MASTERARBEIT

HIGHSPEED FOTOGRAFIE

Embedded Systems und Mikrocontroller im Vergleich

ausgeführt am

Studiengang

Informationstechnologien und Wirtschaftsinformatik

Von: Stefan Knopper, BSc

Personenkennzeichen: 1510320013

Voitsberg, am 12. Dezember 2016 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

Zuerst möchte ich mich bei allen bedanken, die mich bei der Durchführung meiner Masterarbeit begleitet haben und durch ihre fachliche und persönliche Unterstützung zum Gelingen beigetragen haben.

Ganz besonders bedanken möchte ich mich bei meinem Betreuer Dipl. Ing. (FH) Günther Zwetti, der mir während der gesamten Zeit mit Rat und Tat zur Seite gestanden ist. Und dafür, dass er stets auch sehr kurzfristig für mich Zeit hatte.

Ich möchte ich mich auch bei meiner Mutter Irmgard bedanken. Dafür, dass sie jederzeit für mich da ist und immer ein offenes Ohr für meine Probleme und Anliegen hat.

Ganz besonders möchte ich mich bei den beiden wichtigsten Menschen in meinem Leben bedanken, bei meiner Frau Sabrina und bei meiner Tochter Sarah. Dafür, dass sie jederzeit für mich da sind, mich bei meiner Ausbildung unterstützen, motivieren und während der gesamten Studienzeit Rücksicht auf mich genommen haben. Des Weiteren möchte ich mich noch einmal gesondert bei meiner Frau für das Korrekturlesen meiner Masterarbeit und den zahlreichen weiteren Abgabedokumenten während des gesamten Studiums bedanken.

KURZFASSUNG

Aufgrund des technischen Fortschritts der letzten Jahrzehnte hat sich das Bild des Fotografen stark gewandelt. Durch die immer besser werdenden Produktionsverfahren und dem damit verbundenen Fallen der Produktionskosten, sind hochwertige Spiegelreflexkameras bereits für wenige hundert Euro erhältlich. Dies ist einer der Gründe warum die Fotocommunity in den letzten Jahren einen wahren Boom erlebt hat. Fotoenthusiasten rund um die Welt teilen im Internet ihre Erfahrungen und Anleitungen, um in die verschiedenen Bereiche der Fotografie einzutauchen.

Einer dieser Bereiche ist die Highspeed-Fotografie, welche zu den am schwersten zugänglichen Gebieten in der Fotografie zählt, da viele der möglichen Ausprägungen, wie etwa Tropfenfotografie und Ballistik Shooting, nur mit spezieller Hardware umsetzbar sind. Während einfache Settings, wie beispielsweise das Fotografieren eines fallenden Topfens, noch manuell durch Probieren machbar sind, gibt es komplexere Szenarien, wie Tropfen-auf-Tropfen Bilder, die ohne spezielle Hardware nicht realisierbar sind.

Oftmals wird im Hobbyfotografie-Bereich ein Mikrocontroller wie der Arduino UNO eingesetzt. Diese Mikrocontroller haben allerdings den Nachteil, dass sie für Anwender ohne entsprechende Kenntnisse nur schwer programmierbar sind und demBenutzer kein grafisches Interface anbieten, um alle benötigten Einstellungen vornehmen zu können.

Um diese Einschränkungen des Mikrocontrollers zu umgehen, wird in dieser Arbeit ein Raspberry Pi genutzt, welcher ein Embedded System darstellt und damit zwar eine grafische Oberfläche und Netzwerkfähigkeit bietet, jedoch nicht echtzeitfähig ist.

Im Zuge dieser Arbeit wird nun mithilfe einer Literaturrecherche und Experimenten ein Raspberry Pi soweit optimiert, dass er in der Lage ist Echtzeitanwendungen im Bereich der Highspeed-Fotografie durchzuführen. Durch einen Vergleich der Ergebnisse zwischen einer Microcontroller – basierten und einer Raspberry Pi Lösung wird analysiert, ob der Raspberry Pi nach den durchgeführten Verbesserungen in der Lage ist, eine ausreichende Echtzeitfähigkeit für die Highspeed-Fotografie zur Verfügung zu stellen und somit Highspeed Fotografie zu ermöglichen.

ABSTRACT

Due to technical improvements made in the last decade, photography has changed significantly. Improvements in manufacturing processes have lowered production costs and made single-lens reflex cameras available for only a few hundred euros. This is one of the reasons for the enormous increase in the number of photographers in the last few years. Photo enthusiasts around the world are sharing their experiences and tutorials on various aspects of photography.

One of these aspects is high-speed photography. This practice is difficult to practice because special hardware is required to work on complex methods such as drop photography or ballistic shootings.

Hobby photographers often use a microcontroller, such as the Arduino UNO. The disadvantage of this device is that it is hard to program for users lacking prior programming knowledge. Currenly, it is not possible to run software on the device which could assist the user with a graphical user interface for adjusting the settings or sharing data with other users on the internet.

This thesis uses a Raspberry Pi, a type of embedded system, to overcome the limitations of a microcontroller. The Raspberry Pi uses a Linux-based operating system which features a graphical user interface and network connectivity, but does not run real-time applications.

As a part of this thesis, a literature research and experiments were performed to optimize a Raspberry Pi such that it can run real-time software for high-speed photography. To this end, a real-time patch was to the operating system.

After the optimization, several tests were executed for both systems. The comparison and analysis of the results showed if the executed improvements on the operating system of the Raspberry Pi enabled to run real-time applications with sufficient performance for high speed photography on this device.

INHALTSVERZEICHNIS

1 EINLEITUNG

Durch den technischen Fortschritt der letzten Jahrzehnte hat sich das Bild des Fotografen stark gewandelt. Heute ist es durch die Nutzung von digitalen Kameras möglich auf umständliches Entwickeln der Fotos zu verzichten. Die Entwicklung wurde dabei nicht einfacher, es hat sich lediglich das Medium gewandelt. Anstatt von Chemikalien wird heute ein Computer mit spezieller Software genutzt, um Bilder zu bearbeiten.

Durch diesen Fortschritt wurde die Nutzung von Kameras immer einfacher. Durch die Weiterentwicklung bei den Fertigungsprozessen bei der Herstellung von Kameras und durch die Produktion hoher Stückzahlen sind Kameras auch extrem im Preise gefallen. Heute kann ein Einsteigermodell meist schon für unter 500 Euro erworben werden.

All diese Faktoren haben zu einem regelrechten Boom der Fotografie geführt, sodass die meisten in ihrem Familien- und Freundeskreis mindestens einen Hobbyfotografen kennen. Diese Hobbyfotografen sind auch bereit mehrere tausend Euro in ihr Equipment zu investieren.

Im Internet sind unzählige Tutorials online verfügbar, um in die verschiedensten Bereiche der Fotografie einzutauchen. Während einige Sparten, wie die Makrofotografie, sehr einsteigerfreundlich sind und lediglich ein spezielles Objektiv oder spezielle Zwischenringe sowie ein Stativ voraussetzen, gibt es auch anspruchsvollere Themengebiete, bei denen entweder sehr teures Equipment benötigt wird, oder sich aufgrund ihrer Komplexität ein sehr hoher Frustfaktor einstellt.

Zu einem dieser anspruchsvolleren Bereiche zählt die Highspeed-Fotografie. Einerseits kann sehr teures Equipment um mehrere zigtausend Euro gekauft werden, andererseits können auch gute Fotos im Trial-and-Error Verfahren erstellt werden.

Beim zweiten Ansatz ist der Frustfaktor sehr hoch, da teilweise hunderte Fotos gemacht werden müssen, um durch Zufall ein gutes Foto zu bekommen. Der Frustfaktor steigt noch weiter, da diese Verfahren nur für sehr wenige und leichte Settings funktioniert. Alle etwas anspruchsvolleren Szenarien, wie etwa das Fotografieren einer Gewehrkugel im Flug, können mit dieser Methodik nicht umgesetzt werden.

Für all jene, die weder sehr teures Equipment kaufen, noch sich auf einfache Szenarien beschränken wollen, die nur durch Glückstreffer gute Ergebnisse liefern, gibt es noch eine dritte Möglichkeit. Sie können den Vorgang des Fotografierens durch die Nutzung eines Mikrocontrollers automatisieren, um zeitgesteuert den Auslöser genau im richtigen Moment zu drücken.

Dies stellt eine sehr technische Lösung dar und kann daher nur von technisch sehr versierten Fotografen gut genutzt werden. Die meisten verfügbaren Mikrocontroller müssen entweder für jedes Setting extra programmiert werden, oder sie besitzen zig verschiedene Drehknöpfe, um die Zeitspannen für alle Aktoren händisch einzustellen.

Für die programmierbaren Mikrocontroller muss der Fotograf jederzeit dazu bereit sein das Programm anzupassen und den Controller neu zu programmieren. Für nicht-programmierbare Produkte besteht lediglich die Möglichkeit die Zeiten selbst zu berechnen (Entfernungen, Geschwindigkeit, usw.) und danach im Trial-and-Error Verfahren nachzujustieren.

In dieser Arbeit soll ein zusätzlicher Weg geschaffen werden, der es Nutzern ermöglicht ihre Werte, Einstellungen und Settings weiterzugeben, um sie später einfach wieder auszuwählen und sofort loslegen zu können. Um dieses Vorhaben umzusetzen ist die Verwendung eines Mikrocontrollers nicht ausreichend, daher wird versucht ein Embedded System dahingehend zu optimieren, dass es in der Lage ist Sensoren und Aktoren in Echtzeit zu steuern.

1.1 Motivation und Zielsetzung

Mittels der Highspeed-Fotografie können eine Reihe von verschiedenen Impressionen erstellt werden. Eine der häufigsten Formen ist das Erstellen von sogenannten „Frozen Pictures“. Dabei handelt es sich um Bilder, bei denen eine Bewegung gänzlich eingefroren wird. Oftmals sind das Glühbirnen, die bersten oder ein Objekt, das auf einer anderen Oberfläche aufschlägt. Dies könnte beispielsweise ein Tropfen Wasser sein, der auf einer Wasseroberfläche einschlägt, oder eine Frucht, die in dem Moment fotografiert wird, in dem sie in eine Flüssigkeit eintaucht. Die Möglichkeiten sind dabei schier unendlich. Was jedoch alle Bilder gemein haben ist, dass sie versuchen eine schnelle Bewegung scharf festzuhalten. Dies impliziert bereits, dass es bei der Highspeed-Fotografie ein enges zeitliches Fenster gibt.

Um Fotos im Bereich der Highspeed-Fotografie zu machen, werden in der Regel spezielle Geräte und Wissen benötigt. So muss ein Fotograf nicht nur über die Kenntnisse für einen solchen Fotoaufbau verfügen, sondern auch noch über sehr teures Equipment. Alternativ zu speziellen Highspeed-Kameras kann allerdings auch ein Mikrocontroller genutzt werden, um die Steuerung über eine handelsübliche Spiegelreflexkamera zu übernehmen und einen oder mehrere Blitze, durch die eine solche Bewegung (< 1/20.000s) überhaupt erst eingefroren werden kann.

Damit eine Spiegelreflexkamera durch einen Mikrocontroller gesteuert werden kann, muss dieser entsprechend programmiert werden. Ein Mikrocontroller kann aber immer nur einen Programmablauf enthalten. Aus diesem Grund kann ein Fotograf nur Adaptionen an den Einstellungen und am Programm vornehmen, indem er ihn neu programmiert. Viele Fotografen sind dazu jedoch nicht in der Lage, weswegen diese Art der Highspeed-Fotografie bislang nur technisch sehr versierten Personen vorbehalten ist.

Im Zuge dieser Arbeit soll herausgefunden werden, ob und unter welchen Voraussetzungen es möglich ist, mit einem Embedded System, wie etwa einem Raspberry Pi, Echtzeitanwendungen, deren Lauffähigkeit ansonsten ausschließlich auf Mikrocontroller beschränkt ist, auszuführen.

Dadurch soll es ermöglicht werden ein kompaktes Embedded System zu nutzen, um die Kamera und Aktoren zu steuern. Mit dem System soll es weiter möglich sein, zuverlässig Fotos im Bereich der Highspeed-Fotografie zu machen. Außerdem soll es die Möglichkeit bieten einfache

Einstellungen im Programmablauf vornehmen zu können. Dadurch soll ein Neuprogrammieren von Abläufen während eines Fotoshootings vermieden werden.

1.2 Forschungsfrage und Hypothesen

Zur Beantwortung der Frage, unter welchen Voraussetzungen und unter Einhaltung welcher Restriktionen es möglich ist, mit einem Embedded System über digitale In- und Outputs verschiedene Sensoren und Aktoren in Echtzeit auszulesen und anzusteuern, um damit zeitkritische Echtzeitanwendungen auszuführen, wurde eine Alternativhypothese und eine Nullhypothese aufgestellt.

Die Alternativhypothese besagt, dass es durch die Optimierung eines Embedded Systems möglich ist, mit diesem im hoch zeitkritischen Anwendungsbereich der Highspeed Fotografie alle Sensoren und Aktoren in Echtzeit auszulesen und anzusteuern.

Die Nullhypothese besagt, dass es trotz Optimierung eines Embedded Systems nicht möglich ist, mit diesem im hoch zeitkritischen Anwendungsbereich der Highspeed Fotografie alle Sensoren und Aktoren in Echtzeit auszulesen und anzusteuern.

Durch die nachfolgend dargestellte Vorgehensweise soll eine der beiden Hypothesen verworfen werden. Die dann noch gültige Hypothese wird herangezogen, um die Forschungsfrage zu beantworten.

1.3 Vorgehensweise

Mittels Literatur wird der Begriff Echtzeit im Umfeld der IT bestimmt. Zusätzlich wird die Technik in den Digitalkameras und alle Begriffe, die für das Verständnis dieser Arbeit notwendig sind, aufgearbeitet. Weiter wird herausgearbeitet welche Anforderungen die Highspeed-Fotografie an das Fotoequipment und an die Umgebung stellt, um Fotos in der Highspeed-Fotografie Technik zu erstellen.

Anschließend wird erklärt wie diese Komponenten zusammenarbeiten und warum es möglich ist mit normalen Kameras Fotos in dieser Technik zu erstellen. Mit Rücksichtnahme auf die bereits ausgearbeiteten Anforderungen wird ausgearbeitet wie der Vorgang automatisiert werden kann und was dafür notwendig ist. Aufbauend auf diese Anforderungen wird beschrieben wie ein Mikrocontroller oder ein Embedded System eingesetzt werden kann.

Darauf folgend werden die Anforderungen an den Mikrocontroller und das Embedded System aufgearbeitet, um das Embedded System soweit zu optimieren, dass es damit möglich ist Anwendungen in Echtzeit laufen zu lassen. Mögliche Versuchsaufbauten und deren physikalische Grundlagen werden beschrieben, um darauf basierend einen Vergleich der Echtzeitfähigkeit sowie deren Messbarkeit auszuarbeiten und festzustellen.

