Hochschule für angewandte Wissenschaften Hamburg

Fakultät Life Sciences

Department Umwelttechnik

Entwicklung eines mobilen Mess- und Datenerfassungssystems zur Untersuchung des Einflusses fluktuierender Wolkenschatten auf den Ertrag der Photovoltaikanlagen

Bachelorarbeit

im Studiengang Umwelttechnik

Vorgelegt von:

Konstantin Marar

[redacted]

Hamburg

Abgabedatum: 10.08.2022

Erstgutachter:	Prof. Dr. Timon Kampschulte	(HAW Hamburg)
Zweitgutachter:	Dipl.-Ing. (FH) André Schumann	(SolPEG GmbH)

Die Abschlussarbeit wurde betreut und erstellt in Zusammenarbeit mit der Firma SolPEG GmbH

Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle aus fremden Quellen wörtlich oder sinngemäß entnommenen Gedanken habe ich als solche kenntlich gemacht.

Hamburg, den 10.08.2022

_____

Konstantin Marar

Danksagung

Hiermit möchte ich mich bei all denjenigen herzlich bedanken, die mich während der Anfertigung meiner Bachelorarbeit unterstützt und motiviert haben.

Der Dank geht an die Firma SolPEG GmbH, die mich herzlich aufgenommen und mir die Chance gegeben hat, eine Bachelorarbeit mit dem besagten Thema zu schreiben.

Diesbezüglich möchte ich meinen Betreuern Prof. Dr. Timon Kampschulte und Dipl.-Ing. (FH) André Schumann herzlich danken, die mir bei wichtigen Denkansätzen geholfen haben, sowie Dipl.-Ing. Jan-Claas Böhmke der weitgehend in allen elektronischen Fragen zur Hilfe bereitstand.

Ebenfalls möchte ich mich bei meinem Kollegen Martin bedanken, der mich bei meinem Testlauf des Messgeräts an einer PV-Anlage half.

Nicht zuletzt möchte ich mich bei meiner Freundin Olja bedanken, die mich immer wieder aufbaute, motivierte und mit großer Geduld meine Arbeit Korrektur gelesen hat.

Thema der Bachelorarbeit

Entwicklung eines mobilen Mess- und Datenerfassungssystems zur Untersuchung des Einflusses fluktuierender Wolkenschatten auf den Ertrag der Photovoltaikanlagen

Stichworte

Arduino, Arduino Due, Module, Sensoren, Elektronik, Programmieren, Signalverarbeitung

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Entwicklung eines Arduino basierenden mobilen Mess- und Datenerfassungssystems. Es werden die Module und Sensoren, die eine Rolle bei der Datenerfassung spielen, dargelegt und genauer erklärt. Zusätzlich werden die elektronischen Komponenten und die Programmierung aufgezeigt, die für die Signalverarbeitung von Bedeutung sind.

Title of the paper

Development of a mobile measurement and data acquisition system to investigate the influence of fluctuating cloud shadows on the yield of photovoltaic systems.

Keywords

Arduino, Arduino Due, modules, sensors, electronics, programming, signal processing

Abstract

This paper focuses on the development of a mobile measurement and data acquisition system based on Arduino. The modules and sensors that play a role in data acquisition are presented and explained in detail. In addition, the electronic components and programming that are important for signal processing are also shown.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ARM - *Advanced RISC Machines*

CAD - *computer aided design*

CAN - *Controller Area Network*

COM - *Serielle Schnittstelle*

CSV - Character-separated values

DAC - *Digital-Analog-Wandler*

DC - *Direct Current*

FAT - *File Allocation Table*

FS - *Full Scale*

GSM - *Global System for Mobile Communications*

I/O - *Input/Output*

I²C - *Inter-Integrated Circuit*

ICSP - *in-circuit serial programmer*

IDE - *Integrated Development Environment*

LCD - *Liquid Crystal Display*

OC - *Open-Collector*

PE - *protective earth (Schutzleiter)*

PR - *Performance Ratio*

PVC - *Polyvinylchlorid*

PWM - *pulse width modulation*

RTC - *real-time clock*

SD - *Secure Digital*

SDHC - *Secure Digital High Capacity*

SELV - *Safety Extra Low Voltage*

SolPEG - *Solar Power Expert Group*

SPI - *Serial Peripheral Interface, Serial Peripheral Interface*

SRAM - *Static random-access memory*

USB - *Universal Serial Bus*

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Bachelorarbeit entstand in Zusammenarbeit mit der SolPEG (Solar Power Expert Group) GmbH. Das Hauptgeschäftsfeld des Unternehmens ist die Erstellung von Ertragsgutachten für Baufirmen und Betreiber von Photovoltaik-Anlagen. Um die Prognosen der Ertragsgutachten noch genauer und zuverlässiger werden zu lassen, untersucht die SolPEG immer wieder verschiedene spezifische Verluste von PV-Anlagen. Darunter spielen die Ergebnisse aus den vorangegangenen Abschlussarbeiten über Verschattungsverluste, Temperaturverluste oder Leistungsbegrenzungsverluste bis heute eine wichtige Rolle in den von dem Unternehmen erstellten Ertragsgutachten.

Motivation dieser Arbeit ist, dass in Zeiten mit temporär, stark schwankenden Einstrahlungen durch einen Wolkenzug ein Absinken der Performance Ratio Werte in den Monitoringsystemen einiger PV-Anlagen beobachtet wurde. Es wird die These aufgestellt, dass der heterogene Wolkenzug, die zu Strängen in Reihe verschalteten Module partiell verschattet und diese somit eine punktuell gemessene, hohe Einstrahlung nicht umsetzen können. Demnach wird also ein spezieller, nicht durch Herstellungstoleranzen bedingter Mismatch-Verlust als ursächlich vermutet.

Um diese Vermutung zu untersuchen, bedarf es eines Messgeräts mit langer und hochaufgelöster Zeiterfassung, dass die Verteilung der Einstrahlung und die Leistung eines Stranges einer PV-Anlage misst und aufzeichnet. Das Gerät sollte zudem mobil und unabhängig von bestehenden Überwachungssystemen auf verschiedenen PV-Anlagen einsetzbar sein. Eine Marktrecherche führte zu keinen zufriedenstellenden Treffern bereits vorhandener Lösungen. Insbesondere der Wunsch, Einstrahlungsmessungen verschiedener Sensoren nur dann zeitlich hochaufgelöst zu erfassen, wenn es zu unterschiedlichen und schwankenden Einzelmessungen kommt und zusätzlich die gewünschte Flexibilität, Anpassungen an Hard- und Software vornehmen zu können. Im Rahmen dieser Bachelorarbeit hat die SolPEG veranlasst, eine eigene Lösung für den messtechnischen Nachweis der oben beschriebenen Hypothese zu entwickeln.

Ziel dieser Arbeit ist es, eine Datenerfassung auf der Arduino-Plattform zu entwickeln, mit der es möglich ist, die Signale von vier Einstrahlungssensoren, eines Spannungssensors und eines Stromsensors parallel aufzunehmen und nach Bedarf auf eine Speicherkarte zeitlich hochauflösend zu speichern. Zudem soll der Anwender über ein Display die momentanen Messwerte sehen und das Messintervall ändern können. Abschließend soll das Gerät im praktischen Einsatz erprobt werden, mit dem Ziel, die volle Einsatzfähigkeit zu erreichen. Die Auswertung und Untersuchung der aufgezeichneten Daten einer längeren Messkampagne mit Hinblick auf die oben genannte Hypothese sind hingegen nicht Gegenstand dieser Arbeit.

Hierfür wird in Kapitel 2 die genaue Problematik erläutert und erste Ansätze aufgezeigt. In Kapitel 3 wird die Plattform Arduino vorgestellt und die verwendete Hardware, sowie Module und Sensoren beschrieben. Das Unterkapitel 3.3 erläutert die Arduino Software und die verwendeten Bibliotheken. Zusätzlich wird die Funktionsweise eines RC-Tiefpassfilters erklärt, für die Reduzierung von Störsignalen. Kapitel 4 beschäftigt sich mit der Signalverarbeitung, der Elektronik, dem Zusammenbau und das Testen der einzelnen Sensoren. Im Unterkapitel 4.4 wird auf die selbstgeschriebenen Programmcodes eingegangen und als Abschluss wird ein Feldversuch mit dem fertigen Messgerät unternommen.

2 Problemstellung

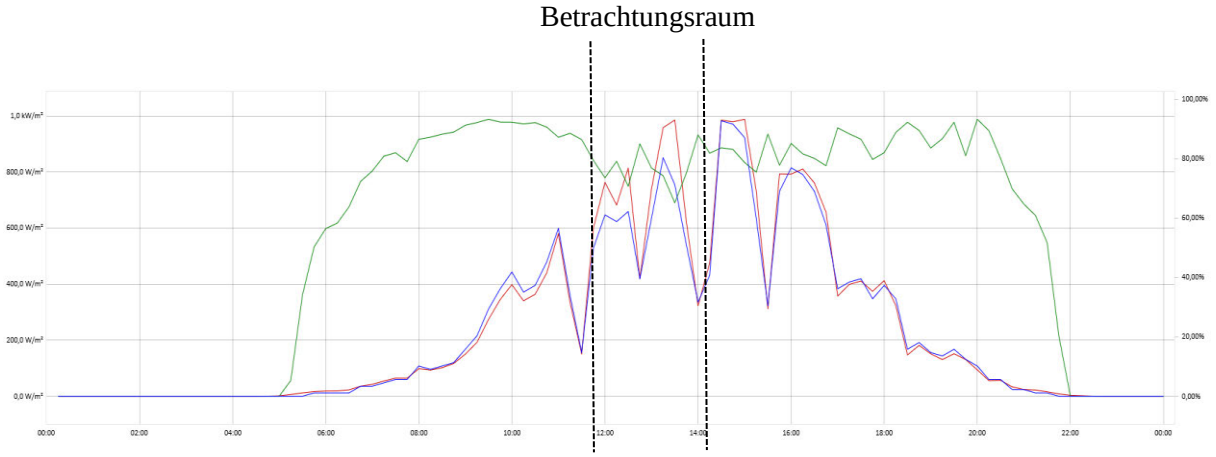


Abbildung 1: Monitoring Daten, Radiation-, PR- und Powerkurve eine PV-Anlage (Liniendiagramm)

Betrachtet man einen Ausschnitt der Monitoring-Daten aus Juni 2021 einer fest installierten Photovoltaik-Anlage in der Abbildung 1, ist die Leistung in kW (blau), die Einstrahlung in W/m² (rot) und die Performance Ratio kurz PR in % (grün) dargestellt. Zur besseren Veranschaulichung der Einstrahlungskurve und Leistungskurve mit unterschiedlichen Einheiten wurde die Skalierung so gewählt, dass Kurven übereinander passen und eine Veränderung der Leistungskurve sofort erkennbar ist. Beobachtet man den Zeitraum von 6 bis 12 Uhr und 14 bis 22 Uhr, sieht man den typischen Verlauf der drei Kurven an einem bewölkten Tag. Dabei muss man erwähnen, dass eine Anlage immer deutlich weniger W pro W/m^2 Einstrahlung produziert. Dies liegt einmal an dem Wirkungsgrad und der Temperatur der PV-Module. Steigt die Einstrahlung der Sonne, steigt somit die Temperatur der Solarzelle und damit sinkt die Spannung und die resultierende Leistung einer Zelle. Man spricht von 0,4 bis 0,5% Leistungsreduzierung pro Kelvin *(siehe S.104, Mertens, 2020)*. Betracht man nun den Zeitraum von 12 bis 14 Uhr, kann man deutlich ein höheres Absinken der Leistungskurve und die dazu abhängige PR-Kurve entdecken. Es wird die Vermutung aufgestellt, dass der heterogene Wolkenzug, die zu Strängen, in Reihe verschalteten Module nur partiell verschattet und es so zu dem Leistungseinbruch führt. Das starke Absinken der Einstrahlungskennlinie ist hier aber nicht zu sehen. Das resultiert daraus, dass die betrachtete PV-Anlage nur zwei Einstrahlungssensoren, ein Pyranometer und eine Referenzelle auf der ganzen Anlage verbaut hat. Vermutlich werden diese zwei Einstrahlungssensoren nicht durch den inhomogenen Wolkenzug partiell verschattet. Auch bei einzelner Verschattung der einzelnen Sensoren werden die Einzelmessungen gemittelt und in den Monitoring Prozess eingespeist. Das führt wiederum dazu, dass eine geringere Absenkung der Einstrahlungskurve im Monitoringsystem zu sehen ist.

Um diese Annahme genauer zu untersuchen für die spätere Analyse der Daten, bedarf es mehrerer Einstrahlungssensoren entlang eines PV-Stranges, die den Wolkenzug mitverfolgen und aufzeichnen können. Dadurch, dass dieses Thema aktuell kaum erforscht ist und dazu keine geeigneten Messgeräte auf dem Markt zu Verfügung stehen, wird in dieser Abschlussarbeit auf die Entwicklung eines speziellen Messsystems eingegangen.

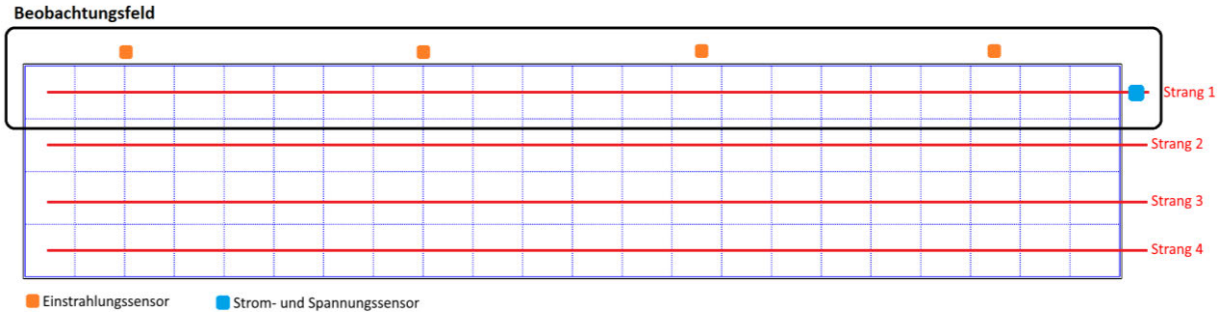


Abbildung 2: Platzierung der Sensoren an einem Strang

Abbildung 2 zeigt die Lage der Sensoren für die spätere Datensammlung. Die platzierten Einstrahlungssensoren sollen bündig mit den PV-Modulen am oberen Strang installiert werden, um denselben Einstrahlungswinkel der Sonneneinstrahlung zu erhalten. Die Platzierung der Sensoren am oberen Strang hat den Vorteil, dass es zu keiner Reihenverschattung durch zu nah stehende Modultische kommen kann. So soll eine detaillierte Verfolgung der Wolken an einem Strang möglich sein. Die Strom- und Spannungsmessung wird in den PV-Strang eingebunden, um die Leistung des Stranges ermitteln zu können.

3 Grundlagen

In diesem Kapitel wird auf die Einzelheiten der Hardware, die für dieses Projekt verwendet wurde, eingegangen. Anschließend wird die Verwendung der Software, deren wichtigen Elemente und deren Verknüpfung mit der Hardware näher erläutert.

3.1 Arduino

Die Open Source Plattform Arduino besteht aus der einfachen Nutzung und der Zugänglichkeit von Hardware und Software. Arduino wurde am Ivrea Interaction Design Institute als einfaches Werkzeug für die schnelle Erstellung von Prototypen entwickelt und richtete sich an Studenten, die keine Erfahrung mit Elektronik und Programmierung haben. Mittlerweile entwickelt, produziert und unterstützt Arduino die Software und elektronische Geräte für eine einfache und kostengünstige Nutzung von Mikrokontrollern. Die Produkte von Arduino unterstützen einfache Alltagsprojekte bis hin zu komplexen wissenschaftlichen Instrumenten. Um diese Open-Source-Plattform hat sich eine weltweite Gemeinschaft von Studenten, Bastlern, Künstlern, Programmierern und Fachleuten gebildet, deren Beiträge eine unglaubliche Menge an zugänglichem Wissen ergeben (*Arduino, 2022*).

Die Hardware besteht aus einem Mikrokontroller, der auf einer Platine sitzt. Dieser wird als Board bezeichnet. Dabei gibt es vom Hersteller verschiedene Boards für verschiedene Aufgabenbereiche. Auf jedem Board sind die notwendigen elektronischen Komponenten wie Widerstände, Kondensatoren, Dioden und Schaltkreise vorhanden. Zudem besitzen die Boards unterschiedliche Schnittstellen und Stecker für die weiteren Verbindungen mit dem Computer und anderen Arduino-Modulen und Sensoren. Der Arduino Uno ist der bekannteste aus der Arduino Familie. Dieser wird besonders in Schulen eingesetzt, um den Kindern das Programmieren näherzubringen. Die Nano- und Minireihe sind mit ihrer kleinen Größe von 4,4 cm und 3,3 cm für Projekte auf kleinem Raum gedacht, wie z.B. im Modellbau. Die Mega- und Due-Reihe ist besonders für große und komplexe Projekt entwickelt worden.

Diese verfügen über bis zu 54 Ein- und Ausgangspins mit einem leistungsstärkeren Mikrokontroller und einem größeren Speicher. Durch diese Vielfalt wird es einem ermöglicht, die benötigten Komponenten flexibel zu wählen und zu erweitern, um die gewünschten Anforderungen zu erreichen.

Darüber hinaus stehen auf der Arduino-Webseite zahlreiche CAD (computer aided design) -Daten und Leiterplattendaten für das kostenlose CAD-Programm Eagle zur Verfügung. Dies ermöglicht, eigene Boards nach Bedarf zu entwickeln.

Um den Mikrokontroller auf bestimmte Sensoren oder Aufgaben reagieren zu lassen, wird das Arduino-IDE (Integrated Development Environment) zu Deutsch Entwicklungsumgebung benötigt, das Lokal auf einem Computer installiert werden muss. Aktuell steht die Version 1.8.19 zur Verfügung. Die Software ist für die gängigen Betriebssysteme wie Windows, Linux und Mac OS verfügbar und wird laufend aktualisiert und erweitert. Ergänzend dazu werden beim Installieren der Software die notwendigen Treiber, die für die serielle Kommunikation des USB-Ports mit dem Mikrokontroller auf dem Arduino Board benötigt werden, mit installiert und konfiguriert. Im Grunde besteht die Arduino-IDE aus einem Java-Programm und ist wie ein Texteditor zum Schreiben von Programmcodes, die auch Sketches genannt werden. Die Arduino Programmiersprache ist angelehnt an die Programmiersprache C bzw. C++. Zusätzlich beinhaltet die Arduino Entwicklungsumgebung sogenannte Bibliotheken. Diese sind fertige Codes, die integriert werden können, um bestimmte Funktionen in den eigenen Sketch zu implementieren. Dadurch wird die eigene Programmierung wesentlich erleichtert.

3.2 Hardware

In den folgenden Abschnitten wird die verwendete Hardware des Projektes beschrieben. Dazu zählen das Arduino Due Board, die Module und die wichtigen Sensoren. Im Rahmen dieser Arbeit mussten keine weiteren Veränderungen an der Hardware vorgenommen werden.

3.2.1 Arduino Due

Für die Entwicklung eines mobilen Mess- und Datenerfassungssystems wurde ein Arduino Due gewählt. Dieser verfügt über 12 analoge Eingangs Pins mit einer Auflösung von 12 Bits. Unkonfiguriert löst dieser mit 10 Bit auf. Mit dem Befehl *„analogReadResolution“* lässt sich die Auflösung ändern (*Arduino-Referenz, 2022*). Die Bits werden beim Arduino über die Analogeingänge in Teilen erfasst. Bei 10 Bits sind es 1024 Teile und bei 12 Bits 4096 Teile. Dabei entspricht bei einer maximalen Spannung von 3,3 V und 10 Bit, eine messbare Spannung von 3,2 mV pro Teil und bei 12 Bit eine Spannung von 0,81 mV pro Teil (*arduino.cc, 2022*).

