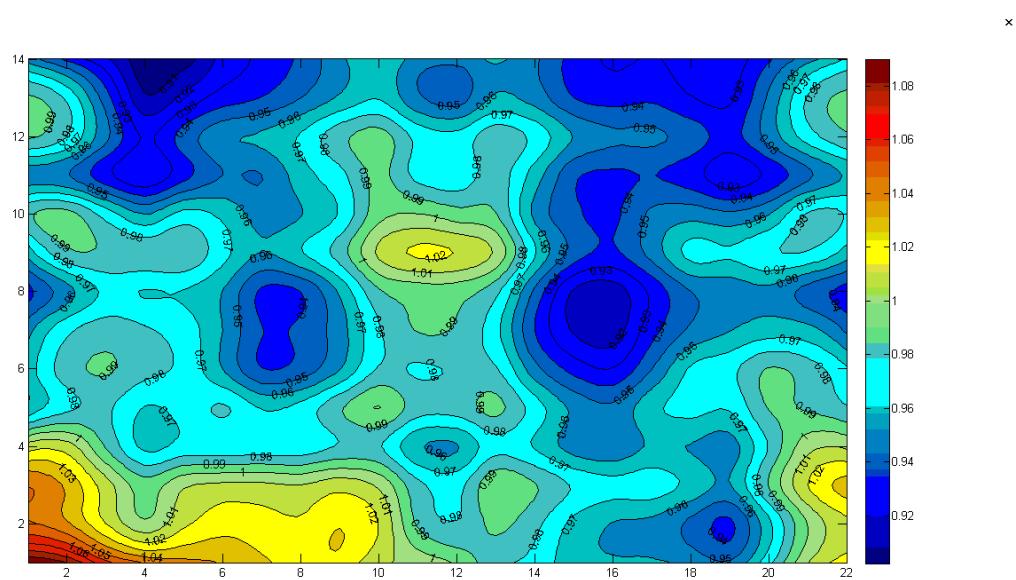


Abbildung 4.3.: Relative Verteilung der Einstrahlung

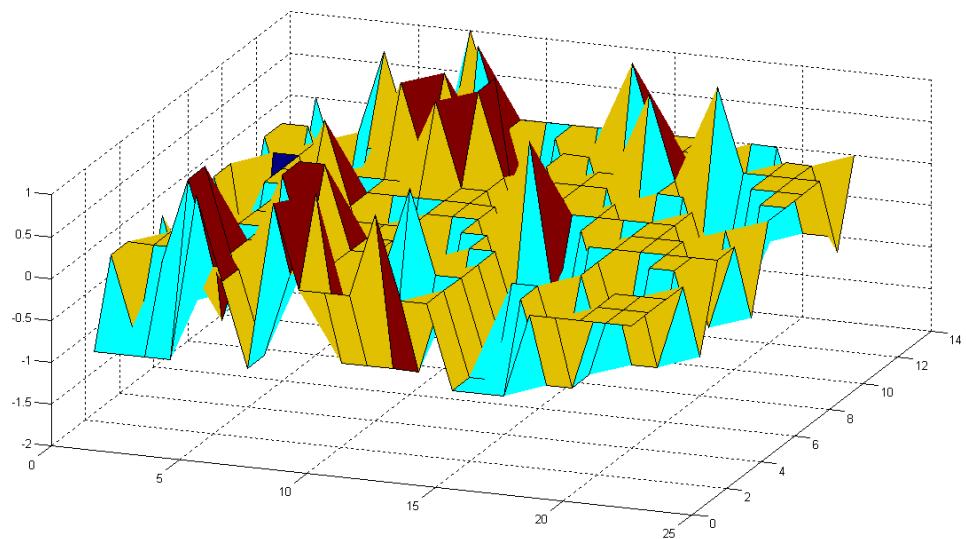


Abbildung 4.4.: Vergleich zweier Messungen

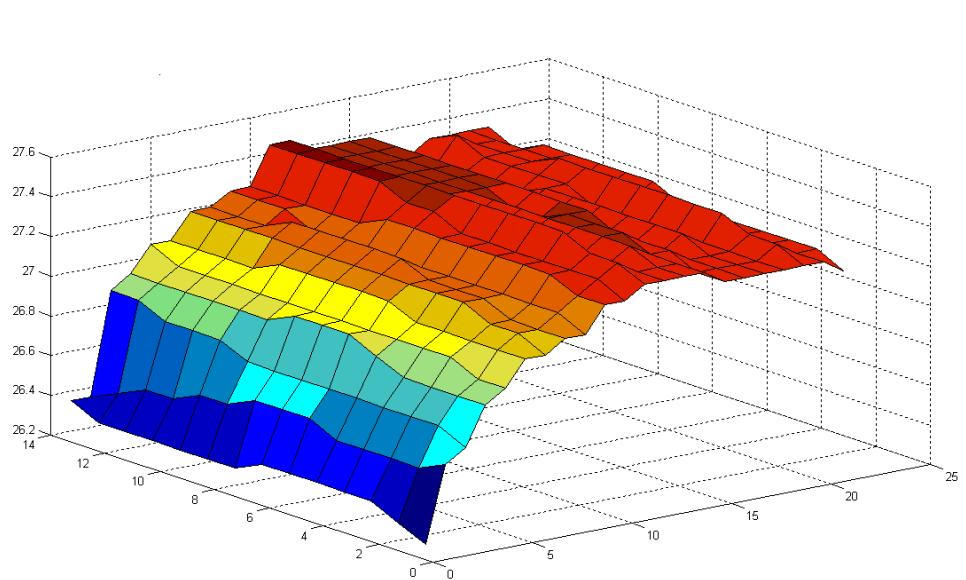


Abbildung 4.5.: Temperatur der Zelle

