Prototypen-Entwicklung eines neuartigen, selbstjustierenden, „look-out“ Lichtsensors

Masterarbeit

MA703

durchgeführt von

Gerd Zeidler BSc

Institut für Elektronik

der Technischen Universität Graz

Betreuer: Ass.-Prof. Dipl.-Ing. Dr. techn. Peter Söser

Dr. Walter Werner, Fa. Zumtobel

Graz, Mai 2012

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..

(Unterschrift)

Inhaltsverzeichnis

1 Kurzfassung/Abstract

1.1 Kurzfassung

Die vorliegende Arbeit befasst sich mit der Prototypen-Entwicklung eines neuartigen, selbstjustierenden Lichtsensors zur Messung der Raumhelligkeit. Aufbauend auf ein vorhandenes System zur tageslichtabhängigen Beleuchtungssteuerung der Fa. Zumtobel wird ein Sensor entwickelt, der das Messsignal des bisherigen Sensors deutlich verbessert. Dadurch ist es möglich die bestehenden, strengen Montagevorschriften des Lichtsensors zu lockern.

Der neue Sensor verwendet zur Helligkeitsmessung statt der bisherigen Photodiode eine CMOS-Minikamera. In Kombination mit einer Mikrocontroller-Steuerung können unterschiedliche Lichtanteile voneinander getrennt und unterschiedlich gewichtet werden. Die Gewichtung erfolgt selbstjustierend, wodurch sich der Sensor automatisch an neue Lichtsituationen anpassen kann.

Zur Datenübertragung und Energieversorgung verfügt der Sensor über eine 4-20 mA Stromschnittstelle um eine nahtlose Integration in das bestehende System zu gewährleisten.

1.2 Abstract

This thesis addresses the development of a prototype of a new, self-adjusting light sensor for measuring the ambient light. Building on an existing system for daylight-dependent lighting control from the company Zumtobel, a new sensor is developed to improve the signal of the old one. Thus it is possible to relax the strict mounting instructions of the light sensor.

The new sensor uses a CMOS-minicamera for light measurement instead of the previous photodiode. In combination with a microcontroller-program, different light components are separated and weighted differently. Weighting is self-adjusting, and causes the sensor to automatically adjust to new lighting conditions.

For data transfer and power supply the sensor has a 4-20 mA current-loop interface for seamless integration into the existing system infrastructure.

2 Motivation und Aufgabenstellung

Moderne Gebäudebeleuchtung stellt hohe Anforderungen an die gewählte Lichtlösung. Die Beleuchtung soll immer ausreichend zur Verfügung stehen, sich in das Gebäudekonzept nahtlos integrieren und einfach bedienbar sein. Gleichzeitig soll der Wartungsaufwand gering und der Stromverbrauch niedrig sein um Kosten zu sparen und die Umwelt zu schonen.

Die Vorarlberger Firma Zumtobel entwickelt seit über 50 Jahren innovative und individuelle Lichtlösungen, die ergonomischen und ökologischen Ansprüchen gerecht werden und einen ästhetischen Mehrwert schaffen. [42]

Ein Teil solch einer innovativen Lichtlösung ist eine tageslichtabhängige Beleuchtungsteuerung. Damit kann die Innenraumbeleuchtung über weite Strecken abhängig von der aktuellen Außenlichtsituation konstant gehalten werden. Die künstliche Beleuchtung wird immer nur dann benötigt wenn nicht genügend Tageslicht zur Verfügung steht. Das schafft ein angenehmes Raumklima und spart Energie.

2.1 Ausgangssituation

Eine bestehende Lösung der Fa. Zumtobel für die tageslichtabhängige Lichtsteuerung verwendet einen an der Decke montierten Lichtsensor, welcher auf eine Fensterflächen gerichtet die Helligkeit der Außenlichtsituation misst. Davon abgeleitet kann ein Steuergerät die Raumleuchten so dimmen, dass die Beleuchtungsstärke etwa konstant bleibt.

Die Lichtsensoren (Markenbezeichnung LSD-Sensoren) arbeiten auf Basis einer speziell angepassten Photodiode, welche dem Hell-Empfinden des menschlichen Auges angepasst ist. Damit diese Sensoren einen guten Messwert liefern, sind genaue Montagevorschriften einzuhalten.

2.1.1 Gesamtsystem und Montagevorschriften

Der Sensor misst die Außenhelligkeit von der Decke oberhalb eines Fensters. In der optischen Achse bzw. im Messfeld dürfen keine Störquellen, wie abgehängte Leuchten, Fensterstürze, Fensterkreuze oder Säulen auftreten. Auch direktes Sonnenlicht ist nicht zulässig.

Jalousien oder Abschattungen vor dem Fenster beeinflussen den Messwert hingegen nicht negativ, solange diese Störquellen bei allen Fenstern des Raumes gleich auftreten. Aus diesem Grund muss das auszuwertende Fenster repräsentativ für den Raum ausgewählt werden.

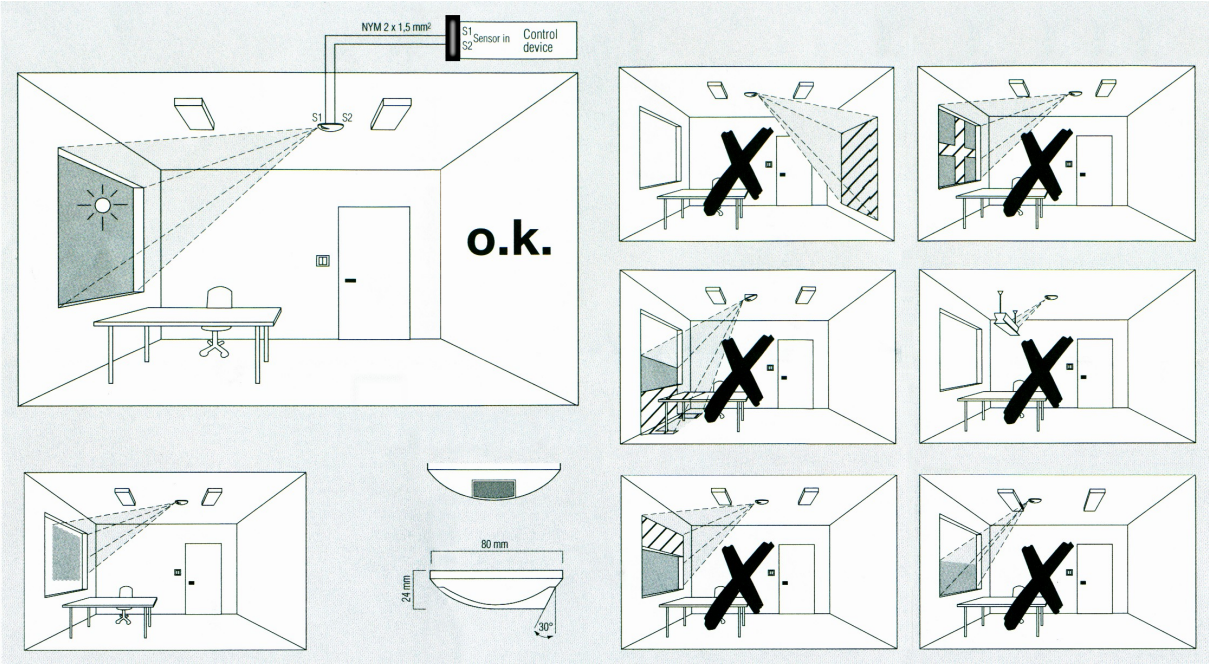

Abbildung 1: Gesamtsystem mit korrekter und fehlerhafter Sensormontage; (Auszug aus Datenblatt LSD-Lichtsensor [43])

Das Leuchtensteuergerät übernimmt den so ermittelten Messwert und regelt die Leuchtenhelligkeit entsprechend. Aufgrund der baulichen Situation (Fenstergröße, Himmelsrichtung, ...) kann sich die resultierende Helligkeit von Raum zu Raum deutlich unterscheiden. Um das auszugleichen wird bei der Installation des Systems einmalig ein so genannter Tag-Systempunkt programmiert. Damit wird die passende Skalierung des Messwertes vorgenommen.

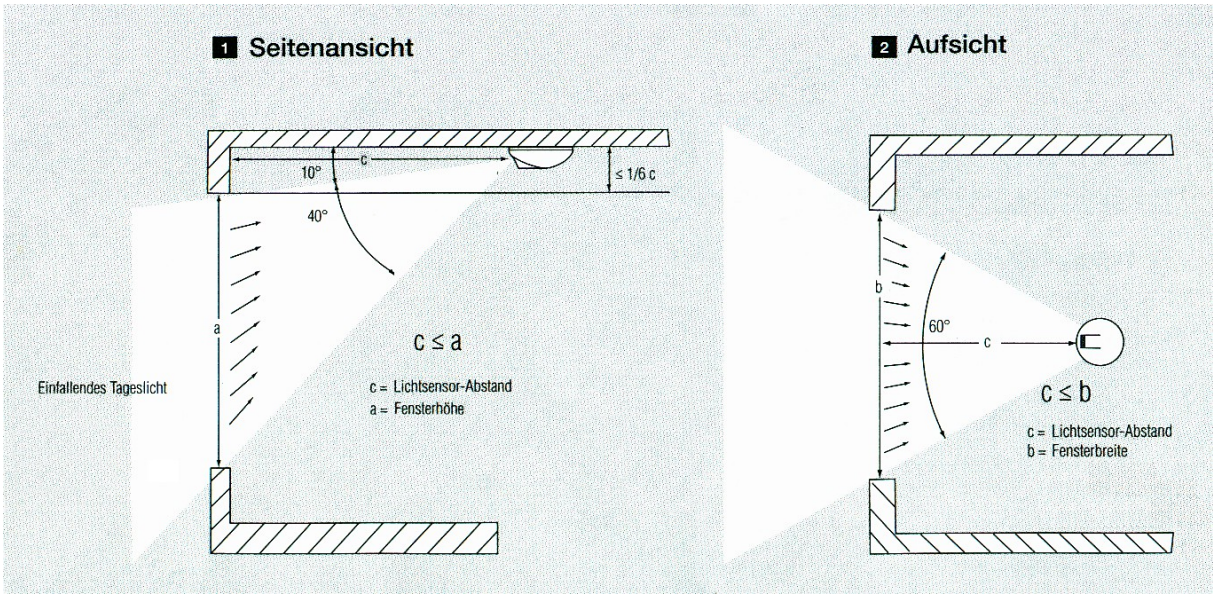

Abbildung 2: Montagevorschriften LSD-Lichtsensor [43]

Abbildung 3: LSD-Lichtsensor (links mit Gehäuse, rechts Innenleben)

Der Lichtsensor besteht aus einer reinen Analogschaltung, welche den Messwert der Photodiode über mehrere Verstärkerstufen in das Ausgangssignal umwandelt. Die Photodiode ist in obiger Abbildung ganz im Vordergrund zu sehen. Der weiße Taster in der Sensormitte dient zur Programmierung eines so genannten Tag-Systempunktes für das Steuergerät. Dazu überträgt der Sensor den maximalen Messwert solange der Taster gedrückt ist. Hinter dem Taster ist die Anschlussklemme für die 4-20 mA Stromschnittstelle zu erkennen.

2.1.2 Übertragung des Messwertes

Der Lichtsensor misst die Helligkeit linear im Bereich von 0-5000 Lux und gibt den Messwert analog über eine Stromschnittstelle mit 4-20 mA aus. 4 mA Schleifenstrom entsprechen dabei einem Messwert von 0 Lux und 20 mA, 5000 Lux Beleuchtungsstärke.[1]

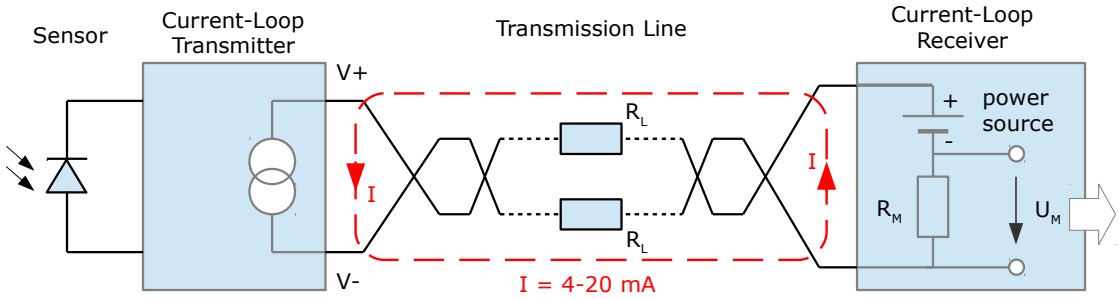

Abbildung 4: Prinzip der 4-20 mA Stromschnittstelle

1 Die Fa. Zumtobel bietet den gleichen Sensor auch mit einer digitalen Schnittstelle an. Diese Variante mit der Digital Addressable Lighting Interface (DALI) genannte Schnittstelle wird in dieser Arbeit nicht berücksichtigt.

Die Stromschnittstelle, oft auch als Stromschleife oder Current-Loop [44] bezeichnet, dient neben der Übermittlung des Messwertes auch zur Versorgung des Sensors. Bei einer Nennspannung von 15 V und 4 mA Schleifenstrom sind das 0,06 Watt. Durch die Leitungswiderstände und den Messwiderstand im Empfänger sinkt die Schleifenspannung bei höheren Schleifenströmen.

Vorteile der Stromschnittstelle

- nur zwei Leitungen für Energieversorgung und Messwertübertragung notwendig
- unempfindlich gegenüber Potentialunterschieden und anderen Störspannungen durch den eingeprägten Strom
- einfache Erkennung einer Leitungsunterbrechung wenn I=0
- gängiger Standard mit einfachem Aufbau

2.1.3 Erweiterungswünsche

Die vorgestellte Lösung zur tageslichtabhängigen Lichtsteuerung funktioniert seit über 15 Jahren zuverlässig und ist in vielen Gebäuden installiert. Trotz dieses Erfolges gibt es Verbesserungswünsche an das System.

Die praktische Erfahrung hat gezeigt, dass die korrekte Montage des Sensors ein Hauptproblem darstellt. Oft wird das System in bestehende Gebäude nachträglich eingebaut. Die vorhandenen Verkabelungen reichen meist nur bis zu den Leuchtmitteln, welche für gewöhnlich im mittleren Bereich des Raumes montiert sind. Für einen Elektroinstallateur ist es relativ einfach die Signalleitungen für den Sensor in die vorhandenen Installationsrohre nachzuziehen, aber relativ aufwändig neue Rohre zu verlegen.

Aus diesem Grund oder einfach weil sich der Monteur der Wichtigkeit der korrekten Sensorplatzierung nicht bewusst ist, wird der Lichtsensor dort angebracht wo dies mit wenig Aufwand möglich ist und nicht dort wo der Sensor laut Installationsvorschrift hingehört. Das Signal des Sensors kann dadurch nicht den korrekten Messwert liefern und die Beleuchtungssteuerung arbeitet nicht wie vorgesehen.

Ebenfalls wünschenswert wäre es, Störquellen, die den Messwert verfälschen, aus dem Signal herauszufiltern zu können. Solche Störgrößen sind z.B. Kunstlichtquellen, Spiegelungen, Abschattungen oder kurzfristige Störungen wie am Fenster vorbeiziehende Passanten, Fahrzeuge oder kleine Wolken.

Um diese Probleme zu lösen, wäre es wünschenswert einen Sensor zu haben, der unabhängig von der Montagesituation selbständig das Signal an den korrekten Wert anpasst. Mit der Lösung dieser Aufgabe beschäftigt sich diese Arbeit.

2.2 Aufgabenstellung

Ausgehend von der oben beschriebenen Ausgangssituation und den Erweiterungswünschen ist für diese Arbeit folgende Aufgabenstellung definiert worden:

„Es soll ein Lichtsensor zur Deckenmontage entwickelt werden, der ein kameraähnliches Bild aufnimmt (z.B. 64-Pixel) und durch Abgleich und Vergleich der Charakteristik der dynamischen Veränderungen der Pixelwerte die verschiedenen Lichtanteile (diffuses Tageslicht, gerichtetes Sonnenlicht, von Sachen und Personen reflektiertes Licht aus der Beleuchtungsanlage) zuverlässig differenzieren und die Komponenten getrennt ermitteln kann.

Als Fertigstellungsgrad ist ein "working prototype"/Funktionsmuster vorgesehen, das bis zu einer Serie von wenigen 10 Stück aufgelegt werden kann und an dem Funktion und Einsatzparameter demonstriert und weiter untersucht werden können."

Natürlich soll sich der Sensor auch nahtlos in das bestehende System integrieren lassen und bei Bedarf vorhandene Sensoren ersetzen können.

3 Grundlagen

Zu Beginn soll ein kurzer Überblick der für die Lösung der Aufgabenstellung notwendigen lichttechnischen Grundlagen wie die menschliche Helligkeitswahrnehmung, physikalische Grundlagen sowie die technischen Möglichkeiten zur Helligkeitsmessung gegeben werden.

3.1 Menschliche Helligkeitswahrnehmung

Das menschliche Auge ist in der Lage elektromagnetische Strahlung mit einer Wellenlänge von ca. 380 nm bis 780 nm wahrzunehmen. [1][8] Dieser Bereich, das sichtbare Licht, reicht von kurzwelliger Ultraviolett-Strahlung (UV) über die Farben Blau, Grün, Gelb, Rot bis zur langwelligen Infrarotstrahlung (IR). Nicht jede Wellenlänge wird dabei mit der gleichen Intensität wahrgenommen.

3.1.1 V-Lambda-Kurve

Die so genannte Helligkeitsempfindlichkeitskurve V(λ) beschreibt den Zusammenhang der spektralen Hell-Empfindlichkeit V des menschlichen Auges aufgetragen über die Wellenlänge λ. Im Bereich λ = 555 nm (Grün) erreicht die Hell-Empfindlichkeit ihr Maximum.

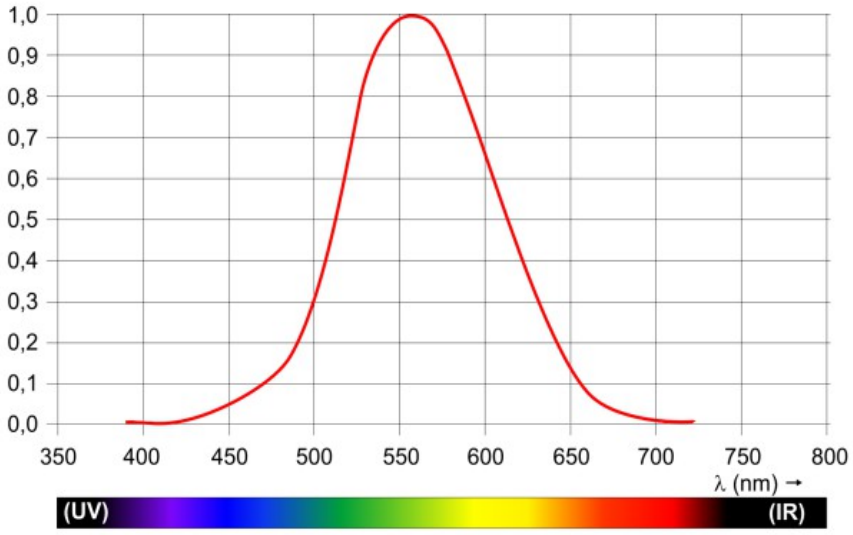

Abbildung 5: Helligkeitsempfindlichkeitskurve V(λ) des menschlichen Auges für das Tagsehen. Das Empfindlichkeitsmaximum liegt bei ca. λ = 555 nm. Die Kurve ist auf den Maximalwert normiert. (Grafik verändert übernommen aus [2]).

Die Ermittlung der V(λ)-Kurve erfolgte empirisch bereits im Jahr 1924 durch die Internationale Beleuchtungskommission, Commission Internationale de l'Éclairage, CIE. [2][6][8] Die 1983 überarbeitete Version ist unter CIE 018.2-1983 [7] publiziert und gilt für den photopischen Bereich das so genannte Tagsehen. Im mesopischen Bereich, dem Dämmerungssehen verschiebt sich die Kurve etwas in Richtung kürzere Wellenlängen (Blaubereich).

Die V(λ)-Kurve bildet die Grundlage für die Definition der physikalischen Einheit Lichtstrom und ist deshalb von großer Bedeutung (siehe Kapitel 3.2.2).

3.1.2 Weber-Fechner-Gesetz [2]

Die menschliche Wahrnehmung der Helligkeit hängt nicht nur von der Wellenlänge der Strahlung ab, sondern zeigt wie bei anderen Sinneseindrücken auch, logarithmisches Verhalten. Der subjektive Sinneseindruck ist proportional zum Logarithmus der objektiven Stärke des physikalischen Reizes. [4]

Mathematisch ausgedrückt ergibt sich folgender Zusammenhang:

$$E_S = c_S \cdot \ln\left(\frac{R_S}{R_{S0}}\right) \quad [4]$$

E_S spiegelt die subjektive Stärke des Sinneseindrucks wider, c_S ist eine Konstante, abhängig von der Art des Sinneseindrucks und R_S/R_{S0} gibt den physikalischen Reiz in normierter Form an.

Für die Helligkeitswahrnehmung bedeutet dieser Zusammenhang, das menschliche Auge ist in der Lage, Dämmerungslicht mit weniger als ein Lux Leuchtstärke und Tageslicht bei Sonnenschein mit mehr als 100000 Lux problemlos zu verarbeiten (siehe auch Kapitel 3.2.3).

3.2 Physikalische Größen

Zur Objektivierung der Helligkeit und messtechnischen Erfassung von Licht sind physikalische Größen notwendig, die in diesem Abschnitt definiert werden. [3]

3.2.1 Lichtstärke [Candela]

Die Lichtstärke I_V gibt die Strahlungsenergie einer Lichtquelle pro Raumwinkel (Steradiant), gewichtet nach der spektralen Helligkeitsempfindlichkeitskurve V(λ) des menschlichen Auges, an.

2 Ernst Heinrich Weber (1795 - 1878) , deutscher Physiologe und Anatom.;
Gustav Theodor Fechner (1801 - 1887) , deutscher Physiker und Natur-Philosoph; Beide befassten sich unter anderem mit der Sinneswahrnehmung von Tieren und begründeten die Psychophysik. [4]

Damit ist die Lichtstärke die photometrische[3] Entsprechung der Strahlungsintensität. Gemessen wird die Lichtstärke in der SI-Basiseinheit Candela (cd). [5][8]

Abgeleitete Einheiten der Lichtstärke sind der Lichtstrom und die Beleuchtungsstärke.

3.2.2 Lichtstrom [Lumen]

Der Lichtstrom Φ_V, angegeben in der Einheit Lumen (lm), gibt die Strahlungsleistung einer Lichtquelle photometrischer Entsprechung an. D.h. der Lichtstrom ist genauso wie die Lichtstärke V(λ)-gewichtet. Anders ausgedrückt bedeutet das, zwei punktförmig Lichtquellen unterschiedlicher Farbe aber mit dem gleichen Lumen-Wert werden als gleich hell wahrgenommen. [8][9][10]

Der Zusammenhang mit der Lichtstärke ist wie folgt definiert: $1\,lm = 1\,cd \cdot sr$

In Worten: Der Lichtstrom entspricht der Lichtstärke mal dem Raumwinkel (Steradiant, sr).

Der Lichtstrom ist die gesamte von einer Lichtquelle in alle Richtungen abgestrahlte Lichtleistung. Gemessen werden kann der Lichtstrom nicht direkt, sondern er wird aus der Lichtstärke und der Geometrie der Lichtausbreitung errechnet.

3.2.3 Beleuchtungsstärke [Lux]

Eine weitere und für dieses Arbeit wichtigste lichttechnische Größe ist die Beleuchtungsstärke E_V. Sie ist definiert als Lichtstrom pro Fläche A_e. [11][8]

$$E_V = \frac{\Phi_V}{A_e} = \frac{I_V}{r^2}$$

Die Beleuchtungsstärke einer punktförmigen Lichtquelle mit der Lichtstärke I_V auf eine Fläche A_e nimmt mit dem Quadrat der Entfernung r ab.

Angegeben wird die Beleuchtungsstärke in der SI-Einheit Lux (lx). [11][8]

$$1\,lx = \frac{1\,lm}{m^2} = \frac{1\,cd \cdot sr}{m^2}$$

Qualitativ betrachtet ist die Beleuchtungsstärke ein Maß dafür, wie intensiv eine Fläche beleuchtet wird.

3 Unter Photometrie werden Messverfahren im Bereich des sichtbaren Lichts sowie des UV-Lichts zusammengefasst. Die Photometrie ist ein Teilgebiet der Physik. Photometrische Größen sind unter anderem: Lichtstärke, Lichtstrom, Lichtmenge, Beleuchtungsstärke, Leuchtdichte etc.

3.3 Technische Helligkeitsmessung

Zur messtechnischen Erfassung von Licht bzw. Helligkeit sind mehrere Messprinzipien bzw. Messaufnehmer bekannt. Meist handelt es sich dabei um Halbleiterbauelemente. Einigen davon widmet sich dieser Abschnitt.

3.3.1 Lichtempfindlicher Widerstand

Ein lichtempfindlicher Widerstand, auch Photowiderstand bzw. englisch Light Dependend Resisotor (LDR), ist ein sperrschichtloses Halbleiterbauelement, dessen Widerstandswert von der Beleuchtungsstärke abhängt. Ein Photowiderstand verhält sich wie ein ohmscher Widerstand, d.h. der Widerstandswert hängt nicht von der angelegten Spannung bzw. Polung sondern nur von der auf das Element auftreffenden Lichtmenge ab. Jedoch ist der LDR, insbesondere bei geringer Beleuchtungsstärke stark temperaturabhängig. [12, Abschnitt 23.2 Photowiderstand]

Bei mittleren Beleuchtungsstärken besteht folgender Zusammenhang zwischen Beleuchtungsstärke E_V und dem Widerstandswert R: $R \sim E_V^{-\gamma}$ mit γ zwischen 0,5 und 1.

Bei großen Beleuchtungsstärken strebt der Widerstandswert gegen den Minimalwert, bei kleinen Beleuchtungsstärken gegen den Dunkelwiderstand.

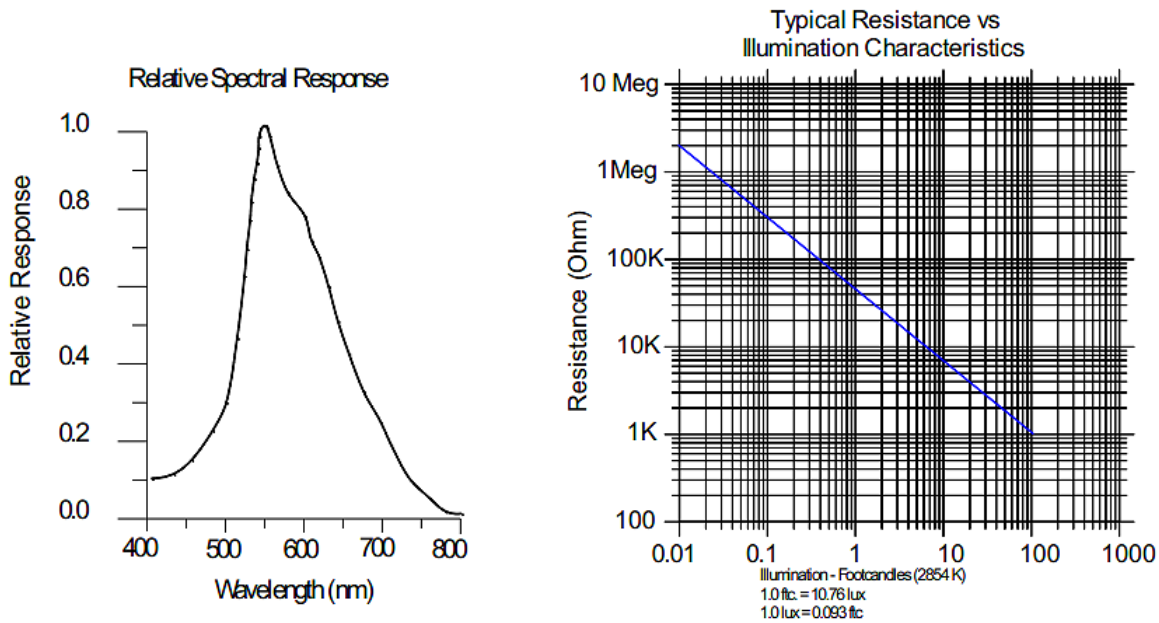

Abbildung 6: Beispiel der spektralen Empfindlichkeit und des Widerstandsverhaltens eines LDRs. [29]

Photowiderstände bestehen oft aus einer Cadmiumsulfid- oder Cadmiumselenid-Schicht, welche im Spektralbereich von 400 nm bis 800 nm empfindlich ist. Spezielle Typen mit V(λ)-Charakteris-

tik sind verfügbar. Empfindlichkeiten im Infrarotbereich (>800 nm) werden mit anderen Materialmischungen, z.B. Bleisulfid oder Indiumantimonid, hergestellt. [12]

Photowiderstände sind in der Lage bis zu sechs Dekaden an Helligkeitsänderung zu verarbeiten. Allerdings benötigen sie zur Einstellung auf einen stationären Wert eine gewisse Zeitspanne. Abhängig von der Beleuchtungsstärke einige Millisekunden bis Sekunden. Aus diesem Grund werden LDRs vorwiegend dort eingesetzt wo es auf schnelle Reaktionszeiten nicht ankommt. Beispielsweise in Belichtungsmessern von Kameras oder Dämmerungssensoren.

3.3.2 Photodiode

Auch die Photodiode ist ein Halbleiterbauelement, das auf Lichteinfall reagiert. Im Gegensatz zum Photowiderstand benötigt eine Photodiode keine externe Spannungsquelle sondern erzeugt bei Lichteinfall einen Strom. Der Sperrstrom einer Diode steigt an, wenn Licht auf den pn-Übergang des Halbleiter auftrifft. Durch die Energiezufuhr des Lichts können aufgrund des inneren photoelektrischen Effektes[4] Elektronen vom Valenzband in das Leitungsband gelangen und dadurch die Leitfähigkeit erhöhen. [12][13][18]

Durch die geeignete Messung des Sperrstroms kann die Beleuchtungsstärke ermittelt werden. Aufgrund des relativ kleinen Photostroms benötigen Photodioden in der Regel einen Verstärker.

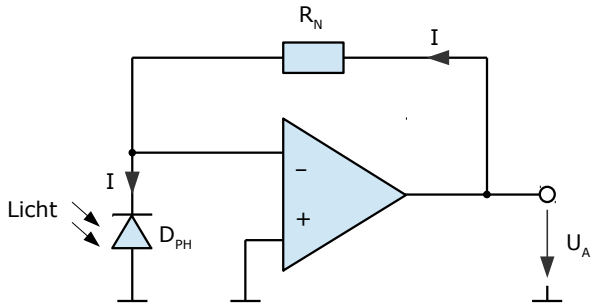

Abbildung 7: Grundschaltung eines Strom-Spannungswandlers für den Photodiodenstrom; $U_A = R_N \cdot I$; [12, Abschnitt 23.3 Photodiode]

Die spektrale Empfindlichkeit von Photodioden ist abhängig vom verwendeten Halbleitermaterial. Bei Silizium liegt der Maximalwert der Photoempfindlichkeit bei ca. 800 nm, bei Germanium

4 Innerer photoelektrischer Effekt: Erhöhung der Leitfähigkeit von Halbleitermaterial bei Bestrahlung (z.B. Licht). Auftreffenden Photonen geben ihre Energie an die Ladungsträger im Halbleiter ab. Ist die eingebrachte Energie größer als die Bandlücke zwischen Valenz- und Leitungsband, entstehen Ladungsträger-Paare. Die Elektronen werden in das energetisch höher gelegene Leitungsband gehoben und erhöhen damit die Leitfähigkeit des Halbleiters. [17][18, Abschnitt 4.2, Injektion von Minoritäts- und Majoritätsträgern]

bei etwa 1400 nm. Durch geeignete Materialwahl und ev. einer zusätzlichen Beschichtung des Diodengehäuses kann jedoch annähernd eine V(λ)-Charakteristik erreicht werden. Ebenso wie Photowiderstände können Photodioden die Beleuchtungsstärke über mehrere Dekaden messen.

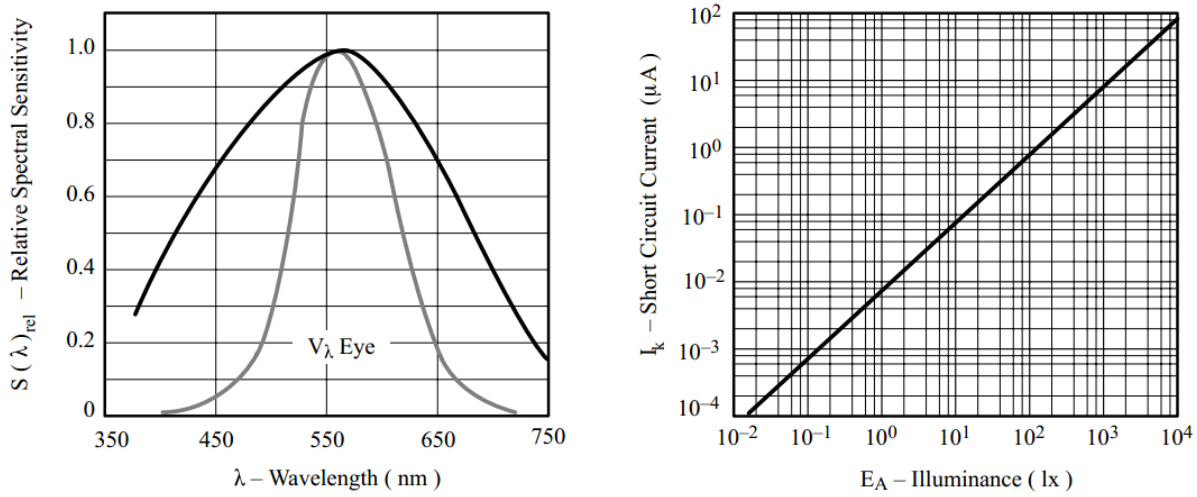

Abbildung 8: Beispiel einer Photodiode mit speziell V(λ) angepasstem Ansprechverhalten. [14]

Photodioden haben wesentlich kürzere Ansprechzeiten als Photowiderstände. Grenzfrequenzen liegen im mehrstelligen MHz- bzw. GHz Bereich. Dadurch sind Photodioden besonders für Applikationen geeignet, die eine schnelle Lichtmessung erfordern. Beispiele sind Empfänger für Optokoppler, Lichtwellenleiter oder Bildsensoren.

3.4 Bildsensoren

Bildsensoren dienen dazu zweidimensionale Abbilder einer Lichtsituation auf elektronischen Weg zu ermitteln. Die Resultate werden in Bildern, einer Matrix aus Bildpunkten (Pixeln) digital gespeichert. Je nach Art der Bilddaten besteht ein Pixel aus einem Helligkeitswert, im Fall von Schwarzweiß-Bildern oder aus mehreren Helligkeitswerten im Fall von Farbbildern.

In jedem Fall werden die Bilddaten über eine Matrix aus lichtempfindlichen Bauteilen detektiert, digitalisiert[5] und gespeichert bzw. an die nachfolgende Verarbeitungseinheit weitergegeben.

5 In dieser Arbeit werden ausschließlich Sensoren mit digitaler Datenschnittstelle betrachtet. Die ebenfalls noch erhältlichen Analogsensoren werden nicht behandelt.

3.4.1 CCD Sensoren

Bei CCD-Sensoren sind lichtempfindliche Elemente, ähnlich einer Photodiode, matrixartig auf ein Halbleiter-Die aufgebracht. Die Abkürzung CCD steht für Charge-Coupled Device, also ein ladungsgekoppeltes Bauelement.

Wie bei der Photodiode werden durch die Energie der Lichtphotonen freie Ladungsträger aufgrund des inneren photoelektrischen Effekts erzeugt. Die Ladungsträger werden dabei in einem so genannten Potentialtopf gesammelt und später zur Weiterverarbeitung verschoben. Im Unterschied zur Photodiode fließt die Ladung nicht sofort ab, daher entsteht kein Photostrom. [16][19]

Die gesammelte Ladung ist proportional der einfallenden Lichtmenge. Da die Ladungsmenge, die der Potentialtopf aufnehmen kann, begrenzt ist, muss die Ladung rechtzeitig und periodisch abgeführt bzw. in einen Zwischenspeicher verschoben werden. Dies geschieht durch Anlegen einer Spannung und Überführung der Ladung in einen nicht lichtempfindlichen Bereich des Sensors. Von dort wird die Ladung über ein Schieberegister zum Ausgangsverstärker weitergeleitet und letztendlich digitalisiert. [16][20, Abschnitt 3.2.1.1 CCD-Wandler-Techniken][21]

Beim Ladungstransport zum Ausgangsverstärker soll das Signal möglichst genau erhalten bleiben um die Bildinformationen nicht zu verfälschen. Ein Verändern der Ladungen während des Transportes durch Lichteinfall ist zu verhindern. Gleichzeitig soll auch das Auslesen möglichst schnell abgewickelt werden, die Lichtempfindlichkeit des Sensors möglichst hoch sein und die Kosten niedrig bleiben. Zu diesem Zweck wurden verschiedene Architekturen von CCD-Sensoren entwickelt. Jede mit unterschiedlichen Vor- und Nachteilen. Exemplarisch sei die Bauform Interline-Transfer-CCD (IT-CCD) herausgegriffen.

Interline-Transfer-CCD

Bei dieser Variante des CCD-Sensors sind die lichtempfindlichen Elemente streifenweise am Halbleiter angeordnet. Jedes Element ist mit einem abgedunkelten Schieberegister verbunden, welches die Ladungen nach jedem Aufnahmevorgang übernimmt und sie vor Veränderung durch Lichteinfall schützt. Am Ende jeder Spalte befindet sich das Ausleseregister, welches die Ladungen zum Ausgangsverstärker transportiert.

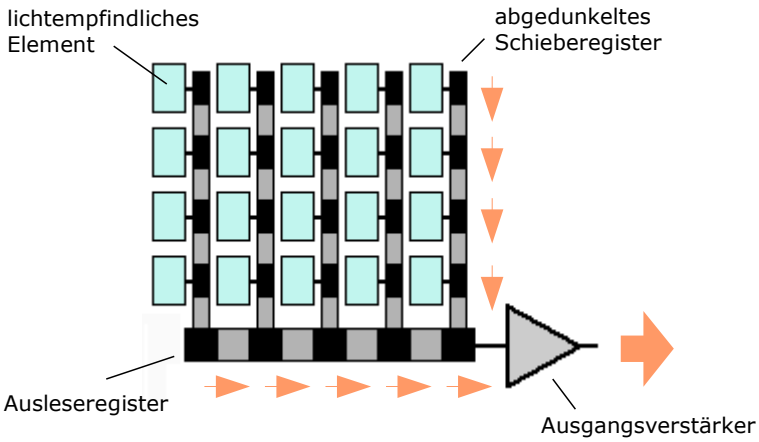

Abbildung 9: Interline-Transfer CCD-Sensor

Vor- und Nachteile von CCD-Sensoren

Beim CCD-Prinzip handelt es sich um eine der ersten Techniken um Bildsensoren herzustellen. Bereits in den 70er Jahren des vorigen Jahrhunderts wurden Sensoren für Fernsehkameras nach diesem Prinzip gebaut, da andere Techniken noch nicht entwickelt waren oder nicht die erforderliche Qualität lieferten.

Die Lichtempfindlichkeit und Bildqualität ist auch einer der Gründe dafür warum CCD-Sensoren auch heute noch in vielen Digitalkameras verbaut werden.

Nachteilig ist, dass CCD-Sensoren einen speziellen Fertigungsprozess benötigen und nicht in einem Standard CMOS-Prozess hergestellt werden können. Dadurch sind die Produktionskosten höher und die Integration zusätzlicher Systeme, wie z.B. ein Bildverarbeitungsprozessor auf dem Sensorchip, schwieriger bzw. gar nicht möglich.

Auch neigen CCD-Sensoren zu besonderen Bildfehlern, die nur bei diesem Sensortyp auftreten. Zu nennen sind der bekannte Blooming-Effekt, das sind ausgefranste und überbelichtete Bildteile an hellen Objekten im Bild, hervorgerufen durch lokale Überbelichtung der Pixel. Wird aufgrund der eingestellten Belichtungszeit die Ladung, die das einfallende Licht am Sensorelement erzeugt, nicht rechtzeitig abtransportiert, kann der Potentialtopf die Energie nicht zur Gänze aufnehmen und gibt die überschüssige Ladung an die Nachbarelemente ab. Diese erscheinen im Bild dann heller als sie aufgrund des Lichteinfalls eigentlich sein sollten. [16]

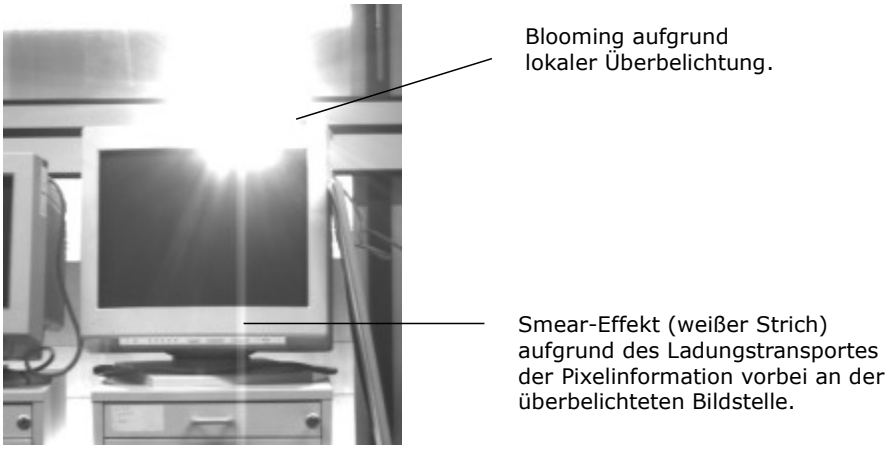

Abbildung 10: Blooming- und Smear-Effekt bei CCD-Sensoren (Bild verändert übernommen aus [22])

Der zweite Effekt, der Smear-Effekt, tritt beim Transport der Ladungen über die Schieberegister zum Ausgangsverstärker auf. Beim Vorbeischieben der Pixel am überbelichteten Bildteil, gibt dieser einen Teil des Ladungsüberschusses an die vorbeiwandernden Pixel ab. Dadurch entstehen helle Striche im Bild. Bei statischen Lichtquellen verlaufen diese Striche senkrecht oder waagerecht, bei beweglichen Lichtquellen in einem anderen Winkel. Gut zu beobachten ist dieser Effekt bei älteren Videoaufnahmen an den nachziehenden hellen Linien, die bei Glanzpunkten oder Scheinwerfern im Bild auftreten. [16]

Moderne CCD-Sensoren verfügen über Möglichkeiten den Blooming- und Smear-Effekt stark zu reduzieren.

3.4.2 CMOS Sensoren

CMOS-Sensoren bestehen im Wesentlichen aus einer Matrix von Photodioden mit passender Auswerteschaltung. Durch den CMOS-Herstellungsprozess ist es einfach möglich am Sensorchip auch gleich die Elektronik für die Bildnachbearbeitung unterzubringen. Das spart Chipfläche und damit Kosten.

Die früher vorhandenen Nachteile gegenüber den CCD-Sensoren, wie z.B. die geringere Lichtempfindlichkeit, werden in den letzten Jahren immer mehr durch verbesserte Fertigungsprozesse stark minimiert bzw. aufgehoben. In vielen Anwendungsgebieten werden CCD-Sensoren immer mehr durch CMOS-Sensoren abgelöst.

Funktionsprinzip

Aktuelle CMOS-Bildsensoren werden meist nach dem Active-Pixel-Prinzip aufgebaut. Dabei befindet sich die Verstärkerschaltung für die Photodiode direkt neben dieser. Das hat den Vorteil eines geringeren Signal-Rauschabstandes (SNR) als bei einem Passive-Pixel-Sensor, bei dem die Signalverarbeitung außerhalb der aktiven Sensorfläche stattfindet. Der Nachteil, die geringere Lichtempfindlichkeit des Sensors aufgrund der nicht lichtaktiven Beschaltung neben der Photodiode, kann durch Aufbringen einer so genannten Mikrolinse kompensiert werden. [20, 3.2.2 CMOS Techniken]

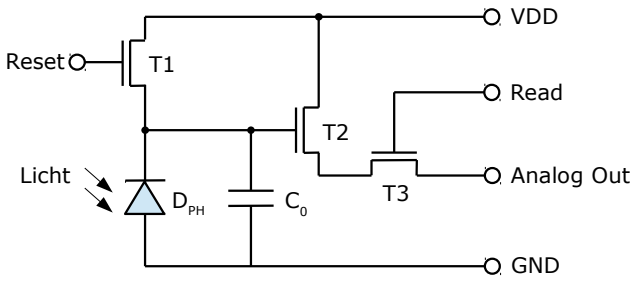

Abbildung 11: Active-Pixel-Sensor Funktionsprinzip [20][23]

In Abbildung 11 dient der Transistor T1 dazu die Schaltung in den Ausgangszustand zu bringen. Der Kondensator C_0 wird während eines Reset-Pules auf VDD geladen. Nach dem Reset bewirkt die mit Licht beaufschlagte Photodiode D_{PH} ein Entladen des Kondensators. Die am Ende der Belichtungszeit am Kondensator vorhandene Spannung ist proportional zum Messwert. Transistor T2 verstärkt das Signal entsprechend und T3 dient zum Auslesen des Messwerts. Nachfolgend wird der Analogwert digitalisiert.

