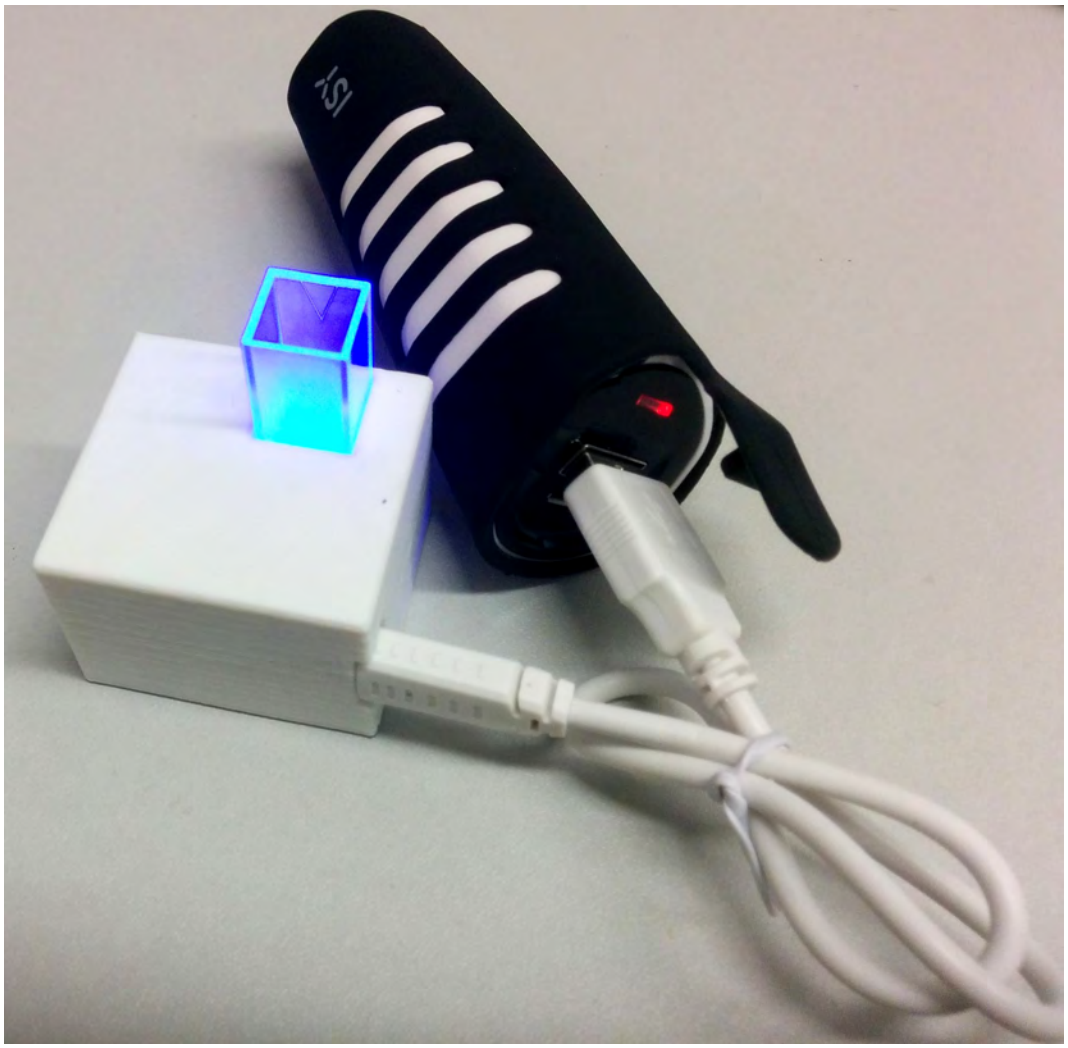
Konzeption und Aufbau eines mobilen Fotometers mit WLAN/ WIFI – Schnittstelle



HAW Hamburg, Labor für Bioprozessautomatisierung

Dipl.-Ing. Ulrich Scheffler

30.10.2018

Ein Projekt von:

Dipl.-Ing. Ulrich Scheffler
HAW Hamburg, Labor für Bioprozessautomatisierung
Ulmenliet 20
21033 Hamburg
email: ulrich.scheffler@haw-hamburg.de

in Kooperation mit dem Schullabor mobile Analytik der HAW Hamburg
email: olaf.elsholz@haw-hamburg.de

Weblinks zu diesem Projekt:

Schullabor mobile Analytik:

Seite mit dieser Anleitung:

Sollte ein Nachbau oder eine Variation dieses Projektes erfolgen, freuen wir uns über Rückmeldungen und Hinweise.

Inhaltsverzeichnis

1 Vorwort

Die MINT[1]-Wissenschaften gelten als Kernwissenschaften für den Standort Deutschland da sie zu den innovativsten und wirtschaftlich wichtigsten Bereichen gehören. Ungünstiger Weise finden sich immer weniger junge Menschen, die sich für diese Themen interessieren lassen – Stichwort: Fachkräftemangel. Hier setzt MINT-Förderung an. Eine Idee dazu: Wenn es gelingt, Jugendliche schon früh in ihrer Schullaufbahn in Kontakt zu bringen mit Fragestellungen, die ihr Interesse an MINT Themen weckt, kann es gelingen, mehr Jugendliche zur Aufnahme einer Ausbildung oder eines Studiums im MINT Fächerkanon zu bewegen und somit dem drohenden Fachkräftemangel entgegenzuwirken.

Hier setzt das im Folgenden vorgestellte Projekt an. Der Selbstbau des Fotometers verknüpft Themen aus den Bereichen.

- Physik — Licht, Optik, LED
- Chemie — Konzentration, Farbreaktionen, (Bouguer)-Lambert-Beer'sches Gesetz
- Elektrotechnik — Strom, Spannung, Bauelemente, Schaltplan, Löten
- Informatik — Programmieren, WIFI, Webserver
- Mathematik — Gleichungen, Wertetabellen
- Kunst — 3D-Modellerstellung und Druck, Webseiten Gestaltung

Bewusst wurde Wert darauf gelegt, ein technisch anspruchsvolles Projekt mit aktueller „High-Tech" - Ausstattung so zu gestalten, dass sich die Projektinhalte trotzdem in kurzer Zeit an Jugendliche vermitteln lassen. Und es wurde nicht das technisch Mögliche aus dem Design heraus geholt, sondern extra Raum gelassen, um die Jugendlichen zu eigenen kreativen Modifikationen anzuregen.

2 Konzeptidee

Im Rahmen dieses Projektes soll ein Fotometer entstehen, dass für geringe Kosten auch von Personen ohne große Vorkenntnisse im Elektronik-, Computer und Softwarebereich selbst herstellbar ist.

Zu den grundlegenden Komponenten eines Fotometers gehören:

- eine Lichtquelle
- ein Monochromator
- eine Messküvette/Messgefäß
- ein Lichtsensor
- ein die Messdaten aufbereitendes System
- etwas zum Bedienen und zum Visualisieren von Messdaten (Benutzer Interface)
- sowie ein Halter für das Messgefäß, um reproduzierbare Messungen zu ermöglichen

Eine wesentliche Eigenschaft eines Fotometers ist die Messung der Lichtabschwächung (Lichtabsorption) bei einer bestimmten Wellenlänge des Lichts. Viele käufliche Fotometer erlauben die Vorauswahl dieser Wellenlänge. Technisch wird dies häufig durch Auffächern eines breitbandigen Lichtstrahls und Auswahl eines kleinen schmalbandigen Bereichs oder durch ein Farbfilter erwirkt. Diese optische Baugruppe – der sogenannte Monochromator – ist ein erheblicher Kostenfaktor. In diesem Projekt soll daher durch Einsatz von LEDs ein Monochromator unnötig werden. LEDs haben ein mehr oder weniger schmalbandiges Lichtspektrum. Durch die Auswahl der LED Farbe kann somit ein für die Messauf-

1 MINT ist eine Abkürzung und wird aus den Anfangsbuchstaben der Wissenschaftsbereiche **M**athematik, **I**nformatik, **N**aturwissenschaften und **T**echnik zusammengesetzt.

gabe „passendes“ Licht gewählt werden. Die Abbildung 1 zeigt schematisch die Lichtintensität in Abhängigkeit von der Wellenlänge für verschieden farbige LEDs.

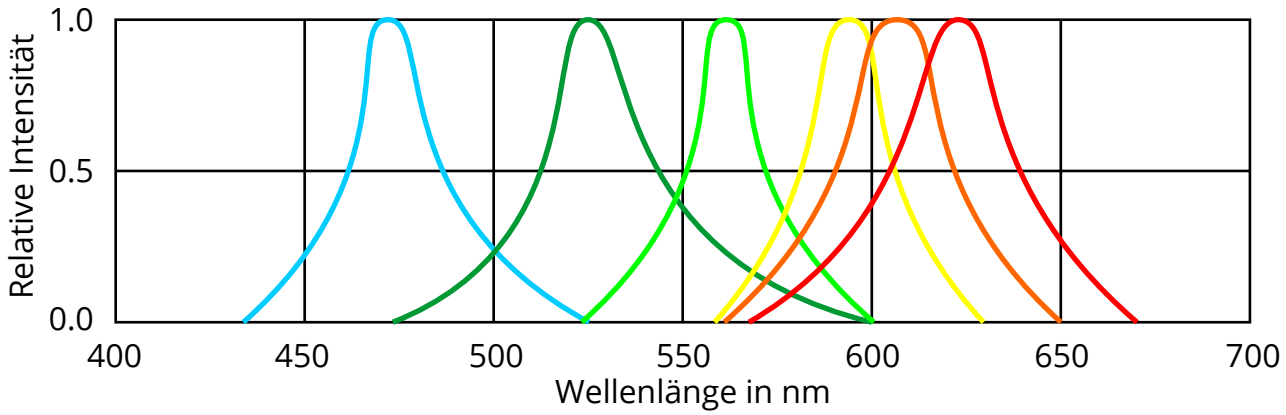


Abbildung 1: Lichtintensität verschieden farbiger LEDs

Die Breite des Spektrums der jeweiligen LED variiert von Hersteller zu Hersteller. Einerseits wäre ein schmaleres Band wünschenswert, um eine hohe Wellenlängenselektivität zu erreichen. Anderseits ermöglicht die Überschneidung der einzelnen Bänder auch die Beschränkung auf einige wenige LEDs, um einen breiten Wellenlängenbereich abdecken zu können.

Für die Auswahl eines geeigneten Lichtsensors sind folgende Kriterien wünschenswert:

- großer Wellenlängen Messbereich, um Messungen von ultraviolettem bis zu infrarotem Licht möglich zu machen
- hohe Auflösung, um Lichtintensitäten fein abgestuft unterscheiden zu können
- wenig Querempfindlichkeiten gegen Umwelteinflüsse (z.B. Temperatur)
- geringer elektrotechnischer Schaltungsaufwand, um an die Messergebnisse zu gelangen

Insbesondere der letzte Punkt legt die Verwendung eines Sensors mit integrierter digitaler Schnittstelle nahe. Signalverstärkende Elektronik und der damit verbundene Aufwand wird damit überflüssig.

Für die Steuerung der LED und die Verarbeitung der Daten des Lichtsensors drängt sich die Verwendung eines kleinen Computers auf. Um die Messdaten anzuzeigen und um Messungen starten zu können, werden noch eine Anzeigemöglichkeit und Bedienknöpfe benötigt. Gelingt es einen kleinen Computer auszuwählen, der seine Daten über eine von Tablet oder Smartphone nutzbare Datenschnittstelle anbieten kann, kann auf eine separate Anzeige und die Bedienknöpfe verzichtet werden.

Um aufseiten des Smartphones auf ein zusätzlich zu erstellendes Programm verzichten zu können, ist es wünschenswert, die Daten und die Möglichkeit Messungen zu starten in Form einer Webseite anzubieten, die von jedem internetfähigem Gerät, also auch von Smartphones, abgerufen werden kann. Die Einbindung eines Smartphones in das Konzept spart Kosten für Display und Bedienelemente und ist geeignet die Akzeptanz des Projekts bei Jugendlichen zu erhöhen. Der benötigte Computer sollte also am besten über eine WLAN/WIFI-Schnittstelle verfügen und die Möglichkeit haben einen kleinen Webserver zu realisieren.

Die einzelnen Bauteile des Fotometers sollen auch von Anfängern im Elektronikbereich handhabbar sein. Lötarbeiten sollen auch von „Erstlötern“ durchführbar und von kurz eingewiesenem Projektbegleitern begutachtbar und korrigierbar sein. Lötarbeiten an empfindlichen oder sehr kleinen Bauteilen kommen somit nicht infrage.

2.1 Anforderungen

Netzwerkschnittstelle

Das Fotometer sollte nicht mit dem Internet verbunden sein, um Sicherheitsbedenken hinsichtlich der Netzwerksicherheit Rechnung zu tragen. Die Absicherung von Geräten, die mit dem Internet verbunden sind, bedarf deutlich erweiterter Computerkenntnis und unterliegt der Bedingung, dass auch in der Folgezeit stetig Aktualisierungen eingebracht werden, um eben diese Netzwerksicherheit auch weiterhin gewährleisten zu können.

Nicht überall und erst recht nicht in vielen Schulgebäuden gibt es Zugang zu einem WLAN/WIFI – Netzwerk. Daher wäre es wünschenswert, wenn das Fotometer ein eigenes WLAN/WIFI – Netzwerk aufbaut und damit auch sicherstellt, dass keine direkte Verbindung zum Internet besteht.

Lichtsensor

Auf dem Markt sind mehrere Modelle mit großem Messbereich und hoher Auflösung verfügbar. Diese Hightech-Sensoren liegen jedoch häufig nur als wenige Millimeter große SMD-Bauteile vor und sind somit für Anfänger im Elektronikbereich eher ungeeignet. Es gibt jedoch auch einige dieser Sensorchips, die schon auf sogenannte „Breakout-Boards" montiert sind. Diese nur unwesentlich teureren Varianten sind auch für „ungeübte Elektroniker" einfach zu verwenden, weil sie die Lötarbeiten an kleinen und empfindlichen Bauteilen vermeiden.

Steuercomputer

Der Steuercomputer soll preiswert bis billig sein. Es gibt eine große Anzahl verschiedener vom technischen Aspekt her geeigneter Systeme z. B. ein „Raspberry Pi" oder „Arduino"-System bzw. eine entsprechende Variante. Das benötigte Entwicklungssystem für die Programmierung der Fotometer-Software sollte keine hohen Zusatzkosten verursachen und muss auch für Programmieranfänger geeignet sowie auch in Schul-Computerräumen, die nicht auf dem neusten technischen Stand sind, nutzbar sein.

3 Design-Entscheidungen

3.1 Steuercomputer

Als Steuercomputer (Mikrocontroller) wird ein Arduino kompatibles Computersystem ausgewählt, weil eine kostenlose Entwicklungsumgebung und viele günstige Modelle verfügbar sind. Die Modellvarianten mit WIFI-Erweiterung sind jedoch vergleichsweise gesehen teuer.

Die Steuercomputer mit Espressif ESP8266 System bringen eine WLAN/WIFI-Schnittstelle jedoch gleich von Haus aus mit, sind sehr günstig und lassen sich als Arduino – Variante in die Software-Entwicklungsumgebung einpassen. Die Anzahl der digitalen und analogen Ein- und Ausgabe Anschlüsse ist zwar geringer als bei den „originalen" Arduinos, sind aber für dieses Projekt mehr als ausreichend. Nachteilig ist jedoch, dass es nur wenige Anbieter auf dem deutschen Markt gibt.

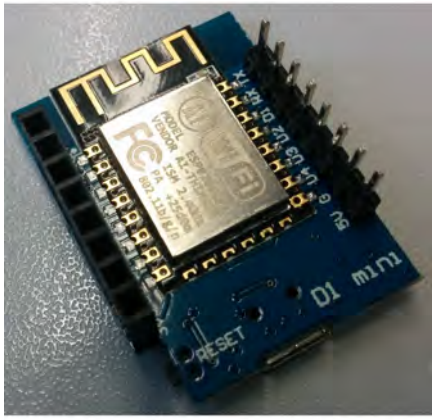


Abbildung 2: WeMos „D1 mini"

Standardmäßig haben die Espressif ESP8266 Systeme keine Programmierschnittstelle über einen USB-Anschluss. Dies wäre jedoch wünschenswert, um eine einfache Handhabung auch für Anfänger

zu ermöglichen. Glücklicherweise sind auch Mikrocontroller-Boards mit Espressif ESP8266 System und mit einer USB-Anschluss-Baugruppe gemeinsam auf einem Board verfügbar. Sie sind etwas teurer, als die Varianten ohne USB-Anschluss sind dafür dann aber sehr einfach in der Anwendung.

Für dieses Projekt wird die Variante mit USB entsprechend einem Design der Firma WeMos, der „D1 mini“, ausgewählt.