Der Due hat außerdem eine höhere Taktfrequenz von 84 MHz als der weitverbreitete Arduino Uno mit 16 MHz. Dies führt zu einer schnelleren Aufnahme, Bearbeitung und Ausgabe von Daten. Wichtig ist, zu beachten, dass der Arduino Due im Gegensatz zu den anderen aus seiner Familie mit einer Spannung von 3,3 V angesteuert wird und nicht wie üblich mit 5 V. Eine Spannung an den Analog/ Digital Pins, die höher als 3,3 V ist, kann das Board dauerhaft beschädigen. Neben dem Micro USB-Anschluss, der benötigt wird, um den Mikroprozessor zu programmieren, verfügt das Board zusätzlich über SPI (Serial Peripheral Interface) und I²C (Inter-Integrated Circuit) Anschlüsse. Diese Anschlüsse ermöglichen zahlreiche Sensoren und Displays, anschließen und steuern zu können.

Das Arduino Due kann mit und ohne Header bestellt werden. Die Header erleichtern das Experimentieren mit dem Arduino, wenn man nicht vorhat, das Board fest in ein Projekt zu verbauen.

Für die Programmierung des Boards wird die Arduino-Entwicklungsumgebung mit Version 1.5 oder höher benötigt.

In der Tabelle 1 sind die wichtigen technischen Daten des Arduino Due aufgelistet.

Tabelle 1: Technische Daten des Arduino Due, (Arduino Documentation, 2022)

Mikrokontroller	AT91SAM3X8E
SRAM	96 KB
Flash Memory	512 KB
Taktfrequenz	84 MHz
Digitale I/O Pins	54
PWM Digitale I/O Pins	12
Analog Pins	12
Betriebsspannung	3,3 V
Eingangsspannung (empfohlen)	7 – 12 V
Stromverbrauch	ca. 100 mA
DC Strom für 3,3 V Pin	Max. 800 mA
DC Strom für 5 V Pin	Max. 800 mA
Verbindungsschnittstelle	Mikro USB
Dimension/ Größe	101,52 x 53,3 mm

Nachfolgend werden die einzelnen Anschlussmöglichkeiten aus der Abbildung 1 beschrieben *(Arduino Input and Output, 2022*).

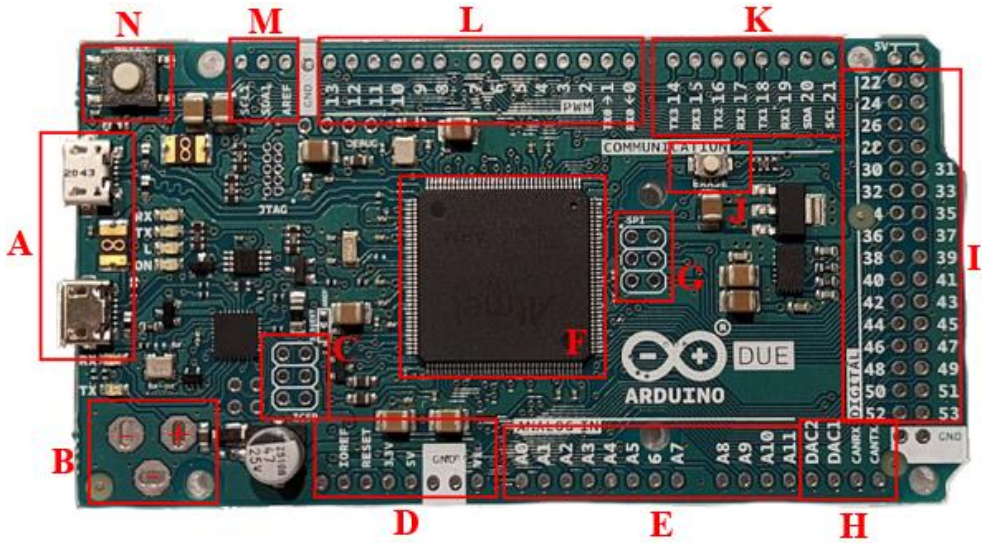


Abbildung 3: Arduino Due; Anschlussmöglichkeiten

A) Nativ port Serial USB (oben), Programming port Serial (unten): Das Arduino Due Board besitzt zwei USB-Schnittstellen, die das Board auch mit Strom versorgen können. Der Upload des Sketches ist auf beiden Ports möglich. Der „Nativ port“ ist direkt mit dem SAM3X Haupt-Mikrokontroller verbunden und der „Programming port“ führt erst einmal einen Umweg über einen zusätzlichen ATMEL 16U2 Mikrokontroller, bevor er auf den eigentlichen Haupt-Mikrokontroller trifft. In der Praxis wird der „Programming port“ für das Schreiben von Sketch benutzt, da dieser bei jedem Neuschreiben des Codes den Bootloader automatisch löscht und resettet. Beim „Nativ port“ muss das Löschen und Resetten des Bootloaders händisch über die Knöpfe **J** und **N** geschehen. Dieser USB-Port hat aber eine schnellere Zugriffszeit.

B) Externe Stromversorgung mit einer Spannung von 7 V bis 12 V: In der Regel ist ein Hohlstecker mit 5,5 mm Außen- und 2,1 mm Innendurchmesser verlötet für eine einfache Stromversorgung. In diesem Projekt wurde ein Board ohne Header verwendet.

C) ICSP (in-circuit serial programmer) Schnittstelle: Ermöglicht die Kommunikation mit einem weiteren Arduino. Zusätzlich hat man die Möglichkeit, über die Schnittstelle ein externes Programmiergerät anzuschließen, um z.B. den Bootloader zu programmieren oder zu entfernen.

D) Stromversorgung: 5 V und 3,3 V Ausgang für externe Stromversorgung von angeschlossenen Modulen an dem Arduino mit einem maximalen Strom von 800 mA. Über den *VIN* Pin (3,3 V) lässt sich der Arduino Due mit Strom versorgen. Der *IOREF* Pin liefert die Spannungsreferenz, mit dem der Mikrokontroller arbeitet.

E) Analogeingänge 12 Pins: Analoge Signaleingänge von 0 bis 3,3 V mit einer maximalen Auflösung bis 12 Bit.

F) Mikrokontroller Atmel SAM3X8E: Ein fest verlöteter 32 Bit ARM Mikrokontroller

G) SPI (Serial Peripheral Interface) Schnittstelle: Ist ein sogenanntes Zweidrahtbussystem, um mehrere elektronische Bausteine, die miteinander kommunizieren, über einer einzigen Schnittstelle zu betreiben. Über den *MISO* Pin werden Daten vom Slave zum Master gesendet und vom *MOSI* Pin die Daten vom Master zum Slave. Der Übertragungstakt wird vom Pin *SCK* geregelt und der *SS* Pin dient dazu, einen speziellen Slave über seine Adresse anzusprechen.

H) DAC (Digital-Analog-Wandler) und CAN (Controller Area Network) Anschlüsse: Mit dem zweikanaligen DAC kann der Arduino Due analog Signale erzeugen, z.B. für eine Audioausgabe von 16 Bit und 44,1 kHz. Die Spannungsausgabe liegt zwischen 0,55 V und 2,75 V. Der CAN-Bus ist ein Bus-System mit einer Datenübertragungsgeschwindigkeit von bis zu 1 Mbit/s, der den seriellen Datenaustausch zwischen Steuermodulen, z.B. in der Automatisierungstechnik ermöglicht.

I) 32 Digitale Pins (ohne Zusatzfunktion): Eine Digitale Ein- und Ausgabe. Diese wird realisiert mit LOW 0 V und HIGH 3,3 V, mit einem maximalen Strom bis 15 mA. Die digitalen Pins des Arduino-Mikrokontrollers sind in der Standardeinstellung als Eingänge programmiert und verfügen über einen hohen Eingangswiderstand.

J) Erase Knopf: Löscht den Flashspeicher des Arduinos und damit auch den zuvor geladen Sketch. Kann zur Not ausgelötet werden, um ein versehentliches Löschen des Flashspeichers zu verhindern.

K) 10 Digitale Pins (mit Zusatzfunktion): Diese verhalten sich wie die Pins unter dem Punkt **I**. Die Pins 0, 1 und 14 bis 19 mit den Markierungen *RX, TX* können als USB-Seriell-Adapter fungieren. Die Pins 20 und 21 mit den Markierungen *SDA* und *SCL* gehören zu der I²C-Schnittstelle. Genau wie die SPI-Schnittstelle erlaubt das I²C eine Kommunikation zwischen mehreren Master Slave Modulen.

L) 12 Digitale PWN (pulse width modulation) Pins: Mit der Pulsweitenmodulation lassen sich die Spannungen von 0 V bis 3,3 V digital modulieren. Die Modulation findet in einem Frequenzbereich von 1000 Hz statt.

M) I²C und Referenzspannung: Die Pins *SDA1* und *SCL1* sind weitere Anschlussmöglichkeiten zu der I²C-Schnittstelle. Der Pin *AREF* kann mit einer externen Referenzspannung verändert werden. Dadurch ist es möglich, den Messbereich einzugrenzen, um dadurch z.B. eine höhere Genauigkeit bei der Spannungsmessung zu erhalten.

N) Reset Knopf: Ermöglicht den händischen neu starten des Arduino Boards.

3.2.2 Arduino-Module

In diesem Abschnitt werden die Module beschrieben, die wichtig für die Realisierung des Projekts sind. Die fertigen Module erleichtern das Arbeiten mit dem Arduino, da es die einzelnen Komponenten auf einer fertigen Platine zu kaufen gibt. Die Module können sich in Form und Farbe von der Leiterplatte von den unten beschriebenen Modulen unterscheiden. Es ist wichtig, auf die bereits verbauten Komponenten der Module zu achten und die zur Verfügung gestellten Datenblätter zu beachten, da es Unterschiede bei den Funktionen durch verschiedene Hersteller geben kann.

RTC DS3231

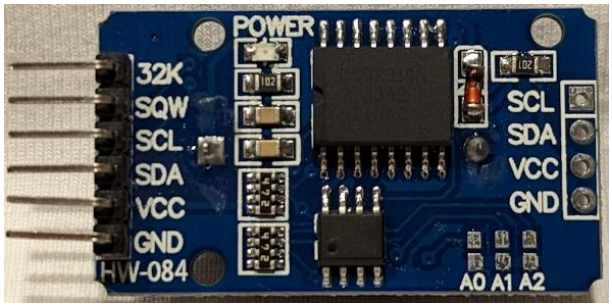


Abbildung 4: RTC DS3231

Die Echtzeituhr oder im englischen Real Time Clock kurz RTC wird benötigt, damit der Arduino weiß, wann die Daten aufgenommen wurden. Der DS3231 Chip ist im Vergleich zu seinem Vorgänger DS1307 wesentlich genauer und hat laut Datenblatt eine Abweichung von nur wenigen Sekunden nach einem Jahr Laufzeit. Zusätzlich werden auch die Schaltjahre bis zum Jahr 2100 berücksichtigt. Lediglich die Umstellung von Sommer und Winterzeit muss manuell angepasst werden. Das DS3231 Modul zählt Jahre, Monate, Wochentage, Tage, Stunden, Minuten und Sekunden.

Damit der RTC auch weiter die aktuelle Zeit richtig anzeigt, nachdem der Arduino vom Strom genommen wurde, besitzt die Platine eine CR2032 Knopfzelle in einem Batteriehalter auf der Rückseite des Moduls, um den Chip weiter mit Strom zu versorgen. Außerdem verfügt das Board über eine Batteriewiederaufladung, wenn der Chip extern mit einer Spannung von 5 V versorgt wird. Wichtig ist, dass viele Hersteller die günstige CR2032 und nicht die wiederaufladbare LIR2032 Knopfzelle beifügen. Die Aufladung der CR2032 kann im schlimmsten Fall einen Brand verursachen. Deshalb sollte man, wenn das Board mit der CR2032 betrieben werden soll, die Board-Spannung auf 3,3 V

runtersetzten oder den 200 Ω Widerstand wie in der Abbildung 5 mit einem roten Pfeil dargestellt entfernen. Das Board kommuniziert mit dem Arduino über die I²C-Schnittstelle. Die Adresse ist von Hersteller mit 0x57 standardmäßig eingestellt und lässt sich über das Kurzschließen der Jumper *A0, A1* und *A2* von 0x50 bis 0x56 einstellen.

Abbildung 5: 200 Ω Widerstand des RTC DS3231

In der Tabelle 2 werden die wichtigen technischen Daten des RTC DS3231 aufgelistet.

Tabelle 2: Technische Daten des RTC DS3231, (Maxim Integrated Products, Inc., 2015)

RTC-Chip	DS3231
Betriebsspannung	3,3 bis 5,5 V
Stromverbrauch	4 mA/ Batterie 0.07 mA
Batterie	CR2032 oder LIR2032
Verbindungsschnittstelle	I²C
Dimension/ Größe	38,5 x 21,8 mm

LCD HD44780

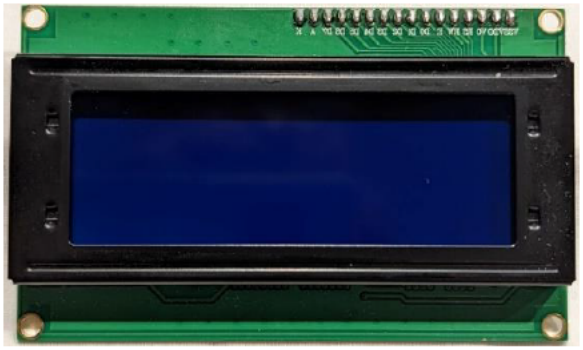


Abbildung 6: LCD HD44780

Das Display wird benötigt, um im laufenden Betrieb die Uhr auf Richtigkeit überprüfen zu können und die Ausgabe der Messwerte der Einstrahlungs-, Strom- und Spannungs-Sensoren darzustellen. Obendrein wird über das Display mit ausgegeben, ob die Logger-Aufzeichnung gestartet oder gestoppt

wurde. Das LCD unterstützt eine Darstellung von 20 Zeichen und 4 Zeilen mit integrierter Hintergrundbeleuchtung.

Über ein auf der Rückseite platzierten Potentiometer lässt sich der LCD-Kontrast einstellen. Des Weiteren kann über einen Jumper auf der Platine oder über die Sketch-Befehle „*lcd.backlight, lcd.nobacklight*“ die Hintergrundbeleuchtung ein- oder ausgeschaltet werden. Die Kommunikation mit dem Arduino erfolgt wie beim RTC DS3231 ebenfalls über eine I²C-Schnittstelle. Die Adresse ist vom Hersteller auf 0x27 standardmäßig eingestellt. Zudem lässt sich über das Kurzschließen der Jumper *A0, A1* und *A2* von 0x20 bis 0x26 einstellen.

In der Tabelle 3 werden die wichtigen technischen Daten des LCD HD44780 aufgelistet.

Tabelle 3: Technische Daten des LCD HD44780, (Hitachi, Ltd., 1998)

Auflösung	20 Zeichen x 4 Zeilen
Betriebsspannung	5 V
Stromverbrauch	4 mA / Hintergrundbeleuchtung 100 mA
Verbindungsschnittstelle	I²C
Dimension/ Größe	98 x 60 mm

Micro-SD-Modul TE417

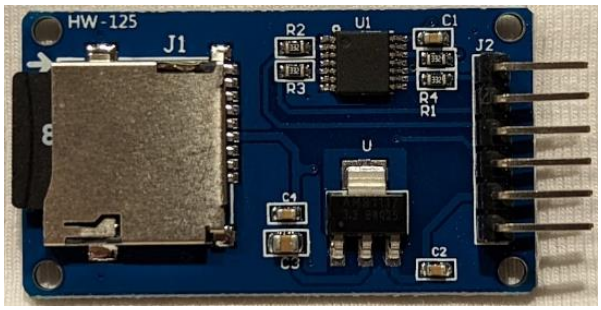


Abbildung 7: Micro-SD-Modul TE417

Das Micro-SD-Modul TE417 ist ein einfacher Adapter, um die Micro SD-Karte mit dem Arduino zu verbinden. Es ermöglicht die Messdaten, die der Arduino erfasst, auf eine Micro SD-Speicherkarte zu schreiben und zu lesen. Es werden Micro SD-Karten bis 2 GB und Micro SDHC-Karten bis 32 GB mit einer FAT16 und FAT32 Formatierung unterstützt. Das Modul ist mit einem Spannungswandler bestückt, der eine Eigenspannung von 4,5 V bis 5,5 V auf die Micro-SD Betriebsspannung von 3,3 V einstellt. Außerdem ist ein Micro-SD Kartenhalter integriert, um die Karte austauschen zu können. Die Kommunikation mit dem Arduino erfolgt über die SPI-Schnittstelle.

In der Tabelle 4 werden die wichtigen technischen Daten des Micro-SD-Modul TE417 aufgelistet.

Tabelle 4: Technische Daten des Micro-SD-Modul TE417, (Texas Instruments/Incorporated, 2022)

Betriebsspannung	4,5 bis 5,5 V
Stromverbrauch	1 bis 20 mA
Verbindungsschnittstelle	SPI
Dimension/ Größe	37 x 22 mm

3.2.3 Sensoren

Das Kapitel beschreibt die Sensoren und deren Funktion. Alle Sensoren besitzen eine Messumformer-Schaltung, die es erlaubt, ein definiertes Ausgangssignal zu erzeugen. Die indizierte Spannung wird verstärkt und in eine Ausgangsgröße von 4 bis 20 mA, 0 bis 5 V oder 2 bis 10 V umgeformt.

Einstrahlungssensor

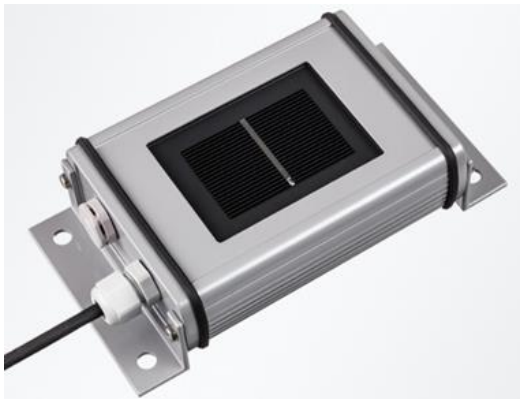


Abbildung 8: Si-420TC-T-K Silizium-Solarstrahlungssensor (Mencke & Tegtmeyer, 2022)

Für dieses Projekt werden vier Strahlungssensoren verwendet, auf Basis einer speziell verkapselten, kleinflächigen monokristallinen Silizium-Solarzelle mit einer Größe von 50 x 33 mm. Für die Einstrahlungsmessung wird die Spannung, die an einem niederohmigen Shunt-Widerstand, der an der Solarzelle kurzgeschlossen ist, gemessen. Dadurch, dass der Kurzschlussstrom einer Solarzelle proportional zu Bestrahlungsstärke ist, ist eine einfache Zuordnung möglich (*Wolff*, 2016). Die gemessene Spannung wird in ein Stromsignal umgewandelt. Die Genauigkeit solcher Referenzzellen liegt in einem Bereich ± 5 bis ± 10% (*siehe S. 273, Mertens, 2020*) und ist für das Vorhaben ausreichend, da das Hauptziel ist, Unterschiede zwischen den vier Sensoren zu erkennen und aufzuzeichnen.

Die Entscheidung zur Verwendung dieser Sensoren wurde aufgrund der aktuellen Verfügbarkeit sowie dem angemessenen Preis gefällt. Die Genauigkeit zu der tatsächlichen Einstrahlung spielt nur eine untergeordnete Rolle, denn für die Messung ist es wichtig, dass alle vier Sensoren eine hohe Genauigkeit untereinander aufweisen. Am einfachsten erzielt man dies durch die Verwendung von baugleichen Sensoren.

In der Tabelle 5 werden die wichtigen technischen Daten des Si-420TC-T-K Silizium-Solarstrahlungssensors aufgelistet.

Tabelle 5: Technische Daten des Si-420TC-T-K Silizium-Solarstrahlungssensor, (Ingenieurbüro Mencke & Tegtmeyer GmbH, 2012)

Linearität	±0,3% für 50 bis 1300 W/m²
Messbereich	0 bis 1200 W/m²
Betriebsspannung	12 bis 28 V
Stromverbrauch	10 bis 46 mA
Verbindungsschnittstelle	Analogausgang, 4 bis 20 mA
Dimension/ Größe	155 x 85 x 39 mm

Stromsensor

Abbildung 9: Stromsensor (CHENYANG TECHNOLOGIES, 2019)

Der verwendete Stromsensor funktioniert nach dem sogenannten Hall-Effekt. Hall-Sensoren basieren auf dem Hall-Effekt, der nach Edwin Herbert Hall, dem Entdecker des Effekts, benannt wurde. Der Effekt besagt, dass immer dann, wenn ein stromdurchflossener Leiter sich in einem Magnetfeld befindet, eine elektrische Spannung induziert wird. Diese Spannung fällt dabei senkrecht zur Stromflussrichtung am Leiter ab und ist direkt proportional zur magnetischen Flussdichte *(Hering/Schönfelder, 2018*). Durch die Veränderung der induzierten Spannung kann die Stromstärke im Leiter gemessen werden.