Im Anschluss wird ein Testsystem entwickelt. Dazu wird dessen Aufbau für beide genutzten Systeme beschrieben und die eingesetzten Schaltungen werden erklärt. Die notwendigen

Programme werden erstellt und vorgestellt. Weiter wird auf die Zusammenhänge zwischen den einzelnen Komponenten eingegangen, um zu erläutern, wie es zum Design des Systems gekommen ist und was als Ergebnis erwartet wird.

Nach der theoretischen Beschreibung der Tests werden sie aufgebaut und durchgeführt. Die Ergebnisse werden erfasst und dokumentiert. Falls es notwendig ist, werden Tests adaptiert, die Änderungen beschrieben und die Tests erneut durchgeführt.

Abschließend werden alle Ergebnisse aufgearbeitet und präsentiert. Die vorgestellten Ergebnisse werden diskutiert und eine Schlussfolgerung erstellt. Anhand dieser wird eine der Hypothesen verworfen und die Forschungsfrage beantwortet. Danach erfolgt noch ein Ausblick darauf, was mit den gewonnenen Erkenntnissen gemacht werden kann.

2 TECHNISCHE GRUNDLAGEN DER FOTOGRAFIE

Das Ziel der Highspeed-Fotografie ist es Ereignisse zu erfassen, die nur sehr kurz auftreten. Sie lassen sich normalerweise nicht unter den üblichen Bedingungen und mit einer Standardausrüstung fotografieren, da die Reaktionszeit des Fotografen zu gering ist und dieser auch nicht die Möglichkeit hat die Auslöseverzögerung der Kamera zu kompensieren. (Steeg, 2014)

Ein weiteres Ziel kann es sein ein Bild zu erzeugen, bei dem keine Bewegungsunschärfe auftritt. Diese entsteht wenn sich ein Objekt während des Fotografierens bewegt. Der sich bewegende Teil ist dabei unscharf. Es muss darauf geachtet werden, dass die Bewegungsunschärfe nicht mit dem Verwackeln der Kamera verwechselt wird. Einfaches Verwackeln kann dadurch erkannt werden, dass nicht nur ein Teil des Bildes unscharf ist, sondern das gesamte Bild. (Steeg, 2014)

Um die Bewegungsunschärfe zu vermeiden, wird im Bereich der Highspeed-Fotografie mit sehr kurzen Belichtungszeiten gearbeitet. Der Zweck ist es einen möglichst kurzen Augenblick zu erfassen, in dem sich auch extrem schnelle Objekte auf dem Bild kaum bewegen. Beispielsweise ein Wassertropfen, der aus einem Meter Höhe herunterfällt und dann auf einer Wasseroberfläche aufschlägt. (Jorns, 2012)

Zum einen ist es sehr schwierig den richtigen Auslösezeitpunkt zu treffen. Wird der Auslöser etwas zu früh gedrückt ist der Tropfen noch in der Luft, wird er hingegen etwas zu spät gedrückt ist das Ereignis schon vorbei. Der richtige Zeitpunkt ist aber nur ein Teil der Highspeed-Fotografie. Ein anderer Teil ist das Festhalten von schnellen Objekten. Wenn das Foto zum richtigen Zeitpunkt gemacht wird, kann es trotzdem sein, dass das Objekt nicht oder nur verschwommen auf dem Foto zu sehen ist. Dies hängt mit der Länge der Belichtung, auch Belichtungszeit genannt, zusammen. (Jorns, 2012)

Um diese beiden Anforderungen zu erfüllen gibt es mehrere Ansätze, die im Folgenden erklärt werden.

2.1 Aufbau von digitalen Kameras

Für die Tests wird eine Sony Alpha 35 und eine Sony Alpha 65 genutzt. Bei diesen Kameras handelt es sich um Digitalkameras mit einem Wechselobjektiv. Sie basieren auf dem Sony Single Lens Translucent (SLT) Prinzip. Damit ähneln sie den digitalen Spiegelreflex-Kameras, welche auch als Digital Single Lens Reflex (DSLR) bekannt sind. Der größte Unterschied liegt im Inneren der Kameras. (Haasz, 2012)

Abbildung 1 zeigt eine Sony Alpha 65. Lichtstrahlen treffen von vorne auf ein auf der Kamera montiertes Objektiv. Das Objektiv kanalisiert die Lichtstrahlen durch mehrere Linsengruppen auf

einen fix montierten, teilweise durchsichtigen Spiegel, der direkt hinter dem Objektiv in der Kamera verbaut ist. In den Objektiven selbst ist eine Blende verbaut, die die Menge an Licht, die das Objektiv passieren kann, reguliert. (Haasz, 2012)

Der teilweise durchsichtige Spiegel teilt die Lichtstrahlen in zwei Teile. Der erste und geringere Teil des Lichts wird auf die Autofokuseinheit in der Kamera umgelenkt. Die Menge entspricht ca. 20%. Da der Spiegel fix montiert ist, wird das Licht auch dann umgelenkt, wenn der Autofokus der Kamera nicht aktiviert ist. (Haasz, 2012)

Das restliche Licht trifft direkt auf den Sensor der Kamera. Bei diesem Design gibt es nur einen digitalen Sucher, der direkt vom Sensor mit Daten gespeist wird. (Haasz, 2012)

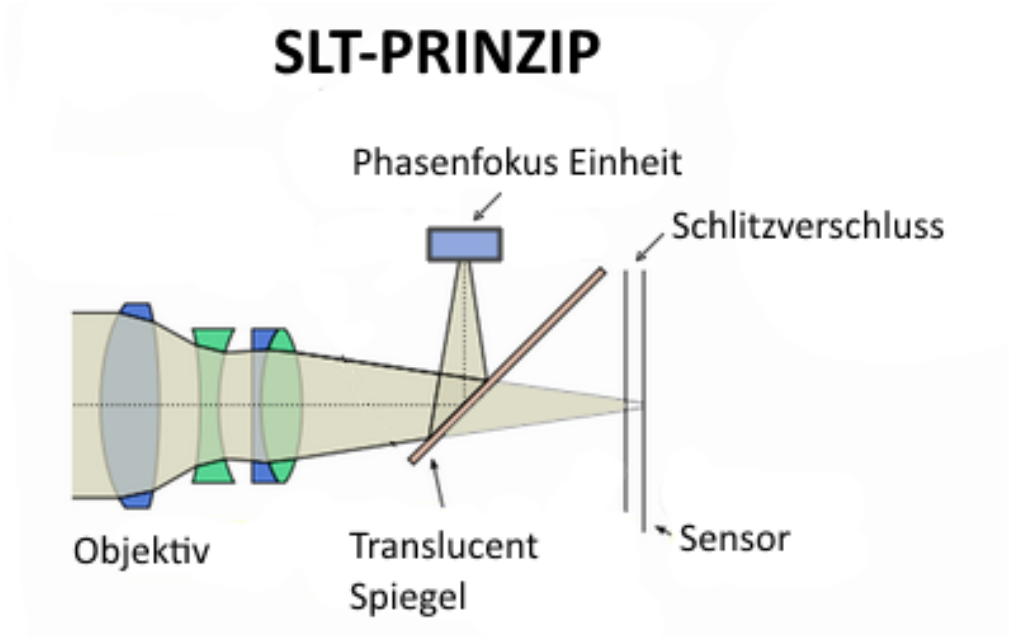

Abbildung 1: Aufbau einer Sony Alpha 65. (Sony, 2011)

Abbildung 2 zeigt den Aufbau einer Spiegelreflex Kamera. Bei DSLR-Kameras trifft das Licht von vorne auf das Objektiv auf und wird, gleich wie bei den SLT Modellen von Sony, durch mehrere Linsengruppen durch das Objektiv in Richtung des Sensors kanalisiert. In diesen Objektiven ist ebenfalls eine Blende verbaut, die der Regulierung der Lichtmenge, die durch das Objektiv geleitet wird, dient. (Gross, 2009)

Hinter dem Objektiv ist ein beweglicher Spiegel montiert. Die Bauart des Spiegels kann von Hersteller zu Hersteller variieren, die Funktionsweise bleibt aber immer die gleiche. Ein Teil des Lichts wird auf das Autofokusmodul umgeleitet und das restliche Licht wird in den Sucher umgeleitet. Bei diesen Modellen sieht der Fotograf die echte Umgebung und kein elektronisches Bild, wenn er durch den Sucher sieht. (Gross, 2009)

Wird der Auslöser gedrückt klappt die Elektronik den Spiegel weg und das gesamte Licht trifft auf den Sensor. Im Unterschied zu SLT-Kameras geht kein zusätzliches Licht verloren. (Gross, 2009)

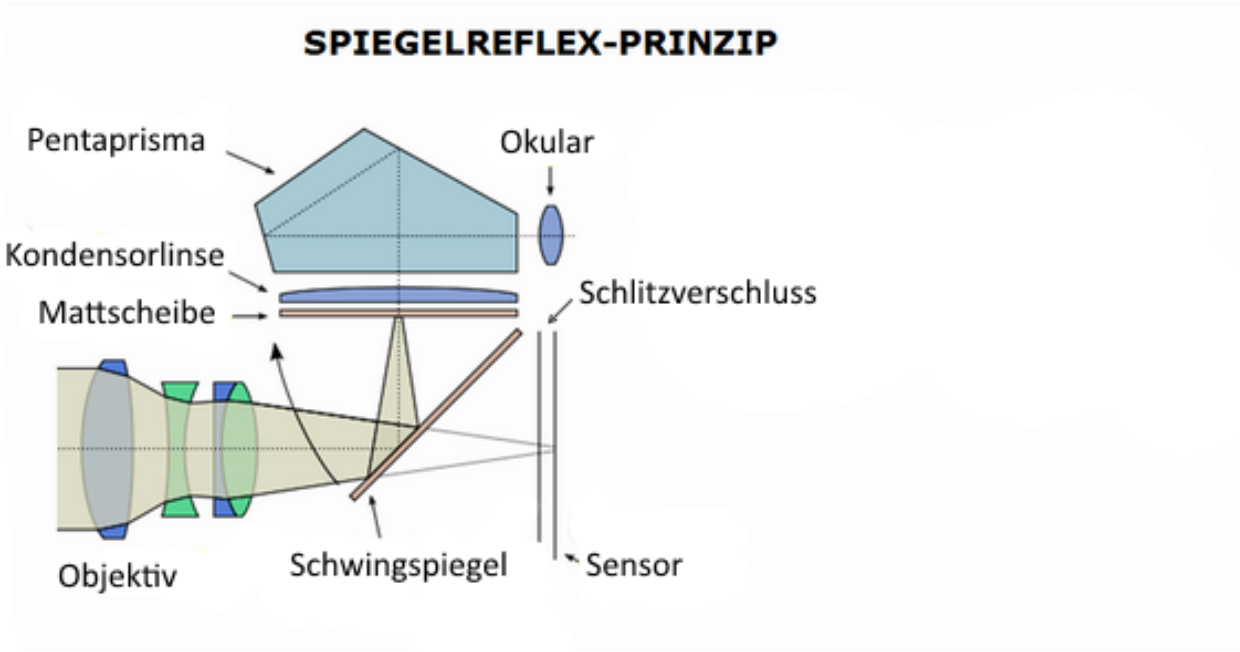

Abbildung 2: Funktionsweise einer Spiegelreflex Kamera in Anlehnung an (Otto & Hillebrand, 2013)

Bei beiden Kameratypen ist zwischen dem Spiegel und dem Sensor ein Verschluss angebracht, der das Auslösen von Aufnahmen steuert. (Gross, 2009) (Haasz, 2012)

Als Sensor kommt in den Testkameras ein Active Pixel Sensor mit Crop-Faktor 1:1.5 (APS-C), der als Complementary metal-oxide-semiconductor (CMOS) gefertigt wurde, zum Einsatz. Die Sensorgröße beträgt wie in Abbildung 3 gezeigt wird, bei einem Crop-Faktor von 1:1.5 im Vergleich zum Kleinbildformat, 23,5 x 15,6 mm. (Sony, 2011) (Sony, 2011)

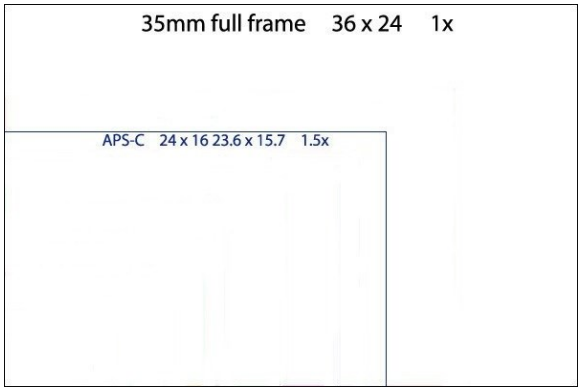

Abbildung 3: Vergleich Kleinbild vs. APS-C. (Sony, 2011)

Der Crop-Faktor verändert die Größe des Bildausschnitts im Vergleich zu anderen Sensorgrößen. Wie bereits in Abbildung 3 gezeigt ist der sichtbare Bildabschnitt eines APS-C Sensors kleiner als bei einem 35 mm Kleinbildsensor. Der Crop-Faktor gibt an, um wieviel das Bild tatsächlich kleiner ist. Weiter kann der Crop-Faktor genutzt werden, um zu berechnen welche Brennweite eingestellt werden muss, damit der sichtbare Bildabschnitt eines APS-C Sensors gleich groß ist wie bei einem 35 mm Kleinbildsensor. Die Formel zur Berechnung ist $APS\,c\,Brennweite \times CropFaktor = Kleinbildformat\,Brennweite$. Wird ein Bild mit einer APS-C Kamera mit einer Brennweite von 50 mm aufgenommen, muss bei einer Kleinbildkamera eine Brennweite von 50

mm x 1,5 = 75 mm gewählt werden, damit ein Bild mit dem exakt gleichen Bildausschnitt gemacht werden kann. (Westphalen, 2013)

Für die Testfälle werden zwei Objektive genutzt. Das Sony SAL1855 und SAL50F18. Beide Objektive gehören zur DT-Reihe von Sony. Diese Reihe bezeichnet Objektive, die speziell für Kameras mit einem APS-C Sensor entwickelt wurden. Diese Objektive weisen einen geringeren Durchmesser auf als Objektive für einen Kleinbildformat Sensor. Dies ist möglich, da der kleinere APS-C Sensor weniger Fläche besitzt und es deswegen möglich ist das Objektiv so zu bauen, dass der einfallende Lichtkegel etwas kleiner ist und genau den kleineren Sensor ausleuchtet. Wird eines dieser Objektive auf einer Kleinbildkamera montiert, ist der äußere Rand des Bildes schwarz, da kein Licht auf diesen Bereich fällt. Diese besondere Bauweise verändert auch die Berechnung der benötigten Einstellungen am Objektiv, um die Belichtung zu korrigieren. (Haasz, 2012)

Die grundlegenden Funktionen und Möglichkeiten der beiden Kameratypen sind sich sehr ähnlich. Und beide Kameratypen weisen alle notwendigen Funktionen auf, um diese Arbeit auf sie anzuwenden. Aus diesem Grund können die hier beschriebenen Methoden und Techniken auf beide Kameratypen angewandt werden, auch wenn im weiteren Verlauf nur SLT-Kameras erwähnt werden.