3.2 Lichtsensor

Abbildung 3: Lichtsensor TSL2561

Als Lichtsensor soll der TSL2561 – Chip der Firma TAOS auf einem Breakout-Board eingesetzt werden. Der Sensor hat eine hohe Auflösung und Empfindlichkeit im sichtbaren Lichtspektrum und ermöglicht auch Messungen im Infrarot-Bereich. Seine Messergebnisse können nach einem vom Hersteller verifiziertem Algorithmus in LUX berechnet werden und es gibt mehrere Versionen eines Breakout-Boards auf dem Weltmarkt, die auch bei Distributoren im Inland verfügbar sind. Auch in einem Fotometrie-Projekt[2] von Oliver Happel hat sich dieser Chip bewährt.

Einige Kenndaten:

- Empfindlichkeit an die des menschlichen Auges angeglichen
- Automatische Unterdrückung des 50/60Hz Flackerns durch Einstrahlung von elektrischen Licht in Innenräumen
- Sehr geringer Energiebedarf (Minimum: 0,75 mW)
- Hohe dynamische Auflösung von 1.000.000 zu 1 durch programmierbare Signalverstärkung und Integrationszeit
- Baugröße (L · B · H): 3,8 mm · 2,6 mm · 1,35 mm

Abbildung 4: TSL2561 Chip

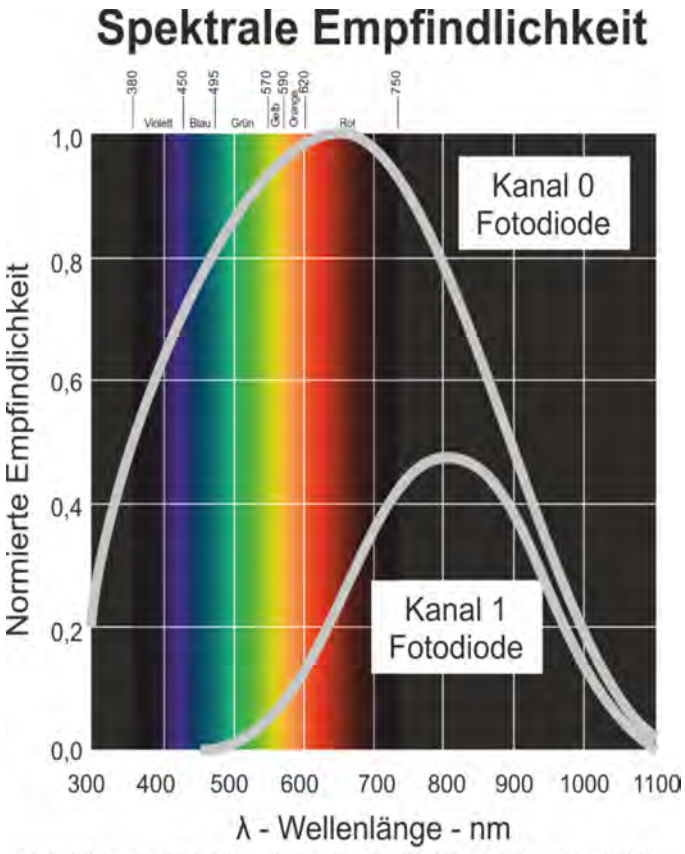


Abbildung 5: Spektrale Empfindlichkeit des TSL2561 Chips

2 Oliver Happel (2015) Analytische Methoden mit dem LED Photometer. AATIS-Praxisheft 25

3.3 Energieversorgung

Die Energieversorgung des Gesamtsystems soll über den USB-Anschluss auf dem „D1 mini“ Modul mit einem USB-Steckernetzteil oder einer USB-Powerbank erfolgen. Bei der Nutzung einer Powerbank wird auch ein mobiler Betrieb des Fotometers möglich. Da die Stromaufnahme der gesamten Schaltung nur gering sein wird, ist auch eine leistungsarme Powerbank ab ca 2000 mAh geeignet.

3.4 Lichtquelle

Für das Fotometer stehen zwei Ausbauvarianten zur Verfügung: eine mit einer einfarbigen LED und eine weitere mit einer dreifarbigen LED. In den folgenden Texten wird ggf. auf die Unterschiede der jeweiligen Variante hingewiesen.

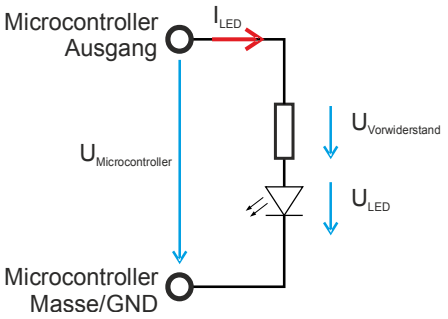


Abbildung 6: Anschluss der LED

LEDs benötigen zum Betrieb einen strombegrenzenden Widerstand. Die Höhe des maximalen Stroms und die zum Betrieb benötigte Spannung lässt sich üblicherweise aus dem Datenblatt entnehmen. In diesem Projekt wird beabsichtigt die LED direkt am Ausgangspin des Mikrocontrollers zu betreiben. In diesem Fall ist es somit auch wichtig den maximal möglichen Strom am Ausgang des Mikrocontrollers zu berücksichtigen. Der ESP8266 Chip kann am Digitalausgang maximal 12 mA bei einer Ausgangsspannung von 3,3 Volt zur Verfügung stellen. Die Betriebsspannung einer LEDs wird oft bei einem Strom von 20 mA angegeben. Bei der hier verwendeten LED soll der Strom jedoch nur 12 mA betragen und es fällt dann auch eine kleinere Spannung als im Datenblatt für 20 mA angegeben über der LED ab. Aus dem Datenblatt der LED lässt sich oft aus einer Kennlinie auch ein Strom bei 12 mA ablesen (z. B. 2 Volt). Über dem Vorwiderstand der LED müssen somit ca 1,3 Volt abfallen (Zusammen 3,3 Volt). Über das Ohmsche Gesetz lässt sich somit der Widerstandswert berechnen: $R=\frac{U}{I}$. In unserem Fall also:

$$R=\frac{1{,}3\,V}{0{,}012\,A}=108{,}33\,\Omega \quad .$$

Einen Widerstand mit 108,33 Ohm gibt es nicht zu kaufen, daher wird in diesem Fall der nächst größere kaufbare Wert – 120 Ohm – gewählt.

Wird eine LED mit einem anderen Spannungsabfall bei maximal 12 mA gewählt, ist die Spannung am Vorwiderstand eine andere und somit ist der Widerstand neu zu berechnen. Bei RGB-LEDs ist der Spannungsabfall bei jeder LED-Farbe anders und muss für jede Farbe einzeln dem Datenblatt entnommen werden. Für die in diesem Projekt mehrfach verwendete RGB-LED „LL-509RGBC2E-006“ (Reichelt Artikel-Nr.: LED LL 5-8000RGB) haben sich folgende Werte für die Vorwiderstände bewährt:

blaue LED: R = 0 Ohm, grüne LED: R = 62 Ohm, rote LED: R = 182 Ohm

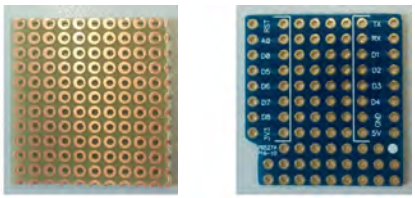
Liegt kein Datenblatt vor, kann der Spannungsabfall experimentell durch Variation des Vorwiderstands ermittelt werden.

Soll nur jeweils eine LED Farbe verwendet werden, lassen sich viele LEDs mit unterschiedlichen Leuchtfarben finden die das Kriterium 12 mA bei 2 Volt erfüllen und dann bei Bedarf gegeneinander einfach austauschbar sind.

3.5 Schaltungsaufbau

Die elektrische Verbindung der einzelnen Baugruppen soll auf einer Platine, die auch die räumliche Ausrichtung der einzelnen Bauteile zueinander bestimmt, erfolgen. Eine preiswerte Lösung hierfür sind sogenannte Lochrasterplatinen. Diese Platine stellt die elektrischen Verbindungen zwischen Mikrocontroller Baugruppe, dem Lichtsensor Modul und der LED her. Eine Sandwitch-Struktur – unten die Mikrocontroller Baugruppe, mittig die Lochrasterplatine und darauf das Lichtsensor-Modul und die LED – ermöglicht einen besonders platzsparenden Aufbau. Da von beiden Seiten Bauteile auf der Platine befestigt werden müssen, ist eine doppelseitige Kupferbelegung sinnvoll. Benötigt wird eine Lochrasterplatine mit 10 mal 11 Löchern. Üblicherweise werden Lochrasterplatinen in größeren Baugrößen angeboten, sie lassen sich jedoch leicht in einzelne Teile unterteilen. Hierzu wird die Platine zersägt oder mit einem Messer so angeritzt, dass einzelnen Löcher genau mittig unterteilt werden, und dann über einer Kante gebrochen. Da somit an den Rändern halbe Lochreihen übrig bleiben wird pro Platine ein Bereich von 11 mal 12 Löchern benötigt. Aus einer handelsüblichen Lochrasterplatine in den Maßen von 100 mm mal 160 mm lassen sich so 15 kleine Platinen gewinnen.

Die einfachere Lösung besteht darin eine entsprechende Platine im gewünschten Format zu kaufen. Passend zum WeMos „D1 mini“ – Modul gibt es so eine Platine („D1 mini“- Proto Board).



4 Software Entwicklung

Die Entwicklung und Änderung der Software kann mit der kostenlos erhältlichen Arduino IDE (integrated development environment / integrierte Entwicklungsumgebung) erfolgen. Diese Software ist auch für Programmiereinsteiger konzipiert und ermöglicht einen einfachen Einstieg in die Welt der Programmierung von Mikrocontrollern. Mit der Arduino IDE kann neuer Programmcode erstellt oder bereits bestehender bearbeitet werden und danach einfach über einen USB Verbindung auf das Microcontroller Board geladen werden. Alles ist im Vergleich zu professionellen Entwicklungsumgebungen etwas einfacher gehalten. So gibt es z. B. Syntax-Highlighting – das ist die farbige Hervorhebung von bekannten Schlüsselworten im Programmcode – aber eine automatische Code-Vervollständigung gibt es (noch) nicht. Dennoch bietet die IDE aber auch erfahrenen Anwendern die meisten Möglichkeiten aktueller Softwareentwicklungssysteme.

Was ist Arduino?

- Unter Arduino versteht man eine aus Soft- und Hardware bestehende Computer Plattform
- Die Hardware besteht aus einem Mikrocontroller mit analogen und digitalen Ein- und Ausgängen
- Die Entwicklungsumgebung (IDE) soll auch technisch weniger Versierten den Zugang zur Programmierung und zu Mikrocontrollern erleichtern
- Es gibt sehr viele Varianten mit und ohne Extras (Preisspanne 2€ bis 100€) und auch diverse Hersteller kompatibler Ausführungen

Die Arduino IDE kann von der Webseite der Arduino Initiative unter „https://www.arduino.cc/“ heruntergeladen werden.

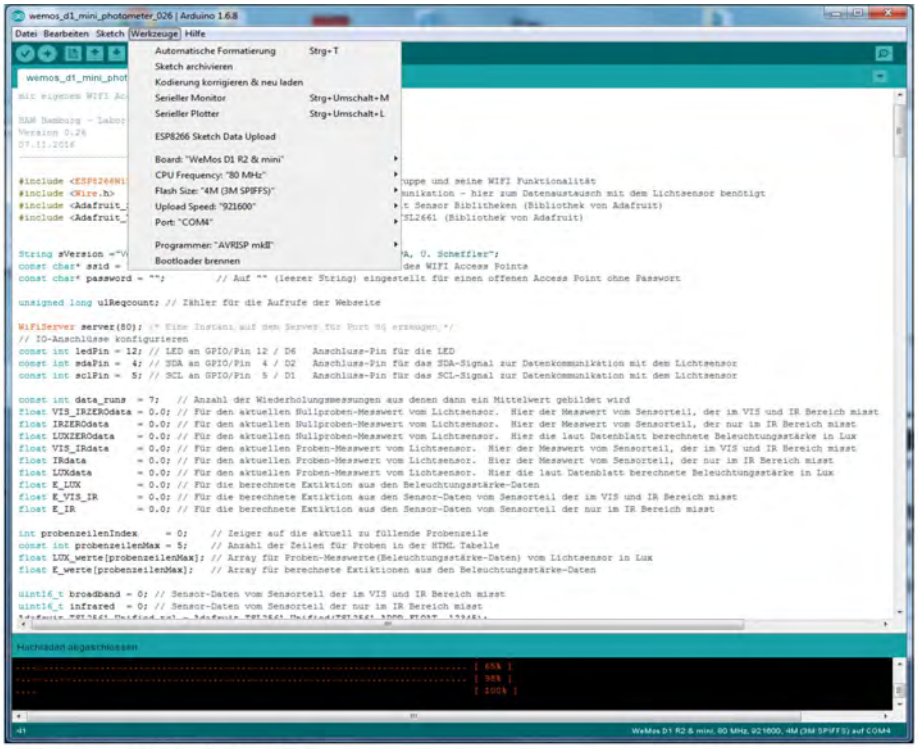


Abbildung 7: Arduino IDE

Für dieses Projekt muss diese Softwareumgebung um die Möglichkeit auch die ESP8266 Systeme programmieren zu können, ergänzt werden. Hierfür eignen sich alle Versionen der Arduino IDE ab Version 1.6.5.

Hilfreiche Weblinks zum Thema Programmierung des ESP8266 mit der Arduino IDE finden sich unter:

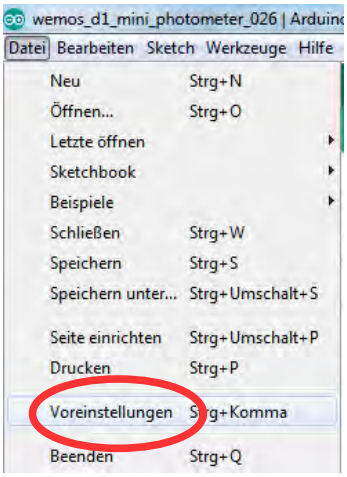
https://arduino-hannover.de/2015/04/08/arduino-ide-mit-dem-esp8266/

https://www.wemos.cc/tutorial/get-started-arduino.html

https://github.com/esp8266/Arduino

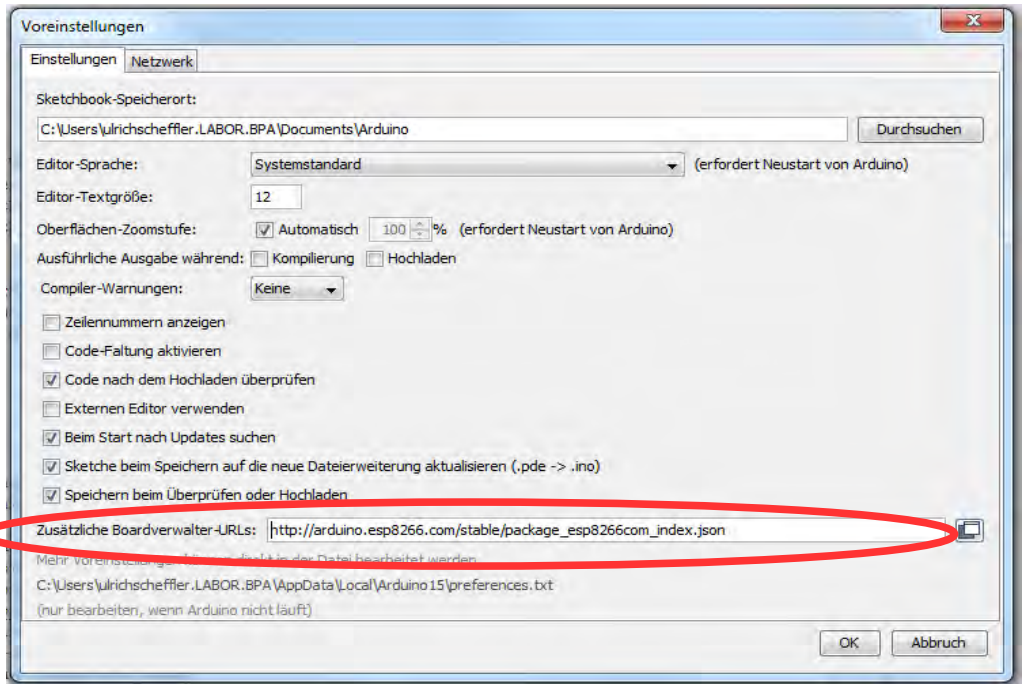
Die Ergänzung der Arduino IDE erfolgt am einfachsten so:

1. Arduino IDE starten.
2. Im Menü "Datei" den Punkt "Voreinstellungen" auswählen.

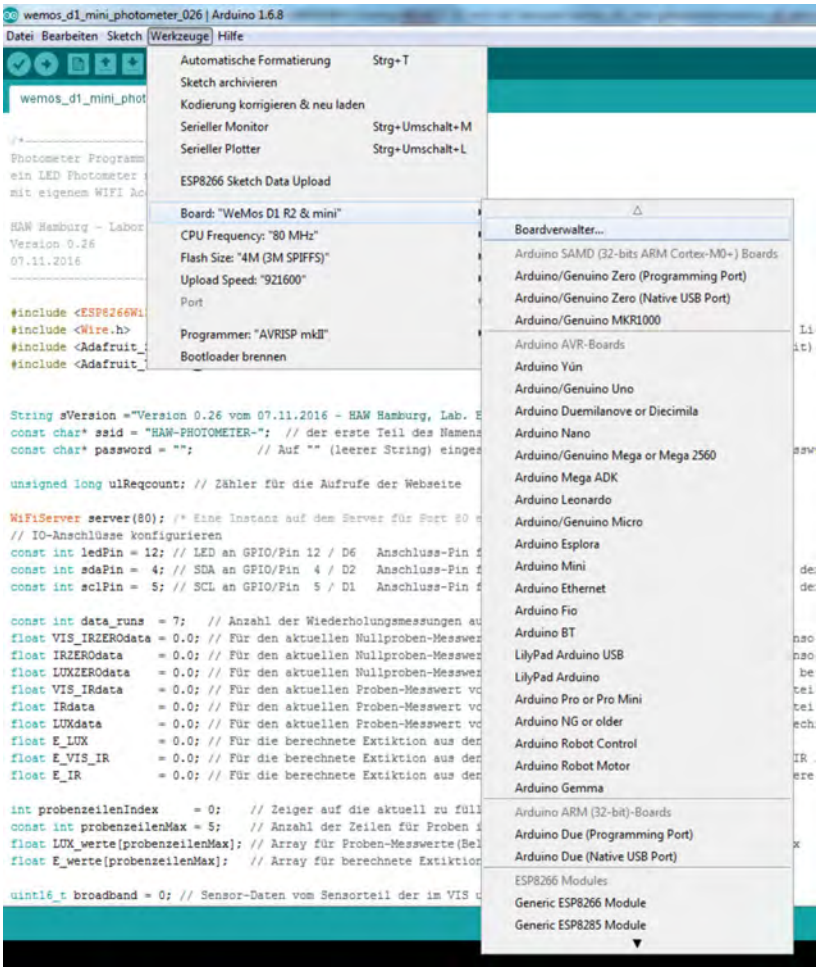


3. Auf der Karteikarte "Einstellungen" unter "Zusätzliche Boardverwalter-URLs" http://arduino.esp8266.com/stable/package_esp8266com_index.json

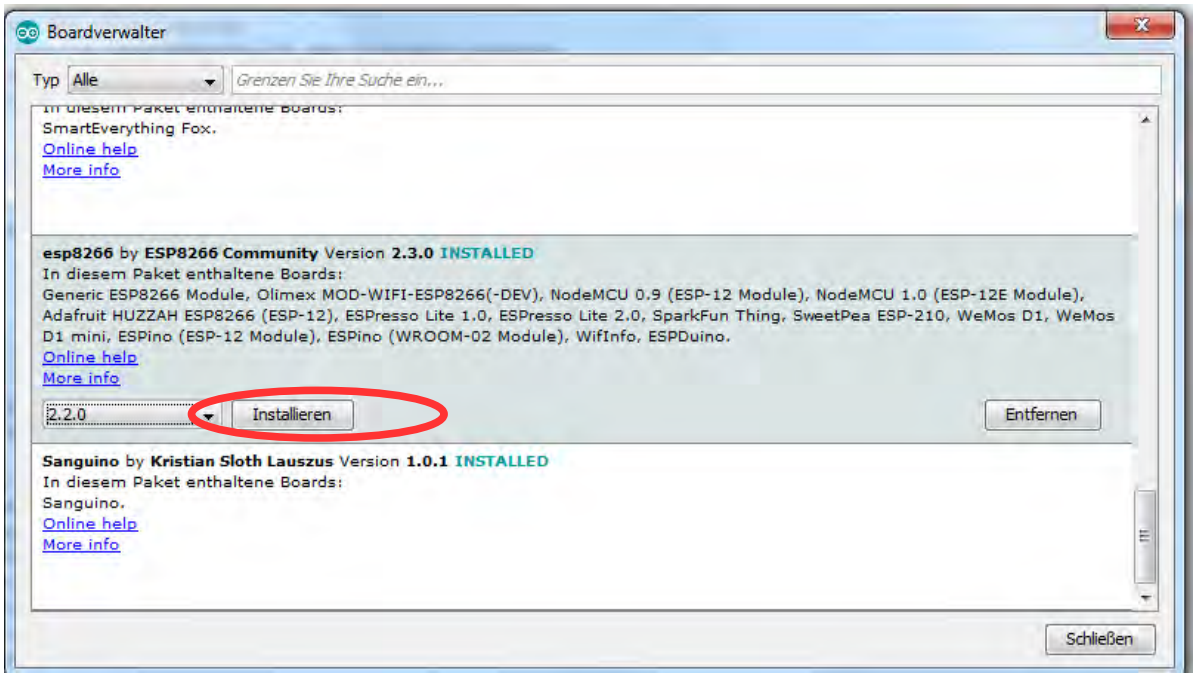
   eintragen.



4. Dann im Menü "Werkzeuge" den Punkt "Board: "..."" auswählen (Hinweis: für "..." steht immer das zuletzt ausgewählte Zielsystem. Hier im Screenshot: Board: "WeMos D1 R2 & mini")

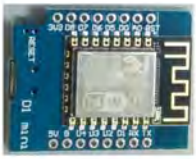


5. Den obersten Punkt "Boardverwalter..." auswählen.



6. Den Eintrag "esp8266 – by ESP8266 community" suchen (meist weit unten) und anklicken und "Installieren" drücken. (Hinweis: Im Screenshot ist die Installation bereits ausgeführt worden.)
7. Arduino IDE beenden und erneut starten.
8. Jetzt kann unter "Werkzeuge" "Board: "..."" der Eintrag "WeMos D1 R2 & mini" ausgewählt werden.
9. Ab jetzt sollten Programme für das Zielsystem "WeMos D1 mini" erzeugt werden können.

5 Bauteilübersicht und -kosten

Mikro-controller	Lichtsensor	Lichtquelle	Vorwider-stand für LED	Adapter-platine	Spannungs-versorgung
		Einfabig  oder Dreifarbig 		 oder 	
WeMos D1 mini	TSL2561	LED	Widerstand	D1 mini Proto-Board oder Lochraster-platine	Powerbank

Kostenübersicht

Nr	Bauteil	Funktion	Bezug in Deutschland			Direktimport
			Anbieter	Artikel Nr	Kosten	Kosten
1	WeMos D1 mini	Mikrocontroller	Eckstein	CP06056	8,65 €	2,20 €
2	TSL2561	Lichtsensor	roboter-bausatz	RBS11640	3,84 €	0,88 €
3	RGB LED	Lichtquelle	Reichelt	LED LL 5-8000RGB	0,61 €	0,15 €
4	Widerstand (62 Ω)	Vorwiderstand LED	Reichelt	METALL 62	0,08 €	0,03 €
5	Widerstand (182 Ω)	Vorwiderstand LED	Reichelt	METAL 182	0,08 €	0,03 €
6a	D1 mini ProtoBoard	Adapterplatine	Eckstein	CP18003	1,95 €	0,51 €
6b	Lochrasterplatine	Adapterplatine	Reichelt	UP 832EP (reicht für 15 Stück, 4,40 €)		
7	Powerbank	Energiever-sorung	Reichelt	GOO 71599	3,99 €	2,59 €
				Summe	**19,20 €**	**6,39 €**

Stand: Oktober 2018

Anbieter in Deutschland

Eckstein GmbH	(Versandkosten ca. 3,49 €)	https://eckstein-shop.de
Reichelt Elektronik	(Versandkosten ca. 5,60 €)	http://www.reichelt.de
Azando (roboter-bausatz)	(Versandkosten ca. 2,99 €)	https://www.roboter-bausatz.de

Anbieterplatform für Direktimport

Aliexpress	(Versandkosten oft inklusive)	https://www.aliexpress.com

6 Zusammenbau

6.1 Löten

Die Mikrocontroller Platine und das Lichtsensor Modul werden in speziellen elektrisch leitfähigen **Tüten** angeliefert. Dies sollte man als Hinweis auf empfindliche Bauteile deuten. Tatsächlich sollten elektrostatische Entladungen über diese Bauteile vermieden werden. Profis legen bei der Verarbeitung ein mit Schutzerde verbundenes Armband an und arbeiten nur auf leitfähigen Unterlagen. Meistens ist jedoch auch eine Verarbeitung dieser Bauteile auf einem einfachen Tisch und ein Abbauen evtl. vorhandener elektrostatische Ladung durch vorheriges Anfassen des Tischbeins aus Metall oder der Heizung gut möglich.

Abbildung 8: Bauteile vor dem Zusammenbau

Nach dem Auspacken können die benötigten Teile herausgesucht werden. In den Tüten für das Mikrocontroller Board und der Adapterplatine sind einige für dieses Projekt nicht benötigte Stift- und Buchsenleisten zu viel drin.

Benötigt werden:

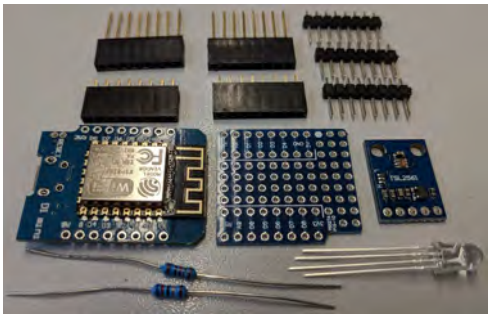


Abbildung 9: Benötigte Bauteile

Anzahl	**Stift-/Buchsenleiste**
2	8-polige Stiftleiste
2	8-polige Buchsenleiste mit kurzen Beinchen
1	4-polige Stiftleiste (Abgetrennt von einer 8-poligen Stiftleiste) für die Version mit einer dreifarbigen LED. 2 polige Stiftleiste für die Version mit nur einer LED Farbe.
1	4-polige Buchsenleiste (Abgetrennt von einer 8-poligen Buchsenleiste) für die Version mit einer dreifarbigen LED. 2 polige Buchsenleiste für die Version mit nur einer LED Farbe.
1	4-polige Buchsenleiste mit um 90° abgewinkelten Beinchen (Abgetrennt von einer 8-poligen Buchsenleiste und danach Beinchen auf der Tischfläche umgebogen).
1	4-polige Stiftleiste (Abgetrennt von einer 8-poligen Stiftleiste).

Auf der Mikrocontroller Platine soll eine 8-polige Stiftleiste und eine 8-polige Buchsenleiste verlötet werden. Das Bild in Abbildung 10 zeigt auf welcher Seite und welche Leiste wo bestückt werden soll. Bewährt hat sich zunächst erstmal jeweils nur einen Pin anzulöten, dann die Ausrichtung zu überprüfen und ggf. Nachjustieren – d. h. die Lötstelle wieder zu erwärmen und das Bauteil ausrichten – und dann erst die restlichen Anschlusspins zu verlöten.

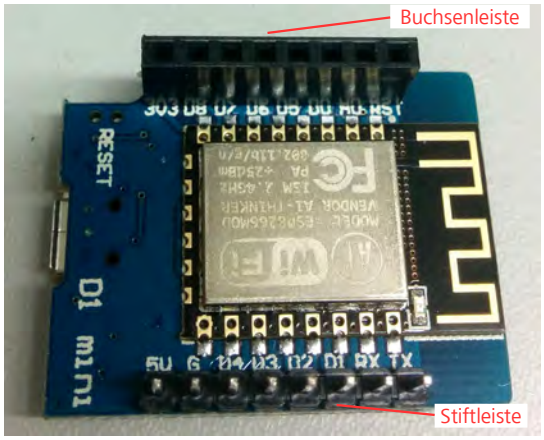


Abbildung 10: WeMos D1 mini mit Stift- und Buchsenleiste

Die Herstellung der Adapterplatine macht am meisten Mühe. In der Zeichnung in Abbildung 14 auf Seite 14 ist ein Bestückungsplan abgebildet. Empfehlenswert ist es zuerst die Widerstände bzw. den Widerstand einzubauen. Die Drahtenden des Widerstands können auf der Lötseite der Platine umgebogen, verlötet und abgeschnitten werden. Die abgeschnittenen Drahtenden können benutzt werden, um die Drahtbrücken auf der Lötseite herzustellen. Dann kann die abgewinkelte 4-polige Buchsenleiste bestückt und verlötet werden. Danach die 4- bzw. 2-polige Buchsenleiste einbauen und verlöten – dabei auf einen rechtwinkligen Einbau achten. Jetzt fehlen noch die beiden 8-poligen Leisten. Auch hierbei gibt das Foto in Abbildung 11 einen Hinweis für den richtigen Einbau.

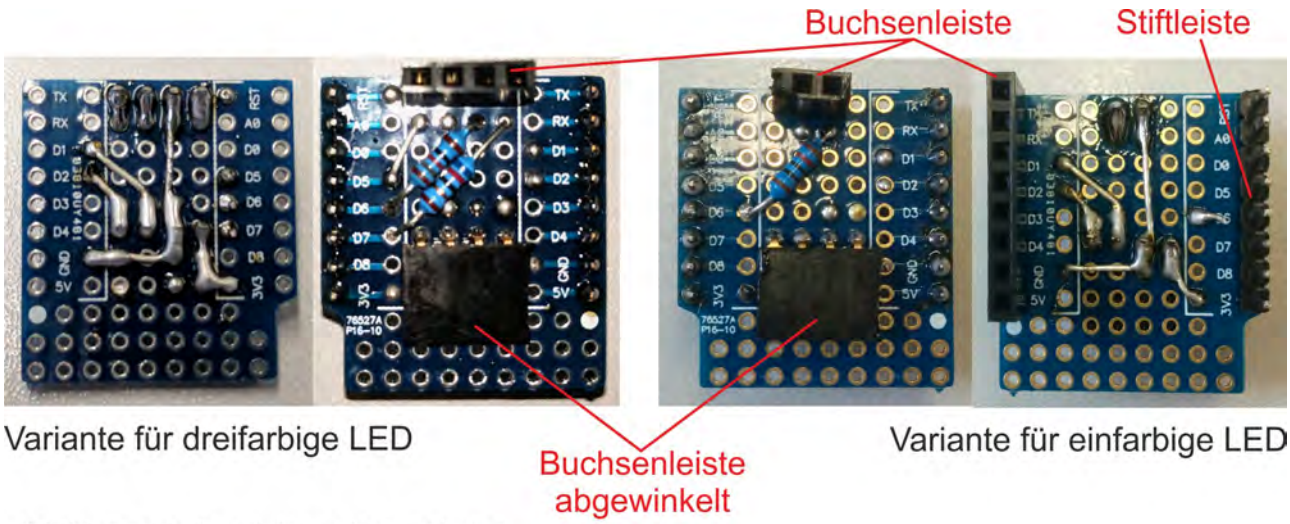


Abbildung 11: Bestückte Adapterplatine

Am hier genutzten Lichtsensor-Modul muss eine 4-polige Stiftleiste so eingelötet werden, dass die lange Seite der Pins auf der Bauteilseite bleibt (siehe Abbildung 12). In das Loch mit der Beschriftung „INT" sollte kein Stift eingebaut werden, er wird nicht benötigt und es kommt später nicht so leicht zu einer falschen Montage. Der eigentliche Lichtsensor-Chip sollte möglichst nicht mit dem Finger angefasst werden, weil sonst ein Fettfilm zurückbleibt, der hinterher im Betrieb störend sein kann.

Anmerkung: Es gibt diverse Varianten des Lichtsensor-Moduls am Markt, die grundsätzlich alle geeignet sind. Wird nicht das hier abgebildete Lichtsensor-Modul (Platinenkennzeichnung auf der Rückseite: GY-2561) verwendet werden, muss ggf. der Anschlussplan an die abweichende Anordnung der Anschlusspins angepasst werden. Orientierung liefern hierbei die Beschriftungen der Anschlüsse. Verwendet werden: VCC, GND, SCL und SDA. Bei Verwendung eines anderen Lichtsensor Moduls ist auch die geometrische Lage des Lichtsensor Chips zu beachten. Er soll im montiertem Zustand der LED möglichst exakt gegenüber stehen, d. h. weder seitlich noch in der Höhe verschoben. Zusätzlich ist in diesem Fall zu beachten, das auch der Abstand des Lichtsensors zur Adapterplatine nicht zu gering werden sollte, da sonst die üblichen Küvetten, die dann später für die Messungen verwendet werden sollen, nicht mehr mittig durchstrahlt werden.

Die Beinchen der LED können kurz nach der Verdickungsstelle abgetrennt werden, sodass die Beinchen danach noch ca 5 mm lang sind. Die LED soll mit einer Stiftleiste im rechten Winkel verlötet werden. Das Bild in Abbildung 12 bietet hierfür eine gute Orientierung, wie es hinterher aussehen soll. Aber Achtung! Die LED hat mehrere Anschlüsse, die in diesem Projekt auch nicht vertauscht werden dürfen (sonst funktioniert die LED nicht - aber meistens geht auch nicht gleich was kaputt). Die LED hat im Inneren einen dicken, fetten Teil und mehrere oder einen kleinen, schlanken Teil. Der Anschluss mit dem dicken, fetten Teil im Inneren der LED ist die Kathode (MINUS Seite) der LED.