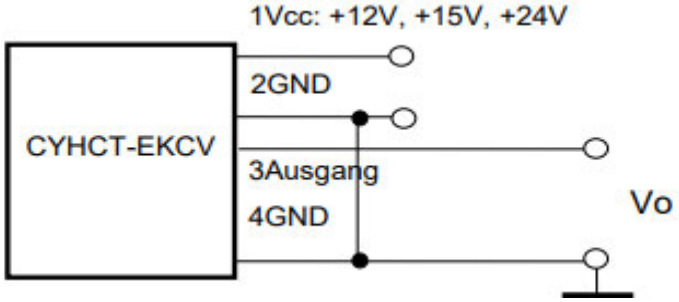


Abbildung 10: Anschlussbelegung des Stromsensors (Jigou Liu, Datenblatt, 2016)

Die Abbildung 10 zeigt die Anschlussbelegung des Sensors. Die Stromversorgung erfolgt über die Pins 1 und 2. In diesem Modul sind Pin 2 und Pin 4 kurzgeschlossen. Der Signalausgang mit der Spannung 0 bis 5 V liegt am Pin 3.

Dieser Sensor erfüllt alle Anforderungen, die für dieses Projekt benötigt werden. Zudem ist der Sensor mit diesen Spezifikationen der günstigste auf dem Markt.

In der Tabelle 6 erfolgt die Auflistung von wichtigen technischen Daten des CYHCT-EKCV-U/B30-34 Stromsensors.

Tabelle 6: Technische Daten des CYHCT-EKCV-U/B30-34 Stromsensors, (Dr.-Ing. habil. Jigou Liu, 2016)

Linearität	< ±0.5% FS
Messbereich	0 ~ ±30 A
Betriebsspannung	24 V
Stromverbrauch	< 25 mA
Verbindungsschnittstelle	Analogausgang, 0 bis 5 V ±1.0%
Dimension/ Größe	46 x 36,8 x 16 mm

Spannungssensor

Abbildung 11: Spannungssensor SCK-M-U-1500V (PHOENIX CONTACT, 2022)

Das Spannungsmessmodul wird zu Messung von PV-Spannungen bis zu 1500 V DC eingesetzt. Dabei liegen die Widerstände Rx mit 20 MΩ parallel zu Masse, wie in der Abbildung 12 zu sehen ist, wodurch die Teilströme über diese Widerstände fließen. Die Teilströme werden gemessen und durch den Messumformer zu einem Analogsignal von 2 bis10 V umgewandelt.

Dieser Spannungssensor kann im Gegensatz zu vielen anderen auf dem Markt verfügbaren Sensoren eine Gleichspannung bis 1500 V DC messen. Aufgrund dieses Kriteriums wurde dieser für dieses Projekt ausgewählt.

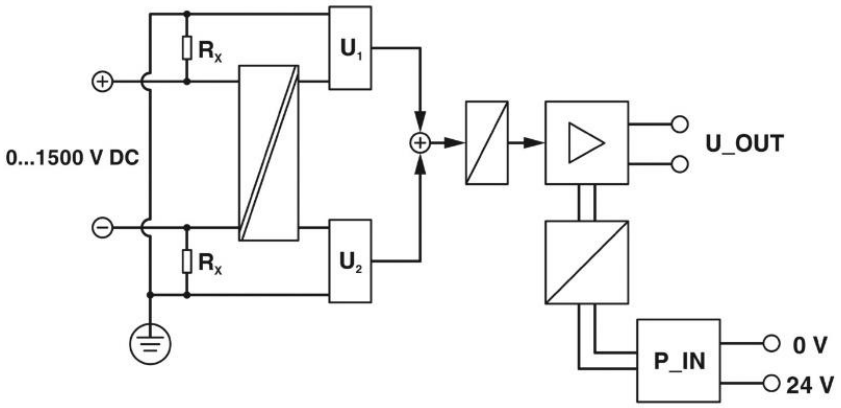


Abbildung 12: Anschlussbelegung des Spannungssensors (PHOENIX CONTACT, Datenblatt, 2013)

Die Stromversorgung des Moduls erfolgt über den P_IN Eingang und die Ausgabe des Signals über den U_OUT Ausgang.

In der Tabelle 7 werden die wichtigen technischen Daten des SCK-M-U-1500V Spannungssensors aufgelistet.

Tabelle 7: Technische Daten des SCK-M-U-1500V Spannungssensors, (PHOENIX CONTACT GmbH & Co. KG, 2013)

Linearität	±1% für 100 bis 1500 V DC
Messbereich	0 bis 1500 V DC
Betriebsspannung	24 V
Stromverbrauch	8 bis 65 mA
Verbindungsschnittstelle	Analogausgang, 2 bis 10 V ±1.0%
Dimension/ Größe	102 x 128,5 x 22,5 mm

3.3 Software

Die kostenlose Arduino-Entwicklungsumgebung ist neben dem Hardware-Teil der andere wichtige Teil der Arduino-Projekte. Dabei erlaubt die Arduino-IDE das Erstellen von Arduino-Sketches und das Hochladen und Testen auf dem Arduino-Board. Die Programmierstruktur von Arduino kann in drei Hauptteile unterteilt werden: Struktur, Werte und Funktionen.

In diesen Unterabschnitten wird ausschließlich auf die neue, zum aktuellen Zeitpunkt verfügbare Softwareversion 1.8.19 in Verbindung mit einem Windows System eingegangen und die wichtigsten Funktionen der Arduino-IDE und deren Anwendung erklärt.

3.3.1 Arduino-IDE

Angeboten wird die kostenlose Software Arduino-IDE auf der Homepage arduino.cc und ständig erneuert. Nach der Installation der Software müssen einige Einstellungen vorgenommen werden, um eine reibungslose Kommunikation mit dem Arduino-Board zu ermöglichen. Damit die Arduino-Entwicklungsumgebung und der lokale Computer mit dem Arduino Due kommunizieren können, wird der ARM-Treiber benötigt. ARM steht für Advanced RISC Machines und der Arduino Due gehört zu den Mikrokontrollern mit der ARM-Architektur. Unter dem Punkt *Werkzeuge > Board: > Boardverwalter…* gelangt man in den Treiber-Katalog. Der richtige Treiber für den Arduino Due nennt sich „Arduino SAM Boards (32-bits ARM Cortex-M3)“. Dieser muss angewählt und mit der aktuellen Version 1.6.12 installiert werden. Nach der Installation ist das richtige Board unter dem Menüpunkt *Werkzeuge > Board: > Arduino ARM (32-bits) Boards > Arduino Due (Programming Port)* zu sehen. Für den Arduino Due muss der Menüpunkt ausgewählt sein. Verbindet man nun das Board mit einem USB-Kabel am Mikro-USB und der USB-A Schnittstelle an den Computer, taucht automatisch unter dem Punkt *Werkzeuge > Port* der aktuelle COM-Anschluss auf. Dieser nennt sich COMx (Arduino Due) X steht hier für einen der USB-Ports und wird mit einer Zahl wiedergegeben. Die Zahl bei jedem USB-Port am Computer kann unterschiedlich dargestellt sein. Verbindet man mehrere Arduino-Boards, kann man diese durch Umschalten der COM Eingänge einzeln ansteuern.

Nachfolgend wird das Arduino-IDE Fenster aus der Abbildung 13 beschrieben.

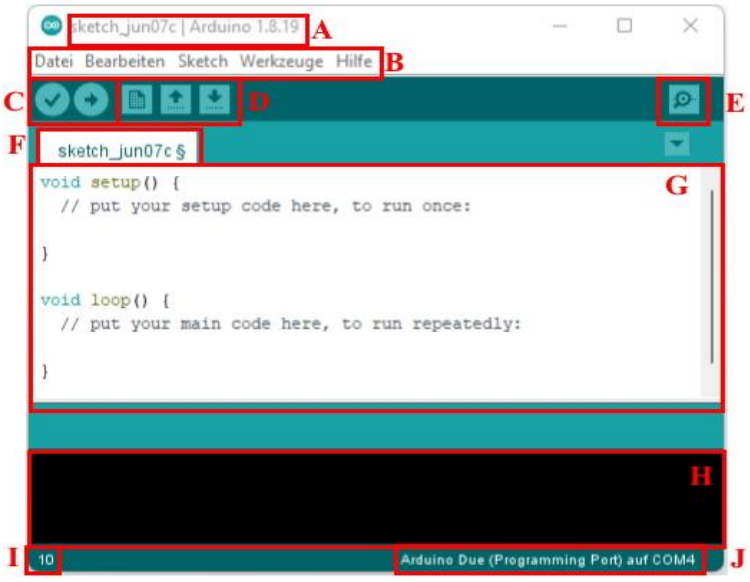


Abbildung 13: Arduino-IDE Fenster

A) Fenstername/ Versionsnummer: Zeigt den Namen des Projekts, unter der dieser gespeichert ist und gleichzeitig die aktuell benutzte Arduino-IDE Version an.

B) Reiterkarte: Unter dieser Reiterkarte befinden sich die Menüpunkte *Datei, Bearbeiten, Sketch, Werkzeuge* und *Hilfe*. Unter *Datei* können neue Projekte erstellt und gespeichert werden oder von der Bibliothek gespeicherten Beispielsketches geladen werden. Zusätzlich findet man unter dieser Reiterkarte die Möglichkeit unter dem Punkt *Voreinstellungen* z.B. die Option, den automatischen Speicherort zu ändern. Der Menüpunkt *Bearbeiten* orientiert sich auf die Formatierung des Programmcodes im Programmierfenster **G**. Unter dem Reiterpunkt *Sketch* befindet sich die Option eigene Bibliothek, die im Abschnitt 3.3.6 noch erklärt wird. Auch das Kompilieren oder das Laden des Sketchs auf das Board kann hier vorgenommen werden. Unter *Werkzeuge* können die Board-Treiber verwaltet, der COM-Anschluss definiert oder gewechselt und der serielle Monitor geöffnet werden. Der letzte Punkt in der Reiterkarte ist die *Hilfe*. Unter diesem Punkt wird auf die arduino.cc Forenseite und Erklärungsseite verwiesen, um weitere Hilfe zu dem Board und dem Arduino-IDE zu erhalten.

C) Kompilieren/ Hochladen (schnell Zugriffsfläche): Die *Kompilierung* oder auch Überprüfung genannt und als Häkchen dargestellt, kann nach dem Schreiben des Sketches durchgeführt werden. Dabei werden die Syntaxfehler im Fenster **H** angezeigt, bevor der Sketch aufs Board geladen wird. Bei der schnell Zugriffsfläche *Hochladen* als Pfeil nach rechts dargestellt, wird automatisch der Sketch kompiliert und auf das Arduino-Board geladen. Nach dem erfolgreichen oder nicht erfolgreichen kompilieren oder hochladen wird eine Bestätigungsmeldung oder eine Fehlermeldung in der Konsole ausgegeben. Beim Hochladen blinken zusätzlich die Kommunikations-Leuchtdioden „*RX*“ und „*TX*“ auf dem Board. Nach dem erfolgreichen Bespielen führt das Arduino-Board automatisch das Programm aus.

D) Datei erstellen, öffnen und speichern (schnell Zugriffsfläche): Unter dieser schnell Zugriffsfläche kann ein neuer Sketch in einem neuen Fenster erstellt, ein bestehender Sketch geöffnet und der aktuelle Sketch gespeichert werden.

E) Serieller Monitor (schnell Zugriffsfläche): Der serielle Monitor dient zur Echtzeitwiedergabe von Ein- und Ausgabewerten im laufenden Programm. Im nächsten Unterkapitel wird dieser näher erklärt.

F) Tabs: Es besteht die Möglichkeit, einzelne Sketche in verschiedenen Fenstern, sowie auch in demselben Fenster, aber dann in mehreren Tabs darzustellen.

G) Codeeditor: In diesem Fenster wird der eigentliche Code geschrieben. Beim Erstellen eines neuen Sketches wird automatisch die Funktion *void setup()* und *void loop()* erstellt. Zudem verfügt der Editor über das Syntax-Highlighting. Dabei erkennt die Syntaxhervorhebung automatisch einzelne Codeanweisungen und färbt diese farblich zur visuellen Unterstützung ein. Bei einer fehlerhaften Kompilierung wird die fehlerhafte Codestelle im Editor rot markiert.

H) Konsole: In diesem Ausgabefenster werden alle Ereignisse angezeigt, wenn z.B. beim Kompilieren oder beim Hochladen eines Sketches Fehler aufgetreten sind. Das Ausgabefenster ist nur ein Informationsbereich. Der Anwender kann hier keine Eingabe vornehmen.

I) Zeile: Zeigt die aktuelle Zeile, wo sich der Cursor momentan befindet.

J) Port: Zeigt den aktuell ausgewählten Port und den USB-COM an.

3.3.2 Serieller Monitor

Der serielle Monitor ist ein nützliches Werkzeug innerhalb der Arduino-IDE und gestattet die Ausgabe und Eingabe über den seriellen Port. Um den seriellen Monitor benutzen zu können, muss das Board, wie in Abschnitt 3.3.1 beschrieben, erst konfiguriert und angeschlossen sein. Im Programmcode muss unter anderem eine virtuelle Schnittstelle erstellt werden. Die Konfigurierung erfolgt über diese Anweisung in der Setup-Funktion, die später unter 3.3.4 Programmstruktur näher beschrieben wird. Die Anweisung lautet:

$$Serial.begin(9600);$$

Der Wert 9600 steht für den 9600 Baud. Baud ist die sogenannte Schrittgeschwindigkeit, ein Maß für die Übertragungsgeschwindigkeit (*Bernstein, 2020*). Als Standard wird der 9600 Baud bevorzugt, kann aber z.B. auf 4800, 14400, 19200, … eingestellt werden. Insgesamt gibt es 15 verschieden festgelegte Baud-Raten. Wichtig ist, dass der zuvor definierte Baud im seriellen Monitor übereinstimmen soll.

Die serielle Ausgabe dient vor allem zur einfachen Ausgabe von Werten, um nach mathematischen Fehlern im Code zu suchen oder um zu erfahren, ob das Ergebnis richtig gerechnet wurde. Bei einer Anzeige von Texten oder Werten im Monitor, muss eine Ausgabeanweisung im Code erfolgen. Ein Beispiel möglicher Anweisung sind im folgenden Codeausschnitt 1 zu sehen.

```
void loop() {    // Endlosschleife des Hauptprogramms
Serial.println (20);                // Gibt „20“ aus
Serial.println ('A');               // Gibt „A“ aus
Serial.println („Hello world.“);     // Gibt „Hello world.“ aus
Serial.println (20, BIN);           // Gibt „10100“ in Binär aus
Serial.println (1.2345, 2);         // Gibt „1.23“ mit zwei Nachkomma aus
Serial.println (1.2345, 4);         // Gibt „1.2345“ mit vier Nachkomma aus

} // Wiederholung der Loop-Funktion
```

Codeausschnitt 1: Beispiel möglicher Ausgabeanweisung von Serial.print

In der Abbildung 14 sind nacheinander die Ausgaben, die zuvor in der Loop-Funktion bestimmt wurden, in einer unendlichen Schleife angezeigt. Außerdem kann hier im Bild unten rechts die zuvor beschriebenen Baud geändert werden.

Ansonsten ist es möglich, auch Eingaben über den Monitor zu tätigen. Wie die Eingabe funktioniert, wird in diesem Kapitel nicht behandelt, da es keine Relevanz für dieses Projekt hat.

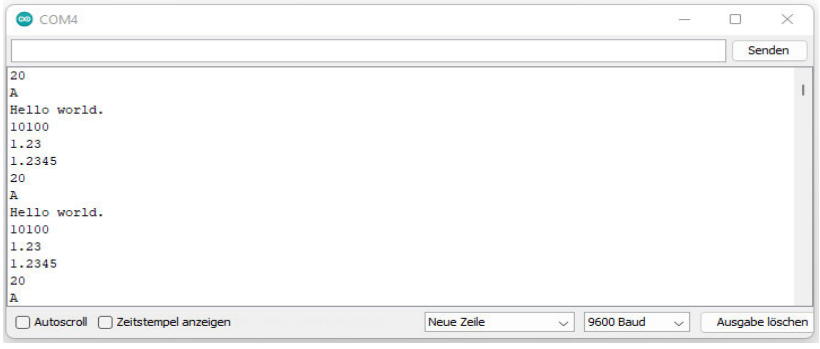


Abbildung 14: Serieller Monitor mit Ausgabe

3.3.3 Datentyp

Ein Datentyp beschreibt mögliche Operationen und den erlaubten Wertebereich einer Variablen. Jeder Datentyp einer bestimmten Variablen verwendet unterschiedlich viel Speicherplatz und muss beim Programmieren diesbezüglich berücksichtigt werden, um möglichst Speicherplatz zu sparen.

In der Tabelle 8 werden die wichtigsten Datentypen und deren Wertebereiche aufgelistet, die für die Programmierung der Arduino-Sketche benötigt werden.

Tabelle 8: Wichtigste Datentypen der Arduino-Programmierung, (Nahrstedt, 2009)

Datentyp	Speicherbedarf	Wertebereich	Erklärung
boolean	1 Byte	„true“/ „false“	Wert für „true“ = 1/ Ja/ Ein Wert für „false“ = 0/ Nein/ Aus
char	1 Byte	-128 bis 127	Wert für die Speicherung eines Zeichens
byte	1 Byte	0 bis 255	Kleiner Speicherbedarf dank niedriger Werte
int	2 Byte	-32.768 bis 32.767	Ganzzahlige Werte ohne Kommastellen
unsigned int	2 Byte	0 bis 65.535	Positive Ganzzahlige Werte ohne Kommastellen
long	4 Bytes	-2.147.483.648 bis 2.147.483.647	Ganzzahlige hohe Werte
unsigned long	4 Bytes	0 bis 4.294.967.295	Positive Ganzzahlige hohe Werte
float	4 Bytes	$\approx -3.4*10^{38}$ bis $3,4*10^{38}$	Werte mit Kommastellen (7 Dezimalstellen)
double	8 Bytes	$\approx -1.7*10^{308}$ bis $1.7*10^{308}$	Werte mit Kommastellen (15 Dezimalstellen)
string	je nach Bedarf	Länge je nach Definition	Zeichenkette (Text oder mehrere Zeichen)
array	je nach Array-Größe	Array mit Größe gemäß Definition	Nummerierte Tabelle vom selben Datentyp

3.3.4 Programmstruktur

Das Arduino-Programm teilt sich in zwei Strukturen auf: *„void setup()“* und *„void loop()“*.

```
void setup() {            // Programmstart
 // Einmalige Anweisung
}
void loop() {             // Hauptschleife
 // Wiederholte Anweisung
}
```

Codeausschnitt 2: Struktur eines Arduino-Sketches

Beim Start oder nach dem Reset des Arduino-Boards läuft der Programmcode typischerweise von oben nach unten ab. Dabei wird die Setup-Funktion einmalig durchlaufen. In dieser Funktion werden die Grundeinstellungen wie Deklaration, Initialisierung, das Setzten von Ein- und Ausgängen oder die Konfiguration der seriellen Schnittstelle vorgenommen, wie im Codeausschnitt 3.1 bereits zu sehen war. Die Setup-Funktion ist in einem Sketch immer notwendig und darf nicht entfernt werden. Bei Nichtbenutzung dieser Funktion bleibt diese ohne Anweisungen leer zurück.

```
int LedAusgang = 13;          // LED an Digital Pin 13
int ButtonEingang = 1;        // Knopf an Digital Pin 1

void setup() {
 pinMode(LedAusgang, OUTPUT);     // Digital Pin 13 als Ausgang
 pinMode(ButtonEingang, INPUT);   // Digital Pin 1 als Eingang
 Serial.begin(9600);              // Initialisierung der seriellen Schnittstelle
}
```

Codeausschnitt 3.1: Beispiel der Definition von Ein- und Ausgängen und Konfiguration der seriellen Schnittstelle

Im zweiten Bereich der Arduino-Sketch Struktur ist die Loop-Funktion mit der Aufgabe eines Hauptprogramms. Die Loop-Funktion startet erst und läuft als Endlosschleife, wenn die Setup-Funktion durchlaufen ist. In dieser Endlosschleife werden Funktionsaufrufe und Anweisungen geschrieben, die für die gewünschte Anwendung benötigt werden, siehe Codeausschnitt 3.2.

```
void loop() {    // Endlosschleife des Hauptprogramms
  digitalWrite(LedAusgang, HIGH);     // LED einschalten
  Serial.println („LED ist an!“);      // serielle Ausgabe
  delay(1000);                         // 1 Sekunde warten
  digitalWrite(LedAusgang, LOW);      // LED ausschalten
  Serial.println („LED ist aus!“);     // serielle Ausgabe
  delay(1500);                         // 1,5 Sekunden warten
} // Wiederholung der Loop-Funktion
```

Codeausschnitt 3.2: Beispiel Hauptendlosschleife mit serieller Ausgabe

3.3.5 Funktionen

Funktionen werden verwendet, um bestimmte Operationen auszuführen und sind innerhalb eines Programms ein kleineres geschlossenes Programm, welches über seinen Namen aufgerufen wird. Innerhalb einer Funktion werden die Werte verarbeitet und als Ergebnis übergeben. Diese Werte werden als Parameter bezeichnet. Wird die Funktion durch ihren Namen im Hauptprogramm angesprochen, wird automatisch die aufgerufene Funktion ausgeführt, die eine Umwandlung, eine Berechnung oder eine einfache Signalausgabe auf einen Ausgangspin zur Folge hat. Ist die Funktion beendet, wird automatisch das Hauptprogramm weiter an der Stelle ausgeführt, an der die Funktion vorher angesprochen wurde.