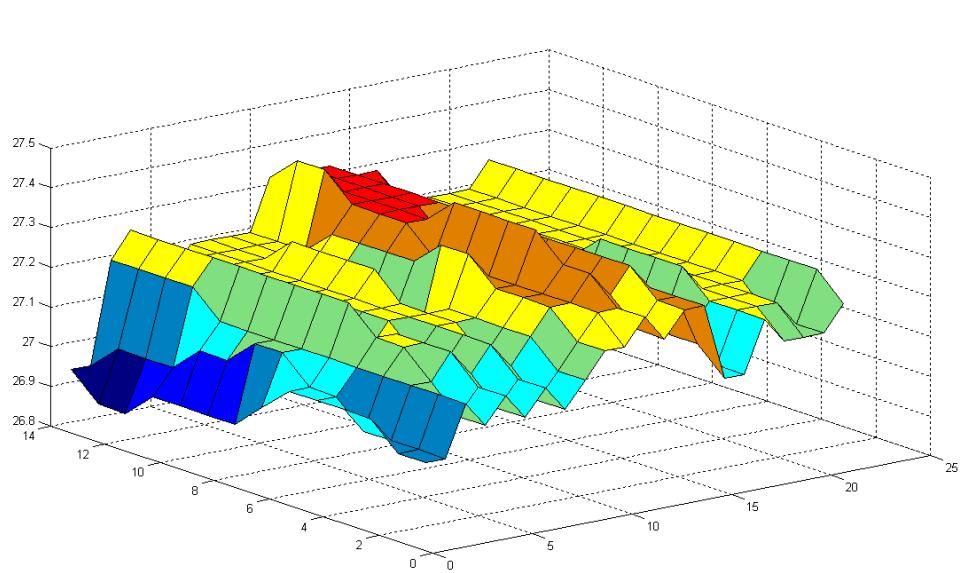


Abbildung 4.6.: konstante Temperatur der Zelle

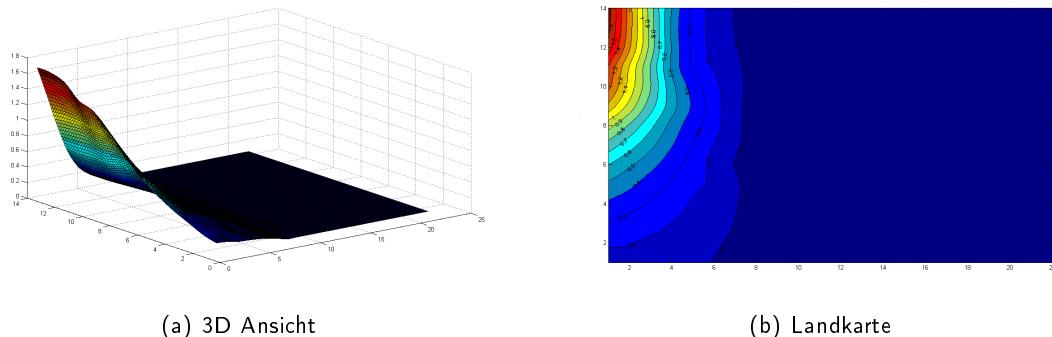


Abbildung 4.7.: Lampe E1

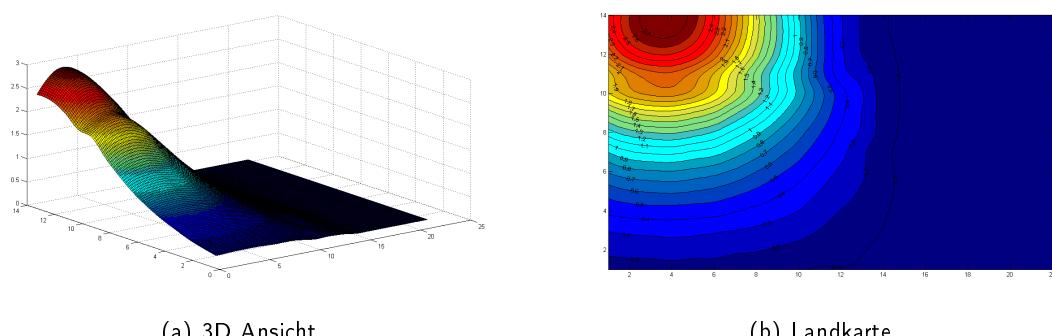
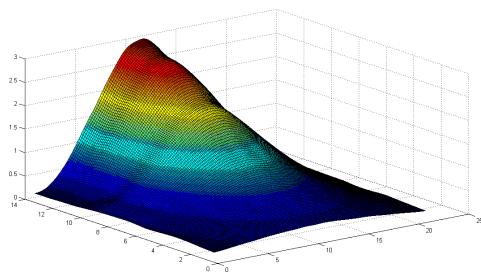
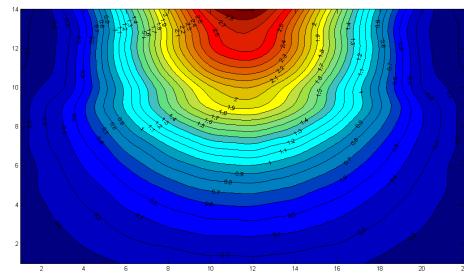