2.2 Kontrolle über den Lichtfluss durch die Blende

Wie bereits weiter oben in Kapitel 2.1 dargestellt, besitzen aktuelle SLT- und DSLR-Kameras austauschbare Objektive, bei denen eine Blende verbaut ist. Diese dient dazu die Menge an Licht, die das Objektiv passieren kann, zu regeln.

Abbildung 4 zeigt ein Objektiv von vorne mit verschiedenen Blendenstellungen. Es ist dabei zu erkennen, dass mit steigender Blendenzahl die Blende immer weiter geschlossen wird und der Lichtdurchlass somit geringer wird.

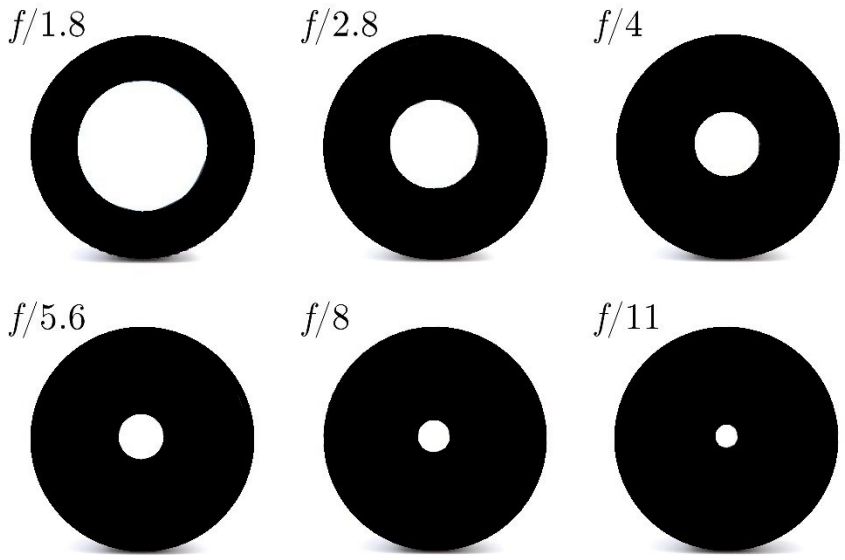

Abbildung 4: Blenden-Stufen bei einem Blick von vorne auf das Objektiv in Anlehnung an (Artmann, 2013).

Diese Blendenzahlen beruhen auf der Berechnung durch Multiplikation der vorhergehenden Blendenzahl mit $\sqrt{2}$. Das Ergebnis der Berechnung wird abgerundet auf eine Nachkommastelle angegeben. Dadurch lässt sich folgende Reihe für die ganzen Blendenzahlen berechnen: f/0.5 – f/0.7 – f/1 – f/1.4 – f/2 – f/2.8 – f/4 – f/5.6 – f/8 – f/11 – f/16 – f/22 – f/32 – f/45 – usw. Diese Zahlen werden auch als Blendenstufen bezeichnet. Bei der gesamten Zahlenreihe wird von einer Blendenreihe gesprochen. Die oben gezeigte Blendenreihe zeigt alle ganzen Blendenstufen von f/0.5 bis f/45 an. Aktuelle SLT- und DSLR-Kameras können zusätzlich zu den ganzen Blenden auch noch halbe und Drittel-Blendenstufen einstellen. (DIN ISO 517:2009-05, 2009)

Die Blendenstufen sind ausgehend von f/1 genormt. Bei f/1 entspricht die Blendenöffnung genau der Brennweite. Jede weitere ganze Blendenstufe halbiert bzw. verdoppelt die Menge an Licht, die durch das Objektiv strömt. Bei f/1.4 kommt nur mehr halb so viel und bei f/2 nur mehr die viertelte Lichtmenge durch das Objektiv im Vergleich zu f/1. (Artmann, 2013)

Das Schließen der Blendenöffnung mit steigender Blendenzahl wird durch die Berechnung der Blende dargestellt. Die Stellung der Blende wird als Blendenöffnung angegeben. Die Angabe erfolgt dabei als Bruchzahl. Die größte hier angeführte Blendenöffnung ist f/0,5. Es können theoretisch noch unendlich viele größere Blenden berechnet werden, da eine Division der Blendenzahl durch 1,4 niemals 0 ergeben wird. Praktisch führen größere Blenden aber zu Objektiven, die sehr teuer wären und nicht mehr mit einer Kamera genutzt werden können. Zum Beispiel hätte eine Blende f/0,09 bei einer Brennweite von 50 mm knapp 556 mm Durchmesser. Dieses Objektiv hätte nicht nur ein sehr hohes Gewicht, die Fotos hätten auch eine extrem kleine Schärfeebene, da diese mit steigender Größe der Blendenöffnung immer weiter abnimmt. (DIN ISO 517:2009-05, 2009) (Artmann, 2013)

Diese Darstellung der Blendenöffnung wurde gewählt, da es sich um eine relative und nicht um eine fixe Öffnung der Blende handelt. Es wird die Brennweite des Objektivs in Relation zur absoluten Öffnung der Blende gesetzt. Bei einer Brennweite von 50 mm ergibt sich folgende Rechnung: 50 mm / 0,5 = 100 mm. Die Öffnung der Blende beträgt somit 100 mm. Bei einer Brennweite von 200 mm muss die absolute Öffnung der Blende bei f/0,5 also bereits 400 mm betragen. (DIN ISO 517:2009-05, 2009)

Diese Werte beziehen sich auf einen Kleinbildformat-Sensor. Die in dieser Arbeit eingesetzten Kameras nutzen einen, wie bereits in Kapitel 2.1 beschrieben wurde, einen APS-C Sensor von Sony. Im Vergleich zum Kleinbildformat besitzt dieser Sensor einen Crop-Faktor von 1:1.5 (Sony, 2011) (Sony, 2011). Weiter wurde in Kapitel 2.1 bereits ausgeführt, dass Kleinbildkameras die angegebenen Brennweiten mit dem Faktor 1,5 multiplizieren müssen, um einen gleichen Bildausschnitt bei den Testfotos zu erreichen.

Tabelle 1 zeigt die absolute Blendenöffnung der ganzen Blendenreihe von f/5.6 bis f/32 bei den Brennweiten 18 mm, 50 mm und 55 mm. Da bei den Tests mit kleineren Blenden gearbeitet wird, werden hier nur Blendenzahlen größer f/5.6 berechnet. Die vorgestellten Brennweiten werden später in dieser Arbeit genutzt, um die Testfälle vorzubereiten. Sie wurden nicht zufällig gewählt, sondern entsprechen der minimalen und maximalen Brennweite von im Handel erhältlichen Objektiven. 18 mm und 55 mm entsprechen beispielsweise einem Standard-Zoom-Objektiv. Da es den meisten SLT- und DSLR-Kameras beim Kauf beiliegt ist es sehr verbreitet. Welche der

gezeigten Blendenstufen das Objektiv wirklich anbietet, variiert dabei zwischen den einzelnen Objektiven.

Blende	***f/5.6***	***f/8***	***f/11***	***f/16***	***f/22***	***f/32***
Brennweite						
18 mm	3,21 mm	2,25 mm	1,64 mm	1,13 mm	0,81 mm	0,56 mm
50 mm	8,93 mm	6,25 mm	4,55 mm	3,13 mm	2,27 mm	1,56 mm
55 mm	9,82 mm	6,88 mm	5,00 mm	3,44 mm	2,50 mm	1,72 mm

Tabelle 1: Absolute Blendenöffnungen

Daraus lässt sich schlussfolgern, das Objektive mit einer größeren Brennweite auch einen höheren Durchmesser haben müssen, um die gleiche Blendenöffnung wie Objektive mit einer kleinen Brennweite zu erreichen.

2.3 Der Verschluss

Die beiden SLT-Kameras benutzen einen vertikalen Schlitzverschluss, um Bilder aufzunehmen (Sony, 2011) (Sony, 2011). Ein Schlitzverschluss besteht aus 2 Verschlussvorhängen, die unabhängig voneinander gesteuert werden können. Ein Vorhang entspricht dabei einer kleinen Platte, die den Sensor verdecken kann, um zu verhindern, dass Licht auf diesen trifft. Es gibt vertikale und horizontale Schlitzverschlüsse. Dies definiert in welcher Achse sich der Verschluss bewegen kann, hat aber keinen Einfluss auf das Ergebnis (Hay & Pritschow, 1931).

In Abbildung 5 ist ein Schlitzverschluss zu sehen. Die Belichtung erfolgt über zwei Vorhänge. Der Erste gibt den Sensor für die Belichtung frei und der Zweite verdeckt den Sensor wieder und beendet damit die Belichtung. Bevor das Bild gemacht wird, ist der Verschluss geschlossen und es kommt kein Licht an den Sensor. Der erste Vorhang markiert den ersten Teil des Verschlusses. Er gibt den Sensor frei, sodass das Licht, welches durch das Objektiv kommt, auf den Sensor auftrifft. Der zweite Vorhang schließt den Verschluss wieder. Er blockiert wieder den Weg des Lichts zum Sensor (Hay & Pritschow, 1931). Bei den SLT-Kameras von Sony gibt es weiter die Möglichkeit den ersten Vorhang elektronisch zu schalten. Wird diese Funktion aktiviert, ist der Verschluss zu Beginn geöffnet und der erste Vorhang ist der Beginn des Auslesens der Sensordaten (Haasz, 2012). Wie in Abbildung 5 zu sehen ist, ist es möglich, dass der zweite Vorhang die Belichtung bereits unterbricht, während der erste Vorhang noch nicht das ganze Bild freigegeben hat.

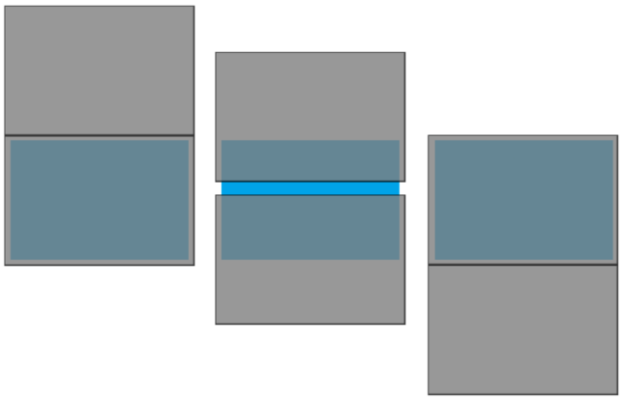

Abbildung 5: Schlitzverschluss mit einem vertikalen Verlauf. (Hay & Pritschow, 1931)

2.4 Funktionsweise des Blitzes

Wie bereits dargestellt wurde ist für die Umsetzung der Testfälle ein Blitzgerät notwendig. Bei der Auswahl eines Blitzgeräts ist auf die sogenannte Abbrennzeit zu achten. Diese gibt an wie lange der Blitz Licht abstrahlt, bis er sozusagen abgebrannt ist. Das vom Blitz abgestrahlte Licht kann vom Sensor der Kamera genutzt werden, um das Motiv aufzunehmen. Die Abbrennzeit wird auch Leuchtdauer genannt und wird in Sekunden angegeben. (Nimmervoll, 2014)

Das Abbrennen des Blitzes erfolgt dabei nach einer Kurve. Nach dem Starten des Blitzes steigt die Lichtkurve bis zu ihrem Höhepunkt stark an, danach fällt sie wieder ab. Je geringer das Restlicht ist, desto schwächer fällt die Lichtkurve. Da dieses schwache Nachleuchten keinen Effekt auf das Bild hat und das Ergebnis stark verzerrt wird, werden diese Kurven abgeschnitten. (Nimmervoll, 2014)

Derzeit haben sich zwei Angaben durchgesetzt: t0.1 und t0.5. Das t steht für die Zeit und die Zahl dahinter sind Prozent. 0.1 entspricht dabei 10% und 0.5 entspricht 50%. t0.1 bedeutet, dass die Messung des Lichts von 10% der Lichtleistung in der steigenden Kurve bis zu 10% der Lichtleistung der fallenden Kurve gemessen wird. In Abbildung 6 ist die Kurve eines Blitzgerätes zu sehen. Auf der Kurve wurden die Abbrennzeiten t0.1 und t0.5 gekennzeichnet. Es ist auch zu erkennen, dass die Zeit t0.1 weit höher ausfällt als t0.5. Obwohl die t0.1 die wichtigere, weil auch aussagekräftigere Zeitangabe ist, wird bei Blitzgeräten oft die t0.5 Zeit angegeben, weil sie sich besser anhört und das Produkt besser darstellt, als es eigentlich ist. (Nimmervoll, 2014)

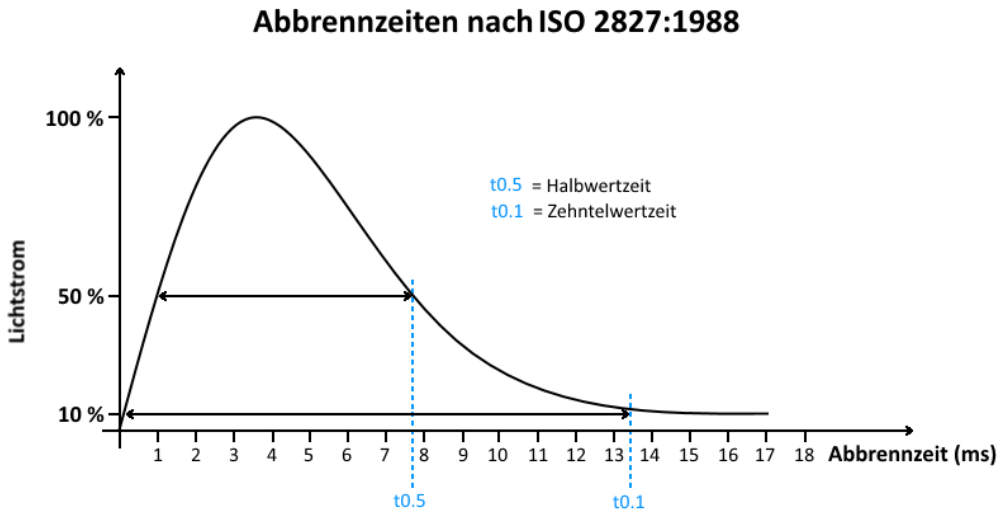

Abbildung 6: Abbrennzeit eines Blitzes in Anlehnung an (Technical Committee ISO/TC 42, 2015)