Der Grundaufbau einer Funktion sieht wie folgt aus:

```
Typ NameDerFunktion (Parameter) {
  Rückgabewert x;
}
```

Codeausschnitt 4: Grundaufbau einer Funktion

Der Typ einer Funktion gibt die Art des Wertes an, die die Funktion zurückgibt. Welche Datentypen zu Verfügung stehen, wurde im vorherigen Unterkapitel beschrieben. Diesbezüglich wird der Datentyp des Rückgabewerts im Aufruf der Funktion vor dem Funktionsnamen gesetzt.

Besitzt eine Funktion keinen Rückgabewert, wird für die Typenbezeichnung ein *„void“* als Schlüsselwort verwendet. In dem Codeabschnitt 4.1 wird ein Beispiel mit samt der Hauptfunktion gezeigt.

```
void loop() {                          // Hauptprogramms
  AusgabeWert(Sensor);                 // Aufruf der Funktion
}
void AusgabeWert (Sensor) {            // Funktion
  wert = AnalogRead(Sensor);           // Liest den Wert am Analogpin
   Serial.println(wert);               // Gibt den Wert am Seriellen Monitor aus
}
```

Codeausschnitt 4.1: Beispiele einer Funktion ohne Rückgabewert

3.3.6 Bibliotheken

Bibliotheken im Englischen „Libraries“ oder „Library“ (Einzahl) werden von den Arduino-Boards in Form von Software genutzt. Es handelt sich um geschlossene Programme, die meist eine Reihe auf die jeweilige Bibliothek speziell erstellte Funktionen mitbringen. Der Vorteil für den Anwender ist, dass er auf die jeweiligen Funktionen zugreifen kann und nicht eigenständig programmieren muss. Beispielweise kann der Anwender mit einer Bibliothek für das Echtzeit-Modul die Uhr nach Stunde, Minute, Sekunde einzeln abfragen und muss nicht die jeweiligen Werte des Speichers auslesen und umrechnen. Deshalb werden die Bibliotheken oft bei externen Sensoren und anderen elektronischen Schaltungen mit dem Arduino eingesetzt.

Der Arduino-IDE bietet für die Arduino-Bibliotheken ein eigenes Verzeichnis. Der Pfad hierzu lautet auf dem lokalen Computer:

Arduino-Verzeichnis\libraries

Ein Neustart der Arduino-IDE, ist erforderlich, wenn eine neue Bibliothek in dem Unterverzeichnis abgelegt wurde. Die jeweilige Bibliothek kann im Menü über den Punkt *Sketch > Bibliothek einbinden* in den aktuellen Sketch eingebunden werden. Die in dem Sketch geladenen Bibliotheken werden am Anfang des Codes mit der Anweisung „include“ eingelesen.

$$\#include\ <RTClib.h>$$

Dass manuelle einfügen der Bibliotheken direkt als ZIP-Datei ist ab der Arduino IDE Version 1.6 möglich. Ausführbar ist es über den Punkt *Sketch > Bibliothek einbinden > .ZIP-Bibliothek hinzufügen*. Ab der Version 1.6.2 gibt es unter dem Punkt *Werkzeuge > Bibliotheken verwalten* die Bibliotheksverwaltung. Diese ermöglicht aus einem online Katalog nach Bibliotheken zu suchen, herunterzuladen und bestehende Bibliotheken zu aktualisieren.

Für das Verständnis, welche Funktionen eine Bibliothek hat, bringen diese oft Beispiele, die die Nutzung der Bibliothek darlegen. Eine detailreichere Beschreibung dieser Funktionen findet man auf den Homepages arduino.cc oder Github.com.

Angehend werden die in dem Projektcode verwendeten Bibliotheken und im Sketch verwendete Funktionen beschrieben.

Alle in diesem Projekt verwendeten Bibliotheken sind im digitalen Anhang 11 zu finden.

Echtzeituhr-Bibliothek

Mit der Echtzeit-Bibliothek <RTClib.h> und dem RTC DS3231 Modul kann die Zeit und das Datum eingelesen, abgespeichert und abgerufen werden. Beim ersten Betrieb des Moduls muss die Uhrzeit und das Datum definiert werden. Dies geschieht mit folgenden Funktionen:

$$rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));$$

Die Zeit wird aus der Systemzeit des verwendeten Computers ausgelesen und in den Speicher des Moduls geladen oder mit der nächsten Funktion die Uhrzeit und das Datum nach dem Schema Jahr, Monat, Tag, Stunde, Minute, Sekunde in den Speicher geladen.

$$rtc.adjust(DateTime(2022, 6, 21, 15, 0, 0));$$

Mit der Klasse *„DateTime“* lässt sich das Datum und die Uhrzeit definieren.

$$DateTime\ myTime\ =\ rtc.now();$$

Es wird ein Objekt *„myTime“* der Klasse *„DateTime“* erzeugt und ordnet diesem Objekt mit *„rtc.now()“* die aktuelle Uhrzeit und das Datum zu. Um nun auf das Jahr, den Monat, den Tag, die Stunde, die Minute und die Sekunde aus der *„DateTime“*-Variablen zuzugreifen, gibt es folgende Befehle:

$$now.year();$$
$$now.month();$$
$$now.day();$$
$$now.hour();$$
$$now.minute();$$
$$now.second();$$

Die Uhrzeit und das Datum lassen sich auch mit den folgenden Befehlen in einem Zeit-Format abspeichern:

$$char\ myTime[] = "hh{:}mm{:}ss";$$
$$char\ myDate[]\ =\ "DD.MM.YYYY";$$

Hier kann *„hh:mm:ss“* und *„DD.MM.YYYY“* beliebig umgestellt und verändert werden, um das gewünschte Zeit Format zu erreichen wie z.B. *„hh-mm-ss“* und *„YY/MM/DD“*.

Speicherkarten-Bibliothek

Mit der Speicherkarten-Bibliothek <SD.h> lassen sich die vorher erfassten Daten auf einer Speicherkarte speichern, auslesen und ändern. Die Bibliothek und die Karte müssen beim Beginn des Sketches mit der folgenden Grundfunktion initialisiert werden:

$$SD.begin();$$

Wenn die SD-Karte richtig in das Karten Modul eingelegt wurde und die Formatierung korrekt ist, wird ein Wert *„true“* oder wenn das nicht der Fall sein sollte, ein *„false“* zurückgegeben.

$$SD.exists(SDname);$$

Diese Funktion prüft, ob auf der Karte die Datei mit dem Dateinamen *„SDname“* vorhanden ist. Zurück kommt ein *„true“*, wenn die Datei existiert oder ein *„false“*, wenn die Datei nicht vorhanden ist.

$$mySD = SD.open(SDname, FILE\_WRITE);$$

Bei dieser wichtigen Funktion öffnet die Datei mit dem Namen *„SDname“* und bereitet diese zum Schreiben vor. Existiert die Datei mit dem Namen nicht, wird durch den Zusatz Befehl *„FILE_WRITE“* eine neue Datei mit dem Namen erstellt und zum Schreiben vorbereitet.

$$mySD.print(now.minute(), DEC);$$

Als Beispiel dient diese Funktion, die die Minuten in die Datei schreibt. Es können alle beliebigen Werte und Texte in die Datei geschrieben werden. Mit der Endung „*DEC*“ werden die Minuten als dezimal Zahl gespeichert.

$$mySD.close();$$

Am Ende des Schreibcodes muss immer mit der folgenden Funktion die zu schreibende Datei geschlossen werden. Geschieht das nicht, kann beim erneuten Wiederholen des Codes nicht auf die Datei zugegriffen werden.

LCD-Bibliothek

Mit der Hilfe der LCD-Bibliothek <LiquidCrystal_I2C.h> kann man einfache LC-Displays mit mehreren Zeilen und Spalten Ansteuern. Dabei muss man erwähnen, dass nur die Bibliothek des Displays mit dem Hitachi HD44780 Chip funktioniert.

$$LiquidCrystal\_I2C\ lcd(0x27,20,4);$$

Zu Anfang wird ein Objekt mit dem Namen „*lcd*“ erstellt, mit der Adresse „*0x27*“ und die Zeichen und Zeilen definiert. Wie man genau die Adresse ändert, wurde bereits im Unterkapitel 3.2.2 behandelt.

$$lcd.init();$$

Mithilfe dieser Funktion wird das LCD initialisiert. Hier werden Anfangswerte den LCD-Variablen zugewiesen, um keine Fehler bei der Ausgabe zu haben.

$$lcd.backlight();$$

In dieser Funktion wird die Hintergrundbeleuchtung des LCD eingeschaltet.

$$lcd.setCursor(12, 1);$$

Setzt den Cursor auf das zwölfte Zeichen und die erste Zeile und stellt an dieser Stelle mit der Funktion

$$lcd.print("TEXT");$$

den gewünschten Text auf dem Display dar.

$$lcd.clear();$$

Diese Funktion löscht die gesamte Ausgabe auf dem Display. Möchte man einzelne Bereiche löschen, muss man mit der zuvor beschriebenen Funktion *„setCurser“* die gewünschte Stelle markieren und mit der *„lcd.print(" ");“* Funktion diesen Bereich löschen. Dabei überschreiben die Leerzeichen in den Anführungszeichen die zuvor markierte Stelle.

OneButton-Bibliothek

Die OneButton-Bibliothek <OneButton.h> erleichtert die Nutzung einzelner Knöpfe mit dem Arduino-Board. Es kann auf diese Bibliothek verzichten werden, wenn die Realisierung mit eigenem Code zu bewerkstelligen ist.

Die Initialisierung eines Knopfes erfolgt mit der Funktion:

$$OneButton\ button\ =\ OneButton(buttonPin, false);$$

Der am Arduino-Board angeschlossene Pin, ist als *„buttonPin“* deklariert. Der zweite Parameter in den Klammern gibt an, ob der ungedrückte Zustand ein LOW- oder HIGH-Pegel entspricht. Das *„false“* in der Funktion steht dafür, dass ein Pulldown-Widerstand verwendet wurde. Ist kein Widerstand vorhanden, stellt man den Parameter auf *„true“* um.

$$button.attachLongPressStart(LongPress);$$

Die Funktion *„attachLongPressStart“* startet nur einmal, wenn der Knopf länger gedrückt wurde. Einmalig wird daraufhin die eigen erstellte Funktion *„LongPress“* ausgeführt.

$$button.attachClick(Click);$$

Die Aktivierung der Funktion erfolgt erst, wenn einmal auf den Knopf gedrückt wurde und führt eine weitere selbsterstellt Funktion *„Click“* einmal aus.

$$button.tick();$$

Diese Funktion muss in die Haupt *„void loop()“* Funktion geschrieben werden. Hiermit werden nach jedem loop Vorgang die Funktionen *„attachLongPressStart“* und *„attachClick“* in der *„void setup()“* Funktion überwacht und ausgeführt, wenn der Knopf betätigt wird.

String-Bibliothek

Die String-Bibliothek <String.h> erstellt eine Instanz der String-Klasse. Dadurch kann ein String aus mehreren verschiedenen Datentypen erstellt werden. Darunter fallen die Datentypen *int, long int, float* oder *double* die miteinander verknüpft werden können. Dabei wird mit dem Befehl:

$$date\ =\ String(now.toString(buf))\ +\ ".csv";$$

ein String mit dem Namen *„date“* erstellt. Der String beinhaltet das Datum Format als *„char“* und mit dem + hinzugefügten Endung *„.csv“*. Diese Bibliothek ist von Anfang an in der Arduino-IDE vorhanden.

MapFloat-Bibliothek

Die mathematische Standardfunktion *„map()“* ist im Arduino-IDE integriert. Dabei bildet die Funktion eine Zahl von einem Bereich in einen anderen ab.

$$map(value, fromLow, fromHigh, toLow, toHigh)$$

Die Zahl, die zugeordnet werden soll, wird als *„value“* dargestellt. Die untere Grenze des aktuellen Wertebereichs als *„fromLow“* und die obere Grenze als *„fromHigh“*. Das Ganze wird in dem Bereich *„toLow“* zu der unteren Grenze und *„toHigh“* in die obere Grenze des Zielbereichs des Wertes abgebildet. Wichtig ist, dass diese mathematische Funktion nur mit positiven und negativen ganzen

Zahlen arbeitet. Integriert man die Bibliothek <MapFloat.h> so lassen sich auch Nachkommastellen bewerkstelligen. Diese sieht wie folgt aus:

$$mapFloat(value, fromLow, fromHigh, toLow, toHigh)$$

3.4 Tiefpassfilter

Ein Tiefpass wird in der Elektrotechnik eingesetzt, um hohe Frequenzen bei der Signalverarbeitung zu sperren oder abzuschwächen und niedrige Frequenzen ungehindert passieren zu lassen. Auch die Bezeichnung Tiefpassfilter oder passiver Tiefpassfilter sind geläufig. Passiv bedeutet, dass die Schaltung zur Signalverarbeitung ohne einen Operationsverstärker verarbeitet wird.

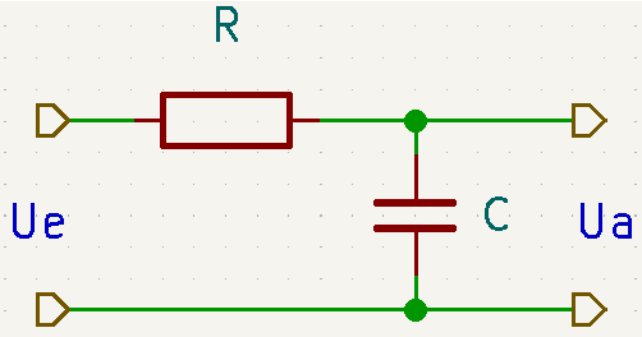


Abbildung 15: RC-Tiefpass 1. Ordnung, (Erstellt mit KiCad 6.0)

In der Abbildung 15 ist ein solcher einfacher Tiefpass zusehen. Der Tiefpass 1. Ordnung besteht aus einem Widerstand *R* und einem Kondensator *C*. Diesbezüglich ist auch der Name RC-Tiefpass gebräuchlich. Entsteht eine schnelle Änderung der Eingangsspannung *Ue*, fällt am Kondensator nahezu keine Spannung ab, wodurch sich auch die Ausgangsspannung *Ua* nahe 0 bewegt. Bei einer langsamen Änderung der Spannung *Ue* fällt ein Teil der Spannung über den Kondensator ab. Das führt dazu, dass die Ausgangsspannung *Ua* sich mit einer Verzögerung in der gleichen Amplitude verändert. Bei sinusförmigen Eingangsspannung, die entstehen können, bei hohen Kabellängen oder unzureichender Abschirmung, erhält man eine abgeschwächte Ausgangsspannung. Wie stark das Ausgangssignal gedämpft wird, hängt von der Frequenz des Eingangssignals und der dadurch aufbauende kapazitive Blindwiderstand *Xc* des Kondensators ab. Die Differenz zwischen Eingangs- und Ausgangsspannung steigt mit steigender Eingangsfrequenz.

Die gedämpfte Ausgangsspannung kann mit der folgenden Formel berechnet werden:

$$U_a = \frac{1}{\sqrt{1 + (\omega * C * R)^2}} * U_e$$

$$Kreisfrequenz\ \omega = 2 * \pi * f$$

Mit U_e Eingangsspannung in V,

C = Kapazität des Kondensators in F,

R = Widerstand in Ω,

f = Frequenz in Hz.

Die Grenzfrequenz wird als die Frequenz bezeichnet, bei der die beiden Werte $R = X_C$ gleich groß sind. Bei einer niedrigen Frequenz ist X_C größer als R, bei höherer Frequenz ist dies umgekehrt. Die Grenzfrequenz bei einem RC Tiefpass erfolgt mit der Formel:

$$f_g = \frac{1}{2 * \pi * R * C}$$

Dabei steht f_g für die Grenzfrequenz in Hz.

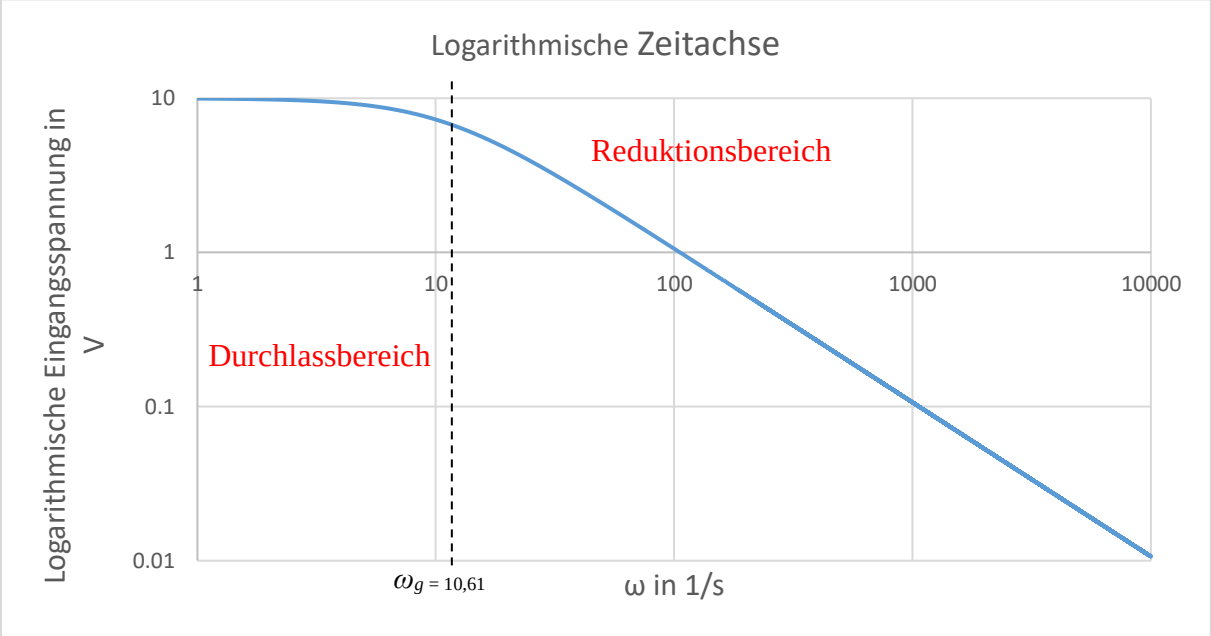


Abbildung 16: Logarithmisches Diagramm der Grenzfrequenz (Liniendiagramm)

In der Abbildung 16 sind die Werte R = 15 kΩ und C = 1 µF angenommen, die auch im späteren Verlauf des Projektes von Bedeutung sind. Die Widerstände für den Spannungsteiler werden in dem Diagramm nicht berücksichtigt. Der Widerstand und der Kondensator sind so gewählt, dass sich eine Grenzfrequenz von 10,61 Hz ergibt. Die späteren Messungen werden unter 10 Hz erfasst und landen im Durchlassbereich. Alle Störsignale, die oberhalb 10 Hz liegen im Reduktionsbereich. Der Tiefpassfilter dämpft das Signal mit der Grenzfrequenz um den Faktor 0,7 bedingt durch den Scheitelfaktor $\frac{1}{\sqrt{2}}$.