Der Zugriff auf die Pixelinformationen eines CMOS-Sensors erfolgt wahlfrei, d.h. diese werden ähnlich wie Speicherbausteine über eine Zeilen- und Spaltenadresse ausgelesen (Read-Signal). Damit ist es möglich nur einen Teil des Sensors auszulesen und so z.B. einfach einen Digitalzoom zu realisieren oder eine niedrigere Bildauflösung mit höherer Bildwiederholrate auszugeben. Das ist einen weiterer Vorteil gegenüber CCD-Sensoren, welche immer alle Pixelwerte in Form von Ladungen zum Ausgangsverstärker herausschieben müssen.

3.4.3 Auflösung und Sensorgröße

Der Begriff Auflösung wird oft für unterschiedliche Dinge benützt und nicht streng unterschieden. Bei Bildsensoren wird damit meist die Gesamtanzahl der lichtempfindlichen Elemente ver-

standen. Zur Unterscheidung werden Sensoren in Auflösungsklassen eingeteilt.

Bezeichnung	Auflösung	Pixelanzahl (gerundet)
QCIF	176 x 144	0,02 Megapixel
CIF	352 x 288	0,1 Megapixel
VGA	640 x 480	0,3 Megapixel
SVGA	800 x 600	0,5 Megapixel
XGA	1024 x 768	0,8 Megapixel
SXGA	1280 x 1024	1,3 Megapixel
UXGA	1600 x 1200	2 Megapixel
...	...	

Tabelle 1: Auflösungsklassen (Auswahl mit Fokus auf vergleichsweise geringe Auflösungen [35][62])

Die Anzahl der Megapixel sagt aber noch nichts über die Größe der einzelnen Pixel aus. Dafür kann ein anderer Auflösungsbegriff, DPI (Dots per Inch), herangezogen werden. Dieser stellt die Relation zwischen Sensorgröße und Pixelanzahl her. Je größer die einzelnen Pixel, desto großer der Bildsensor bei gleichbleibender Pixelanzahl.

Mit der Pixelgröße steigt für gewöhnlich auch die Lichtempfindlichkeit des Sensors. Ein 5-Megapixel-Sensor mit 2/3“ Diagonale ist bei gleicher Fertigungstechnologie lichtempfindlicher und rauschärmer als ein 5-Megapixel-Sensor mit 1/5“ Diagonale. Meist werden Bildsensoren mit einem Seitenverhältnis von 4:3 hergestellt. In jüngster Zeit sind aber auch die Formate 16:9 oder 16:10 populär geworden.

Format	Maß B x H [mm]	typische Verwendung
1/6“	2,4 x 1,8	Mobiltelefon
1/5“	2,8 x 2,1	Mobiltelefon / Webcam
1/2.5“	5,8 x 4,3	Kompaktkamera
1/1,8“	7,2 x 5,4	Kompaktkamera
2/3“	8,8 x 6,6	Kompaktkamera (High End)
Four-Thirds	17,3 x 13	Digitale Spiegelreflexkamera / Systemkamera
APS-C	22,2 x 14,8	Digitale Spiegelreflexkamera / Systemkamera
Kleinbild	36 x 24	Digitale Spiegelreflexkamera (High End)

Tabelle 2: Sensorgrößen und Verwendungszweck (4:3 Format, Auswahl [63])

3.4.4 Farbsensoren

Bildsensoren, egal ob nach dem CCD- oder CMOS-Prinzip, messen die Stärke des einfallenden Lichts. Die Wellenlänge und damit die Lichtfarbe bleiben unberücksichtigt. Um ein Farbbild aufnehmen zu können, muss das Licht vor dem Auftreffen auf den Sensor in die Grundfarben Rot, Grün und Blau zerlegt werden.[6]

Eine Möglichkeit ist, mittels eines speziellen Prismas das Bild in seine Farbkanäle zu trennen und jeden Kanal einem separaten Bildsensor zuzuführen. Jedes Pixel kann dann gleichzeitig in jeder Grundfarbe erfasst werden. Diese Lösung ist aufwändig und teuer, wird aber von speziellen hochwertigen Fernsehkameras genutzt. [20, Abschnitt 3.2.3.3 Dreichip-Farbkameras]

Ein anderer Ansatz ist die Ausnutzung der wellenlängenabhängigen Lichtabsorption von Halbleitermaterial. Unterschiedliche Farbanteile des Lichts dringen unterschiedlich tief ins Halbleitermaterial ein. Dieser Ansatz gleicht dem Aufbau von analogem Farbfilm, bei dem unterschiedlich farbempfindliche Schichten übereinander aufgebracht sind. Im Halbleiter müssen die Sensorelemente daher ebenfalls übereinander gefertigt werden. Ganz oben befindet sich der Blausensor, in der Mitte der Grünsensor und unten der Rotsensor. Die Fa. Foveon stellt ihren „X3 Sensor“ in dieser Bauart her. [24]

Die günstigste und am häufigsten verwendete Lösung für das Farbproblem stellt eine spezielle Anordnung von Farbfiltern vor den Sensorelementen dar. Das so genannte Bayer-Pattern.

Bayer-Pattern[7]

Die Anordnung der Farbfilter im Bayer-Pattern macht sich die Tatsache zu Nutze, dass das menschliche Auge für die Farbe Grün die höchste Empfindlichkeit aufweist (Maximalwert der V(λ)-Kurve, vgl. Kapitel 3.1.1 V-Lambda-Kurve).

Die Farbfilter werden vor jedes Sensorelement schachbrettartig in den Grundfarben Rot, Grün und Blau gesetzt, wobei die Fläche zu 50% mit Grünfilter, 25% Rotfilter und 25% mit Blaufilter gefüllt wird. Nachstehende Abbildung zeigt das Schema:

6 Zusätzlich muss bei den meisten Sensoren die Infrarot-Empfindlichkeit des Silizium-Halbleiters gedämpft werden da die Farbwiedergabe sonst unnatürlich wirkt. Dies geschieht mit einem Infrarot-Sperrfilter.

7 Bryce E. Bayer, Ingenieur der Fa. Eastman Kodak, hat die nach ihm benannte Farbfilteranordnung 1975 zum Patent angemeldet. (Patent US3971065)

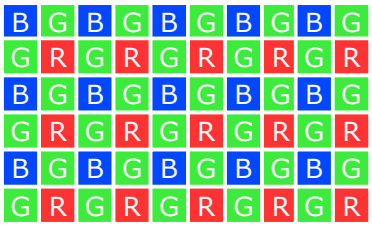

Abbildung 12: Farbfilteranordnung im Bayer-Pattern

Jedes Pixel erfasst also nur die Helligkeit einer Grundfarbe. Die beiden fehlenden Farbanteile werden mittels Interpolation aus den Nachbarpixeln ermittelt. Die Qualität der Interpolation ist entscheidend für das Bildergebnis. Einfache Algorithmen sind z.B. Nearest-Neighbor-, Bi-Linear- oder Bi-Cubic-Interpolation. Qualitativ hochwertigere Interpolationsalgorithmen sind meist Firmengeheimnis der Sensor- bzw. Kamerahersteller und somit nicht öffentlich zugänglich. [26][20, Abschnitt 3.2.3.2 Bayer Farbfilter]

Ein Nachteil der Bayer-Anordnung sei noch erwähnt: Durch die Farbfilter vor jedem Sensorelement geht ein gewisser Teil des Lichtes verloren. Dadurch sinkt die Lichtempfindlichkeit des Sensors. Es gibt daher auch andere Möglichkeiten die Farbfilter anzuordnen. Eine Möglichkeit ist die Verwendung zusätzlicher Weißer, d.h. ungefilterte Pixel, am Sensor. Diese messen das Luminanzsignal des vollen Spektrums. Die Farbinterpolation ist etwas aufwändiger als beim klassischen Bayer-Pattern, die Lichtempfindlichkeit des Sensors steigt jedoch deutlich. [25]

Farbsignal

Da das Licht in die Grundfarben Rot, Grün und Blau separiert wird, ist es sinnvoll diese Werte im Farbsignal zu übertragen. Jedem Pixel wird also ein RGB-Wert zugewiesen aus welchem die Ausgangsfarbe durch additive Farbmischung wieder rekonstruiert werden kann. [27]

3.4.5 Schwarzweiß Signal

Die meisten heute hergestellten Bildsensoren sind Farbsensoren. D.h. obwohl das lichtempfindliche Element eigentlich nur das Luminanzsignal des Lichts messen kann, gibt ein Bildsensor aufgrund der aufgebrachten Farbfilter ein RGB-Signal aus[8].

Aus diesem Signal muss, um ein Schwarzweiß-Bild zu erhalten, das Luminanzsignal, d.h. das Helligkeitsbild aus den Farbanteilen rekonstruiert werden. Dazu sind mehrere Verfahren bekannt, die auf folgende Relation zurückgreifen [28]:

$$Y = K_R \cdot R + K_G \cdot G + K_B \cdot B$$

8 Schwarzweiß-Sensoren oder Sensoren mit anderen Spektralfiltern werden in dieser Arbeit nicht betrachtet.

Das Luminanzsignal Y wird aus den Rot-, Grün- und Blauanteilen (R, G, B) gewichtet mit den Faktoren K_R, K_G und K_B gewonnen. Die einfachste Möglichkeit ist nur den Grünanteil zu verwenden da das menschliche Auge für diese Farbe am empfindlichsten ist (K_R=0, K_G=1, K_B =0). Eine etwas natürlichere Umsetzung ergibt sich mit den Faktoren K_R=0,3, K_G=0,59, K_B =0,11. Andere, speziell auf einen bestimmten Sensortyp abgestimmte, Werte sind natürlich genauso möglich.

3.4.6 Gammakorrektur

Bildsensoren messen die einfallende Lichtmenge proportional zur Beleuchtungsstärke. Das menschliche Auge erfasst Helligkeitswerte logarithmisch um einen sehr großen Helligkeitsumfang wahrzunehmen (siehe Kapitel 3.1.2). Damit die von den Sensoren erfassten Messwerte für den Menschen „linear“ erscheinen, muss die Empfindlichkeitskurve korrigiert werden. Dies geschieht mit der so genannten Gammafunktion.

Gammafunktion

Die Gammafunktion ist eine einfache Exponentialfunktion mit dem Wert γ als Exponenten:

$$A = E^{\gamma}$$

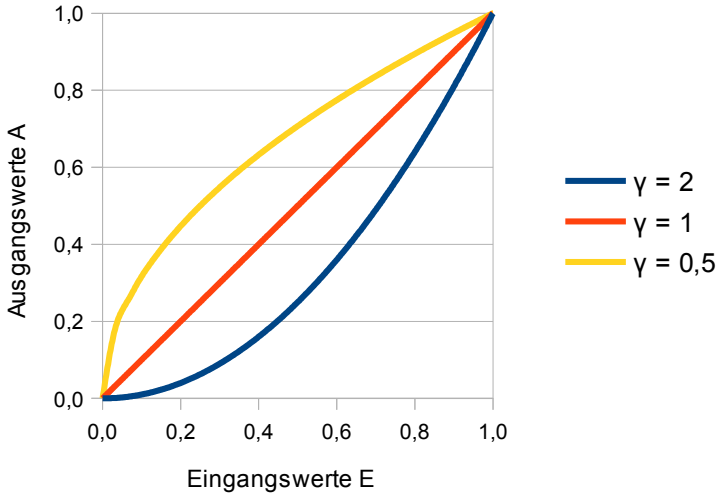

Abbildung 13: Gammafunktion

Das Eingangssignal E, normiert auf das Intervall 0-1, wird nichtlinear auf das Ausgangssignal A umgesetzt. Viele Kamerachips verwenden ein Gamma von etwa γ=0,45.

Als Gammakorrektur bezeichnet man die Umkehrung des zuvor verrechneten Gammawertes. Mit $\gamma' = 1/\gamma$ kann ein Signal wieder in den vorherigen Zustand gebracht werden[9]. [38] [20, Abschnitt 6.1 Lineare Grauwertkorrekturen/Gammakorrektur]

Komplexere Korrekturfunktionen

Je nach Charakteristik des Bildsensors verwenden Kamerahersteller oft auch komplexere Funktionen, das Ausgabesignal möglichst natürlich wirken zu lassen. Die Gammafunktion kann z.B. für jeden Farbkanal einzeln mit unterschiedlichen Gammawerten korrigiert werden. Auch beliebig komplexe Korrekturkurven sind möglich. Man spricht in diesem Fall oft auch von einer Kamerakurve. [32, Abschnitt 3.2, Rekonstruktion der Kamerakurve]

3.4.7 Dynamikumfang

Als Dynamikumfang bezeichnet man den Quotient aus maximal und minimal erfassbarer Signalstärke. Im Bereich der Bilderfassung wird oft auch der Begriff Kontrastumfang verwendet.

Angegeben werden kann der Dynamikumfang z.B. als Verhältnis (z.B. 1:2000), in dB (z.B. 66 dB), in Bit (z.B. 10 Bit) oder in diesem Bereich am häufigsten in Zeit- bzw. Blendenstufen (z.B. 8 Blendenstufen). [30][31] Siehe dazu auch Kapitel 3.5.2.

Bildsensoren sind bei weitem nicht in der Lage die Helligkeitsempfindlichkeit von dunkel zu hell des menschlichen Auges abzubilden. Auch können sie nicht die Werte von Photowiderständen oder diskreten Photodioden erreichen.

Um Bildsensoren dennoch zur Lichtmessung über einen weiten Bereich einsetzen zu können, behilft man sich eines Tricks. Mehrere Bilder mit unterschiedlichen Belichtungseinstellungen, eine so genannte Belichtungsreihe, werden zu einem High-Dynamic-Range-Bild (HDR-Bild) verrechnet. Damit kann die Beleuchtungsstärke über einen weiten Bereich gemessen werden. Wie genau die HDR-Bildberechnung in dieser Arbeit funktioniert ist in Kapitel 4.4 beschrieben. Eine allgemeine Abhandlung zu HDR-Bildern findet sich in [32].

3.4.8 Videosignal und Datenübertragung

Zu den Aufgaben moderner Bildsensoren gehört nicht nur die Bilderfassung selbst sondern auch die Digitalisierung, Zwischenspeicherung, Nachbehandlung und Ausgabe der Bilddaten.

Im einfachsten Fall übermittelt der Bildsensor RAW-Daten, das sind unbearbeitete Messwerte der Pixel direkt nach der Analog-Digital-Wandlung. Ein nachfolgender Digital-Signalprozessor (DSP)

9 Natürlich kann aufgrund der Digitalisierung und der damit verbundenen endlichen Genauigkeit das Ursprungssignal nur annähernd wiederhergestellt werden.

übernimmt die Farbinterpolation, Gammakorrektur, Weißabgleich usw.. Der nachfolgende Verarbeitungsschritt formatiert die Daten im gewünschten Ausgabeformat und gibt diese über eine Digitalschnittstelle aus. Oft ist der DSP bei modernen Bildsensoren bereits am selben Chip integriert.

Digitalschnittstelle für Videodaten

Die Videoschnittstelle bei Bildsensoren ist nicht streng genormt. Es hat sich aber ein Quasi-Standard gebildet, der von vielen Herstellern mehr oder weniger implementiert wird. Die Bild- bzw. Videodaten werden hintereinander Zeile für Zeile und Pixel für Pixel über einen parallelen Port, meist 8 oder 10 Bit breit, ausgegeben. Zur Synchronisation des Datenstroms dienen Zusatzsignale VSYNC, HSYNC bzw. HREF und PCLK.

VSYNC markiert den Beginn eines neuen Frames, also den Bildanfang. HSYNC/HREF markiert eine neue Zeile (Scanline) und PCLK (Pixeltakt) signalisiert die Gültigkeit der Daten an den Datenleitungen. Die Art der Triggersignale, zustandsgetriggert, steigende-/fallende Flanke oder Eizelpuls ist nicht festgelegt und kann bei vielen Sensortypen per Software eingestellt werden. [35]

Für solche und andere Einstellungen verfügen Bildsensoren über eine separate Schnittstelle zur Parametereinstellung. Diese ist in den meisten Fällen kompatibel zum bekannten I2C-Standard der Fa. Philips (jetzt NXP) [34], auch wenn manche Sensorhersteller eine andere Bezeichnung dafür verwenden, z.B. Serial Camera Control Bus (SCCB) bei der Fa. OmniVision. [36]

Ausgabeformate

Wie bei der Datenschnittstelle gibt es auch bei den Ausgabeformaten eine Vielzahl von Varianten. Die meisten Sensoren können aber zumindest ein RGB- und ein YUV-Format ausgeben. [35]

Nachfolgend sind drei gängige Formate herausgegriffen.

<u>RGB565</u>

Pro Pixel werden zwei Byte Daten übertragen. Rot- und die Blaukomponente mit 5 Bit Auflösung und die Grünkomponente mit 6 Bit. Bei einem 8-Bit breiten Datenport ergibt sich folgende Aufteilung:

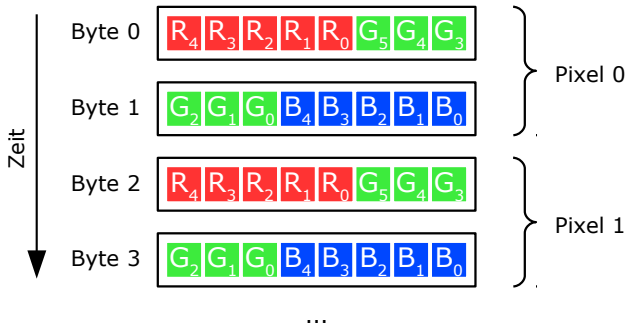

Abbildung 14: Datenstrom RGB565 Format

YUV422

Dieses Format passiert auf der analogen Fernsehtechnik, welche das YUV-Farbmodell als Grundlage verwendet. Y stellt dabei das Luminanzsignal dar und die Komponenten U und V das sogenannte Chroma. Im Luminanzsignal ist die Bildhelligkeit und im Chroma sind die Farbinformation enthalten. Weitere Details sind in [28] zu finden.

Der Datenstrom des YUV422 Signals fasst jeweils zwei Pixel zu einem Makropixel zusammen und überträgt dieses mit 4 Byte.

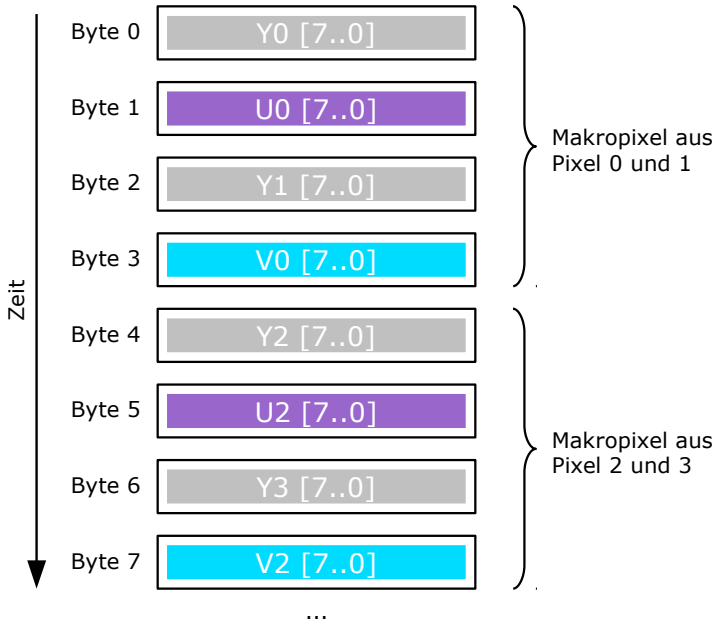

Abbildung 15: Datenstrom YUV422 Format

Interessant ist die Tatsache, dass die Farbwerte nur für jedes 2. Pixel übertragen werden. Der Helligkeitswert jedoch für jedes Pixel. Die fehlende Farbinformation muss vor der Anzeige rekonstruiert werden. Wie das geschieht ist anschaulich in [37] gezeigt.

Y8

Bei diesem Format wird nur das Luminanzsignal und damit ein Schwarzweiß-Bild übertragen. Jedes ausgegebene Byte entspricht dem Helligkeitswert eines Pixels.

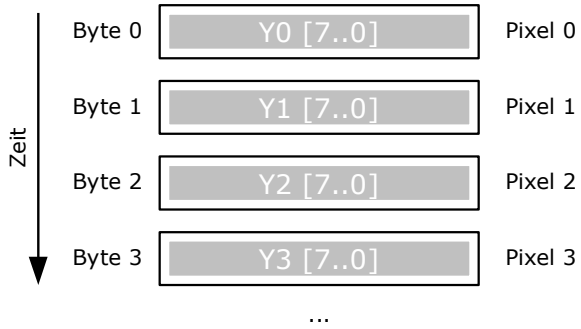

Abbildung 16: Datenstrom Y8 Format

Für die Helligkeitsmessung in dieser Arbeit spielt das Y8 Format eine wichtige Rolle.

3.5 Abbildungssystem

Unter dem Begriff Abbildungssystem werden in dieser Arbeit alle fototechnischen Grundlagen verstanden, die zusätzlich zum Bildsensor für den Aufbau eines Kamerasystem notwendig sind. Neben optischen Grundlagen werden auch fototechnische Begriffe wie Belichtungszeit oder Sensorempfindlichkeit kurz angerissen.

3.5.1 Abbildung

Damit die einzelnen Pixel des Bildsensors den korrekten Lichtanteil des Bildes eines realen Objektes erfassen, muss vor den Sensor ein optisches System gesetzt werden. Diese Objektive genannten Vorrichtungen können als Spiegel- oder Linsensystem ausgeführt werden. [20, Abschnitt 3.2.4.1 Optische Grundlagen]

Abbildung 17 zeigt das Prinzip. Die Lichtstrahlen vom Gegenstand G werden durch das Objektiv, hier vereinfacht durch eine Linse dargestellt, auf der Bildebene hinter dem Objektiv abgebildet (B). Der Strahlengang führt dabei immer durch den Brennpunkt F.

Um eine scharfe Abbildung zu erhalten, muss der Bildsensor in die Bildebene gebracht werden. Diesen Vorgang nennt man fokussieren. Da sich Gegenstände unterschiedlich weit vom Objektiv entfernt befinden können, können immer nur einige davon scharf abgebildet werden. Man spricht hier von der Schärfentiefe.

Da real ausgeführte Objektive und Bildsensoren klein im Vergleich zur Gegenstandsweite sind, können weite Bereiche eines Bildes scharf abgebildet werden. Umgekehrt ist es stilistisch oft wünschenswert nur bestimmte Teile eines Bildes scharf abzubilden, beispielsweise ein scharfes Portrait einer Person vor einem unscharf verschwimmenden Hintergrund. Dies gelingt umso leichter, je größer Objektiv und Bildsensor sind. Das ist ein Grund dafür warum z.B. digitale Spiegelreflexkameras im Vergleich zu Kompakt- oder Handykameras für diese Aufgabe viel besser geeignet sind.

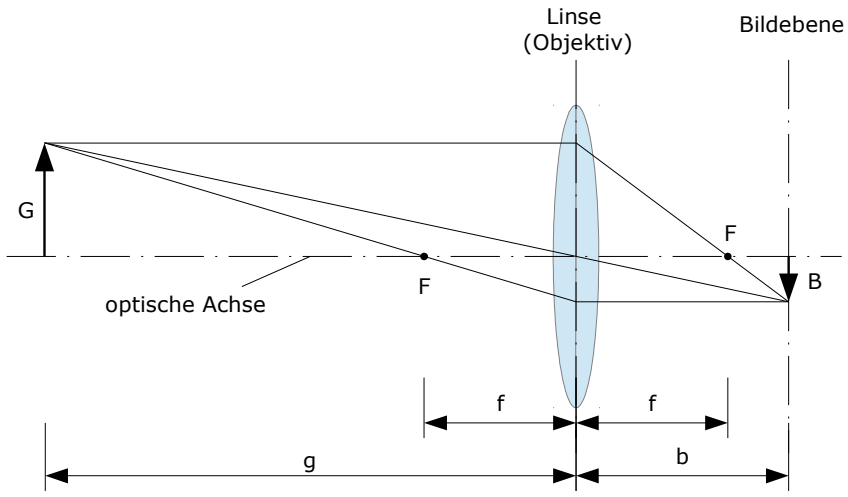

Abbildung 17: Prinzip einer optischen Abbildung

Für die Abbildung gelten folgende Beziehungen:

$$\frac{1}{g} + \frac{1}{b} = \frac{1}{f}$$ (Linsengleichung)

und

$$\frac{B}{G} = \frac{b}{g} = m$$

mit

g … Gegenstandsweite, G … Gegenstandsgröße

b … Bildweite, B … Bildgröße

f … Brennweite

m … Abbildungsmaßstab

3.5.2 Sensorempfindlichkeit, Blende, Belichtungszeit

Für ein korrekt belichtetes Bild sind die drei Parameter Sensorempfindlichkeit, Blende und Belichtungszeit verantwortlich.

Sensorempfindlichkeit

Unter Sensorempfindlichkeit versteht man den Verstärkungsfaktor, mit dem das Messsignal nach einer Belichtung multipliziert wird. Um die Empfindlichkeitswerte verschiedener Bildsensoren miteinander vergleichen zu können, wurde die Empfindlichkeit nach ISO normiert. Gängige Werte sind [31]:

…	ISO50	ISO100	ISO200	ISO400	ISO800	ISO1600	…

Viele Sensoren haben eine Grundempfindlichkeit zwischen ISO50 und ISO200, oft ISO100. Die Werte sind so gestaltet, dass von einem ISO-Schritt zum nächsten jeweils der halbe bzw. der doppelte Messwert ermittelt wird.

Bei analogem Filmmaterial ist der Empfindlichkeitswert chemisch fix vorgegeben. Nur durch Auswechseln der Filmrolle kann der Wert verändert werden. Bei Digitalkameras kann der ISO-Wert oft über einen gewissen Bereich eingestellt werden. Dies geschieht durch Änderung des Verstärkungsfaktors des Signals. Höher verstärkte Signale neigen zu höherem Rauschen. Der Signal-Rauschabstand sinkt.

Blende

Die Blende dient zur Steuerung der einfallenden Lichtmenge durch das Objektiv. Oft werden sogenannte Irisblenden eingesetzt, welche über eine Mechanik die Eintrittsöffnung für das Licht stufenlos einstellen können. Um wieder eine Vergleichbarkeit unterschiedlicher Objektive miteinander herzustellen, wurde eine Blendenreihe eingeführt. Es handelt sich dabei um relative Werte aus dem Quotienten von Brennweite zu effektivem Öffnungsquerschnitt. [31][39]

…	F1.4	F2	F2.8	F4	F5.6	F8	F11	F16	F22	…

Zwischen den Werten liegt jeweils (gerundet) der Faktor $\sqrt{2}$. Dadurch halbiert bzw. verdoppelt sich die Fläche und damit die einfallende Lichtmenge von Schritt zu Schritt. [33]

Der Blendenwert hat auch einen großen Einfluss auf die Schärfentiefe einer Abbildung. Eine offene Blende (kleiner Blendenwert) führt zu einer geringen Schärfentiefe wogegen eine geschlossene Blende (großer Blendenwert) zu großer Schärfentiefe führt.

Belichtungszeit

Das ist jene Zeit, die ein lichtempfindliches Element pro Messvorgang der Bestrahlung ausgesetzt wird. Die Lichtmenge wird also über die Zeit integriert.

In der Fotografie wird die Belichtungszeit in Bruchteilen von Sekunden angegeben [31]:

…	1/1000s	1/500s	1/250s	1/125s	1/60s	1/30s	…

Die Werte sind dabei so gewählt, dass pro Zeitschritt jeweils (gerundet) die Hälfte der Lichtmenge auf den Sensor fällt.

Für die Bildgestaltung spielt die Belichtungszeit auch eine wichtige Rolle. Schnelle Vorgänge müssen mit einer kurzen Belichtungszeit aufgenommen werden um sie im Bild scharf einzufrieren. (z.B. Sportaufnahmen). Andernfalls erscheinen Schieren, die sogenannte Bewegungsunschärfe in der Aufnahme.

Das Tripple Sensorempfindlichkeit, Blende und Belichtungszeit ist eng miteinander verknüpft und die Werte können sich in gewissem Rahmen gegenseitig ersetzen. Z.B. sind folgende Belichtungskombinationen äquivalent, d.h. sie führen alle zu einem gleich (hell) belichtetem Bild:

ISO100, F4.0, 1/125s.
ISO50, F2.8, 1/125s
ISO400, F4.0, 1/500s

Nicht verschwiegen werden darf aber, dass stilistisch alle Kombinationen zu unterschiedlichen Bildergebnissen führen können. Abhängig von Motiv und Lichtsituation. Kombinationen mit niedrigem ISO-Wert führen tendenziell zu rauscharmen Bilden, Kombinationen mit kurzer Belichtungszeit zu Aufnahmen mit wenig Bewegungsunschärfe und solche mit offener Blende zu geringerer Schärfentiefe.

Der Unterschied von einer Belichtungsstufe zur nächsten wird oft auch als Zeitstufe, Blendenstufe oder F-Stop bezeichnet.

3.6 Kameramodule

Kameramodule sind Baugruppen, die Bildsensor, Objektiv und Ansteuerelektronik zu einer Einheit zusammenfassen. Dadurch ist es möglich die elektronisch wie mechanisch teils aufwändige Konstruktion in neue Gerätedesigns einfach zu integrieren. Nach außen kommunizieren Kameramodule über eine einfache Schnittstelle, wie in Kapitel 3.4.8 „Videosignal und Datenübertragung“ beschrieben, und benötigen nur noch Spannungs- und Takt-Versorgung vom umgebenden System.

Kameramodule sind in vielen verschiedenen Bauformen, Größen und Qualitätsstufen erhältlich. Diese Arbeit konzentriert sich auf relativ kleine Module wie sie z.B. in Mobiltelefonen oder Webcams zu finden sind. Ein typischer Aufbau ist nachfolgend gezeigt.

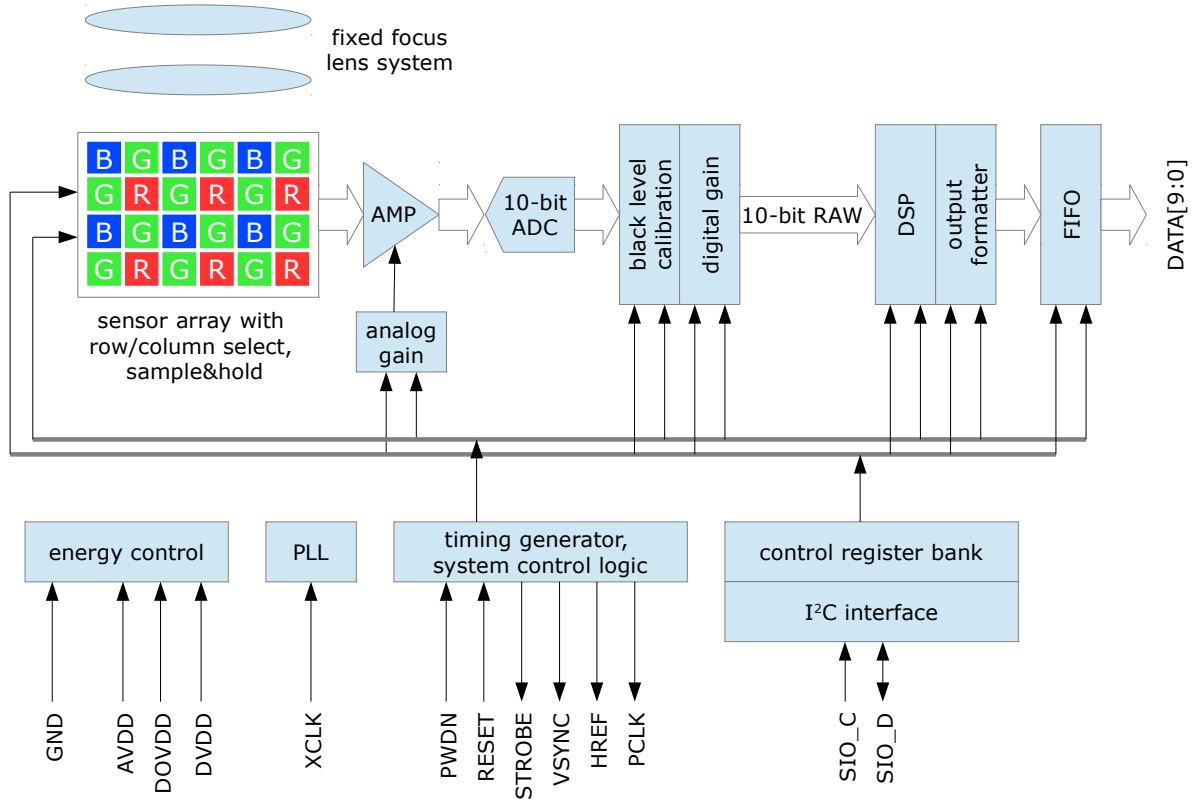

Abbildung 18: Aufbau eines typischen Kameramoduls

Kameramodule für Mobiltelefone werden oft mit einer einfachen Fixfocus-Optik ausgestattet. Die kleine Sensorgröße und die vergleichsweise geringen Qualitätsansprüche kommen diesem Aufbau entgegen. In letzter Zeit finden sich aber auch bei diesen kleinen Kameramodulen Varianten mit Autofocus-Objektiv. Eingesetzt werden diese z.B. in High-End Smartphones.

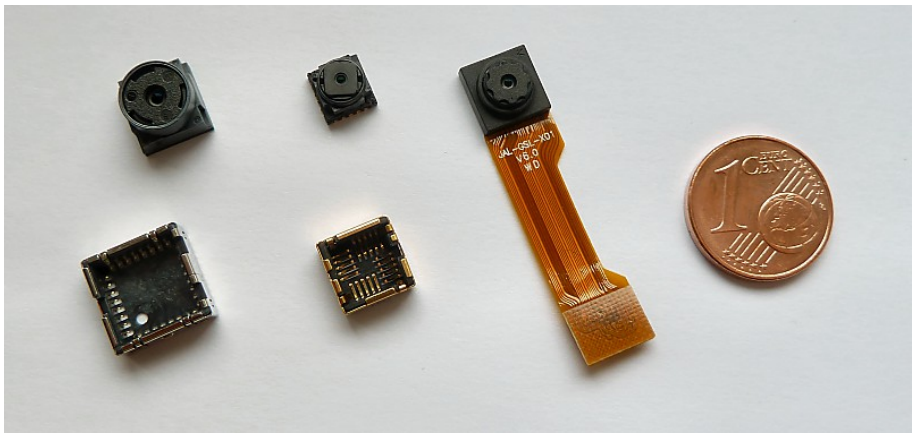

Abbildung 19: Bauformen von Kameramodulen
von links nach rechts: STM VS6624 1.3 Megapixel SXGA CMOS Sensor (8x8x6.5 mm) [41], Toshiba TCM8230MD 0.3 Megapixel VGA CMOS Sensor (6x6x5mm) [40] und OmniVision OV2655 2 Megapixel UXGA CMOS Sensor (8x8x4 mm) [35]; Die ersten beiden Kameras sind gesockelte Modelle, die Dritte verfügt über einen Flexkabelanschluss. Dieses Modell wird auch im Sensorprototypen verwendet.

4 Algorithmus zur Helligkeitsmessung

Zu Beginn der Arbeit war noch nicht klar wie die Aufgabenstellung, die Ermittlung eines korrekten Messsignals des Sensors auch bei ungünstiger Montage und anderen Störeinflüssen, gelöst werden soll. Klar war, das einfallende Licht soll in verschiedene Anteile zerlegt werden und nur solche, die das korrekte Helligkeitssignal bestmöglich wiedergeben, sollen für die Messung herangezogen werden.

Eine Überlegung war, die Lichtanteile, genauer die Messfelder, über eine einfache Facettenoptik zu trennen und je einer Photodiode zuzuführen. Die Diodensignale sollten dann mit einem noch zu entwerfenden Algorithmus ausgewertet werden.

Bei diesem Ansatz stellen sich einige Fragen: Wie viele Messfelder sind sinnvoll? Wie könnte ein Testaufbau aussehen? Wie groß könnte eine Einheit aus Optik+Photodioden gebaut werden. Welcher Hersteller könnte so etwas fertigen bzw. fertig liefern? Wie soll die Signalverarbeitung für Testaufbau und Prototyp aussehen?

All diese Fragen führten zum Entschluss eine digitale Fotokamera als Ausgangspunkt der Algorithmus-Entwicklung zu verwenden. Die Gestaltung der Messfelder ist flexibel, die Signal- bzw. Bildverarbeitung am PC ohne speziellen Hardwareaufbau zum Einlesen der Daten möglich und falls ein funktionierender Algorithmus gefunden wird, ist die Frage der Bauform und des Herstellers relativ einfach zu lösen: Ein Kameramodul könnte verwendet werden.

4.1 Photodiode vs. Bildsensor

Diskrete Photodioden zur Lichtmessung und Bildsensoren sind für komplett unterschiedliche Einsatzzwecke entwickelt worden. Obwohl beide Bauteile im Grunde nur Lichthelligkeit messen, sind die Unterschiede doch beträchtlich. Zu klären bleibt also die Frage, ob eine Fotokamera grundsätzlich zur Licht- bzw. Helligkeitsmessung verwendet werden kann. Das versucht dieser Abschnitt zu beantworten.

4.1.1 Dynamikumfang

Im Zumtobel LSD-Lichtsensor wird zur Helligkeitsmessung eine spezielle, zu diesem Zweck hergestellte Photodiode verwendet. Durch die große lichtaktive Fläche von 7,5 mm^2 kann das Bauteil Beleuchtungsstärken von 0,01 bis 10000 Lux linear erfassen [14]. Der LSD-Sensor ist so eingestellt, dass ein Maximalwert von 5000 Lux ausgegeben wird. Die in Kapitel 3.3.2 gezeigte Emp-

findlichkeitskurve einer Photodiode ist genau jene der im LSD-Sensor verwendeten Type Vishay BPW21R.

Im Gegensatz dazu besteht ein Bildsensor aus einer Matrix von vergleichsweise winzigen lichtempfindlichen Elementen. Die Größe der Photodioden von CMOS-Sensoren befinden sich oft im Bereich von wenigen Mikrometern. 1,75 x 1,75 µm Pixelgröße beispielsweise beim Bildsensor OV2655 [35]. So kleine Strukturen erreichen nicht den Dynamikumfang von diskreten Bauelementen. Wie dieses Problem für die Helligkeitsmessung gelöst werden kann ist weiter unten im Abschnitt 4.4 HDR-Bilder gezeigt.

4.1.2 Spektrale Empfindlichkeit

Eine weitere Fragestellung betrifft die spektrale Empfindlichkeit des Sensorelementes. Die Vishay BPW21R Photodiode ist V(λ)-korrigiert. D.h. das Empfindlichkeits-Maximum liegt bei etwa 550 nm - 570 nm Wellenlänge. Erreicht wird das durch eine spezielle Beschichtung, welche hauptsächlich grünes Licht durchlässt und andere Wellenlängen, insbesondere im IR-Bereich, blockt.

Bei Bildsensoren sitzen Farbfilter vor den einzelnen Pixeln (vgl. Kapitel 3.4.4). Das daraus resultierende RGB-Signal repräsentiert die Helligkeit der einzelnen Farben. Ein Gesamt-Helligkeitssignal kann aus dem RGB-Signal entweder aus der Grün-Komponente oder aus allen drei Farbkomponenten gewonnen werden (vgl. Kapitel 3.4.5). Nachstehende Abbildung zeigt die Verhältnisse.

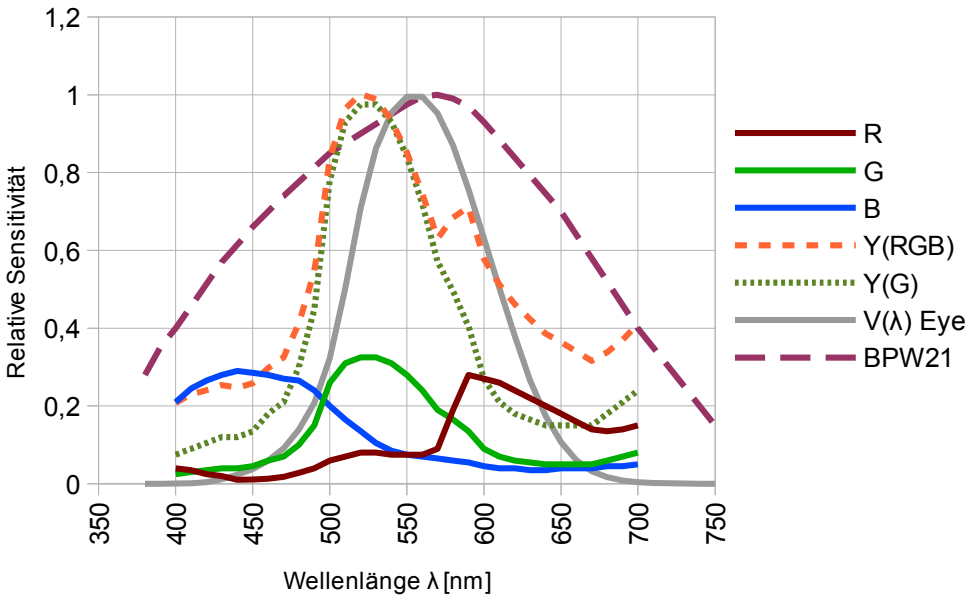

Abbildung 20: Spektrale Empfindlichkeiten im Vergleich

Die Daten der RGB-Empfindlichkeitskurven in Abbildung 20 sind angenähert und stammen aus mehreren Datenblättern. Verschiedene Bildsensoren haben deutlich unterschiedliche spektrale Empfindlichkeiten. Der qualitative Kurvenverlauf ist aber meist ähnlich. Die BPW21-Kurve stammt ebenso wie die V(λ)-Kurve aus dem Datenblatt der Photodiode [14].

Die Kurven Y(G) und Y(RGB) wurden über die Beziehung aus Kapitel 3.4.4 ermittelt. Die Faktoren sind für Y(G): K_R=0, K_G=1, K_B =0 und für Y(RGB): K_R=0,3, K_G=0,59, K_B =0,11. Für die Darstellung sind die Kurven auf den Maximalwert 1 skaliert.

Keine Empfindlichkeitskurve, weder die der Photodiode, noch die errechneten aus den Bildsensordaten entsprechen genau der V(λ)-Kurve. Die Näherungen sind aber gut genug um den Helligkeitseindruck für die menschliche Wahrnehmung ausreichend genau wiederzugeben.

4.1.3 Winkelabhängigkeit

Auch der Einfallswinkel des Lichts hat einen Einfluss auf das Messsignal. Das Signal ist am größten wenn das auftreffende Licht normal zur lichtaktiven Fläche einfällt. Dieses Phänomen trifft sowohl auf Photodioden als auch auf Bildsensoren zu.

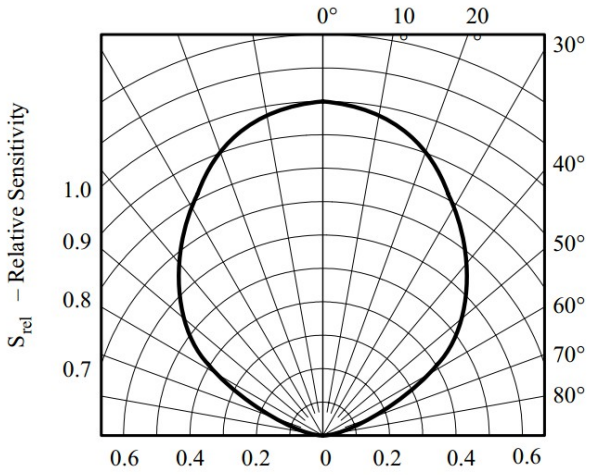

Abbildung 21: Winkelabhängigkeit der Lichtempfindlichkeit bei Photodiode BPW21 [14]

Bei Bildsensoren wird der Einfallswinkel des Lichtes als Chief Ray Angle (CRA) bezeichnet. Wie bei diskreten Photodioden sinkt die Sensibilität mit dem Winkel. Manche Hersteller geben in den Datenblättern Auskunft zum CRA (z.B. in [35]), andere nicht.

Wichtig ist der CRA vor allem für die Objektiv-Hersteller. Das Objektiv vor dem Bildsensor hat die Aufgabe den Strahlengang so zu korrigieren, dass das Licht der Abbildung möglichst normal

auf diesen auftrifft. Gelingt das nicht ausreichend entstehen Abschattungen an den Bildrändern. D.h. die Helligkeit des Bildes fällt ausgehend vom Zentrum zu den Rändern hin ab.

Dieser Bildfehler kann relativ einfach nachträglich korrigiert werden, indem die Signalverstärkung zu den äußeren Pixeln hin schrittweise angehoben wird. Viele Bildsensoren bzw. Kameramodule verfügen bereits intern über diese Korrekturmöglichkeit. Die Funktion wird oft als Lens Shading Compensation [41] oder einfach Lens Correction [35] bezeichnet.