2.5 Blitzsynchronisationszeit

Die Blitzsynchronisationszeit gibt die geringste Zeitspanne an, die benötigt wird, um den ganzen Sensor durch den ersten Vorhang freizugeben und das Bild mit dem Blitz zu belichten. Bei einer Verschlusszeit, die geringer als die Blitzsynchronisationszeit ist, wird das Bild nicht gleichmäßig belichtet. Der Grund dafür liegt darin, dass der Sensor nie ganz frei liegt und das Licht des Blitzes so nur Teile des Sensors belichten kann. (Jorns, 2012)

Damit auch unterhalb der Blitzsynchronisationszeit ein Bild mittels Blitz aufgehellt werden kann, wurde die Technik der High-Speed-Synchronisation (HSS) eingeführt. Diese Technik ähnelt einem Stroboskop-Blitz. Anstatt eines einzelnen starken Blitzes werden viele schwache Blitze hintereinander abgegeben. Da bei diesem Vorgang nur ein kleiner Teil des Sensors frei ist, wird ein Großteil des Lichts vom Verschluss geblockt. Es kann also nur eine kleine Menge Licht den Verschluss passieren. Dadurch ist erheblich mehr Licht notwendig, um das Bild zu belichten, als es bei einem einfachen Blitz notwendig wäre. (George, 2008)

Abbildung 7 zeigt einen Vergleich zwischen einem normalen und einem HSS-Blitz. Oben ist ein normaler Blitz zu sehen, der oberhalb der Blitzsynchronisationszeit gemacht wird. Der Blitz erlischt dabei, bevor der zweite Vorhang geschlossen wird. Der untere Teil des Bildes zeigt einen Blitz, der mittels HSS unterhalb der Blitzsynchronisationszeit gemacht wird. Der Sensor liegt dabei zu keinem Zeitpunkt vollkommen frei, es steht aber über den gesamten Belichtungszeitraum die gleiche Menge Licht zur Verfügung, um das Bild zu belichten. (George, 2008) (Hogl, 2016)

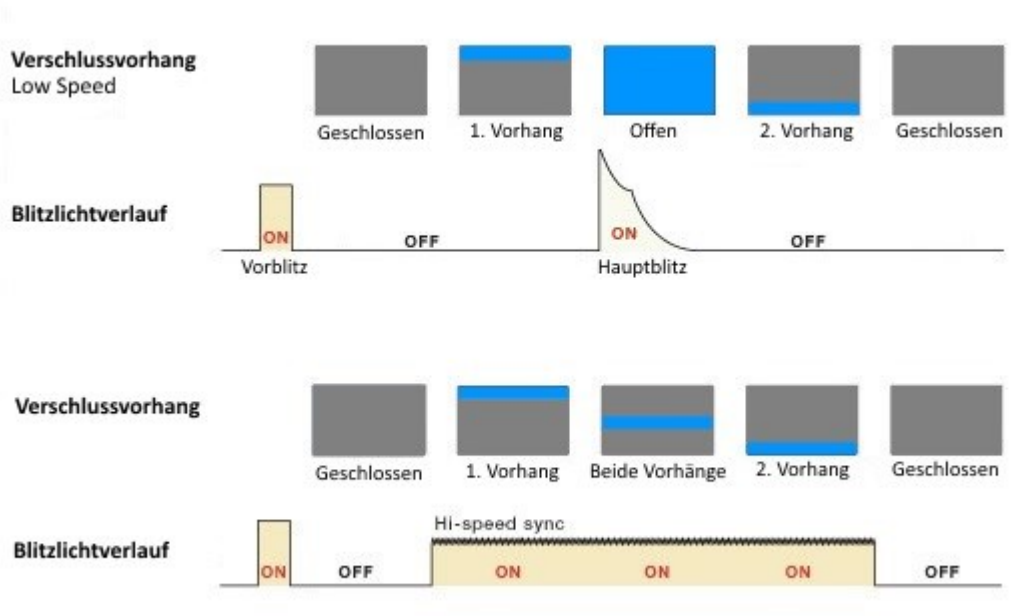

Abbildung 7: Einzelblitz vs. HSS-Blitz in Anlehnung an (Hogl, 2016).

Die Blitzsynchronisationszeit beträgt bei beiden verwendeten Kameras 1/160 Sekunde. Bei Belichtungszeiten, die geringer sind, kann kein einfacher Blitz verwendet werden, um das Bild korrekt zu belichten. (Sony, 2011) (Sony, 2011)

2.6 Bewegungsunschärfe

Bewegt sich ein Motiv während ein Foto gemacht wird, kann es sein, dass das Motiv unscharf wird. Dies wird Bewegungsunschärfe genannt. Dabei ist zwischen der normalen Unschärfe, die durch Verwackeln der Kamera, falsche Fokussierung oder falscher Blendenstellung entsteht, und der Bewegungsunschärfe zu unterscheiden. Bei der Bewegungsunschärfe handelt es sich in der Regel um eine partielle Unschärfe. Es sind dabei nur die sich bewegenden Teile des Bildes unscharf. Wird die Kamera verwackelt, falsch fokussiert oder eine falsche Blendenstellung genutzt, ist entweder das gesamte Bild unscharf oder die Schärfenebene ist nicht auf dem Objekt, das fokussiert sein soll. (Gockel, 2014)

Um Bewegungsunschärfe zu vermeiden, muss ein Foto mit einer möglichst kurzen Belichtungszeit aufgenommen werden, sodass sich das Objekt in diesem Zeitraum fast nicht bewegt. (Gockel, 2014)

Wird mit sehr kurzen Verschlusszeiten gearbeitet, ist es möglich sehr schnelle Bewegungen scharf abzubilden. Dies wird auch „Einfrieren“ genannt, da das Bild wirkt als wäre alles darauf festgefroren. Die Verschlusszeit kann jedoch nicht unendlich verkürzt werden ohne dass es zu anderen Problemen kommt. Durch eine kurze Verschlusszeit wird der Sensor dem Licht nur sehr kurz ausgesetzt. Dadurch kommt zu wenig Licht auf den Sensor um das Bild zu erfassen und

richtig zu belichten. Aus diesem Grund wird das Bild sehr dunkel oder ist oftmals sogar nur schwarz. (Nimmervoll, 2014) (Steeg, 2014)

Bei Erhöhung der Belichtungszeit trifft zwar mehr Licht auf den Sensor und das Bild wird heller, jedoch tritt ab einer gewissen Belichtungszeit wieder die Bewegungsunschärfe auf. Daher ist es notwendig die Menge an Licht, die in diesem kurzen Zeitraum auf den Sensor auftrifft, zu erhöhen. Dies kann durch die Verwendung von Blitzgeräten erreicht werden. (Nimmervoll, 2014) (Steeg, 2014)

Um das gesamte Bild mittels Blitz gleichmäßig auszuleuchten, muss auf allen Bildpunkten des Sensors die gleiche Menge an Licht auftreffen (Nimmervoll, 2014). Dies kann mit folgenden Möglichkeiten erreicht werden:

1) Belichtung über der Blitzsynchronisationszeit: Es kann mit einem normalen Blitz gearbeitet werden, jedoch können nur sehr langsame Objekte fotografiert werden, wenn es zu keiner Bewegungsunschärfe kommen soll. (Nimmervoll, 2014) (Steeg, 2014)

2) Belichtung unterhalb der Blitzsynchronisationszeit: Es kann wie bereits in Kapitel 2.5 erwähnt, nur fotografiert werden, wenn das Blitzgerät den HSS-Modus unterstützt. Dabei hängt die kürzeste Belichtungszeit von der Geschwindigkeit des Kameraverschlusses ab. Bei den beiden verwendeten Modellen ist dies 1/4000 Sekunde (Sony, 2011) (Sony, 2011). Noch kürzere Belichtungszeiten können mit schnelleren und weitaus teureren Kameras erzielt werden oder durch eine Technik, bei welcher der Blitz die einzige Quelle für Licht ist (Nimmervoll, 2014).

2.7 Den Blitz als Verschlussersatz nutzen

Damit mit den verwendeten Kameras eine Belichtungszeit unter 1/4000 Sekunde erreicht werden kann, muss eine Technik eingesetzt werden, bei der ein einzelner Blitz das gesamte Bild belichtet. Wie bereits in Kapitel 2.5 festgestellt, muss dafür die Verschlusszeit über der Blitzsynchronisationszeit liegen, da ansonsten nicht der gesamte Sensor frei liegt und nur Teile des Bildes belichtet werden. (Hogl, 2016)

Um das Auftreten der in Kapitel 2.6 erklärten Bewegungsunschärfe zu verhindern, darf das Bild aber nur sehr kurz belichtet werden. Das Ziel liegt dabei unter 1/4000 Sekunde, also unter der kürzesten Belichtungszeit, die mit den verwendeten Kameras normalerweise erreicht werden kann. (Nimmervoll, 2014)

Dafür muss der Lichteinfall durch das Umgebungslicht in die Kamera möglichst gut verringert werden. Dies kann durch Abdunkeln des Raums, dem Verwenden von Filtern und dem Schließen der Blende erfolgen. Durch diese Maßnahmen kann die Verschlusszeit soweit erhöht werden, dass diese über jener der Blitzsynchronisationszeit liegt, ohne dabei das Bild zu belichten. Wird hingegen zu viel Licht weggenommen, kann es vorkommen, dass das Bild auch durch den Einsatz des Blitzes nicht genug Licht bekommt und unterbelichtet wird. Das Ziel ist es deswegen bei einer Belichtungszeit von 1/160 Sekunde gerade noch ein nicht belichtetes, also schwarzes Bild zu erhalten. (Nimmervoll, 2014)

Desto besser die Annäherung an diese Grenze gelingt, desto geringer kann die Leistung des Blitzes sein. In Kapitel 2.5 wurde bereits dargestellt, dass der Blitz umso schneller abbrennt desto geringer die Blitzleitung ist. Durch den Einsatz dieser Technik kann bei einer Verschlusszeit von 1/160 Sekunde und einer minimalen Blitzleistung eine gesamte Belichtungszeit von bis zu 1/66660 Sekunde und geringer erreicht werden. Die tatsächlich erreichbare Belichtungszeit hängt vom verwendeten Blitzgerät ab (Nimmervoll, 2014).

Wenn das Motiv trotz des Blitzes nicht richtig ausgeleuchtet wird, kann die Leistung des Blitzes angehoben werden. Dadurch steigt aber auch die Brenndauer an. Wird also die Blitzleistung angehoben, kann es sein, dass wieder die Bewegungsunschärfe auftritt. Alternativ zum Anheben der Blitzleistung kann auch ein zusätzlicher Blitz genutzt werden. Zwei Blitzgeräte auf minimaler Leistung geben erheblich mehr Licht als nur einer. Beim Einsatz von mehreren Blitzgeräten ist darauf zu achten, dass sie exakt zur gleichen Zeit blitzen und die gleiche Abbrenndauer haben. Ist die Abbrenndauer unterschiedlich, kommt es zu einer ungleichmäßigen Ausleuchtung des Motivs und zu Unschärfen. (Gockel, 2014)

Wenn die Blitzgeräte nicht exakt zur gleichen Zeit blitzen, kommt es zum sogenannten Rolling-Shutter-Effekt. Dieser entsteht, wenn ein Bild durch mehrere Blitze belichtet wird. Sich langsam oder gar nicht bewegende Teile sehen dabei normal aus. Sich sehr schnell bewegende Teile sind je nach Auslöseabstand und Geschwindigkeit der Objekte entweder verschwommen oder scharf, aber an zwei Positionen im Bild zu erkennen. Als Beispiel kann der Rotor eines Flugzeugpropellers dienen. Bei einem Blitz sieht man ein Rotorblatt. Bei zwei schnell aufeinander folgenden Blitzen ist derselbe Propeller zweimal auf dem Bild zu sehen, wenn der zweite Blitz zeitverzögert auslöst. Aus diesem Grund wird bei der Verwendung von mehreren Blitzgeräten empfohlen, dass nur baugleiche Blitze aus derselben Produktionsserie genutzt werden. (Gockel, 2014) (Nimmervoll, 2014)

2.8 Das Auslöse-Timing

Ein weiterer Punkt warum das manuelle Fotografieren von Highspeed-Fotos nur schwer möglich ist, liegt am Timing beim Auslösen. Wie bereits dargelegt wurde, soll ein Bild möglichst schnell aufgenommen werden, damit schnelle Bewegungen eingefroren werden. Desto schneller die Bewegung ist, desto schwieriger ist es das Foto zum richtigen Zeitpunkt zu machen. Eine Frucht, die in eine Wasserschale geworfen wird, kann noch nach dem Trial-and-Error Prinzip fotografiert werden. Wenn genug Fotos gemacht werden, ist sicherlich das ein oder andere Mal ein gutes Foto dabei. Wenn es aber darum geht eine Gewehrkugel oder etwas ähnlich Schnelles zu fotografieren, wird dies im Normalfall nicht funktionieren. (Nimmervoll, 2014) (Steeg, 2014)

Es gibt folgende Einflussfaktoren, die für Verzögerungen bei einer Aufnahme verantwortlich sind:

1) Der Fotograf: Der größte Unsicherheitsfaktor bei der Fotografie ist der Fotograf selbst. Je nach Tagesverfassung des Fotografen reagiert er unterschiedlich schnell auf Ereignisse. Dadurch ist nicht vorhersehbar wie groß die Zeitverzögerung bis zum Betätigen des Auslösers ist. Selbst wenn es möglich wäre beim Auslösen immer innerhalb einer gewissen Toleranzgrenze zu

bleiben, wäre es dennoch nicht möglich eine Gewehrkugel zu fotografieren, da das Objekt vom menschlichen Auge nicht erfasst werden kann. (Steeg, 2014)

2) Die Auslöseverzögerung der Kamera: alle Digitalkameras besitzen eine Auslöseverzögerung. Sie beschreibt die Zeitspanne zwischen dem Betätigen des Auslösers und dem Erstellen des Fotos. Diese Auslöseverzögerung kann nicht verhindert werden und muss korrigiert werden, indem der Auslöser um die Auslöseverzögerung zu früh betätigt wird. (Nimmervoll, 2014) Bei den verwendeten SLT-Kameras liegt die Auslöseverzögerung bei 223 Millisekunden bei der a35 und 138 Millisekunden bei der a65. (Sony, 2011) (Sony, 2011)