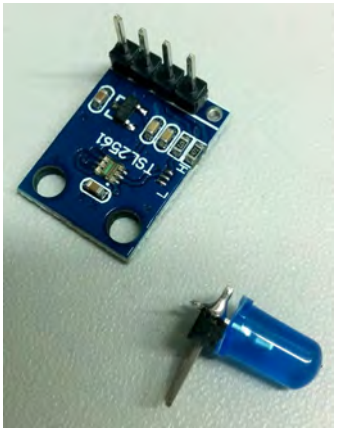


Abbildung 12: Sensor Modul und LED mit Stiftleisten

Folgender Ablauf hat sich bewährt: Stiftleiste mit den langen Pins in die Buchsenleiste der Adapterplatine einstecken. Platine so drehen, dass diese neue Kombination oben ist. Die einfarbige LED kann jetzt so an die Stiftleiste gelötet werden, dass die Kathode (MINUS Seite) der LED links ist - bei der dreifarbige LED ist die Kathode der zweite Pin von links.

Abbildung 13: LEDs mit Stiftleisten

links dreifabige LED rechts einfabige LED

Das Bild in Abbildung 20 auf Seite 16 lässt erkennen wie es für die einfarbige LED hinterher aussehen soll.

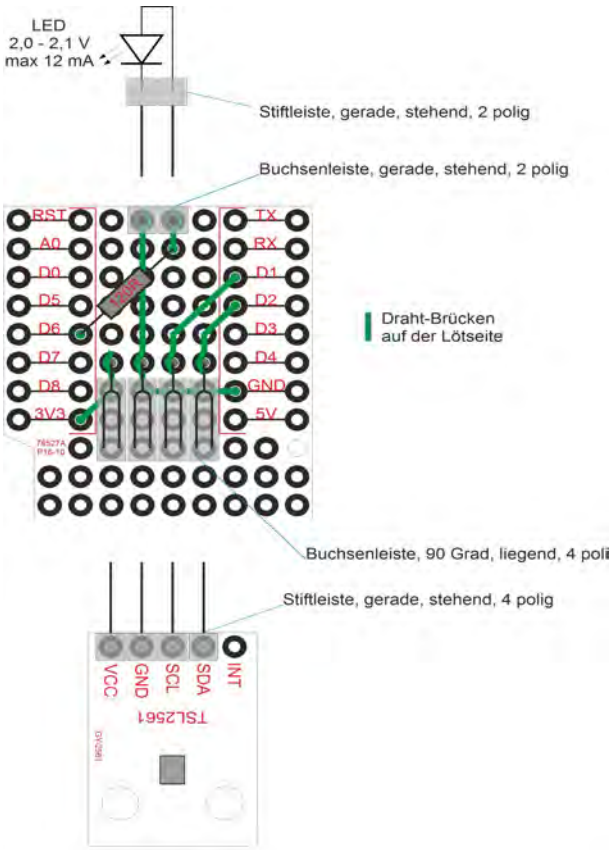


Abbildung 14: Bestückungsplan Adapterplatine für einfarbige LED Variante

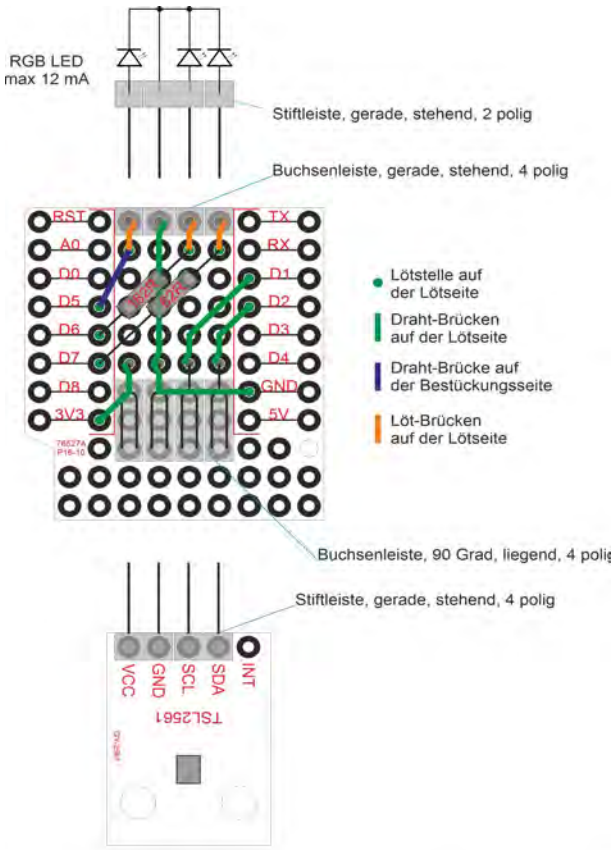


Abbildung 15: Bestückungsplan Adapterplatine für dreifarbige LED Variante

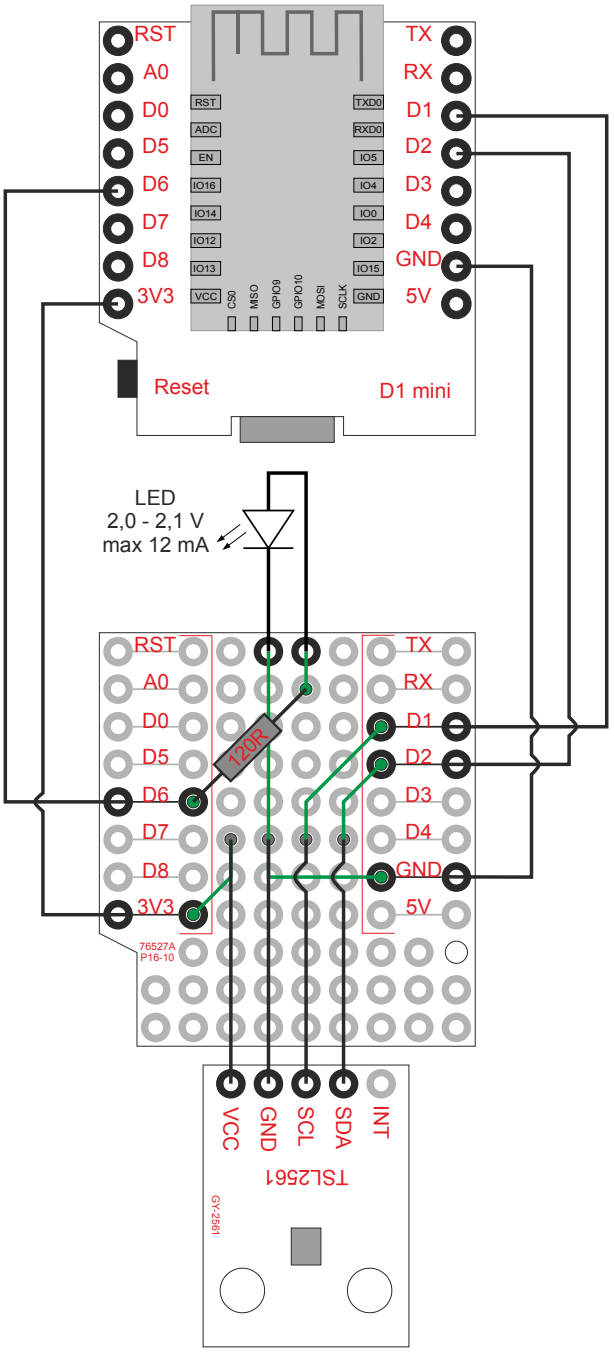


Abbildung 16: Schaltplan des Fotometers für einfarbige LED Variante

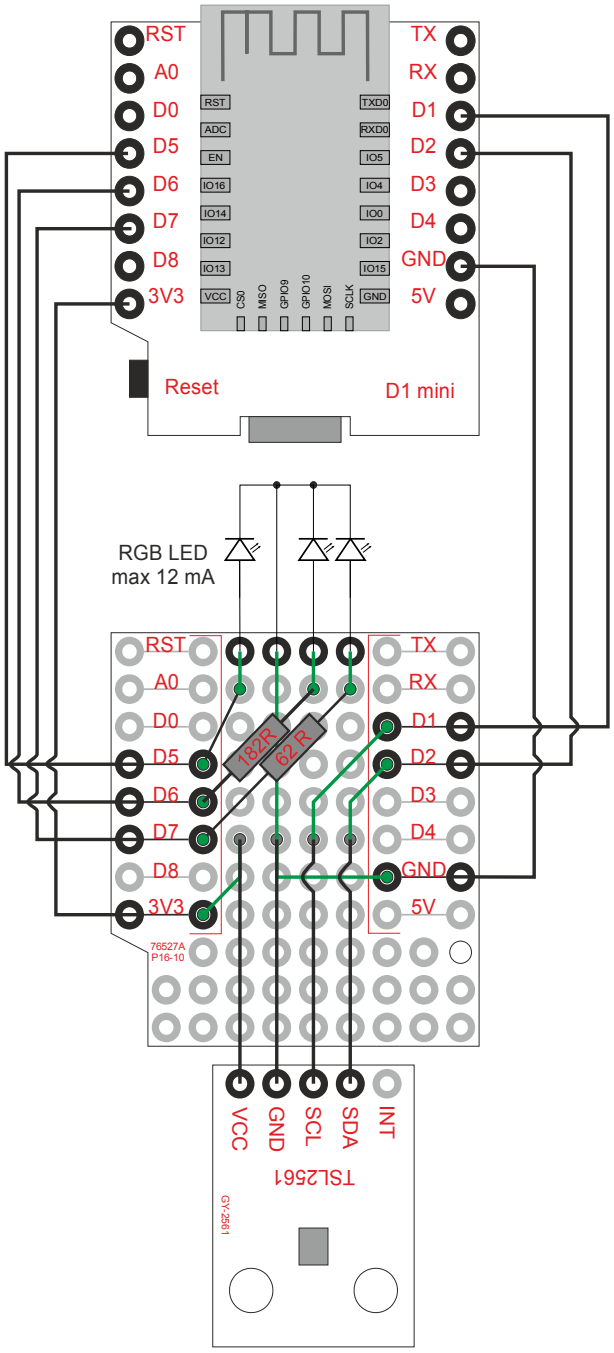


Abbildung 17: Schaltplan des Fotometers für dreifarbige LED Variante

6.2 Zusammenstecken

Wenn alles richtig verlötet wurde, sollte sich die Adapterplatine nun so auf die Mikrocontroller-Platine stecken lassen, dass beide Platinen die kleine Aussparung an der einen Ecke genau übereinander liegend haben. Siehe hierzu auch die Fotos in Abbildung 19 und Abbildung 20.

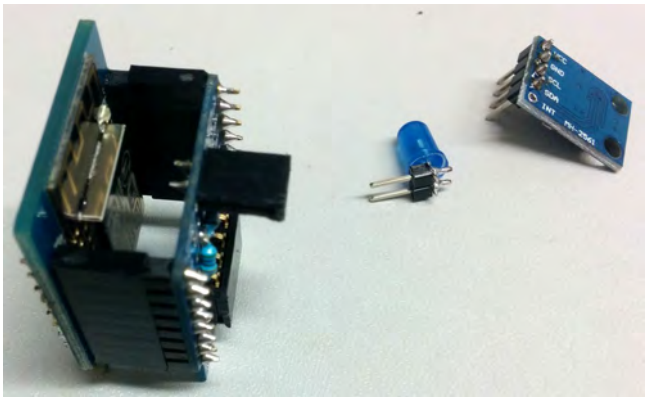


Abbildung 18: Microcontroller- und Adapterplatine zusammenstecken

Das Lichtsensor Modul und die LED lassen sich nun einfach einstecken.

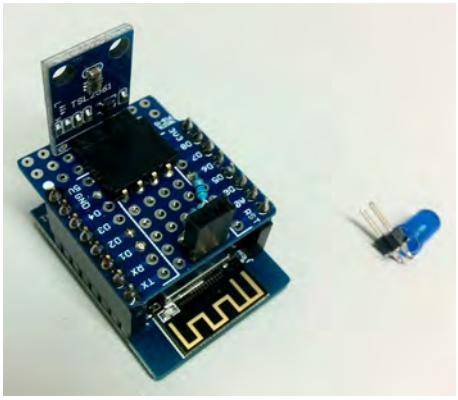


Abbildung 19: Lichtsensor einstecken

Abbildung 20: Alle Bauteile zusammengesteckt

6.3 Software einspielen

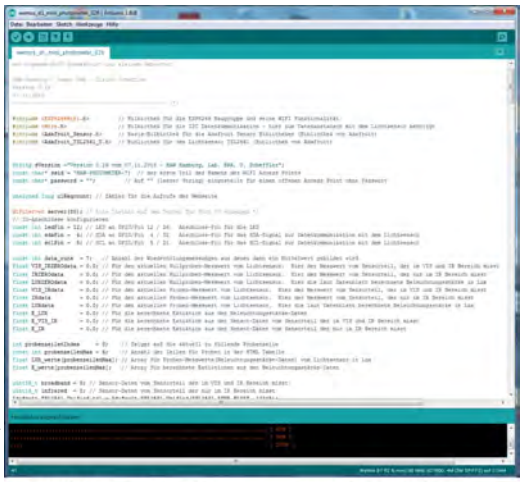


Abbildung 21: Programm einspielen

Zum Einspielen der Software muss der WeMos D1 mini über seine USB Schnittstelle mit dem PC verbunden werden. Evtl. muss jetzt noch einmalig eine Treibersoftware für den USB Chip des WeMos D1 mini installiert werden. Der passende Treiber für die CH340G Chip Serie ist z. B. unter:

https://www.wemos.cc/downloads/CH341SER_win.zip

zu finden. Die Treibersoftware stellt die Simulation einer seriellen Schnittstelle – ein sogenannter COM Port – in Windows zur Verfügung. Windows identifiziert ggf. mehrere vorhandene COM Ports anhand einer nachgestellten Nummer z. B. COM4. Diese Bezeichnung ist wichtig für die Verwendung in der Arduino IDE, da der IDE Software mitgeteilt werden muss, wo unser zu programmierendes Mikrocontroller-System angeschlossen ist.

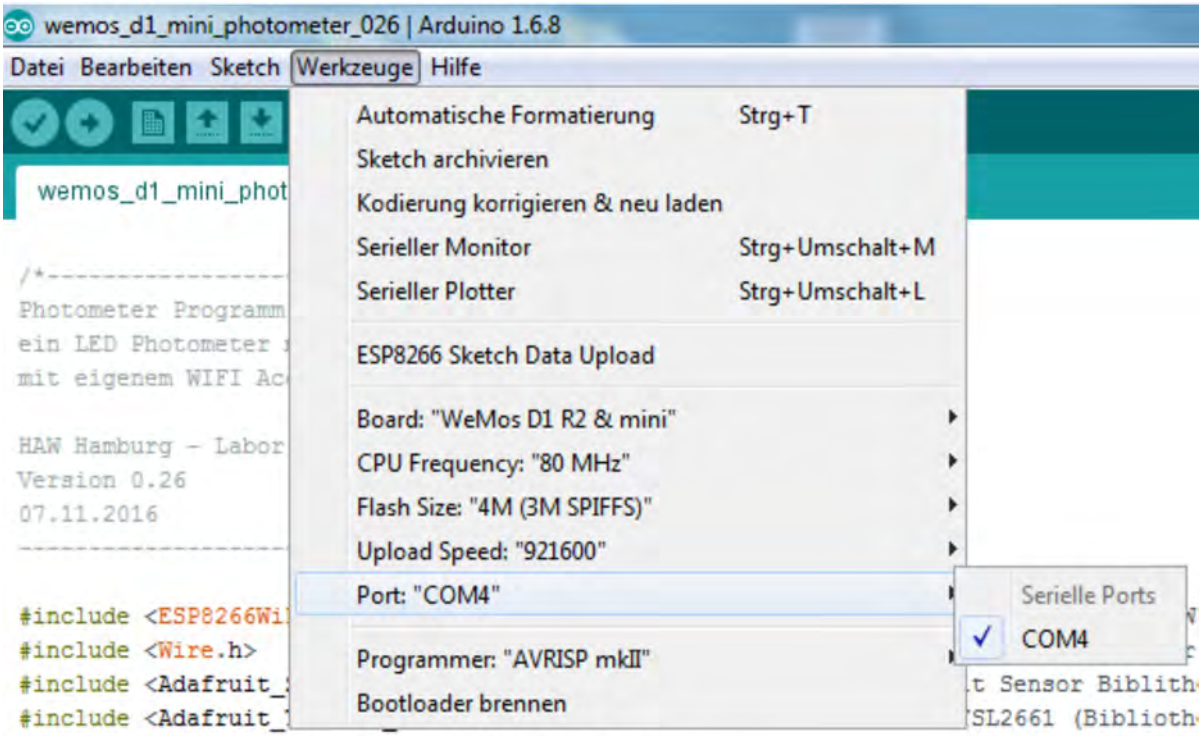


Abbildung 22: Auswahl des Programmieranschlusses

Hierzu im Werkzeug – Menü im Untermenü „Port“ den gewünschten (richtigen) Anschluss auswählen.