4 Entwurf, Signalverarbeitung und Zusammenbau

In diesem Kapitel wird der Entwurf des Messgeräts beschrieben sowie die dazugehörigen Signalkabel, die Spannungsversorgung und die Aufnahmedauer.

4.1.1 Entwurf des Messgeräts

Abbildung 17 zeigt das konzipierte Erscheinungsbild des Messgeräts. Das Gehäuse, in dem sich der Arduino Due und die zusätzliche Elektronik befindet, ist aus einem stabilen, wasserdichten Polypropylen-Koffer mit den Außenmassen 265 x 240 x 125 mm.

Abbildung 17: Konzipiertes Erscheinungsbild des Messgeräts, (Erstellt mit Google SketchUp)

Die an der linken Seite platzierten Buchsen ermöglichen den Anschluss der Sensoren. Ein Stecker hinter der Box dient der Stromversorgung über das Netzteil. Der Koffer und die Buchsen sind so gewählt, dass problemlos über einen längeren Zeitraum unter verschiedenen Wetterbedingungen, unbeschadet Daten aufgenommen werden können. Im Inneren befindet sich das Display, der Micro-SD-Slot, der An/Aus Schalter und der Start/Stopp Knopf zum Starten der Logger Funktion. Auf der linken Seite befindet sich außerdem eine Aussparung für das Netzteil und das notwendige Netzkabel. Zukünftig kann diese Aussparung auch als Platzhalter für einen Akku dienen, um auf die Außenversorgung zu verzichten.

4.1.2 Spannungsversorgung

Die Spannungsversorgung erfolgt über ein spezielles SELV Netzteil. SELV steht für „Safety Extra Low Voltage“ zu Deutsch „Schutz durch Sicherheitskleinspannung“. Das SELV-Netzteil besitzt keine Erdung, sowie keine Verbindung zu einem Schutzleiter. Aufgrund der geringeren Spannungshöhe und der Isolierung ist ein besonderer Schutz gegen elektrischen Schlag nach *IEC 61558-2-6, 2010* gewährleistet. Die Ausgangsspannung des Netzteils beträgt 24 V und ermöglicht eine direkte Versorgung der Messsensoren. Die Versorgung des Arduino Due erfolgt mit einem DC/DC-Wandler TSR1-24120 von 12 V.

4.1.3 Signalkabel

Entlang des Stranges werden die Einstrahlungssensoren bündig befestigt, wie in der Abbildung 2 (Problemstellung) zu sehen ist. Dadurch, dass die horizontal verlaufenden PV-Stränge eine Länge von ca. 30 m haben können, müssen dementsprechend auch die Signalkabel lang genug sein. Es ergeben sich Längen von:

- 30 m für Einstrahlungssensor 1
- 25 m für Einstrahlungssensor 2
- 20 m für Einstrahlungssensor 3
- 15 m für Einstrahlungssensor 4
- 5 m für den Strom- und Spannungssensor

Die Kabel sind mit jeweils einem wasserdichten Aviation Stecker, mit einer Größe von GX16 - 4 Pin verlötet, um beim Transport die Sensoren und die Messeinheit vom Kabel trennen zu können. Zudem können die Kabel auch im Falle eines Defekts oder, falls längere Kabel benötigt werden, schnell ausgetauscht werden. Die vierädrigen Kabelstränge mit einem Querschnitt von 0,14 mm^2 sind darüber hinaus, mit einer Abschirmung gegen Störsignale ausgestattet.

Ebenfalls werden auch die Sensoren über die langen Kabel mit Spannung versorgt. Durch den Widerstand der langen Kabel kommt es bei der Versorgung der Sensoren zu einem Spannungsabfall. Der Kabelwiderstand wurde mit einem Multimeter UNI-T UT136b gemessen.

- 7,8 Ω bei 30 m Kabellänge
- 6,8 Ω bei 25 m Kabellänge
- 5,2 Ω bei 20 m Kabellänge
- 4,0 Ω bei 15 m Kabellänge
- 1,2 Ω bei 5 m Kabellänge

4.1.4 Speicherplatz

In diesem Projekt werden die aufgezeichneten Daten auf eine 16 GB Speicherkarte gespeichert. Die reale Kapazität beträgt ca. 14.8 GB. Wie viel Speicher bei einem Messintervall von 0,5 bis 10 Sekunden benötigt wird, zeigt die Tabelle 9. Die Rechnung der Speicherdauer bezieht sich auf eine CSV (Character-separated values)-Datei. In diesem Datenformat werden die Daten auf der Speicherkarte gespeichert. Folgende Werte werden in der Datei erfasst: Zeit, Datum, Messdaten von den

Einstrahlungssensoren eins bis vier, Messdaten vom Stromsensor und dem Spannungssensor. Die Datenmenge kann deutlich reduziert werden, wenn der Programmcode die momentane Einstrahlung misst und bei niedriger oder keiner Einstrahlung, wie z.B. nachts, die Datenaufnahme pausiert. Resultierend ergibt sich eine Tagesaufzeichnung von durchschnittlich 12 Stunden.

Tabelle 9: Speicherverbrauch und Speicherdauer auf einer 16 GB Speicherkarte

Mess-Intervall \ Zeit	1 Stunde	12 Stunden	24 Stunden	Maximale Speicherdauer bei 12 Stunden	Maximale Speicherdauer bei 24 Stunden
0,5 Sekunden	0,98 MB	4,86 MB	8,38 MB	30 Tage	17 Tage
1 Sekunde	0,49 MB	2,43 MB	4,19 MB	60 Tage	35 Tage
2 Sekunden	0,25 MB	1,22 MB	2,10 MB	121 Tage	70 Tage
5 Sekunden	0,10 MB	0,49 MB	0,84 MB	302 Tage	176 Tage
10 Sekunden	0,06 MB	0,28 MB	0,48 MB	528 Tage	308 Tage

4.2 Anschluss der Hardware

In diesem Abschnitt wird zuerst auf die Pinbelegung des Arduino Due eingegangen und drauf folgend die Maßnahmen zur Signalumwandlung der Sensoren ausführlicher erklärt.

4.2.1 Anschließen von Modulen am Arduino

Die verschiedenen Module, die in den Grundlagen aufgezählt wurden, haben verschiedene Kommunikations-Standards. Das Display und die Echtzeituhr kommunizieren über das I^2C, das Micro-SD-Modul dagegen über die SPI Schnittstelle. In der Tabelle 10 werden die Schnittstellen genannt und wie diese mit den Pins, auf dem Arduino-Board, verbunden werden.

Tabelle 10: Pinbelegung der Module am Arduino Due

Modul		Arduino Due
Alle Module	GND	GND
	VCC	5 V
RTC-Modul LCD-Modul	SCL	D21
	SDA	D20
Micro-SD-Modul	CS	D10
	SCK	(SPI) SCK
	MOSI	(SPI) MOSI
	MISO	(SPI) MISO

Wichtig ist zu erwähnen das I^2C Schnittstelle eine Open-Collector (OC) Schaltung ist. Kommunizieren mehrere Module in Master und Slave an derselben Schnittstelle, kann es passieren, dass z.B. der Master ein HIGH und der Slave ein LOW sendet. Dadurch kann ein Kurzschluss in der Datenleitung entstehen, was zu einer Beschädigung des Mikrokontroller führen kann. Vorbeugend, müssen die Pins SCL und SDA mit einem Pull-Up Widerstand in der Regel 4,7 bis 10 kΩ versehen werden. Die in diesem Fall verwendeten Module haben laut Datenblatt bereits integrierte Pull-Up Widerstände, deswegen kann diese Maßnahme vernachlässigt werden.

4.2.2 Anschlüsse der Sensoren am Arduino Due

Die drei verschiedenen Sensorarten geben unterschiedliche analoge Spannungs- und Strom-Signale ab, wie bereits unter dem Kapitel Grundlagen erklärt. Dabei können diese Signale nicht direkt an den Arduino angeschlossen werden, da der Arduino Due nur Spannungssignale bis 3,3 V verarbeiten kann. Um die Signale von den Sensoren verarbeiten zu können, werden folgende elektronische Schaltungen benötigt.

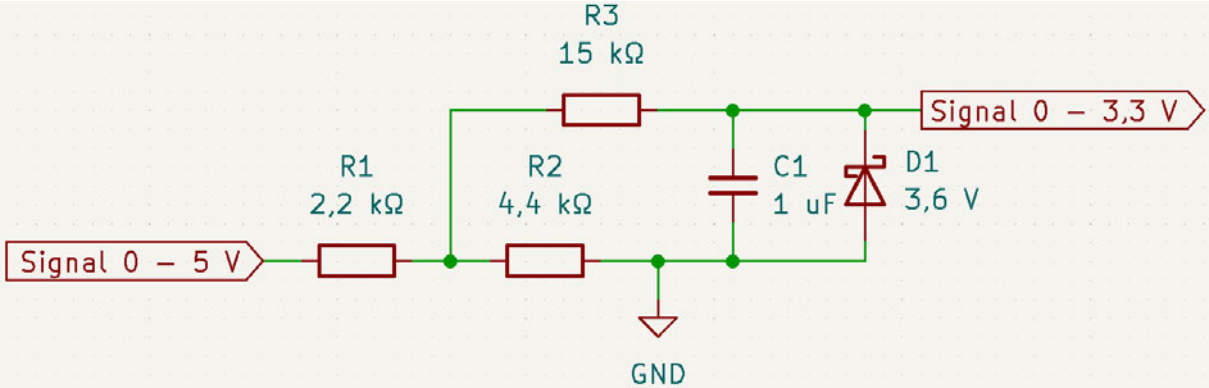


Abbildung 18: Spannungsteiler für den Stromsensor mit einem RC-Tiefpassfilter, (Erstellt mit KiCad 6.0)

Die Abbildung 18 zeigt den Aufbau der Signalverarbeitung vom Stromsensor zum Arduino Due. Das analoge Signal des Stromsensors ist von 0 bis 5 V hoch, deshalb muss dieses über einen Spannungsteiler, bis maximal 3,3 V gesenkt werden. Der Spannungsteiler wird mit der folgenden Formel berechnet:

$$U_{Signal} = U_{MAX} * \frac{R_2}{R_1 + R_2}$$

Mit U_{Signal} Eingangsspannung in V,

U_{MAX} = Maximalspannung in V,

R = Widerstand in Ω.

Mit den in der Abbildung 18 dokumentierten Widerstandswerten fällt eine Spannung U_{Signal} von 3,3 V bei einer Maximal-Spannung U_{MAX} von 5 V in der Theorie ab. Aufgrund, dass es sich um kein Stromsignal, sondern um ein Spannungssignal handelt, muss auch der Kabelwiderstand berücksichtigt werden. Die ohmsche Messung mit einem Multimeter ergab einen Kabelwiderstand von ca. 1,2 Ω. Dieser Widerstand ist aber so klein, dass er kaum eine Relevanz auf die Signalverarbeitung hat.

Die Zener-Diode IN4729A in der Abb.18 *D1* wird in der Sperrrichtung betrieben und dient zum Schutz vor höheren Spannungen als 3,6 V. Im Falle, dass in Folge eines Fehlers eine höhere Spannung als 3,6 V am Signaleingang aufläuft, wird die Durchbruchspannung nach dem Zener-Effekt (*siehe S. 61, Stiny, 2015*) erreicht und die Zener-Diode schließt den Stromkreis kurz. Die Widerstände *R1, R2, R3* fangen daraufhin den Kurzschlussstrom auf. Der Arduino darf zwar an den analogen Eingängen eine maximale Spannung von 3,3 V bekommen, dieser kann aber auch mit den zusätzlichen 0,3 V umgehen. Diese erhöhte Spannung wird dann aber nicht mehr in 12 Bit dargestellt.

Der Widerstand *R3* mit 15 kΩ und der Kondensator *C1* mit 1 µF bilden einen RC-Tiefpassfilter der 1. Ordnung. Berücksichtigt man die Widerstände *R1* und *R2*, so hat die Schaltung einen Gesamtwiderstand von 16,46 kΩ, wodurch sich eine Grenzfrequenz des Tiefpassfilters auf 9,67 Hz einstellt.

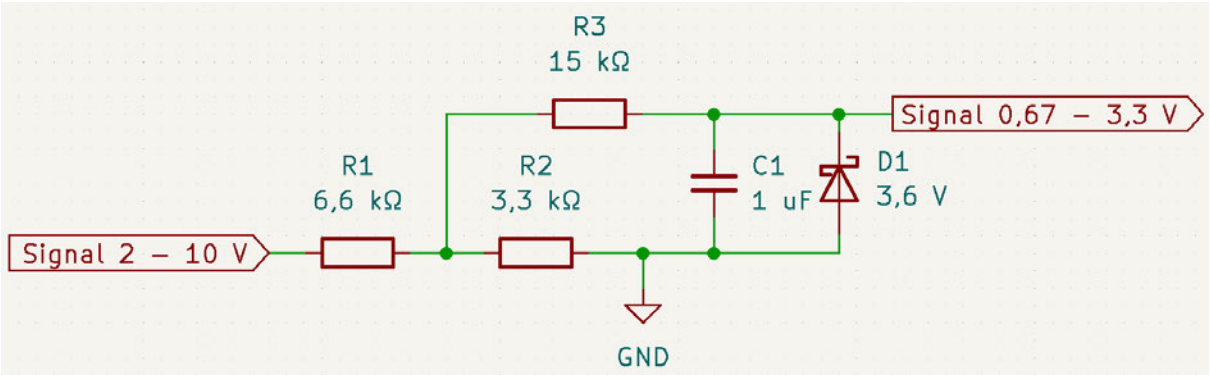


Abbildung 19: Spannungsteiler für den Spannungssensor mit einem RC-Tiefpassfilter, (Erstellt mit KiCad 6.0)

Der Spannungssensor liefert ein analoges Signal von 2 bis 10 V, wie in der Abbildung 19 zu sehen ist. Mit der obigen Formel für den Spannungsteiler und mit dem zusätzlich gleichen Kabelwiderstand wird auch hier ein maximales Ausgangssignal von 3,3 V erzielt. Zum Schutz vor erhöhter Spannung wird ebenfalls eine Zener-Diode in der Schaltung verwendet.

Die Grenzfrequenz liegt bei 9,25 Hz.

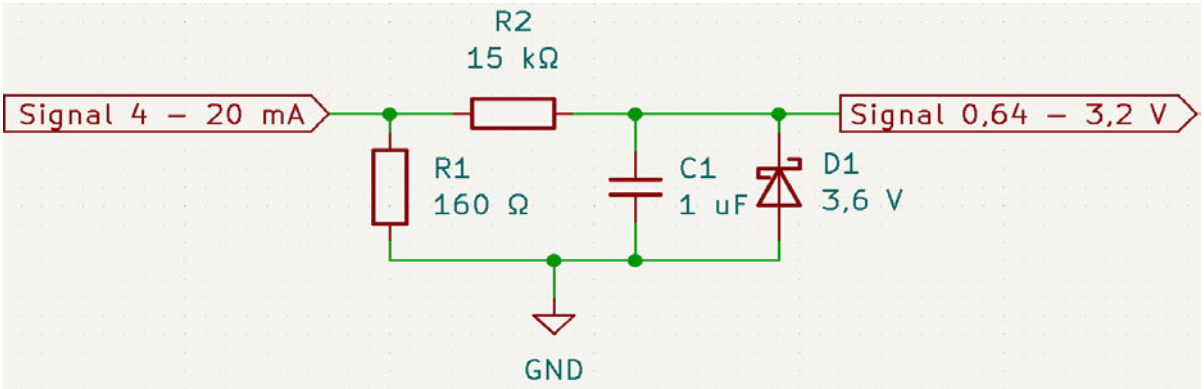


Abbildung 20: Einfache Umwandlung von Strom zum Spannungssignal mit einem RC-Tiefpassfilter, (Erstellt mit KiCad 6.0)

Die Einstrahlungssensoren liefern ein Analogsignal von 4 bis 20 mA, wie in der Abbildung 20 zu sehen ist. Legt man einen 160 Ω Widerstand *R1* parallel von Signal zur Masse, so lässt sich mit der einfachen Formel des ohmschen Gesetzes die Spannung ableiten.

$$U = R * I$$

Mit *U* Spannung in V,

R = Widerstand in Ω,

I = Strom in A.

Bei 4 mA entsteht eine Spannung von 0,64 V und bei 20 mA eine Spannung von 3,2 V. Zum Schutz vor zu hoher Spannung wird auch in dieser Schaltung eine Zener-Diode und ein Widerstand wie oben bereits erläutert verwendet.

Die Grenzfrequenz liegt bei 10,5 Hz.

Die Tabelle 11 zeigt die Verbindung der Sensoren nach der Signalumwandlung an dem Arduino Due.

Tabelle 11: Pinbelegung der Sensoren am Arduino Due

Sensoren		Arduino Due
Alle Sensoren	Masse	GND
Einstrahlungssensor 1	Orange	A0
Einstrahlungssensor 2	Orange	A1
Einstrahlungssensor 3	Orange	A2
Einstrahlungssensor 4	Orange	A3
Stromsensor	Ausgang	A4
Spannungssensor	U_Out	A5

Der gesamte Schaltplan ist im Anhang 1 zu finden.

4.2.3 Zusammenbau

Die im Abschnitt 4.2.2 beschriebenen elektronischen Bauteile sind auf einer Lochplatine aufgelötet und verschaltet. Bei diesem Projekt handelt es sich um einen Prototyp, deshalb wurde auf eine extra dafür erstellte Lötplatine verzichtet. Internetanbieter wie z.B. „PCBWay.com“ bieten nur die Produktion von individuellen Lötplatinen in größeren Abnahmemengen an. In der Abbildung 21 ist der Aufbau im Inneren der Box zusehen. Die Lochplatine und der Arduino Due sind auf einer Kunststoffplatte mit dem RTC-Modul verschraubt und die Kunststoffplatte ist wiederum mit dem Gehäuse verschraubt, um ein Verrutschen beim Transport zu verhindern. Die Anschlüsse für die Sensoren sind links montiert und werden, wie auch die Module, über Steckverbindungen an der Lochplatte verbunden. Dies sorgt dafür, dass alle wichtigen Komponenten wie Module, Spannungsversorgung und Anschlüsse unkompliziert ausgetauscht werden können.

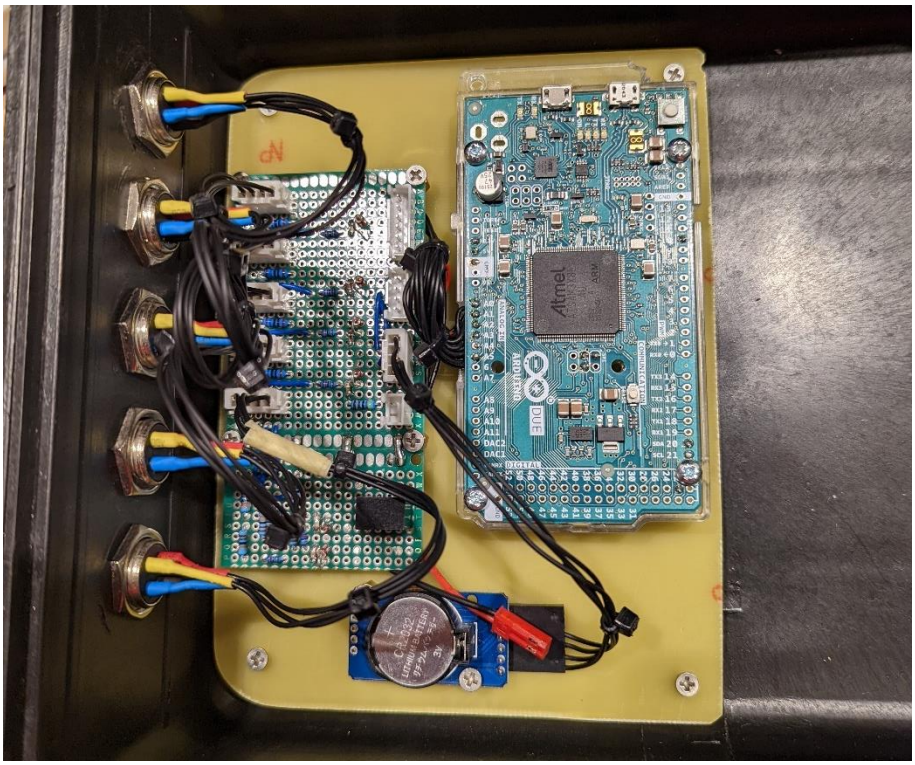


Abbildung 21: Verbaute Lochplatine mit dem Arduino Due und dem RTC-Modul

Das Display und das SD-Modul befinden sich auf der Aluabdeckung, die die Elektronik vor Berührung und einem Kurzschluss schützt. Die Abbildung 22 zeigt die Abdeckung beim fertig zusammengebauten Messgerät. Die Abdeckung besteht aus einer 1,5 mm dicken Aluplatte. Diese wurde in die gewünschte Form mit einer Metallschere geschnitten und zurechtgebogen. Zusätzlich wurden Aussparungen für die Komponenten herausgefräst. Des Weiteren befindet sich ein Tastknopf zur Auswahl des Messintervalls und zum Starten der Logger-Funktion, sowie ein An/Aus-Knopf auf der Abdeckung. Rechts ist das Netzteil und das Netzteilkabel positioniert. Die Sensoranschlüsse, die in der Abbildung 23 zu sehen sind, sind für die richtige Anschlussweise beschriftet und mit Gummikappen, für den Transport vor Schmutz und Feuchtigkeit geschützt.