3) Das Fokussieren: Damit ist das Scharfstellen des Motivs gemeint. Vor jedem Foto muss geprüft werden, ob das Motiv scharf ist und anschließend muss die Fokussierung gegebenenfalls nachjustiert werden. Dafür sind im Wesentlichen zwei Teile in der Kamera verantwortlich. (Kelby, 2007)

Die erste Komponente ist das Fokusmodul in der Kamera. Wie in Kapitel 2.1 bereits beschrieben, wird bei den eingesetzten SLT-Kameras das Phasen-Fokusmodul durchgängig mit Licht versorgt, was dazu führt, dass es zu den schnellsten Fokusmodulen auf dem Markt gehört. (Sony, 2011) (Sony, 2011)

Der zweite wesentliche Bestandteil ist die Mechanik, die den Fokus nachjustiert. Bei Sony gibt es derzeit drei Arten wie der Fokus nachgestellt werden kann. Wenn das Objektiv selbst keinen Fokusmotor besitzt, kommt ein Stangengetriebe zum Einsatz. Dazu ist in den Kameras selbst ein Antriebsmotor eingebaut, dessen Stangenkopplung das Stangengetriebe im Objektiv antreibt. Dies ist die langsamste automatische Fokussierung. Der schnellste Fokusmotor von Sony ist ein Ultraschallmotor. Dieser wird nur bei Objektiven eingesetzt, die für den professionellen Einsatz vorgesehen sind, da dieser Motor schwer und groß ist. Der dritte Fokusmotor wird bei Sony Smooth-Autofokus-Motor (SAM) genannt. Es handelt sich dabei um einen kleinen Elektromotor, der im Objektiv untergebracht ist. (Haasz, 2012)

Je nach Antriebsart ergibt das eine unterschiedlich lange Zeitspanne zum Fokussieren. Zusätzlich hängt die Zeit auch noch davon ab, wie weit der eingestellte Fokusbereich vom Zielfokusbereich entfernt liegt. Liegen die Punkte nah beieinander, schaffen es alle Motoren sehr schnell das Motiv scharf zu stellen. Muss zum Scharfstellen aber der gesamte Fokusbereich durchlaufen werden, braucht dies je nach Fokusmotor mehrere Sekunden. Aus diesem Grund muss automatisches Fokussieren in der Highspeed-Fotografie nach Möglichkeit verhindert werden. (Haasz, 2012) (Kelby, 2007)

4) Die Blitzverzögerung: Blitzgeräte besitzen, gleich wie digitale Fotoapparate, ebenfalls eine Auslöseverzögerung. Diese kann normalerweise ignoriert werden, da die Zeitspanne, in der der Blitz ausgelöst werden muss, recht großzügig ist. Im HSS-Modus beginnt der Blitz kurz vor der Belichtung und hält das Lichtlevel aufrecht, bis die Belichtung abgeschlossen ist. Bei Belichtungszeiten über der Blitzsynchronisationszeit sind kleine Zeitverzögerungen nicht relevant, solange der Blitz innerhalb des Zeitfensters auslöst, in dem der Sensor zur Gänze dem Licht ausgesetzt ist. (Hogl, 2016)

2.9 Funktionsweise der automatisierten Fotografie

Der Grundgedanke hinter der automatisierten Fotografie ist es einen möglichst großen Teil der Faktoren, die auf das Motiv einwirken, zu automatisieren. Damit soll der Zeitpunkt eines Vorganges möglichst genau berechnet, und damit auf Ereignisse immer in einem genau definierten Zeitraum reagiert werden. (Steeg, 2014)

Zusätzlich ist es notwendig alle Aktionen, die keine fixe Zeitdauer haben, zu vermeiden. Ist es zum Fotografieren eines Motivs notwendig eine Aktion durchzuführen, die bei jeder Ausführung unterschiedlich lange braucht, ist es nicht möglich das Motiv zu fotografieren, wenn es sich um ein Highspeed-Foto handelt. Als Beispiel kann das Fokussieren auf das Motiv genannt werden. Das Fokussieren braucht jedes Mal unterschiedlich lange. Dauert es einmal 45 ms und einmal 150 ms ist der zeitliche Unterschied so groß, dass mindestens eines der Fotos das Motiv verfehlt haben muss. Das Fokussieren kann umgangen werden, indem zuerst auf den Punkt fokussiert wird, auf dem später das Motiv scharf sein soll, und danach der Autofokus abgestellt wird. Auch alle anderen eventuell auftretenden Aktionen, die bei jeder Anwendung unterschiedlich lange dauern, müssen beseitigt werden. (Nimmervoll, 2014) (Steeg, 2014)

Bei einem typischen Aufbau wird das Auslösen der Kamera, der Blitze und aller Aktoren, wie beispielsweise Ventile und Sensoren, von einem externen Gerät gesteuert und überwacht. Die meisten derzeit auf dem Markt erhältlichen Steuergeräte basieren auf einem Mikrocontroller. Die Spannweite reicht von eigens dafür hergestellten Geräten bis zu Projekten, die auf universellen Geräten wie dem Arduino Uno aufsetzen. (Nimmervoll, 2014)

Bei den meisten Mikrocontrollern gibt es 3 – 12 Ausgänge, an die Geräte gekoppelt werden können. Zu jedem Ausgang gibt es noch einen Regler, der es erlaubt eine Zeitverzögerung einzustellen. Die Kontroller geben beim Start ein Signal am ersten Ausgang aus, warten danach für die voreingestellte Zeit und geben dann das Signal auf den nächsten Ausgang. Auf diese Art kann sequenziell vom ersten Ausgang bis zum letzten ein Ablauf mit zeitlichen Verzögerungen erstellt werden. Mit solchen Geräten ist es möglich sehr gute Ergebnisse zu erzielen. Die genauen Einstellungen sind aber bei jedem Setting durch Probieren neu einzustellen, da keine genauen Angaben in Millisekunden möglich sind. Bei teureren Geräten, wie dem StopShot von Cognisys, gibt es bereits ein LCD-Display, mit dem alle Zeiten auf die Millisekunde genau eingestellt werden können. (Steeg, 2014) (Nimmervoll, 2014)

Die Königsklasse bei den Mikrocontrollern ist derzeit der GlimpseCatcher von Claude Dunkel. Er ist ein Addon-Board, das direkt auf einen Arduino Uno aufgesteckt werden kann. Über ein Programm kann der GlimpseCatcher aktualisiert werden. Es besitzt 2 Trigger, an denen Sensoren angeschlossen werden können und 12 Ports für Aktoren. Sechs davon sind für Ventile und sechs für Kameras und Blitze. Alle Ports können Mikrosekunden-genau gesteuert werden. Die Möglichkeiten beschränken sich aber auch bei diesem Gerät lediglich darauf zeitliche Abläufe einzustellen, bei denen an den verschiedenen Ausgangsports ein Signal anliegt. (Nimmervoll, 2014)

Alternativ zu den kommerziellen Produkten ist es auch möglich Mikrocontroller-Boards, wie den Arduino, selbst zu programmieren und für die automatisierte Fotografie zu nutzen. Der Vorteil

dies zu tun besteht darin, viel flexibler reagieren zu können. Im Vergleich zum GlimpseCatcher ist es möglich frei zu definieren wie viele der Ports für Sensoren oder Blitze genutzt werden und wie viele für die Steuerung von Kameras. (Brühlmann, 2015)

Weiter kann auch eine größere Anzahl verschiedener Sensoren unterstützt werden, da diese frei programmiert werden können. So kann beispielsweise ein Ultraschallsensor genutzt werden, um Entfernungen zu bestimmen. Bei einer vorher festgelegten Entfernung kann die Kamera ausgelöst werden. (Brühlmann, 2015)

2.10 Beurteilung der Schärfe eines Bildes

Um die Schärfe eines Bildes zu beurteilen ist es notwendig die Schärfekreise und den Rolling-Shutter-Effekt zu berücksichtigen. Damit beurteilt werden kann, ob ein Bild unscharf ist, oder ob ein Teil des Bildes doppelt dargestellt wird. (Uhlig, 2007)

Der Schärfekreis, oder auch Zerstreuungskreis genannt, bezeichnet wie groß ein Bildpunkt dargestellt wird. Desto größer ein Bildpunkt ist, desto besser wird er als unscharf wahrgenommen. Ab welcher Größe ein Bildpunkt als unscharf erkannt wird, hängt maßgeblich vom Sensor und dessen Größe ab. Als Maß kann 1/1500 der Bilddiagonale genommen werden. Ab dieser Größe können Bildpunkte von einem Durchschnittsauge gerade noch als unscharf erkannt werden. (Uhlig, 2007)

In den Testreihen werden, wie bereits weiter oben erwähnt, APS-C Sensoren von Sony genutzt. Diese besitzen eine Bilddiagonale von ~ 27 mm. Dies führt zu einem Schärfekreisdurchmesser von 0,018 mm. Jeder Punkt, der diese Größe erreicht oder überschreitet, kann mit dem freien Auge als unscharf wahrgenommen werden.

Bei der Projektion eines Motivs auf einen Sensor werden mehrere Ebenen durchlaufen, die das Bild modellieren, damit es abgebildet werden kann. Die Einstellebene bezeichnet die entfernte Ebene, die in der Abbildung die minimalste Unschärfe aufweist. In der Fotografie ist es die Ebene, auf die der Fotograf versucht das Bild scharf zu stellen. Jede Abweichung von dieser Ebene nach vorne oder hinten vergrößert den Schärfekreis. Die Grenzen nach vorne und hinten, ab denen das Bild als unscharf erkannt wird, ist auch als Tiefenschärfe bekannt. (Monaghan, O'Connor, Cleary, & Connolly, 2015)

Die Hauptebene liegt hinter der Einstellebene und bezeichnet die Eintrittspupille, bei welcher das Licht eintritt. Dies ist im Fall einer Kamera die erste Linse des Objektivs. Die letzte Linse des Objektivs kennzeichnet das Ende der Hauptebene. Nach diesem Zeitpunkt erfolgt ein Bündeln des Lichtstrahls auf die Bildebene. Die Bildebene bezeichnet jene Ebene, auf der das Bild aufgezeichnet wird. In der digitalen Fotografie ist dies der Sensor der Kamera. Im Idealfall fällt der Schärfepunkt, bei dem der Schärfekreis sein minimales Ausmaß hat, genau auf die Bildebene. (Monaghan, O'Connor, Cleary, & Connolly, 2015)

In Abbildung 8 sind die Ebenen und der Verlauf des Lichts schematisch dargestellt. Wenn der aufgenommene Punkt hinter der Einstell-Ebene liegt, erfolgt die Bündelung des Lichtpunktes zu früh und der minimal unscharfe Punkt liegt nicht auf der Bildebene. Auf der Bildebene selbst

entsteht ein Punkt, dessen Durchmesser mit der Entfernung des Bildpunktes zur Bildebene steigt. (Grau & Pansiot, 2012)

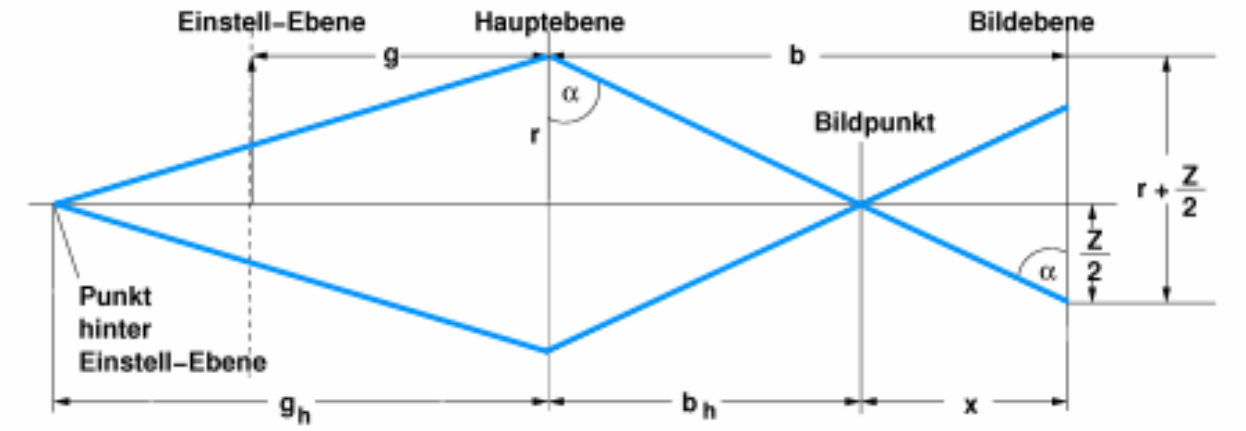

Abbildung 8: Schärfekreis auf der Bildebene in Anlehnung an (Hay & Pritschow, 1931).

Weiter hat der Rolling-Shutter-Effekt eine Auswirkung auf die Beurteilung der Schärfe eines Fotos. Er beschreibt nicht eine Unschärfe selbst, sondern das mehrfache Abbilden eines Teilbereichs des Motivs, weil es sich zu schnell bewegt. Hierbei wird derselbe Pixel mehrfach abgebildet. Ab einem Abstand von 2 Winkelminuten können solche doppelten Aufzeichnungen als getrennte Objekte erkannt werden. Bei einem geringeren Abstand werden sie nicht als getrennt erfasst und das Auge glaubt das Bild ist unscharf, da es als ein großer Punkt bzw. Schärfekreis erkannt wird. (Grau & Pansiot, 2012)

Eine Winkelminute ist der sechzigste Teil eines Grades. Dies bedeutet, dass mit steigender Entfernung des Objektes auch der Abstand zweier Punkte steigen kann, ohne eine Winkelminute zu überschreiten. (Grau & Pansiot, 2012)

Durch den Schärfekreis und den Rolling-Shutter-Effekt kann die Schärfe eines Bildes festgehalten werden. Wenn bei einem einzufrierenden Motiv der sich schnell bewegende Teil weder zu große Schärfekreise noch einen Rolling-Shutter-Effekt aufweist, kann davon ausgegangen werden, dass das Bild scharf abgelichtet wurde. (Grau & Pansiot, 2012)

3 TECHNISCHE BESCHREIBUNG DER HARDWARE

Im folgenden Kapitel wird beschrieben welche Hardware derzeit in der Highspeed-Fotografie eingesetzt wird und wie diese funktioniert. Weiter wird aufgezeigt welche Hardware noch genutzt werden kann und wie dies umgesetzt werden kann.

Im Folgenden wird ein Arduino UNO beschrieben, der von (Nimmervoll, 2014) getestet wurde. Es wird die Funktionsweise des Geräts erläutert und auf die potentiellen Möglichkeiten eingegangen. Es wird auch aufgezeigt wo die Grenzen des Systems liegen und was nicht mehr damit machbar ist. Weiter wird auch die Echtzeitfähigkeit dieses Systems erläutert.