Abbildung 4.8.: Lampe E2

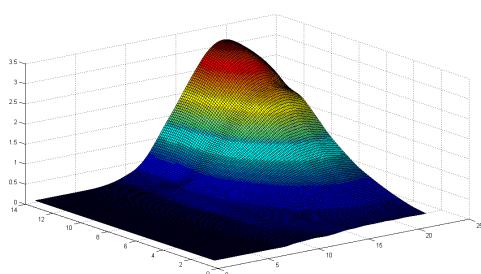


(a) 3D Ansicht

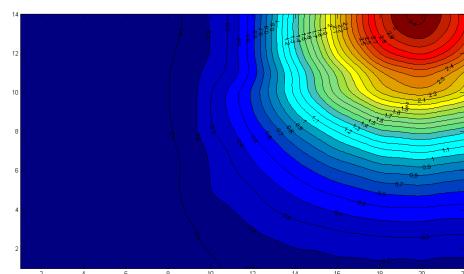


(b) Landkarte

Abbildung 4.9.: Lampe E3

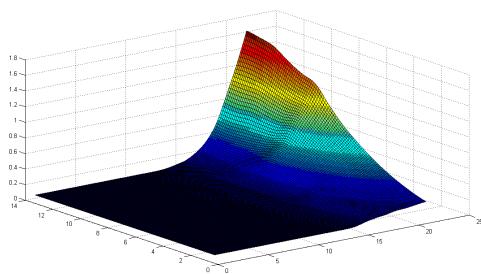


(a) 3D Ansicht

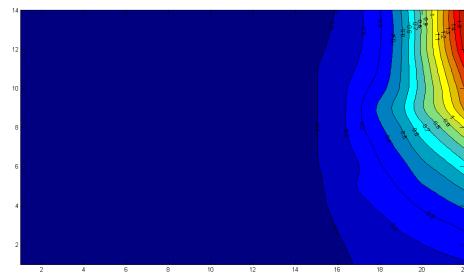


(b) Landkarte

Abbildung 4.10.: Lampe E4

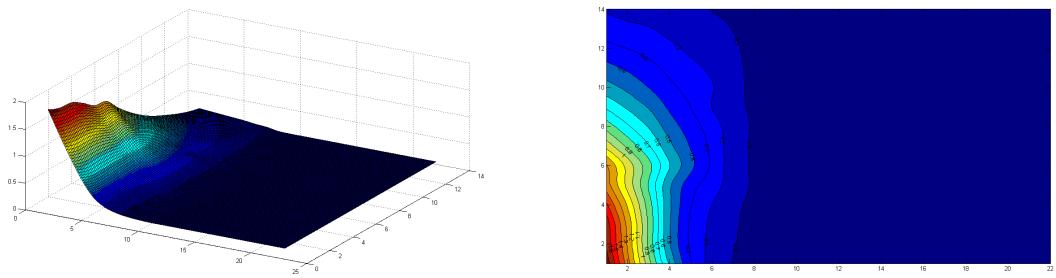


(a) 3D Ansicht



(b) Landkarte

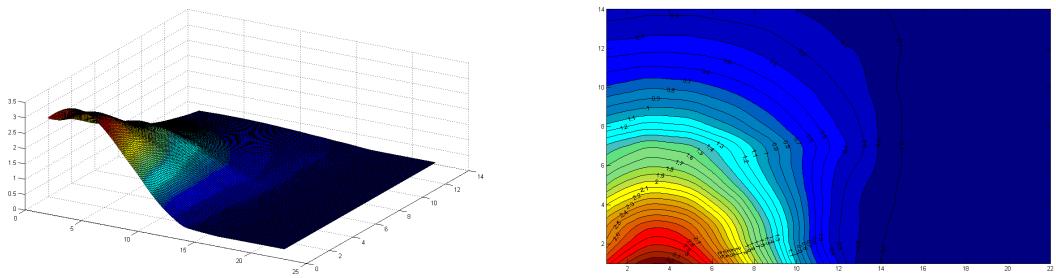
Abbildung 4.11.: Lampe E5



(a) 3D Ansicht

(b) Landkarte

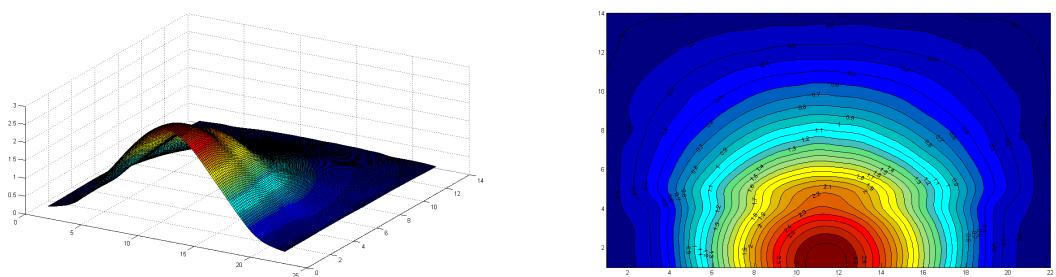
Abbildung 4.12.: Lampe E6



(a) 3D Ansicht

(b) Landkarte

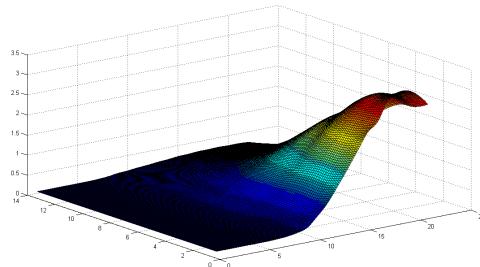
Abbildung 4.13.: Lampe E7



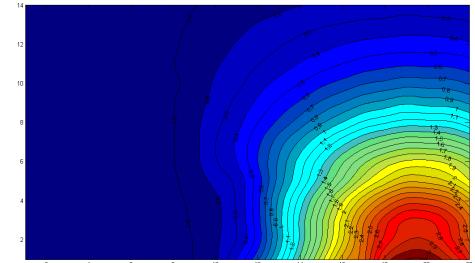
(a) 3D Ansicht

(b) Landkarte

Abbildung 4.14.: Lampe E8

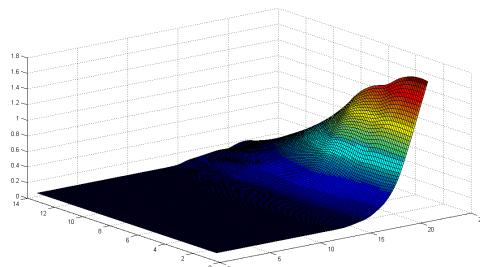


(a) 3D Ansicht

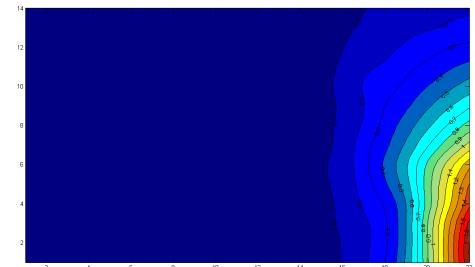


(b) Landkarte

Abbildung 4.15.: Lampe E9



(a) 3D Ansicht



(b) Landkarte

Abbildung 4.16.: Lampe E10

die Alterung der Lampen zu dokumentieren. Diese Art von Messroboter kann in jedem statio-nären Sonnensimulator entsprechender Größe zur Vermessung der Bestrahlungsstärkeverteilung verwendet werden.

Literaturverzeichnis

- [1] "Technology Roadmaps: Solar photovoltaic energy." International Energy Agency, 2010, http://www.iea.org/publications/freepublications/publication/pv_roadmap.pdf [Zugang am 21.5.2012].