Abbildung 22: Fertiges Messgerät, offen

Abbildung 23: Fertiges Messgerät, geschlossen

Der Spannungssensor und der Stromsensor befinden sich in einem staub- und wasserdichten, sowie vor Berührung geschütztem Verteilerkasten aus PVC-Kunststoff mit den Maßen 190 x 140 x 70 mm. Abbildung 24 zeigt das innere des Verteilerkastens. Zum Schutz vor Verrutschen sind die Sensoren mit der Box verschraubt. Kabelverschraubungen schützen ebenfalls die nach außen führenden PV-Kabel vor Zug, Schmutz und Wasser. An den Kabelenden werden passende PV-Stecker angebracht, die die einfache serielle Einbindung in einen PV-Strang (z.B. am Eingang eines Strang-Wechselrichters) ermöglichen.

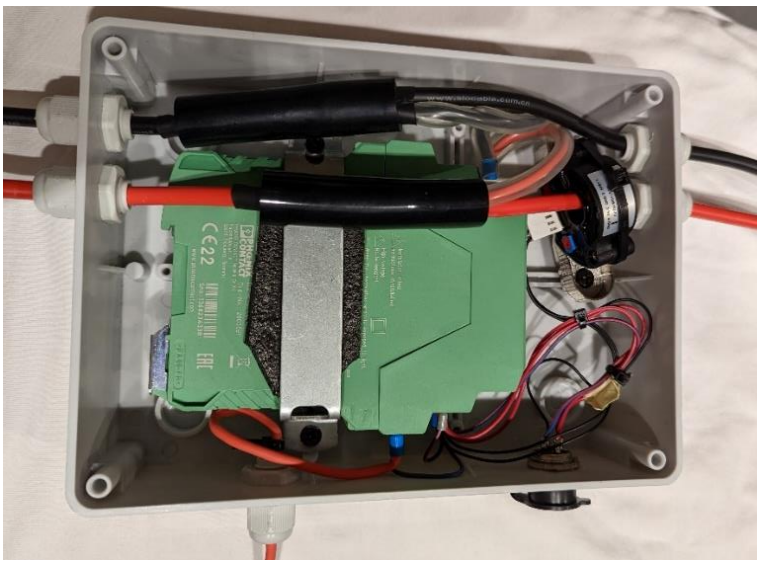


Abbildung 24: Inneres des Verteilerkastens mit einem Strom- und Spannungssensor und den durchgezogenen PV-Kabeln

Der Stromsensor befindet sich links im Verteilerkasten und ist mit diesem am Boden verschraubt. Die Strommessung erfolgt berührungsfrei an dem positiven PV-Kabel. Beim Abgreifen der Spannung an den PV-Kabeln ist besondere Vorsicht geboten. Die PV-Kabel sind mit einer doppelten Isolierung ausgestattet. Durch den 4 mm^2 starken Kabelquerschnitt sind diese somit bis zu einer Spannung von 1500 V DC ausgelegt. Die Isolierung beider Kabel ist punktuell entfernt, um den Anschluss des Spannungssensors zu ermöglichen. Zum Schutz vor Berührung und Lichtbögen sind diese Stellen mit

zwei Schrumpfschläuchen und einem Silikonschlauch geschützt. Der unten im Bild zu sehende rote Schutzleiter PE ist nach dem Handbuch des Spannungssensors mit einem Kabelquerschnitt von 2,5 mm^2 Kabel ausgeführt und muss z.B. an das geerdete PV-Gerüst angeschlossen werden. Diese Schutzerdung ist eine Sicherheitseinrichtung zum Schutz vor elektrischem Schlag. Des Weiteren wird diese zur Messung der Spannung am PV-Strang mit einbezogen.

Der gesamte Schaltplan der Verschaltung der Sensoren im Verteilerkasten ist im Anhang 2 zu finden.

4.3 Messung der einzelnen Sensoren

In diesem Kapitel werden die Sensoren getestet und Messungen möglichst unter Laborbedingungen vollzogen, um eine hohe Genauigkeit der Sensoren zu erzielen. Unter anderem werden noch Messkennlinien aufgezeigt und erläutert.

4.3.1 Messungen der Einstrahlungssensoren

Um den Wolkenzug an einem Strang möglichst genau verfolgen zu können, besteht die Anforderung an die Einstrahlungssensoren, dass die gemessene Einstrahlung möglichst gleich bei allen vier Sensoren ist. Für die Kalibrierung wurde ein sonniger Tag gewählt, mit möglichst hoher und schwankender Einstrahlung. In der Abbildung 25 ist der Aufbau der Sensoren zu sehen.

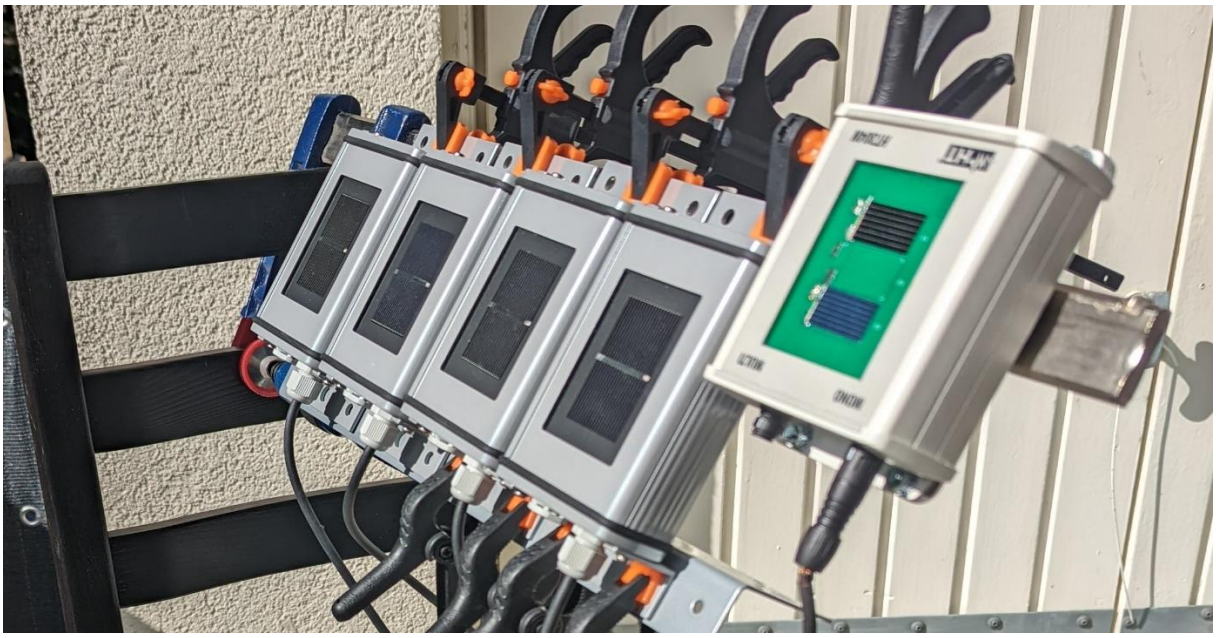


Abbildung 25: Testaufbau der vier Einstrahlungssensoren und dem Referenzsensor auf dem Balkon

Hinsichtlich des Versuchsaufbaus wurde darauf geachtet, dass die Einstrahlungssensoren möglichst bündig und mit demselben Einstrahlungswinkel zur Sonne hin ausgerichtet sind. Zu Referenzzwecken wurde ein geeichter Einstrahlungssensor „HT304N“ und ein Messgerät „SOLAR-02“ der Firma „HT Instruments“ hinzugezogen, um die aufgenommenen Werte vergleichen zu können. Leider muss erwähnt werden, dass der „SOLAR-02“ allein über keine Logger Funktion verfügt. Das heißt, die Werte müssen manuell abgelesen und in die Tabelle eingetragen werden. Diese Vorgehensweise führt zu einer gewissen Abweichung. Für eine höhere Genauigkeit der Messung wurde das Display der beiden

Messgeräte abgefilmt und nachträglich in die Tabelle eingetragen. Hierfür wurde das aufgenommene Videomaterial Frame für Frame durchgespielt und alle Veränderungen auf den Messgeräten in einer Excel Tabelle dokumentiert. Daraus ergibt sich das Liniendiagramm in Abb. 26.

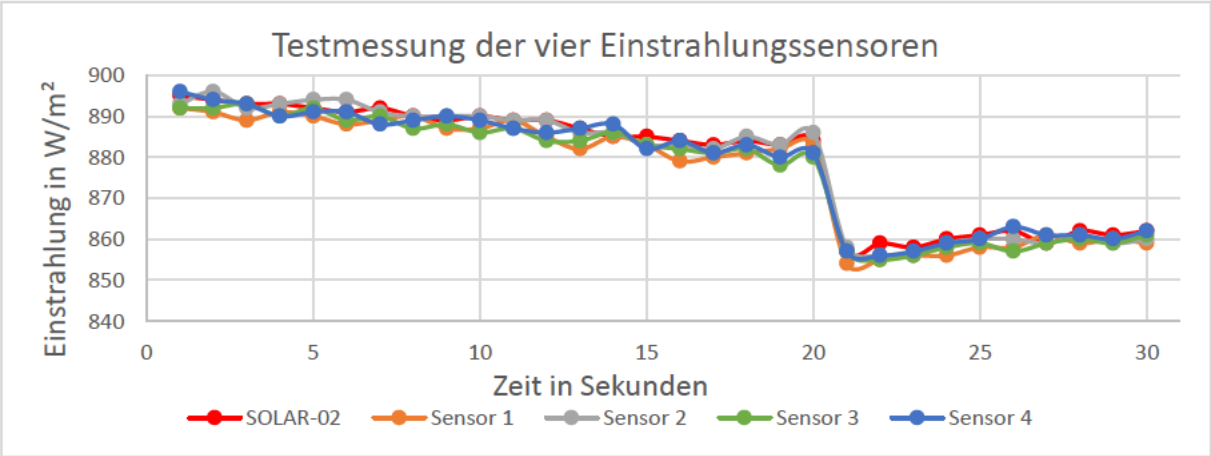


Abbildung 26: Testmessung mit vier Einstrahlungssensoren und dem Referenzsensor (Liniendiagramm)

Dargestellt werden in Abbildung 26, die vier Kennlinien der Sensoren und die Messung mit dem „SOLAR-02". Die Messung erstreckt sich in einem Zeitintervall jeweils von 30 Sekunden. Die vier Einstrahlungssensoren und der „SOLAR-02" messen in einem Intervall von eine Sekunde. Deutlich erkennbar ist, dass die Sensoren und das Referenzmessgerät eine fast identische Messung vornehmen. Diese Übereinstimmung kann nur erreicht werden, wenn in der Programmierung die Messgrenzen der einzelnen Sensoren angepasst werden, bis die Messung identisch mit dem geeichten Messgerät übereinstimmt. Die Kalibrierung der einzelnen Sensoren steht und fällt mit dem verwendeten Referenzmessgerät. Beim Vergleich der einzelnen Sensoren zu dem Referenzsensor ergibt sich eine prozentuale Abweichung von -0,6% bis 0,3%. Betrachtet man die Sensoren untereinander, ergibt sich vom Mittelwert eine prozentuale Abweichung von -0,3% bis 0,4% in einem Einstrahlungsbereich von 854 bis 896 W/m². Bedenkt man, dass die Erfassung der Messwerte optisch erfolgte und die Messung dadurch an Genauigkeit verlieren kann, wäre es optimistischer, die prozentualen Abweichungen zum Referenzgerät und die Abweichung im Mittelwert zueinander mit ±1% zu schätzen.

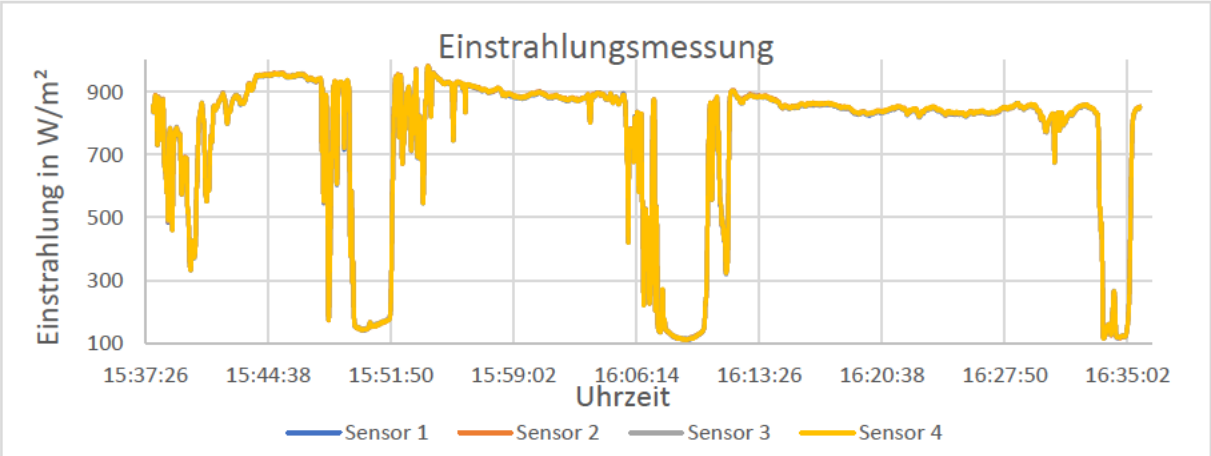


Abbildung 27: Einstrahlungsmessung von einer Stunde mit vier Einstrahlungssensoren (Liniendiagramm)

Nach der vollendeten Kalibrierung der Sensoren zum Referenzsensor wurde eine einstündige Einstrahlungsmessung vorgenommen. Das Messintervall der einzelnen Sensoren beträgt eine Sekunde. In der Abbildung 27 kann man deutlich feststellen, dass alle vier Kennlinien übereinander liegen. Betrachtet man die prozentuale Abweichung im Mittelwert zueinander, ergibt sich ein Wert von ±2,9% in einem Einstrahlungsbereich von 109 bis 983 W/m^2. Aufgrund der Messung im Freien über einen langen Zeitraum ist die Abweichung von ±2,9% nicht repräsentativ. Messfehler können durch Verschattungen in Form von Vogelzug, Wolkenzug oder durch umliegende Bäume entstehen und somit die Abweichung negativ beeinflussen.

4.3.2 Messung des Spannungssensors

Der Spannungssensor SCK-M-U-1500V hat laut dem Handbuch eine Linearität von ±1% in einem Bereich von 100 bis 1500 V DC. Für die Überprüfung ist auch im Handbuch eine Messskala mit den theoretisch messbaren Werten angegeben, wie in der Tabelle 12 zu sehen ist.

Tabelle 12: Gültige Spannungsbereiche des Spannungssensors (verkürzt), (PHOENIX CONTACT GmbH & Co. KG, 2013)

Strangspannung in V	0	100	200	300	400	500	800	1000	1200	1500
Ausgansspannung am Sensor in V	2,00	2,53	3,07	3,60	4,13	4,67	6,27	7,33	8,40	10,0
Reduziert durch den Spannungsteiler in V	0,678	0,845	1,026	1,204	1,381	1,561	2,095	2,451	2,808	3,343

Bei der Testung solch hohen Gleichspannungen benötigt man ein Hochspannungs-Gleichstrom Netzteil. Das zur Verfügung stehende Labornetzteil „Delta Elektronika SM660-AR-11“ zur Überprüfung der Linearität, hat eine Maximalspannung von 660 V DC. Mit dieser Spannung wurde am Arduino Due eine Ausgangsspannung von 1.842 V gemessen. Die Tabelle 12 gibt die Strangspannung in Hunderterschritten als Wert in V an, deshalb wurde der zuvor gemessene Wert mit der Formel:

$$y = \frac{y2 - y1}{x2 - x1} * (x - x1) + y1$$

linear interpoliert. Das Ergebnis von 1.846 V stimmt fast mit dem vorher gemessenen Wert überein. Im Resultat wurde mit dem Spannungssensor am Arduino eine Spannung von 658,4 V gemessen und ist damit 1.6 V niedriger als die Ausgangsspannung des Labornetzteils. Es entsteht eine prozentuale Abweichung von -0,2%. Geringere Spannungen unter 660 V treffen zunehmend die theoretischen Werte aus dem Handbuch. Die Vermutung liegt nahe, dass bei Spannungen über 660 V die Abweichung höher ausfallen könnte. Um dies zu überprüfen und die Messung zu präzisieren, wird ein 1500 V DC Netzteil benötigt. Dieses stand in der Entwicklungszeit des eigenen Messgerätes nicht zur Verfügung.

Die Abbildung 28 zeigt die tatsächlich gemessenen Werte, die wie auch im Handbuch beschrieben, ein lineares Verhalten aufweisen.

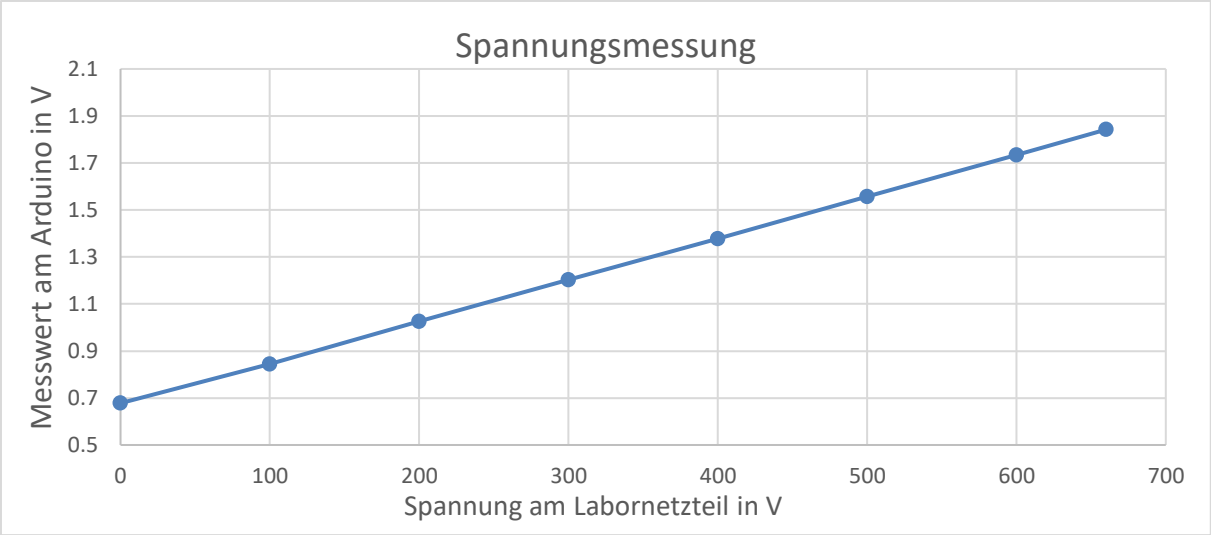


Abbildung 28: Gleichspannungsmessung von 0 bis 660 V (Liniendiagramm)

4.3.3 Messung des Stromsensors

Der Stromsensor CYHCT-EKCV-U/B30-34 wurde wie auch der Spannungssensor auf Linearität getestet. Den Strom lieferte ein Labornetzteil „PeakTech 6155“. Als Last wurden drei parallelgeschaltete 100 W Halogenlampen verwendet. Das Labornetzteil wurde dann um jeweils ein Ampere erhöht und Werte der Messung aufgezeichnet. In der Abbildung 29 sind die Werte als Kennlinie erfasst. Ein lineares Verhalten des Stromsensors, wie auch im Datenblatt beschrieben, wird dargelegt.