Darauf folgend wird das Embedded-System vorgestellt, das modifiziert werden soll, um in der Highspeed-Fotografie eingesetzt zu werden. Es wird die Hardware erklärt und welche Möglichkeiten es gibt diese zu nutzen. Es wird das Operating System (OS) des Geräts vorgestellt und wie es verändert werden soll, damit es echtzeitfähig wird. Weiter wird auf die Grenzen des Systems eingegangen, um damit aufzuzeigen ab welchem Punkt es schwer wird eine Echtzeitfähigkeit nachzuweisen.

3.1 Definition des Begriffs Echtzeit

Der Begriff der Echtzeit wird oft mit sehr schnellen Systemen oder mit welchen, die unmittelbar auf ein Event reagieren, in Verbindung gebracht. Diese Definition trifft aber nicht zu. Echtzeit definiert nicht wie lange ein System benötigt, um eine Aktion durchzuführen. Für den Begriff Echtzeit haben sich derzeit zwei Beschreibungen hervorgetan. (Laplante, February 2006)

In der ersten Definition dieses Begriffs, arbeitet ein System in Echtzeit, wenn es die geforderte Aktion und die benötigten Daten rechtzeitig zur Verfügung stellt. (Laplante, February 2006)

Aus dieser Betrachtungsperspektive arbeiten fast alle Systeme in Echtzeit, die derzeit eingesetzt werden. Je geringer das Zeitfenster ist, in dem Aktionen durchgeführt und Daten bereitgestellt werden, desto schwerer wird es ein Echtzeitsystem umzusetzen. (Laplante, February 2006)

Für den Bereich Highspeed-Fotografie bedeutet dies, dass ein System nicht als echtzeitfähig angesehen wird, wenn ein Foto in 300 Millisekunden gemacht werden kann, für das Motiv aber eine Reaktionszeit von maximal 250 Millisekunden notwendig ist. Eine Bankomatkasse, die eine Anfrage in 20 Sekunden bearbeitet, wird aber als Echtzeit angesehen, wenn der Bezahlvorgang erfolgreich abgeschlossen wird.

Die zweite Definition dieses Begriffs definiert Echtzeit als berechenbares System mit einer definierten Grenze. Wenn ein System sich innerhalb dieser definierten Grenze aufhält, gilt es als Echtzeit-System. Bei dieser Definition gibt es noch die Unterscheidung zwischen harter Echtzeit, weicher Echtzeit und statischer Echtzeit. (Furht, et al., 1991) (Gwinn, 2005)

Von harter Echtzeit wird gesprochen, wenn bei einem System die maximale Zeit bis zur Durchführung garantiert werden kann. Solche Systeme sind notwendig, wenn ein großer Schaden mit dem Verletzen der Echtzeitregel einhergeht. Reagiert ein System beim Abtrennen einer Brennstufe einer Rakete nicht innerhalb einer gewissen Zeit, kann dies zum Verlust der gesamten Rakete führen. Hier werden nur Geräte eingesetzt, die eine harte Echtzeitausführung garantieren können. (Furht, et al., 1991)

Von weicher Echtzeit wird gesprochen, wenn ein System sich normalerweise innerhalb der festgelegten Parameter bewegt, es aber sporadisch zu Ausreißern kommen kann. Diese Definition kann beispielsweise auf eine Bankomatkasse angewandt werden. In der Regel reagiert das System innerhalb seiner festgelegten Grenzen. Kommt es zu einer Latenzschwankung bei der Verbindung und ein Aufruf dauert länger, hat dies keine Auswirkung auf die Beurteilung der Echtzeitfähigkeit des Systems. (Furht, et al., 1991)

Statische Echtzeit ist der weichen Echtzeit sehr ähnlich. Es agiert im Normalfall gleich wie ein weiches Echtzeitsystem innerhalb seiner festgelegten Parameter und darf manchmal von diesen abweichen. Der Unterschied besteht bei diesen Systemen in der Brauchbarkeit der Ergebnisse. Bei einem weichen Echtzeitsystem ist eine Verfehlung der Echtzeitparameter, wie bereits dargelegt, nicht tragisch und das Ergebnis ist auch nach dem Ablauf des Echtzeitfensters noch gültig. Bei einem statischen Echtzeitsystem sind die Ergebnisse nach dem Ablauf des Zeitfensters unbrauchbar und müssen verworfen werden. (Furht, et al., 1991)

Für den Anwendungsfall der Highspeed-Fotografie bedeutet dies, dass sie in den Bereich der statischen Echtzeitsysteme fällt. Durch ein verpasstes Zeitfenster entsteht kein großer Schaden, da der Vorgang einfach wiederholt werden kann. Läuft das System außerhalb seiner Parameter, wird das Motiv aber verfehlt und das entstandene Ergebnis ist nicht brauchbar.

Die Grenze, wann nicht mehr von einem Echtzeitsystem gesprochen werden kann, ist bei harten Echtzeitsystem einfach zu ziehen, da es nicht mehr als echtzeitfähig betrachtet werden kann, wenn es das Einhalten der Zeitlimits nicht garantieren kann. Bei statischen Echtzeitsystemen ist dies nicht so einfach festzulegen. Dies ist immer davon abhängig, ab wann das Ergebnis nicht mehr akzeptiert wird. (Laplante, February 2006)

In der Highspeed-Fotografie bedeutet dies, dass eine Echtzeitfähigkeit nicht zweifelsfrei festgestellt werden kann, es sei denn, die Versuche ergeben, dass die Umsetzung unmöglich ist und die benötigten Werte gar nicht erreicht werden.

Um eine Aussage darüber zu machen wie sich das System verhält, wird es mit einem bestehenden Mikrocontroller-Board verglichen. Die Ergebnisse können gegenübergestellt werden, um eine Aussage darüber zu treffen, ob das neue System eine ähnliche Qualität aufweisen kann.

3.2 Beschreibung eines echtzeitfähigen Arduino UNO

Für das Automatisieren der Highspeed-Fotografie soll ein Mikrocontroller mit einem Embedded-System verglichen werden. Ein Mikrocontroller ist ein Integrated Circut (IC) Chip, also ein Chip,

der viele integrierte Schaltungen enthält. Ein IC-Chip allein ist nicht ausreichend, um in der Highspeed-Fotografie eingesetzt zu werden, da ihm sehr viele Eigenschaften und Funktionen fehlen, die dafür unbedingt notwendig sind. Solch ein Chip allein hat beispielsweise nicht die Möglichkeit direkt an einen PC angeschlossen zu werden, damit er programmiert werden kann. All diese notwendigen Schnittstellen und Funktionen können auf einem Board untergebracht werden, auf dem dieser Mikrocontroller montiert wird. Ein solches Board ist zum Beispiel ein Arduino UNO. (Bartmann, 2011) Außerhalb der Vereinigten Staaten ist dieses Board auch als Genuino UNO bekannt. Im weiteren Verlauf dieser Arbeit wird aber nur der Name Ardunio genutzt. (Arduino, 2016)

3.2.1 Hardware

Die folgende Abbildung 9 zeigt die Basisversion des Arduino UNO Revision 3 Mikrocontroller-Boards.

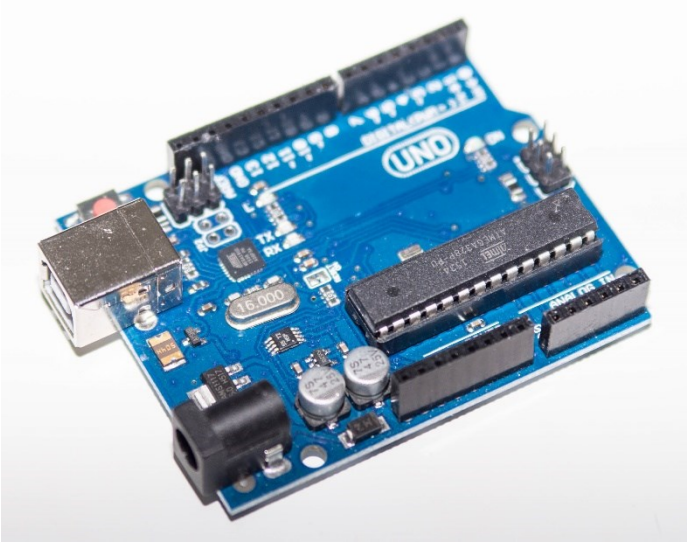

Abbildung 9: Arduino UNO Rev.3 in Anlehnung an (Arduino, 2016)

Das Herzstück des Arduino UNO Controller-Boards ist sein Mikrocontroller. Auf dem Arduino wird ein ATmega 328/P von Atmel eingesetzt. (Arduino, 2016)

Der Mikrocontroller hat 28 Pins und setzt auf einer 8-bit Reduced Instruction Set Computer (RISC) Architektur auf. Er versteht 131 Instruktionen, von denen die meisten innerhalb eines Taktzyklus ausgeführt werden können. Bei einer maximalen Taktung von 20 Mhz ergibt dies bis zu 20 Million Instructions Per Second (MIPS).

Weiters verfügt der ATmega IC über 23 General Purpose Input/Output (GPIO) Lines, die frei programmiert werden können. Er besitzt zwei 8-bit Counter und einen 16-bit Counter. (Atmel Corporation, 2016) Zusätzlich dazu verfügt der Arduino noch über einen 16 Mhz Quarzkristall, der eine noch genauere Taktung zulässt.

3.2.2 Einschränkungen

Nicht alle 23 GPIO-Pins des ATmega 328/P werden durch das Arduino Board unterstützt. Auf Abbildung 10 ist zu erkennen, dass einige der Pins mehrfach belegt sind, weswegen nicht alle theoretisch verfügbaren GPIO-Pins auch tatsächlich zur Verfügung stehen. (Atmel Corporation, 2016)

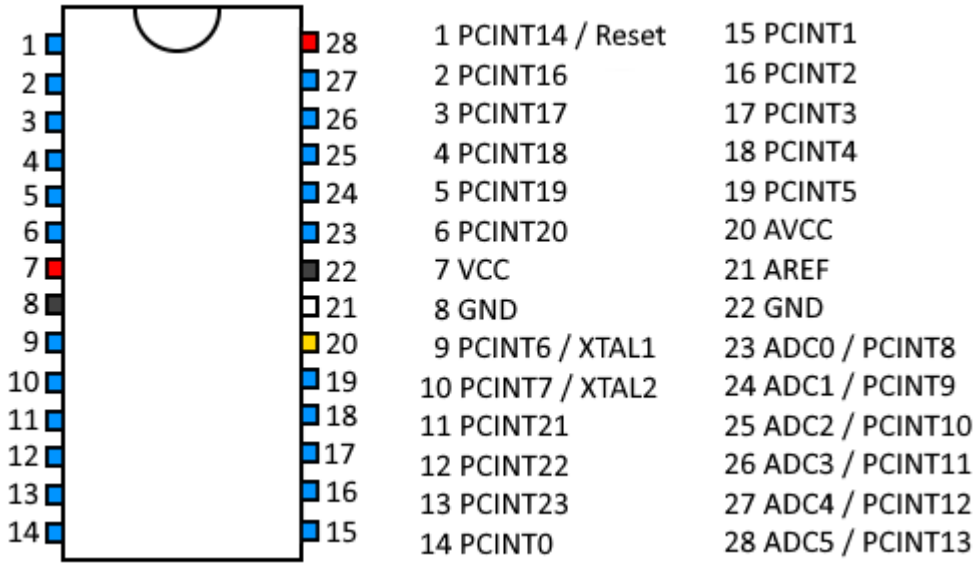

Abbildung 10: Pin-Belegung des ATmega 328/P in Anlehnung an (Atmel Corporation, 2016)

Wie Abbildung 11 zeigt, werden die Pins 23 bis 28 auf die Analog-Pins des Arduino geschalten und die Pins 1, 9 und 10 auf Reset, XTAL 1 und 2. Wenn diese Pins von den theoretisch verfügbaren abgezogen werden, bleiben noch 14 digitale GPIO-Pins übrig. Aus diesem Grund können bei einem Arduino UNO Board nur 14 Pins selbst programmiert werden. (Arduino, 2016)

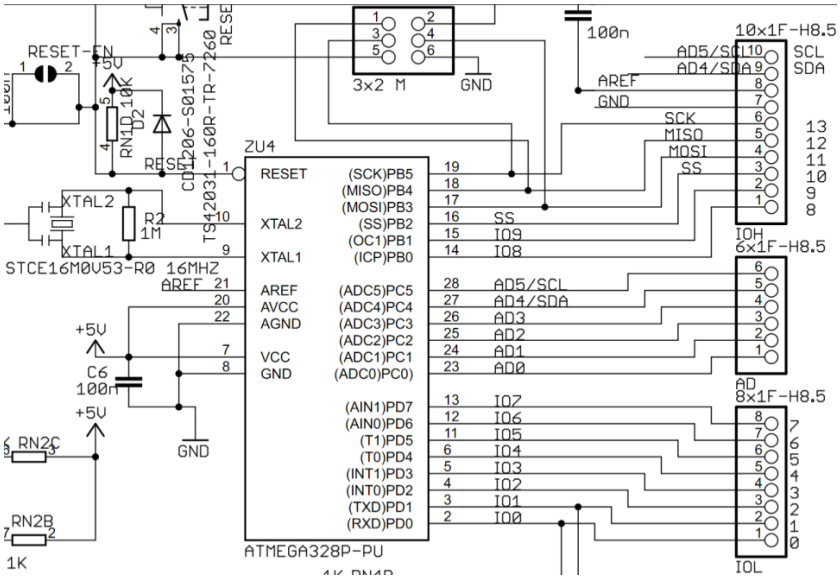

Abbildung 11: Ausschnitt aus der Arduino UNO Schaltung in Anlehnung an (Arduino, 2016)

Von den 14 GPIO-Pins, die frei programmiert werden können, ist es möglich 6 Pins für das Mapping zwischen analog und einem digitalen Signal heranzuziehen. Für diese Verfahren wird das Pulse Width Modulation (PWM) Verfahren herangezogen.

Der Arduino nutzt ein 500 Herz (Hz) Signal für das PWM-Verfahren. Die grünen Linien auf Abbildung 12 zeigen ein Zeitintervall von 2 Millisekunden. Die Länge des Rechtecksignals kann in einen Prozentwert umgerechnet werden, um ein Analogsignal abzubilden.