4.1.4 Bildwinkel

Der Bildwinkel beschreibt jenen Bereich einer Szene, welcher durch das Kamerasystem abgebildet werden kann. Da ein Bildsensor meist ein rechteckiges Format hat, unterscheiden sich der horizontale und vertikale Winkel entsprechend.

Um vergleichbare Ergebnisse zu erhalten, muss der Bildwinkel der Kamera dem des LSD-Lichtsensors angepasst werden. Beim LSD-Lichtsensor beträgt der Winkel 60° horizontal und 40° vertikal (vgl. Kapitel 2.1.1). Dies kann durch Verwendung einer passenden Optik, korrekter Einstellung des Zooms oder dem Weglassen von Bildinformation an den Rändern erreicht werden.

4.1.5 Schlussfolgerung

Trotz aller Unterschiede zwischen diskreter Photodiode und Bildsensor sollte es möglich sein, ein für die Lichtmessung ausreichend genaues Helligkeitssignal mit einem Kamerasystem zu gewinnen. Das größte Problem ist der geringe Dynamikumfang des Bildsensors. Durch die verschiedenen Ausleseprinzipien, stetig bei der Photodiode, periodisch über eine definierte Belichtungszeit beim Bildsensor, können mehrere Belichtungsstufen erstellt und so der Dynamikumfang vergrößert werden.

In anderen Bereichen hat ein Bildsensor Vorteile gegenüber einer Photodiode. Die spektrale Empfindlichkeit kann durch unterschiedliche Gewichtung der Farbanteile angepasst und die Winkelabhängigkeit des Lichteinfalls kann elektronisch kompensiert werden.

Der größte Vorteil ist aber die vollkommen freie Bild-Nachbearbeitung zur Gewinnung gewichteter Lichtanteile und so eine Verbesserung des Messsignals.

4.2 Testsetup

Zur Gewinnung von Testdaten und zur Überprüfung der Annahmen zur Helligkeitsmessung mittels Kamera wurde ein Testaufbau angefertigt. Die Details sind in diesem Abschnitt beschrieben.

4.2.1 Wahl der Kamera

Die Aufgabenstellung verlangt nach einer Kamera, die über einen längeren Zeitraum periodisch Bilder mit definierten Belichtungseinstellungen aufnehmen kann. Daher ergibt sich folgendes Anforderungsprofil:

- Möglichkeit zur Fernsteuerung über PC
- Einstellmöglichkeit von Empfindlichkeit, Blende, Belichtungszeit
- Kamerabetrieb über einen längeren Zeitraum (> 1 Tag)
- Speicherung der Bilddaten auf dem PC
- Gute Bildqualität, insbesondere im Bezug auf die Genauigkeit der Belichtungseinstellungen; Die Bildauflösung ist hingegen nicht von Bedeutung

Diese Anforderungen sind nicht so einfach zu erfüllen wie anfangs angenommen. Meist verfügen nur teure digitale Spiegelreflexkameras mit passendem Zubehör über alle Möglichkeiten. Kompaktkameras bieten zwar eine für die Aufgabenstellung mehr als ausreichende Bildqualität und lassen oft auch manuelle Belichtungseinstellungen zu, eine Fernsteuermöglichkeit fehlt aber meistens.

Eine Ausnahme bilden älter Modelle der Fa. Canon. Bis vor einigen Jahren hatten viele höherwertige Kompaktkameras die Möglichkeit zur Fernsteuerung via PC. Bei jüngeren Modellen wurde dieses Feature wie bei anderen Herstellern eingespart. Findige Hacker haben aber eine Möglichkeit gefunden in die Kamera-Firmware einzugreifen und diese und andere Funktionen wieder freizuschalten. Zusammengefasst sind die als Canon Hack Development Kit (CHDK) bezeichneten Entwicklungen auf einer Webseite, welche in [46] zu finden ist.

Für das Testsetup konnte günstig eine ältere Kamera vom Typ Canon S30 erstanden werden, welche alle oben genannten Anforderungen erfüllt. [45]

4.2.2 Referenzmessung mit LSD-Lichtsensoren

Im ersten Schritt der Algorithmus-Entwicklung soll gezeigt werden, dass die Lichtmessung mit einer Kamera dasselbe Signal liefern kann wie das der bestehenden LSD-Lichtsensoren auf Basis der speziellen Photodiode.

Zu diesem Zweck müssen mehrere LSD-Sensoren an verschiedenen Stellen platziert und die Signale zeitgleich mit den Kamerabildern aufgezeichnet werden. Die so gewonnenen Signale dienen dann als Referenz für das Kamerasignal.

Die Auswertung des Messsignals aus der 4-20 mA Schnittstelle der LSD-Sensoren kann einfach über den Spannungsabfall an einem Messwiderstand in eine für einen ADC verwertbare Spannung umgewandelt werden. Eine später auch für den Phototypenaufbau verwendete Mikrocontrollerplatine erfüllt diese Anforderungen und wurde entsprechend beschaltet und programmiert. [53]

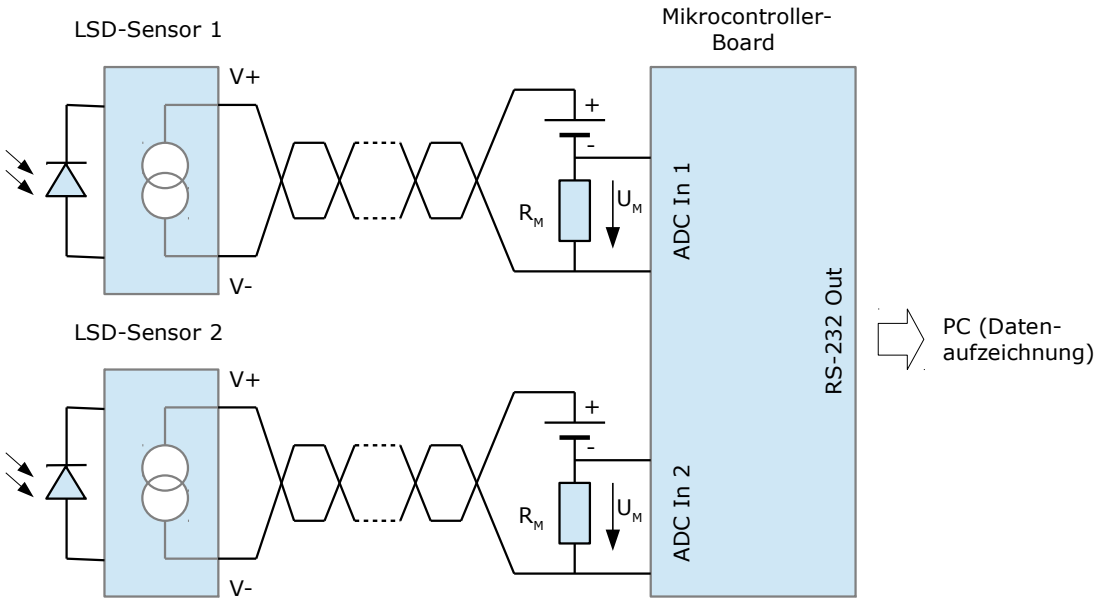

Abbildung 22: LSD-Sensor Auswerteschaltung und Datenaufzeichnung

Die ADC-Eingänge des Mikrocontrollers erhalten über einen Messwiderstand R_M von 56 Ω die Werte aus der Stromschleife. Je nach Schleifenstrom entsteht ein Spannungsabfall von 0,224 bis 1,12 V. Die Referenzspannung des ADCs wurde auf 1,25 V (interne Referenz) festgelegt, wodurch die Messwerte gut in den Messbereich passen.

Der Mikrocontroller digitalisiert die Messwerte mit einer Genauigkeit von 12 Bit (4096 Werte), was bei einem Messbereich des Sensors von 0-5000 Lux, abgebildet auf die Spannung 0,224-1,12 V und einer Referenzspannung von 1,25 V etwa den Wertebereich von 734-3670 nach der Digitalisierung entspricht. Dadurch entsteht eine Auflösung von ca. 1,7 Lux/Digit.

Übertragen wird der Messwert über die am Mikrocontroller-Board befindliche RS232-Schnittstelle. Auf PC-Seite nimmt ein kleines Programm den Messwert entgegen und speichert ihn für die spätere Verwendung.

4.2.3 Aufbau

Für den Aufbau der Versuchsanordnung dient ein gewöhnliches, ins Freie gerichtete Fenster als Ausgangspunkt. Ein LSD-Lichtsensor ist korrekt, nach Montagevorschrift platziert oberhalb des Fensters angebracht. Dieser Sensor liefert das Referenzsignal.

Weiter hinten im Raum, ebenfalls an der Decke befindet sich die Kamera und direkt daneben ein weiterer LSD-Lichtsensor, welcher ein zweites Referenzsignal liefert. Dieses zweite Referenzsignal entspricht nicht dem korrekten Messsignal das für eine Beleuchtungssteuerung verwendet werden kann, allerdings kann damit der Messwert der Kamera mit dem Messwert des LSD-Sensors verglichen werden. Ziel ist es sowohl den Messwert des korrekt montieren Sensors als auch den Messwert des zweiten Sensors aus dem Kamerasignal ableiten zu können. Nachfolgende Abbildungen zeigen den Aufbau.

Abbildung 23: Messaufbau - Lichtmessung mit Kamera
Der Pfeil markiert die LSD-Lichtsensoren oberhalb des Fensters

Abbildung 24: LSD-Lichtsensor neben der Kamera für die Referenzmessung

Abbildung 23 zeigt die oberhalb des Fensters an der Decke korrekt montierten LSD-Lichtsensoren. Auf dieser Abbildung sind noch beide Sensoren am Fenster montiert. Die Kamera befindet sich weiter im Rauminneren knapp unterhalb der Decke. Im Vordergrund ist der Laptop für die Datenaufzeichnung zu sehen. Abbildung 24 zeigt den 2. Sensor neben der Kamera nach dem Umbau.

4.3 Randbedingungen

Durch die technischen Gegebenheiten, verfügbare Kamera, Mikrocontroller, Datenübertragung, etc. sowie die Zielsetzungen Stromverbrauch, Bauform, Kosten usw. ergeben sich einige Randbedingungen, die die Entwicklung des Algorithmus, beeinflussen.

4.3.1 Bildauflösung

Im Hinblick auf die spätere Implementierung des Helligkeits-Messalgorithmus, auf einem Mikrocontroller mit sehr beschränkten Ressourcen, ist es vorteilhaft eine vergleichsweise geringe Bildauflösung zu verwenden. Angenommen der Mikrocontroller verfügt über 16 kB SRAM und es müssen gleichzeitig mehrere Bilder im RAM vorrätig gehalten werden, dann darf ein Bild nur wenige Kilobyte groß sein.

Durch eine geringe Bildauflösung entsteht noch ein weiterer Vorteil: Das Helligkeitssignal wird einer Tiefpassfilterung unterzogen. Hohe Ortsfrequenzen, welche sich durch hohe Kontraste bzw. Grauwert-Unterschiede ausdrücken, werden unterdrückt. Lokale Glanzpunkte im Bild, die z.B. durch Spiegelungen entstehen, können durch eine Mittelwertbildung minimiert werden. Solche Glanzpunkte tragen nicht wesentlich zur Raumhelligkeit bei, können den Messwert aber deutlich verfälschen.

Diese Überlegungen führen zu einer gewählten Auflösung von 40x30 Pixel. Daraus ergibt sich bei 8 Bit/Pixel ein Speicherverbrauch von 1,2 kB bzw. 2,4 kB bei 16 Bit/Pixel.

4.3.2 Datenformat

Die Bilddaten der ferngesteuerten Kamera werden am PC im bekannten JPG-Format abgelegt. Dieses Format verwendet intern ein YUV-Farbmodell, ähnlich dem in Kapitel 3.4.8 bzw. in [28] beschriebenen. Damit würde die Helligkeitskomponente des Signals bereits nativ vorliegen. In der Programmierumgebung (vgl. Kapitel 5.1.3) steht nach dem Einlesen eines JPG-Bildes jedoch eine Matrix aus RGB-Werten mit je 8 Bit Auflösung zur Verfügung. Mit dem in Kapitel 3.4.5 vor-

gestellten Verfahren wird daraus die Y-Komponente ermittelt. Diese Vorgangsweise scheint etwas umständlich, ist aber programmiertechnisch am einfachsten umsetzbar.

Das so gewonnene Y-Signal soll auch im Hinblick auf die spätere Umsetzung eines Prototypen mit Kameramodul und Mikrocontroller vergleichbar sein.

4.3.3 Messbereich

Der Dynamikumfang eines Bildsensors ist begrenzt. Der Messbereich des Helligkeitssignals soll aber einen Bereich von 0-5000 Lux abdecken können. Der Ausgabewert entspricht dabei in erster Näherung[10], der gemittelten, über das ganze Bild aufgenommenen Helligkeit. Einzelne Bildteile können dabei 5000 Lux deutlich überschreiten. Es ist deshalb unerlässlich den Dynamikbereich des Sensors zu erweitern. Der nachfolgende Abschnitt beschreibt das Verfahren.

4.4 HDR-Bilder

Als High-Dynamic-Range-Bild (HDR-Bild) wird ein Bild mit erhöhtem Dynamikumfang bezeichnet. Oft, so auch im vorliegendem Fall, wird das HDR-Bild aus einer Belichtungsreihe „gewöhnlicher" Low-Dynamic-Range (LDR)[11] Bilder ermittelt.

Eine Belichtungsreihe besteht aus zwei oder mehreren Bildern derselben Szene, aufgenommen mit unterschiedlichen Belichtungseinstellungen. Es ist dabei freigestellt ob Sensorempfindlichkeit, Blende oder Belichtungszeit variiert wird.

Aus den unterschiedlichen Aufnahmen kann Pixel für Pixel ein Bild mit erhöhtem Kontrastumfang errechnet werden. In einer als Gesamtheit überbelichteten Aufnahme sind dunkle Stellen korrekt belichtet, in unterbelichteten Aufnahmen hingegen helle Stellen wie z.B. Wolken am Himmel. So finden sich in jeder Belichtungsstufe Bildteile, die zum HDR-Bild beitragen können.

Für die Zusammenführung der Belichtungsstufen sind mehrere Verfahren bekannt. Im Grunde bauen alle darauf auf die Pixel-Helligkeitswerte zu linearisieren, auf den neuen Ausgabebereich zu skalieren und gewichtet nach Korrektheit des Messwertes zu verrechnen. Der Hauptunterschied liegt in der Wahl der Gewichtungsfunktion. In [32] sind die verschiedenen Möglichkeiten beschrieben. In dieser Arbeit wird eine einfache, in Hinblick auf eine Implementierung am Mikrocontroller optimierte, Gewichtungsfunktion verwendet.

Eine weitere Voraussetzung für die Gewinnung eines HDR-Bildes betrifft die Unbeweglichkeit von Szene und Kamera während der Aufnahme der Belichtungsreihe. Meist werden die Aufnah-

10 Später sollen die Pixel-Messwerte unterschiedlich gewichtet zum Messsignal beitragen.
11 Die Bezeichnung LDR (Low Dynamic Range) ist nicht zu verwechseln mit einem LDR-Photowiderstand.

men einer Belichtungsreihe (kurz) nacheinander angefertigt. Damit die einzelnen Pixel aus den unterschiedlichen Aufnahmen einwandfrei miteinander verrechnet werden können, sie repräsentieren ja den gleichen Bildpunkt mit unterschiedlichen Helligkeiten, dürfen sich weder Szene noch die Kamera bewegen. Jede Veränderung verschlechtert das Ergebnis. Da über die Zeit unveränderliche Szenen selten sind, wurden Verfahren entwickelt, die Veränderungen in den Einzelbildern tolerieren können. Auch hier sei wieder auf [32] und [50] verwiesen.

Für die Lichtmessung sind die Voraussetzungen, HDR-Bilder aufnehmen zu können, denkbar gut, da sich der Sensor fix montiert an der Decke befindet und eine (meist) unbewegte Szene fotografiert. Kleinere Bewegungen im Bild, z.B. von Bäumen oder Wolken, aber auch von vorbeigehenden Personen werden als Störgrößen akzeptiert und nicht weiter behandelt.

4.4.1 Linearisierung der Messwerte

Kameras bzw. Bildsensoren geben Helligkeitswerte nicht ideal linear wieder. Diese Nichtlinearitäten sind gewollt vom Hersteller eingefügt, um einen, speziell auf den Aufnahmechip abgestimmten, möglichst natürlichen Bildeindruck zu vermitteln. Für die HDR-Bildberechnung werden lineare Helligkeitswerte benötigt um die einzelnen Belichtungsstufen zusammensetzen zu können.

Die Kurvenform der so genannten Kamerakurve hat meist einen Gammakurven ähnlichen Verlauf. Das ursprünglich lineare Helligkeitssignal des Bildsensors wird so der logarithmischen Helligkeitsempfindung des menschlichen Auges angepasst. Siehe dazu auch die Kapitel 3.4.6 und 3.1.2.

Der Kurvenverlauf der Kamerakurve kann von einer idealen Gammakurve abweichen und für jede Farbkomponente unterschiedlich aussehen. Im Allgemeinen kann eine inverse Gammafunktion, die Gammakorrektur die Werte ausreichend gut linearisieren.

Kamerakurven sind normalerweise von den Herstellern nicht dokumentiert. Daher muss die gesuchte Kurve zunächst aus einer Sammlung von Bilddaten rekonstruiert werden. Hintergrundinformationen dafür sind in [32] und in [48] zu finden. Für diese Arbeit genügt es die Kurve für die eingesetzte Kamera zu ermitteln. Dabei hilft die frei erhältliche Software HDR Shop [47].

In Abbildung 25 ist die rekonstruierte Kurve der Canon S30 zu sehen. Es handelt sich dabei bereits um die Korrekturfunktion. Die so gefundene Kamerakurve kann für sämtliches Bildmaterial der betreffenden Kamera zur Linearisierung verwendet werden.

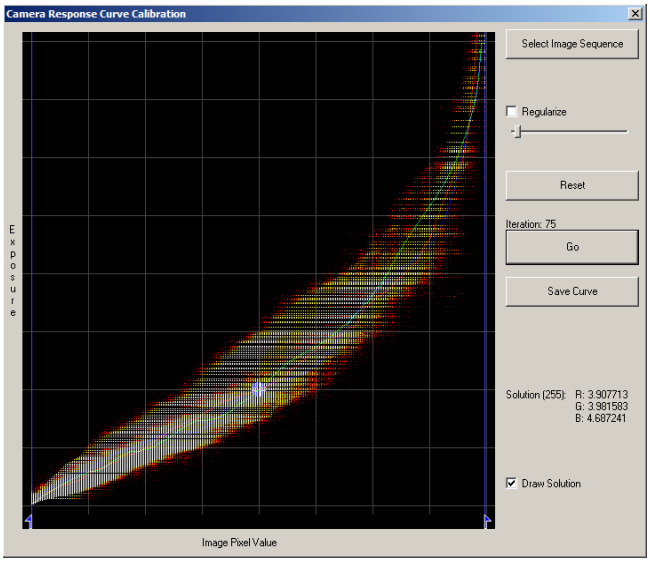

Abbildung 25: Kamerakurve für Canon S30; Ermittelt mit der Software HDR-Shop [47]

4.4.2 Skalierung und Gewichtung

Nach Ermittlung der linearen Helligkeitswerte der Bilddaten müssen diese passend auf den Zielbereich skaliert werden. Jede Belichtungsstufe benötigt dazu einen eigenen Skalierungsfaktor. Dadurch werden die Werte vergleichbar und können miteinander verrechnet werden.

Im konkreten Fall werden die Daten auf die Lux-Skala abgebildet. Es wird davon ausgegangen, dass die Daten der LDR-Bilder als Schwarzweiß-Bild mit 8 Bit/Pixel vorliegen. Das HDR-Zielbild soll 16 Bit/Pixel umfassen. Der Minimalwert wird mit 0 Lux und der Maximalwert mit 10000 Lux festgelegt. Damit ergibt sich eine Auflösung von 0,15 Lux/Digit. Beobachtungen haben gezeigt, dass einzelne Bildteile deutlich über 5000 Lux hell werden können auch wenn der Ausgabewert des Sensors sein Maximum von 5000 Lux noch nicht erreicht hat. 10000 Lux Maximalwert ist deshalb eine praktikable Größe.

Versuche haben ergeben, dass drei Belichtungsstufen, getrennt durch zwei F-Stops (vgl. Kapitel 3.5.2) genügen, den Dynamikumfang soweit zu erweitern um den gewünschten Messbereich abzudecken.

In den einzelnen Belichtungsstufen finden sich Bildteile, deren Messwerte sich überschneiden. Es muss daher irgendwie entschieden werden, welcher Messwert ins HDR-Bild aufgenommen wird. In der vorliegenden Implementierung übernehmen das einfache, stückweise stetige, Gewichtungsfunktionen, welche mit den Messwerten multipliziert werden. Die Gewichtungsfunktionen der LDR-Bilder überlappen sich dabei in den Übergangsbereichen. Es muss sichergestellt sein,

dass der Gesamt-Gewichtungsfaktor an jeder Stelle 1 beträgt.

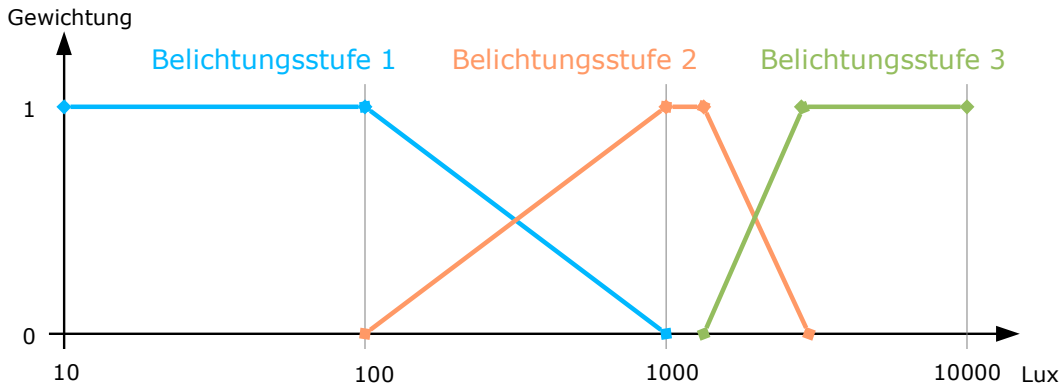

Abbildung 26: Gewichtungsfunktionen der Belichtungsstufen

Zu klären bleibt noch die Frage, wie die Skalierungsfaktoren und Gewichtungsfunktionen ermittelt werden. Hier kommt die Referenzmessung des LSD-Lichtsensors direkt neben der Kamera ins Spiel. Die Parameter werden mit Hilfe des eigens dafür entwickelten Lichtsensor-Testprogramms (siehe Anhang, Abschnitt 6.2) solange gezielt verändert, bis das Signal des LSD-Lichtsensors mit dem aus dem HDR-Bild übereinstimmt. Da beide Sensoren praktisch an derselben Stelle messen, muss auch das Helligkeitssignal übereinstimmen.

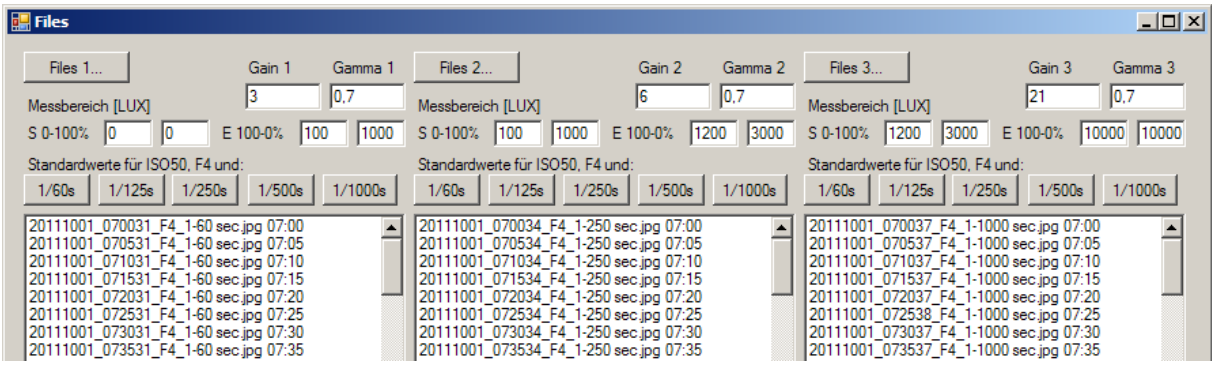

Abbildung 27: Parametereinstellung für HDR-Bildberechnung

4.4.3 Signalverarbeitung

Die nachfolgende Darstellung fasst den gesamten Prozess HDR-Bildberechnung noch einmal zusammen.

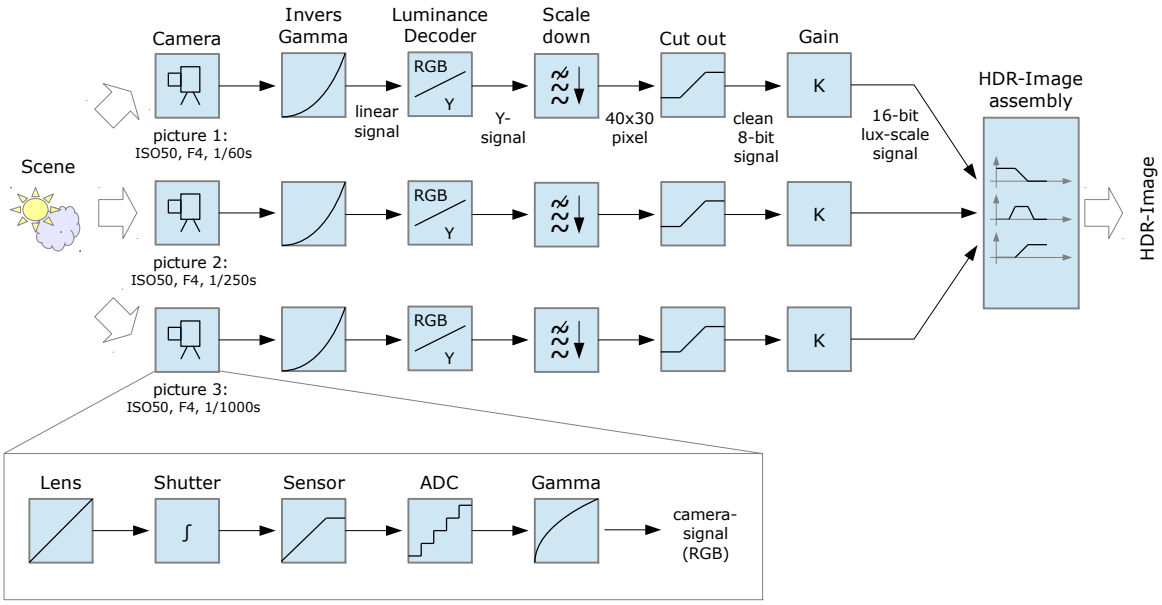

Abbildung 28: Signalverarbeitung zur HDR-Bildberechnung

Die Einzelbilder der Belichtungsstufen werden zuerst einer Gammakorrektur unterzogen, in Grauwerte umgerechnet und auf die Zielauflösung von 40x30 Pixel skaliert. Die so entstandenen 8-Bit Helligkeitsbilder durchlaufen eine Filterung, bei der Werte an den Rändern des Wertebereichs herausgenommen werden. Da die Skalierung durch eine Mittelwertbildung der Pixel des hochauflösenden Bildes erfolgt, kann angenommen werden, dass in Randwerten viele Informationen von über- bzw. unterbelichteten Pixeln enthalten sind. Das verfälscht den Messwert weil die Größe der Fehlbelichtung unbekannt ist. Als Threshold-Werte dienen 32 und 224.

Im nächsten Schritt werden die Helligkeitswerte auf die Lux-Skala skaliert und gewichtet über die passende Funktion in das HDR-Bild eingetragen. So entsteht aus mehreren 8-Bit Graustufenbildern ein 16-Bit HDR-Bild.

Abbildung 29: HDR-Bildberechnung
oben: Belichtungsreihe der Szene; unten: linearisiere und skalierte Helligkeitsbilder sowie fertiges HDR-Bild

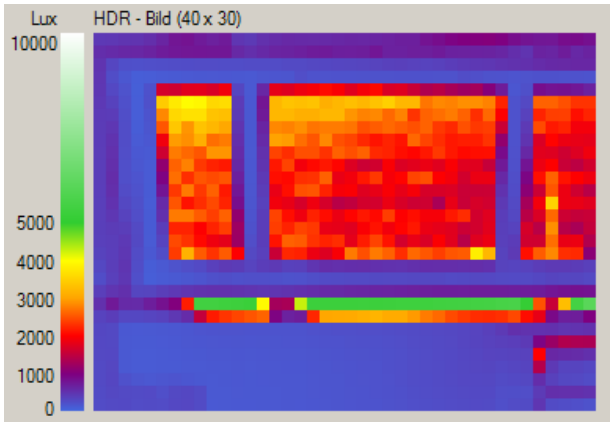

Abbildung 30: HDR-Bild in Falschfarbdarstellung mit Lux-Skala

4.4.4 Lichtmessung mit den HDR-Bildern

Nachdem ein HDR-Bild vorliegt, soll überprüft werden, wie gut damit die Lichthelligkeit im Vergleich zum LSD-Lichtsensor gemessen werden kann. Es wird davon ausgegangen, dass sowohl die Kamera als auch der Lichtsensor etwa den gleichen Blickwinkel haben und damit die Lichtmenge einfangen. Das Helligkeitssignal wird als Mittelwert aller Pixel aus dem HDR-Bild berechnet.

Die nachstehende Abbildung zeigt die Helligkeitsverläufe von Kamera und Lichtsensor über einen Zeitraum von mehreren Tagen. Man erkennt die gute Übereinstimmung der Messwerte.

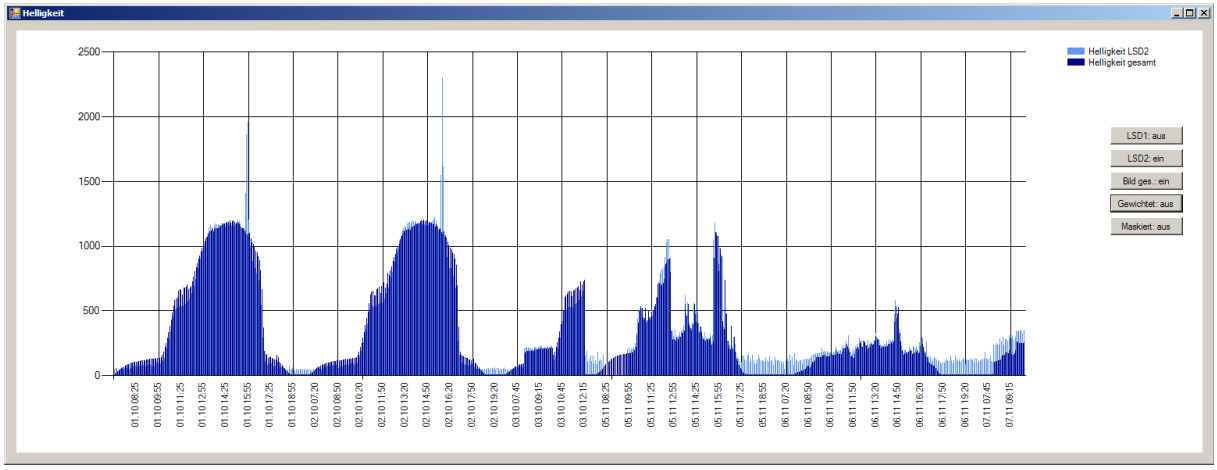

Abbildung 31: Helligkeitsverlauf über etwa 5 Tage (ohne Nachtstunden)
Hellblau: Messwert LSD-Lichtsensors neben der Kamera; Dunkelblau: Messwert aus dem HDR-Bild

Die kleineren Abweichungen der Messwerte können aufgrund der noch vorhandenen Nichtlinearitäten im Helligkeitssignal und aufgrund der nicht ganz deckungsgleichen Montage des LSD-Lichtsensors und der Kamera erklärt werden. Ganz kleine Messsignale (<100 Lux) werden nicht mehr korrekt erfasst. Um diese Signale auch zu erfassen, müsste noch eine Belichtungsstufe mit längere Belichtungszeit in die HDR-Bildberechnung einfließen.

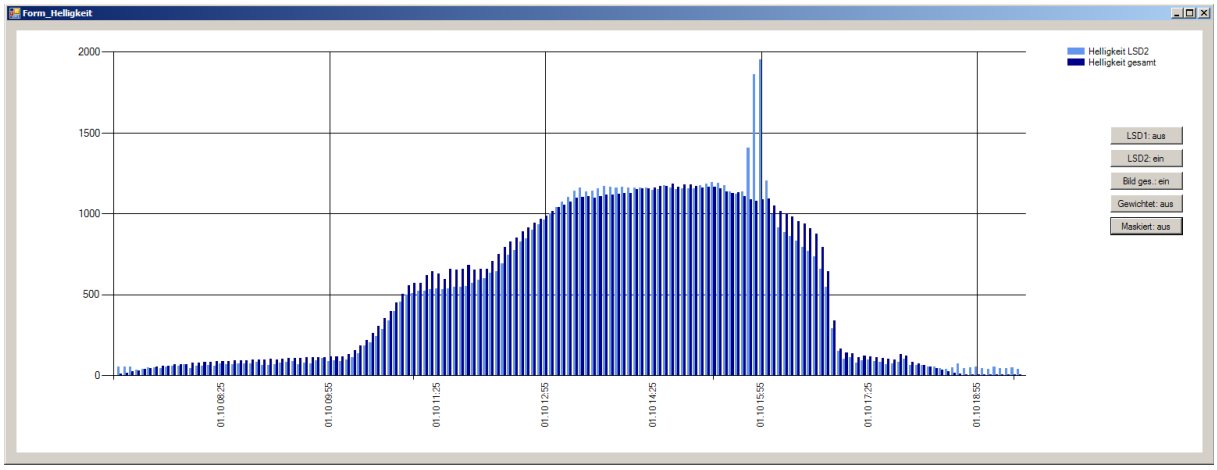

Abbildung 32: Helligkeitsverlauf wie oben, erster Tag (Alle Messwerte in Lux)

Die großen Ausreißer im Messwert, jeweils um ca. 16:00, entstehen aufgrund einer Spiegelung an der Fensterbank. Die Spiegelung ist im Kamerabild nur an wenigen Pixeln zu sehen. Diese Pixel sind in allen Belichtungsstufen übersteuert und liefern daher einen zu geringen Messwert. Aufgrund der Mittelwertbildung verschwindet der Peak im Gesamtsignal der Kamera.

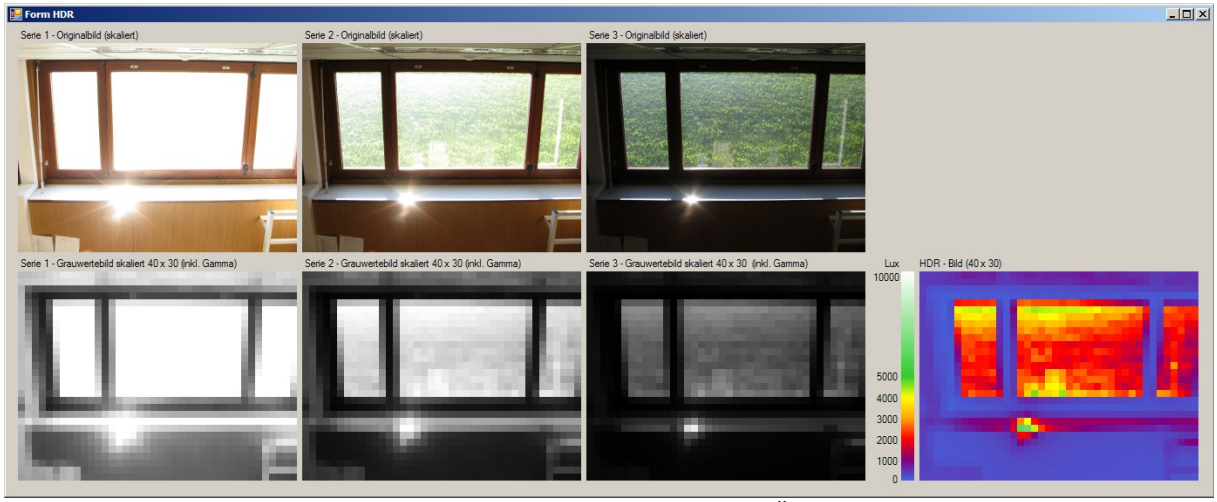

Abbildung 33: Spiegelung an der Fensterbank (ca. 16:00): Übersteuerung einzelner Pixel

Wird die Spiegelung von den Sensoren nicht mehr erfasst, weil diese durch den Sonnenverlauf außerhalb der Blickfelder gewandert ist, gleichen sich die beiden Signale wieder an.

Das reflektierte Licht der Spiegelung trägt nur geringfügig zur Raumhelligkeit bei.[12] Angenommen die Raumbeleuchtung wird aufgrund des kurzfristig hohen Helligkeitssignals des LSD-Sensors stark gedimmt, so ergibt sich eine zu geringe Raumhelligkeit. In dieser Situation liefert die Kameramessung ein besseres Signal als der (nicht nach Vorschrift montierte) LSD-Lichtsensor.

4.5 Gewichtung der Helligkeitsmesswerte

Durch das HDR-Bild kann die Helligkeit an jeder Stelle über den vollen Messbereich ermittelt werden. Der Mittelwert über alle Pixel entspricht dem Messsignal des LSD-Lichtsensors neben der Kamera. Um von der Kameraposition ein Signal zu erhalten, das dem korrekt oberhalb des Fensters montierten Lichtsensors entspricht, müssen die Pixel-Messwerte passend gewichtet werden.

Um solch einen Helligkeits-Gewichtungsalgorithmus zu entwickeln, wird von folgenden Annahmen ausgegangen:

- Der gesuchte Messwert wird (hauptsächlich) durch die Pixel der Fensterfläche repräsentiert. Das kann durch die Montagevorschriften des LSD-Lichtsensors argumentiert werden, da dieser nur auf offene Fensterflächen blicken darf.

12 Ergebnis eigener Beobachtungen über mehrere Wochen. Zumindest für den betrachteten Raum kann diese Aussage getroffen werden.

- „Gleichartige Lichtanteile verhalten sich gleichartig“ Das bedeutet, Pixel, die Tageslichtanteile repräsentieren, folgen mehr oder weniger dem Verlauf des (Gesamt-)Tageslichtes. Pixel, die mit Kunstlicht beaufschlagt sind, folgen der Kunstlichtkurve.
- Durch Beobachtung der Pixelwerte über einen längeren Zeitraum können die einzelnen Lichtklassen voneinander isoliert werden.

Wenn diese Annahmen stimmen, kann eine Gewichtungsmatrix erstellt werden, mit deren Hilfe ein korrektes Messsignal gewonnen werden kann.

4.5.1 Beobachtung des Lichtverlaufes

Die Beobachtung des Helligkeitsverlaufs über einen längeren Zeitraum zeigt auf, welche Bildteile starken Änderungen unterworfen sind und welche Bildteile vergleichsweise konstant bleiben. Durch die Berechnung der Änderungen in drei Skalen, Dezibel dB, Prozent % und Absolutwert in Lux treten diese Änderungen noch deutlicher hervor.

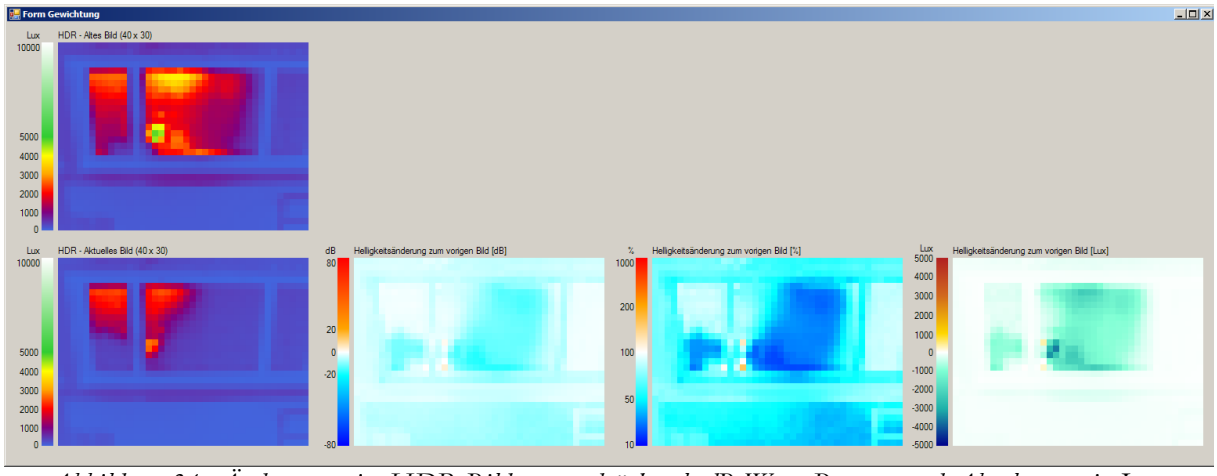

Abbildung 34: Änderungen im HDR-Bild , ausgedrückt als dB-Wert, Prozent und Absolutwert in Lux

Abbildung 34 zeigt ganz deutlich, dass die größten Helligkeitsänderungen im Bereich der Fensterfläche stattfinden. Am deutlichsten treten die Unterschiede in der Absolutwertskala heraus. Während Fensterrahmen und Innenflächen beinahe die gleiche Färbung aufweisen, sind Teile der Fensterfläche deutlich hervorgehoben. Aus diesem Grund werden nachfolgend alle Berechnungen des Helligkeits-Gewichtungsalgorithmus, in Absolutwerten durchgeführt. Eine Umrechnung in dB oder Prozent ist nicht notwendig.

4.5.2 Gewichtungsmatrix

Die Errechnung des korrekten Messwerts geschieht über die so bezeichnete Gewichtungsmatrix.

Pro Pixel wird ein Wert gespeichert, mit welchem der jeweilige Helligkeits-Messwert multipliziert wird. Die Summe dieser Produkte ergibt den Gesamt-Messwert. Aus diesem Grund muss die Summe aller Elemente der Matrix immer 1 betragen.

Initialisiert wird jedes Element der Gewichtungsmatrix mit dem Wert 1/(Anzahl der Pixel). Bei 40x30 Pixel Bildgröße ist das 1/1200 oder 0,000833. Damit sind alle Pixel gleich gewichtet und eine Anwendung auf das HDR-Bild entspricht einer Mittelwertbildung.

Für die Implementierung am Mikrocontroller stellt die Größe der Gewichtungsmatrix ein gewisses Problem dar. Die Multiplikatoren müssen als Fließkommazahl gespeichert werden um eine ausreichende Genauigkeit zu erreichen. Bei float-Werten sind das 4 Byte/Pixel was in einer Speichergröße von 4,8 kB resultiert. Zusammen mit drei LDR-Bildern (je. 1,2 kB) und einem HDR-Bild (2,4 kB) kann das den kleinen Speicher eines Mikrocontrollers stark belasten.