Um das Programm zum Mikrocontroller übertragen zu können, muss es zunächst übersetzt werden, d. h. in eine für den Mikrocontroller geeignete Form gebracht werden. Durch drücken des runden Pfeil-Knopfes – in Abbildung 23 die runde weiße Fläche mit dem Pfeil unterhalb des „B“ vom Bearbeiten Menü – wird das Programm übersetzt und im Anschluss daran an den Mikrocontroller gesendet.

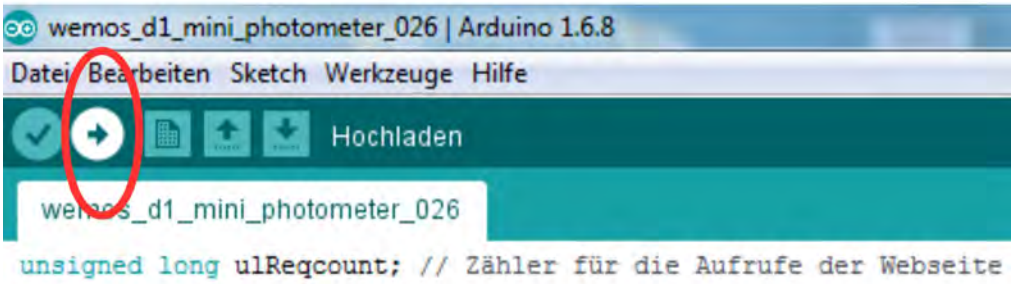


Abbildung 23: Transfer des Programms zum Microcontroller starten

7 Gehäuse(ein)bau

Grundsätzlich lässt sich das Fotometer auch ohne Gehäuse nutzen. Da eine Messung von Licht beabsichtigt ist, kommt es jedoch durch das Umgebungslicht zu Störungen, die die Messung von verlässlichen Ergebnissen deutlich erschweren. Es sind einfache Gehäuse-Varianten aus Papier (nicht so haltbar, wenn Flüssigkeiten im Spiel sind), Holz oder aus nicht mehr benötigten Kunststoffverpackungen (z. B. runde Kaugummi-Dose) einsetzbar.

Abbildung 24: Ins Gehäuse einbauen

Für dieses Projekt steht jedoch auch ein Gehäuseentwurf als STL-Datei für den Ausdruck auf einem 3D-Drucker bereit. Dieser Gehäuseentwurf wurde mit dem 3D-Zeichenprogramm OpenSCAD erstellt. Dieses Programm ist freie Software und steht unter der GPL Version 2 Lizenz kostenfrei zum Download unter http://www.openscad.org zur Verfügung. Der Entwurf für dieses Gehäuse ist in Form einer Textdatei, die in OpenSCAD eingelesen werden kann, im Anhang unter 11.4 Gehäusemodell-Code für OpenSCADab Seite 31 als Text zu finden oder kann beim Autor via email angefragt werden. Somit sind eigene Anpassungen leicht möglich. OpenSCAD kann STL-Datei erzeugen. Diese STL Dateien können dann mit einem Slicer – Programm (es gibt auch hierfür diverse freie Software) für den 3D-Drucker weiter verarbeitet werden, um die Steuerdateien für den 3D-Drucker zu erstellen. Für einen schnellen Einstieg in OpenSCAD bieten sich die folgenden beide deutschsprachigen Einführungen an:

https://www.tuebix.org/downloads/tuebix.2015.knopper-openscad.pdf

http://campis-fab.blogspot.de/2011/09/ein-kleines-openscad-tutorial.html

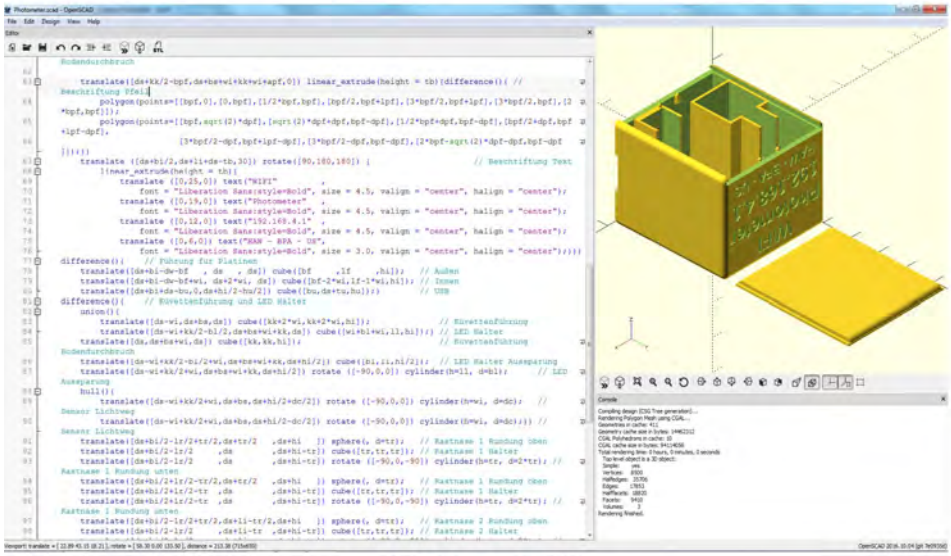


Abbildung 25: Gehäuseentwurf in Open SCAD

Die in den Abbildungen gezeigten Gehäuse wurden aus dem Kunststoff PLA mit einer Schichthöhe von 0.2 mm gedruckt. PLA (Polylactide, umgangssprachlich auch Polymilchsäure genannt) ist sicherlich nicht der beste Kunststoff für den Einsatz in einem Umfeld, in dem Chemikalien eine Rolle spielen können. PLA ist aber sehr einfach zu verarbeiten, gilt als biologisch abbaubarer Kunststoff, kann bis ca 50 °C Umgebungstemperatur verwendet werden und erzeugt auch beim Druckvorgang keine gesundheitsgefährdenden Dämpfe.

8 Inbetriebnahme

Für einen ersten Test muss die Powerbank oder das Netzteil an den USB-Anschluss des Fotometers angeschlossen werden. Die eingebaute LED sollte jetzt für ca 5 Sekunden leuchten und danach wieder ausgehen.

Abbildung 26: Fertig zum Ausprobieren

9 Benutzung

Das Fotometer baut ein eigenes WLAN/WIFI Netzwerk auf. Um sich mit diesem Netzwerk zu verbinden, muss am Smartphone zunächst eine Liste der verfügbaren WLAN/WIFI-Netzwerke aufgerufen werden. Nach Antippen des Eintrags „HAW-Photometer-...“ (statt der Punkte steht die Seriennummer des Fotometers) verbindet sich das Smartphone mit dem Fotometer. Ein Password wird nicht abgefragt.

Abbildung 27: Liste der verfügbaren WLAN/WIFI Netzwerke

Abbildung 28: WLAN/WIFI Netzwerk auswählen

Abbildung 29: Ggf. Netzwerk ohne Internet zulassen

Einige Smartphones reagieren noch mit einer extra Nachfrage, ob eine Verbindung mit einem Netzwerk ohne Internetverbindung gewünscht ist. Für das Fotometer ist „ohne Internet“ ein Teil des Sicherheitskonzepts und somit ausdrücklich gewünscht.

Im Webbrowser des Smartphones kann jetzt die Webseite des Fotometers aufgerufen werden. Hierzu wird in die Adresszeile des Webbrowsers die IP-Adresse: 192.168.4.1 eingeben und die Webseite abgerufen.

Es wird eine Webseite mit Bedienköpfen und einer Tabelle dargestellt.

Abbildung 30: Im Webbrowser Webseite mit IP-Adresse 192.168.4.1 aufrufen

Abbildung 31: Drücken von „Leerprobe“ führt eine Messung aus und trägt die Daten in die Tabelle ein

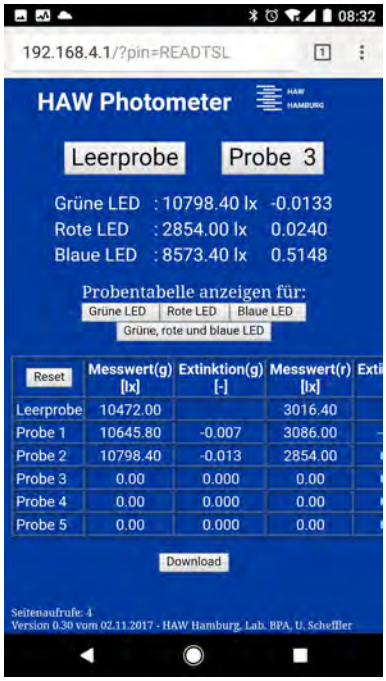


Abbildung 32: Drücken von „Probe“ misst, berechnet die Extinktion und ergänzt die Tabelle

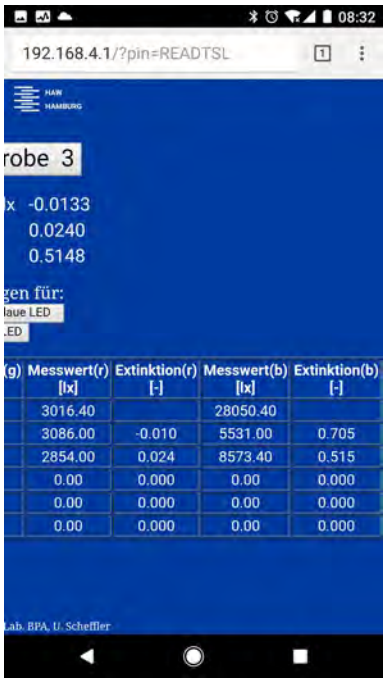


Abbildung 33: Die Tabelle ist breiter und die Anzeige kann seitlich verschoben werden

Abbildung 34: Auch Zoomen ist möglich

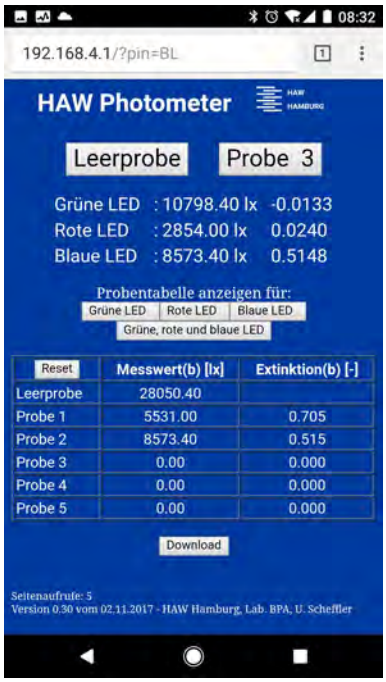


Abbildung 35: Durch Drücken von einem der „... LED „Buttons wird nur der jeweilige Ausschnitt der Tabelle angezeigt

Der Button „Download“ bewirkt ein Herunterladen einer CSV-Datei mit den Daten der Tabelle. Eine CSV-Datei kann in den meisten Tabellenkalkulationsprogrammen einfach weiterverarbeitet werden.

Für die einfarbige Modellvariante fehlen die Auswahlbuttons für die LED Farben.

10 Programmcode

Der Programmcode für das Fotometer mit dem D1 mini wurde in der Arduino IDE erstellt und umfasst ca 480 Zeilen. Es ist im Anhang unter 11.3 Programmcode ab Seite 26 zu finden und kann auch beim Autor via email angefragt werden. Im Folgenden eine kurze Beschreibung, um die Orientierung im Code zu erleichtern und eigene Anpassungen zu erleichtern.

Gleich zu Beginn kann durch Auskommentieren bzw. Kommentieren gewählt werden, ob eine Version für eine einfarbige LED oder eine dreifarbige LED erzeugt werden soll.

Soll eine dreifarbige LED verwendet werden müssen in Zeile 11 die beiden Schrägstiche („//“) am Anfang der Zeile entfernt werden und in Zeile 12 müssen am beginn der Zeile zwei Schrägstriche eingefügt werden. Die beiden Schrägstriche bedeuten, dass die entsprechende Zeile nur Kommentare enthält und nicht als Programmcode gelesen werden sollen.

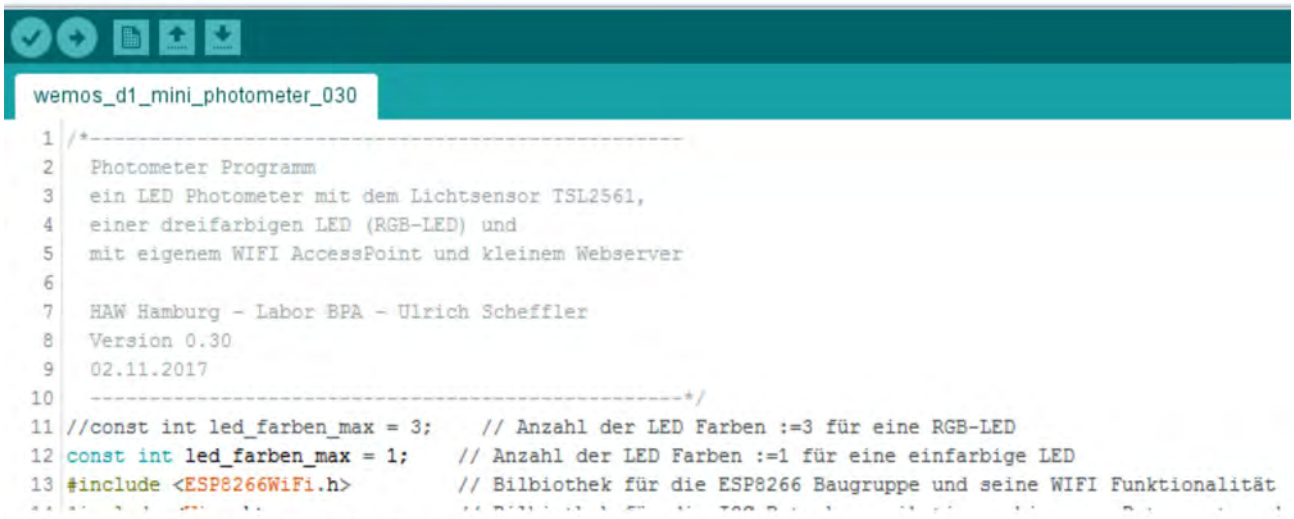


Abbildung 36: Progammcode Ausschnitt: Ein- oder dreifarbige LED

Arduino typisch gliedert der Programmcode sich in einen Setup-Teil und einen Loop-Teil.

Der Setup-Teil „void setup()“ wird einmalig beim Programmstart ausgeführt. In ihm werden der WIFI-Accesspoint und der Web Server mit Startparametern versorgt und initialisiert. Zudem wird die LED für 6 Sekunden eingeschaltet, um dem Benutzer die Betriebsbereitschaft zu signalisieren.

Der Hauptteil des Programms ist die Endlosschleife „void loop()“. Hier wird zunächst geprüft, ob ein Teilnehmer via WIFI verbunden ist und ob eine gültige Anfrage – also eine Anforderung durch den Webbrowser des Benutzers – an den Webserver gestellt wurde. Ist dies der Fall, wird die Anfrage genauer untersucht und in Abhängigkeit dieser Untersuchung begonnen eine Antwort – also eine neue Webseite – des Webservers zusammen zu stellen.

Wurde eine ungültige Anfrage an den Webserver gestellt, wird eine Fehlerseite als Antwort erzeugt.

War die Anfrage gültig, wird noch unterschieden, ob der Bedienknopf „Leerprobe“ oder „Probe“ gedrückt wurde. In beiden Fällen wird eine Messung durchgeführt. Das bedeutet es wird die LED eingeschaltet und Messdaten vom Sensor abgerufen. Die Messung wird siebenmal wiederholt, um ggf. Messfehler und Ausreißer zu unterdrücken – der kleinste und der größte Messwert werden aussortiert und aus den verbleibenden fünf Messwerten wird ein Mittelwert gebildet. Wurde eine Probe vermessen und es stand schon eine Messung einer Leerprobe zur Verfügung wird sodann die Extinktion berechnet.

Alle gemessenen und berechneten Werte werden dann in Textstrings eingebunden aus denen der Antwort-String – im Programmcode „sResponse“ – des Webservers zusammengesetzt wird. Danach wird die Antwort des Webservers d. h. die komplette Beschreibung der Webseite „ausgeliefert“ und die Endlosschleife kehrt an ihren Anfang zur Bearbeitung einer möglichen nächsten Anfrage zurück.

10.1 Benötigte Bibliotheken

Um erweiterte Funktionalitäten einzubinden, werden im Programmcode vier zusätzliche Bibliotheken verwendet. „Wire“ ist Bestandteil der Arduino IDE und wird für die digitale Kommunikationsverbindung mit Zusatzkomponenten – hier der Lichtsensor – benötigt. „ESP8266WiFi“ stammt aus dem installierten ESP8266-Zusatzpaket für die Arduino IDE und stellt die WLAN/WIFI Funktionalitäten zur Verfügung. Die Lizenzbedingungen, auch die der Teilkomponenten, finden sich unter: https://github.com/esp8266/Arduino. Die beiden Bibliotheken „Adafruit_Sensor“ und „Adafruit_TSL2561“ stammen von der Firma Adafruit und stellen die Funktionalität für den Lichtsensor zur Verfügung. Ihre Lizenzbedingungen finden sich in den Headerfiles.