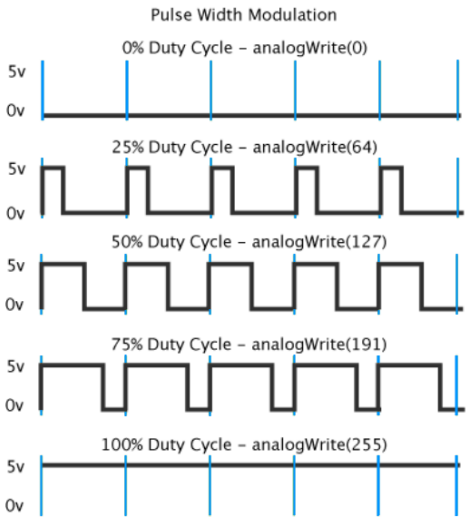

Abbildung 12: PWM-Modulation bei einem Arduino UNO in Anlehnung an (Arduino, 2016)

Da der ATmega 328 Chip lediglich 31,5 Kilobyte (kb) Speicher für Programme besitzt, sind die Anwendungsgebiete auf einfache und kurze Programme beschränkt. Eine weitere Einschränkung ist die RISC-Architektur, welche nur sehr wenige und einfache Befehlssätze unterstützt. (Barrett, 2010) Für einfache Aufgaben, wie das Automatisieren in der Highspeed-Fotografie, ist dies ausreichend. Ein eigenes Betriebssystem, welches mit einer Touch-GUI gesteuert werden kann, ist mit dieser Speichermenge aber nicht realisierbar. Um dies dennoch zu bewerkstelligen müsste viel Code auf eine SD-Karte ausgelagert werden. Dies würde eine Abkehr von einem reinen Mikrocontroller hin zu einem Computer bedeuten. Damit würden dann auch die Vorteile des Mikrocontrollers verschwinden. (Odendahl, Finn, & Wenger, 2009) (Kappel, 2016)

3.2.3 Erweiterungsmöglichkeiten

Wenn der Arduino UNO nicht genug Möglichkeiten bietet, um alle Anforderungen an ihn zu erfüllen, kann er auch erweitert werden. Damit auch komplexere Aufgaben gelöst werden können gibt es zwei Vorgehensweisen, die sich bisher weitestgehend durchgesetzt haben. (Odendahl, Finn, & Wenger, 2009)

Es kann ein alternatives Arduino Board mit einem speziellen Design verwendet werden, das die Anforderungen besser abbilden kann. Auf dem Markt ist eine Vielzahl von alternativen Arduino Boards erhältlich. Die Spannweite erstreckt sich von einfachen alternativen Layouts, bis hin zu Boards mit anderen IC-Chips, die für spezielle Anforderungen entwickelt wurden. (Odendahl, Finn, & Wenger, 2009)

Ein alternatives Design ist zum Beispiel der Arduino Mega. Er nutzt einen ATmega 1280 IC-Chip anstatt des ATmega 328, der auf dem Arduino UNO verbaut ist. Das Mega Board bietet sich vor allem dann an, wenn das UNO Board nicht genug GPIO-Pins bietet, um alle Aktoren zu steuern. Insgesamt hat es 54 GPIO-Pins, von denen 15 in der Lage sind PWM-Signale zu verarbeiten. (Arduino, 2016)

Alternativ zum Wechsel des Arduino Boards kann auch der Arduino UNO, durch die Verwendung von sogenannten Shields, erweitert werden. Ein Shield ist ein eigenständiges Board, das die Funktionalität des Arduino ergänzt. Ein Beispiel dafür ist das Relaise Shield, auf Abbildung 13. (Odendahl, Finn, & Wenger, 2009)

Dieses Shield passt genau auf einen Arduino UNO und stellt 4 Relaise zur Verfügung. Der Vorteil diese Shields ist, dass es genau auf einen Arduino passt. Sobalt dieses Board aufgesteckt wurde ist keine zusätzliche Verkabelung notwendig um die Relaise in Betrieb zu nehmen. In der Bedienungsanleitung des Shields ist angegeben welche Pins zum Schalten der Relaise genutzt werden müssen. (Arduino S.R.L., 2016)

Abbildung 13: Arduino UNO mit einem Relaise Shield.

Zu dem hier vorgestellten Battery Shield gibt es noch unzählige weitere Shields für den Arduino, unter anderem ein Music, MiKro SD, GPS, Touch, Ethernet und Wireless Shield, um nur einige zu nennen. (Odendahl, Finn, & Wenger, 2009)

3.2.4 Echtzeitfähigkeit

Normalerweise führen Computer ihre Befehle möglichst schnell aus. Der Begriff Echtzeit lässt zwar die Vermutung zu, dass eine Anwendung möglichst schnell laufen soll, dies ist aber nicht richtig. Echtzeit bedeutet, wie bereits in Kapitel 3.1 beschrieben, dass eine Anwendung garantieren kann, innerhalb eines definierten Zeitfensters zu reagieren und die angeforderten Daten und Aktionen innerhalb eines definierten Zeitfensters zu liefern. Um diese Anforderung zu erfüllen, muss der Mikrocontroller in der Lage sein möglichst gleichmäßig und genau zu reagieren. Damit die notwendige Genauigkeit gegeben ist, muss der Mikrocontroller in der Lage sein, gleichmäßige Pulse zu generieren. Die Schlagzahl des Pulses gibt an wie genau auf ein Ereignis reagiert werden kann. Der Arduino nutzt dafür einen oszillierenden 16 Mhz Quarzkristall, der die Taktung übernimmt. (The University of North Carolina at Chapel Hill, 2006) (M. Kuo & H. Lee, 2001)

Der Kristall gibt jede Sekunde 16 Millionen Pulse ab. Damit beträgt die kleinste Zeiteinheit für das Board 1/16000000 Sekunde. Dies ist gleichzeitig die kleinste theoretisch machbare getaktete Zeiteinheit und speziell für Befehle wichtig, bei denen nicht nur die Schnelligkeit, sondern auch die Genauigkeit zählt. Im Bereich der Highspeed-Fotografie ist dies für Wartebefehle (Delays), und auch für das PWM-Verfahren, das in Kapitel 3.2.2 erklärt wurde, sehr wichtig. (M. Kuo & H. Lee, 2001) (Ras, 2016)

Wie bereits aufgezeigt wurde kann der Arduino so programmiert werden, dass er auf eine Mikrosekunde genau reagiert. Es gibt eine Einschränkung bei der Benutzung der Mikrosekunden-genauen Delay-Anweisung. Diese arbeitet nur dann akkurat wenn das Delay nicht länger als 16383 Mikrosekunden dauert. Jede größere Zahl kann kein akkurates Delay erzeugen. Auf der offiziellen Seite von Arduino wird darauf verwiesen, dass es bei größeren Delays notwendig ist Millisekunden-genaue Delays auszulösen. Um dennoch Mikrosekunden genau zu arbeiten, kann die *Delay() Anweisung genutzt werden um auf eine Millisekunde genau eine Verögerung einzubauen* und danach kann *DelayMicroseconds()* genutzt werden um die noch fehlenden Mikrosekunden anzuhängen. (Arduino, 2016)

3.3 Beschreibung des Raspberry Pi

Der Raspberry Pi verfügt, im Vergleich zum Arduino Board, eine wesentlich stärkere Hardware. Dies beginnt bei einer Mehrkern-CPU, erstreckt sich über zahlreiche weitere Features wie ein spezieller Displayanschluss und ein Netzwerkinterface, bis hin zu 40 GPIO-Pins. Der Grund für die relativ starke Hardware liegt darin, dass der Haupteinsatzzweck eines Raspberry Pi nicht in der Ansteuerung von einfachen Aktoren liegt, sondern darin ein möglichst günstiger, aber dennoch vollwertiger Computer zu sein. Da es bei der Steuerung von Aktoren nicht rein auf die Leistungsfähigkeit eines Systems ankommt, sondern vor allem darauf wie die einzelnen Komponenten miteinander kommunizieren, hat der Raspberry Pi erhebliche Probleme damit die gleichen Anwendungsfälle annähernd gleich gut wie ein Arduino abzuarbeiten. (Kofler, Kühnast, & Scherbeck, 2016) (Kuehnel, 2015)

Der große Vorteil des Raspberry Pi liegt aber darin, dass das System mit einem vollwertigen Betriebssystem ausgestattet ist, dessen Möglichkeiten erst die Erstellung einer speziellen Software zum einfachen Erstellen und Verteilen von Fotografie-Settings und Ideen ermöglichen. So ist schon das Anbinden eines Touch-Displays mit einem Arduino erheblich aufwendiger. Die technischen Einschränkungen bezüglich der GPIO-Ports machen es aber unmöglich alle benötigten Komponenten, wie die Aktoren, das Display, externen Speicher usw., zeitgleich zu betreiben. (Kofler, Kühnast, & Scherbeck, 2016) (Kuehnel, 2015)

3.3.1 Hardware

Für die Tests wird ein Raspberry Pi V3 Model B eingesetzt. Dieses Modell besitzt 1GB RAM und einen Broadcom BCM2837 Quadcore Prozessor. Weiter besitzt er über einen eigenen Anschluss für das Display, wodurch keine GPIO-Pins belegt werden, wie es bei einem Arduino der Fall ist.

Insgesamt besitzt der Raspberry Pi über 40 frei programmierbare GPIO-Pins. (Raspberry Pi Foundation, 2016)

3.3.2 Einschränkungen

Wie auch beim Arduino haben einige der GPIO-Pins eine festgelegte Belegung, die nicht geändert werden kann, wodurch weniger als die angegebenen 40 Pins für Aktoren zur Verfügung stehen. Die nachfolgende Abbildung 14 zeigt den GPIO-Anschluss des Raspberry Pi. Es ist darauf ersichtlich, dass lediglich 26 der 40 Pins für die freie Programmierung zur Verfügung stehen. Die restlichen Pins sind für Strom, Erdung und ID EEPROM reserviert. (Ras, 2016)

Da für die Testfälle lediglich wenige Pins benötigt werden, ist die Anzahl der verfügbaren Pins kein Ausschlussargument für den Raspberry Pi. Weiter ist es möglich die Anzahl der Pins mit einem MUX-Board zu vervielfältigen. Ein MUX-Board bewerkstelligt, dass Vervielfältigen durch die Kombination von verschiedenen Pins. Da ein digitaler Pin 2 verschiedene Stati hat, können mit den digitalen Pins 2^n verschiedene virtuelle Pins abgebildet werden. Damit die Schaltung der einzelnen Pins nicht manuell erfolgen muss, gibt es Mux-Boards. Ein Mux-Board mit 8 Pins hat die Möglichkeit 2^8 virtuelle Pins zu erzeugen. Der Nutzer selbst kann einfach 256 verschiedene Pins auf dem Mux-Board belegen und dieses wandelt die Signale in eine Schaltkombination für 8 Pins auf dem Raspberry Pi um. (Kofler, Kühnast, & Scherbeck, 2016)

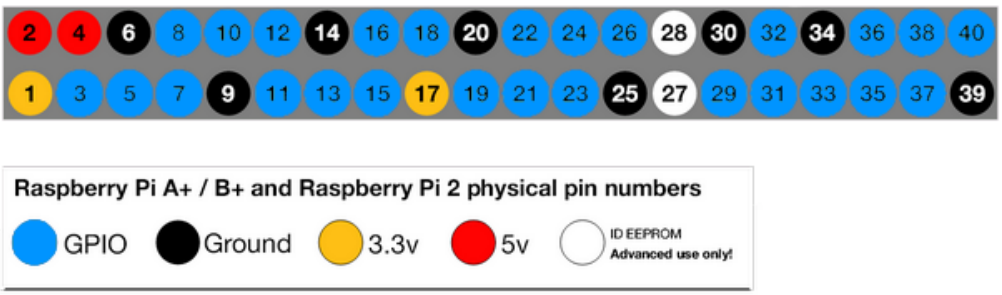

Abbildung 14: Pin-Belegung des GPIO-Ports auf dem Raspberry Pi in Anlehnung an (Raspberry Pi Foundation, 2016).

Alle GPIO-Pins auf dem Raspberry Pi sind nur für 3,3 V geeignet und besitzen keine Fähigkeit 5 V zu verarbeiten. Falls ein 5 V Signal auf einem GPIO-Pin anliegt, hat dies zur Folge, dass der Raspberry Pi Schaden nimmt. Aus diesem Grund müssen alle Tests so ausgelegt werden, dass sie nur die 3,3 V Source nutzen. (Raspberry Pi Foundation, 2016) Weiter ist darauf zu achten, dass alle Teile der Schaltung, die eine Höhere Spannung nutzen, mittels eines Optokopplers vom Stromkreis des Raspberry Pi's getrennt sind, damit Schäden am Gerät vermieden werden.

Der Raspberry Pi besitzt auch über PWM-fähige Pins. 2 Pins sind hochauflösend und die restlichen GPIO-Pins besitzen eine PWM-Auflösung, die ähnlich der des Arduino Ports ist. Somit ist jedes GPIO-Pin auf dem Raspberry Pi in der Lage 256 verschiedene Signalabstufungen zu unterscheiden. (Raspberry Pi Foundation, 2016) (Arduino, 2016)

3.3.3 Erweiterungsmöglichkeiten

Für den Raspberry Pi sind eine Vielzahl von Erweiterungsmodulen verfügbar. Es gibt für fast jeden Anwendungsfall bereits spezielle Erweiterungsboards. Für die angestrebte Anwendung im Bereich der Highspeed-Fotografie ist aber bereits die vorhandene Ausstattung völlig ausreichend. Es kann der Fall eintreten, dass der Raspberry Pi weit schlechter auf Interrupts und Timings reagiert als der Arduino. Der Grund dafür kann der 1 Mhz Taktgeber sein, wenn er zu ungenau arbeitet. Sollte dies der Fall sein, muss ein RTC-Erweiterungsboard für die Tests genutzt werden. (Raspberry Pi Foundation, 2016)

3.3.4 Betriebssystem

Für den Raspberry Pi gibt es zahlreiche Betriebssysteme. Die meisten von ihnen nutzen Standardmäßig keinen Echtzeitfähigen Scheduler. Dazu zählen alle Linuxderivate und Windows wodurch sie auch keine Real-Time-Operating-Systems sind (RTOS). (Raspberry Pi Foundation, 2016) (Ras, 2016)

Um ein Linux-System echtzeitfähig zu bekommen, ist es notwendig den Kernel auf RT-Preemption zu patchen. Es gibt zahlreiche Linux-Projekte, die auf dem RT-Patch aufbauen. Eines dieser Projekte, die auch ein fertiges Image zum Download anbieten, ist EMLID. Damit die Tests mit einem solchen System durchgeführt werden können, ist es entweder notwendig selbst im Vorfeld ein RT-Image zu erstellen oder ein fertiges Image eines solchen Projekts herunter zu laden. Als Basis für ein solches System dient in den meisten Fällen das Raspbian Image von der Raspberry Pi Homepage. (EMLID LIMITED, 2016) (Raspberry Pi Foundation, 2016)

Neben den zahlreichen Linux-Systemen sind auch spezielle RTOS-Systeme verfügbar. Für die Tests mit dem Raspberry Pi wird ChibiOS herangezogen. Es ist ein eigenständiges Unix-Style Betriebssystem, welches 1989 begonnen wurde. Der Vorteil von ChibiOS/RT liegt darin, dass es von Grund auf als RTOS konzipiert wurde. Die aktuelle Version ist voll präemptiv. (Di Sirio, 2016)

3.3.5 Echtzeit und Grenzen

Um echtzeitfähig zu sein, muss ein System in der Lage sein, schnell und genau auf Ereignisse zu reagieren. Wie bereits in Kapitel 3.3.1 dargelegt wurde, besitzt der Raspberry Pi keinen Quarzkristall als Taktgeber, um regelmäßige Takte zu erzeugen. Dadurch kann er nicht gezielt und genau reagieren. Es gibt nur die Möglichkeit eine Anforderung möglichst schnell abzuarbeiten oder mit einer elektronisch erzeugten Taktung zu arbeiten. Für die elektronische Erzeugung von Takten besitzt er einen im Prozessor sitzenden 1 Mhz Taktgeber. Dieser ist mit 1 Millionen Takten pro Sekunde um ein vielfaches langsamer und damit ungenauer als jener des Arduino Boards. Um hier mit dem Arduino Board gleichzuziehen, benötigt der Raspberry Pi ein eigenes Real-Time-Clock (RTC) Board. (Raspberry Pi Foundation, 2016)

Ein weiteres Problem bei der Umsetzung eines echtzeitfähigen System ist das Betriebssystem. Derzeit sind Linux-Derivate auf dem Raspberry Pi die am weitesten verbreiteten Betriebssysteme.