- [2] "PSL 8 - Load and Measuring Device," Datenblatt, http://www.bergerlichttechnik.de/resources/Berger_Lichttechnik_PSL8.pdf [Zugang am 15.5.2012].

Abbildungsverzeichnis

1.1. Sonnenspektrum	2
1.2. Sonnensimulator	2
1.3. Schematischer Aufbau einer Solarzelle	3
1.4. Das Eindiodenmodell einer Solarzelle	4
1.5. Dunkel- und Hellkennlinien	5
1.6. Abhangigkeit des MPP von der Einstrahlung	5
2.1. Einige mogliche Fahrmanoer eines Mecanum-Rad-Roboters	7
2.2. Die Felge des Mecanum Rades	8
2.3. Eine der 16 Rollen pro Reifen	8
2.4. Ein gedrucktes Rad samt Rolle	8
2.5. Chassis	10
2.6. Chassis	11
2.7. H-Brucke	11
2.8. H-Brucke Schaltung	12
2.9. Der Strom durch den Phototransistor ist abhangig von der Reflektivitat des Untergrundes	13
2.10. Der Schaltplan des Sensorarrays	13
2.11. Das Layout des Sensorarrays	14
2.12. Das unter dem Roboter montierte Sensorarray	15
2.13. Arduino Mega 2560	16
2.14. Schaltung der Temperaturmessung	16
2.15. Arduino Mega 2560	17
2.16. Arduino Entwicklungsumgebung	17
3.1. Ausgleichsgerade Temperaturmessung A	21
3.2. Ausgleichsgerade Temperaturmessung B	22
3.3. Ein Photo der Messzelle	23
3.4. Abhangigkeit des Kurzschlussstromes von der Temperatur	23
3.5. Lineraer Zusammenhang zwischen Strom und dem ADC-Wert	25
4.1. Messbahn auf den Platten	28
4.2. Kurzschlussstromes in Ampere	30
4.3. Relative Verteilung der Einstrahlung	30
4.4. Vergleich zweier Messungen	31
4.5. Temperatur der Zelle	31
4.6. konstante Temperatur der Zelle	32
4.7. Lampe E1	32
4.8. Lampe E2	32
4.9. Lampe E3	33