Hier gibt es Informationen zu den beiden Bibliotheken (auf Englisch):

https://github.com/adafruit/Adafruit_TSL2561/

https://github.com/adafruit/Adafruit_Sensor/

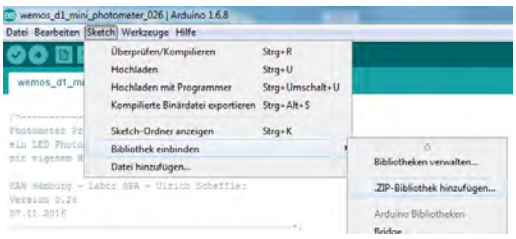
Hier können sie heruntergeladen werden:

https://github.com/adafruit/Adafruit_TSL2561/archive/master.zip

https://github.com/adafruit/Adafruit_Sensor/archive/master.zip

Um die Adafruit-Bibliotheken für die Arduino IDE verfügbar zu machen, müssen sie an einer bestimmten Stelle im Verzeichnissystem des PCs hinterlegt werden. Am einfachsten gelingt dies folgender maßen:

1. Die benötigten Bibliotheken als ZIP-Files herunterladen. Achtung: die ZIP-Files nicht auspacken, sondern so lassen!
2. In der Arduino IDE im Menü „Sketch“ das Untermenü „Bibliothek einbinden“ anwählen. Im aufklappenden Untermenü den Punkt „.ZIP-Biblothek hinzufügen...“ auswählen.



   Daraufhin erscheint ein File-Auswahl-Dialog.
3. Im File-Auswahl-Dialog die gewünschte ZIP-Datei auswählen und „Öffnen“ anklicken.
4. Diese Prozedur (1. bis 3.) für alle gewünschten Bibliotheken wiederholen.
5. Die Arduino IDE beenden und erneut starten.

Die Bibliotheken sind jetzt aus ihren ZIP-Files ausgepackt worden und an der richtigen Stelle im Filesystem für die Verwendung im Fotometer-Programm hinterlegt worden.

11 Anhang

11.1 Aktuelle Anmerkungen und Ergänzungen

Webseite wir nur langsam, unvollständig oder gar nicht dartgestellt

Mittlerweile funktionieren auch aktuellere Arduino IDE Versionen (1.8.5 haben wir getestet) mit dem ESP8266-Boards. Aber die zusätzlich benötigten ESP8266 Board-Addons machen mit ihren letzten Revisionen (ab 2.4.1) noch Schwierigkeiten. Die Version 2.4.0 funktioniert aber prima.

Wie aktiviert man eine bestimmte Version des Board Addons? Das ESP8266-Addon muss wie in der Anleitung beschrieben installiert (ab Seite 7) werden (JSON- String „eintragen“), dann im Boardmanger den Eintrag für den ESP8266 wiederfinden. Jetzt nicht die letzte Version, sondern die Version 2.4.0 auswählen und erst dann „installieren“ auswählen. Hinweis: Das Auswählen einer bestimmten Version ist auch nachträglich möglich.

WIFI Verbindungsaufbau

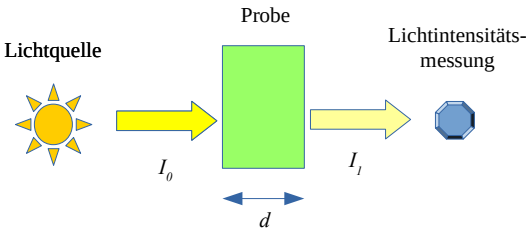
Mit der WIFI Funktionalität gab es bisher keine Schwierigkeiten. Aber da das vom Fotometer aufgebaute WIFI Netz kein Internet anbietet, verhalten sich einige Clients (Handys, Tablets, ...) etwas zickig. Insbesondere Apple Geräte versuchen recht lange doch noch eine Internet-Verbindung herzustellen und erst dann, wenn sie aufgeben, stellen sie dieses Netzwerk auch als VERBUNDEN dem Nutzer zur Verfügung. Dieser Vorgang kann auch gerne mal über eine Minute dauern (halbe Ewigkeit).

3D-Druck des Gehäuses

Tipp für den 3D-Druck des Gehäuses: Vom Lichtsensor-Modul gibt es verschiedene Bauformen. Wenn das Sensormodul bei Aliexpress oder bei „Roboter-Bausatz.de“ gekauft worden ist, ist es vermutlich mit der Typnummer GY-2561 gekennzeichnet. Dieses Modul ist sehr gut geeignet, weil es besonders geringe Abmessungen hat. Aber es gibt zwei verschieden Ausführungen: eine mit zwei großen Löchern und eine mit drei großen Löchern in der Platine. Die Löcher sind für das Fotometer nicht entscheidend aber die Position des Lichtsensors auf dem Modul ist leicht unterschiedlich. Im Sourcecode für das Gehäuse sind beide Versionen (2 Löcher und 3 Löcher) zu finden. Die jeweils nicht benötigten Zeilen müssen deaktiviert werden. Auch das Lichtsensor-Modul von Adafruit, das in Deutschland sehr einfach zu bekommen ist, ist geeignet. Es ist jedoch deutlich größer und auch hier muss das Gehäuse entsprechend angepasst werden (siehe Gehäuse-Sourcecode). Extra-Hinweis: Das Adafruit Modul hat auch ein anderes Pin-Layout, sodass auch die selbst zu lötende Adapterplatine etwas anders ausfällt. Auch die Verwendung von Rundküvetten (bzw. Reagenzgläsern) ist möglich. Hierfür sind die benötigten Werte und Programmzeilen im Sourcecode des Gehäusemodells zu finden.

11.2 Lambert-Beer‘sches Gesetz

Das Lambert-Beer'sche Gesetz oder Bouguer-Lambert-Beer'sche Gesetz beschreibt die Abschwächung der Intensität des Lichts beim Durchgang durch ein Medium mit einer absorbierenden Substanz unter Berücksichtigung der Konzentration der absorbierenden Substanz und ihrer Schichtdicke.



$$K = \frac{E_\lambda}{c_B}$$

$$E_\lambda = lg\left(\frac{I_0}{I_1}\right) = \varepsilon_\lambda \cdot c \cdot d$$

E_λ := Extinktion bei der Wellenlänge λ (Ohne Einheit)

I_0 := Intensität des Lichtes vor der oder ohne die Probe (Einheit: $W \cdot m^{-2}$)

I_1 := Intensität des Lichtes nach der Probe (Einheit: $W \cdot m^{-2}$)

c := Stoffmengenkonzentration der absorbierenden Substanz in der Flüssigkeit (Einheit: $mol \cdot l^{-1}$)

ε_λ := dekadischer Extinktionskoeffizient bei der Wellenlänge λ. ε_λ ist spezifisch für die absorbierende Substanz. Wird die Konzentration in Mol/l angegeben, so wird ε_λ als dekadischer molarer Extinktionskoeffizient notiert (Einheit: $m^2 \cdot mol^{-1}$).

d := Schichtdicke der Probe Körpers (Einheit: m)

Anwendung in der Fotometrie:

$$E_\lambda = lg\left(\frac{I_0}{I_1}\right) = \varepsilon_\lambda \cdot c \cdot d$$

Setz man voraus, dass sich weder die Schichtdicke d noch die untersuchte Substanz mit ihrem spezifischen Extinktionskoeffizienten ε_λ ändert, ergibt sich ein linearer Zusammenhang mit dem Proportionalitätsfaktor K zwischen Konzentration c und der Extinktion E_λ.

$$E_\lambda = K \cdot c$$

Liegt eine Probe ohne die gesuchte Substanz (Leerprobe) und eine Probe mit bekannter Konzentration c_B vor, kann K über die Gleichung $K = \frac{E_\lambda}{c_B}$ berechnet werden. Mit diesem K können nun für weitere unbekannte Proben die Konzentrationen c nach Messung der Extinktion ε_λ ermittelt werden.

$$c = \frac{E_\lambda}{K}$$

Arbeitsblatt

Name: **Datum:**
LED-Farbe: **Fotometer Nr.**

Bekannte Proben:

Konzen-tration	**Testlösung [µl]**	**Wasser [ml]**	**Messwert I [LUX]**	**Extinktion ε_λ**	**K**
0	0	3,0			
1/30	100	2,9			
1/10	300	2,7			
2/5	600	2,4			
1/3	1000	2,0			
	3000	0,0			

Unbekannte Proben:

Proben Nr.:	**Messwert I [LUX]**	**Extinktion ε_λ**	**Konzentration**

11.3 Programmcode

```
/*-----------------------------------------------

  Photometer Programm
  ein LED Photometer mit dem Lichtsensor TSL2561,
  einer dreifarbigen LED (RGB-LED) und
  mit eigenem WIFI AccessPoint und kleinem Webserver

  HAW Hamburg - Labor BPA - Ulrich Scheffler
  Version 0.30
  02.11.2017
  -----------------------------------------------*/
//const int led_farben_max = 3;    // Anzahl der LED Farben :=3 für eine RGB-LED
const int led_farben_max = 1;    // Anzahl der LED Farben :=1 für eine einfarbige LED
#include <ESP8266WiFi.h>         // Bilbiothek für die ESP8266 Baugruppe und seine WIFI Funktionalität
#include <Wire.h>                // Bilbiothek für die I2C Datenkummunikation - hier zum Datenaustausch mit dem Lichtsensor benötigt
#include <Adafruit_Sensor.h>     // Basis-Bilbiothek für die Adafruit Sensor Biblitheken (Bibliothek von Adafruit)
#include <Adafruit_TSL2561_U.h>  // Bibliothek für den Lichtsensor TSL2661 (Bibliothek von Adafruit)
String sVersion = "Version 0.30 -";
String sVersion2 = " vom 02.11.2017 - HAW Hamburg, Lab. BPA, U. Scheffler";

const char* ssid = "HAW-PHOTOMETER-";       // der erste Teil des Namens des WIFI Access Points
const char* password = "";                  // Auf "" (leerer String) eingestellt für einen offenen Access Point ohne Passwort
unsigned long ulReqcount;                   // Zähler für die Aufrufe der Webseite
int ledPin[] = {13, 12, 14};                // grüne LED an GPIO Pin 13/D7, rote LED an GPIO Pin 12/D6, blaue LED an GPIO Pin 14/D5
(hier für Lucky Light: LL-509RGBC2E-006)
const String farbkennung[] = {"Gr&uumlne LED  ", "Rote LED  ", "Blaue LED  ", "Gr&uumlne, rote und blaue LED"}; // Texte
für die RGB-Auswahl Buttons (Grün, Rot, Blau)
const int sdaPin =  4;                      // SDA an GPIO/Pin  4 / D2   Anschluss-Pin für das SDA-Signal zur Datenkommunikation mit
dem Lichtsensor
const int sclPin =  5;                      // SCL an GPIO/Pin  5 / D1   Anschluss-Pin für das SCL-Signal zur Datenkommunikation mit
dem Lichtsensor
const int data_runs  = 7;                   // Anzahl der Wiederholungsmessungen aus denen dann ein Mittelwert gebildet wird
float VIS_IRZEROdata[] = {0.0, 0.0, 0.0};   // Für die aktuellen Nullproben-Messwerte vom Lichtsensor.  Hier der Messwert vom Sensor-
teil, der im VIS und IR Bereich misst
float IRZEROdata[]     = {0.0, 0.0, 0.0};   // Für die aktuellen Nullproben-Messwerte vom Lichtsensor.  Hier der Messwert vom Sensor-
teil, der nur im IR Bereich misst
float LUXZEROdata[]    = {0.0, 0.0, 0.0};   // Für die aktuellen Nullproben-Messwerte vom Lichtsensor.  Hier die laut Datenblatt
berechnete Beleuchtungsstärke in Lux
float VIS_IRdata[]     = {0.0, 0.0, 0.0};   // Für die aktuellen Proben-Messwerte vom Lichtsensor.  Hier der Messwert vom Sensorteil,
der im VIS und IR Bereich misst
float IRdata[]         = {0.0, 0.0, 0.0};   // Für die aktuellen Proben-Messwerte vom Lichtsensor.  Hier der Messwert vom Sensorteil,
der nur im IR Bereich misst
float LUXdata[]        = {0.0, 0.0, 0.0};   // Für die aktuellen Proben-Messwerte vom Lichtsensor.  Hier die laut Datenblatt berech-
nete Beleuchtungsstärke in Lux
float E_LUX[]          = {0.0, 0.0, 0.0};   // Für die berechnete Extiktionen aus den Beleuchtungsstärke-Daten
float E_VIS_IR[]       = {0.0, 0.0, 0.0};   // Für die berechnete Extiktionen aus den Sensor-Daten vom Sensorteil der im VIS und IR
Bereich misst
float E_IR[]           = {0.0, 0.0, 0.0};   // Für die berechnete Extiktionen aus den Sensor-Daten vom Sensorteil der nur im IR
Bereich misst
int probenzeilenIndex     = 0;              // Zeiger auf die aktuell zu füllende Probenzeile
const int probenzeilenMax = 5;              // Anzahl der Zeilen für Proben in der HTML Tabelle
float LUX_werte[3][probenzeilenMax];        // Array für Proben-Messwerte(Beleuchtungsstärke-Daten) vom Lichtsensor in Lux
float E_werte[3][probenzeilenMax];          // Array für berechnete Extiktionen aus den Beleuchtungsstärke-Daten
String anzeige            = "a";            // Kennung für die Anzeige der Tabellenanteile a=alle, g=grün, r=rot, b=blau
bool download             = false;          // Flag: Wenn True Download der DatenTabelle gewünscht, wenn False dan nicht.
String datentabelle       = "";             // Nimmt Datentabelle für den Download auf
const String trennzeichen = "\t";           // Spalten Trennzeichen für die Datentabelle (\t := Tabulatorzeichen)
uint16_t broadband = 0;                     // Sensor-Daten vom Sensorteil der im VIS und IR Bereich misst
uint16_t infrared  = 0;                     // Sensor-Daten vom Sensorteil der nur im IR Bereich misst
WiFiServer server(80);                      // Eine Instanz auf dem Server für Port 80 erzeugen
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);  //Objekt anlegen für den TSL2561 Sensor

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void displaySensorDetails(void) { // Zeigt einige Basisinformationen über den Sensor an
  sensor_t sensor;
  tsl.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  //Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  //Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" lux");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" lux");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" lux");
  delay(500);
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void configureSensor(void) { // Verstärkung und Integrationszeit konfigurieren
  // tsl.setGain(TSL2561_GAIN_1X);      // 1-fache Verstärkung      ... bei hoher Beleuchtungsstärke, um eine Sensor-Übersättigung zu
verhindern
  // tsl.setGain(TSL2561_GAIN_16X);     // 16-fache Verstärkung     ... bei niediger Beleuchtungsstärke, um die Sensor-Empfindlich-
keit zu erhöhen
  tsl.enableAutoRange(true);            // Automatische Verstärkung ... Verstärkung wechselt automatisch zwischen 1-fach und 16-fach
  // Das Ändern der Integrationszeit bewirkt eine Änderung der Sensor-Genauigkeit bzw. seiner Auflösung (402ms = 16-bit data)
  tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* 13ms: Schnell aber niedrige Auflösung */
  //tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS);   /* 101ms: Mittlere Geschwindigkeit und Auflösung */
  //tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_402MS);   /* 402ms: Langsamste Geschwindigkeit aber hohe Auflösung (16-bit
Daten)*/
  // Einstellungen zur Info ausgeben
  Serial.print  ("Gain:         "); Serial.println("Auto");
  Serial.print  ("Timing:       "); Serial.println("13 ms");
  //Serial.print  ("Timing:       "); Serial.println("101 ms");
  Serial.println("------------------------------------");
}

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
bool readSensor(int color) {  // Sensor-Messdaten auslesen
  sensors_event_t event;      // Ein neues TSL2561 Lichtsensor Ereigniss einrichten
  int ok = 0;                 // Zähler für geglückte Messungen
  float LUX[data_runs + 1];   // Daten Array zur Aufname der Einzelmessungen (ein Element mehr als Summenspeicher)
  float VIS_IR[data_runs + 1]; // Daten Array zur Aufname der Einzelmessungen
```

```
  float IR[data_runs + 1];    // Daten Array zur Aufname der Einzelmessungen
  float LUX_max = 0.0, LUX_min = 0.0;
  for (int j = 0; j <= data_runs; j++) { // Daten Arrays mit "0.0"-Werten vorbelegen
    LUX[j]    = 0.0;
    VIS_IR[j] = 0.0;
    IR[j]     = 0.0;
  }
  if (led_farben_max > 1) {
    digitalWrite(ledPin[color], HIGH); // LED einschalten
  }
  else {
    digitalWrite(ledPin[1], HIGH); // LED einschalten
  }
  for (int j = 0; j <= data_runs - 1; j++) { // Messungen "data_runs"-mal wiederholen und Einzelmessungen in Daten Array eintragen
    tsl.getEvent(&event);               // Eine Messung durchführen
    if (event.light) {                  // Wird nur dann "Wahr" wenn erfolgreich gemesen wurde
      LUX[j] = (event.light * 1.0);     // LUX-Wert abholen
      if (j == 0) {
        LUX_max = LUX[j];  // Min- und Max-Werte am Anfang auf den ersten Messwert setzten
        LUX_min = LUX[j];
      }
      if (LUX[j] > LUX_max) LUX_max = LUX[j];          // Neuen Max-Wert suchen und ggf neu setzten
      if (LUX[j] < LUX_min) LUX_min = LUX[j];          // Neuen Min-Wert suchen und ggf neu setzten
      tsl.getLuminosity (&broadband, &infrared);       // VIS-IR- und IR-Werte auslesen
      VIS_IR[j] = (broadband * 1.0);                   // VIS-IR-Wert zuweisen
      IR[j]     = (infrared * 1.0);                    // IR-Wert zuweisen
      delay(25);                                       // Zwischen den Messungen warten (Zeit in Milli-Sekunden)
      ok += 1;                                         // Zähler für geglückte Messungen um 1 erhöhen
    }
    else { // Wenn "event.light = 0 lux" ist, dann ist der Sensor möglicher Weise auch gesättigt und es können keinen Daten generiert
werden!
      Serial.println("Der Sensor-Messwert ist unbrauchbar. -> Sensor fehler!");
  } }
  if (led_farben_max > 1) {
    digitalWrite(ledPin[color], LOW); // LED wieder ausschalten
  }
  else {
    digitalWrite(ledPin[1], LOW); // LED wieder ausschalten
  }
  if (ok >= data_runs) {                  // Nur wenn alle Einzelmessungen erfolgreich durchgeführt wurden ...
    for (int j = 0; j <= data_runs - 1; j++) { // Einzelmessungen aufaddieren
      LUX[data_runs]    += LUX[j];
      VIS_IR[data_runs] += VIS_IR[j];
      IR[data_runs]     += IR[j];
    }
    // Die aufaddierte Summe der Einzelwerte ohne den Max-Wert und den Min-Wert geteilt durch Anzahl der Einzelmessungen minus 2
ergibt den bereinigten Mittelwert
    LUX[data_runs]    = (LUX[data_runs] - LUX_max - LUX_min)    / (data_runs * 1.0 - 2.0);
    // Die aufaddierte Summe der Einzelwerte geteilt durch Anzahl der Einzelmessungen ergibt den jeweiligen Mittelwert
    VIS_IR[data_runs] = VIS_IR[data_runs] / (data_runs * 1.0);
    IR[data_runs]     = IR[data_runs]     / (data_runs * 1.0);
    LUXdata[color]    = LUX[data_runs];    // Berechneten LUX-Wert in die Datentabelle übertragen
    VIS_IRdata[color] = VIS_IR[data_runs]; // Berechneten VIS-IR-Wert in die Datentabelle übertragen
    IRdata[color]     = IR[data_runs];     // Berechneten IR-Wert in die Datentabelle übertragen
    return true;
  }
  else {
    return false;
  }
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void cleardata() {
  Serial.println("Cleardata(RESET) aufgerufen!" );
  for (int zeilennummer = 0; zeilennummer < probenzeilenMax; zeilennummer++) { // Datentabelle mit "0.0"-Werten vorbelegen
    for (int ledfarbe = 0; ledfarbe < led_farben_max; ledfarbe++) {
      LUX_werte[ledfarbe][zeilennummer] = 0.0;
      E_werte[ledfarbe][zeilennummer]   = 0.0;
      LUXZEROdata[ledfarbe]             = 0.0;
      VIS_IRZEROdata[ledfarbe]          = 0.0;
      IRZEROdata[ledfarbe]              = 0.0;
    }
  }
  probenzeilenIndex = 0;
}

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
void setup() {
  // Globale Voreinstellungen
  for (int led = 0; led < led_farben_max; led++) {
    if (led_farben_max == 1)  pinMode(ledPin[1]  , OUTPUT);  // Wenn nur eine LED an GPIO Pin 12/D6 (für einfarbige LED), den
Anschluss für die LED als Ausgang konfigurieren
    if (led_farben_max > 1)  pinMode(ledPin[led], OUTPUT);   // Die Anschlüsse für die LEDs als Ausgänge konfigurieren
  }
  ulReqcount = 0;                   // Seitenaufrufszähler auf 0 setzten
  Serial.begin(9600);               // Serielle Verbindung initialisieren
  WiFi.mode(WIFI_AP);               // WIFI Access Point Modus initialisieren
  if (led_farben_max > 1) {
    for (int led = 0; led < led_farben_max; led++) {
      digitalWrite(ledPin[led], HIGH); // LED einschalten
      delay(2000);                     // 2000 ms warten
      digitalWrite(ledPin[led], LOW);  // LED ausschalten
    }
  }
  else {
    digitalWrite(ledPin[1], HIGH); // LED einschalten
    delay(6000);                   // 6000 ms warten
    digitalWrite(ledPin[1], LOW);  // LED ausschalten
  }
  for (int zeilennummer = 0; zeilennummer < probenzeilenMax; zeilennummer++) { // Arrays mit "0.0"-Werten vorbelegen
    for (int ledfarbe = 0; ledfarbe < led_farben_max; ledfarbe++) {
      LUX_werte[ledfarbe][zeilennummer] = 0.0;
      E_werte[ledfarbe][zeilennummer]   = 0.0;
    }
  }
  Serial.println("");
  Serial.println(String(sVersion + String (led_farben_max) + sVersion2));
  // Das Anhängsel für den Access Point Namen (SSID) aus der MAC ID des ESP Moduls generieren.
  // Um den Namen nicht zu lang werden zu lassen, werden hier nur die letzten 2 Bytes verwendet.
```

```
  // Der Aufwand dient dazu, um einen möglichst einmaligen Access Point Namen für das jeweils verwendete ESP Modul entstehten zu las-
sen.
  uint8_t mac[WL_MAC_ADDR_LENGTH];
  WiFi.softAPmacAddress(mac);
  String macID = String((mac[WL_MAC_ADDR_LENGTH - 2] * 256 +  mac[WL_MAC_ADDR_LENGTH - 1]), DEC);
  macID.toUpperCase();
  String AP_NameString = ssid + macID;
  char AP_NameChar[AP_NameString.length() + 1];
  memset(AP_NameChar, 0, AP_NameString.length() + 1);
  for (int i = 0; i < AP_NameString.length(); i++) AP_NameChar[i] = AP_NameString.charAt(i);
  // WIFI Access Point und den Web Server starten
  WiFi.softAP(AP_NameChar, password);
  server.begin();
  Serial.print("WIFI Access Point gestartet.   Name (SSID) : "); Serial.println(AP_NameChar);
  Serial.print("WEB-Server erreichbar unter der IP-Adresse): "); Serial.println(WiFi.softAPIP());
  Serial.println("");
  // Lichtsensor TSL2561
  Wire.pins(sdaPin, sclPin);  // Wire.pins(int sda, int scl) // Anschlusspins für das I2C-Protokoll zuordnen
  if (!tsl.begin())  {        // Sensor initialisieren und im Fehlerfall eine Meldung ausgeben
    Serial.print("Ooops, es konnte kein TSL2561-Sensor erkannt werden ...! (Hardware Fehler) Programm Abbruch!!! Reset nötig!!!");
    while (1);
  }
  displaySensorDetails();     // Zeigt einige Basisinformationen über den Sensor an
  configureSensor();          // Verstärkung und Integrationszeit konfigurieren
  Serial.println("");
}

/*****************************************************************************************************************/
void loop() {
  WiFiClient client = server.available(); // Prüfen ob ein WIFI-Client verbunden ist
  if (!client) return;                   // Wenn keiner verbunden ist, wieder zum Anfang. Also warten bis der WIFI-Client Daten
gesendet hat
  Serial.println("Neuer WIFI-Client");    // Jetzt ist ein Client verbunden ...
  unsigned long ultimeout = millis() + 250;
  while (!client.available() && (millis() < ultimeout))delay(1);
  if (millis() > ultimeout) {
    Serial.println("WIFI-Client Fehler: Connection-Time-Out!");
    return;
  }
  String sRequest = client.readStringUntil('\r');  // Die Erste Zeile der Anfrage lesen
  client.flush();
  if (sRequest == "") {  // WIFI-Client stoppen, wenn die Anfrage leer ist
    Serial.println("WIFI-Client Fehler: Leere Anfrage! -> WIFI-Client angehalten");
    client.stop();
    return;
  }
  // "get path"; Das Ende des Pfads ist entweder ein " " oder ein "?"  (Syntax z.B. GET /?pin=SENSOR_A HTTP/1.1)
  String sPath = "", sParam = "", sCmd = "";
  String sGetstart = "GET ";
  int iStart, iEndSpace, iEndQuest;
  iStart = sRequest.indexOf(sGetstart);
  if (iStart >= 0) {
    iStart += +sGetstart.length();
    iEndSpace = sRequest.indexOf(" ", iStart);
    iEndQuest = sRequest.indexOf("?", iStart);
    // Den Pfad und die Parameter isolieren
    if (iEndSpace > 0) {
      if (iEndQuest > 0) { // Pfad und Parameter
        sPath  = sRequest.substring(iStart, iEndQuest);
        sParam = sRequest.substring(iEndQuest, iEndSpace);
      }
      else { // Pfad aber keine Parameter
        sPath  = sRequest.substring(iStart, iEndSpace);
      }
    }
  }
  // Den Befehl isolieren
  if (sParam.length() > 0) {
    int iEqu = sParam.indexOf("=");
    if (iEqu >= 0) {
      sCmd = sParam.substring(iEqu + 1, sParam.length());
      Serial.println(sCmd);
    }
  }
  // Die HTML Seite erzeugen
  String sResponse, sHeader;
  String sResponseStart = "";
  String sResponseTab   = "";
  if (sPath != "/") { // Für unpassenden Pfad eine 404-Fehlerseite generieren
    sResponse = "<html><head><title>404 Not Found</title></head><body><h1>Not Found</h1><p>Die angeforderte Webseite (URL) gibt es
auf diesem Server nicht.</p></body></html>";
    sHeader  = "HTTP/1.1 404 Not found\r\n";
    sHeader += "Content-Length: ";
    sHeader += sResponse.length();
    sHeader += "\r\n Content-Type: text/html\r\n Connection: close\r\n\r\n";
  }
  else {  // Für passenden Pfad ...
    if (sCmd.length() > 0) {  // Auf die Parameter reagieren ...
      Serial.print("Command: "); Serial.println(sCmd);
      if (sCmd.indexOf("READZERO") >= 0) {
        for (int color = 0; color < led_farben_max; color++) {
          if (!readSensor(color)) Serial.println("Sensor Fehler"); // Messung durchführen und Sensordaten auslesen - ggf. Fehler mel-
den
          else {
            LUXZEROdata[color]    = LUXdata[color];
            VIS_IRZEROdata[color] = VIS_IRdata[color];
            IRZEROdata[color]     = IRdata[color];
          }
        }
      }
      if (sCmd.indexOf("READTSL") >= 0)  { // Wenn Button "Probe" gedrückt, ...
        for (int color = 0; color < led_farben_max; color++) {
          if (!readSensor(color)) Serial.println("Sensor Fehler"); // Messung durchführen und Sensordaten auslesen - ggf. Fehler mel-
den
          LUX_werte[color][probenzeilenIndex] =  LUXdata[color];
          // Die Extinktionen berechnen
          if (LUXdata[color] > 0.0 & LUXZEROdata[color] > 0.0) {
            E_LUX[color] = -log10((LUXdata[color] * 1.0) / (LUXZEROdata[color] * 1.0));
          }
          else {
```

```
          E_LUX[color] = 0.0;
        }
        if (VIS_IRdata[color] > 0.0 & VIS_IRZEROdata[color] > 0.0) {
          E_VIS_IR[color] = -log10((VIS_IRdata[color] * 1.0) / (VIS_IRZEROdata[color] * 1.0));
        }
        else {
          E_VIS_IR[color] = 0.0;
        }
        if (IRdata[color] > 0.0 & IRZEROdata[color] > 0.0) {
          E_IR[color] = -log10((IRdata[color] * 1.0) / (IRZEROdata[color] * 1.0));
        }
        else {
          E_IR[color] = 0.0;
        }
      }
      probenzeilenIndex += 1;
      if (probenzeilenIndex >= probenzeilenMax) probenzeilenIndex = 0;
    }

    if (sCmd.indexOf("CLEARDATA") >= 0) {
      cleardata()    ; // Wenn Button "Reset" gedrückt, ...
    }
    if (led_farben_max > 1) {
      if (sCmd.indexOf("GR")       >= 0) {
        anzeige = "g"   ; // Wenn Button "grün" gedrückt, ...
      }
      if (sCmd.indexOf("RT")       >= 0) {
        anzeige = "r"   ; // Wenn Button "rot" gedrückt, ...
      }
      if (sCmd.indexOf("BL")       >= 0) {
        anzeige = "b"   ; // Wenn Button "blau" gedrückt, ...
      }
      if (sCmd.indexOf("ALL")      >= 0) {
        anzeige = "a"   ; // Wenn Button "alle" gedrückt, ...
      }
    }
    if (sCmd.indexOf("DOWNLOAD") >= 0) {
      download = true; // Wenn Button "Download" gedrückt, ...
    }
  }
  // Die Extinktion mit ggf neuer Leerprobe erneut berechnen
  for (int color = 0; color < led_farben_max; color++) {
    for (int zeilennummer = 0; zeilennummer < probenzeilenMax; zeilennummer++) {
      if (LUX_werte[color][zeilennummer] > 0.0 & LUXZEROdata[color] > 0.0) {
        E_werte[color][zeilennummer] = -log10((LUX_werte[color][zeilennummer] * 1.0) / (LUXZEROdata[color] * 1.0));
      }
      else {
        E_werte[color][zeilennummer] = 0.0;
      }
    }
  }
  ulReqcount++;  // Seitenaufrufzähler um 1 raufzählen
  sResponseStart  = "<html><head><title>HAW Photometer</title></head><body>";
  sResponseStart += "<font color=\"#FFFFFF\"><body bgcolor=\"#003CA0\">";  // Hintergrundfarbe
  sResponseStart += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=yes\">";
  sResponseStart += "<p style=\"text-align: center; font-family: 'Helvetica', sans-serif; font-size: 27; font-weight:bold; \">HAW
Photometer    ";
  sResponseStart += "<svg xmlns=\"http://www.w3.org/2000/svg\" xml:space=\"preserve\" width=\"27.1229mm\" height=\"7.2751mm\" view-
Box=\"0 0 125324 33615\"  xmlns:xlink=\"http://www.w3.org/1999/xlink\">";
  sResponseStart += "<defs><style type=\"text/css\"><![CDATA[.fil0 {fill:white}.fnt0 {font-weight:bold;font-size:16300.1px;font-
family:'Arial'}]]></style></defs>";
  sResponseStart += "<polygon class=\"fil0\" points=\"0,2484 24461,2484 24461,260 0,260 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"0,11378 24461,11378 24461,9154 0,9154 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"0,20273 24461,20273 24461,18049 0,18049 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"0,29168 24461,29168 24461,26945 0,26945 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"8896,6931 33357,6931 33357,4707 8896,4707 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"8896,15826 33357,15826 33357,13602 8896,13602 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"8896,24721 33357,24721 33357,22497 8896,22497 \"/>";
  sResponseStart += "<polygon class=\"fil0\" points=\"8896,33615 33357,33615 33357,31392 8896,31392 \"/>";
  sResponseStart += "<text x=\"41201\" y=\"11671\"  class=\"fil0 fnt0\">HAW</text><text x=\"41201\" y=\"30498\"  class=\"fil0
fnt0\">HAMBURG</text>";
  sResponseStart += "</svg>";
  sResponseStart += "</p>";
  sResponseStart += "<p style='text-align: center'><a href=\"?pin=READZERO\"><button style='font-size:26px'>Leerprobe</button></
a>";
  sResponseStart += String("        <a href=\"?pin=READTSL\"> <button style='font-size:26px'>Probe  " +
String(probenzeilenIndex + 1) + "</button></a></p>");
  sResponseStart += "<p style='text-align: center'><table border='0' style=\"font-family: 'Helvetica', sans-serif; color: white;
margin-left: auto; margin-right: auto; font-size: 20 \">";
  for (int color = 0; color < led_farben_max; color++) {
    if (led_farben_max > 1) {
      sResponseStart += String("<tr><td>" + farbkennung[color] + "</td><td>:</td><td>" + String(LUXdata[color]) + " lx  </
td><td>" + String(E_LUX[color], 4) + "</td></tr>");
    }
    else {
      sResponseStart += String("<tr><td>" + String(LUXdata[color]) + " lx  </td><td>" + String(E_LUX[color], 4) + "</td></
tr>");
    }
  }
  sResponseStart += "</table></p>";
  sResponseStart += "<p style='text-align: center'>";
  sResponseStart += "Probentabelle anzeigen f&uumlr:<BR>";
  if (led_farben_max > 1) {
    sResponseStart += String("<a href=\"?pin=GR\"><button>" + farbkennung[0] + "</button></a>");
    sResponseStart += String("<a href=\"?pin=RT\"><button>" + farbkennung[1] + "</button></a>");
    sResponseStart += String("<a href=\"?pin=BL\"><button>" + farbkennung[2] + "</button></a>");
    sResponseStart += String("<a href=\"?pin=ALL\"><button>" + farbkennung[3] + "</button></a>");
  }
  sResponseStart += "</p>";
  if (led_farben_max > 1) {
    datentabelle = String(" " + trennzeichen + "(g)" + trennzeichen + "(g)" + trennzeichen + "(r)" + trennzeichen + "(r)" + trenn-
zeichen + "(b)" + trennzeichen + "(b)\r\n");
    datentabelle += String(" "  + trennzeichen + "M" + trennzeichen + "E" + trennzeichen + "M" + trennzeichen + "E" + trennzeichen
+ "M" + trennzeichen + "E\r\n");
    datentabelle += String(" " + trennzeichen + "[lx]" + trennzeichen + "[-]" + trennzeichen + "[lx]" + trennzeichen + "[-]" +
trennzeichen + "[lx]" + trennzeichen + "[-]\r\n");
    datentabelle += String("Leerprobe"  + trennzeichen + String(LUXZEROdata[0], 2)  + trennzeichen + "-" + trennzeichen +
String(LUXZEROdata[1], 2)  + trennzeichen + "-" + trennzeichen + String(LUXZEROdata[2], 2)  + trennzeichen +  "-\r\n");
    for (int i = 0; i < probenzeilenMax; i++) {
      datentabelle += ("Probe " + String(i + 1));
```

```
        datentabelle += (trennzeichen +  (String(LUX_werte[0][i], 2)) + trennzeichen + (String(String(E_werte[0][i], 3))));
        datentabelle += (trennzeichen +  (String(LUX_werte[1][i], 2)) + trennzeichen + (String(String(E_werte[1][i], 3))));
        datentabelle += (trennzeichen +  (String(LUX_werte[2][i], 2)) + trennzeichen + (String(String(E_werte[2][i], 3))));
        datentabelle += "\r\n";
      }
    }
    else {
      datentabelle =  String(" "  + trennzeichen + "M" + trennzeichen + "E\r\n");
      datentabelle += String(" " + trennzeichen + "[lx]" + trennzeichen + "[-]\r\n");
      datentabelle += String("Leerprobe"  + trennzeichen + String(LUXZEROdata[0], 2)  + trennzeichen + "-\r\n");
      for (int i = 0; i < probenzeilenMax; i++) {
        datentabelle += ("Probe " + String(i + 1));
        datentabelle += (trennzeichen +  (String(LUX_werte[0][i], 2)) + trennzeichen + (String(String(E_werte[0][i], 3))));
        datentabelle += "\r\n";
      }
    }
    Serial.println(datentabelle);
    if (led_farben_max == 1) {
      anzeige = "g"; // Wenn nur eine einfabige LED verbaut ist, wird immer nur die reduzierte Tabelle angezeigt
    }
    sResponseTab += "<table border='1' width='100%' style=\"font-family: 'Helvetica', sans-serif; color: white;\"><tr><th><a href=\"?
pin=CLEARDATA\"> <button>Reset</button></a></th>";
    if (anzeige == "a" or anzeige == "g") sResponseTab += "<th>Messwert(g) [lx]</th> <th>Extinktion(g) [-]</th>";
    if (anzeige == "a" or anzeige == "r") sResponseTab += "<th>Messwert(r) [lx]</th> <th>Extinktion(r) [-]</th>";
    if (anzeige == "a" or anzeige == "b") sResponseTab += "<th>Messwert(b) [lx]</th> <th>Extinktion(b) [-]</th>";
    sResponseTab += "</tr>";
    sResponseTab += "<tr><td>Leerprobe</td>";
    if (anzeige == "a" or anzeige == "g") {
      sResponseTab += String("<td style='text-align: center'>" + String(LUXZEROdata[0]) + "</td><td></td>");
    }
    if (anzeige == "a" or anzeige == "r") {
      sResponseTab += String("<td style='text-align: center'>" + String(LUXZEROdata[1]) + "</td><td></td>");
    }
    if (anzeige == "a" or anzeige == "b") {
      sResponseTab += String("<td style='text-align: center'>" + String(LUXZEROdata[2]) + "</td><td></td>");
    }
    sResponseTab += "</tr>";
    for (int i = 0; i < probenzeilenMax; i++) {
      sResponseTab += "<tr>";
      sResponseTab += String("<td>Probe " + String(i + 1) + "</td>");
      if (anzeige == "a" or anzeige == "g") {
        sResponseTab += String("<td style='text-align: center'>" + String(LUX_werte[0][i], 2) + "</td>");
        sResponseTab += String("<td style='text-align: center'>" + String(E_werte[0][i]  , 3) + "</td>");
      }
      if (anzeige == "a" or anzeige == "r") {
        sResponseTab += String("<td style='text-align: center'>" + String(LUX_werte[1][i], 2) + "</td>");
        sResponseTab += String("<td style='text-align: center'>" + String(E_werte[1][i]  , 3) + "</td>");
      }
      if (anzeige == "a" or anzeige == "b") {
        sResponseTab += String("<td style='text-align: center'>" + String(LUX_werte[2][i], 2) + "</td>");
        sResponseTab += String("<td style='text-align: center'>" + String(E_werte[2][i]  , 3) + "</td>");
      }
      sResponseTab += "</tr>";
    }
    sResponseTab += "</table>";
    sResponseTab += "<p style='text-align: center'><a href=\"?pin=DOWNLOAD\"><button>Download</button></a></p><BR>";
    sResponse += String("<FONT SIZE=-2>Seitenaufrufe: " + String(ulReqcount) + "<BR>"); // Aufrufzähler anzeigen
    sResponse += String(sVersion + String (led_farben_max) + sVersion2 + "<BR>");       // Versionshinweis anzeigen
    sResponse += "</body></html>";

    sHeader  = "HTTP/1.1 200 OK\r\n Content-Length: ";
    sHeader += sResponse.length() + sResponseStart.length() + sResponseTab.length();
    sHeader += "\r\n Content-Type: text/html\r\n Connection: close\r\n\r\n";
  }
  /* Die Antwort an den WIFI-Client senden */
  if (download) { // Wenn der Downloadbutton gedrückt wurde, soll eine Datei gestreamt werden ...
    download = false;
    sResponse  = "HTTP/1.1 200 OK\r\n";
    sResponse += "Content-Type: text/csv; charset=utf-8 \r\n";
    sResponse += "Content-Transfer-Encoding: binary \r\n";
    sResponse += "Content-Disposition: attachment; filename=\"photometer_daten.csv\" \r\n";
    sResponse += "Pragma: no-cache Expires: 0 \r\n";
    sResponse += "Content-Length:";
    sResponse += String(datentabelle.length());
    sResponse += " \r\n";
    sResponse += " Connection: close";
    sResponse += "\r\n\r\n";
    sResponse += datentabelle;
    client.print(sResponse);   // Die Antwort an den WIFI-Client senden
    delay(1000);               // 1000 ms warten
  }
  else {  // Die HTML Seite ausgeben
    client.print(sHeader);
    client.print(sResponseStart);
    client.print(sResponseTab);
    client.print(sResponse);
  }
  // und den WIFI-Client stoppen
  client.stop();
  Serial.println("WIFI-Client getrennt");
}
```