4.5.3 Algorithmus

Der Gewichtungsalgorithmus geht von der oben formulierten Annahme des gleichartigen Verhaltens gleichartiger Lichtanteile aus. Versuche zeigen, dass folgende Implementierung Fensterflächen recht zuverlässig finden kann.[13]

1. Gewichtungsmatrix initialisieren (Faktor = 1/1200)
2. Ermittlung des Gradienten der Helligkeitsänderung vom Gesamtbildes
   (bezogen auf des vorhergehende Bild)
3. Schleife über alle Pixel
   - Vergleich des aktuellen Pixelwertes mit dem Gradienten
     - Falls der Pixelwert NICHT in der Nähe des Gradienten liegt (zwischen 0,1*Gradient und 10*Gradient): gutes Pixel: Pixelgewichtung erhöhen (Faktor 1,1)
       Andernfalls: schlechtes Pixel: Gewichtung verringern (Faktor 0,91)
   - Minimales und maximales Pixelgewicht beschränken, damit einzelne Pixel nicht die gesamte Gewichtung an sich ziehen können und niedrig gewichtete Pixel eine Chance haben wieder an Gewicht zu gewinnen falls diese „gutes“ Tageslicht repräsentieren.
     Maximalgewicht: 0,003 (0,3%), Minimalgewicht: 0,00001 (0,001%)

13 Die Implementierung dieses Algorithmus im Sourcecode ist im Anhang, Abschnitt 6.1.2 zu finden.

4. Gewichtungsmatrix normalisieren (Gesamtgewicht der Matrix auf 1 setzen)

Die folgenden Abbildungen zeigen den Aufbau der Gewichtungsmatrix über einige Stunden:

Abbildung 35: Neu initialisierte Gewichtungsmatrix; Alle Pixel sind gleich gewichtet

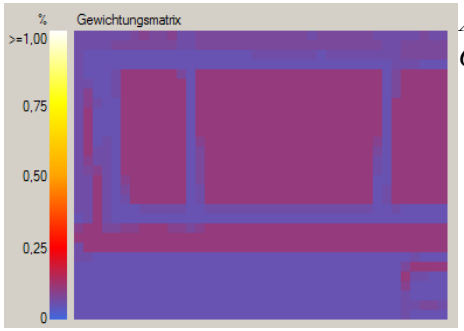

Abbildung 36: Gewichtungsmatrix nach 1h

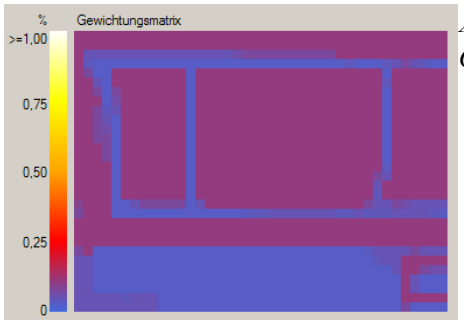

Abbildung 37: Gewichtungsmatrix nach 2h

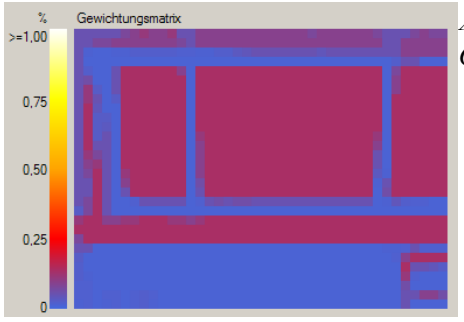

Abbildung 38: Gewichtungsmatrix nach 3h

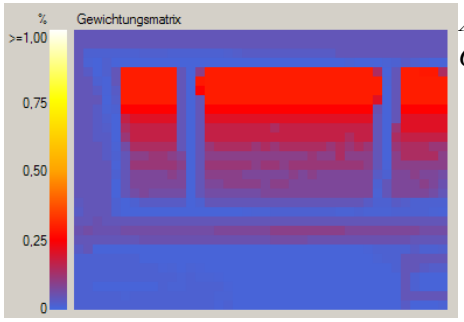

Abbildung 39: Gewichtungsmatrix nach 4h

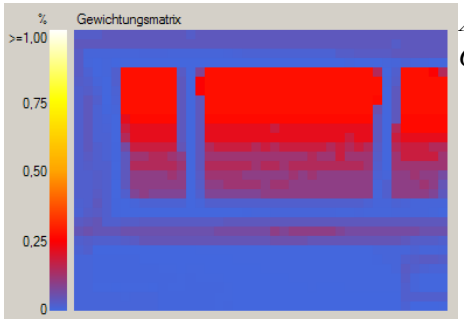

Abbildung 40: Gewichtungsmatrix nach 5h

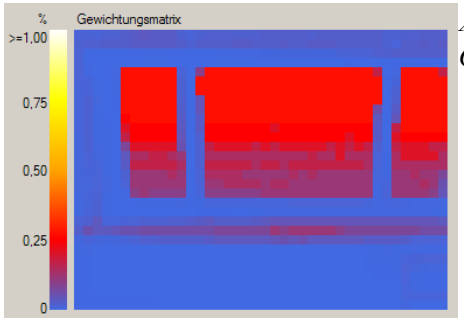

Abbildung 42: Gewichtungsmatrix nach 6h

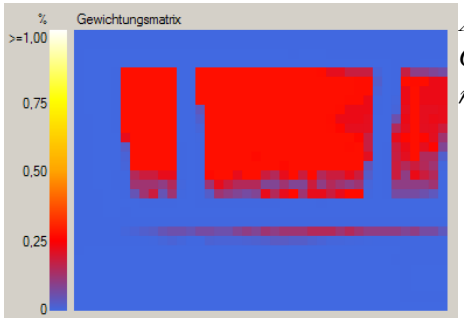

Abbildung 43: Gewichtungsmatrix nach mehreren Tagen

Bereits nach einigen Stunden treten die Fensterflächen immer deutlicher hervor. Die anfangs noch vorhandenen Gewichtungen an den Innenwänden werden über die Zeit immer weiter zurückgedrängt. Nach einigen Tagen sind falsch gewichteten Stellen fast vollständig verschwunden.

Bemerkenswert ist auch, dass das Zeitintervall zwischen den einzelnen Aufnahmen für die Bildung der Gewichtungsmatrix nur eine untergeordnete Rolle spielt. Im vorliegendem Beispiel beträgt das Intervall 5 Minuten. Intervalle mit 1 Minute oder 20 Sekunden führen zum selben Ergebnis.

Auch der Startzeitpunkt der Messung hat keinen Einfluss auf das Endergebnis. Implementierungsbedingt wird die Gewichtungsmatrix hauptsächlich in Abschnitten steigender Helligkeit aufgebaut, z.B. von Morgen bis Mittag. In Zeiten abnehmender Helligkeit, kommt es nur zu wenigen Änderungen in den Pixelgewichtungen. Wird beispielsweise der Startzeitpunk der Messreihe nach dem Tagesmaximum auf den Nachmittag gelegt, so ändert sich die Gewichtungsmatrix in den ersten Stunden kaum und der Aufbau beginnt am folgenden Tag. Das bedeutet nichts anderes als ein spät am Tag montierter Sensor braucht etwas länger um sich auf die jeweilige Lichtsituation einzustellen.

In einem Punkt entspricht der beschriebene Algorithmus nicht ganz der formulierten Annahme über das gleichartige Verhalten der Lichtanteile: Verglichen wird der Pixel-Helligkeitswert mit dem Gradienten der Gesamthelligkeit. Wenn diese Werte nicht nahe beisammen liegen wird das betreffende Pixel höher gewichtet. Eigentlich müsste lt. Annahme der Pixelgradient mit dem Gesamtgradienten verglichen werden. Wird dies so implementiert entsteht folgende Gewichtungsmatrix:

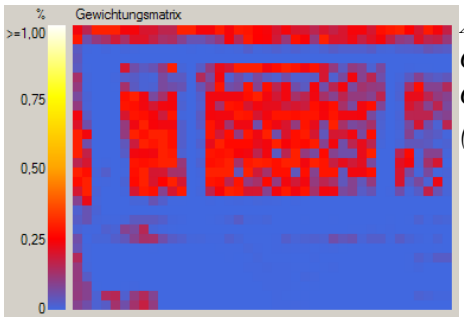

Abbildung 44: Gewichtungsmatrix Vergleich Gesamtgradient mit Pixelgradient (Ergebnis nach 2,5 Tagen)

Das Ergebnis ist auch relativ gut, verglichen mit der vorigen Version sind aber an vielen Stellen Innenraumflächen deutlich zu hoch gewichtet. Daher wird für die weiteren Untersuchungen die erste Variante verwendet.

4.5.4 Messungen

Die nachfolgenden Abbildungen zeigen Messergebnisse aufgrund der ermittelten Gewichtungsmatrix im Vergleich zur berechneten Gesamthelligkeit und den Daten der LSD-Lichtsensoren.

Als Erstes soll das Messergebnis des gewichteten Signals mit dem Messsignal des korrekt oberhalb des Fensters montieren LSD-Lichtsensors verglichen werden.

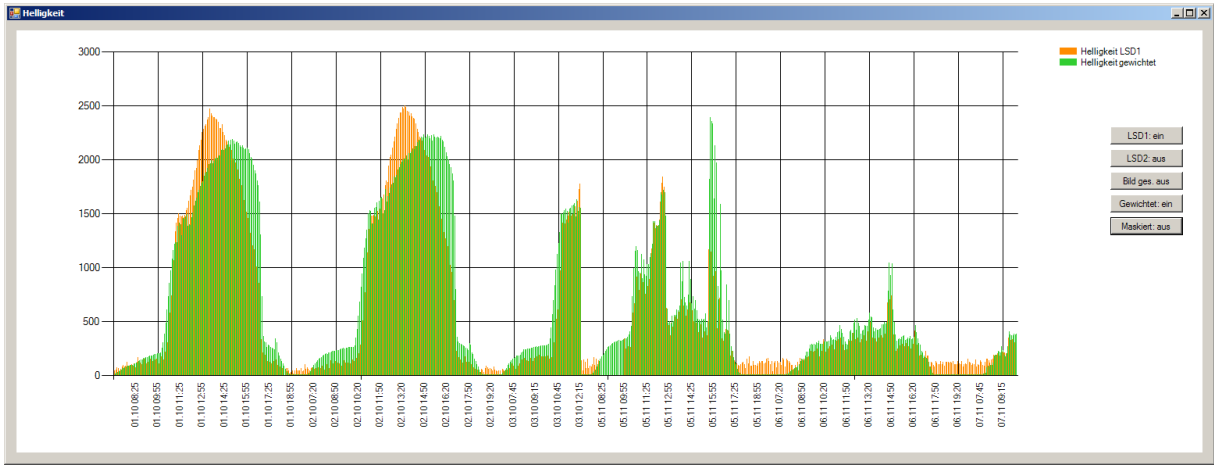

Abbildung 45: Gewichtetes Kamerasignal vs. LSD-Lichtensor am Fenster.
Signal aufgezeichnet über mehrere Tage jeweils ohne Nachtstunden.
orange Kurve: LSD-Lichtsensor; grüne Kurve: Kamerasignal

Das errechnete und das gemessene Signal stimmt von der Größenordnung her sehr gut überein. Die ersten beiden Tage, beide mit Sonnenschein und kaum Wolken, zeigen Abweichungen in den hellen Mittagsstunden. Die nächste Abbildung zeigt die Unterschiede deutlicher:

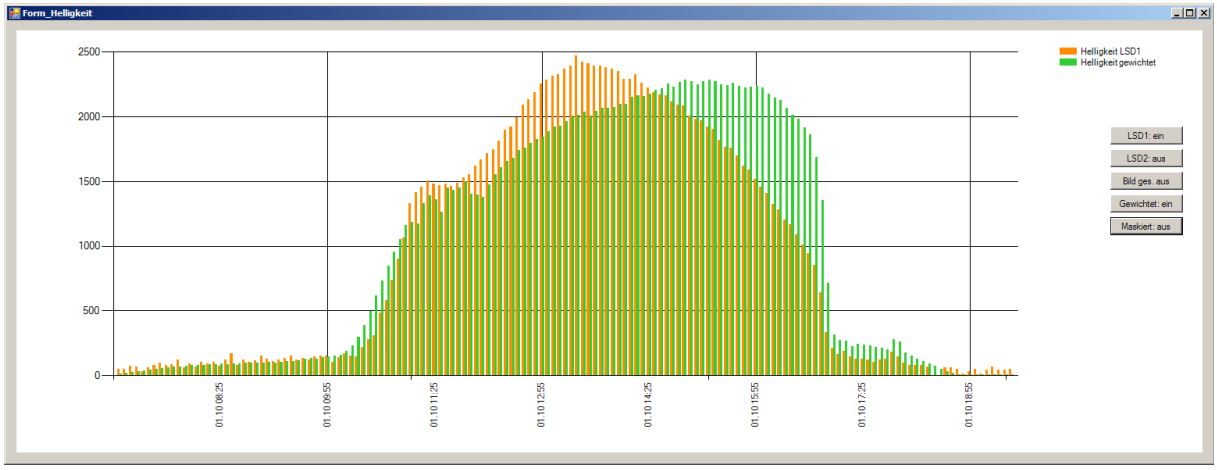

Abbildung 46: Wie vorherige Abbildung, erster Tag vergrößert.

Es kann vermutet werden, dass, bedingt durch die Montagesituation, die Sonneneinstrahlung den Sensor am Fenster früher erreicht als die Kamera in der Raummitte. Das könnte die Helligkeits-maxima zu den unterschiedlichen Zeitpunkten erklären.

Stützen könnte diese These auch die Beobachtung der Signale der beiden LSD-Lichtsensoren. Auch hier zeigt sich, der Sensor in der Raummitte erreicht das Helligkeitsmaximum später als der Sensor am Fenster.

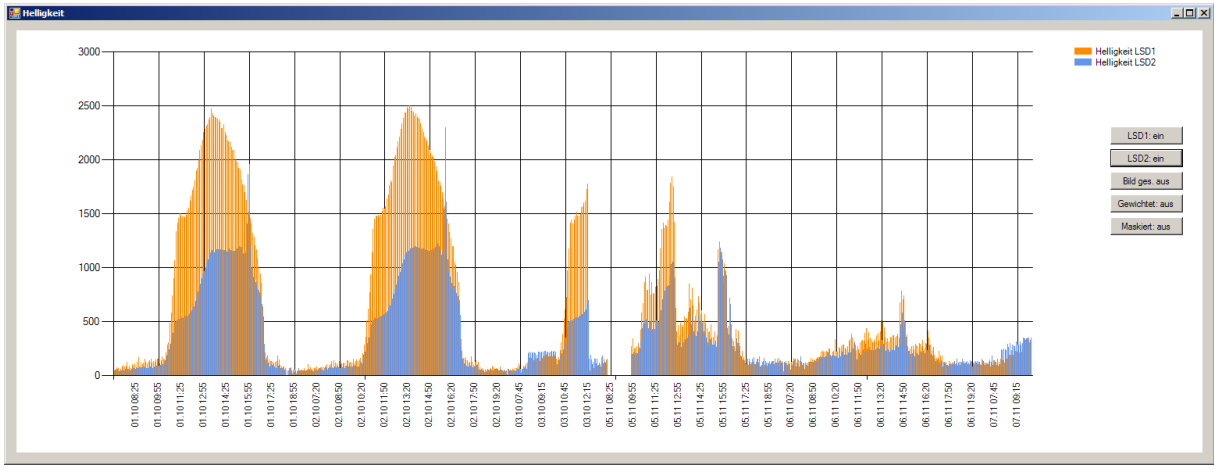

Abbildung 47: Sensor am Fenster (orange Kurve) vs. Sensor an Kameraposition (hellblaue Kurve)

Hingegen unterscheiden sich die Zeitpunkte der Helligkeitsmaxima der Gesamthelligkeit des Bildes und der gewichteten Helligkeit kaum. Lediglich die Skalierung und geringfügig auch die Kurvenform weichen voneinander ab.

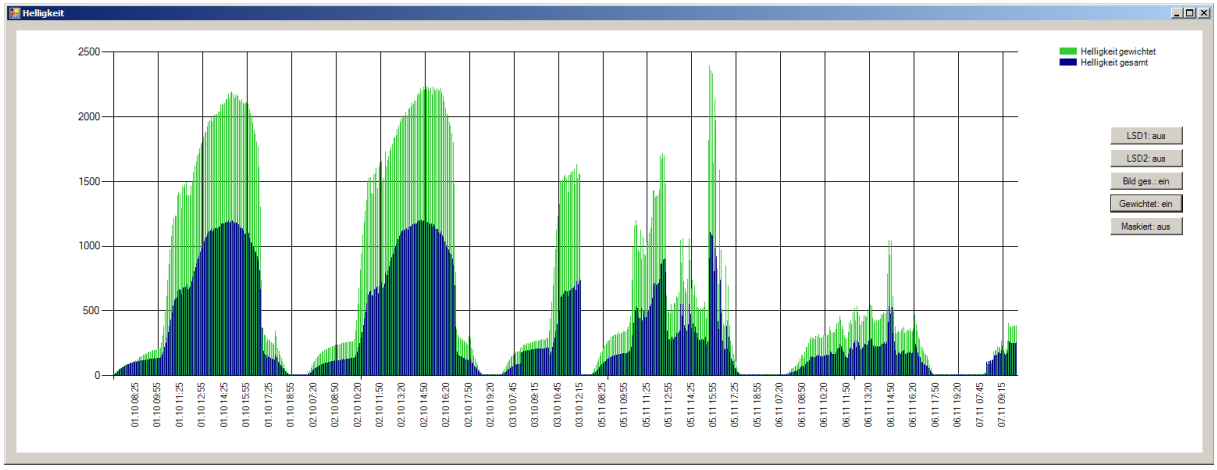

Abbildung 48: Gesamthelligkeit (blaue Kurve) vs. gewichtete Helligkeit (grüne Kurve)

Betrachtet man die Signale an Tagen mit wenig Sonnenschein (letzter Tag in den obigen Abbildungen), so stellt kann festgestellt werden, dass die Maximalwertverschiebung hier nicht auftritt. Das gemessene und das errechnete Signal stimmen sehr gut überein.

Es stellt sich die Frage, welches Signal, das des LSD-Lichtsensors beim Fenster oder das gewichtete Signal der Kamera, die Raumhelligkeit besser wiedergibt?

Ein Argument für die Kameraposition ist die Helligkeitswahrnehmung aus der Raummitte. Von hier kann das in den Raum einfallende Licht an einer breiteren Fensterfront gemessen werden.

Die Raummitte ist oft auch der Bereich, in dem sich Personen bevorzugt aufhalten. Diese nehmen dann eine ähnliche Lichtsituation wahr.

Auf der anderen Seite steht der LSD-Lichtsensor, welcher bereits über Jahre erfolgreich in vielen Gebäuden eingesetzt wird. Das Sensorsignal ist in jedem Fall ausreichend genau für eine gute, tageslichtabhängige Beleuchtungssteuerung.

Vielleicht ist diese Fragestellung auch gar nicht so relevant, da die Signale in den Bereichen geringerer Beleuchtungsstärken, gut übereinstimmen. Das sind auch vorwiegend jene Bereiche, in denen Kunstlicht zugeschaltet werden muss. Bei hohen Messwerten sorgt das Tageslicht für die Raumbeleuchtung.

Eine abschließende Klärung ist an dieser Stelle nicht möglich. Dazu wären weitere Messungen in unterschiedlichen Montagesituationen über längere Zeiträume notwendig. Diese Arbeit begnügt sich mit den erreichten Ergebnissen und konzentriert sich in weiterer Folge auf die technische Umsetzung eines Sensor-Prototypen.

5 Prototyp

Die Entwicklung eines Sensorprototypen ist neben dem Algorithmus zur Helligkeitsmessung und Gewichtung das zweite zentrale Thema dieser Arbeit. Mittels Prototyp soll geklärt werden, ob die theoretischen Überlegungen auch praxisgerecht umgesetzt werden können. Dazu gehört die Wahl der passenden Komponenten, der elektronische Aufbau, Systemprogrammierung, Fertigung sowie die Abschätzung der Kosten für ein allfälliges Serienprodukt. Idealerweise fügt sich so ein Prototyp elektrisch wie mechanisch in die bestehende Infrastruktur nahtlos ein.

Ein weiteres Ziel ist die Gewinnung von Messdaten zur kontinuierlichen Verbesserung der Sensor-Software bzw. des Helligkeits-Messalgorithmus.

5.1 Randbedingungen

Auch die Prototypen-Entwicklung wird von Randbedingungen begleitet, welche die Lösungsmöglichkeiten einschränken bzw. in eine gewisse Richtung lenken. Zu nennen sind u.a.: Vorgaben seitens der Fa. Zumtobel, die Verfügbarkeit einzelner Bauteile, Beschaffungs- und Fertigungsmöglichkeiten. Die folgenden Abschnitte beschreiben die Situation im Detail.

5.1.1 Wahl des Kameramoduls

Das zentrale Bauelement des Lichtsensors ist ein passendes Kameramodul. Es dient dazu die Lichtmenge zum messen und muss entsprechend sorgfältig ausgewählt werden. Die wichtigsten Anforderungen können wie folgt formuliert werden:

- Möglichkeit zur manuellen Belichtungssteuerung (kein Automatikfunktionen)
- einfache Schnittstelle, geeignet zur Anbindung an einen Mikrocontroller
- mehrere Ausgabeformate, am Besten ein Helligkeitswert
- geringer Energieverbrauch
- passender mechanischer Aufbau (Modul mit Flexkabel ist einer Version mit Sockel vorzuziehen)
- ausreichend große Auflösung (mindestens 40x30 Pixel - somit kein wirkliches Kriterium)
- langfristige Verfügbarkeit (für ein evtl. folgendes Serienprodukt)
- gute Dokumentation (!)

Gerade der letzte Punkt stellt die größte Hürde bei der Beschaffung eines Kameramoduls dar. Aus nicht näher bekanntem Grund stellen Kameramodulhersteller bzw. eigentlich die Hersteller der Bildsensoren, die Datenblätter zu ihren Produkten nicht öffentlich zur Verfügung, wie dies bei anderen elektronischen Bauteilen üblich ist. Meist ist nur ein kurzer „Product-Brief" mit den wichtigsten technischen Daten erhältlich. Auch eine direkte Anfrage bei den Herstellern verläuft meist negativ.

Da ein Kameramodul ein sehr komplexes Stück Technik ist, werden zur erfolgreichen Ansteuerung detaillierte Informationen zu Aufbau, interner Signalverarbeitung, Registerbelegung und der Schnittstelle nach außen benötigt. Ein nicht unerheblicher Teil des Zeitbudgets dieser Arbeit ist für die Suche und Beschaffung des Kameramoduls verwendet worden.

Trotz aller Schwierigkeiten konnte mit Hilfe der Fa. Zumtobel ein elektronisch und mechanisch passendes Modul gefunden werden. Type des Bildsensors: OmniVision OV2655 [35]. Scheinbar hat nicht einmal der chinesische Hersteller des Kameramoduls eine gültige Lizenz für die von ihm zur Verfügung gestellten Datenblätter. Zumindest deutet das, nicht auf den Herstellernamen lautende Wasserzeichen im Datenblatt, darauf hin.
Abbildung 19 in in Kapitel 3.6 zeigt das Modul mit Flexkabel-Anschluss.

5.1.2 Wahl des Mikrocontrollers

Im Gegensatz zum Kameramodul ist die Wahl des Mikrocontrollers einfach. Die Fa. Zumtobel möchte bei aktuellen und zukünftigen Entwicklungen einen Controller aus der EFM32-Familie der norwegischen Firma Energy Micro verwenden. Der Grund dafür ist, dass bei Zumtobel-Produkten die Energieeffizienz oft im Vordergrund steht. Energy Micro produziert - zumindest nach diversen Werbeaussagen - die energieeffizientesten Mikrocontroller auf Basis des ARM Cortex M3. [51]

Für die ersten Gehversuche mit einem EFM32 bietet die Fa. Olimex Entwicklungs-Boards mit dem Typ EFM32G880F128 an. [53] Das ist die größte Variante aus der so genannten „Gecko"-Serie des Mikrocontrollers. Die wichtigsten technischen Daten lauten: Prozessorkern ARM-Cortex M3; bis zu 32 MHz Takt; 16 kB SRAM; 128 kB onchip Flash-Speicher; knapp 90 Anschluss-Pins mit diversen Funktionen wie GPIO, UART, I^2C, SPI, ADC, DAC, Timer, usw.. Für eine genaue Aufstellung sei auf das Datenblatt [15] bzw. das Reference-Manual [52] verwiesen.

Die Verfügbarkeit der genannten Entwickler-Boards und die Vorgaben seitens der Fa. Zumtobel führten zum Entschluss genau diesen Mikrocontroller für den Sensor-Prototypen zu verwenden.

5.1.3 Entwicklungsumgebungen

Für die Programmierung von ARM-basierten Mikrocontrollern sind einige gute Entwicklungsumgebungen verfügbar. Eine davon ist „Keil uVision 4“. Diese Entwicklungsumgebung bietet alle Features, die zur professionellen Mikrocontroller-Entwicklung notwendig sind. Dazu gehört neben einem guten Editor, einer übersichtlichen Codeverwaltung und einem Firmware-Programmer auch ein guter Debugger. Dieser Debugger ermöglicht in Kombination mit einem passenden Programmiergerät das Sourcecode-Debugging direkt am Chip. Bis zu einer Codegröße von 32 kB kann die Entwicklungsumgebung kostenlos genützt werden. Diese Größe ist für dieses Projekt ausreichend.

Die Sensor-Firmware ist in der Sprache C auf Basis des Cortex M3 CMSIS, Cortex Microcontroller Software Interface Standard [61] und unter Zuhilfenahme der Energy Micro Library erstellt. Eine Firmwarebeschreibung mit Quellendokumentation ist im Anhang, Abschnitt 6.1.2 zu finden.

Auf PC-Seite ist zur Entgegennahme und Aufzeichnung der Sensordaten eine Software notwendig. Dieser Programmteil ist im Lichtsensor-Testprogramm integriert und wie die gesamte Software in der Sprache C# programmiert (siehe Anhang, Abschnitt 6.2).
Als Entwicklungsumgebung dient Microsoft Visual Studio Express 2010.

5.1.4 Datenübertragung

Der neue Lichtsensor soll, zusätzlich zur Messwertübertragung über die 4-20 mA Stromschnittstelle, in der Lage sein, die Sensordaten digital zu übermitteln. Dadurch können neben dem errechneten Messwert auch komplexe Daten wie z.B. die Bilder der einzelnen Belichtungsstufen, die Gewichtungsmatrix oder beliebige Statusdaten übermittelt werden. Diese Daten können am PC gespeichert und ggf. zur Verbesserung des Algorithmus, verwendet werden.

Aus Gründen der einfachen Handhabung, sowohl aus elektronischer als auch programmiertechnischer Sicht, wird die Lösung über eine RS-232 Schnittstelle gewählt. Genauer gesagt befindet sich am Sensor-Protoypen eine UART-Schnittstelle, deren Signal erst auf einer Adapter-Platine in ein RS-232 kompatibles Signal umgewandelt wird. Der Grund dafür liegt im begrenzten Platzangebot auf der Sensorplatine. Eine genormte RS-232 D-Sub-Buchse ist vergleichsweise groß, sodass alle nicht für den Sensorbetrieb notwendige Peripherie auf eine separate Platine ausgelagert wird. Außerdem hält diese Lösung die Option offen, durch Austausch der Adapterplatine den Sensor an andere Schnittstellen, z.B. USB oder Bluetooth, anzubinden. Der Lichtsensor erhält zu diesem Zweck eine Sensor-IO Schnittstelle, welche neben dem UART des Mikrocontrollers auch

weitere GPIO-Pins herausführt. U.a. soll es auch möglich sein, die Daten von analogen LSD-Lichtsensoren aufzuzeichnen.

Moderne PCs oder Laptops verfügen meist nicht mehr über eine RS-232 Schnittstelle. Aus diesem Grund ist ein USB/RS-232 Adapter notwendig. Solche, für wenige Euro erhältliche, Adapter verhalten sich aus programmiertechnischer Sicht völlig transparent, wodurch der Vorteil der einfachen Ansteuerung erhalten bleibt.

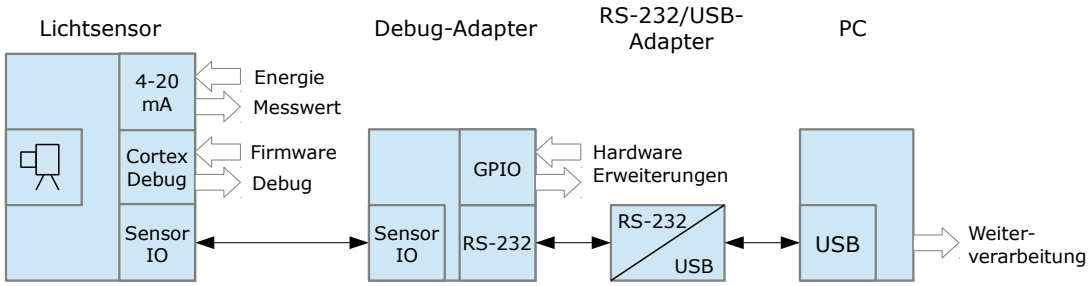

Abbildung 49: Kommunikation Sensorsystem

Um beliebige Daten vom Sensor zum PC und umgekehrt übertragen zu können, ist es sinnvoll, ein Übertragungsprotokoll einzusetzen. Dadurch ist auch jederzeit eine Erweiterung möglich.

Über das nachfolgend gezeigte Protokoll werden die Daten Byte für Byte über die UART-Schnittstelle des Mikrocontrollers zur RS-232 Schnittstelle des PCs übertragen. Auf beiden Seiten übernimmt eine Software-Routine den Datenstrom, überprüft ihn auf Korrektheit und gibt die Nutzdaten (Payload) an die zuständige Funktion weiter. Die Kommunikation wird in Form von Datenblöcken, man könne auch Nachrichten dazu sagen, abgewickelt.

Protokollaufbau (1 Block)

[1 Byte: SIG]	[1 Byte: TYP]	[2 Byte: LEN]	[LEN Bytes DATA]	[1 Byte XCK]

[SIG] Signatur für Blockanfang und Datenrichtung
0F: Signatur Sensor -> PC
F0: Signatur PC -> Sensor

[TYP] Art des Datenblocks
0x00: Statusdaten allgemein (Messwerte, ...)
0x01: Image1 LDR (40x30)
0x02: Image2 LDR (40x30)
0x03: Image3 LDR (40x30)
0x04: Image4 LDR (40x30) (optional)
0x05: Image Full Scale (z.B. 320x240) (optional)

0x06: HDR Image (40x30)
0x07: Gewichtungsmatrix (40x30)
0x08: Kamera Registersatz [2Byte Adresse] [1Byte Data] ...
...
0xEF:

0xF0: GetData: Daten senden Sensor->PC
0xF1: Baudrate ändern (Standard 115200)
0xF2:
...
0xFF:

[LEN] Länge des Datenblocks in Byte (Payload)

[DATA] Payload (LEN Bytes)

[XCK] XOR Prüfbyte (XOR Verknüpfung des gesamten Frames, Startwert 0x00)

Der Kommunikationsablauf beginnt mit dem Senden des Signatur-Bytes, welches den Beginn eines Frames signalisiert. Es folgen Typ und Länge des Datenblocks. Das LEN-Feld ist mit 2 Byte definiert, daher kann ein Datenblock max. 2^16 = 64 kB groß werden. Zur Fehlererkennung ist ein einfaches XOR Prüfbyte vorhanden. Bei Bedarf könnte auch eine bessere CRC-Checksumme verwendet werden.

5.1.5 Versorgung über Stromschnittstelle

Ziel ist es, den Sensor-Prototypen ausschließlich über die 4-20 mA Stromschnittstelle mit Energie zu versorgen. Wie in Abschnitt 2.1.2 beschrieben, muss der Lichtsensor bei 4 mA Schleifenstrom und 15 V Versorgung[14] mit etwa 0,06 W auskommen. Das ist nicht besonders viel und daher ist bei der Auswahl aller Komponenten der Sensorschaltung auf die Energieeffizienz zu achten.

Die Alternative, eine zusätzliche Stromversorgung bereitzustellen, soll nach Möglichkeit vermieden werden. Dies würde für einen reinen Prototypen zwar akzeptabel erscheinen, für ein Serienprodukt jedoch nicht. Der Wartungsaufwand den ein ständiger Batteriewechsel verursacht, wäre viel zu hoch. Nach Möglichkeit sollte der Prototyp einem Serienprodukt möglichst nahe kommen.

14 Die Fa. Zumtobel hat für das LSD-Lichtsensor-System 15 V Nennspannung für die Stromschleife definiert. Die Steuergeräte stellen diesen Pegel zur Verfügung. Bei langen Zuleitungen kann Aufgrund der Leitungswiderstände die tatsächlich nutzbare Spannung deutlich geringer ausfallen. Bei etwa 12 V Nutzspannung sollte der Sensor immer noch funktionieren.

5.1.6 Bauform

Auch die Bauform soll möglichst seriennah gestaltet sein. Das bedeutet, das vorhandene Gehäuse soll nach Möglichkeit weiterverwendet werden können. Dadurch entstehen Einschränkungen in der zur Verfügung stehenden PCB-Fläche und den Bauteilgrößen.

5.2 Ansteuerung der Kamera

Nachdem die Eckpunkte des Prototypen abgesteckt sind, ist die Ansteuerung der Kamera mit dem Mikrocontroller das erste Problem, welches es zu lösen gilt.

5.2.1 Kamerainterface

Das Kameramodul OV2655 verfügt über ein „Standard"-Interface, wie in Kapitel 3.6 beschrieben. Abbildung 18 zeigt den Aufbau.

Das Modul benötigt vom Hostsystem drei Versorgungsspannungen AVDD, DVDD und DOVDD. Die Spannungen AVDD und DOVDD sind mit 2,8 V festgelegt, müssen aber über ein RC-Filter voneinander getrennt sein. Die Spannung von DVDD beträgt 1,5 V. AVDD wird für den Analogteil, DVDD für den Digitalteil und DOVDD für das digitale IO-System verwendet. Genauer Details sind im Datenblatt [35] zu finden.

Zur Taktversorgung muss über den Pin XCLK ein Clocksignal im Bereich von 6-27 MHz zugeführt werden. Eine PLL am Chip, gesteuert über mehrere Registerwerte, erzeugt daraus den internen Takt sowie das Pixelclock-Signal PCLK. Daraus abgeleitet ergibt sich auch die Bildwiederholrate des Sensorchips.

Das Kameramodul verfügt über einen großen Registersatz für Parameter und zur Steuerung diverser Funktionen. Diese Register können von außen gelesen und geschrieben werden. Eine I^2C kompatible Schnittstelle übernimmt dabei die Kommunikation. Die Daten- und Clockleitung sind mit SIO_D und SIO_C bezeichnet. Das Hostsystem kann die Baudrate über einen weiten Bereich vorgeben. U.a. sind die Standardwerte 100 kHz oder 400 kHz möglich.

Die Bilddaten werden über einen 10-Bit parallelen Datenport DATA[9..0] zur Verfügung gestellt. Die meisten vom Kameramodul beherrschten Ausgabeformate verwenden 8-Bit Daten. In Kapitel 3.4.8 sind dazu einige Beispiele zu finden. Solche Formate werden über die höherwertigen Leitungen DATA[9..2] ausgegeben. DATA[1..0] bleiben in diesem Fall unbenutzt. Der Bildsensor verfügt über die die Möglichkeit 10-Bit RAW-Daten direkt auszugeben. Nur in diesem Fall werden alle 10 Datenleitungen verwendet. RAW-Daten sind die Daten direkt nach dem ADC des

Bildsensors. D.h. darin sind lediglich die Helligkeitswerte der Photodioden unter den Bayer-Pattern (vgl. Kapitel 3.4.4) enthalten. Eine Farbinterpolation hat noch nicht stattgefunden.

Leider kann dieses Format für das vorliegende Projekt nicht verwendet werden. Der Kameramodulhersteller hat nur die 8 höherwertigen Datenleitungen zum Stecker am Ende des Flexkabels herausgeführt. Die Datenleitungen DATA[9..2] werden daher in weiterer Folge als D[7..0] bzw. CAM_D[7..0] bezeichnet.

Der Sensorchip gibt die Bilddaten ununterbrochen in Form eines Datenstroms über die parallele Schnittstelle aus. Zur Synchronisation dienen die Signale VSYNC, HREF und PCLK. VSYNC signalisiert den Start eines neuen Frames, HREF eine neue Zeile. PCLK ist der Pixeltakt, welcher, einstellbar ob mit steigender oder fallender Flanke, die Gültigkeit der Daten an den Datenleitungen signalisiert.

5.2.2 Anbindung an den Mikrocontroller

Über die oben beschriebene Kameraschnittstelle soll die Anbindung an den Mikrocontroller erfolgen. Die Gesamtschaltung soll möglichst einfach ausfallen und daher ist es Ziel möglichst wenig Bausteine einzusetzen. Der EFM32G880F128 bietet viele Anschlussmöglichkeiten und sollte deshalb recht gut für die Ansteuerung geeignet sein.

Da der Mikrocontroller als Versorgungsspannung 1,8-3,6 V akzeptiert, kann dieser wie das Kameramodul mit 2,8 V betrieben werden. Das spart eine getrennte Spannungsversorgung und die evtl. notwendigen Pegelwandler zwischen den Chips. Die zusätzlichen 1,5 V für die DVDD-Leitung des Kameramoduls kann der Mikrocontroller auch direkt über einen der internen DACs zur Verfügung stellen.

Auch der Chiptakt kann vom Mikrocontroller zur Verfügung gestellt werden. Einer von drei Timer/Counter kann dazu verwendet werden. Der betreffende Baustein muss dazu in den Output-Compare-Modus geschaltet und ein GPIO-Pin als Taktausgang programmiert werden. Bis zur halben Taktfrequenz des Mikrocontrollers sind so als externe Taktquelle möglich. Bei 32 MHz Mikrocontroller-Takt sind das 16 MHz für das Kameramodul. Das passt genau in den definierten Frequenzbereich.

Nach dem Start des Kameramoduls durch Einschalten der Spannungsversorgungen und Zuführung eines Taktsignals, erwartet das Modul eine Initialisierungssequenz via SCCB-Interface. Diese I^2C-kompatible Schnittstelle kann ebenfalls direkt via Mikrocontroller angesteuert werden. [36] Dazu lässt sich das I^2C-Modul entsprechend programmieren. Selbst die obligatorischen Pull-Up

Widerstände können intern zugeschaltet werden. Eine externe Bestückung ist daher nicht notwendig.

Bleibt noch das Einlesen des Datenstroms über die parallele Schnittstelle. Dazu eignen sich grundsätzlich die GPIO-Pins des Mikrocontrollers. Getriggert über das PCLK-Signal kann das Datenbyte CAM_D[7..0] parallel eingelesen werden. VSYNC und HREF steuern dabei den Speicher-Pointer damit die Daten an die korrekte Adresse geschrieben werden. Nach einem Durchlauf ist so ein Bild im Speicher abgelegt und kann weiterverarbeitet werden. Soweit die Theorie.

Ein Blick ins Datenblatt des Kameramoduls [35] und in das des Mikrocontrollers [52] zeigt sehr schnell, dass diese Vorgangsweise nicht funktionieren kann. Bei voller Auflösung von 1600x1200 Pixel, und einer Bildwiederholrate von 15 FPS (Frames per Second) erzeugt das Modul im RGB565 Format (siehe Kapitel 3.4.8) einen Datenstrom von 1600 x 1200 x 2 x 15 = 57,6 MB/s. Ein Bild benötigt demnach 3,84 MB Speicherplatz. Das ist viel zu viel für die 16 kB RAM des Mikrocontrollers.

Zum Glück kann das Kameramodul die Zielauflösung und die Datenrate über weite Bereiche einstellen. Die ausgegebene Bildauflösung kann praktisch beliebig klein skaliert werden. Auch die gewünschten 40x30 Pixel wären möglich. Die Datenrate lässt sich ebenfalls verringern. Das führt aber gleichzeitig zur Verringerung der Bildwiederholrate. Und über die Bildwiederholrate ist die Belichtungszeit[15] verknüpft. 1 FPS z.B. erzeugt bei Tageslicht ein stark überbelichtetes Bild da sich die Belichtungszeit in diesem Beispiel um den Faktor 15 verlängert. Über Registereinstellungen für Precharge-Zeitpunkt, Analog-Gain und Digital-Gain kann dieser Effekt verringert werden, jedoch gibt es hier Grenzen. Ein weiter Nachteil entsteht durch die Verringerung der Framerate: Für eine Aufnahme muss die Kamera viel länger aktiviert bleiben. Das erhöht den Energieverbrauch beträchtlich. Aus diesen Gründen bietet der Ansatz über die Verringerung der Bildwiederholfrequenz keine gute Lösungsmöglichkeit.

Bei einer Auflösung von 40x30 Pixel, 15 FPS und RGB565-Format entsteht lediglich ein Datenstrom von 40 x 30 x 2 x 15 = 36 kB/s und eine Bildgröße von 2,4 kB. Der Pixeltakt müsste also lediglich 36 kHz betragen. Das wäre kein Problem für den Mikrocontroller. Ein interruptgesteuerter Trigger auf das PCLK-Signal oder sogar eine reine Softwarelösung sind ausreichend um die Daten zu samplen. Leider spielt das Kameramodul bei dieser Überlegung nicht mit.

15 Der OV2655 verwendet einen so genannten Rolling Shutter. Das bedeutet, nach dem Auslesen einer Bildzeile wird ein Precharge der Ladekondensatoren durchgeführt, welche die dazugehörenden Photodioden wieder entladen. Ein Frame später wird dieselbe Zeile erneut gelesen. Je nachdem wann nach dem Lesen der Precharge durchgeführt wird, ergibt sich die Belichtungszeit. Der Maximalwert beträgt dabei 1/Framerate. Bei 15 FPS ist das 1/15s. Weiterführende Infos sind in Kapitel 3.4.2 und im Datenblatt [35] zu finden.

Das PCLK-Signal wird immer so schnell getaktet wie notwendig ist, um bei der eingestellten Framerate, die maximale Auflösung auszugeben. Bei 15 FPS und 1600x1200 Pixel sind das 57,6 MHz. Kleinere Auflösungen nehmen lediglich eine kürzere Zeitspanne im zur Verfügung stehenden Übertragungsfenster ein. Die Samplingfrequenz bleibt aber unverändert hoch.

57,6 MHz liegen weit über der Mikrocontroller-Taktfrequenz und das Samplen der Daten ist damit ausgeschlossen. Die Frequenz, mit der der EFM32 Daten samplen kann, liegt außerdem weit unterhalb seiner Taktfrequenz. Für das Lesen von Daten von einem GPIO-Port benötigt der Prozessor 5-6 Taktzyklen. Die maximale Samplingfrequenz beträgt damit nur etwas mehr als 5 MHz. Diese Werte sind durch Versuch ermittelt und vom Energy Micro Support bestätigt.

Vermutlich kommt dieser vergleichsweise schlechte Wert durch Optimierung des Prozessors auf Low-Power zustande. Andere Mikrocontroller, auch auf Basis des ARM-Cortex M3, sind in dieser Disziplin weit schneller.

Ebenfalls interessant ist die Tatsache, dass eine Datenübertragung mittels DMA-Transfer die Situation nicht verbessert. Lediglich der Prozessorkern könnte sich während dem Einlesevorgang um andere Dinge, z.B. die Verarbeitung der zuvor eingelesenen Bildzeile, kümmern.

Gewählte Datensample-Lösung

Die genannten Schwierigkeiten führen bald zum Gedanken einfach einen schnelleren Prozessor zu verwenden. Die ebenfalls auf dem Cortex-M3 Kern basierende Mikrocontrollerserie STM32F2xx von STMicroelectronics verfügt sogar bereits über ein fertiges Kamerainterface, das alle Anforderungen erfüllt, und der Prozessor kann auf weit über 100 MHz getaktet werden. [54] Eigentlich ideal für die Aufgabenstellung. Das Problem ist nur, dieser Mikrocontroller ist mehr als doppelt so teuer wie der EFM32 und passt nicht in das Energiekonzept. Außerdem wäre damit der Wunsch von Fa. Zumtobel, einen EFM32 zu verwenden, nicht erfüllt.

Die Kombination, nur eine kleine Bildauflösung und ein Helligkeitssignal von der Kamera zu benötigen, eröffnet eine neue Möglichkeit. Es muss nicht jedes Pixel des Bildes gelesen werden können. Eine Zielauflösung von 40x30 Pixel ist erreichend.

Im konkreten Beispiel heißt das, es könnte etwa jedes 11. Byte aus dem Datenstrom gesampled werden (Annahme 57,6 MHz Pixeltakt und 5,3 MHz Samplefrequenz). Die Voraussetzung dafür ist aber, dass jedes gelesene Byte die gesamte Helligkeitsinformation eines Pixels beinhaltet. Im RGB565- oder YUYV-Format ist das nicht der Fall. Hier tragen 2 bzw. 4 nacheinander folgende Bytes die Pixelinformationen (vgl. Kapitel 3.4.8).

Das Kameramodul kennt aber auch das Y8-Ausgabeformat. Jedes ausgegebene Byte entspricht dem Helligkeitswert eines Pixels. Damit ist diese Idee umsetzbar.

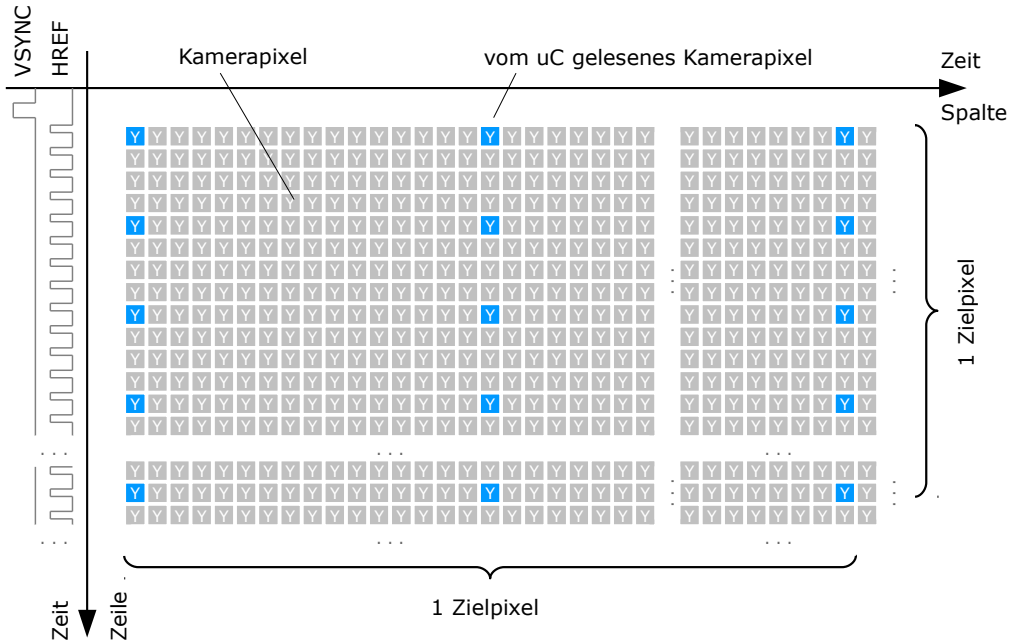

Abbildung 50: Sampling der Bilddaten mit Mikrocontroller (uC)

Die Umsetzung im Mikrocontroller geschieht folgendermaßen: Die Signale VSYNC und HREF triggern jeweils einen Interrupt. VSYNC kennzeichnet den Beginn eines neuen Frames und initialisiert den Bild- und Zeilen-Pointer so, dass diese auf den Beginn des Bildbuffers bzw. des Zeilenbuffers im Speicher zeigen. Mit dem HREF-Signal beginnt das Lesen einer Bildzeile. Dabei wird eine (optimierte) Programm-Schleife verwendet, welche die Werte vom Datenport liest und in den Zeilenbuffer schreibt. Mit den gewählten Einstellungen des Kameramoduls sind genau 100 Samples pro Bildzeile möglich.