4.10. Lampe E4	33
4.11. Lampe E5	33
4.12. Lampe E6	34
4.13. Lampe E7	34
4.14. Lampe E8	34
4.15. Lampe E9	35
4.16. Lampe E10	35

Tabellenverzeichnis

1.1. Spektrale Strahlungsverteilung nach IEC 60904-9	2
1.2. Anforderungen an die 3 verschiedenen Simulatorklassen	3
3.1. Die Messwerte der Kalibrierung Messschaltung A	21
3.2. Die Messwerte der Kalibrierung Messschaltung B	22
3.3. Kalibrierung Strommessung	24
3.4. Temperaturabhängigkeit der Temperaturmessung	25
3.5. Temperaturabhängigkeit der Temperaturmessung	25
4.1. Ein Ausschnitt der Messwertedatei	29

Abkürzungsverzeichnis

www World Wide Web
URL Uniform Resource Locator

A. Sourcecode Arduino

```
/*
Programm zum Steuern des Sonnensimulatormessroboters
Version 1.0
*/

#include <SD.h>

void vor(int sped);
void zuruck(int sped);
void left(int sped);
void right(int sped);
void tl(int sped);
void tr(int sped);
void ztl();
void ztr();
void vtl();
void vtr();
void halt();

// ADCs zum Auslesen der 3x3 Matrix
const int analogInPin0 = A9;
const int analogInPin1 = A2;
const int analogInPin2 = A5;
const int analogInPin3 = A8;
const int analogInPin4 = A1;
const int analogInPin5 = A4;
const int analogInPin6 = A7;
const int analogInPin7 = A0;
const int analogInPin8 = A3;
const int analogInPin9 = A6;

// Ausgang zum Schalten der Infrarot-Leds
const int analogInPin10 = A10;

// ADC für Messwerte
const int analogInPin12 = A12;
const int analogInPin13 = A13;
const int analogInPin14 = A14;
const int analogInPin15 = A15;
```

```
// Digitalausgänge zum Ansteuern der Motoren
int IN1M1 = 2;
int IN2M1 = 3;
int IN1M2 = 4;
int IN2M2 = 5;
int IN1M3 = 6;
int IN2M3 = 9;
int IN1M4 = 7;
int IN2M4 = 8;

// Sensorwerte der optischen Sensoren (beleuchtet)
int sensorValue0 = 0;           // value read from the pot
int sensorValue1 = 0;           // value read from the pot
int sensorValue2 = 0;           // value read from the pot
int sensorValue3 = 0;           // value read from the pot
int sensorValue4 = 0;           // value read from the pot
int sensorValue5 = 0;           // value read from the pot
int sensorValue6 = 0;           // value read from the pot
int sensorValue7 = 0;           // value read from the pot
int sensorValue8 = 0;           // value read from the pot
int sensorValue9 = 0;           // value read from the pot

// Sensorwerte der optischen Sensoren (unbeleuchtet)
int sensorValue0d = 0;          // value read from the pot
int sensorValue1d = 0;          // value read from the pot
int sensorValue2d = 0;          // value read from the pot
int sensorValue3d = 0;          // value read from the pot
int sensorValue4d = 0;          // value read from the pot
int sensorValue5d = 0;          // value read from the pot
int sensorValue6d = 0;          // value read from the pot
int sensorValue7d = 0;          // value read from the pot
int sensorValue8d = 0;          // value read from the pot
int sensorValue9d = 0;          // value read from the pot

// für Auswertung der opischen Sensoren
int A = 0;
int B = 0;
int C = 0;
int D = 0;
int E = 0;
int F = 0;
int G = 0;
int H = 0;
int I = 0;
int S = 0;

// Grenzwerte für hell (=white) und dunkel (=black)
```

```
int white = 400;
int white2 = 250;
int bleak = 100;
int bleak2 = 150;

// zur Berechnung der Lage der LInie
float x1,x2,x3,d,k;
float y1,y2,y3,d2,k2;

// einige Hilfsvariablen
char mode='s';
char richtung='v';
int _stop=0;
int vor_ein=0;
int zuruck_ein=0;

// für die Auswertung der Messwerte
long int temp_modul = 0;
long int temp_i=0;
long int _Isc = 0;
long int time = 0;

// zur Schreiben auf die SD Karte
const int chipSelect = 53;

// Zahlvariable für Messpunkt, Spalte und Reihe
int count = 0;
int count2 = 0;
int row = 1;

// zur Schreiben auf die SD Karte
String dataString = "";

// notwendig für die Kommunikation mit SD Karte
void setup() {
    pinMode(A10, OUTPUT);
    pinMode(53, OUTPUT);
    SD.begin(chipSelect);
}

// Definiert alle möglichen Bewegungen
void zuruck(int sped)
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,sped);
    analogWrite(IN1M2,0);
    analogWrite(IN2M2,sped);
```

```
analogWrite(IN1M3,0);
analogWrite(IN2M3,sped);
analogWrite(IN1M4,0);
analogWrite(IN2M4,sped);
}

void vor(int sped)
{
    analogWrite(IN1M1,sped);
    analogWrite(IN2M1,0);
    analogWrite(IN1M2,sped);
    analogWrite(IN2M2,0);
    analogWrite(IN1M3,sped);
    analogWrite(IN2M3,0);
    analogWrite(IN1M4,sped);
    analogWrite(IN2M4,0);
}

void left(int sped)
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,sped);
    analogWrite(IN1M2,sped);
    analogWrite(IN2M2,0);
    analogWrite(IN1M3,sped);
    analogWrite(IN2M3,0);
    analogWrite(IN1M4,0);