Die Kalibrierung erfolgte bis 20 A. Aufgezeigt wird ein ideales Ergebnis dadurch, dass die eingegangenen gemessenen Stromwerte des Arduinos mit den Ausgangswerten des Netzteils übereinstimmen.

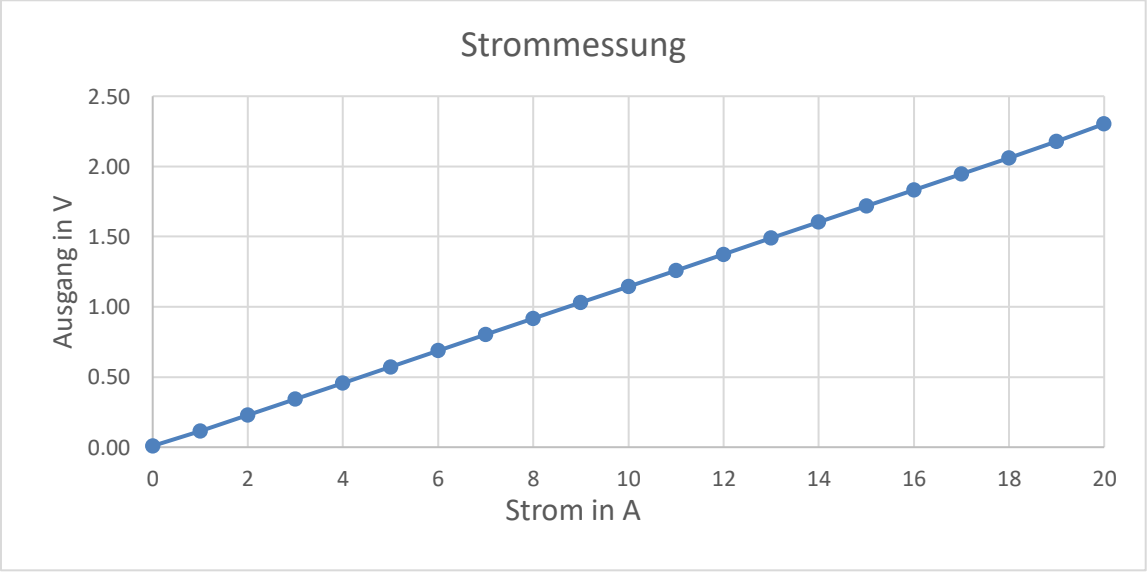


Abbildung 29: Strommessung von 0 bis 20 A (Liniendiagramm)

4.4 Programmierung

Die Programmierstruktur und die Funktionsweise der einzelnen Bibliotheken und Funktionen wurden in dem Kapitel Grundlagen ausreichen behandelt. In diesem Abschnitt wird daher auf die Kalibrierfunktion als Programmcode genauer eingegangen und zum besseren Verständnis das Fließdiagramm des gesamten Codes dargestellt.

```
void showMeasuredData(){
 rawADCValue = 0;    // Abschnitt 1
 rawADCValue1 = 0;
 rawADCValue2 = 0;
 rawADCValue3 = 0;
 rawADCValue4 = 0;
 rawADCValue5 = 0;

 for(int i = 0; i < Interval; i++){    // Abschnitt 2
  rawADCValue += analogRead(sensorPin0);
  rawADCValue1 += analogRead(sensorPin1);
  rawADCValue2 += analogRead(sensorPin2);
  rawADCValue3 += analogRead(sensorPin3);
  rawADCValue4 += analogRead(sensorPin4);
  rawADCValue5 += analogRead(sensorPin5);
  delay(5);
 }    // Abschnitt 3

 ADCVoltage = (float)(rawADCValue / Interval) * ((3260 - offset0)/ 4095.0);
 ADCVoltage1 = (float)(rawADCValue1 / Interval) * ((3250 – offset1)/ 4095.0);
 ADCVoltage2 = (float)(rawADCValue2 / Interval) * ((3250 – offset2)/ 4095.0);
 ADCVoltage3 = (float)(rawADCValue3 / Interval) * ((3250 – offset3)/ 4095.0);
 ADCVoltage4 = (float)(rawADCValue4 / Interval) * ((3250 - offset4)/ 4095.0);
 ADCVoltage5 = (float)(rawADCValue5 / Interval) * ((3265 - offset5)/ 4095.0);

 Current = mapFloat(ADCVoltage, 636, 3162, 4, 20);     // Abschnitt 4
 Current1 = mapFloat(ADCVoltage1, 636, 3154, 4, 20);
 Current2 = mapFloat(ADCVoltage2, 636, 3150, 4, 20);
 Current3 = mapFloat(ADCVoltage3, 636, 3134, 4, 20);

 Irradiation = mapFloat(Current, 4, 20, 0, 1200);    // Abschnitt 5
 Irradiation1 = mapFloat(Current1, 4, 20, 0, 1200);
 Irradiation2 = mapFloat(Current2, 4, 20, 0, 1200);
 Irradiation3 = mapFloat(Current3, 4, 20, 0, 1200);

 valueDC = mapFloat(ADCVoltage4, 678, 3300, 2, 10);     // Abschnitt 6
 MVoltage = (375.00/2.00)* valueDC - 375;

 MCurrent = mapFloat(ADCVoltage5, 9, 2300, 0, 20);     // Abschnitt 7
}
```

Codeausschnitt 5: Kalibrierfunktion

Der Codeausschnitt 5 stellt die Kalibrierfunktion mit dem Namen „*showMeasuredData*“ dar. Der Programmiercode für die Ausgabe der Werte auf dem LC-Display wurde weggelassen, da diese ausführlich im Kapitel 3.3 (Software) bereits erklärt wurden. Für die einfachere Erklärung ist der Codeausschnitt 5 in sieben Abschnitte unterteilt.

Abschnitt 1: Die Eingangswerte aller analog Pins werden beim Wiederholen der Funktion auf den Wert 0 resettet. Das ist für die Vorbereitung für den Abschnitt 2 wichtig, denn dort wird ein Mittelwert gebildet.

Abschnitt 2: Mittelwertbildung aller Eingangswerte. Das Intervall ist in diesem Fall auf 10 gesetzt. Dies bedeutet, dass die For-Schleife aus zehn gemessenen Werten ein Mittelwert bildet. Die Zeit für die Mittelwertbildung beträgt 50 Millisekunden. Mit einem höheren Intervall von mehr als 10 würden die Messdaten zwar besser geglättet, dies führe unweigerlich zu einer höheren Verzögerung bei der Aufnahme der Messsignale.

Abschnitt 3: Hier werden die gemittelten Rohwerte aller Sensoren von 0 bis 4095 Teilen in eine Spannung von 0 bis 3,3 V umgerechnet. Bei dem ersten „*ADCVoltage*“ als Beispiel steht die Zahl 3260 für die maximale Spannung der analogen Pin-Versorgung in mV. Diese muss mit einem Multimeter gemessen werden. Mit dem „*offset*“ kann dieser Messwert später nach oben oder nach unten nachkalibriert werden, ohne die Funktion verändern zu müssen.

Abschnitt 4: Die Spannung, die in dem vorherigen Abschnitt errechnet wurde, wird nun in einen Stromwert von 4 bis 20 mA umgerechnet. Dies gilt nur für die Einstrahlungssensoren. In dem Kapitelabschnitt 4.2.2 ist die einfache Stromspannungsumwandlung erklärt und beinhaltet einen 160 Ω Widerstand, der aus dem Analogstrom eine Spannung umwandelt. In die Rechnung fließt dieser Widerstand mit ein. Daraus ergibt sich die einfache Rechnung von 160 Ω * 4 mA, eine untere Spannung von 640 mV, sowie bei 160 Ω * 20 mA eine obere Spannung von 3200 mA. Aufgrund des bereits vorhanden Toleranzwertes des Widerstandes, muss jeder Widerstand vorher gemessen werden. Dementsprechend ergibt sich bei einem realen Widerstandsmesswert von 159 Ω eine untere Spannung von 636 mV und eine obere Spannung von 3180 mV. Die obere Grenze variiert stark von anderen Widerständen, die in der Signalleitung vorhanden sind. Daher wurde die obere Spannung so lange runtergesetzt, bis der errechnete Einstrahlungswert, den vom kalibrierten Messgerät trifft.

Abschnitt 5: Der ermittelte Strom 4 bis 20 mA aus dem Abschnitt 4 wird nun linear in eine Einstrahlung von 0 bis 1200 W/m^2 umgesetzt. Bei der Verwendung von Einstrahlungssensoren mit einem Messebereich von 0 bis 1500 W/m^2 kann hier die obere Grenze von 1200 auf 1500 geändert werden.

Abschnitt 6: Hier handelt es sich um die Messwerte des Spannungssensors. Die umgerechnete Spannung aus Abschnitt 3 wird zunächst linear von 2 bis 10 V umgewandelt. Die Formel zur Umrechnung des analogen Signals des Sensors ist aus dem Sensorhandbuch entnommen und im digitalen Anhang PHOENIX CONTACT GmbH & Co. KG, 2013 in der Tabelle 4.4 „Berechnung Systemspannung“ zu finden.

Abschnitt 7: Dargestellt werden in diesem Abschnitt die Messwerte des Stromsensors. Die untere und obere Spannungsgrenze wurde bei 0 A und 20 A mit einem Multimeter am Arduino gemessen. Theoretisch kann der Stromsensor bis 30 A messen. Für die Projektzwecke ist dies aber irrelevant, da einzelne PV-Stränge so hohe Ströme nie erreichen. Diese liegen weit unter 20 A.

Die Abbildung 30 zeigt das Flussdiagramm des gesamten Programmcodes ohne konkrete Zahlen, für das einfachere Verständnis. Der gesamte Programmcode ist im digitalen Anhang 10 zu finden.

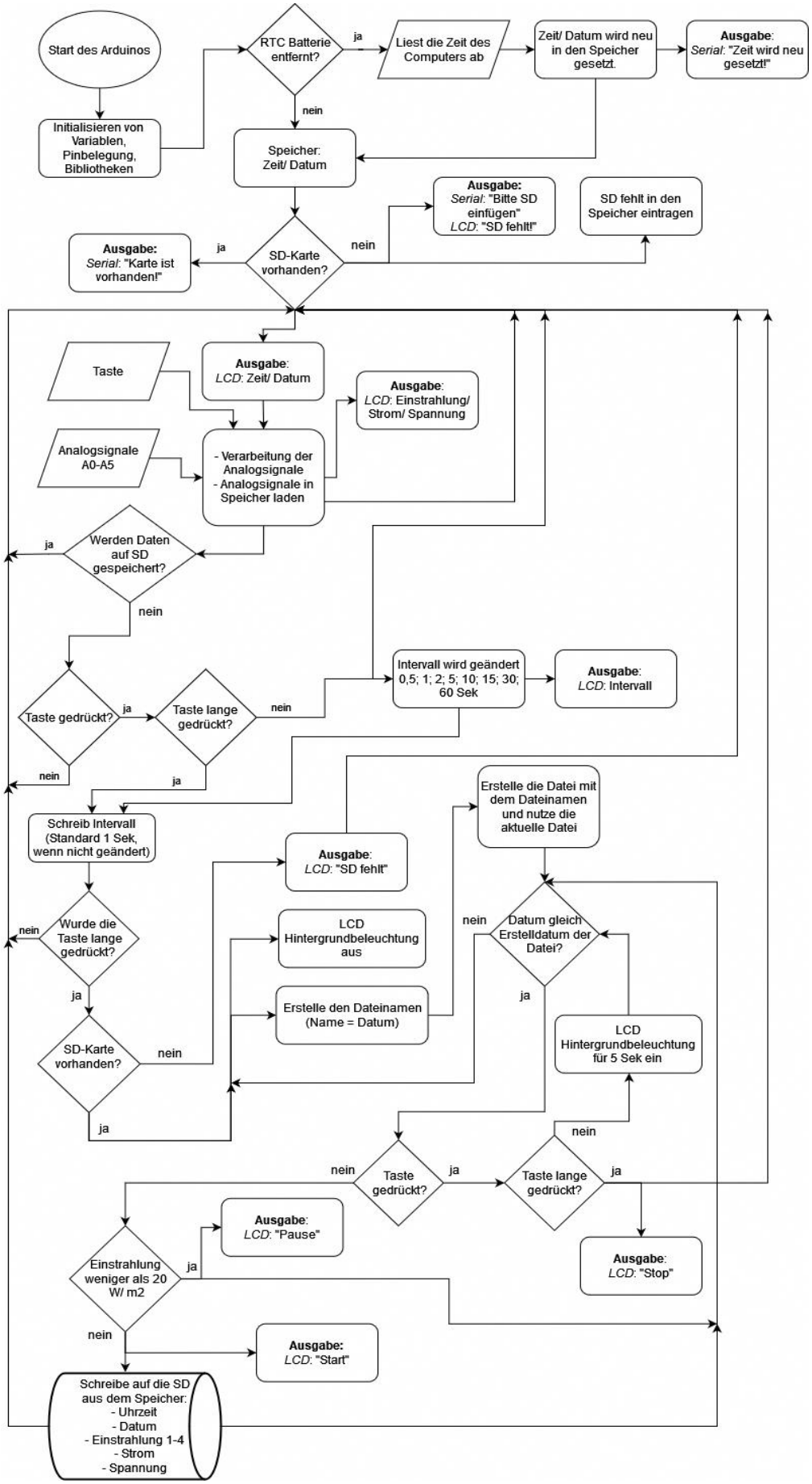
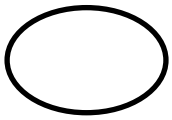
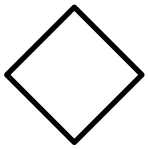
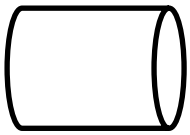


Abbildung 30: Flussdiagramm des fertigen Programmcodes (Erstellt mit diagrams.net)

Für die bessere Verständlichkeit des Flussdiagramms sind als Übersicht die Formen und deren Funktionen in der Tabelle 13 aufgelistet, als auch beschrieben.

Tabelle 13: Übersicht der Formen die im Flussdiagramm (Abbildung 30) verwendet wurden

	Start-/Endsymbol	Das Oval stellt den Beginn oder das Ende eines Prozesses dar
	Prozess-/Tätigkeitssymbol	Das Rechteck stellt eine Aktion, Tätigkeit oder Funktion dar
	Entscheidungssymbol	Die Raute kennzeichnet eine Abfrage die z.B. ja oder nein sein könnte
	Datensymbol	Das Parallelogramm stellt Datensätze dar wie z.B. Input oder Output
	Containersymbol	Ein umgedrehtes Containersymbol stellt eine Festplatte, in diesem Fall eine SD-Karte dar
	Verbindungspfeil	Der Verbindungspfeil verbindet alle Flusselemente miteinander und verdeutlicht die zeitliche Reihenfolge der Prozessschritte

5 Testlauf des Messgeräts an einer PV-Anlage

Der Testlauf erfolge auf einer fest ausgerichteten PV-Anlage in der Gemeinde Tating in Schleswig-Holstein. Die Anlage ist ca. 120 km südöstlich von Hamburg entfernt. Ein PV-Strang beinhaltet 22 Module von dem Hersteller „Canadian Solar“ mit einer Gesamtlänge von ca. 23 m und einem Neigungswinkel von 15°.

Die vier Einstrahlungssensoren wurden wie in der Abbildung 2 (Kapitel 2) mit einem Abstand von ca. 5,7 m an dem oberen Strang voneinander angebracht. Die Abbildung 31 zeigt die befestigten Sensoren vom Boden aus.

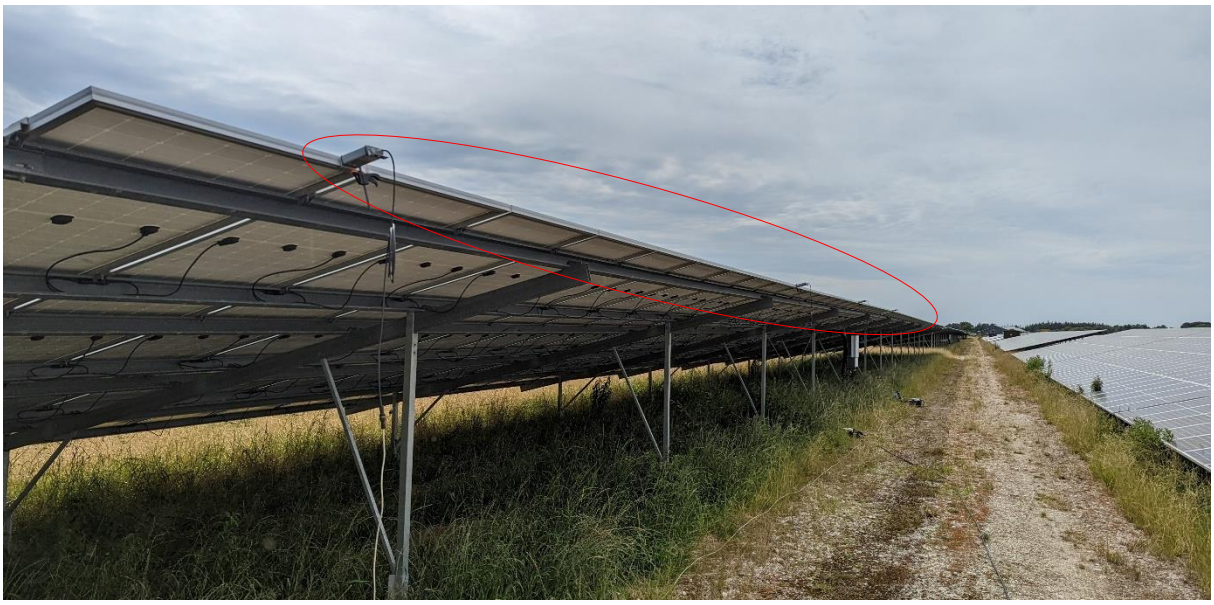


Abbildung 31: Vier Einstrahlungssensoren befestigt an dem oberen Modul Strang, (rot markiert)

Die Einstrahlungssensoren wurden provisorisch mit einer Einhandzwinge am Modulrahmen befestigt. Es ist drauf zu achten, dass die Sensoren so bündig wie möglich platziert werden, denn schon kleine fehlerhafte Neigungen der Sensoren und die stark nach Osten oder Westen stehende Sonnenrichtung können stärkere abweichende Messwerte verursachen.

Der Strom- und Spannungssensor im Verteilerkasten wurden am Strangende, am Wechselrichter des Herstellers „DELTA ELECTRONICS“, wie in der Abbildung 32 zu sehen ist, zwischengeschaltet. Wichtig ist, dass der Wechselrichter beim Zwischenschalten der Sensoren vollständig ausgeschaltet werden muss, um die vorhandene Last zu nehmen. Zusätzlich erfolgt zur Sicherheit eine Kontrolle mit einer Stromzange, die die Strangausgänge auf Last prüft. Diese Vorgehensweise verhindert, dass beim Abstöpseln der Strangeingänge des Wechselrichters ein Lichtbogen und somit ein elektrischer Schlag verursacht werden kann. Zum weiteren Schutz von Kurzschlüssen im System und zur genaueren Messung muss der Schutzleiter PE des Spannungssensors an die Erdung (z.B. PV-Gerüst) geklemmt werden. Das erfolgt mit dem an der Seite positioniertem roten Kabel mithilfe einer Krokodilklemme.