Diese 100 Samples im Zeilenbuffer werden sofort nach dem Einlesen auf die horizontale Zielauflösung von 40 Pixel skaliert und in den Bildbuffer übertragen. Genauer gesagt werden die Pixelwerte im Bildbuffer aufaddiert und erst am Ende des Einlesevorgangs zur Mittelwertbildung dividiert. Das hat Performancegründe, da eine Addition viel schneller zu berechnen ist als eine Division. Am Ende des Einlesevorgangs ist jedoch genug Zeit die Divisionen durchzuführen.

Trotz dieser Optimierungen benötigt die Zeilenverarbeitung direkt nach dem Einlesen einige Rechenzeit. Dadurch kann nur jede 4. Bildzeile gelesen werden. Das Verschieben der Zeilenverarbeitung ans Ende des Einlesevorgangs würde zwar die Zeilenfrequenz erhöhen, allerdings müss-

ten dann wesentlich mehr Daten zwischengespeichert werden. Das würde den kleine Mikrocontroller-RAM zu sehr belasten.

So kann aus den 1600x1200 Kamerapixel ein Feld von 100x300 Pixel gesampled und auf 40x30 Pixel Zielauflösung heruntergerechnet werden. Das alles mit einem eigentlich für diese Aufgabe viel zu langsamen Mikrocontroller.

Die nachstehende Abbildung zeigt den Testaufbau für das Kamerainterface.

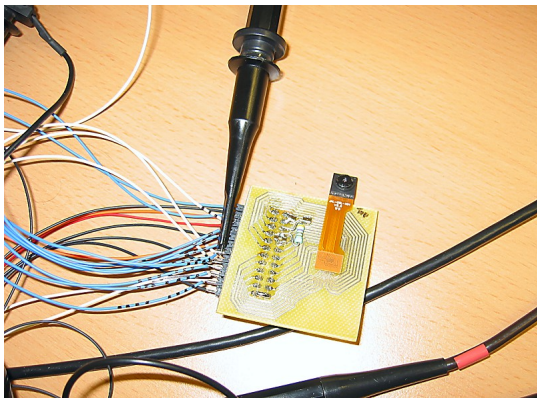

Abbildung 51: Kameramodul auf Breakout-Board mit Anschlüssen zum Mikrocontroller

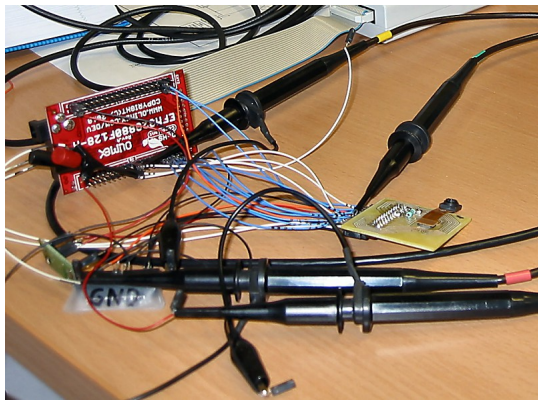

Abbildung 52: Mikrocontroller-Board und Kameramodul Testaufbau

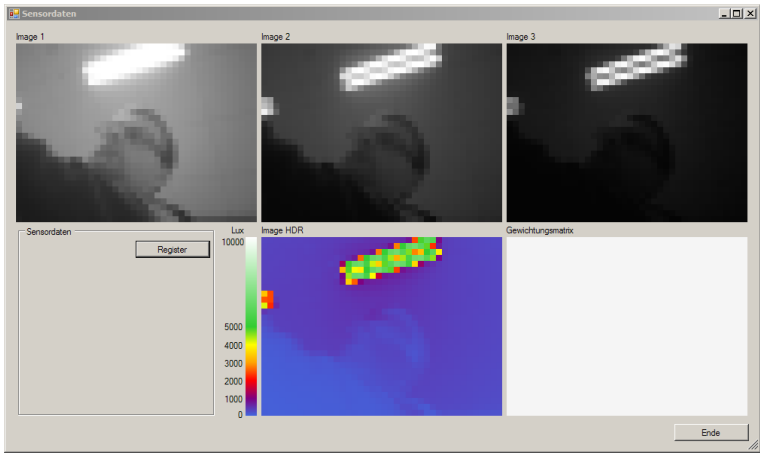

Abbildung 53: Kameradaten von Testplatine: Belichtungsreihe mit 3 Aufnahmen und resultierendes HDR-Bild

5.3 Energieversorgung

Wie bereits in Abschnitt 5.1.5 angesprochen soll die Versorgung des Sensors ausschließlich über die Stromschnittstelle erfolgen. Die daraus entnehmbare Energie von ca. 60 mW im Worst-Case

ist nicht besonders groß. Sie reicht aber aus, um z.B. eine Messbrücke und einige Operationsverstärker für die Signalübertragung zu betreiben. Der analoge Zumtobel LSD-Lichtsensor ist ähnlich aufgebaut, mit dem Unterschied, dass als Messfühler eine Photodiode statt einer Messbrücke eingesetzt wird. Für solche und ähnliche Sensoren ist die 4-20 mA Schnittstelle konzipiert.

5.3.1 Ermittlung der Stromaufnahme

Um abschätzen zu können, ob und wie die Versorgung des Sensors entworfen werden kann, ist zuerst die Stromaufnahme der verwendeten Bauteile zu bestimmen. Ein Blick in die Datenblätter gibt einen ersten Anhaltspunkt. [35][15]

Der Mikrocontroller benötigt demnach 180 µA/MHz bei VDD=3,0 V. Dieser Wert gilt für die Codeausführung aus dem Flash-Speicher im Energy-Mode EM0, also bei Vollbetrieb. Dazu kommt noch der Verbrauch der aktivierten Peripherie wie UART, Timer, I^2C, DAC, ADC, ...

Unter der Vernachlässigung der verwendeten 2,8 V Versorgung ergibt sich bei 32 MHz Taktfrequenz ein Stromverbrauch von ca. 5,8 mA. Dazu kommen noch geschätzte 1 mA für die aktivierte Peripherie. In den Energy-Modi EM1 und EM2 beträgt der Verbrauch 45 bzw. 0,9 µA/MHz. In diesen Energiesparmodi ist der Prozessorkern deaktiviert und je nach Modus sind unterschiedlich viele Peripheriemodule aktiv.

Das Kameramodul benötigt lt. Datenblatt noch wesentlich mehr Strom. Angegeben ist 250 mW im Active-Mode und 75 µA im Standby-Mode. Unter der Annahme, der Verbrauch kommt nur auf der 2,8 V Schiene zustande, ergibt sich ein Stromverbrauch von knapp 90 mA im Active-Mode.

Andere Verbraucher wie z.B. Spannungsregler, Querströme in Spannungsteilern, usw. werden pauschal mit 1 mA angenommen.

Stromprofil

Aus den ermittelten Werten ist erkennbar, der Stromverbrauch variiert je nach Betriebszustand sehr stark. Zur Veranschaulichung dient folgendes Stromprofil:

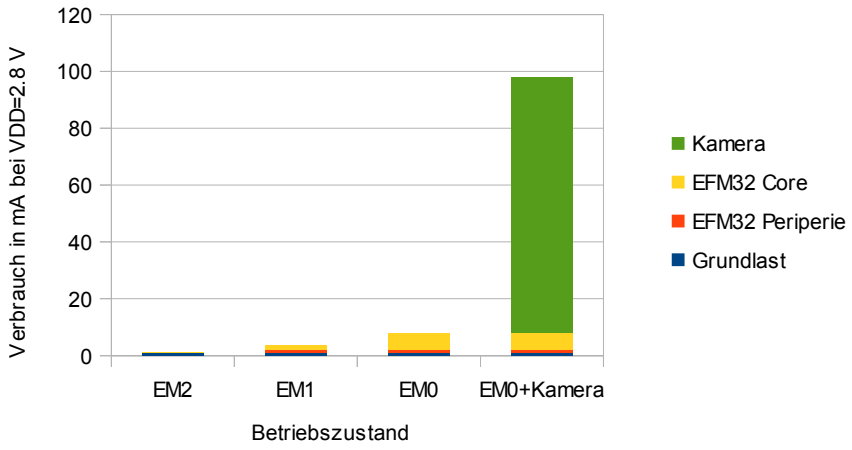

Abbildung 54: Stromprofil Lichtsensor

Messungen am Testaufbau der Schaltung bestätigen dieses Stromprofil großteils. Die Werte des EFM32 und die Grundlast stimmen ganz gut mit der Wirklichkeit überein, der Stromverbrauch der Kamera stellt den Worst-Case dar. Gemessen wurde ein Maximalwert von ca. 85 mA Gesamtverbrauch.

5.3.2 Energiespeicher

Die Strommessungen am Testaufbau zeigen ganz klar, ein Dauerbetrieb des Kameramoduls ist nur mittels Versorgung aus der Stromschnittstelle nicht möglich. Um den Sensor dennoch über dieses Schnittstelle betreiben zu können, muss der Kamerabetrieb auf kürzestmögliche Intervalle begrenzt werden und der fehlende Strom aus einem Speicherkondensator zur Verfügung gestellt werden. In den Pausen zwischen den Aufnahmen kann sich dieser Kondensator mit der überschüssigen Energie wieder aufladen. Um diesen Ladevorgang möglichst schnell abzuschießen, sollte in dieser Zeit auch der Verbrauch der restlichen Schaltung möglichst gering sein, d.h der Mikrocontroller soll so viel Zeit wie möglich in den Energiesparmodi EM1 und EM2 verbringen.

Der Lichtsensor benötigt zur Messung der Lichtsituation eine Belichtungsreihe mit drei Aufnahmen unterschiedlicher Einstellungen (siehe Kapitel 4.4). Für die Ansteuerung des Kameramoduls bedeutet das, das Modul wird aktiviert, erhält die erste Konfiguration via I^2C-Bus vom Mikrocontroller und gibt im Anschluss daran die Bilder aus. Das Kameramodul benötigt allerdings zwei

Bilder bis die Einstellungen vollständig übernommen sind. D.h. erst das 3. Bild kann für die Lichtmessung verwendet werden. Es sind also insgesamt 9 Aufnahmen für einen Messwert anzufertigen. Die Zeit, die dafür benötigt wird, beträgt, inklusive Kamerastart und Übermittlung der Konfigurationen, maximal 1,2 s.

Eine Abschätzung der Größe des Speicherkondensators kann über die entnommene Ladung erfolgen. Über die Beziehung $\Delta Q = I_L * \Delta t$ und $C = \Delta Q/\Delta U$ ergibt sich folgender Zusammenhang:

$$C_L = \frac{I_L \cdot \Delta t}{(U_{Lmax} - U_{Lmin})} \quad \text{bzw.} \quad T = \Delta t = \frac{C \cdot (U_{Lmax} - U_{Lmin})}{I_L}$$ falls die überbrückbare Zeit mit einer gegebenen Kapazität gesucht ist.

Aus den bisherigen Überlegungen können folgende Eckdaten abgelesen werden:

Mit aktivierter Kamera benötigt der Sensor gerundet 98 mA für 1,2 s. In den Pausen dazwischen wird mit 2 mA gerechnet, da die Kamera deaktiviert ist und der Prozessor die meiste Zeit im Sleep-Modus EM2 verbringen kann. Von der Stromschnittstelle können konstant 4 mA entnommen werden. Der nutzbare Spannungshub am Speicherkondensator sei mit U_{Lmax} = 12 V und U_{Lmin} = 3,5V angenommen. Die Differenz zu den 2,8 V sind für den Spannungsregler reserviert.

Diese Werte eingesetzt in die obige Beziehung ergeben ein C_L = (98-4 mA) * 1,2 s / (12-3,5 V) = 13300 µF. Das ist ein sehr großer Wert. Die Ladezeit mit den verbleibenden 2 mA ergibt sich mit T = 13300 µF * (12-3,5 V) / 2 mA = 56,5 s. Es ist also gerundet alle Minuten ein Messwert möglich. Das ist auch ohne große Ansprüche an die Reaktionsgeschwindigkeit für eine Lichtmessung zu wenig. Außerdem passen die 13300 µF, ausgeführt als Elektrolyt-Kondensator[16], mechanisch nicht in das vorhandene Gehäuse. Sie sind dafür einfach zu groß.

Bleibt nur die Schaltung zu optimieren. Da die 4 mA aus der Stromschnittstelle bei einer relativ hohen Spannung zur Verfügung stehen, ist es sinnvoll einen DC/DC-Konverter zur Spannungsregelung einzusetzen. Dadurch kann ein Großteil der Leistung von der hohen Spannung auf die niedrige Spannung umgesetzt werden. Bei einer angenommenen Effizienz von 70 % (Worst-Case) sind das - wieder angenommen eine maximale Arbeitsspannung von 12 V - immerhin noch 12 V * 4 mA * 70 % = 0,0336 W. Diese 33,6 mW ergeben bei 2,8 V einen Strom von 12 mA. Abzüglich der Grundlast von 2 mA bleibt immer noch der 5-fache Strom für die Schaltungsversor-

16 Eine Alternative wäre die Verwendung von Doppelschichtkondensatoren (Ultracaps, Supercaps, ...). Diese sind von der Bauform klein genug um in das Gehäuse zu passen. Allerdings haben solche Kondensatoren andere Nachteile: Neben dem relativ hohen Innenwiderstand (ESR) und den großen Leckströmen ist vor allem die geringe Spannungsfestigkeit dieser Kondensatoren ein Problem. Pro Zelle beträgt diese, abhängig vom verwendeten Elektrolyt nur etwa 3 V. Eine Serienschaltung ist zwar möglich, allerdings ist dann eine Load-Balancing-Schaltung vorzusehen um einzelne Zellen nicht zu überlasten. Außerdem wird der Vorteil der Bauform durch die Verwendung mehrere Doppelschichtkondensatoren wieder relativiert.

gung im Vergleich zur vorigen Annahme. Das alleine verringert die notwendige Kondensatorkapazität aber noch nicht ausreichend.

Die drei Aufnahmen die zur HDR-Bildberechnung benötigt werden, können auch getrennt voneinander angefertigt werden. Obwohl eine zeitnahe Aufnahme der Einzelbilder das Ergebnis des HDR-Bildes verbessert, kann in diesem Fall eine gewisse Toleranz akzeptiert werden. Die Lichtsituation ändert sich nicht so schnell.

Durch diesen Ansatz und weiteren Optimierungen, z.B durch Erhöhung der Taktrate des I^2C-Busses, kann die aktive Zeit der Kamera auf 0,33 s gedrückt werden. Eingesetzt in die Formel ergibt sich daraus C_L = (98-12 mA) * 0,33 s / (12-3,5 V) = 3340 µF.

Für das Laden des Kondensators stehen nach wie vor die 4 mA abzüglich der Grundlast zur Verfügung. Aufgrund des DC/DC-Konverters kann die Grundlast jetzt niedriger angesetzt werden. 0,7 mA an der 12 V-Seite sei hier angenommen. Daraus ergibt sich die Ladezeit wie folgt:
T = 3340 µF * (12-3,5 V) / 3,3 mA = 8,6 s. Damit ist ein Messwert ca. alle 26 s möglich[17].

Die so ermittelten Werte für den Worst-Case sind geeignet um darauf das weitere Schaltungskonzept aufzubauen.

5.3.3 Schaltungskonzept

Die Versorgung der Sensorschaltung über die 4-20 mA Schnittstelle muss so erfolgen, dass der ausgegebene Messwert nicht beeinflusst wird. Es muss daher einerseits der Schleifenstrom proportional zum Messwert geregelt werden und andererseits der Strom für die Schaltungsversorgung so begrenzt werden, dass die Schleifenstromregelung noch arbeiten kann. Benötigt die Sensorschaltung kurzfristig mehr Strom als aktuell zur Verfügung steht, muss dieser aus dem Speicherkondensator genommen werden.

Schleifenstromregelung

Die Schleifenstromregelung erfolgt über eine Operationsverstärkerschaltung mit Transistor, welcher so angesteuert wird, dass sich in der Stromschleife der gewünschte Strom einstellt. Wichtig ist, dass bei der Regelung des Schleifenstroms auch der Rückstrom aus der Schaltungsversorgung mit berücksichtigt wird, um ein korrektes Messsignal zu übermitteln. Abbildung 55, Abschnitt 5.4.1 zeigt das Prinzip.

17 Eine weitere Verbesserung der Messzykluszeit kann dadurch erreicht werden, dass in Situationen mit wenig Licht nur das erste (hellste) Bild für die HDR-Bildberechnung verwendet wird. Die beiden anderen Bilder enthalten in diesem Fall keine relevante Information.

Strombegrenzung

Um eine Beeinflussung der Stromschleife durch die Sensorschaltung zu vermeiden, muss der Strom knapp unterhalb des aktuell eingestellten Messwertes begrenzt werden. Eine kleine Differenz zum Messwert benötigt die Schleifenstromregelung zum Arbeiten.

Ein einfacher Widerstand ist für diesen Zweck nicht geeignet. Dieser könnte zwar so dimensioniert werden, dass bei Schleifen-Maximalspannung der Strom auf knapp 4 mA begrenzt wird. Bei höheren Schleifenstromwerten bricht aber die Versorgungsspannung ein und der Widerstand begrenzt den Strom dadurch auf einen kleineren Wert. Dies würde zur Situation führen, falls mehr Strom aus der Schleife entnommen werden dürfte, die Versorgung trotzdem weniger Strom zur Verfügung haben würde. Das kann keine gute Lösung sein.

Der nächst bessere Ansatz, eine konstante Strombegrenzung von z.B. 3,9 mA zu bauen, führt zwar zu verbesserten Ergebnissen, allerdings wird immer noch ein Teil des verfügbaren Stromes verschwendet.

Fazit: Eine gesteuerte Strombegrenzung, abhängig vom aktuell eingestellten Schleifenstrom soll Verwendung finden. Wie bei der Schleifenstromregelung soll die Steuerung über das VSET-Signal des Mikrocontrollers erfolgen.

Energiespeicher

Als Speicherkondensator wird ein Elko mit passender Kapazität eingesetzt. Aufgrund des weiter oben ermittelten Wertes von ca. 3300 µF sollte es auch möglich sein eine Bauform zu finden, welche in das Sensorgehäuse passt.

DC/DC-Konverter

Um einen möglichst großen Teil der zur Verfügung stehenden Energie aus der Stromschleife für die Schaltungsversorgung verwenden zu können, soll ein DC/DC-Konverter mit möglichst gutem Wirkungsgrad Verwendung finden. Der DC/DC-Konverter wandelt die Spannung am Speicherkondensator in die Nähe der Zielspannung von 2,8 V. Die verbleibende Restspannung übernimmt ein LDO-Spannungsregler, welcher eine wesentlich stabilere Ausgangsspannung liefert als der DC/DC-Konverter.

LDO-Spannungsregler

Die relativ großen Ripple in der Ausgangsspannung des DC/DC-Konverters sind nicht gut für die Versorgung der Analogbereiche des Mikrocontrollers und des Kameramoduls geeignet. Daher wird ein Low-Noise LDO-Spannungsregler zur Stabilisierung verwendet. Um das Kameramodul ganz von der Versorgung trennen zu könne, wird dafür ein separater LDO, gesteuert über den

Mikrocontroller, verwendet. Das erhöht zwar etwas den Bauteilaufwand, verringert aber den Stromverbrauch und vereinfacht die Ansteuerung.

Mikrocontroller

Der Mikrocontroller hat die Aufgabe, die Kamera anzusteuern, den Messwert zu berechnen und über den DAC zur Verfügung zu stellen. Außerdem bedient der Mikrocontroller das Sensor-IO Interface, wertet den Taster am Sensor aus und steuert die LEDs zur Statusanzeige. Eine weitere Aufgabe ist das Messen der Spannung am Speicherkondensator um beurteilen zu können, wann genug Energie vorhanden ist, um eine Bildaufnahme zu starten.

Kameramodul

Das Kameramodul kommuniziert direkt mit dem Mikrocontroller. Zusätzliche Bauteile, außer der separate, ebenfalls über den Mikrocontroller gesteuerte, LDO-Spannungsregler sind nicht notwendig.

5.4 Simulation der Schaltung

Zur Überprüfung des erarbeiteten Schaltungskonzepts wird eine SPICE-Simulation durchgeführt. Dabei kommt die bekannte Software LTspice von Linear Technology zum Einsatz. [55] Der Grund dafür liegt neben der einfachen Bedienung in der großen mitgelieferten Bauteilbibliothek. Um der realen Schaltung möglichst nahe zu kommen, finden in der Simulation dieselben Bauteile Verwendung die später auch im Lichtsensor-Prototypen eingesetzt werden.

5.4.1 Modellierung der Schaltung

Eine Simulation kann natürlich nicht alle Aspekte einer realen Schaltung umfassen. In diesem Fall ist z.B. das Verhalten des Mikrocontrollers und des Kameramoduls nicht simulierbar. Dazu müsste ein Modell der Bausteine inklusive des Firmware-Programms vorliegen. Das ist natürlich nicht der Fall. Aus diesem Grund wird das Verhalten des Prozessors in den für die Simulation wichtigen Punkten nachgebildet. Das sind z.B. die ausgegebenen Steuerspannungen oder das Lastprofil. Die nachfolgenden Abbildungen 55 zeigt den Aufbau.

Der auszugebende Messwert und damit der Schleifenstrom wird als Spannung Vset über den DAC des Mikrocontrollers zur Verfügung gestellt. Über den Widerstand R6 wird diese Spannung in einen Strom umgewandelt, welcher seinerseits über den Widerstand R5 abfließen muss. Das erzeugt an R5 einen Spannungsabfall. Der Knoten zwischen den Widerständen R5 und R6 ist mit dem nicht-invertierenden OPV-Eingang verbunden.

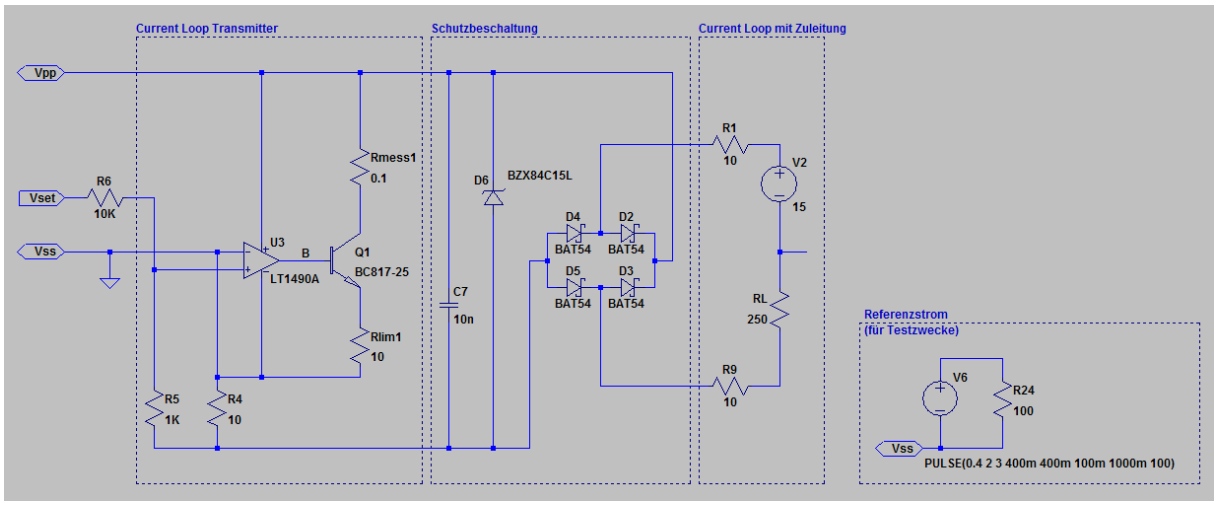

Abbildung 55: Schaltung für Simulation - Teil 1
Schleifenregelung, Schutzbeschaltung und Modell der Stromschleife

Über den Transistor Q1 und den niederohmigen Widerstand R4 fließt der Schleifenstrom. Der Emitter-Anschluss von Q1 wird über Rlim1 an den invertierenden OPV-Eingang zurückgeführt und bildet die virtuelle Schaltungsmasse. Der OPV versucht die Differenzspannung an seinen Eingängen gegen 0 auszuregeln und steuert den Transistor soweit auf, dass an R4 und R5 die gleiche Spannung abfällt. So entsteht ein Schleifenstrom proportional zu Vset.

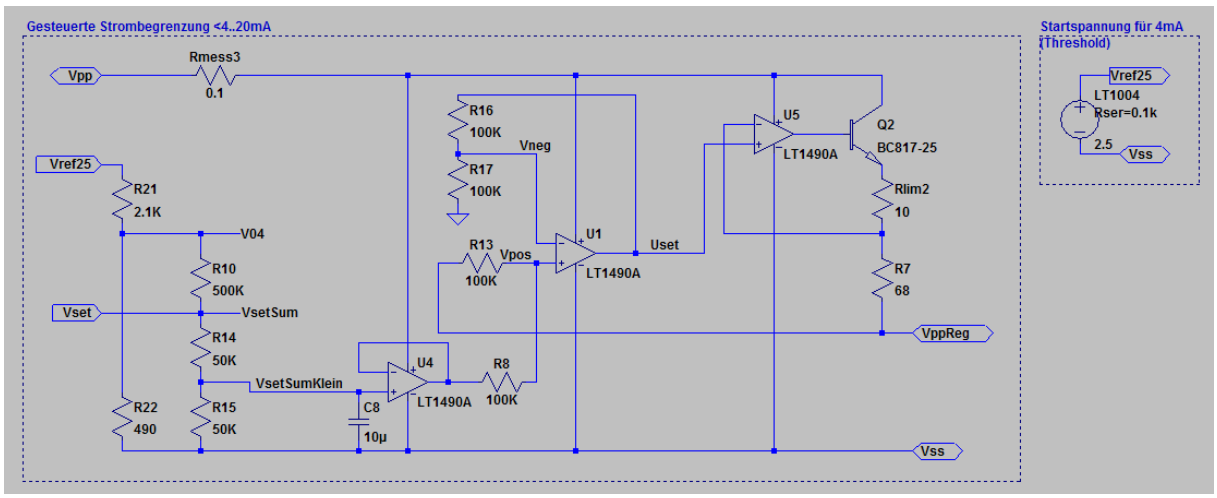

Abbildung 56: Schaltung für Simulation - Teil 2
Gesteuerte Strombegrenzung

Über den Operationsverstärker U1, welcher als nicht-invertierender Addierer geschaltet ist, wird die Spannung Vset (genauer VsetSumKlein) zur Spannung des Kondensators VppReg dazu addiert. Die resultierende Spannung dient zur Ansteuerung der eigentlichen Strombegrenzungsstufe. Sie ist nach dem gleichen Prinzip aufgebaut wie die Schleifenregelung. Der Grund, warum

nicht direkt Vset sondern VsetSumKlein für die die Ansteuerung verwendet wird, ist darin zu suchen, dass der Widerstand R7 nicht zu groß werden soll, damit der Spannungsabfall klein bleibt und somit mehr Arbeitsspannung am Speicherkondensator zur Verfügung steht.

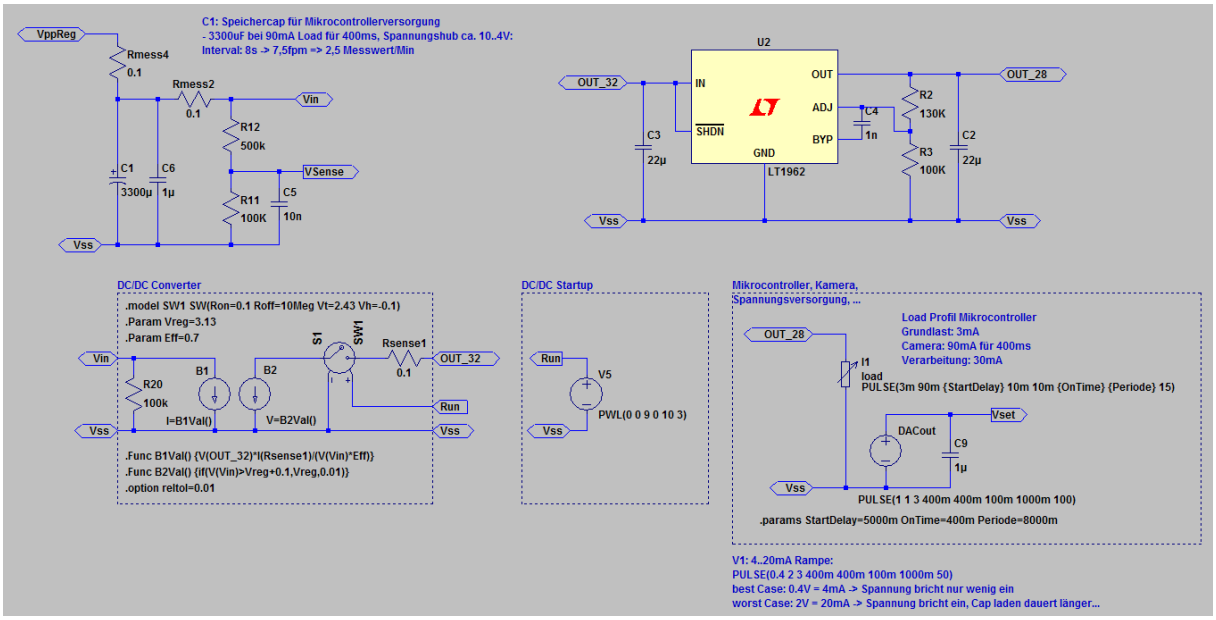

Abbildung 57: Schaltung für Simulation - Teil 3
Energiespeicher, DC/DC-Konverter, Spannungsregler sowie gesteuerte Last mit Lastprofil

Im 3. Teil der Simulationsschaltung findet sich der Energiespeicher samt Messschaltung zur Spannungsmessung VSense, der DC/DC-Konverter und ein LDO-Spannungsregler. Der Stromverbrauch der Schaltung (Mikrocontroller, Kameramodul, usw.) wird als gesteuerte Last modelliert. Das Lastprofil entspricht den weiter oben ermittelten Werten.

Eine Besonderheit trifft auf den DC/DC-Konverter zu. Dieser ist nicht als LTspice-Modell eingebunden, obwohl dieses verfügbar wäre, sondern als Verhaltensmodell (Behavioral-Modell).

5.4.2 DC/DC-Konverter Behavioral-Modell

Die Schaltung muss im vorliegenden Fall über eine relativ lange Zeitspanne simuliert werden um eine Aussage zur Funktion treffen zu können. Bis zu mehreren Minuten wäre wünschenswert. Dabei soll die Simulationszeit in einem vernünftigen Rahmen bleiben. Während bei den Modellen der Operationsverstärker, Spannungsregler und passiven Komponenten keine Simulationszeit-Probleme auftreten, kann eine Transientenanalyse des DC/DC-Konverters nur bis zu einigen Millisekunden (sinnvoll) berechnet werden. Aus diesem Grund muss der Baustein durch ein angenähertes Behavioral-Modell ersetzt werden.

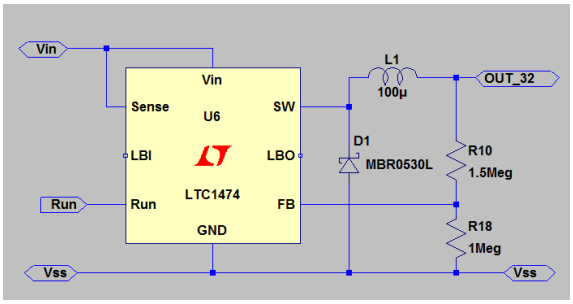

Abbildung 58:
LTspice Modell des DC/DC-Konverters

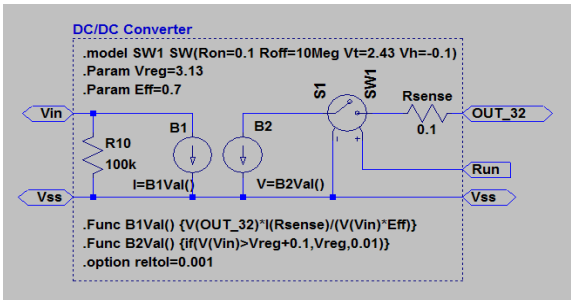

Abbildung 59:
Behavioral-Modell des DC/DC-Konverters

Mit diesem vereinfachten Modell kann die Gesamtschaltung auf dem vorhandenen PC mit etwa 4-facher Echtzeitgeschwindigkeit simuliert werden, verglichen mit mehreren Stunden ohne diese Optimierung.

5.4.3 Simulationsergebnisse

Die Simulationsergebnisse zeigen ein erfreuliches Bild. Die weiter oben gemachten Abschätzungen können bestätigt werden. Eine Speicherkapazität von 3300 µF ist ausreichend für das ermittelte Lastprofil.

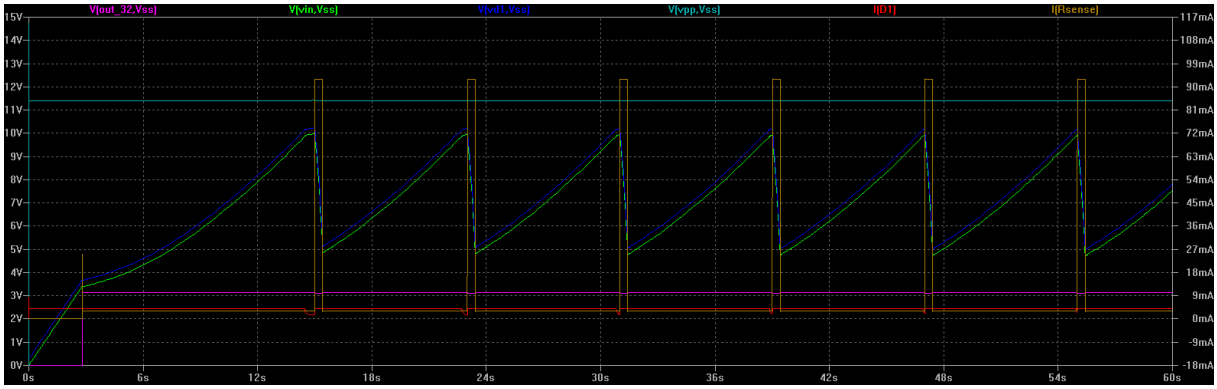

Abbildung 60: Simulationsergebnis Worst-Case Simulation

Obige Abbildung zeigt den Spannungsverlauf am Speicherkondensator (blaue und grüne Kurve) über die Zeit mit dem gewählten Lastprofil (braune Kurve). Die Schleifenspannung (hellblaue Kurve) beträgt in diesem Fall etwa 11,5 V. Als Schleifenstrom wurde konstant 4 mA gewählt (hier nicht eingezeichnet).

Dieses Szenario bildet die Worst-Case Situation. Die maximale Kapazität des Speicherkondensators wird benötigt. In dieser Simulation kann alle 8 Sekunden ein Bild aufgenommen werden. Damit ergibt sich eine Messwertfolge von 24 s. Das ist sogar etwas besser als die in Abschnitt 5.3.2 berechneten 26 s.

5.5 Praktische Realisierung

Ermutigt durch die Vorarbeiten zur Kameraansteuerung und die vielversprechenden Simulationsergebnisse kann an den praktischen Aufbau der Schaltung herangegangen werden. Als Layout-Werkzeug dient, wie in solchen Fällen am Institut für Elektronik der TU-Graz üblich, die PCB Design Software Cadsoft Eagle. [56]

5.5.1 Gesamtschaltung Lichtsensor NG

Zur Unterscheidung vom bisherigen LSD-Lichtsensor erhält der neue Sensor den Namenszusatz NG. NG steht für New Generation. Die fertige Schaltung ist nachfolgend abgebildet und beschrieben.

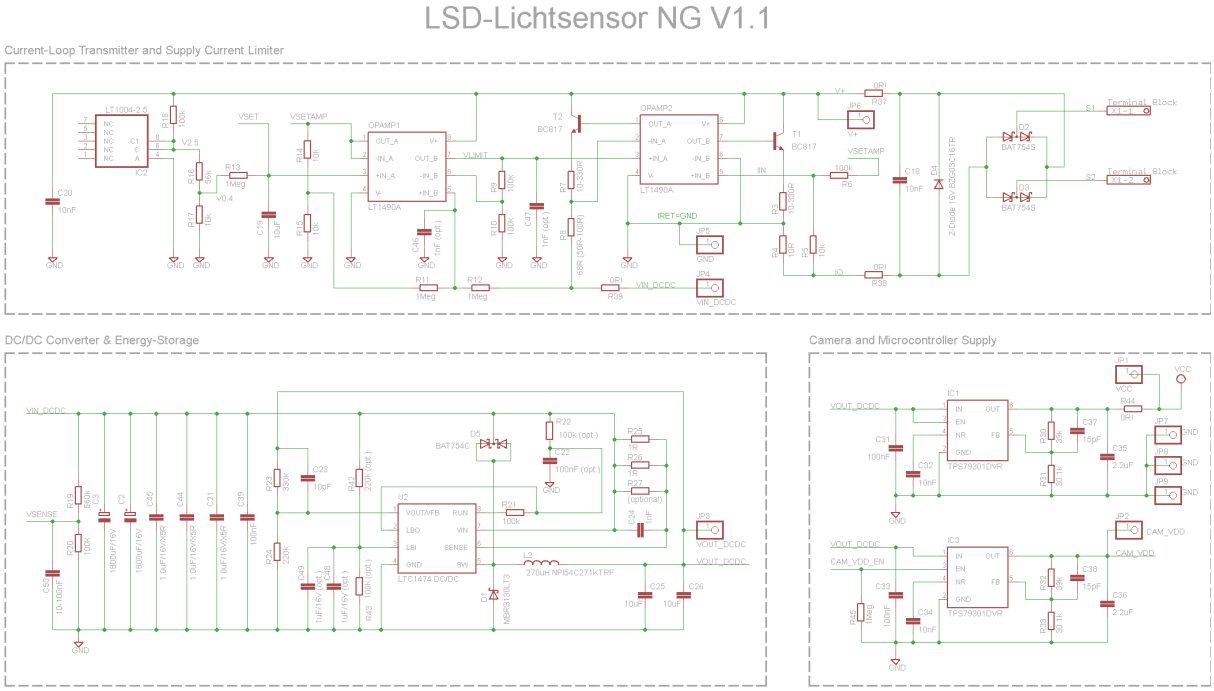

Abbildung 61: LSD-Lichtsensor NG V1.1 - Teil 1
Module: Current-Loop Transmitter and Supply Current Limiter,
DC/DC Converter & Energy Storage sowie Camera and Microcontroller Supply

Current-Loop Transmitter and Supply Current Limiter

Die Schaltung kann beginnend in diesem Modul von der rechten Seite aus gelesen werden. Die Stromschleife wird über die Diodenbrücke D2 und D3 verpolsicher angeschlossen. Die Z-Diode D4 dient als Schutz gegen Überspannung und C18 zur Stabilisierung. Wie beim original LSD-Lichtsensor liegt die Nennspannung der Schaltung bei 15 V.

Der Doppel-Operationsverstärker OPAMP2 implementiert über den Transistor T1 die Schleifenregelung und über T2 die Strombegrenzung für die Schaltung. Alle verwendeten Operationsverstärker sind vom Typ LT1490A [57]. Dabei handelt es sich um einen Low-Power Typ mit Single Supply und Rail-to-Rail Ausgängen. Neben diesen Features ist das vorhandene LTSpice-Modell der Hauptgrund für die Wahl. Geschwindigkeit und Bandbreite spielen in dieser Anwendung nur ein untergeordnete Rolle.

Die Schleifenregelung wird wie in Abschnitt 5.3.3 beschrieben, über die Steuerspannung VSETAMP, angesteuert. Über R6 wird daraus ein Strom erzeugt, welcher an R5 zu einem Spannungsabfall führt. Diese Spannung wird vom OPV mit dem Spannungsabfall an R4 verglichen und über T1 ausgeregelt. Der Massepunkt der Schaltung befindet sich am Konten IRET=GND. Dadurch wird der gesamte Rückstrom (IRET) der Schaltung über R4 abgeleitet und dadurch in der Schleifenstromregelung mit berücksichtigt.

Auf der Seite der Strombegrenzung (T2) verhält es sich ähnlich. Nur wird hier der Spannungsabfall an R8 mit der Steuerspannung VLIMIT verglichen. Der Wert von R8 ist mit 68 Ω so gewählt, dass die Strombegrenzung den aktuellen Schleifenstrom nicht überschreitet und für die Schleifenstromregelung noch ein kleiner Spielraum bleibt.

Der zweite Doppel-Operationsverstärker OPAMP1 ist auf einer Seite als Spannungsfolger für VSET geschaltet und auf der anderen als nicht-invertierender Addierer. VSET wird vom DAC des Mikrocontrollers erzeugt und über die relativ große Kapazität C19 gestützt. Der Spannungsfolger verhindert ein Entladen des Kondensators und gibt das Signal als VSETAMP weiter. Im Energiesparmodus EM2 arbeitet der DAC nicht, weshalb die Spannung in dieser Zeit aufrechterhalten wird. Der Schleifenstrom und damit der Messwert soll ja während er Energiespar-Phasen des Mikrocontrollers unverändert ausgegeben werden.

Der Addierer addiert die Hälfte (R14/R15) der Spannung VSETAMP mit der Spannung VIN_DCDC. Die so erzeugte Steuerspannung ist das oben erwähnte VLIMIT für die Strombegrenzung. Als Beschaltung dient R9/R10 und R11/R12.

Ganz auf der linken Seite befindet sich das Startup-Netzwerk mit der LT1004-2.5 Spannungsreferenz. Dieser Teil sorgt für den Start der Schaltung. Nach dem Anlegen der Spannung steht

VSET noch nicht zur Verfügung, wodurch wiederum der Transistor T2 sperrt. Damit ist die Strombegrenzung auf ein Maximum eingestellt und der Mikrocontroller kann nicht starten.

Um diesen Teufelskreis zu durchbrechen, wird über den hochohmigen Widerstand R13 der Kondensator C19 geladen wodurch in weiterer Folge T2 immer mehr aufsteuert. Dadurch kann ein Strom zu den Speicherkondensatoren (C2 & C3) fließen und auch diese aufladen. Irgendwann starten die Spannungsregler und damit der Mikrocontroller. Zu diesem Zeitpunkt ist in den Speicherkondensatoren gerade genug Energie vorhanden um ein kurzes Programm auszuführen. Der Mikrocontroller stellt VSET auf den Maximalwert und geht sofort wieder in den Sleep-Modus. Damit ist ein Schleifenstrom von 20 mA eingestellt und die Strombegrenzung entsprechend offen. Die Speicherkondensatoren laden sich schnell auf. Der Mikrocontroller erwacht periodisch aus dem Sleep-Modus und misst die Kondensatorspannung über einen seiner ADCs. Ist genug Energie vorhanden, startet die Aufnahmesequenz. Der erste Messwert wird ermittelt und der Schleifenstrom entsprechend angepasst. Ab hier herrscht Normalbetrieb.

Zurück zum Startup-Netzwerk: Sobald VSET zur Verfügung steht, wird die Startspannung an R13 übersteuert. Diese ist ab dem Zeitpunkt nicht mehr relevant. Die Spannungsreferenz LT1004-2.5 ist eigentlich gar nicht notwendig, da der genaue Wert der Startspannung nicht von Bedeutung ist. Ein einfacher Spannungsteiler reicht vollkommen. Aus diesem Grund ist die Spannungsreferenz an den meisten Prototypen nicht bestückt.

DC/DC Converter & Energy Storage

Nach der Strombegrenzung findet die Spannung VIN_DCDC ihren Weg zu den Speicherkondensatoren C2 & C3 und in weiterer Folge über die Widerstände R25/R26/R27 zum DC/DC-Konverter vom Typ LTC1474. [58]

Bleiben wir vorerst bei den Speicherkondensatoren: Diese 2 x 1800 µF Elkos sind extra nach kleiner Bauform ausgesucht um in dem vorhanden Bauraum Platz zu finden. Weitere Kriterien sind die Spannungsfestigkeit von 16 V und eine vergleichsweise lange Lebensdauer von 10000 h bei 105° C.

Über den Spannungsteiler R19/R20 wird die Kondensatorspannung über das Signal VSENSE zum Mikrocontroller geführt. C50 dient als Stützkondensator. Die Kapazitäten C44/C45/C21 und C39 sind lt. Datenblatt des LTC1474 angebracht und fangen kurze hohe Stromspitzen ab.