    analogWrite(IN2M4,sped);
}

void right(int sped)
{
    analogWrite(IN1M1,sped);
    analogWrite(IN2M1,0);
    analogWrite(IN1M2,0);
    analogWrite(IN2M2,sped);
    analogWrite(IN1M3,0);
    analogWrite(IN2M3,sped);
    analogWrite(IN1M4,sped);
    analogWrite(IN2M4,0);
}

void tr(int sped)
{
    analogWrite(IN1M1,sped);
    analogWrite(IN2M1,0);
    analogWrite(IN1M2,0);
```

```
analogWrite(IN2M2,sped);
analogWrite(IN1M3,sped);
analogWrite(IN2M3,0);
analogWrite(IN1M4,0);
analogWrite(IN2M4,sped);
}

void tl(int sped)
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,sped);
    analogWrite(IN1M2,sped);
    analogWrite(IN2M2,0);
    analogWrite(IN1M3,0);
    analogWrite(IN2M3,sped);
    analogWrite(IN1M4,sped);
    analogWrite(IN2M4,0);
}

void vtr()
{
    analogWrite(IN1M1,64);
    analogWrite(IN2M1,0);
    analogWrite(IN1M2,0);
    analogWrite(IN2M2,64);
    analogWrite(IN1M3,128);
    analogWrite(IN2M3,0);
    analogWrite(IN1M4,128);
    analogWrite(IN2M4,0);
}

void vtl()
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,64);
    analogWrite(IN1M2,64);
    analogWrite(IN2M2,0);
    analogWrite(IN1M3,128);
    analogWrite(IN2M3,0);
    analogWrite(IN1M4,128);
    analogWrite(IN2M4,0);
}

void ztr()
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,64);
```

```
analogWrite(IN1M2,0);
analogWrite(IN2M2,64);
analogWrite(IN1M3,0);
analogWrite(IN2M3,64);
analogWrite(IN1M4,64);
analogWrite(IN2M4,0);
}

void ztl()
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,64);
    analogWrite(IN1M2,0);
    analogWrite(IN2M2,64);
    analogWrite(IN1M3,64);
    analogWrite(IN2M3,0);
    analogWrite(IN1M4,0);
    analogWrite(IN2M4,64);
}

void halt()
{
    analogWrite(IN1M1,0);
    analogWrite(IN2M1,0);
    analogWrite(IN1M2,0);
    analogWrite(IN2M2,0);
    analogWrite(IN1M3,0);
    analogWrite(IN2M3,0);
    analogWrite(IN1M4,0);
    analogWrite(IN2M4,0);
}

// Hauptschleife
void loop()
{
    // Werte ohne Beleuchtung:
    sensorValue0d = analogRead(analogInPin0);
    sensorValue1d = analogRead(analogInPin1);
    sensorValue2d = analogRead(analogInPin2);
    sensorValue3d = analogRead(analogInPin3);
    sensorValue4d = analogRead(analogInPin4);
    sensorValue5d = analogRead(analogInPin5);
    sensorValue6d = analogRead(analogInPin6);
    sensorValue7d = analogRead(analogInPin7);
    sensorValue8d = analogRead(analogInPin8);
```

```
sensorValue9d = analogRead(analogInPin9);
digitalWrite(A10, HIGH); // LED ein
delay(2);
// Werte mit Beleuchtung:
sensorValue0 = analogRead(analogInPin0);
sensorValue1 = analogRead(analogInPin1);
sensorValue2 = analogRead(analogInPin2);
sensorValue3 = analogRead(analogInPin3);
sensorValue4 = analogRead(analogInPin4);
sensorValue5 = analogRead(analogInPin5);
sensorValue6 = analogRead(analogInPin6);
sensorValue7 = analogRead(analogInPin7);
sensorValue8 = analogRead(analogInPin8);
sensorValue9 = analogRead(analogInPin9);
digitalWrite(A10, LOW);

A = sensorValue1 - sensorValue1d;
B = sensorValue2 - sensorValue2d;
C = sensorValue3 - sensorValue3d;
D = sensorValue4 - sensorValue4d;
E = sensorValue5 - sensorValue5d;
F = sensorValue6 - sensorValue6d;
G = sensorValue7 - sensorValue7d;
H = sensorValue8 - sensorValue8d;
I = sensorValue9 - sensorValue9d;

S = sensorValue0 - sensorValue0d;

// zum Erkennen der Linie bei vor- oder zurückfahren
x1 = ((C-A)/float(2*A-4*B+2*C));
x2 = ((F-D)/float(2*D-4*E+2*F));
x3 = ((I-G)/float(2*G-4*H+2*I));

d = (x1 + x2 + x3)/3.; // ~ Abstand von der Ideallinie
k = (x1-x3)/2.; // Steigung = Verdrehung

// zum Erkennen der Linie bei links- oder rechtsfahren
y1 = ((G-A)/float(2*A-4*D+2*G));
y2 = ((H-B)/float(2*B-4*E+2*H));
y3 = ((I-C)/float(2*C-4*F+2*I));

d2 = (y1+y2+y3)/3.;
k2 = (y3-y1)/2.;

if( S < bleak2) _stop=0;

halt();
```

```
if(E>bleak2)
{

    switch(mode)
    {
        case 's': // Start
        {
            halt();
            delay(60000); // 1 Minuten warten
            mode='v';
        }
        break;

        // Messmode
        case 'e': // Ende
        {
            halt();
        }
        break;

        case 'm': // Messen
        {
            count = count + 1; // Anzahl der Haltepunkte
            count2 = count2 + 1;
            dataString = "";
            _stop=1;
            halt();

            delay(250); // damit die Strome der Fahrmotoren keinen Einfluss auf die AD-Wand

            _Isc = 0;
            temp_modul = 0;
            temp_i = 0;

            // Mittelung über jeweils 500 Messwerte
            for(int i = 0; i < 500 ; i++)
            {
                _Isc = _Isc + analogRead(analogInPin13);