11.4 Gehäusemodell-Code für OpenSCAD

Anmerkung: Das Gehäuse ist passend für das Lichtsensormodul GY-2561. Wird ein anderes Lichtsensormodul verwendet, muss der Entwurf angepasst werden.

```
// Gehäuse für Photometer mit WeMos D1 mini
// U. Scheffler 09.01.2017

$fn=50;
//Innenmaße
ds =  1.6; // Dicke Seitenwand
bi = 35.0; // Breite innen
li = 36.0; // Länge innen
wi =  0.8; // Wandstärke Innen
hi = 26.0; // Höhe innen
kr =  1.6; // Kanten Radius
//Deckel
dd =  2.0; // Dicke Deckel
df =  0.8; // Deckel Führungsnut
fl =  0.2; // Freilauf für Deckelführung
//Rastnasen
tr = 1.2; // Tiefe der Rastnase
lr = 8.0; // Länge der Rastnase
//USB Anschluss
bu =  8.0; // Breite USB
hu = 11.0; // Höhe USB
tu =  0.4; // Zusätzliche Tiefe USB
//Küvette
kk = 13.2; // Kantenlänge für Küvettenaussparung
//Sensor
bs =  5.0; // Einbaubreite Lichtsensor
dc =  3.0; // Duchmesser Sensor Chip
// Führung für Platinen
dw =  5.0; // Dicke für WeMos D1 mini
bf = 10.0; // Breite Führungsblock
lf =  8.0; // Länge Führungsblock
duw=  3.0; // Dicke unter WiMos D1 mini
afu= 30.0; // Abstand der Führung unter der Platine zur USB Seite
// LED Führung
bl =  5.5; // Breite des Ausschitts für die LED
ll =  7.5; // Länge der Auflagefläche für die LED
// Beschriftungen
tb =  0.4; // Einprägetiefe
bpf=  4.0; // Breite Pfeilschaft
lpf= 10.0; // Länge Pfeilschaft
dpf=  0.5; // Breite der Prägelinie
apf=  1.0; // Abstand Pfeil zum Küvettendurchbruch

module roundcube(x,y,z,r){  // "cube" mit abgerundeten Kanten und Radius = r
    hull(){
        translate([  r,  r,  r]) sphere(r=r);
        translate([x-r,  r,  r]) sphere(r=r);
        translate([  r,y-r,  r]) sphere(r=r);
        translate([x-r,y-r,  r]) sphere(r=r);
        translate([  r,  r,z-r]) sphere(r=r);
        translate([x-r,  r,z-r]) sphere(r=r);
        translate([  r,y-r,z-r]) sphere(r=r);
        translate([x-r,y-r,z-r]) sphere(r=r);}
}

module gehaeuse(){   // Gehäuse
    difference(){
        translate([ 0,  0,  0]) roundcube(ds+bi+ds,ds+li+ds,ds+hi+dd,kr);                // Außenhülle
        translate([ds,ds,ds]) cube([   bi    ,    li    ,    hi+dd]);                   // Innenraum
        translate([ds+bi+ds-bu,0,ds+hi/2-hu/2]) cube([bu,ds+tu,hu]);                    // USB Anschluss
        translate([ds   ,0,ds+hi]) cube([   bi    ,ds+li+ds,dd  ]);                  // Deckelloch
        translate([ds-df,0,ds+hi]) cube([df+bi+df,ds+li+ds,dd/2]);                      // Deckelführung
        translate([ds,ds+bs+wi,0]) cube([kk,kk,ds]);                                    // Küvettenführung Bodendurchbruch
        translate([ds+kk/2-bpf,ds+bs+wi+kk+wi+apf,0]) linear_extrude(height = tb){difference(){ // Beschriftung Pfeil
            polygon(points=[[bpf,0],[0,bpf],[1/2*bpf,bpf],[bpf/2,bpf+lpf],[3*bpf/2,bpf+lpf],[3*bpf/2,bpf],[2*bpf,bpf]]);
            polygon(points=[[bpf,sqrt(2)*dpf],[sqrt(2)*dpf+dpf,bpf-dpf],[1/2*bpf+dpf,bpf-dpf],[bpf/2+dpf,bpf+lpf-dpf],
                            [3*bpf/2-dpf,bpf+lpf-dpf],[3*bpf/2-dpf,bpf-dpf],[2*bpf-sqrt(2)*dpf-dpf,bpf-dpf]]);}}
        translate ([ds+bi/2,ds+li+ds-tb,30]) rotate([90,180,180]) {                     // Beschriftung Text
            linear_extrude(height = tb){
                translate ([0,25,0]) text("WIFI"         ,
                    font = "Liberation Sans:style=Bold", size = 4.5, valign = "center", halign = "center");
                translate ([0,19,0]) text("Photometer"   ,
                    font = "Liberation Sans:style=Bold", size = 4.5, valign = "center", halign = "center");
                translate ([0,12,0]) text("192.168.4.1"  ,
                    font = "Liberation Sans:style=Bold", size = 4.5, valign = "center", halign = "center");
                translate ([0,6,0]) text("HAW - BPA - US",
                    font = "Liberation Sans:style=Bold", size = 3.0, valign = "center", halign = "center");}}}
    difference(){    // Führung für Platinen
        translate([ds+bi-dw-bf   , ds   , ds]) cube([bf     ,lf     ,hi]);   // Außen
        translate([ds+bi-dw-bf+wi, ds+2*wi, ds]) cube([bf-2*wi,lf-1*wi,hi]); // Innen
        translate([ds+bi+ds-bu,0,ds+hi/2-hu/2]) cube([bu,ds+tu,hu]);         // USB
    }
    difference(){    // Küvettenführung und LED Halter
        union(){
            translate([ds-wi,ds+bs,ds]) cube([kk+2*wi,kk+2*wi,hi]);              // Küvettenführung
            translate([ds-wi+kk/2-bl/2,ds+bs+wi+kk,ds]) cube([wi+bl+wi,ll,hi]);  // LED Halter
        }
        translate([ds,ds+bs+wi,ds]) cube([kk,kk,hi]);                            // Küvettenführung Bodendurchbruch
        translate([ds-wi+kk/2-bl/2+wi,ds+bs+wi+kk,ds+hi/2]) cube([bl,ll,hi/2]);  // LED Halter Aussparung
        translate([ds-wi+kk/2+wi,ds+bs+wi+kk,ds+hi/2]) rotate ([-90,0,0]) cylinder(h=ll, d=bl);      // LED Aussparung
        hull(){
            translate([ds-wi+kk/2+wi,ds+bs,ds+hi/2+dc/2]) rotate ([-90,0,0]) cylinder(h=wi, d=dc);   // Sensor Lichtweg
            translate([ds-wi+kk/2+wi,ds+bs,ds+hi/2-dc/2]) rotate ([-90,0,0]) cylinder(h=wi, d=dc);   // Sensor Lichtweg
        }
    }
    translate([ds+bi-duw,ds+afu,ds]) cube([duw,wi,hi]);// Platinenführung
    translate([ds+bi/2-lr/2+tr/2,ds+tr/2   ,ds+hi   ]) sphere(, d=tr);   // Rastnase 1 Rundung oben
    translate([ds+bi/2-lr/2     ,ds        ,ds+hi-tr]) cube([tr,tr,tr]); // Rastnase 1 Halter
    translate([ds+bi/2-lr/2     ,ds        ,ds+hi-tr]) rotate ([-90,0,-90]) cylinder(h=tr, d=2*tr); // Rastnase 1 Rundung unten
```

```
    translate([ds+bi/2+lr/2-tr/2,ds+tr/2   ,ds+hi   ]) sphere(, d=tr);   // Rastnase 1 Rundung oben
    translate([ds+bi/2+lr/2-tr  ,ds        ,ds+hi-tr]) cube([tr,tr,tr]); // Rastnase 1 Halter
    translate([ds+bi/2+lr/2-tr  ,ds        ,ds+hi-tr]) rotate ([-90,0,-90]) cylinder(h=tr, d=2*tr); // Rastnase 1 Rundung unten
    translate([ds+bi/2-lr/2+tr/2,ds+li-tr/2,ds+hi   ]) sphere(, d=tr);   // Rastnase 2 Rundung oben
    translate([ds+bi/2-lr/2     ,ds+li-tr  ,ds+hi-tr]) cube([tr,tr,tr]); // Rastnase 2 Halter
    translate([ds+bi/2-lr/2     ,ds+li     ,ds+hi-tr]) rotate ([-90,0,-90]) cylinder(h=tr, d=2*tr); // Rastnase 2 Rundung unten
    translate([ds+bi/2+lr/2-tr/2,ds+li-tr/2,ds+hi   ]) sphere(, d=tr);   // Rastnase 2 Rundung oben
    translate([ds+bi/2+lr/2-tr  ,ds+li-tr  ,ds+hi-tr]) cube([tr,tr,tr]); // Rastnase 2 Halter
    translate([ds+bi/2+lr/2-tr  ,ds+li     ,ds+hi-tr]) rotate ([-90,0,-90]) cylinder(h=tr, d=2*tr); // Rastnase 2 Rundung unten
}

module deckel(){   // Deckel
    difference(){
        union(){
            translate([ds+fl,0,0]) cube([bi-2*fl,ds+li+ds,dd-kr  ]);        // Deckel
            difference(){                                                   // mit Abrundung an Kante
                hull(){
                    translate([ds+fl,0+kr       ,dd-kr]) rotate([0,90,0]) cylinder(h=bi-2*fl, r=kr);
                    translate([ds+fl,ds+li+ds-kr,dd-kr]) rotate([0,90,0]) cylinder(h=bi-2*fl, r=kr);}
                translate([ds-df+fl-20,0-20,-20]) cube([df+bi+df-2*fl+40,ds+li+ds+40,dd/2-2*fl+20]);}
        translate([ds-df+fl,0,0]) cube([df+bi+df-2*fl,ds+li+ds,dd/2-2*fl]);}// Deckelführung
        translate([ds+bi/2-lr/2-fl,ds      ,0]) cube([lr+2*fl,tr,tr]);      // Aussparung Rastnase
        translate([ds+bi/2-lr/2-fl,ds+li-tr,0]) cube([lr+2*fl,tr,tr]);}     // Aussparung Rastnase
}

gehaeuse();
translate([0,ds+li+ds+3,0]) deckel();
```