Der LTC1474 DC/DC-Konverter benötigt für den Betrieb einige externe Beschaltung. Besonders wichtig ist die Auswahl der Catch-Diode D1. Diese soll schnell schalten und einen geringen Forward-Voltage-Drop aufweisen. Auch ein geringer Leckstrom ist wichtig. In diesem Fall wird

die im Datenblatt für maximale Effizienz empfohlene MBRS130, eine Schottky-Diode mit V_F = 0,3 V, I_L = 20 µA, verwendet. Ebenso wichtig ist die Wahl der Induktivität L2. Der optimale Wert hängt vor allem vom erwarteten Maximalstrom ab. Dieser liegt hier bei 100 mA woraus sich laut Datenblatt der gewählte Wert von 270 µH ableitet. In diesem Zusammenhang sind auch die weiter oben schon angesprochenen Widerstände R25/R26/R27 zu betrachten. Diese dienen als Strombegrenzung für den LTC1474.

Beim Betrieb des Sensors über den Debug-Adapter kann es vorkommen, dass die Ausgangsspannung VOUT_DCDC über der Eingangsspannung VIN_DCDC liegt. Zum Schutz des DC/DC-Konverters ist die Diode D5 angebracht. Sie dient als Latchup-Protection bei Reverse-Current.

Die Feedback-Schleife bildet der Spannungsteiler R23/R23 mit der Frequenzgangkorrektur C23. Damit lässt sich die Ausgangsspannung festlegen. Gewählt wurde ca. 3,1 V. Damit verbleiben 0,3 V Spielraum für die nachgeschalteten LDO-Spannungsregler.

Die Netzwerke, die zum LBI- und LBO-Pin des LTC1474 führen, dienen Testzwecken um den Startzeitpunkt des DC/DC-Konverters einzustellen. In der fertigen Version sind sie nicht bestückt.

Camera and Microcontroller Supply

Die beiden LDO-Spannungsregler vom Typ TPS7930 [59] stellen die stabilisierte Eingangsspannung von 2,8 V für den Mikrocontroller und das Kameramodul zur Verfügung. Beide Stufen sind identisch aufgebaut, mit dem einen Unterschied, dass der Regler für das Kameramodul vom Mikrocontroller über das Signal CAM_VDD_EN gesteuert wird. R45 dient als Pull-Down-Widerstand und ist wichtig während des Startups wenn der Mikrocontroller noch nicht zur Verfügung steht. Er verhindert das ungewollte Einschalten des Kameramoduls. Die restliche Beschaltung ist laut Datenblatt ausgeführt.

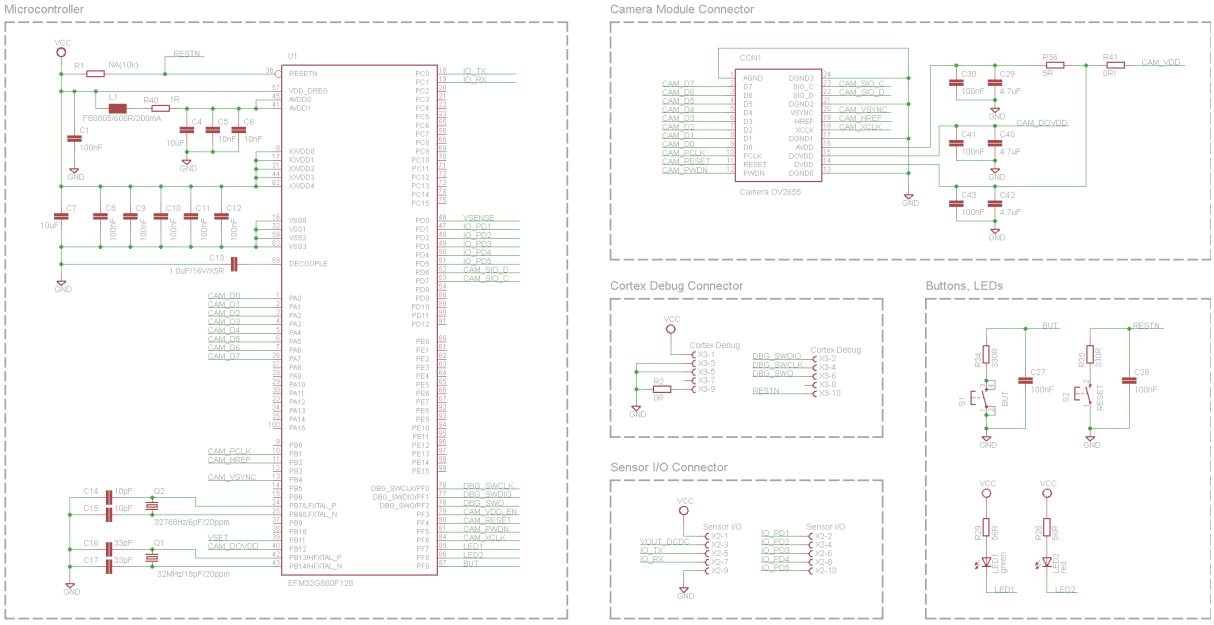

Abbildung 62: LSD-Lichtsensor NG V1.1 - Teil 2
Module: Microcontroller, Camera Module Connector,
Cortex Debug Connector, Sensor I/O Connector sowie Buttons & LEDs

Microcontroller

Der Mikrocontroller EFM32G880F128 ist großteils wie in den Datenblätten beschrieben beschalten. Links oben findet sich das Reset-Signal RESTN über einen Pull-Up Widerstand R1 nach VCC verbunden. RESTN wird vom Programmiergerät oder einem Taster auf der Platine gesteuert. Darunter befinden sich die Versorgungsanschlüsse des Prozessors mit einem Filter für die Analogspannung (L1/R40) und den Decoupling-Kondensatoren C4-C13.

Port A des Prozessors, Pins PA[7..0], dient als Datenport für die Kamera CAM_D[7..0]. Die Pins PB1, PB2 und PB4 sind mit den Kamerasignalen CAM_PCLK, CAM_HREF und CAM_VSYNC verbunden. CAM_PCLK wird in der derzeitigen Implementierung nicht verwendet und ist nur aus Gründen der Vollständigkeit mit dem Pin verbunden (siehe Kapitel 5.2.2).

Über Pin PB11 wird die Spannung für die Scheifenregelung VSET und über Pin PB12 die 1,5 V Versorgungsspannung für das Kameramodul bereitgestellt. Beide Analogsignale werden von einem DAC des Mikrocontrollers ausgegeben.

Die restlichen beschalteten Pins auf Port B sind für die Quarzoszillatoren reserviert. Einmal die 32 MHz (Q1) für den Prozessortakt und einmal 32768 Hz (Q2) für Echtzeituhr und andere Low-Frequency Peripherie des Mikrocontrollers.

Auf der rechten Seite des Mikrocontrollers finden sich, von oben nach unten, die Anschlüsse IO_TX und IO_RX für die Kommunikation via UART, VSENSE, ein ADC-Eingang zur Span-

nungsmessung am Speicherkondensator und die GPIO-Pins IO_PD1-IO_PD5, welche zum Sensor-IO Stecker geführt werden.

CAM_SIO_D und CAM_SIO_C dienen zur Kommunikation mit dem Kameramodul via I^2C-Schnittstelle.

Auf den Pins PF0-PF2 befindet sich die Programmier- und Debug-Schnittstelle des Mikrocontrollers. Diese Signale sind zum Cortex-Debug Stecker geführt.

PF3-PF6 mit den Signalen CAM_VDD_EN, CAM_RESET, CAM_PWDN und CAM_XCLK dienen zur Kameraansteuerung.

Bleiben noch die Ausgänge PF7, PF8 mit den Signalen LED1 und LED2 zur Ansteuerung zweier Signal-Leuchtdioden und der Eingang PF9 zur Abfrage des Tasters S1 in der Sensormitte. Dieser Taster dient u.a. zur Programmierung des Tag-Systempunktes für das Beleuchtungssteuergerät.

Camera Module Connector

Dieser Teil umfasst den Kameramodul-Stecker mit den oben genannten Signalen sowie die Decoupling-Kondensatoren für die drei Versorgungsspannungen des Kameramoduls lt. Datenblatt. Die Analogspannung AVDD ist via R36 von der Digital-Versorgung getrennt. Der Kamerastecker hat ein Rastermaß von lediglich 0,4 mm und befindet sich wie der Mikrocontroller auf der Unterseite des PCB-Boards.

Cortex Debug Connector

Beim Cortex-Debug Connector handelt es sich um einen genormten 10-Pin Pfostenstecker mit 1,27 mm (0,05 “) Rastermaß, zur Programmierung von Cortex M3 Prozessoren. Dieser Stecker hat eine wesentlich kleinere Bauform als der ebenfalls gängige JTAG-Programmierstecker.

Sensor-IO Connector

Mechanisch den gleichen Steckertyp wie der Cortex-Debug Connector verwendet auch der Stecker für Sensor-IO Interface. Über diesen Stecker kann der Sensor mit dem Hostsystem kommunizieren (IO_TX, IO_RX). Die Leitungen IO_PD[5..1] können entweder als beliebig Programmierbare GPIO-Pins für zukünftige Erweiterungen verwendet werden oder via Mikrocontroller-ADCs zum Messen von Signalen anderer LSD-Lichtsensoren dienen.

Buttons, LEDs

Der LSD-Lichtsensor NG verfügt über je zwei Taster und zwei LEDs. Ein Taster (S2) dient als mechanischer Reset-Knopf, Taster S1, platziert in der Platinenmitte zur Programmierung des

Tag-Systempunktes wie beim vorhanden LSD-Lichtsensor. R34/C27 und R35/C28 sind zum Entprellen der Kontakte vorgesehen.

Die beiden LEDs, eine Rot, eine Grün, signalisieren verschiedene Betriebszustände des Sensors. Der Mikrocontroller kann bis zu 20 mA Strom pro Pin liefern und so die LEDs direkt treiben.

5.5.2 Debug-Adapter

Der Sensor Debug-Adapter beherbergt alle Funktionen, die mit der reinen Sensorfunktionalität nicht direkt zusammenhängen und nur für die Programmierung oder Analysezwecke gebraucht werden. Außerdem sind auf dieser Platine alle Bauteile mit großem Formfaktor untergebracht.

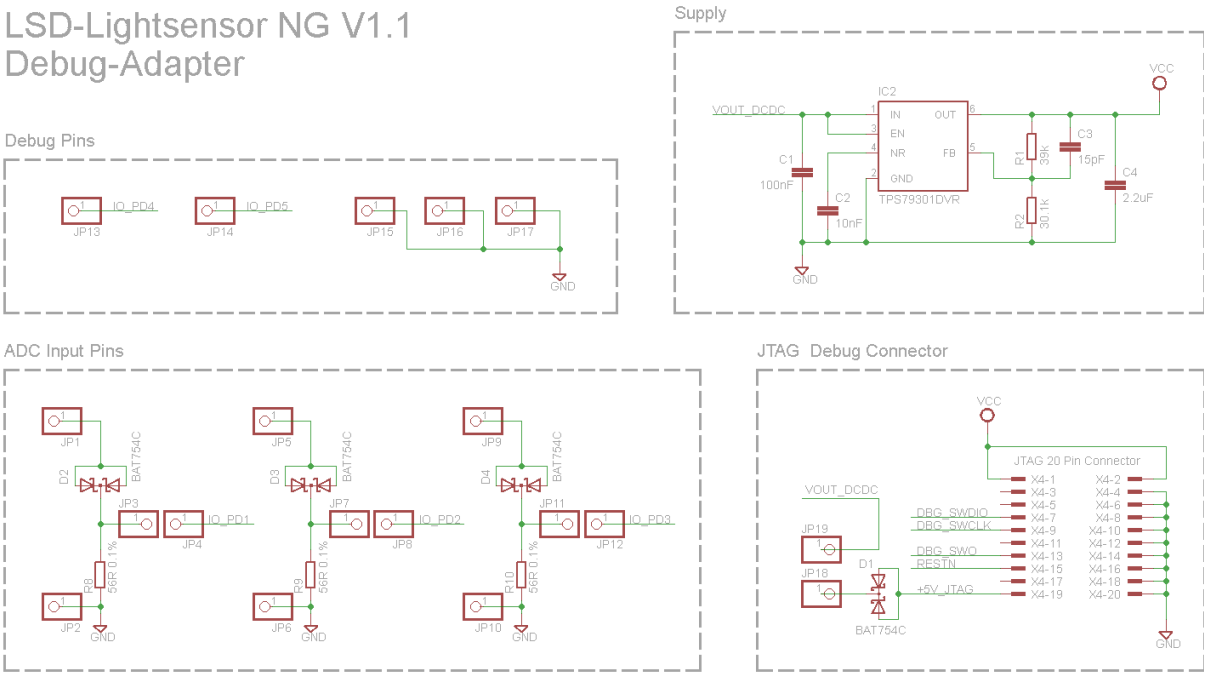

Abbildung 63: LSD-Lichtsensor NG Debug-Adapter - Teil 1
Module: Debug Pins, Supply, ADC Input Pins und JTAG Debug-Connector

Debug Pins, ADC Input Pins & Sensor I/O Connector

Einige GPIO-Pins des Mikrocontrollers sowie mehre Ground-Pins sind hier zu finden. Drei der GPIO-Pins können mit den ADCs des Mikrocontrollers verbunden und so zur Spannungsmessung verwendet werden. Zur Messung von Referenzsignalen aus der Stromschnittstelle anderer LSD-Lichtensoren sind 56 Ω Messwiderstände und ein Verpolschutz mittels Diode implementiert. Über die Steckbrücken lässt sich diese Funktion individuell mit den Mikrocontroller-Pins verbinden.

Alle beschriebenen Pins werden über den Sensor-IO Stecker (siehe unten) mit der LSD-Lichtsensor NG Platine verbunden. Die Versorgungsleitungen sind durch Steckbrücken trennbar.

Supply

Der Spannungsregler IC2 bezieht seine Eingangsspannung aus dem JTAG-Programmieradapter (+5V_JTAG) und versorgt den RS232 Transceiver. Auf Wunsch kann er auch die Versorgung eines angeschlossenen Lichtsensors NG übernehmen, einstellbar über Steckbrücken.

Beim Spannungsregler handelt es sich um den gleichen Typ wie er auch auf der Sensorplatine zu finden ist. Auch die eingestellte Spannung von 2,8 V ist identisch.

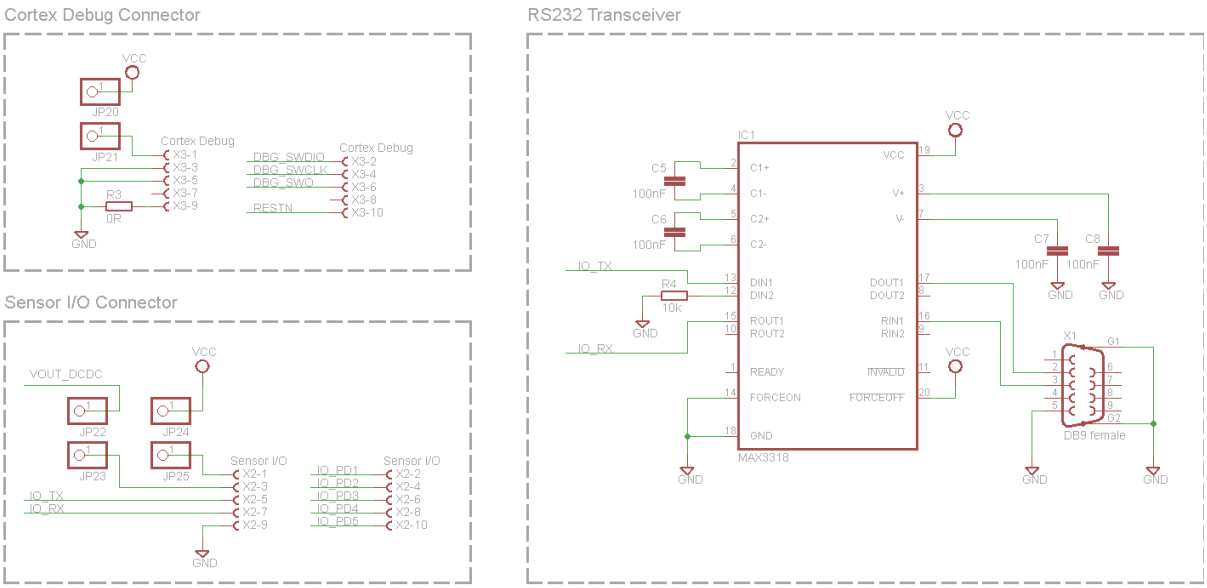

Abbildung 64: LSD-Lichtsensor NG Debug-Adapter - Teil 2
Module: RS232 Transceiver, Cortex Debug Connector sowie Sensor IO Connector

JTAG Debug Connector & Cortex Debug Connector

Die meisten Programmiergeräte verfügen über einen JTAG-Stecker zur Mikrocontroller-Programmierung. Alle Signale des JTAG-Steckers sind mit Ausnahme der 5 V Versorgung 1:1 auf den Cortex Debug Connector geroutet. Die 5 V Versorgung ist einerseits, trennbar durch eine Steckbrücke, mit der Leitung VOUT_DCDC der Sensorplatine verbunden, anderseits dient sie als Eingangsspannung für den Spannungsregler. Diode D1 schützt das Programmiergerät vor Überspannung von der Sensorplatine.

RS232 Transceiver

Die Umsetzung der UART-Signale des Mikrocontrollers zu RS-232 kompatiblen Signalen erledigt der Baustein IC1. Dieser RS232 Transceiver vom Typ Texas Instruments MAX3318E [60] arbeitet ähnlich dem bekannten MAX232. Der Hauptunterschied ist der niedrige mögliche Eingangsspannungsbereich von 2,25-3 V. Viele andere Wandlerbausteine brauchen eine höhere Versorgungsspannung. Um mehrere Spannungs-Domains in der Schaltung zu vermeiden ist diese Feature hier wichtig. Die Beschaltung erfolgt lt. Datenblatt. [60]

5.5.3 Layout und Aufbau

Das Layout der Sensorplatine ist gekennzeichnet durch die relativ begrenzten Platzverhältnisse. So soll die Platine elektrisch einwandfrei funktionieren und mechanisch in das vorhandene Sensorgehäuse passen. Die unregelmäßige, achteckige Kontur muss gefräst werden. Auch die vier Bohrungen sind wegen dem mechanischem Verdrehschutz genau zu fertigen. Um diesen zu realisieren sind die Abstände zwischen der großen und der kleinen Bohrung auf der linken und der rechten Seite geringfügig unterschiedlich.

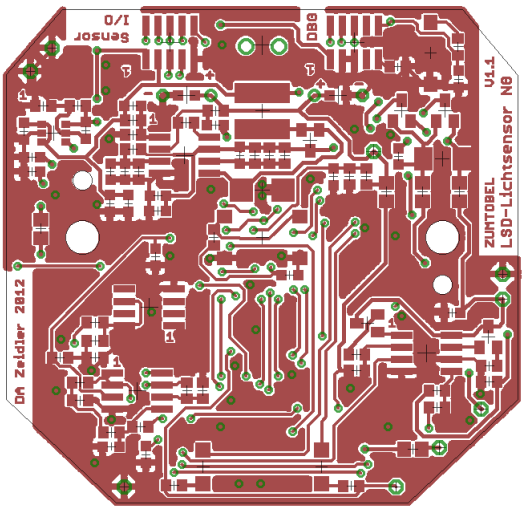

Abbildung 65: LSD-Lichtsensor NG Layout Oberseite mit Stromregelung, DC/DC-Konverter, Stecker, Taster und LEDs

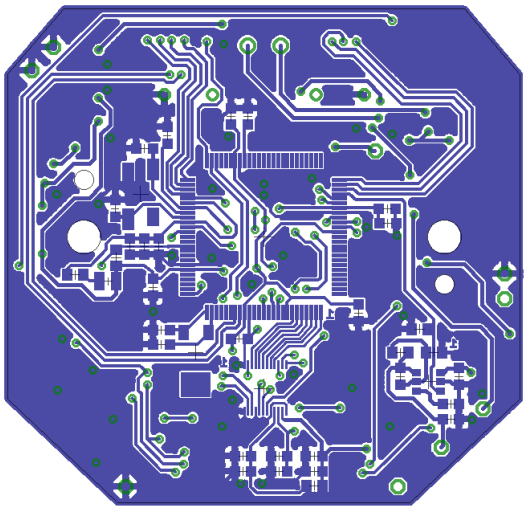

Abbildung 66: LSD-Lichtsensor NG Layout Unterseite mit Mikrocontroller und Kamerastecker

Wie im Layout zu erkennen ist, ist die Platine doppelseitig bestückt. Der Mikrocontroller mit Beschaltung sowie der Kamerastecker mit dem Spannungsregler für die Kamera befindet sich auf der Unterseite, alle anderen Bauteile auf der Oberseite. Die nächsten Abbildungen zeigen den Sensor fertig aufgebaut.

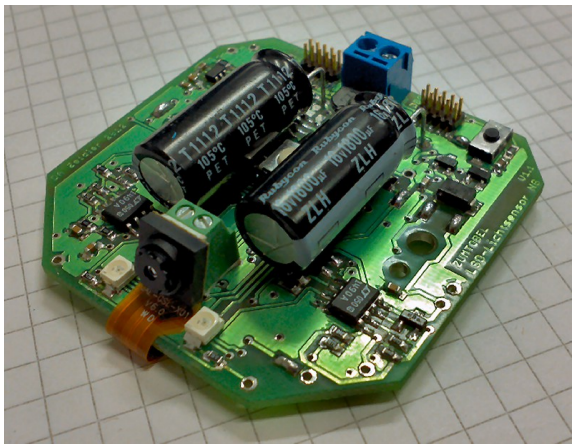

Abbildung 67: LSD-Lichtsensor NG fertig aufgebaut (Oberseite)

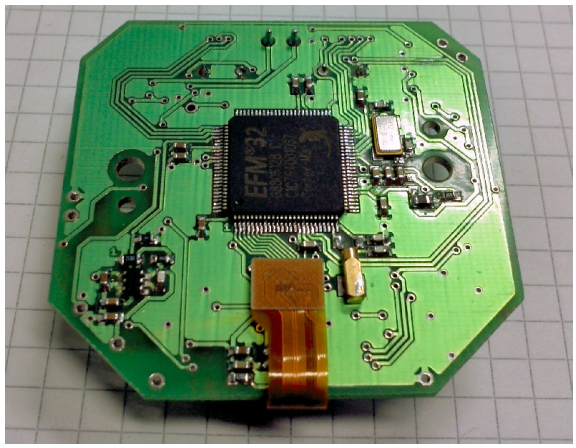

Abbildung 68: LSD-Lichtsensor NG fertig aufgebaut (Unterseite)

Im linken Bild erkennt man im Vordergrund das Kameramodul, links und rechts daneben die beiden LEDs und dahinter die zwei großen Speicher-Elkos. Ganz im Hintergrund ist die Klemme für die 4-20 mA Stromschnittstelle zu sehen (blau) und die Stiftleisten für den Debug-Adapter bzw. das Sensor-IO Interface. Zwischen den beiden Elkos befindet sich der Taster für die Tag-Systempunkt-Programmierung. Die Stromregelung, DC/DC-Konverter usw. verteilen sich auf der Oberfläche.

Das rechte Bild zeigt die Unterseite des Sensors. Zu sehen ist der Mikrocontroller mit Beschaltung sowie das Flexkabel des Kameramoduls befestigt am Stecker.

Debug-Adapter

Weniger aufwändig präsentiert sich das Layout der Debug-Adapter Platine. Da keine Größeneinschränkungen vorhanden sind, können die Teile bequem platziert werden. Den meisten Platz brauchen der JTAG-Stecker und die Buchse der RS-232-Schnittstelle. Die folgenden Abbildungen zeigen das Layout und die fertig aufgebaute Version. Die Platinen-Unterseite enthält keine Bauteile und nur wenige Leiterbahnen. Daher ist sie nicht abgebildet.

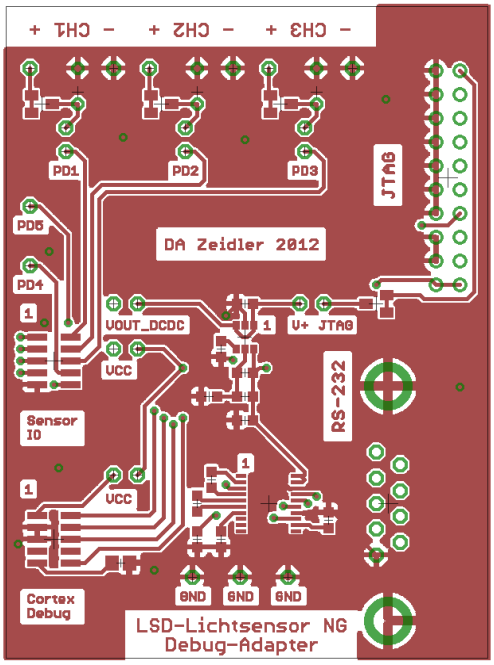

Abbildung 69: LSD-Lichtsensor NG Debug-Adapter - Layout

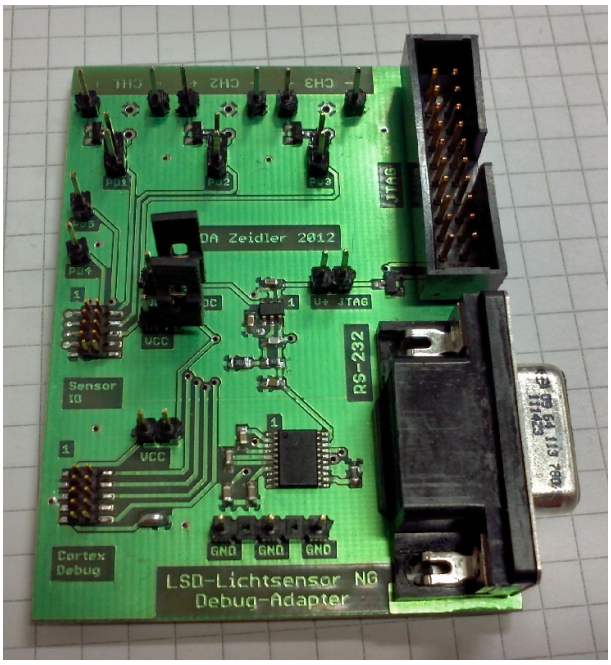

Abbildung 70: LSD-Lichtsensor NG Debug-Adapter - fertig aufgebaut

5.6 Messungen mit dem Prototypen

Der folgende Abschnitt dieser Arbeit widmet sich der praktischen Überprüfung der Funktion des Lichtsensor-Prototypen. Es wird untersucht, welche Betriebsparameter möglich sind und welche Messergebnisse die Lichtmessung liefert.

5.6.1 Messaufbau

Der Messaufbau bildet die Umgebung der LSD-Lichtsensoren in der 4-20 mA Schnittstelle nach. Ein Labornetzgerät stellt die Versorgung zur Verfügung und über ein Multimeter wird der Schleifenstrom gemessen. Zur Messung der Spannung am Speicherkondensator dient ein Oszilloskop. Ein PC zur Datenaufzeichnung ist via Debug-Adapter mit der Sensorplatine verbunden.

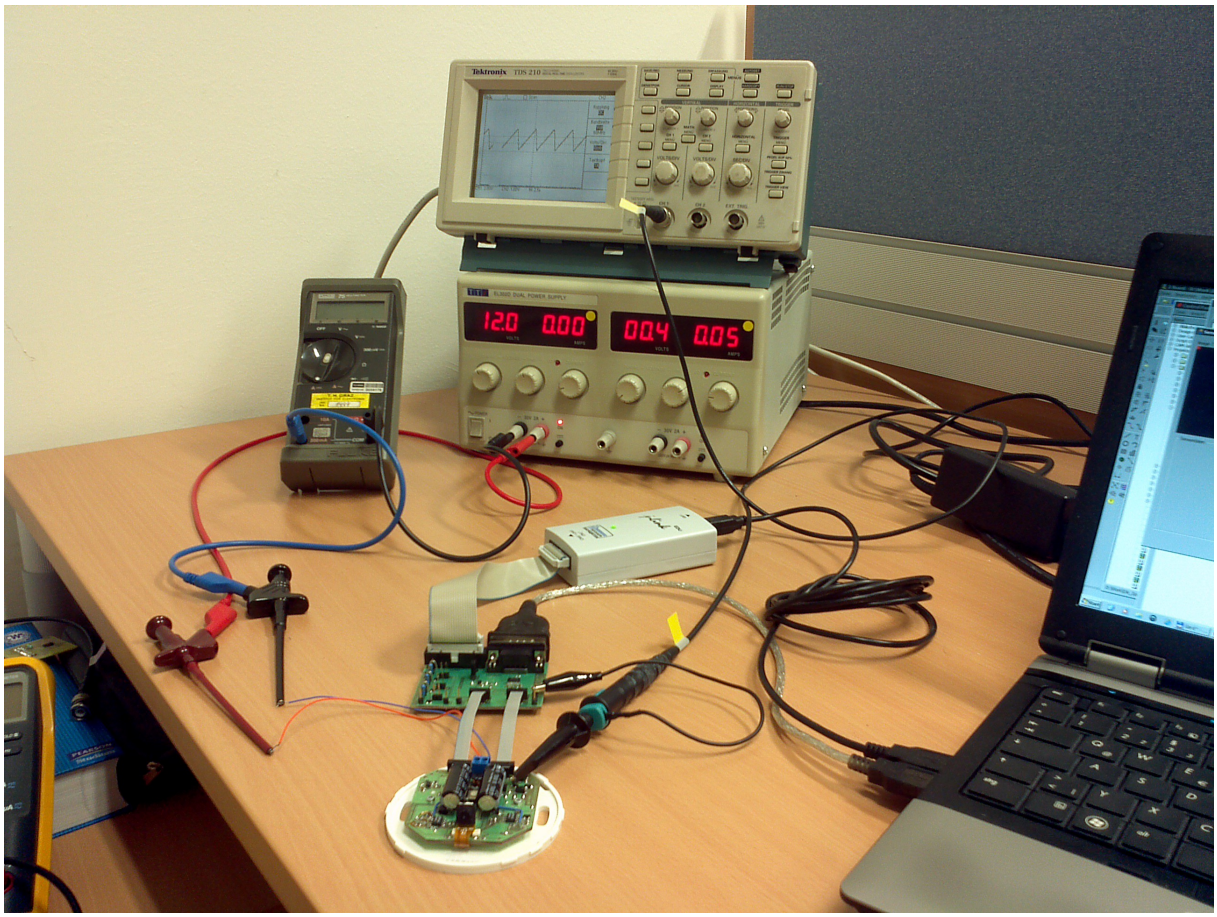

Abbildung 71: Messaufbau LSD-Lichtsensor NG in Stromschleife

5.6.2 Betriebsgrenzen

Die erste Messung testet die Betriebsgrenzen des Sensors. Abhängig vom eingestellten Schleifenstrom wird die minimale Schleifenspannung ermittelt, bei der der Sensor noch zuverlässig funktioniert. Auch die Bildfolgezeiten und damit die Messwertfolge wird ermittelt. Um diese Messwerte zu erhalten ist eine speziell angepasste Firmware notwendig, die den Schleifenstrom unabhängig vom Lichtmesswert einstellen kann. Die folgende Tabelle zeigt die Ergebnisse.

Schleifenstrom	notwendige Kondensatorspannung	Schleifen-spannung	min. Bildfolge (bei 12V)	Messwertfolge
4-8 mA	9,2 V	10,4 V	ca. 8 s	ca. 24 s
8-12 mA	9,0 V	10,2 V	ca. 3,6 s	ca. 10,8 s
12-16 mA	8,8 V	10,0 V	ca. 2,3 s	ca. 6,9 s
16-20 mA	8,6 V	9,8 V	ca. 1,6 s	ca. 4,8 s

Tabelle 3: Betriebsgrenzen LSD-Lichtsensor NG Prototyp

Die ersten Messungen an dem Prototypen zeigen positive Ergebnisse. Die verbaute Speicherkapazität von 2 x 1800 µF ist für alle Betriebsfälle ausreichend. Es ist sogar etwas Spielraum für mögliche Erweiterungen vorhanden. Die minimale Schleifenspannung von 9,8-10,4 V, je nach Betriebsfall, ist deutlich besser als die eingangs geforderten 12 V.

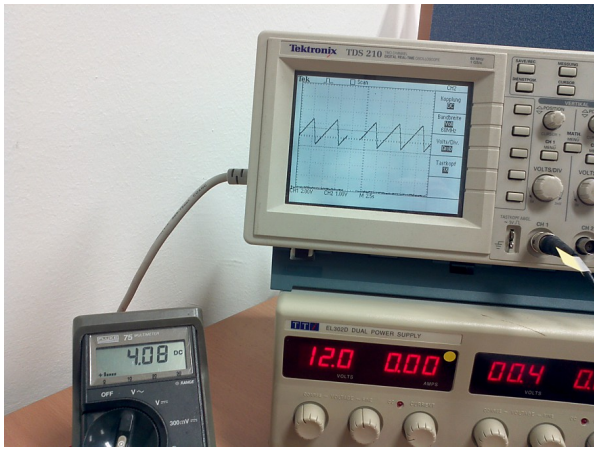

Abbildung 72: Messung bei 4 mA Schleifenstrom

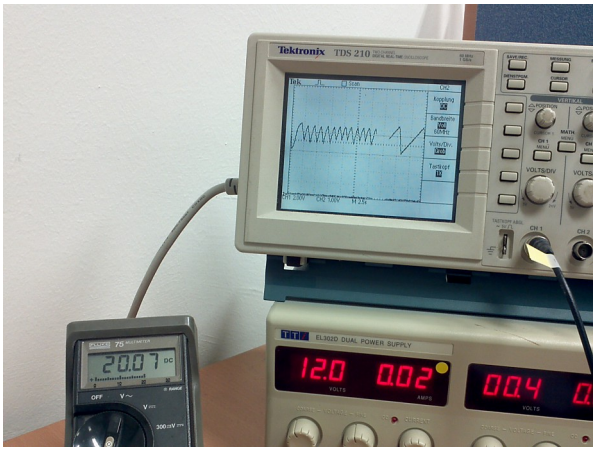

Abbildung 73: Messung bei 20 mA Schleifenstrom

Der Unterschied von 1,2 V zwischen der notwendigen Spannung am Speicherkondensator und der minimale Schleifenspannung wird hauptsächlich durch die Diodenbrücke am Eingang der Stromschleife verursacht. Ein anderer Grund dafür ist, dass der Massepunkt der Schaltung etwas oberhalb der negativen Schleifenspannung liegt (siehe Kapitel 5.5.1).

Nachdem diese Werte ermittelt sind, kann die Firmware so angepasst werden, dass die Bildaufzeichnung immer erst dann gestartet wird, wenn zumindest die notwendige Spannung am Speicherkondensator vorhanden ist. Andererseits wird der Kondensator immer bis zur höchstmöglichen Spannung aufgeladen um die Bildfolge zu erhöhen. Beurteilen kann der Mikrocontroller den Wert der maximalen Spannung, in dem sich die Messwerte nicht mehr bzw. nur noch sehr gering ändern.

5.6.3 Lichtmessung

Der Sensor verwendet zur Lichtmessung und Qualifizierung der Lichtanteile den in Kapitel 4.5.3 vorgestellten Algorithmus. Da die Berechnung des HDR-Bildes funktioniert, sind bei der Helligkeitsgewichtung keine Überraschungen zu erwarten. Für die nachfolgenden Messungen wird das selbe Fenster wie bisher verwendet, allerdings mit geändertem Blickwinkel.

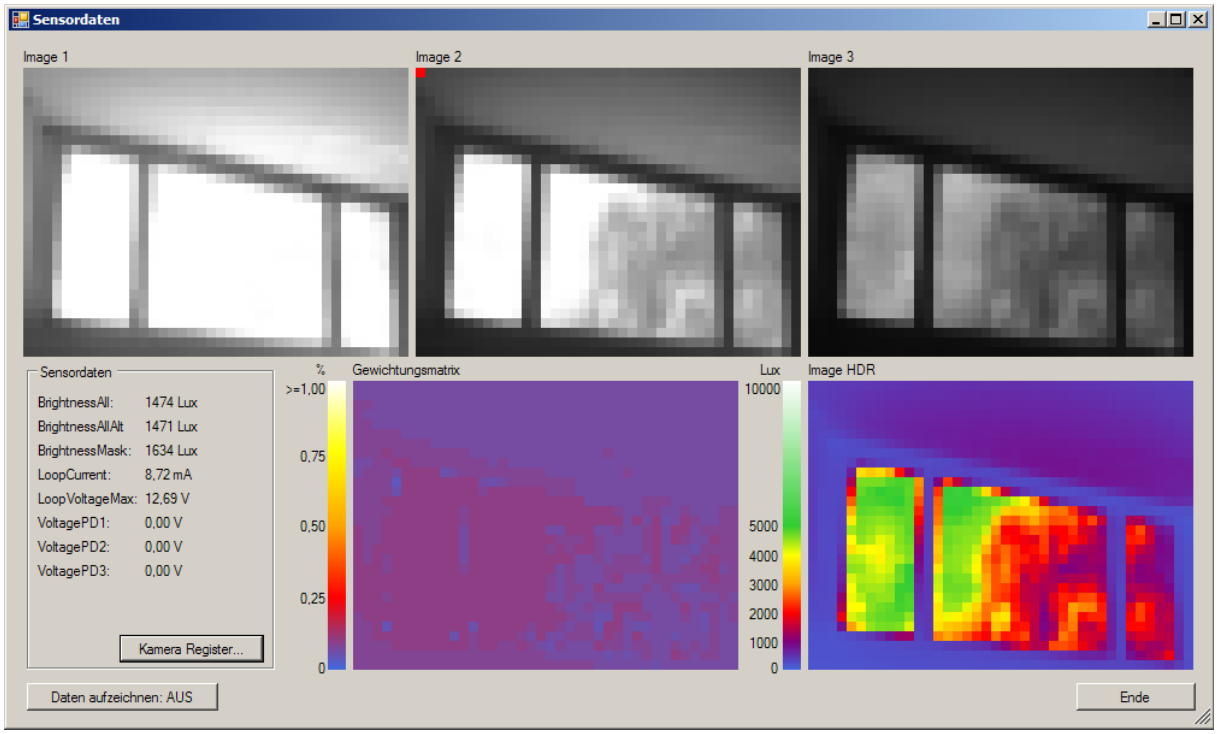

Abbildung 74: Sensormesswerte vom LSD-Lichtsensor NG Prototyp

Die obige Abbildung zeigt die Messwerten vom Sensor-Prototypen. Alle Werte stammen direkt vom Sensor, das Bildschirm-Fenster dient nur der Anzeige. Die oberen drei SW-Bilder zeigen die bekannte Belichtungsreihe, rechts unten ist das daraus errechnete HDR-Bild zu sehen. Links daneben befindet sich die Abbildung der Gewichtungsmatrix kurz nach der Initialisierung. Auf der linken Seite sind die aktuellen Messwerte zu erkennen.

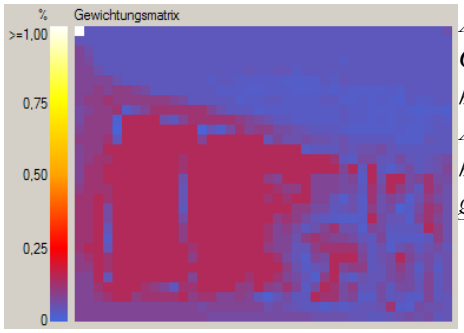

Abbildung 75: Gewichtungsmatrix nach einigen hundert Messzyklen; Der Algorithmus hat einige Bereiche höher und andere niedriger gewichtet

Abbildung 76: Gewichtungsmatrix nach weiteren Messzyklen; Die Fensterflächen treten bereits deutlich hervor

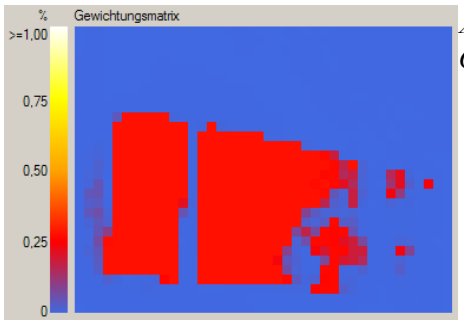

Abbildung 77:
Gewichtungsmatrix nach 6h;

In den obigen Abbildungen der Gewichtungsmatrix fällt ein Teil der Fensterfläche in den Bereich der niedrigen Gewichtung. Das ist auf ein dunkles Objekt (Baum) vor dem Fenster zurückzuführen. Hier besteht sicher noch etwas Optimierungspotential für den Algorithmus um auch solche Fensterflächen sicher erkennen zu können.

Andererseits werden kaum Flächen höher gewichtet, die nicht zum Fenster gehören. Das würden den Messwert stärker verfälschen als die nicht erkannten Fensterteile.

5.7 Schlussbemerkung

Durch die Fertigstellung und den Test der Sensor-Prototypen ist gezeigt worden, dass die Lichtmessung mit Hilfe eines Kameramoduls analog zum bisherigen LSD-Lichtsensor auf Basis einer Photodiode funktioniert. Durch einen geeigneten Pixel-Gewichtungsalgorithmus können Fensterflächen zuverlässig erkannt und die für die Lichtmessung relevanten Lichtanteile herausgerechnet werden. Dadurch ergibt sich die in der Aufgabenstellung gewünschte Verbesserung des Messsignales. Die Installationsvorschriften des LSD-Lichtsensors können also gelockert werden.

Auch ist es gelungen den Sensor ausschließlich über die 4-20 mA Stromschnittstelle zu versorgen, trotz des vergleichsweise hohen Stromverbrauchs des Kameramoduls. Die dazu notwendigen Speicherkondensatoren konnten im bestehenden Sensorgehäuse untergebracht werden.

Um den Sensor zur Serienreife fertig zu entwickeln, sind aber noch einige Tests und Optimierungsschritte notwendig:

Auf der Algorithmus-Seite sind das weitergehende Kalibrierungsarbeiten, d.h. der Vergleich des Messsignals mit den vorhandenen LSD-Lichtsensoren an weiteren, repräsentativ ausgewählten Standorten. Ebenso kann der Algorithmus zur Erkennung von Kunstlichtquellen erweitert werden um auch diese Störgröße aus dem Messsignal zu entfernen. Ideen hierfür sind bereits vorhanden.

Auf der Firmware-Seite könnten Optimierungen zur Messwert-Ausgabe (Stichwort Energy-Modi des Mikrocontrollers noch besser nützen) sowie eine weitere Verbesserung der Ansteuerung des Kameramoduls realisiert werden. Wenn es gelingt die Aufnahmezeit für ein Bild zu verringern, kann entweder ein kleinerer Speicherkondensator Verwendung finden oder bei gleicher Kapazität mehrere Bilder hintereinander aufgenommen werden. Dadurch würde sich die Messwertfolge erhöhen und somit die Reaktionszeit der gesamten Lichtsteuerung.

Auch bei der Sensor-Hardware kann optimiert werden. Die nur für Debuggingzwecke verbauten Bauteile wie Stecker, zweite LED, Reset-Taster, etc. können weggelassen werden. Für die Programmierung der Firmware ist z.B kein Cortex-Debug-Stecker notwendig. Der Prozessor verfügt im Auslieferungszustand über einen seriellen Bootloader über welchen die Firmware via UART eingespielt werden kann.

Beim Prozessormodell kann auf eine kleinere (und billigere) Variante mit weniger Anschlusspins zurückgegriffen werden. Es gibt auch Mikrocontroller-Varianten, die bereits Operationsverstärker integriert haben. Möglicherweise könnten so die externen Operationsverstärker eingespart werden.

6 Anhang

Der Anhang umfasst eine Beschreibung der Firmware-Funktionen samt Quellendokumentation sowie eine Beschreibung der PC-Software zur Auswertung der Sensordaten.

6.1 Lichtsensor-Firmware

Wie in Kapitel 5.1.3 dargelegt, entstand die Sensor-Firmware mit Hilfe der Entwicklungsumgebung der Firma Keil. Als Programmiersprache findet ausschließlich C und kein Assemblercode Verwendung. Die folgende Abbildung zeigt die Entwicklungsumgebung samt Programmstruktur.