                temp_modul = temp_modul + analogRead(analogInPin12);
                temp_i = temp_i + analogRead(analogInPin15);
            }
            _Isc = int(_Isc/500.0);
            temp_modul = int(temp_modul/500.0);
            temp_i = int(temp_i/500.0);
            time = millis();
        }
    }
}
```

```
// Schreiben auf SD-Karte
dataString += String(time);
dataString += "\t";
dataString += String(_Isc);
dataString += "\t";
dataString += String(temp_modul);
dataString += "\t";
dataString += String(temp_i);
dataString += "\t";
dataString += String(count);
dataString += "\t";
dataString += String(count2);
dataString += "\t";
dataString += String(row);

// Schreibe auf SD Card
File dataFile = SD.open("datalog.txt", FILE_WRITE);
if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
}

// zurück in den Bewegungsmodus
if(richtung=='v') mode='v';
if(richtung=='z') mode='z';

}
break;

case 'v': // Vorwärtsfahren
{
    richtung='v';

    if(d>0.02) vtr();
    else
    {
        if(d>-0.02) vor(128);
        else
            vtl();
    }

//if((C<bleak)&&(A<bleak))
if(A<bleak2)
{
    if(k>0.01) tr(64);
    else
```

```
{  
    if(k<-0.01) tl(64);  
}  
  
if(d>0.025) right(64);  
else  
{  
    if(d<-0.025) left(64);  
}  
}  
if((A>white)&&(B>white)) mode='a';  
  
if(_stop==0)&&(S>white2)) mode='m';  
  
}  
break;  
  
case 'z': // Rückwärtsfahren  
{  
  
richtung='z';  
  
if(d>0.02) ztr();  
else  
{  
    if(d>-0.02) zuruck(128);  
    else ztl();  
}  
  
//if((G<bleak)&&(I<bleak))  
if(G<bleak2)  
{  
    if(k>0.01) tr(64);  
    else  
{  
        if(k<-0.01) tl(64);  
    }  
}  
if(d>0.025) right(64);  
else  
{  
    if(d<-0.025) left(64);  
}  
}  
if((G>white)&&(H>white)) mode='c';
```

```
if(( _stop==0)&&(S>white2)) mode='m';

if((I<bleak)&&(H<bleak)&&(G<bleak)) mode='e';

}

break;

case 'r':      //nach rechts
{
    richtung='r';
    count2 = 0;

    right(128);
    if(D>white2)
    {
        if(d2>0.04) vor_ein=1;
        if(d2<0.02) vor_ein=0;
        if(d2<-0.04) zuruck_ein=1;
        if(d2>-0.02) zuruck_ein=0;

        if(vor_ein==1) vor(64);
        if(zuruck_ein==1) zuruck(64);

        if((vor_ein==0)&&(zuruck_ein==0))
        {
            if(k2>0.02) tr(64);
            else
            {
                if(k2<-0.02) tl(64);
            }
        }
    }

    if((D<bleak)&&(G<bleak)&&(A<bleak))
    {
        mode='b';
    }

}
break;

case 'l':      // auch nach rechts
{
    richtung=='l';
    count2 = 0;

    right(128);
```

```
//if((C<bleak)&&(I<bleak))
if(D>white2)
{
    if(d2>0.04) vor_ein=1;
    if(d2<0.02) vor_ein=0;
    if(d2<-0.04) zuruck_ein=1;
    if(d2>-0.02) zuruck_ein=0;

    if(vor_ein==1) vor(64);
    if(zuruck_ein==1) zuruck(64);

    if((vor_ein==0)&&(zuruck_ein==0))
    {
        if(k2>0.02) tr(64);
        else
        {
            if(k2<-0.02) tl(64);
        }
    }
}

if((D<bleak)&&(G<bleak)&&(A<bleak))
{
    mode='d';
}

}
break;

case 'a': // Um die Ecke fahren
{
    if((A>bleak2)&&(B>bleak2)) vor(64);
    else
    {
        right(64);
        delay(700);
        halt();
        row = row +1 ;

        mode='r';
    }
}
break;

case 'b': // Um die Ecke fahren
{
```

```
zuruck(64);
delay(850);
halt();
delay(500);

mode='x';

}

break;

case 'x':
{
    if(d>0.025) right(64);

    if(d<-0.025) left(64);

    if((d<0.1)&&(d>-0.1)) mode='y';
}
break;

case 'y':
{
    if(k>0.01) tr(32);

    if(k<-0.01) tl(32);

    if((k<0.04)&&(k>-0.04)) mode='z';
}
break;

case 'c': // Um die Ecke fahren
{
    if((H>bleak2)&&(G>bleak2 )) zuruck(64);
    else
    {
        right(64);
        delay(700);
        halt();
        row = row +1 ;

        mode='l';
    }
}
break;

case 'd': // Um die Ecke fahren
```

```
{  
  
    vor(64);  
    delay(850);  
    halt();  
    delay(500);  
    mode='q';  
  
}  
break;  
  
case 'q':  
{  
    if(d>0.025) right(64);  
  
    if(d<-0.025) left(64);  
  
    if((d<0.1)&&(d>-0.1)) mode='p';  
}  
break;  
  
case 'p':  
{  
    if(k>0.01) tr(32);  
  
    if(k<-0.01) tl(32);  
  
    if((k<0.04)&&(k>-0.04)) mode='v';  
}  
break;  
}  
}  
  
else  
{  
    if((richtung=='v')||(richtung=='z'))  
    {  
        if((A>bleak2)|| (D>bleak2)|| (G>bleak2)|| (C>bleak2)|| (F>bleak2)|| (I>bleak2))  
        {  
            if((A+D+G)>(C+F+I)) right(128);  
            else left(128);  
        }  
        else halt();  
    }  
}
```

```
if((richtung=='r')||(richtung=='l'))
{
    if((A>bleak2)|| (B>bleak2)|| (C>bleak2)|| (G>bleak2)|| (H>bleak2)|| (I>bleak2))
    {
        if((A+B+C)>(G+H+I)) vor(64);
        else zuruck(64);
    }
    else halt();
}

delay(5);

}
```