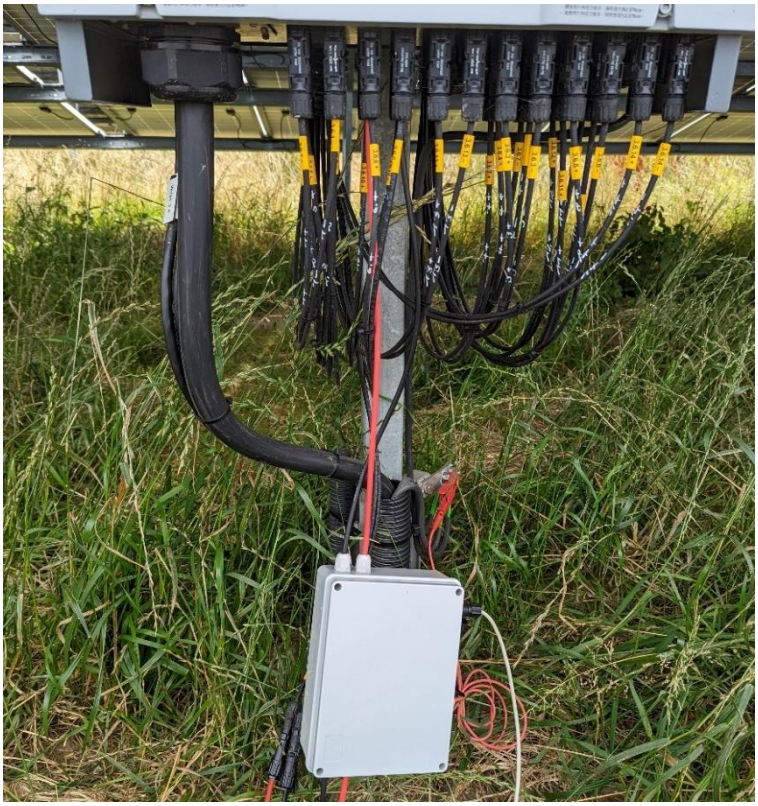


Abbildung 32: Strom- und Spannungssensor im Verteilerkasten am Wechselrichter zwischengeschaltet

Die Abbildung 33 zeigt das LC-Display des Messgeräts. Darauf ist die Uhrzeit und das Datum im oberen Feld zusehen, die Einstrahlung der vier Sensoren unten links und die Spannung und der Strom unten rechts. Über dem Spannungswert ist das Informationsfeld, welches entweder das Messintervall oder die Aufzeichnung Start/Stopp anzeigt.

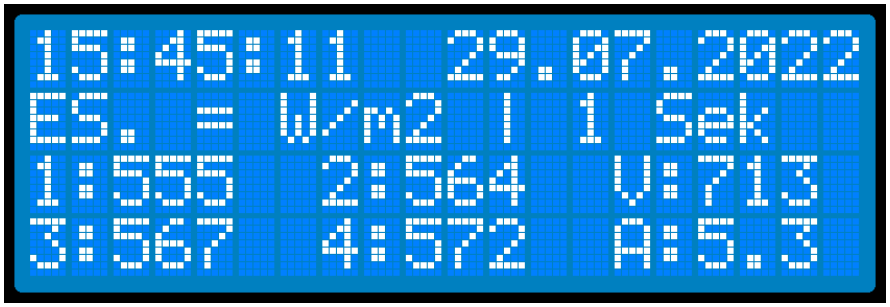


Abbildung 33: Darstellung des LCDs am Messgerät (erstellt mit LCD Screenshot Generator, avtanski.net)

Am Messtag war der Himmel homogen bewölkt und somit konnte keine Messung mit heterogenem Wolkenzug gemacht werden. Um trotz allem die Veränderung der Sensoren und die Leistung des Stranges prüfen zu können, wurde mit einer Decke von 2 x 2 m die ungleichmäßige Verschattung so gut wie möglich simuliert. Die Decke wurde von Anfang bis zum Enden über den Strang hängend langsam rübergezogen, wie in der Abbildung 34 zu sehen ist. Eine Simulation von weichen Schatten ist leider nicht gelungen aufgrund der Gegebenheiten der Anlage. Hier war es nicht möglich z.B. die Sensoren am untersten Strang zu platzieren und von einer bestimmten Entfernung die Decke über den Strang zu ziehen.

Abbildung 34: Platzierung der Decke am Modul Strang zur Wolken Simulierung (helfende Person: Martin Dassau)

In der Abbildung 35 sind die vier Einstrahlungssensoren in Orange, Gelb, Grau und Grün und die Leistung resultierend aus Strom und Spannung des Stranges in Blau abgebildet. Die Kennlinie der Leistung wurde so gelegt, dass man eine Veränderung leichter optisch feststellen kann. Erkennbar ist, dass die Decke beim Gehen die Sensoren nacheinander verschattet und dass deren gemessene Einstrahlung und die Leistung des Stranges dementsprechend einbricht. Aufgrund der Witterung am Testtag mit leichtem Wind hat die Decke ständig geflattert, dadurch sind schwankende Einstrahlungs- und Leistungswerte aufgezeichnet worden. Die Leistung normalisiert sich zum Ende nach der Entfernung der Decke wieder.

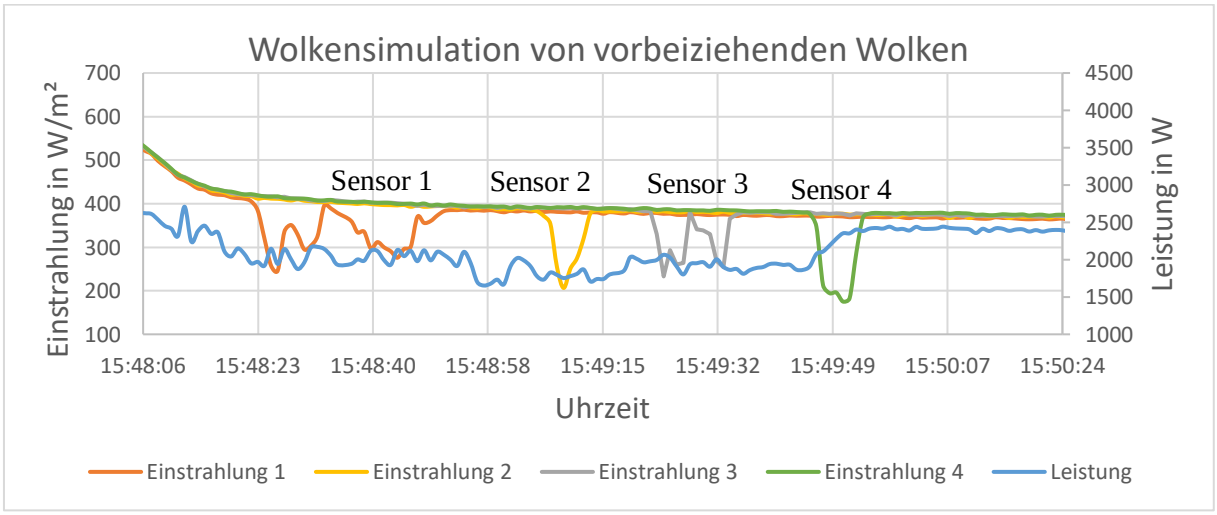


Abbildung 35: Einstrahlungs- und Leistungskurve bei der Simulation der vorbeiziehenden Wolken (Liniendiagramm)

Der Wolkenzug wurde von Osten nach Westen simuliert. Bei einem Wolkenzug von Norden nach Süden oder umgekehrt würden theoretisch alle vier Sensoren gleichzeitig verschattet. Dies führe dazu, dass die Leistung deutlich höher einbrechen würde, als in der Abbildung 35 zu sehen ist. Somit kann auch, je nachdem welche Kurve als Erstes einbricht (Leistung oder Einstrahlung), zusätzlich die Richtung des Wolkenzuges bestimmt werden.

Anschließend wurde eine Messung von 30 Minuten durchgeführt, mit einem Messintervall von einer Sekunde. Vergleicht man die Abbildung 27 (Kapitel 4.3.1) mit der Abbildung 36, erkennt man sofort, dass die Einstrahlungsmessung der einzelnen Sensoren nicht übereinanderliegen, sondern versetzt darunter. Die Ursache dafür ist, dass die Sensoren mit größerem Abstand zueinander am Strang angebracht wurden und nicht direkt nebeneinander wie im Versuchsaufbau in Abbildung 25 (Kapitel 4.3.1). Zugleich spielt die Sonnenrichtung, die um die Uhrzeit bei ca. 55° West lag, mit der leichten Ost-Westneigung des PV-Tisches eine wichtige Rolle für diese Versetzung der Kennlinien.

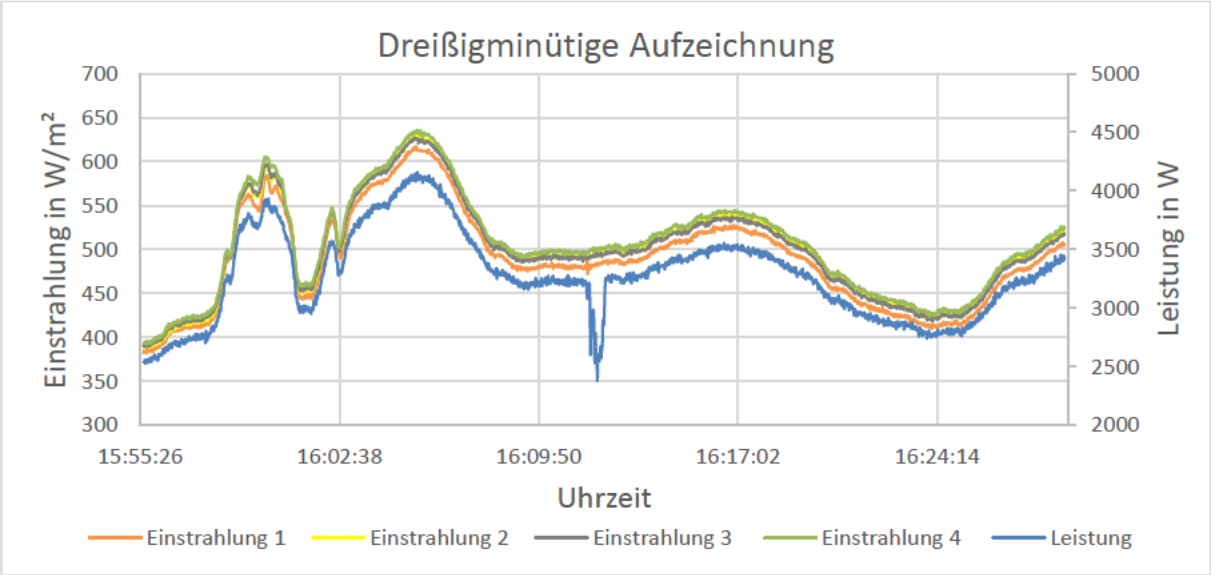


Abbildung 36: Dreißigminütige Aufzeichnung der Daten (Liniendiagramm)

Um ca. 16.12 Uhr wurde die Decke auf paar Module für ca. 2 Minuten gelegt, um den Leistungsverlust kurzzeitig zu simulieren.

Anschließend kann man sagen, dass das Projekt mit dem Testlauf des Messgeräts an einer PV-Anlage die zu Anfang gestellte Erwartung, wie die gleiche Messung aller Einstrahlungssensoren untereinander, eine Veränderung der einzelnen Einstrahlungsmessungen durch einen heterogenen Wolkenzug (hier simuliert mit einer Decke) und den dadurch resultierenden Leistungsabfall der Messung an dem PV-Strang erfüllt hat. Für ein Resultat über die heterogene Verschattung müssten Langzeitdaten mit dem Messgerät erfasst und anschließend analysiert werden. Die Langzeitaufnahme und Auswertung der Daten sollen in Zukunft in einer separaten Abschlussarbeit erfolgen.

6 Zusammenfassung und Ausblick

Zusammenfassung

Diese Arbeit beschreibt den Aufbau eines Messgeräts, der an einem PV-Strang angebracht wurde, um den heterogenen Wolkenzug zu analysieren.

Zunächst wurde die Plattform „Arduino“, die in diesem Projekt verwendet wird, erläutert und auf das Arduino Due Board genauer eingegangen. Dadurch, dass das Board allein keine Messungen vollziehen kann, wurden die zusätzlichen Module und Sensoren, die in diesem Projekt verwendet wurden, erläutert und die Funktionsweise und gegeben falls Fehler, die bei der Benutzung entstehen können, aufgezeigt. Ebenfalls wurde auf die Software weiter eingegangen und die Bedienung der Arduino-IDE erklärt. Die Kommunikation der Module und Sensoren wurden anhand von Bibliotheken und Funktionen aufgezeigt. Außerdem die Codestruktur näher geschildert.

Der nächste Schritt beschreibt die Signalverarbeitung und die damit zusätzlich erklärten Maßnahmen, die zur Senkung der Signalspannung durch einen Spannungsteiler mit den hierfür verwendeten Widerständen und Dioden. Der verwendete RC-Tiefpassfilter der 1. Ordnung wird ebenfalls als Maßnahme beleuchtet, der zusätzlich zur Rauschunterdrückung dient. Zusätzlich wurden die verwendeten Signalkabel und das SELV Netzteil zur Spannungsversorgung konkretisiert.

Der vorletzte Schritt befasst sich mit dem Zusammenbau des Messgeräts aus den zuvor beschriebenen Modulen und Sensoren. Dabei ist zu erwähnen, dass die Anbindung des Spannungssensors an den PV-Kabeln einen besonderen Schutz erfordert, wie die dreifache Isolierung zum Schutz vor Berührung und Lichtbögen.

Im letzten Kapitel wurden die Sensoren möglichst unter Laborbedingungen gemessen und die Programmierung zur Umrechnung der Messwerte aufgezeigt. Die Kalibrierung der Einstrahlungssensoren ergab eine Abweichung von ±1% zu dem verwendeten Referenzmessgerät und zugleich im Mittelwert untereinander. Der Spannungssensor hatte bei der Messung mit einem Hochspannungsnetzteil eine prozentuale Abweichung von -0,2% bei einer Spannung von 660 V DC. Mit dem Stromsensor wurden ideale Messungen vollzogen, die dem Stromnetzteil entsprachen. Abschließend sind dies bereits sehr zufriedenstellende Werte mit kleineren Abweichungen, jedoch könnten diese nochmals optimiert werden, durch ein höheres Hochspannungsnetzteil von 1500 V DC.

Zugleich wurde das Messgerät auf der PV-Anlage in Tating in Schleswig-Holstein aufgebaut und ein heterogener Wolkenzug simuliert, da zum Testzeitpunkt keine heterogenen Wolken vorhanden waren. Die so aufgenommenen Messdaten wurden dokumentiert und die Funktion und der Erfolg des Projekts nachgewiesen.

Ausblick

Die Arbeit stellt den Versuch dar, ein möglichst funktionierendes Messgerät zur Analyse des heterogenen Wolkenzugs und damit den resultierenden Leistungsverlust an einem PV-Strang zu entwickeln. Zur Absicherung der Funktionsweise des Geräts müssen möglichst viele Messungen auf verschieden Photovoltaikanlagen gemacht werden, um die Langlebigkeit und Messstabilität zu prüfen. Nur so lässt sich klären, ob das eigen entwickelte Messgerät reale Werte über einen längeren Zeitraum verlässlich aufzeichnen kann. Den Wunsch, Einstrahlungsmessungen verschiedener Sensoren nur dann zeitlich hochaufgelöst zu erfassen, wenn es zu unterschiedlichen und schwankenden Einzelmessungen kommt, wurde verworfen. Die Zeitachse würde mehrere verschiedenen Auflösungen aufweisen, was die Auswertung der Daten erschwert. Dank der großen Speicherkarte lassen sich von Anfang an die Daten hochaufgelöst über einem längeren Zeitraum erfassen. Hinsichtlich kann auch eine 32 GB große Speicherkarte verwendet werden.

Zusätzlich dazu sollte der Spannungssensor mit einem Hochspannungsnetzteil bis 1500 V DC getestet und nachkalibriert werden, für eine noch genauere Spannungsmessung. In Zukunft kann das Messgerät mit einem Akku und mit einem GSM (Global System for Mobile Communications) -Modul ausgestattet werden, um die Daten per Internet abgreifen und bearbeiten zu können. Eine weitere Überlegung wäre es, die Signalkabel durch Funkmodule auszutauschen und das Analogsignal zu digitalisieren, um das Gewicht der Kabel beim Transport und mögliche Analogsignalstörungen zu minimieren.

Die gewonnenen Langzeitdaten müssen ebenfalls mit anderen PV-Anlagen verglichen und näher analysiert werden, um Rückschlüsse auf die heterogene Verschattung treffen und damit den besagten Mismatch in weiteren Ertragsgutachten einfließen lassen zu können. Diesbezüglich ist eine weitere Abschlussarbeit geplant, die sich mit der Datenauswertung beschäftigen soll.

Literaturverzeichnis

Arduino (2022): What is Arduino? | Arduino, <https://www.arduino.cc/en/Guide/Introduction> [Zugriff 2022-08-01]

Arduino Documentation (2022): Arduino Documentation, <https://docs.arduino.cc/hardware/due> [Zugriff 2022-08-01]

Arduino Input and Output (2022): Arduino Due, <https://store.arduino.cc/products/arduino-due> [Zugriff 2022-08-01]

arduino.cc (Hrsg.) (2022): analogReadResolution - Arduino-Referenz, <https://www.arduino.cc/reference/de/language/functions/zero-due-mkr-family/analogreadresolution/> [Zugriff 2022-08-01]

Arduino-Referenz (2022): analogRead - Arduino-Referenz, <https://www.arduino.cc/reference/de/language/functions/analog-io/analogread/> [Zugriff 2022-08-01]

Bernstein, Herbert (2020): Mikrocontroller: Grundlagen der Hard- und Software der Mikrocontroller ATtiny2313, ATtiny26 und ATmega32, 2. Aufl., Wiesbaden: Springer Fachmedien Wiesbaden; Springer Vieweg, 2020

Dr.-Ing. habil. Jigou Liu (2016): aufklappbarer AC Hall-Effekt Stromsensor CYHCS-KD: Verfügbar im digitalen Anhang (2016)

Hering, Ekbert/Schönfelder, Gert (Hrsg.) (2018): Sensoren in Wissenschaft und Technik: Funktionsweise und Einsatzgebiete, 2. Aufl., Wiesbaden: Springer Vieweg, 2018

Hitachi, Ltd. (1998): HD44780U (LCD-II), (Dot Matrix Liquid Crystal Display Controller/Driver): Verfügbar im digitalen Anhang (1998)

Ingenieurbüro Mencke & Tegtmeyer GmbH (2012): Si-420TC-T-K Silizium-Solarstrahlungssensor: Verfügbar im digitalen Anhang (2012)

Maxim Integrated Products, Inc. (2015): DS3231 Datenblatt: Verfügbar im digitalen Anhang (2015)

Mertens, Konrad (2020): Photovoltaik: Lehrbuch zu Grundlagen, Technologie und Praxis, 5., aktualisierte Auflage, Cincinnati: Hanser Publications, 2020

Nahrstedt, Harald (2009): C++ für Ingenieure: Vieweg Verlag, 1 Auflage, 2009

PHOENIX CONTACT GmbH & Co. KG (2013): Anwenderhandbuch UM DE Solarcheck Familie: Ausschnitt verfügbar im digitalen Anhang (2013), S. 25–30

61558-2-6 (2010): Sicherheit von Transformatoren, Drosseln, Netzgeräten und dergleichen für Versorgungsspannungen bis 1 100 V

Stiny, Leonhard (2015): Aktive elektronische Bauelemente: Aufbau, Struktur, Wirkungsweise, Eigenschaften und praktischer Einsatz diskreter und integrierter Halbleiter-Bauteile, 2. Aufl., Wiesbaden: Springer Fachmedien Wiesbaden, 2015

Texas Instruments/Incorporated (2022): SN74LVC125A Quadruple Bus Buffer Gate With 3-State Outputs datasheet (Rev. Q): Verfügbar im digitalen Anhang (2022)

Wolff, Marcus (2016): Sensor-Technologien, Band 1, Berlin/Boston: De Gruyter Oldenbourg, 2016

Anhang

1. Schaltplan „Power Ratio Logger“ (1 Seite)
2. Schaltplan Verteilerkasten (1 Seite)

Digitaler Anhang

Zu finden auf der mitgegeben SD-Karte.

1. Flussdiagramm „Power Ratio Logger“ (1 Seite)
2. Schaltplan „Power Ratio Logger“ (1 Seite)
3. Schaltplan Verteilerkasten (1 Seite)
4. DS3231 Datenblatt (20 Seiten)
5. Hall-Effekt Stromsensor CYHCT-EKCV Datenblatt (2 Seiten)
6. LCD HD44780 Datenblatt (60 Seiten)
7. sn74lvc125a Micro SD-Datenblatt (29 Seiten)
8. SCK-M-U-1500V_Spannungsmessung Datenblatt (6 Seiten)
9. Si-420TC-T-K Silizium-Solarstrahlungssensor Datenblatt (2 Seiten)
10. Programmcode „Power Ratio Logger“ (.txt/ .ino) (12 Seiten)
11. Bibliotheken in der Datei: Libraries.rar