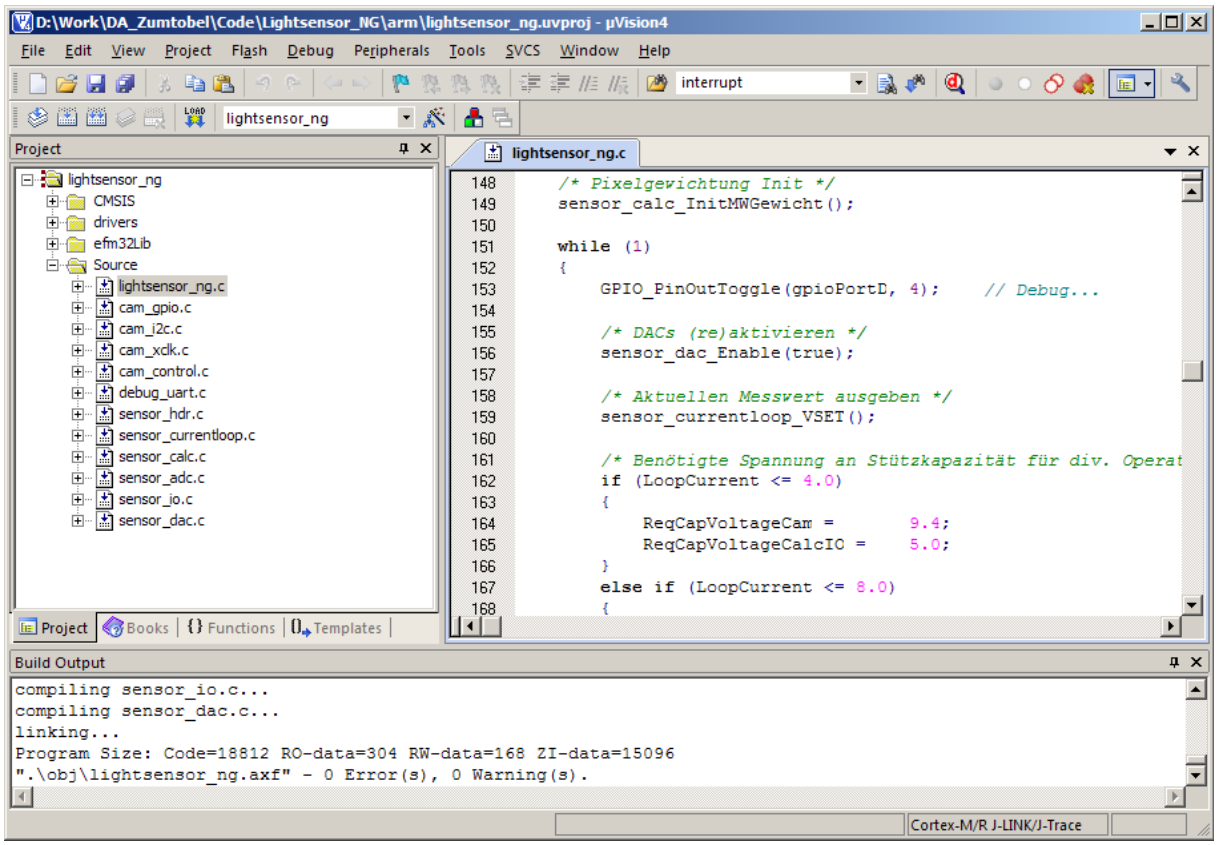

Abbildung 78: Entwicklungsumgebung und Programmstruktur

In der linken Fensterhälfte ist der Sourcecode-Baum dargestellt. Das Projekt besteht aus mehreren Files, die thematisch getrennt sind. Aufgrund der guten, vom Hersteller des Mikrocontrollers mitgelieferten Libraries, kann die Ansteuerung der Mikrocontroller-Module mit relativ wenig Sourcecode erfolgen. Das hält das Programm übersichtlich.

6.1.1 Funktionen

Aus Sicht des Anwenders verfügt die Firmware über folgende Funktionen: Nach dem Start des Sensors, sobald genug Energie in den Speicherkondensatoren vorhanden ist, beginnt ein Messzyklus. Eine Belichtungsreihe mit drei Bildern wird aufgenommen und daraus das HDR-Bild berechnet. Im Anschluss wird die Gewichtungsmatrix aktualisiert, der Messwert berechnet und via Stromschnittstelle übermittelt. Zwischendurch, jeweils wenn ein Teil der Berechnung fertig ist, werden die Daten auch über die digitale Schnittstelle ausgegeben. Diese Schleife wiederholt sich bis die Stromversorgung unterbrochen wird.

Bei geringen Schleifenströmen dauert es mehrere Sekunden bis genug Energie vorhanden ist um eine Aufnahme anzufertigen. Um dem Anwender ein Feedback über den aktuellen, internen Zustand zu geben, blinkt während des Aufladevorgangs die rote LED immer wieder kurz auf. Beim Auslösen einer Bildaufnahme blinkt die grüne LED kurz auf. Diese Blinks benötigen kaum Energie und stören das Energiemanagement nicht. Möglich ist das durch das Mikrocontroller-Feature den Treiberstrom der Ausgangs-Pins auf 0,6 mA zu begrenzen.

Drückt der User den Taster zur Tag-Systempunkt-Programmierung, wird der Schleifenstrom auf den Maximalwert von 20 mA geregelt und die grüne LED leuchtet hell auf. In diesem Fall wird der Treiberstrom nicht begrenzt, wodurch bis zu 20 mA/Pin zur Verfügung stehen.

Der Taster hat noch eine zweite Funktion: Langes Drücken löst den Testmodus aus. Der besteht darin, den Schleifenstrom alle paar Sekunden treppenförmig von 4 mA zu 20 mA zu erhöhen. Nachher kehrt der Sensor in den Normalbetrieb zurück.

6.1.2 Quellendokumentation

Nachfolgend sind wichtige Teile der Sensor-Firmware im Quellcode dokumentiert. Es handelt sich aufgrund des Umfangs um Auszüge, welche jeweils nur die relevanten Stellen zeigen.

globals.h

Die Sensor-Firmware arbeitet viel mit globalen Daten. LDR-Bilder, HDR-Bild, Gewichtungsmatrix, usw. In den nachfolgenden Codeteilen wird immer wieder auf diese Variablen Bezug genommen.

```
//---------------------------------------------------------------------------
//  Globale Variablen
//---------------------------------------------------------------------------

// In main.c ist EXTERN bereits als Leerstring definiert, somit werden die
// globalen Variablen einmal deklariert (z.B. int a), in allen anderen Files
// werden die Variablen als extern eingebunden da EXTERN nicht deklariert
// ist (z.B. extern int a).
#ifndef EXTERN
#define EXTERN extern
#endif

/*****************************************************************************//**
```

```
 * @brief Globale Variablen
 ***************************************************************************/

// Buffer für Capture
EXTERN volatile uint8_t lineBuffer[128];       // Linebuffer für Capture (0.5kB)
EXTERN volatile uint16_t imageBuffer[30][40];  // Imagebuffer für Capture (2.4kB)

// LDR Imagedaten
EXTERN volatile uint8_t image1[30][40];        // LDR Image1 (1.2kB)
EXTERN volatile uint8_t image2[30][40];        // LDR Image2 (1.2kB)
EXTERN volatile uint8_t image3[30][40];        // LDR Image3 (1.2kB)

// HDR Image
EXTERN volatile uint16_t imageHDR[30][40];     // HDR Image (2.4kB)

// Pixelwert vorherige Aufnahme bzw. Pixelgradient
EXTERN volatile uint8_t imageHDRPixGradient[30][40];    // HDR Image: Pixelgradient statt Pixelwert wegen Speicherverbrauch

// Gewichtungsmatrix
EXTERN volatile float MWGewicht[30][40];       // Gewichtungsmatrix (4.8kB)

// Flags, Zählvariablen
EXTERN volatile bool camCaptureActive;         // true solange ein Bild gelesen wird
EXTERN volatile uint8_t* ptrBuffer;            // Zeiger im Zeilenbuffer
EXTERN volatile  int countHREF;
EXTERN volatile  int countHREFmax;
EXTERN volatile  int countVSYNC;
EXTERN volatile  int countPCLK;


// Globale Messdaten
EXTERN volatile  int BrightnessAll;            // Helligkeitsmesswert Gesamtbildes
EXTERN volatile  int BrightnessAllAlt;         // Helligkeitsmesswert Gesamtbildes (alter Wert)
EXTERN volatile  int BrightnessMask;           // Helligkeitsmesswert aufgrund der Gewichtungsmatrix
EXTERN volatile  float LoopCurrent;            // Schleifenstrom in mA
EXTERN volatile  float LoopVoltageMax;         // Schleifenspannung (Maximalwert des letzen Messzyklus)
EXTERN volatile  float VoltagePD1;             // ADC Sample von PD1 (Spannung auf bezogen auf 2.5V Referenz)
EXTERN volatile  float VoltagePD2;             // ADC Sample von PD2 (Spannung auf bezogen auf 2.5V Referenz)
EXTERN volatile  float VoltagePD3;             // ADC Sample von PD3 (Spannung auf bezogen auf 2.5V Referenz)

/* Defines */
#define I2C_ADDR_W 0x60;
#define I2C_ADDR_R 0x61;
//#define HFXO_FREQUENCY           32000000
//#define UART_PERCLK_FREQUENCY    HFXO_FREQUENCY
//#define UART_BAUDRATE            115200

/* Typedefs */
typedef struct
{
    uint16_t     addr;
    uint8_t  data;
} CamData_TypeDef;

/* Enums */
typedef enum
{
    txStatus = 0x00,              // Statusdaten allgemein (Messwerte, ...)
    txImage1 = 0x01,              // LDR Image1
    txImage2 = 0x02,              // LDR Image2
    txImage3 = 0x03,              // LDR Image3
    txImage4 = 0x04,              // LDR Image4 (optional)
    txImage5 = 0x05,              // LDR Image5 (optional)
    txImageFullScale = 0x06,      // Image mit Maximalauflösung (z.B. 320x200, optional)
    txImageHDR = 0x07,            // HDR Image
    txMWGewicht = 0x08,           // Gewichtungsmatrix
    txCamRegister3000 = 0x09,     // Registersatz der Kamera senden Offset 0x3000
    txCamRegister3100 = 0x10,     // Registersatz der Kamera senden Offset 0x3100
    txCamRegister3300 = 0x11,     // Registersatz der Kamera senden Offset 0x3300
    txCamRegister3400 = 0x12,     // Registersatz der Kamera senden Offset 0x3400
    txCamRegister3600 = 0x13      // Registersatz der Kamera senden Offset 0x3600

} TxFrameTyp_Enum;

typedef enum
{
    rxSendData = 0x80,
    rxCamRegister = 0x81

} RxFrameTyp_Enum;
```

lightsensor_ng.c

Dieses File beinhaltet den Startpunkt des Firmware-Programms. Die Startroutine mit der obligatorischen Endlosschleife zur Lichtmessung ist auszugsweise abgedruckt.

```
/**************************************************************************
 * @brief  Main function
 *************************************************************************/
int main(void)
{
    /* Initialize chip */
    CHIP_Init();
```

```
/* Enable HFXO as high frequency clock (32MHz) */
CMU->OSCENCMD = CMU_OSCENCMD_HFXOEN;
while (!(CMU->STATUS & CMU_STATUS_HFXORDY)) ;
CMU->CMD = CMU_CMD_HFCLKSEL_HFXO;

/* Enable GPIO clock */
CMU->HFPERCLKEN0 |= CMU_HFPERCLKEN0_GPIO;

/* CAM_VDD_EN (PF3) AUS!!! -> sonst zieht die Kamera Strom und der Startup funktioniert nicht !!! */
GPIO_PinModeSet(gpioPortF, 3, gpioModeWiredOrPullDown, 0);  //init: CAM_VDD (AVDD+DVDD) aus  (mit Pull-Down!)

/* ADC Setup */
sensor_adc_Config();

/* Warten auf genug Startup Energy in Kap. */
sensor_adc_WaitStartup(3.6, 500);

/* DAC: 4mA Schleifenstrom setzen: Achtung nicht vorher, bis hierher muss die Startenergie ausreichen...*/
sensor_dac_Setup();
sensor_dac_Enable(true);
LoopCurrent = 20.0;                  // Startup: 20mA damit das erste Aufladen schnell erfolgt
sensor_currentloop_VSET();           // Ausgabe Schleifenstrom

/* Wait for enough startup energy in cap. */
sensor_adc_WaitStartup(4.5, 100);

/* LEDs, Button Setup */
GPIO_PinModeSet(gpioPortF, 7, gpioModePushPullDrive, 1);    // LED green
GPIO_PinModeSet(gpioPortF, 8, gpioModePushPullDrive, 1);    // LED red
GPIO_PinModeSet(gpioPortF, 9, gpioModeInputPullFilter, 1);  // Button
GPIO_DriveModeSet(gpioPortF, gpioDriveModeLowest);          // Drivemode: Lowest

/* Debug-Pin(s) Setup */
GPIO_PinModeSet(gpioPortD, 4, gpioModePushPullDrive, 0);
GPIO_DriveModeSet(gpioPortD, gpioDriveModeLowest);
GPIO_PinOutToggle(gpioPortD, 4);                            // Debug Puls

/* Setup UART */
CMU->HFPERCLKEN0 |= CMU_HFPERCLKEN0_USART1;    // Enable clock for USART1
UART_setupUart(1, 0);                          // Location: 0 (PC0, PC1), USART1

/* Variablen (hier definiert um besser debuggen zu können) */
static uint32_t sample;
static float VSense;
static char buffer[30];
static float VREF = 2.5;                       // VSS = 2.5V definiert in ADCGetSample

/* Statusblink rote + grüne LED */
sensor_io_RedBlink();
sensor_io_GreenBlink();

/* Statusausgabe Spannung am Kondensator */
sample = sensor_adc_GetSample(adcSingleInpCh0);
VSense = (float)sample * (VREF/4096.0) * (660.0/100.0);  //Spannungsteiler für VSense: 560k/100k
sprintf(buffer, "INIT VSenseStart  %4u  %1.3f V\n", sample, VSense);
USART1_sendBuffer(buffer, sizeof(buffer));

float ReqCapVoltageCam;                    // benötigte Cap-Spannung bei gg. Schleifenstrom für Kamera
float ReqCapVoltageCalcIO;                 // benötigte Cap-Spannung bei gg. Schleifenstrom für Berechung und I/O

/* Pixelgewichtung Init */
sensor_calc_InitMWGewicht();

while (1)
{
    GPIO_PinOutToggle(gpioPortD, 4);       // Debug...

    /* DACs (re)aktivieren */
    sensor_dac_Enable(true);

    /* Aktuellen Messwert ausgeben */
    sensor_currentloop_VSET();

    /* Benötigte Spannung an Stützkapazität für div. Operationen berechnen */
    if (LoopCurrent <= 4.0)
    {
        ReqCapVoltageCam =         9.4;
        ReqCapVoltageCalcIO =      5.0;
    }
    else if  (LoopCurrent <= 8.0)
    {
        ReqCapVoltageCam =         9.2;
        ReqCapVoltageCalcIO =      5.0;
    }
    else if  (LoopCurrent <= 12.0)
    {
        ReqCapVoltageCam =         9.0;
        ReqCapVoltageCalcIO =      5.0;
    }
    else if  (LoopCurrent <= 16.0)
    {
        ReqCapVoltageCam =         8.8;
        ReqCapVoltageCalcIO =      5.0;
    }
    else
    {
        ReqCapVoltageCam =         8.6;
        ReqCapVoltageCalcIO =      5.0;
    }
```

```
        /* Image 1 */

        /* Warten auf genug Startup Energy in Kap. */
        sensor_adc_Wait4Energy(ReqCapVoltageCam, true);

        GPIO_PinOutToggle(gpioPortD, 4);    // Debug...

        /* Kameramodul aktivieren */
        cam_control_Init();
        cam_control_CamON();              // Kamera start (VDD ein + Reset + Start)
        cam_control_ConfigInit();         // Config laden (Y8, ca. 10MHz PCLK, ...)
        cam_control_EnableGPIO(true);     // Kamera GPIO enable

        cam_control_ConfigImage1();       // Config laden (Y8, ca. 16MHz PCLK, ...)

        GPIO_PinOutToggle(gpioPortD, 4);  // Debug...

        cam_control_StartCapture();       // VSYNC aktivieren, Variablen Reset
        while (camCaptureActive);         // Auf Bild warten

        GPIO_PinOutToggle(gpioPortD, 4);  // Debug...

        cam_control_CamOFF();             // Kamera ausschalten - Energie sparen

        /* Warten auf genug Startup Energy in Kap. */
        sensor_adc_Wait4Energy(ReqCapVoltageCalcIO, false);

        GPIO_PinOutToggle(gpioPortD, 4);  // Debug...

        /* Imagebuffer nach Image1 schreiben und Bild berechnen */
        for(int imgX=0; imgX<40; imgX++)       // 40 spalten
            for(int imgY=0; imgY<30; imgY++)   // 30 zeilen
                    if (imgX & 1)              // ungerades Pixel
                        image1[imgY][imgX] = (uint8_t)(imageBuffer[imgY][imgX] / 30);    // Mittelwert berechnen
                    else                       // gerades Pixel
                        image1[imgY][imgX] = (uint8_t)(imageBuffer[imgY][imgX] / 20);    // Mittelwert berechnen

        /* Bild senden */
        debug_uart_SendData(txImage1);

[...] Bilder 2 und 3

        /* HDR Image */
        sensor_hdr_CalcHDRImage();
        debug_uart_SendData(txImageHDR);

        /* Warten auf genug Startup Energy in Kap. */
        sensor_adc_Wait4Energy(ReqCapVoltageCalcIO, false);

        /* Pixelgewichtung Update: Gewichtungsmatrix aktualisieren */
        sensor_calc_UpdatePixelGewicht();

        /* Gewichtungsmatrix Ausgabe */
        debug_uart_SendData(txMWGewicht);

        GPIO_PinOutToggle(gpioPortD, 4);      // Debug...

        /* Helligkeitswert ausgeben */
        sensor_calc_BrightnessAll();          // Gesamthelligkeit des Bildes berechnen
        LoopCurrent = (4.0f + ((float)BrightnessAll * 16.0f / 5000.0f));  // 5000 Lux auf 4-20mA aufteilen

        sensor_calc_BrightnessMask();         // Helligkeit aufgrund der Gewichtungsmatrix

        /* Statusdaten */

        /* Warten auf genug Startup Energy in Kap. */
        sensor_adc_Wait4Energy(ReqCapVoltageCalcIO, false);

        /* Messwerte */
        debug_uart_SendData(txStatus);
}
```

cam_gpio.c

Dieses File beinhaltet alle Routinen zur Ansteuerung der Kamera via GPIO-Pins. Nachfolgend ist die Routine zum samplen der Bilddaten abgedruckt.

```
/**************************************************************************//**
 * @brief GPIO_EVEN_IRQHandler  (HREF + VSYNC)
 * Interrupt Service Routine Even GPIO Interrupt Line
 *****************************************************************************/
void GPIO_EVEN_IRQHandler(void)
{
    // Info: 670ns bis hierher (ca. 21 Takte) seit Interrupttrigger

    // Variablendeklarationen (alle auf Funktionsebene damit sich der Debugger leichter tut)
    int imgY;                         // Bildindex Y
    int imgX;                         // Bildindex X
    int s;                            // Index für Samples/Pixel
    int smax;                         // Index für Samples/Pixel Obergrenze
    int i;                            // Index in lineBuffer
```

```
    // HREF und VSYNC Interrupts deaktivieren
    GPIO->IEN = 0x00;                 // alle Interrupts aus

    if (GPIO->IF & (1 << 2))          //HREF
    {
        // Info: 1380ns bis hierher seit Interrupttrigger (mit GPIO-IEN = 0x00; mit BITBAND ca. 2200ns)

        // für Debug: Puls auf PF1 (Sample start)
        GPIO->P[gpioPortF].DOUTTGL = 1 << 8;
        GPIO->P[gpioPortF].DOUTTGL = 1 << 8;

        // Linebuffer lesen
        for(i=0; i<SAMPLES; i++)      //Samples lesen - das geht sich genau aus für eine Zeile bei passenden Einstellungen
            lineBuffer[i] = (uint8_t)(GPIO->P[gpioPortA].DIN);

        // Info: hier ist das Ende der Zeile (HREF) erreicht, es folgt eine kurze Lücke

        // für Debug: Puls(e) auf PF1: Ende des Samplens markieren
        GPIO->P[gpioPortF].DOUTTGL = 1 << 8;
        GPIO->P[gpioPortF].DOUTTGL = 1 << 8;
        GPIO->P[gpioPortF].DOUTTGL = 1 << 8;
        GPIO->P[gpioPortF].DOUTTGL = 1 << 8;


        // Verarbeiten der Zeile
        // Zeilenindex bestimmen (1200 Zeilen müssen auf 30 Zeilen abgebildet werden 1200/4 = 300; 300/30=10)
        imgY = (countHREF / 10);      // (jede 4. Zeile wird gesampled, daher 40/4=10)
        i=0;

        // Zeilendaten in imageBuffer übertragen
        for(imgX=0; imgX<40; imgX++)  // 40 spalten in imageBuffer
        {
            smax = 2;
            if (imgX & 1)             // ungerades Pixel?
                smax = 3;

            for(s=0; s<smax; s++)     // 3 samples für jedes ungerade Pixel
                if (imgY < 30)        // Bounds Prüfung: wird anscheinend öfters überschritten, verhindert abstürze
                    (imageBuffer[imgY][imgX]) = (uint16_t)((imageBuffer[imgY][imgX]) + lineBuffer[i++]);
                    // samples aufaddieren (dividiert wird später) ACHTUNG: += Operator funktioniert hier anscheinend nicht
        }

        // kurze Warteschleife um ca. in die Mitte des nächsten HREF Zyklus zu kommen
        for(i=0; i<200; i++);         // Wartescheife so, dass jede 4. Zeile gesampled wird

        // Zeilenzähler anpassen
        countHREF++;

        // Interrupt Flag clear (alle)
        GPIO->IFC = 0xFF;

        // HREF und VSYNC Interrupts reaktivieren
        BITBAND_Peripheral(&(GPIO->IEN), 2, (unsigned int)true);    //HREF
        BITBAND_Peripheral(&(GPIO->IEN), 4, (unsigned int)true);    //VSYNC
    }
    else
    {
        // VSYNC
        countVSYNC++;

        // Interrupt Flag clear (GPIO->IFC Register, bit 2, bit 4)
        GPIO_IntClear(1 << 2);
        GPIO_IntClear(1 << 4);

        if (countVSYNC > 2)  // nach 3 Bildern Capture abbrechen
        {
            // Capture fertig signalisieren, Interrupts bleiben deaktiviert
            camCaptureActive = false;
        }
        else
        {
            GPIO->P[gpioPortF].DOUTTGL = 1 << 8;
            GPIO->P[gpioPortF].DOUTTGL = 1 << 8;

            countHREFmax = countHREF;     // Gesamtzeilen merken für Debug
            countHREF = 0;                // Href Counter reset
            countPCLK = 0;                // für debug

            // Imagebuffer löschen
            for(int imgX=0; imgX<40; imgX++)  // 40 spalten
                for(int imgY=0; imgY<30; imgY++)  // 30 zeilen
                    imageBuffer[imgY][imgX] = 0;

            // HREF und VSYNC Interrupts (re)aktivieren
            BITBAND_Peripheral(&(GPIO->IEN), 2, (unsigned int)true);    //HREF
            BITBAND_Peripheral(&(GPIO->IEN), 4, (unsigned int)true);    //VSYNC
        }
    }
}
```

cam_i2c.c

Das Übertragen der Konfigurationsdaten via I^2C an das Kameramodul ist in diesem File zusammengefasst. OmniVision nennt die I^2C-Schnittstelle SCCB.

```
/*************************************************************************//**
 * @brief SCCB schreiben: 1 Byte an Adresse
 *       addr: 16 Bit Adresse
 *       data: 8 Bit Daten
 *       (I2C Adresse ist global)
 *************************************************************************/
void cam_i2c_Write(uint16_t addr, uint8_t data)
{
    I2C_TransferSeq_TypeDef   seq;// Datenstruktur Sendesequenz
    I2C_TransferReturn_TypeDef    ret;// Rückgabecode

    uint8_t dataOut[3];                         // zu sendene Daten
    dataOut[0] = (uint8_t)((addr & 0xFF00) >> 8);  // Adresse High Byte
    dataOut[1] = (uint8_t)(addr & 0x00FF);         // Adresse Low Byte
    dataOut[2] = data;                          // Daten

    // Sequenzdaten setzen
    seq.addr = I2C_ADDR_W;
    seq.flags = I2C_FLAG_WRITE;
    seq.buf[0].data = dataOut;
    seq.buf[0].len = 3;
    //seq.buf[1].data = dataIn;
    //seq.buf[1].len = 3;

    // Do a polled transfer
    ret = I2C_TransferInit(I2C0, &seq);
    while (ret == i2cTransferInProgress)
    {
        ret = I2C_Transfer(I2C0);
    }
}

/*************************************************************************//**
 * @brief SCCB lesen: 1 Byte von Adresse
 *        addr: 16 Bit Adresse
 *        (I2C Adresse ist global)
 *************************************************************************/
uint8_t cam_i2c_Read(uint16_t addr)
{
    I2C_TransferSeq_TypeDef seq;                // Datenstruktur Sendesequenz
    I2C_TransferReturn_TypeDef ret;             // Rückgabecode

    // Adresse schreiben
    uint8_t dataOut[2];                         // zu sendene Daten
    dataOut[0] = (uint8_t)((addr & 0xFF00) >> 8);  // Adresse High Byte
    dataOut[1] = (uint8_t)(addr & 0x00FF);         // Adresse Low Byte

    uint8_t dataIn[1];                          // zu lesende Daten

    // Sequenzdaten setzen
    seq.addr = I2C_ADDR_R;
    seq.flags = I2C_FLAG_WRITE_READ;
    seq.buf[0].data = dataOut;
    seq.buf[0].len = 2;
    seq.buf[1].data = dataIn;
    seq.buf[1].len = 1;

    // Do a polled transfer
    ret = I2C_TransferInit(I2C0, &seq);
    while (ret == i2cTransferInProgress)
    {
        ret = I2C_Transfer(I2C0);
    }

    return dataIn[0];
}
```

cam_xclk.c

In cam_xclk.c wird nur das Clocksignal von 16 MHz für das Kameramodul via Timer erzeugt. Der Standardcode dazu kann in den Appnotes zum Prozessor nachgelesen werden und ist daher hier nicht abgedruckt.

cam_control.c

Das File cam_control.c implementiert den Power-Up und Power-Down des Kameramoduls laut Appnote. Außerdem sind hier die Registereinstellungen für das Modul zu finden.

```
/***************************************************************************//**
 * @brief Camera On - Start Kamera mit Power on
 ******************************************************************************/
void cam_control_CamON(void)
{
    // Spannungsversorgung Kamera ein
    // DOVDD=ein (2.8V über GPIO), AVDD=ein (2.8V über GPIO), DVDD=ein (1.5V über DAC)
    cam_gpio_DOVDD_AVDD(true);
    cam_gpio_DVDD(true);

    RTC_Trigger(3, cam_control_RTCUpdate);         // (warten in EM1)
    EMU_EnterEM1();

    // Anfangszustand: RESET=0, PWDN=0, XCLK=0, SIO_C=1(pull-up), SIO_D=1(pull-up); DOVDD=aus, DVDD=aus, AVDD=aus
    cam_gpio_Reset(true);
    cam_gpio_PWDN(false);
    cam_xclk_Enable(false);

    RTC_Trigger(3, cam_control_RTCUpdate);         // min 3ms lt. Appnote (warten in EM1)
    EMU_EnterEM1();

    // Reset=false, XCLK=ein
    cam_gpio_Reset(false);
    cam_xclk_Enable(true);

    RTC_Trigger(100, cam_control_RTCUpdate);       // min 100ms lt. Appnote (warten in EM1)
    EMU_EnterEM1();

    // SCCB Soft Reset
    cam_i2c_Write(0x3012, 0x80);

    RTC_Trigger(5, cam_control_RTCUpdate);         // min 5ms lt. Appnote (warten in EM1)
    EMU_EnterEM1();

    // ab hier kann über SCCB (I2C) die Kamerakonfiguration erstellt werden
}


/***************************************************************************//**
 * @brief Camera Off - Kamera Power off
 ******************************************************************************/
void cam_control_CamOFF(void)
{
    // lt. Appnote zuerst XCLK aus, RESET low dann Versorgung aus
    cam_xclk_Enable(false);                        // XCLK aus
    cam_gpio_PWDN(true);
    cam_gpio_Reset(true);                          // RESET true

    RTC_Delay(1);

    // DOVDD=aus, AVDD=aus, DVDD=aus
    cam_gpio_DVDD(false);
    cam_gpio_DOVDD_AVDD(false);

    // ab hier sollte die Kamera keinen Strom mehr verbrauchen
}


/***************************************************************************//**
 * @brief Camera (Re)Start Capture
 ******************************************************************************/
void cam_control_StartCapture(void)
{
     // Anfangszustand Zählvariablen
    countHREF = 0;
    countHREFmax = 0;
    countPCLK = 0;
    countVSYNC = 0;

    // Imagebuffer löschen - wichtig hier falls nur 1 Frame gecaptured wird!!!
    for(int imgX=0; imgX<40; imgX++)               // 40 spalten
        for(int imgY=0; imgY<30; imgY++)           // 30 zeilen
            imageBuffer[imgY][imgX] = 0;

    // VSYNC Interrupt aktivieren und CaptureActiv Flag setzen
    camCaptureActive = true;
    cam_gpio_VSYNCEnable();
}

/***************************************************************************//**
 * @brief Camera GPIO Enable
 ******************************************************************************/
void cam_control_EnableGPIO(bool en)
{
    if (en)
    {
        cam_i2c_Write(0x30B0, 0xFC);      // IO enable D2-D7 (Bit 2..7)
//      cam_i2c_Write(0x30B1, 0x2F);      // IO enable D8-D9 (Bit 0..1), HREF (Bit2), PCLK (Bit3), VSYNC(Bit5)
        cam_i2c_Write(0x30B1, 0x27);      // IO enable D8-D9 (Bit 0..1), HREF (Bit2), VSYNC(Bit5) - kein PCLK!
        cam_i2c_Write(0x30B2, 0x00);      // IO PAD drive capability (0..1x, 1..2x, 3..3x, 4..4x)
    }
    else
    {
        cam_i2c_Write(0x30B0, 0x00);      // IO enable D2-D7 (Bit 2..7)
        cam_i2c_Write(0x30B1, 0x00);      // IO enable D8-D9 (Bit 0..1), HREF (Bit2), PCLK (Bit3), VSYNC(Bit5)
        cam_i2c_Write(0x30B2, 0x00);      // IO PAD drive capability (0..1x, 1..2x, 3..3x, 4..4x)
    }
}
```

```
/*****************************************************************************//**
 * @brief Camera Config Setup
 *****************************************************************************/
void cam_control_ConfigInit(void)
{
    // IO & Clock & Analog Setup
    cam_i2c_Write(0x308c, 0x80);  // (0x00) Timing Control 12 (DIS_MIPI_RW, grp_wr_en)
    cam_i2c_Write(0x308d, 0x0e);  // (0x00) Timing Control 13 (Bit4: MIPI disable)
//  cam_i2c_Write(0x360b, 0x00);  // (0x00) DVP_CTRL0B (testpattern, testmode)

//  cam_i2c_Write(0x300e, 0x34);  // (0x34) PLL Control 1 (Second camera, PLL divider control bits)
//  cam_i2c_Write(0x300f, 0xa6);  // (0xA6) PLL Control 2 (FreqDiv, Bit8Div, Bypass, InDiv)
    cam_i2c_Write(0x3010, 0x81);  // (0x81) PLL Control 3 (DvpDiv, LaneDiv, SensorDiv, ScaleDiv)
    cam_i2c_Write(0x3082, 0x01);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x30f4, 0x01);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x3090, 0x43);  // Array Mirror On/Off + ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x3091, 0xc0);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x30ac, 0x42);  // ??? (Reserved lt. Datenblatt)

    // Block ev. nicht notwendig... (bewirkt ein etwas helleres Bild: die nachfolgenden Parameter sind auf diese Werte abgestimmt)
    cam_i2c_Write(0x30d1, 0x08);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x30a8, 0x55);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x3015, 0x02);  // (0x02) - Auto Control 3 (Dummy Frame, AGC gain ceiling)
    cam_i2c_Write(0x3093, 0x00);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x307e, 0xe5);  // (0xC5) Timing Control 8 (Digital Gain Source from AGC)
    cam_i2c_Write(0x3079, 0x00);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x30aa, 0x42);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x3017, 0x40);  // (0x00) - Auto Control 5 (Drop Frame enable, Manual banding counter)
    cam_i2c_Write(0x30f3, 0x83);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x306a, 0x0c);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x306d, 0x00);  // (0x08) - Black Level Control D (Bit7:0 Reserved lt. Datenblatt)
    cam_i2c_Write(0x336a, 0x3c);  // (0x3F) ISP_CTRL_6A (Bit7:0 Reserved lt. Datenblatt)

    cam_i2c_Write(0x3076, 0x6a);  // (0x72) - Timing Control 0 (Reserved + VSYNC drop option)
    cam_i2c_Write(0x30d9, 0x95);  // ??? (Reserved lt. Datenblatt) Das Register ist aber extrem wichtig für die Funktion der Kamera
    cam_i2c_Write(0x3016, 0x82);  // (0x80) - Auto Control (div AGC Options)
    cam_i2c_Write(0x3601, 0x30);  // (0xB0) DVP Control 01 (div. Digital Video Port Einstellungen,
                                  //  sollte eigentlich 0x30 nach init sein lt. Datenblatt)
    cam_i2c_Write(0x304e, 0x88);  // ??? (Reserved lt. Datenblatt)
    cam_i2c_Write(0x30f1, 0x82);  // ??? (Reserved lt. Datenblatt)
//  cam_i2c_Write(0x3011, 0x02);  // Clock Rate Control (Frequency doubler, PLL bypass, Clock divider)


    // Lens correction (funktioniert nur in Kombination mit "other Functions")
    // Code siehe oben...

    cam_i2c_Write(0x3400, 0x20);      // FORMAT_CTRL0: Format Control; 0x20..Y8 (s/w)

    // Gain  / Exposure
    cam_i2c_Write(0x3013, 0x00);      // ACG/AEC off
    cam_i2c_Write(0x3000, 0x00);      // Gain 0x0000
    cam_i2c_Write(0x3001, 0x00);
    cam_i2c_Write(0x3002, 0x00);      // Exposure 0x0000
    cam_i2c_Write(0x3003, 0x00);
}


/*****************************************************************************//**
 * @brief Config LDR-Image1
 *****************************************************************************/
void cam_control_ConfigImage1(void)
{
    cam_i2c_Write(0x3000, 0x00);      // Gain
    cam_i2c_Write(0x3001, 0x00);
    cam_i2c_Write(0x3002, 0x00);      // Exposure  0x00D0 ... ISO100, F4, 1/60s
    cam_i2c_Write(0x3003, 0xD0);
}

/*****************************************************************************//**
 * @brief Config LDR-Image2
 *****************************************************************************/
void cam_control_ConfigImage2(void)
{
    cam_i2c_Write(0x3000, 0x00);      // Gain
    cam_i2c_Write(0x3001, 0x00);
    cam_i2c_Write(0x3002, 0x00);      // Exposure  0x0028 ... ISO100, F4, 1/250s
    cam_i2c_Write(0x3003, 0x28);
}

/*****************************************************************************//**
 * @brief Config LDR-Image3
 *****************************************************************************/
void cam_control_ConfigImage3(void)
{
    cam_i2c_Write(0x3000, 0x00);      // Gain
    cam_i2c_Write(0x3001, 0x00);
    cam_i2c_Write(0x3002, 0x00);      // Exposure  0x0008 ... ISO100, F4, 1/1000s
    cam_i2c_Write(0x3003, 0x08);
}
```

debug_uart.c

Dieses File implementiert das Protokoll zur Kommunikation mit dem PC via UART-Schnittstelle.

```
void debug_uart_SendData(TxFrameTyp_Enum FrameTyp)
{
```

```
uint8_t txHeader[4];  // Frame Header
int size;                // Größe Datenblock
CK = 0x00;               // Checkbyte Reset

// Adresse und Datenbuffer für Kameraregister
uint16_t     camAddr;
uint8_t      camData[0x100];


switch(FrameTyp)
{
    case txStatus:
        size = 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4;              // Size Datenblock (ohne Header)
        txHeader[0] = 0x0F;
        txHeader[1] = txStatus;
        txHeader[2] = size >> 8;
        txHeader[3] = 0xFF & size;
        debug_uart_CalcCK((char*) txHeader, 4);             // Header (4 Byte)
        USART1_sendBuffer((char*) txHeader, 4);
        debug_uart_CalcCK((char*) &BrightnessAll, 4);       // BrightnessAll (int, 4 Byte)
        USART1_sendBuffer((char*) &BrightnessAll, 4);
        debug_uart_CalcCK((char*) &BrightnessAllAlt, 4);    // BrightnessAllAlt (int, 4 Byte)
        USART1_sendBuffer((char*) &BrightnessAllAlt, 4);
        debug_uart_CalcCK((char*) &BrightnessMask, 4);      // BrightnessMask (int, 4 Byte)
        USART1_sendBuffer((char*) &BrightnessMask, 4);
        debug_uart_CalcCK((char*) &LoopCurrent, 4);         // LoopCurrent (float, 4 Byte)
        USART1_sendBuffer((char*) &LoopCurrent, 4);
        debug_uart_CalcCK((char*) &LoopVoltageMax, 4);      // LoopVoltageMax (float, 4 Byte)
        USART1_sendBuffer((char*) &LoopVoltageMax, 4);
        debug_uart_CalcCK((char*) &VoltagePD1, 4);          // VoltagePD1 (float, 4 Byte)
        USART1_sendBuffer((char*) &VoltagePD1, 4);
        debug_uart_CalcCK((char*) &VoltagePD2, 4);          // VoltagePD2 (float, 4 Byte)
        USART1_sendBuffer((char*) &VoltagePD2, 4);
        debug_uart_CalcCK((char*) &VoltagePD3, 4);          // VoltagePD3 (float, 4 Byte)
        USART1_sendBuffer((char*) &VoltagePD3, 4);
        USART1_sendBuffer((char*) CK, 1);                   // Checkbyte senden
        break;

    case txImage1:
        size = sizeof(image1);
        txHeader[0] = 0x0F;
        txHeader[1] = txImage1;
        txHeader[2] = size >> 8;
        txHeader[3] = 0xFF & size;
        debug_uart_CalcCK((char*) txHeader, 4);
        debug_uart_CalcCK((char*) &image1[0][0], size);
        USART1_sendBuffer((char*) txHeader, 4);
        USART1_sendBuffer((char*) &image1[0][0], size);
        USART1_sendBuffer((char*) CK, 1);
        break;

    case txImage2:
        size = sizeof(image2);
        txHeader[0] = 0x0F;
        txHeader[1] = txImage2;
        txHeader[2] = size >> 8;
        txHeader[3] = 0xFF & size;
        debug_uart_CalcCK((char*) txHeader, 4);
        debug_uart_CalcCK((char*) &image2[0][0], size);
        USART1_sendBuffer((char*) txHeader, 4);
        USART1_sendBuffer((char*) &image2[0][0], size);
        USART1_sendBuffer((char*) CK, 1);
        break;

    case txImage3:
        size = sizeof(image3);
        txHeader[0] = 0x0F;
        txHeader[1] = txImage3;
        txHeader[2] = size >> 8;
        txHeader[3] = 0xFF & size;
        debug_uart_CalcCK((char*) txHeader, 4);
        debug_uart_CalcCK((char*) &image3[0][0], size);
        USART1_sendBuffer((char*) txHeader, 4);
        USART1_sendBuffer((char*) &image3[0][0], size);
        USART1_sendBuffer((char*) CK, 1);
        break;

    case txImageHDR:
        size = sizeof(imageHDR);
        txHeader[0] = 0x0F;
        txHeader[1] = txImageHDR;
        txHeader[2] = size >> 8;
        txHeader[3] = 0xFF & size;
        debug_uart_CalcCK((char*) txHeader, 4);
        debug_uart_CalcCK((char*) &imageHDR[0][0], size);
        USART1_sendBuffer((char*) txHeader, 4);
        USART1_sendBuffer((char*) &imageHDR[0][0], size);
        USART1_sendBuffer((char*) CK, 1);
        break;

    case txMWGewicht:
        size = sizeof(MWGewicht);
        txHeader[0] = 0x0F;
        txHeader[1] = txMWGewicht;
        txHeader[2] = size >> 8;
        txHeader[3] = 0xFF & size;
        debug_uart_CalcCK((char*) txHeader, 4);                 //Prüfbyte Header
        debug_uart_CalcCK((char*) &MWGewicht[0][0], size);      //Prüfbyte Daten (weiterrechnen)
        USART1_sendBuffer((char*) txHeader, 4);
```

```
            USART1_sendBuffer((char*) &MWGewicht[0][0], size);
            USART1_sendBuffer((char*) CK, 1);
            break;

[...] weitere Datenframes

        default:
            break;
    }

}

// XOR Prüfbyte berechnen
void debug_uart_CalcCK(char* dataBuffer, int len)
{
    for (int i = 0; i < len;  i++)
    {
    if (dataBuffer != 0)
    {
            /* XOR Prüfbyte über Buffer rechnen */
            CK ^= *dataBuffer;
            dataBuffer++;
        }
    }
}
```

sensor_hdr.c

Die Routinen zum Berechnen des HDR-Bildes sind in diesem File zusammengefasst.

```
/*****************************************************************************
 * @brief  HDR Image aus Image1..n berechen und nach imageHDR speichern
 *****************************************************************************/
void sensor_hdr_CalcHDRImage(void)
{
    float pixelVal8;        // Pixelfarbe aus 8Bit LDR Bild
    float Gain;             // aktuelle Verstärkung auf Lux
    float Shift;            // Shift der Pixelwerte (in Lux)
    float Gewicht;          // Gewichtung des Pixelwertes
    float pixelValLux;      // 16 Bit HDR Pixelwert
    uint16_t pixelVal16;      // 16 Bit HDR Pixelwert
    uint16_t pixelVal16Res;   // 16 Bit HDR Pixelwert (Gesamt)


    // HDR Bild bauen
    for(int imgX=0; imgX<40; imgX++)       // 40 spalten
        for(int imgY=0; imgY<30; imgY++)  // 30 zeilen
        {
            pixelVal16Res = 0;

            /* Image 1 */
           pixelVal8 = (float)image1[imgY][imgX];
           Gain = 3;
           Shift = 0;
           pixelValLux = pixelVal8 * Gain - Shift;
           Gewicht = sensor_hdr_GetGewicht((int)pixelValLux, 0, 0, 100, 1000);
                                                    // Pixelgewichtung ermitteln (Parameter: Start0, Start100, Ende100, Ende0)
           pixelVal16 = (uint16_t)(pixelValLux * Gewicht); // Gain und Gewicht berücksichtigen
           pixelVal16Res += pixelVal16;

            /* Image 2 */
           pixelVal8 = (float)image2[imgY][imgX];
           Gain = 6;
           Shift = 0;
           pixelValLux = pixelVal8 * Gain - Shift;
           Gewicht = sensor_hdr_GetGewicht((int)pixelValLux, 100, 1000, 1200, 3000);
                                                    // Pixelgewichtung ermitteln (Parameter: Start0, Start100, Ende100, Ende0)
           pixelVal16 = (uint16_t)(pixelValLux * Gewicht); // Gain und Gewicht berücksichtigen
           pixelVal16Res += pixelVal16;

            /* Image 3 */
           pixelVal8 = (float)image3[imgY][imgX];
           Gain = 20;
           Shift = 0;
           pixelValLux = pixelVal8 * Gain - Shift;
           Gewicht = sensor_hdr_GetGewicht((int)pixelValLux, 1200, 3000, 10000, 10000);
                                                    // Pixelgewichtung ermitteln (Parameter: Start0, Start100, Ende100, Ende0)
           pixelVal16 = (uint16_t)(pixelValLux * Gewicht); // Gain und Gewicht berücksichtigen
           pixelVal16Res += pixelVal16;

            int pixValAlt = imageHDR[imgY][imgX];          // alter Pixelwert für Gradient

           // Nullwerte in HDR-Bild vermeiden:
            // 0 Lux auf 1 Lux setzen ->  vermeidet Probleme bei Berechnen des Änderungsfaktors von einem Bild zum anderen
           if (pixelVal16Res < 1)
               imageHDR[imgY][imgX] = 1;
           else
               imageHDR[imgY][imgX] = pixelVal16Res;

            /* Pixelgradient merken (braucht weniger speicher als das HDR Image (16Bit) zu speichern.. (für Gewichtungsalgorithmus) */
            int pixGradient = (imageHDR[imgY][imgX] - pixValAlt) + 127; // negative Werte auf positive abbilden
            if (pixGradient >= 0xFF)
                pixGradient = 0xFF;       // Maximalwert: 255, 8Bit
            if (pixGradient <= 0x00)
                pixGradient = 0x00;       // Minimalwert: 0, keine negativen Werte (8Bit)
            imageHDRPixGradient[imgY][imgX] = (uint8_t)pixGradient;
```

```
                                        // Gradient in 8-Bit Matrix merken (1.2kB statt 2.4kB bei 16-Bit Matrix)
        }

}

float sensor_hdr_GetGewicht(int Value, int Start0, int Start100, int Ende100, int Ende0)
{
    float Gewicht = 0.0f;

    if ((Start100 - Start0) == 0)       // Div./0 abfangen falls Startrampenwertegleich sind
        Start100 = Start0 + 1;
    if ((Ende0 - Ende100) == 0)         // Div./0 abfangen falls Endrampenwerte gleich sind
        Ende100 = Ende0 - 1;

    if (Value >= Start0 && Value <= Start100)             // Startrampe
        Gewicht = (float)(Value - Start0) / (float)(Start100 - Start0);
    else if (Value > Start100 && Value < Ende100)         // Max-Messbereich
        Gewicht = 1.0f;
    else if (Value >= Ende100 && Value <= Ende0)          // Endrampe
        Gewicht = (float)(Ende0 - Value) / (float)(Ende0 - Ende100);

    return Gewicht;
}
```