B. Sourcecode Auswertung

```
% Graphische Auswertung der Robotermesswerte

close all;
clc;
clear all;

% "Offnen der Datei, welche eine Messfahrt mit genau 308 Messwerten
% enthalten muss
fid = fopen('data1.txt', 'r');
a = fscanf(fid, '%g %g', [7 308])
a = a';
fclose(fid)

j=1; k=1; l=1;

% Umwandeln der 1 dimensionalen Modultemperatur ADC-Werte in eine Matrix:
for(i=0:307)
    if(mod(i,28)>13) k = 28- mod(i,28);
    else k = mod(i,14)+1;
    end
    Tm(k,l)=a(i+1,3);
    if(mod(i+1,14)==0) l=l+1;
    end
end

% Umrechnung der ADC-Werte in Temperatur:
Rm = (Tm +4038.9)/40.107; % ADC --> Widerstand
TTm = (Rm-100.03)/0.3879; % Widerstand --> Temperatur

j=1; k=1; l=1;

% Umwandeln der 1 dimensionalen Innentemperatur ADC-Werte in eine Matrix:
for(i=0:307)
    if(mod(i,28)>13) k = 28- mod(i,28);
    else k = mod(i,14)+1;
    end
    Ti(k,l)=a(i+1,4);
    if(mod(i+1,14)==0) l=l+1;
    end
```

```

end

% Umrechnung der ADC-Werte in Temperatur:
Ri = (Ti +4023.6)/40.027; % ADC --> Widerstand
TTi = (Ri-100.03)/0.3879; % Widerstand --> Temperatur

j=1; k=1; l=1;

% Umwandeln der 1 dimensionalen Kurzschlusstrom ADC-Werte in eine Matrix:
for(i=0:307)
    if(mod(i,28)>13) k = 28- mod(i,28);
    else k = mod(i,14)+1;
    end
    I(k,l)=a(i+1,2);
    if(mod(i+1,14)==0) l=l+1;
    end
end

% Umrechnung der ADC-Werte in Ampere:

A = (I +4.5766)/119.2; % ADC --> Ampere
A = A - (TTm-25)*0.004; % Temperaturkorrektur

[XI,YI] = meshgrid(1:.125:22, 1:.125:14);

Ai = interp2(A,XI,YI,'cubic'); % Interpolation

A_max = max(max(Ai));
A_min = min(min(Ai));

Normiert = Ai / A_max * 1.1;

w = (A_max - A_min)/(A_max + A_min) * 100 % maximale Abweichung in Prozent

% Graphische Darstellung des Stromes
figure;
surf(XI,YI,Ai); % in Ampere

% Graphische Darstellung des normierten Stromes
zlevs2 = 0.9:0.01:1.1;
figure;
[C,h] = contourf(XI,YI, Normiert, zlevs2);
set(h, 'ShowText', 'on', 'TextStep', get(h, 'LevelStep')*2)
colorbar;

```

```
% Graphische Darstellung der Modultemperatur
figure;
surf(TTm);

% Graphische Darstellung der Umgebungstemperatur
figure;
surf(TTi);
```