sensor_currentloop.c

Die Ansteuerung der Stromschnittstelle via DAC ist eine sehr kurze Routine wie nachfolgender Auszug zeigt.

```
/*************************************************************************//**
 * @brief  VSET setzen
 * Sollwert für Schleifenregelung: gepuffert über Speicher-Cap
 *************************************************************************/
void sensor_currentloop_VSET(void)
{
    // Schleifenstrom über DAC ausgeben
    // Nichtlinearität ggf. berücksichtigen (derzeit nicht implementiert)

    float VRef = 2.5;                 // 2.5V interne Referenz
    float VSet;
    uint32_t DAC_Value;

    if (LoopCurrent > 20.0)           // Strombegrenzung auf max 20mA
        LoopCurrent = 20.0;

    VSet = LoopCurrent / 10.0;        // VSet aus LoopCurrent berechnen

    // Sollwert ausgeben: wird über Cap zwischengespeichert...
    DAC_Value = (uint32_t)((VSet * 4096) / VRef);       // Ausgabe: 2V..20mA, 0.4V..4mA
    sensor_dac_WriteData(DAC0, DAC_Value, 0);           // Signal für TI Current Loop Baustein (DAC Channel 0, PB11)

    RTC_Trigger(300, sensor_currentloop_RTCUpdate);     // Warten bis sich Wert am Speicherkondensator stabilisiert hat
                                                        // 100ms sollten für 10uF Cap ausreichen
    EMU_EnterEM1();                                     // hier nur EM1 Sleepmode! EM2 würde den DAC deaktivieren!

}
```

sensor_calc.c

Die sensor_calc.c implementiert den Helligkeitsgewichtungs-Algorithmus. Außerdem sind Funktionen zur Errechnung des Helligkeits-Messwertes enthalten.

```
/*************************************************************************
 * @brief
 * Gesamthelligkeit des Bildes berechnen
 *************************************************************************/
void sensor_calc_BrightnessAll(void)
{
    int Brightness = 0;

    // alten Helligkeitswert merken (für Gewichtungsalgorithmus)
    BrightnessAllAlt = BrightnessAll;

    // HDR Bild analysieren
    for (int y=0; y<SizeY; y++)
        for (int x=0; x<SizeX; x++)
            Brightness += (int)imageHDR[y][x];

    BrightnessAll = Brightness / (SizeX*SizeY);
}

/*************************************************************************
 * @brief
 * Bildhelligkeit aufgrund der Gewichtungsmatrix berechnen
 *************************************************************************/
```

```
void sensor_calc_BrightnessMask(void)
{
    float resc = 0;

    for (int y=0; y<SizeY; y++)
        for (int x=0; x<SizeX; x++)
        {
            // jeden Pixelwert mit seinem Gewicht multiplizieren; das Gesamtgewicht der Gewichtungsmatrix ist immer 1!
            resc += imageHDR[y][x] * MWGewicht[y][x];
        }

    BrightnessMask = (int)resc;
}

/*****************************************************************************
 * @brief
 * Gewichtungsmatrix initialisieren
 *****************************************************************************/
void sensor_calc_InitMWGewicht(void)
{
    float Divisor = SizeX * SizeY;

    for (int y=0; y<SizeY; y++)
        for (int x=0; x<SizeX; x++)
        {
            MWGewicht[y][x] = 1.0 / Divisor;
        }
}


/*****************************************************************************
 * @brief
 * Gewichtungsmatrix normalisieren
 *****************************************************************************/
void sensor_calc_Normalize(void)
{
    float GesamtGew = 0;

    // aktuelles Gesamtgewicht der Matrix ermitteln
    for (int y = 0; y < SizeY; y++)
        for (int x = 0; x < SizeX; x++)
        {
            GesamtGew += MWGewicht[y][x];
        }

    // Matrix normalisieren (Gesamtgewicht = 1)
    for (int y = 0; y < SizeY; y++)
        for (int x = 0; x < SizeX; x++)
        {
            MWGewicht[y][x] /= GesamtGew;
        }
}


/*****************************************************************************
 * @brief
 * Gewichtungsmatrix speichern in Flash
 *****************************************************************************/
void sensor_calc_SaveMWGewicht(void)
{
    // 2do: MWGewicht speichern im Flash
}


/*****************************************************************************
 * @brief
 * Gewichtungsmatrix aus Flash laden
 *****************************************************************************/
void sensor_calc_LoadMWGewicht(void)
{
    // 2do: MWGewicht aus Flash laden
}


/*****************************************************************************
 * @brief
 * Pixelgewichtung in Gewichtungsmatrix aktualisieren
 *****************************************************************************/
void sensor_calc_UpdatePixelGewicht(void)
{
    // Update Pixelgewichtung

    static float Gradient;
    static float Gewicht;
    static float PixVal;


    if (BrightnessAll > 100 && BrightnessAllAlt > 100)      // Grundhelligkeit der Bilder: > 100 Lux
    {
        // 2do: ev. zugeschaltete Kunstlichtquellen aus Messwert ausschließen

        // Globale Richtung der Helligkeitsänderung bestimmen
        Gradient = BrightnessAll - BrightnessAllAlt;

        for (int y = 0; y < SizeY; y++)
            for (int x = 0; x < SizeX; x++)
            {
                Gewicht = MWGewicht[y][x];                          // aktuelles Pixelgewicht
                PixVal = imageHDR[y][x];                            // Wert aus HDR Bild
```

```
                Gradient = (float)imageHDRPixGradient[y][x] - 127;   // Pixelgradient (aus 8-Bit Matrix wegen Speicherverbrauch)

                // Gute Lichtaanteile suchen und bewerten (geringe Änderung in die richtige Richtung)
                // Schlechte Lichtanteile abwerten (Änderungen in die falsche Richtung)

                // **** Geplante Version: Vergleich Gesamtegradient mit Pixelgradient ****
                //if (PixGradient > 0.1f * Gradient && PixGradient < 10f * Gradient)
                //    Gewicht = Gewicht * 1.1f;     // schlechtes Pixel: Gewicht verringern
                //else
                //    Gewicht = Gewicht * 0.91f;    // gutes Pixel: Gewicht erhöhen

                // implementierte Version: Vergleich des Gradienten mit dem aktuellem Pixelwert
                if (PixVal > 0.1 * Gradient && PixVal < 10.0 * Gradient)
                    Gewicht = Gewicht * 0.91;       // schlechtes Pixel: Gewicht verringern
                else
                    Gewicht = Gewicht * 1.1;        // gutes Pixel: Gewicht erhöhen

                if (Gewicht > 0.003)
                    Gewicht = 0.003;                // max Pixelgewicht: 0.3% = 0.003

                if (Gewicht < 0.00001)
                    Gewicht = 0.00001;              // min Pixelgewicht: 0.001% = 0.00001

                 // neues Pixelgewicht speichern
                 MWGewicht[y][x] = Gewicht;
            }
    }

    // Gewichtungsmatrix normalisieren
    sensor_calc_Normalize();

    // Gewichtungsmatrix in Flash speichern
    sensor_calc_SaveMWGewicht();
}
```

sensor_adc.c

Routinen zur Ansteuerung der ADCs. Hauptsächlich wird damit die Spannung am Speicherkondensator gemessen.

```
/**************************************************************************//**
* @brief
*   Warten bis genug Energie im Cap gespeichert ist (mit Status-LED)
*   aktuellen Messwert periodisch setzen
*****************************************************************************/
void sensor_adc_Wait4Energy(float VCap, bool recharge)
{

    static   float VSense;
    static   float VSenseOld;
    static   float VDelta;
    static   int tick = 0;

    do
    {
        RTC_Trigger(4, sensor_adc_RTCUpdate);          //  Messwert Ausgabe über DAC (EM1) auf Cap (refresh) (Default: 20)
        EMU_EnterEM1();

        RTC_Trigger(20, sensor_adc_RTCUpdate);         //  Warten in EM2, Stützcap überbrückt die Zeit
        EMU_EnterEM2(true);

        sensor_io_TestButton();                        // Button abfragen

        if (tick >= 20)
        {
            sensor_io_RedBlink();
            tick = 0;
        }
        tick++;

        VSense = sensor_adc_GetVSense();

    } while (VSense < VCap);                           // Abbrechen wenn VCap Sollwert erreicht hat



    if (recharge == true)                              // Nachladen bis max. Spannung?
    {
        // Cap nachladen bis sich Spannung nicht mehr verändert: Damit arbeitet der Sensor mit der höchstmöglichen Spannung bzw.
Effizienz
        do
        {
            VSenseOld = VSense;

            RTC_Trigger(4, sensor_adc_RTCUpdate);      // Messwert Ausgabe über DAC (EM1) auf Cap (refresh)
            EMU_EnterEM1();

            RTC_Trigger(20, sensor_adc_RTCUpdate);     // Warten in EM2, Stützcap überbrückt die Zeit
            EMU_EnterEM2(true);

            sensor_io_TestButton();                    // Button abfragen

            // Roter Blink während dem Laden
            if (tick >= 20)
```

```
            {
                sensor_io_RedBlink();
                tick = 0;
            }
            tick++;

            // Spannung am Kondensator ermitteln
            VSense = sensor_adc_GetVSense();

            // VDelta berechnen
            VDelta = VSense - VSenseOld;
            if (VDelta < 0)
                VDelta = -VDelta;
        } while (VDelta > 0.01);                        // Abbrechen wenn Cap nicht weiter geladen wird


        // Maximalwert der Schleifenspannung in diesem Zyklus merken (für Debug...)
        LoopVoltageMax = sensor_adc_GetVSense() + (LoopCurrent/1000.0)*10.0 + 0.6;
        // Schleifenspannung = Spannung Ladekondensator + Spannungsabfall 10R Widerstand + Sannungsabfall Diodenbrücke
    }
    else
    {
        // Kurz nachladen damit geforderte Spannung sicher vorhanden ist
        RTC_Trigger(2, sensor_adc_RTCUpdate);          //  Messwert Ausgabe über DAC (EM1) auf Cap (refresh)
        EMU_EnterEM1();

        RTC_Trigger(30, sensor_adc_RTCUpdate);         //  Warten in EM2, Stützcap überbrückt die Zeit
        EMU_EnterEM2(true);
    }


    // Grüner Blink wenn Sollspannung erreicht
    sensor_io_GreenBlink();

}

/**************************************************************************//**
* @brief
*   VSense Sample von ADC lesen
*   Aufgrund von Rauschen im Messsignal liefert der ADC nicht immer den korrekten
*   Messwert, daher mehrere Samples nehmen um den Messwert zu glätten.
*****************************************************************************/
float sensor_adc_GetVSense(void)
{
    static   uint32_t     sample;
    static float          VREF = 2.5;                   // VREF = 2.5V (interne Referenz)
    static float          VSense[3];                    // 3 Samples
    static float          VSenseRes;


    VSenseRes = 100;                                    // Maxwert initialisieren...

    // 3 Samples nehmen
    for (int i=0; i<3; i++)
    {
        sample = sensor_adc_GetSample(adcSingleInpCh0);
        VSense[i] = (float)sample * (VREF/4096.0) * (660.0/100.0) * 1.065;       // Spannungsteiler für VSense: 560k/100k; 1.xxx ...
Korrekturfaktor

        if (VSenseRes > VSense[i])                      // kleinsten Messwert suchen
            VSenseRes = VSense[i];
    }

    // kleinsten Wert zurückgeben
    return VSenseRes;
}
```

sensor_io.c

Dieses Modul sammelt alle Routine die IO-Funktionalität beinhalten und nicht schon an anderer Stelle implementiert ist. Das ist z.B. die LED-Ansteuerung oder Button-Abfrage.

```
void sensor_io_RTCUpdate(void)
{
    // Code nach Sleepmode-Reaktivierung
    int but;
    but = GPIO_PinInGet(gpioPortF, 9);

    // Button lange gedrückt?
    if (but == 0)
    {
        ButtonLongPress += 1;

        if (ButtonLongPress >= 300)       // 20ms * 300 = 6s (die 20ms kommen aus    sensor_io_TestButton()!)
        {
            ButtonLongPress = 0;          // Longpress-Mode zurücksetzen
            sensor_io_ButtonLongPress();  // Debug-Routine
        }
    }
    else
        ButtonLongPress = 0;
}

//void sensor_io_RTCUpdateButtonLongPress(void)
```

```
//{
//}

/***************************************************************************
 * @brief
 * Grüne LED ein (High-Power)
 ***************************************************************************/
void sensor_io_GreenON(void)
{
    // Achtung: hier muss genug Schleifenstrom zur Verfügung stehen um die LED zu treiben (extern einstellen!)

    GPIO_DriveModeSet(gpioPortF, gpioDriveModeHigh);            // Drivemode: High

    GPIO_PinOutClear(gpioPortF, 7);
    GPIO_PinOutSet(gpioPortF, 8);
}

/***************************************************************************
 * @brief
 * Grüne LED aus
 ***************************************************************************/
void sensor_io_GreenOFF(void)
{

    GPIO_PinOutSet(gpioPortF, 7);
    GPIO_PinOutSet(gpioPortF, 8);

    GPIO_DriveModeSet(gpioPortF, gpioDriveModeLowest);          // Drivemode: Lowest

}

/***************************************************************************
 * @brief
 * Grüne LED Blink (Low-Power)
 ***************************************************************************/
void sensor_io_GreenBlink(void)
{
    GPIO_DriveModeSet(gpioPortF, gpioDriveModeLowest);          // Drivemode: Lowest

    GPIO_PinOutClear(gpioPortF, 7);
    GPIO_PinOutSet(gpioPortF, 8);

    RTC_Trigger(30, sensor_io_RTCUpdate);
    EMU_EnterEM2(true);

    GPIO_PinOutSet(gpioPortF, 7);
    GPIO_PinOutSet(gpioPortF, 8);
}

/***************************************************************************
 * @brief
 * Rote LED Blink (Low-Power)
 ***************************************************************************/
void sensor_io_RedBlink(void)
{
    GPIO_DriveModeSet(gpioPortF, gpioDriveModeLowest);          // Drivemode: Lowest

    GPIO_PinOutClear(gpioPortF, 8);
    GPIO_PinOutSet(gpioPortF, 7);

    RTC_Trigger(30, sensor_io_RTCUpdate);
    EMU_EnterEM2(true);

    GPIO_PinOutSet(gpioPortF, 8);
    GPIO_PinOutSet(gpioPortF, 7);
}


/***************************************************************************
 * @brief
 * Button testen und wenn gedrückt Aktion ausführen
 ***************************************************************************/
void sensor_io_TestButton(void)
{
    int but;
    float LoopCurrentOldVal;

    ButtonLongPress = 0;

    /* Button abfragen */
    but = GPIO_PinInGet(gpioPortF, 9);

    if(but == 0)
    {
        LoopCurrentOldVal = LoopCurrent;                // aktuellen Schleifenstrom merken

        // 20mA Schleifenstrom für LED und Messpunkt
        LoopCurrent = 20.0;
        sensor_currentloop_VSET();

        sensor_io_GreenON();

        while (but == 0)
        {
            but = GPIO_PinInGet(gpioPortF, 9);

            RTC_Trigger(20, sensor_io_RTCUpdate);       // Warten in EM1 (Messwert ausgeben, Energie ist hier genug vorhanden)
            EMU_EnterEM1();
        }
```

```
        sensor_io_GreenOFF();

        // Schleifenstrom zurücksetzen
        LoopCurrent = LoopCurrentOldVal;
        sensor_currentloop_VSET();
    }
}

/***************************************************************************
 * @brief
 * Button LongPress: Debug Routine ausführen
 ***************************************************************************/
void sensor_io_ButtonLongPress(void)
{
    // **** ACHTUNG: In dieser Routine dürfen keine RTC-Funktionen aufgerufen werden!
    //               Auch nicht in Unterfunktionen (Prüfen!)
    //               Grund: Diese Routine wird aus sensor_io_RTCUpdate() aufgerufen.


    float TestCurrent;
    uint32_t DAC_Value;
    long wait = 0;

    // Debug: Schleifenstromrampe
    for(int i=0; i<9; i++)
    {
        sensor_io_GreenOFF();
        // ev. hier Camera-Off; Normalerweise ist die Kamera hier aber immer aus!

        TestCurrent = 4.0 + (float)i * 2.0;

        // Messwert ausgeben
        DAC_Value = (uint32_t)(((TestCurrent/10) * 4096) / 2.5);    // Ausgabe: 2V..20mA, 0.4V..4mA
        sensor_dac_WriteData(DAC0, DAC_Value, 0);                   // Signal für TI Current Loop Baustein (DAC Channel 0, PB11)

        // hier Ausgabe der Daten über UART

        for(int j=0; j<10; j++)
        {
            if(i % 2 == 0)
            {
                // rote LED
                GPIO_DriveModeSet(gpioPortF, gpioDriveModeLowest);  // Drivemode: Lowest
                GPIO_PinOutClear(gpioPortF, 8);
                GPIO_PinOutSet(gpioPortF, 7);
            }
            else
            {
                // grüne LED
                GPIO_DriveModeSet(gpioPortF, gpioDriveModeLow);     // Drivemode: Low
                                                                    // (grüne LED braucht etwas mehr Strom bei gleicher Helligkeit)
                GPIO_PinOutClear(gpioPortF, 7);
                GPIO_PinOutSet(gpioPortF, 8);
            }

            // Warteschleife (RTC hier nicht verwenden!)
            for (wait=0; wait<1000000; wait++);

            // LEDs aus
            GPIO_PinOutSet(gpioPortF, 8);
            GPIO_PinOutSet(gpioPortF, 7);

            // Warteschleife (RTC hier nicht verwenden!)
            for (wait=0; wait<1000000; wait++);
        }
    }
}
```

sensor_dac.c

Das letzte File der Sensor-Firmware initialisiert die DACs und gibt die gewünschte Spannung am eingestellten Port aus. Der Code ist identisch zu den EFM32 Appnote-Beispielen.

6.2 Lichtsensor-Testprogramm

Für die Entwicklung des Helligkeitsgewichtungs-Algorithmus und zur Kommunikation mit dem Sensor-Prototypen ist im Laufe des Projektes ein Lichtsensor-Testprogramm entstanden. Mit diesem Programm können Daten visualisiert und Parameter ausprobiert werden. Der Status des Programms kann als Prototyp bezeichnet werden. Alle Funktionen sind nur so weit implemen-

tiert wie zur Entwicklung des Lichtsensors notwendig war. Für eine allfällige allgemeinere Weiterverwendung des Programms wären einige Anpassungen, wie z.B. bessere Fehlerabsicherungen, bessere Programmstruktur etc. notwendig.

Programmiert ist die Software in der Sprache C# mit Visual Studio Express 2010.

6.2.1 Funktionsübersicht

Nach dem Start präsentiert sich das Lichtsensor-Testprogramm mit dem nachstehenden Hauptbildschirm. Von hier aus sind sämtliche Funktionen erreichbar.

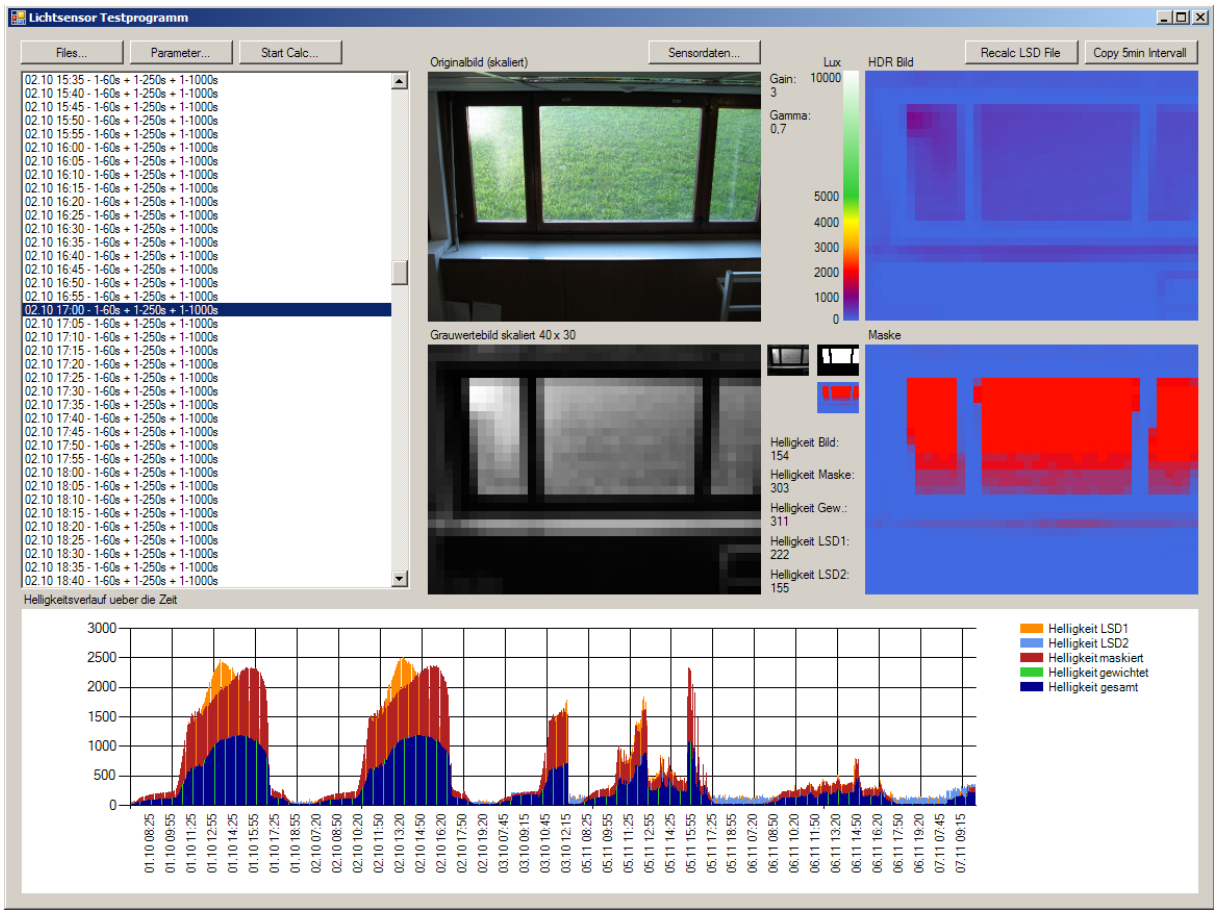

Abbildung 79: Lichtsensor-Testprogramm Hauptbildschirm

In Abbildung 79 linke Seite, ist eine Auflistung von Belichtungsreihen zu sehen. Das sind die Messdaten der Kamera. Ein Eintrag umfasse alle Messdaten zum gewählten Zeitpunkt (Belichtungsreihe, skalierte Bilder, HDR-Bild, Gewichtungsmatrix und die Messdaten externer LSD-Lichtsensoren).

In der Mitte oben ist das erste Bild der Belichtungsreihe in (skalierter) Originalgröße abgebildet. Darunter das daraus berechnete, gammakorregierte Grauwertebild mit 40 x 30 Pixel Auflösung. Rechts oben ist das aktuelle HDR-Bild und darunter die aktuelle Gewichtungsmatrix zu sehen. Das Diagramm in der unteren Bildschirmhälfte enthält die gemessenen errechneten Helligkeits-Messdaten. Ein Doppelklick auf einen der Listeneinträge öffnet die Fenster „Form HDR“ und „Form Gewichtung“ welche weiter Details zum aktuellen Messpunkt preisgeben.

Abbildung 80: Form HDR

Form HDR zeigt alle drei Aufnahmen der Belichtungsreihe im Original und SW-Bild. Rechts in der unteren Reihe noch einmal das resultierende HDR-Bild in Falschfarbdarstellung zu sehen. Ein weiteres Feature, das hier nicht abgebildet werden kann, ist das Aufscheinen des Lux-Messwertes an der betreffenden Bildstelle, wenn der Cursor mit der Maus über das HDR-Bild bewegt wird. Dadurch können Messungen an bestimmten Stellen vorgenommen werden.

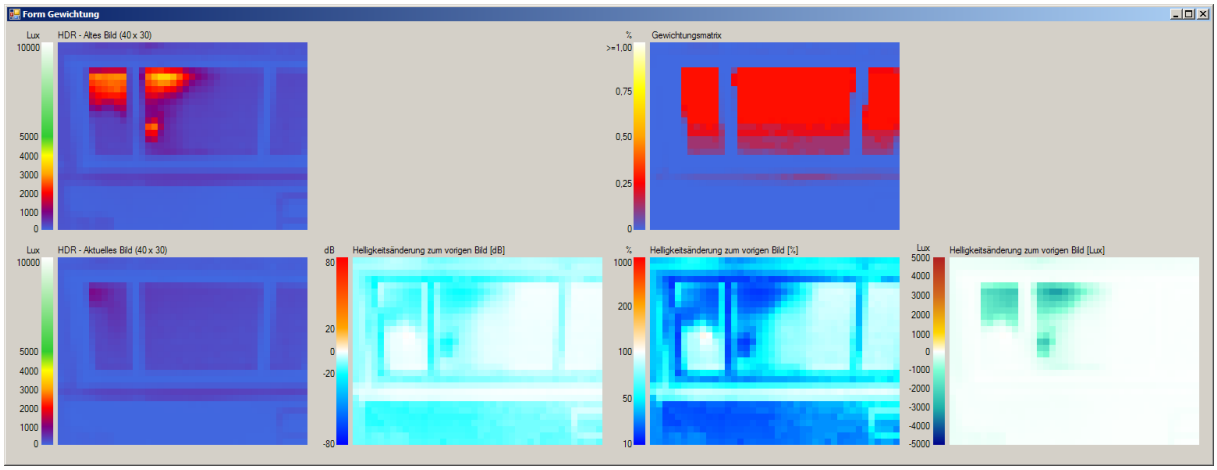

Abbildung 81: Form Gewichtung

Die Form Gewichtung zeigt alle Daten die zur Errechnung der Gewichtungsmatrix führen. Die beiden Bilder auf der linken Seite zeigen das aktuelle und vorhergehende HDR-Bild. Die Veränderungen der Messwerte zwischen den Bildern zeigen die verbleibenden Plätze in der unteren Bildreihe. Dabei kommen drei verschiedene Skalen zum Einsatz: dB, Prozent und Absolutwert. Mit diesen Darstellungen wurde die geeignetste Skala zur Berechnung der Gewichtungsmatrix ermittelt. Das rechte Bild in der oberen Reihe zeigt die aktuelle Gewichtungsmatrix.

Auch in dieser Form können alle Messwerte via Mauscursor an jeder beliebigen Stelle angezeigt werden.

Ein Doppelklick auf das Diagramm im Hauptbildschirm (Abbildung 79) führt zu einer vergrößerten Darstellung der Helligkeitskurven wie nachstehenden Abbildung zeigt:

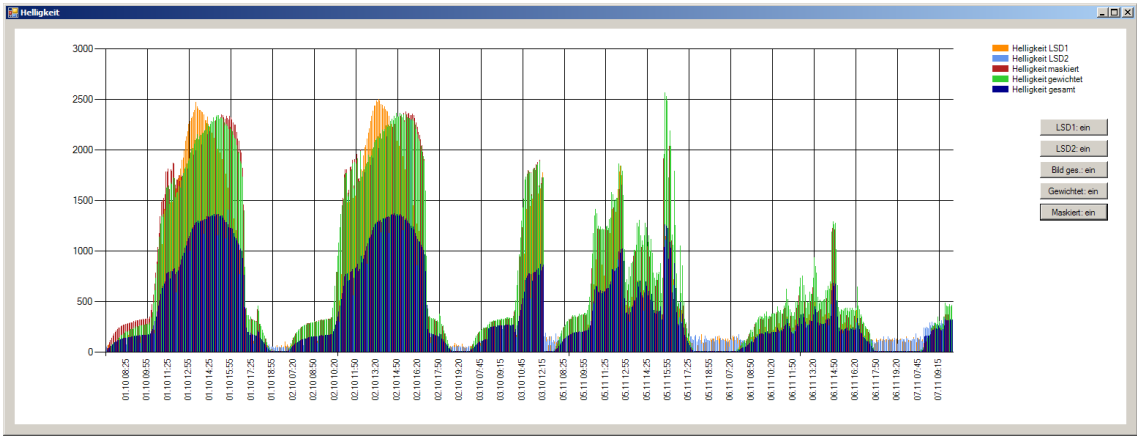

Abbildung 82: Form Helligkeit - Darstellung der Helligkeitskurven

Jeder Messwert, egal ob errechnet oder über einen externen LSD-Lichtsensor gemessen, kann im Diagramm individuell ein- bzw. ausgeblendet werden. Dadurch können beliebige Werte in Relation zueinander gesetzt werden.

Bevor überhaupt ein Messwert angezeigt werden kann, müssen die Rohdaten wie die LDR-Bilder und das Messwertfile der externen LSD-Sensoren, sowie die zu verwendenden Parameter dem Programm bekanntgegeben werden. Die beiden Buttons „Files...“ und „Parameter...“ im Hauptbildschirm (Abbildung 79) führen zu Dialogen über welche das möglich ist. Abbildung 83 und 84 zeigt die Fenster. In den Parametereinstellungen findet sich auch die Möglichkeit ein „harte“ Maske auszuwählen. Diese dient dazu ein Referenzsignal zu erzeugen.

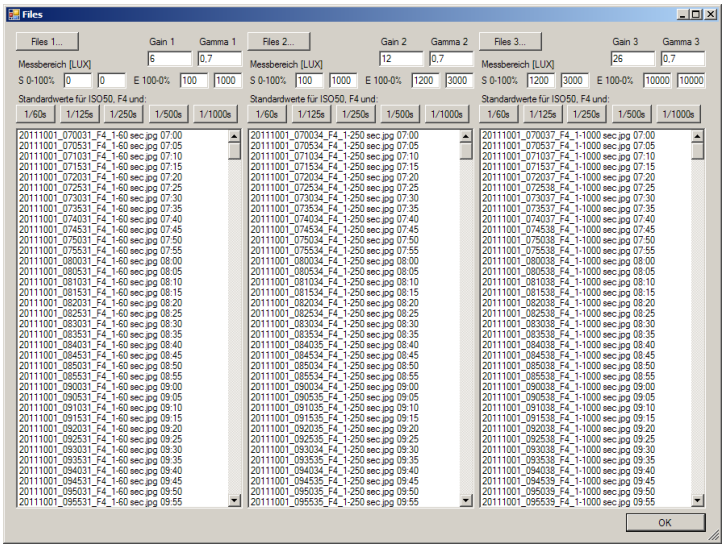

Abbildung 83: Laden der Belichtungsreihen und HDR-Bild-Parameter

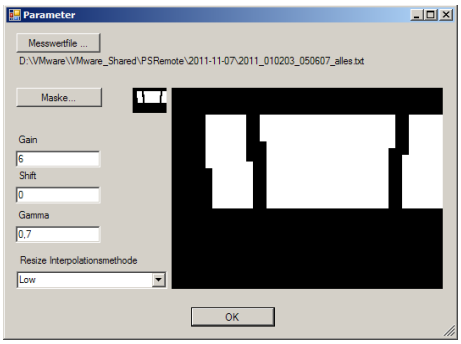

Abbildung 84: Parametereinstellungen

Via Button „Sensordaten“ (Abbildung 79) gelangt man zum Fenster für die Darstellung der vom LSD-Lichtsensor NG übermittelten Daten. Das Fenster aktualisiert seinen Inhalt laufend, immer wenn neue Daten eingetroffen sind. So kann das Verhalten des Sensors in Echtzeit beobachtet werden. Für eine spätere Datenanalyse können die Sensordaten auch aufgezeichnet werden.

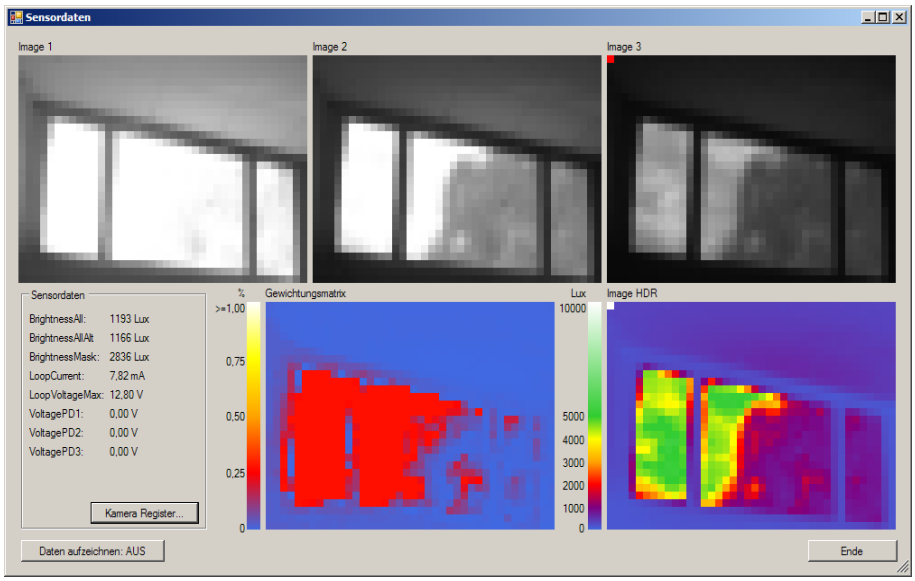

Abbildung 85: Sensordaten - Liveansicht

Für Debuggingzwecke besteht die Möglichkeit, die aktuellen Registerwerte des Kameramoduls anzeigen zu lassen: Button „Kamera Register...“.

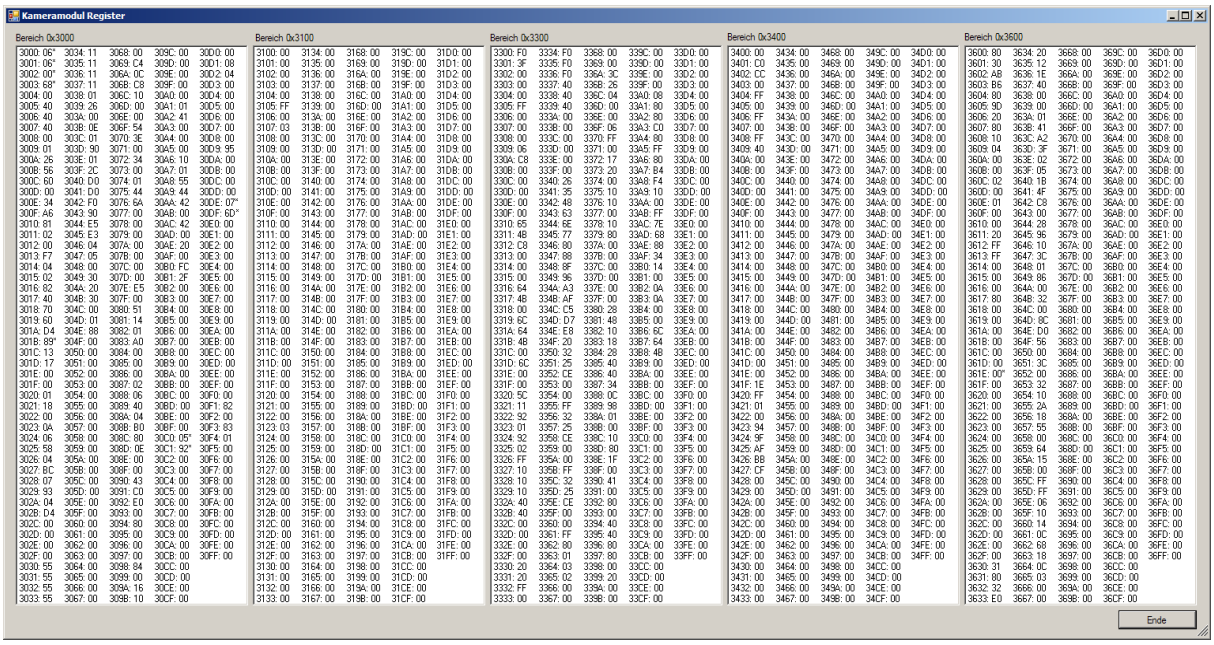

Abbildung 86: Kameramodul - Registerwerte

Literaturverzeichnis

[1] Wikipedia: *Licht*, URL: http://de.wikipedia.org/wiki/Licht (2012)

[2] Wikipedia: *V-Lambda-Kurve*, URL: http://de.wikipedia.org/wiki/V-Lambda-Kurve (2012)

[3] Wikipedia: *Helligkeit*, URL: http://de.wikipedia.org/wiki/Helligkeit (2012)

[4] Wikipedia: *Weber-Fechner-Gesetz*, URL: http://de.wikipedia.org/wiki/Weber-Fechner-Gesetz (2012)

[5] Wikipedia: *Lichtstärke (Photometrie)*, URL: http://de.wikipedia.org/wiki/Lichtstärke_(Photometrie) (2012)

[6] Internationale Beleuchtungskommission CIE, URL: http://www.cie.co.at/ (2012)

[7] Internationale Beleuchtungskommission CIE, *The Basis of Physical Photometry*, CIE 018.2-1983 2nd ed. (reprinted 1996), ISBN 9789290340188

[8] Bass M., Enoch J. M., Lakshminarayanan V.: *Handbook of Optics, Volume III Vision and Vision Optics, OSA* Optical Technology Division, ISBN 9780071636025

[9] Wikipedia: *Lichtstrom*, URL: http://de.wikipedia.org/wiki/Lichtstrom (2012)

[10] Wikipedia: *Lumen (Einheit)*, URL: http://de.wikipedia.org/wiki/Lumen_(Einheit) (2012)

[11] Wikipedia: *Beleuchtungsstärke*, URL: http://de.wikipedia.org/wiki/Beleuchtungsstärke (2012)

[12] Tietze U., Schenk Ch.: *Halbleiter-Schaltungstechnik*, 12. Auflage, Springer 2002, ISBN 3540428946

[13] Hartl H., Krasser E., Pribyl W., Söser P., Winkler G.: *Elektronische Schaltungstechnik, Pearson Studium 2008, ISBN 9783827373212*

[14] Vishay Telefunken: *Silicon PN Photodiode BPW21R* (Datenblatt)

[15] Energy Micro: *EFM32G880 DATASHEET* (Datenblatt), URL: http://www.energymicro.com/products/efm32g880f128-efm32g880f64-efm32g880f32 (2012)

[16] Wikipedia: *CCD-Sensor*, URL: http://de.wikipedia.org/wiki/CCD-Sensor (2012)

[17] Wikipedia: *Photoelektrischer_Effekt / Innerer photoelektrischer Effekt*, URL: http://de.wikipedia.org/wiki/Photoelektrischer_Effekt#Innerer_photoelektrischer_Effekt (2012)

[18] Müller R: *Grundlagen der Halbleiter-Elektronik*, 7. Auflage, Springer 1995, ISBN 3540589120

[19] Wikipedia: *Potentialtopf*, URL: http://de.wikipedia.org/wiki/Potentialtopf (2012)

[20] Erhardt A.: *Einführung in die Digitale Bildverarbeitung*, Vieweg+Teubner 2008, ISBN 351900478X

[21] Zeitschrift Elektor: *Digitale Fotografie, Elektronik ersetzt Film*, Ausgabe 10/98 S54

[22] Wikipedia: *Smear*, URL: http://de.wikipedia.org/wiki/Smear (2012)

[23] Wikipedia: *Active Pixel Sensor*, URL: http://de.wikipedia.org/wiki/Active_Pixel_Sensor (2012)

[24] Foveon: *Direct Image Sensors* , URL: http://www.foveon.com (2012)

[25] Digital Photographie Review: *Kodak High Sensitivity Image Sensor Tech*, URL: http://www.dpreview.com/news/2007/6/14/kodakhighsens (2012)

[26] Gunturk B. K., Glotzbach J., Altunbasak Y., Schafer R. W., Mersereau R. M.: *Demosaicking: Color Filter Array Interpolation in Single-Chip Digital Camereas*, Draft for the IEEE SPM Special Issue on Color Image Processing

[27] Wikipedia: *Additive Farbmischung*, URL: http://de.wikipedia.org/wiki/Additive_Farbmischung (2012)

[28] Wikipedia: *YUV-Farbmodell*, URL: http://de.wikipedia.org/wiki/YUV-Farbmodell (2012)

[29] Silonex Inc.: *NORPS-12 Plastic Packaged CdS Photocell* (Datenblatt)

[30] Wikipedia: *Dynamikumfang*, URL: http://de.wikipedia.org/wiki/Dynamikumfang (2012)

[31] Striewisch T.: *Der Fotolehrgang im Internet*, URL: http://www.fotolehrgang.de (2012)

[32] Fode A.: *Robuste Generierung von High Dynamic Range Bildern* (Diplomarbeit), Institut für Computervisualistik, Universität Koblenz Landau, 2004

[33] Wikipedia: *Blendenstufe*, URL: http://de.wikipedia.org/wiki/Blendenstufe (2012)

[34] NXP: *UM10204 I2C-bus specification and user manual*, Feb. 2012, URL: http://www.nxp.com/acrobat/usermanuals/UM10204_3.pdf (2012)

[35] OmniVision: *OV2655 1/5" color CMOS UXGA (2 megapixel) image sensor* (Datenblatt, nicht öffentlich zugänglich!), Version 2.0, 2009

[36] OmniVision: *OmniVision Serial Camera Control Bus (SCCB)* (Functional Specification), Version 2.2, 2007, URL: http://www.ovt.com

[37] Cine4Home: *Die Varianten des YUV Komponenten-Signals Chroma Upsampling und die Probleme*, URL: http://www.cine4home.de/knowhow/ChromaUpsampling/ChromaUpsampling.htm (2012)

[38] Wikipedia: *Gammakorrektur*, URL: http://de.wikipedia.org/wiki/Gammakorrektur (2012)

[39] Wikipedia: *Blendenreihe (Optik)*, URL: http://de.wikipedia.org/wiki/Blendenreihe_(Optik) (2012)

[40] Toshiba: *TCM8230MD VGA CAMERA MODULE* (Datenblatt)

[41] STMicroelectronics: *VS6624 1.3 Megapixel single-chip camera module* (Datenblatt)

[42] Zumtobel Lighting GmbH, URL: http://www.zumtobel.com (2012)

[43] Zumtobel Lighting GmbH: *LSD Lichtsensor (Deckenmontage) für die Erfassung des in den Raum einfallenden Tageslichts* (Datenblatt/Montagevorschriften)

[44] Burr-Brown/Texas Instruments: *XTR115 4-20mA CURRENT LOOP TRANSMITTERS* (Datenblatt)

[45] Digitalkamera.de: *Datenblatt für Canon PowerShot S30* (Online Datenblatt), URL: http://www.digitalkamera.de/Kamera/Canon/PowerShot_S30.aspx (2012)

[46] Canon Hack Development Kit, URL: http://chdk.wikia.com/wiki/CHDK (2012)

[47] Software HDR Shop, URL: http://gl.ict.usc.edu/HDRShop/ (2012)

[48] Grossberg M. D., *Nayar S. K.: Determining the Camera Response from Images: What Is Knowable?*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 11, NOVEMBER 2003

[50] Lu, Huang, Wu, Cheng, Chuang: *High Dynamic Range Image Reconstruction from Handheld Cameras*, National Taiwan University, 2009

[51] Energy Micro - Energy Friendly Microcontrollers and Radios, URL: http://www.energymicro.com (2012)

[52] Energy Micro: *EFM32G Reference Manual Gecko Series* (Reference Manual), URL: http://www.energymicro.com/products/efm32g880f128-efm32g880f64-efm32g880f32 (2012)

[53] Olimex: *EFM32G880F128-H Development Board* (Users Manual), URL: http://www.olimex.com/dev/pdf/ARM/EM/EFM32G880F128-H.pdf (2012)

[54] STMicroelectronics: *STM32 F2 series of high-performance MCUs*, URL: http://www.st.com/internet/mcu/subclass/1520.jsp (2012)

[55] Linear Technology: *LTspice* (Software zur Schaltungssimulation), URL: http://www.linear.com/designtools/software/ (2012)

[56] Cadsoft: *CadSoft EAGLE PCB Design Software*, URL: http://www.cadsoftusa.com/eagle-pcb-design-software/?lang=de (2012)

[57] Linear Technology: *LT1490A/LT1491A, Dual/Quad Over-The-Top, Micropower Rail-to-Rail, Input and Output Op Amps* (Datenblatt)

[58] Linear Technology: *LTC1474/LTC1475, Low Quiescent Current, High Efficiency Step-Down Converters* (Datenblatt)

[59] Texas Instruments: *TPS793xx, ULTRALOW-NOISE, HIGHPSRR, FAST RF 200mA-LOW-DROPOUT LINEAR REGULATORS* (Datenblatt)

[60] Texas Instruments: *MAX3318E 2.5 V 460 kbps RS 232 TRANSCEIVER WITH ±15 kV ESD PROTECTION* (Datenblatt)

[61] ARM: *CMSIS - Cortex Microcontroller Software Interface Standard*, URL: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php

[62] Wikipedia: *List of common resulutions*, URL: http://en.wikipedia.org/wiki/List_of_common_resolutions (2012)

[63] Wikipedia: *Image sensor format*, URL: http://en.wikipedia.org/wiki/Image_sensor_format (2012)