Für die Tests wird ein Raspbian genutzt. Dieses Betriebssystem basiert auf Debian und ist das offiziell empfohlene Betriebssystem für den Raspberry Pi. (Raspberry Pi Foundation, 2016)

Die Verteilung der Arbeitszeit des Prozessors erfolgt über einen so genannten Scheduler. Alle laufenden Anwendungen können sich bei diesem einreihen und auf die Zuteilung von Prozessorzeit warten. Der Scheduler garantiert aber keine maximale Wartezeit, bis Prozessorzeit für einen Prozess zur Verfügung steht. Sehr oft wird für einen Scheduler ist das Round-Robin Verfahren eingesetzt. Dabei wird einem Prozess ein Zeitfenster zur Verfügung gestellt in dem er den Prozessor nutzen kann. Nach dem Ablauf der Zeit wird der Prozess pausiert und wieder hintern in der Warteschlange eingereiht. Der Prozess kann keine Tasks verarbeiten während er keine Prozessorzeit hat. (Silberschatz, Galvin, & Gagne, 2005) (Tanenbaum & Bos, 2014) (Chang, 1987)

Die Zeit, die ein einzelner Prozess warten muss, hängt maßgeblich von der Anzahl der ausgeführten Programme ab. Aus diesem Grund kann es sein, dass das Testprogramm für die Durchführung der Tests beim ersten Durchlauf der Testreihe immer sehr zeitnah reagiert und alle Tests erfolgreich absolviert werden und bei einem weiteren Durchlauf kein einziger Test erfolgreich ist, da der Prozess nie zur richtigen Zeit seinen Timeslot für den Prozessor hat. (Silberschatz, Galvin, & Gagne, 2005)

Wird im Programm ein *Delay(250)* eingebaut, weil 250 Millisekunden gewartet werden muss, so kann beim Standard-Scheduler nicht garantiert werden, dass das Programm nach 250 Millisekunden weiter läuft. Der Grund dafür liegt in der Funktionsweise des Schedulers. Dieser rotiert in einer Schleife über alle Prozesse und teilt ihnen nacheinander Prozessorzeit zu. Wenn ein Programm ein *Delay()* durchführt, wird es ausgelassen, bis die Zeit für das *Delay()* abgelaufen ist. Danach wird es wieder in den Scheduler eingereiht und bekommt nach der neuen Einreihung Prozessorzeit zugewiesen. (Tanenbaum & Bos, 2014)

Durch dieses Verhalten ist es nicht möglich vorherzusagen wann ein Programm reagiert. In einem normalen Anwendungsfall fällt dieses Verhalten nicht auf, da es egal ist, ob ein Programm nach 10 ms oder erst nach 300 ms reagiert. Im Fall der Highspeed-Fotografie kann dieser Unterschied jedoch der Grund sein warum ein Bild verfehlt wird. (Tanenbaum & Bos, 2014)

3.4 Wichtige Komponenten für die Tests

In diesem Kapitel sind alle für die Tests wichtigen Hardwarekomponenten aufgeführt und erklärt. Es kommen, von einfachen einzelnen bis zu fertigen Schaltkreisen, verschiedene Bauteile zum Einsatz. Zusätzlich zu den elektronischen Bauteilen, die für die Tests selbst notwendig sind, werden weitere Teile benötigt, wie Blitzgeräte, Funkauslöser und Adapter, um die Tests durchzuführen.

3.4.1 Bread-Board

Das Bread-Board ist ein so genanntes Entwicklungsboard. Wie in Abbildung 15 zu sehen ist, handelt es sich dabei um ein Kunststoffboard mit zahlreichen Löchern. Die einzelnen Löcher sind

mit Metallschienen miteinander verbunden und können genutzt werden, um elektronische Bauteile aufzustecken. Die Verbindungen der einzelnen Löcher sind so gestaltet, dass es mit diesem Board möglich ist komplexe Schaltungen zu erstellen und zu testen. In einem späteren Entwicklungszyklus kann dieses Board gegen eine geätzte Platine getauscht werden. Abbildung 15 zeigt schematisch wie die Metallschienen im Board verlegt sind. Es ist zu erkennen, dass das Aufsetzen eines Mikrocontrollers genau auf der Mittelschiene dazu führt, dass jeder Pin auf einer einzelnen Metallschiene liegt. Dadurch ist es möglich auch komplexe Schaltungen zu erstellen.

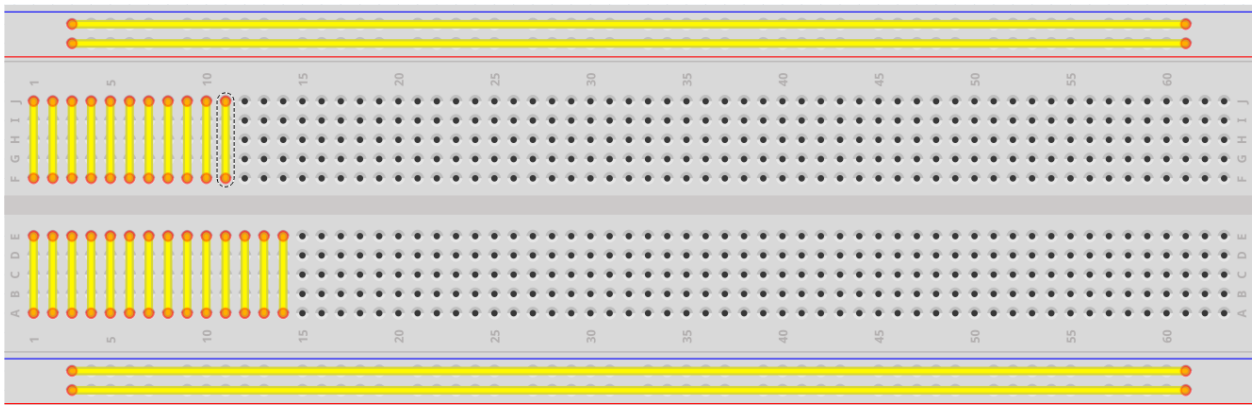

Abbildung 15: Verlegeschema auf einem Entwicklungsboard

3.4.2 Widerstand

Eine in elektronischen Schaltungen sehr oft verwendete Komponente ist der in Abbildung 16 zu sehende Widerstand. Er dient dazu die Spannung zu regulieren. Wenn die eingespeiste Spannung vor einem anderen Bauteil zu groß ist, kann ein Widerstand genutzt werden, um die Spannung soweit zu reduzieren, damit das Bauteil nicht beschädigt wird. Eine weitere Einsatzmöglichkeit für einen Widerstand ist das direkte Ableiten von plötzlichen Spannungsschwankungen aus dem negativen Pol. Dazu wird der Widerstand mit einem Connector vor dem zu schützenden Bauteil angeschlossen und mit dem zweiten Anschluss direkt auf den negativen Pol der Stromquelle. (Schnabel, 2007)

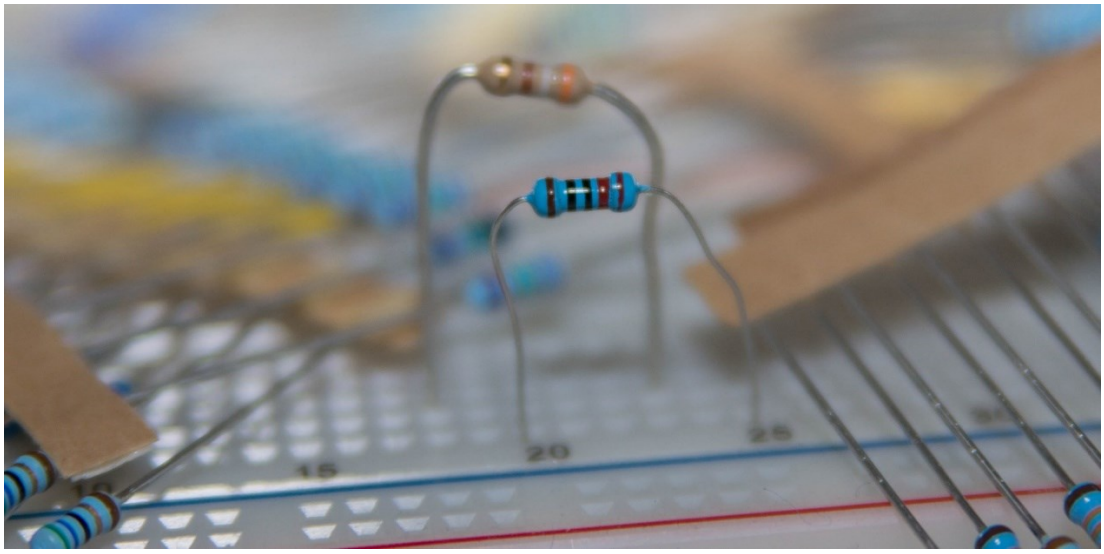

Abbildung 16: Widerstand

Bei diesen Testfällen werden Kohleschichtwiderstände eingesetzt. Dies ist, einfach ausgedrückt, ein kleines Röhrchen aus einem nichtleitenden Material. Auf dieses Trägermaterial wird vollflächig oder spiralförmig ein schlecht leitendes Material, in unserem Fall Kohle, aufgebracht, an dessen Enden 2 Anschlüsse angebracht sind. (Schnabel, 2007)

3.4.3 Optokoppler

Ein Optokoppler ist ein Baustein, der dazu genutzt wird, zwei elektrische Stromkreise galvanisch voneinander zu trennen. Auch dieser Baustein dient dem Schutz der Schaltung und der verwendeten Komponenten. Abbildung 17 zeigt den bei diesen Schaltungen eingesetzten Optokoppler des Typs 4N35. Er wird zwischen allen Komponenten geschalten, die über eine eigene Stromversorgung verfügen. (Schnabel, 2007)

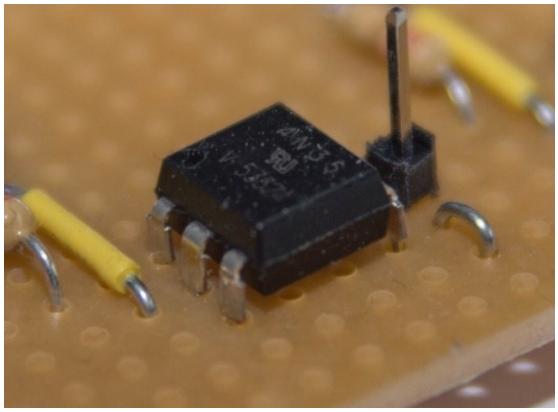

Abbildung 17: Optokoppler

Nachfolgend wird auf Abbildung 18 das Schaltzeichen eines Optokopplers dargestellt, anhand dessen seine Funktionsweise erläutert wird. Pin 1 ist rechts oben auf dem Schaltbild. Dieser bildet zusammen mit Pin 2 die Anode und die Kathode für eine Leuchtdiode. Wenn auf diesen 2 Pins ein geschlossener Stromkreis angelegt wird, leuchtet die Diode. Das Licht wird von einem gegenüberliegenden Empfänger, einer sogenannten Fotodiode, aufgefangen. Dadurch kann ein Signal, das an Pin 1 und 2 angelegt wird, auf die Pins 5 und 6 übertragen werden, ohne dass eine elektrische Verbindung besteht. Das Trennen von Schaltkreisen mit eigenem Stromkreis ist notwendig, damit es zu keinen fehlgeleiteten Stromflüssen kommen kann, die das System beschädigen. Mit so einem Bauteil ist es möglich, dass Signale von einem Raspberry Pi oder Arduino, die mit 3,3 oder 5 Volt (v) laufen, auf ein Gerät übertragen wird, das mit 12 V (oder mehr) läuft. Wenn dieses Bauteil zwischengeschalten wird, kann es nicht passieren, dass die 12 V durch einen Fehler ein Bauteil, wie den Arduino, zerstören. (Schnabel, 2007)

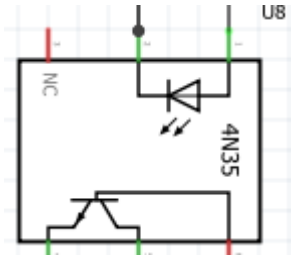

Abbildung 18: Schaltbild eines Optokopplers in Anlehnung an (Schnabel, 2007).

3.4.4 Schaltrelais

Wie bereits erwähnt wurde, stellen der Arduino und der Raspberry Pi lediglich 3,3 V und 5 V als Spannung, für den Betrieb von anderen Geräten, zur Verfügung. Für die Testfälle werden Magnetventile gebraucht, um den Fluss der Tropfflüssigkeit zu steuern. Diese Ventile sind ab einer Schaltspannung von 12 V zu bekommen. Um sie zu öffnen, wird ein Relais gebraucht, welches mit 12 Volt umgehen kann. Abbildung 19 zeigt das für diese Testversuche eingesetzt Relais. Es ist ein 12 V Schaltrelais, welches bis zu 250 V durchschalten kann.

Zwischen den linken Anschlüssen auf dem Bild und dem Relais weiter rechts ist ein kleines weißes Bauteil zu erkennen. Dabei handelt es sich um einen Optokoppler, der den Stromkreis bei den Anschlüssen auf der linken Seite von denen auf der rechten Seite trennt.

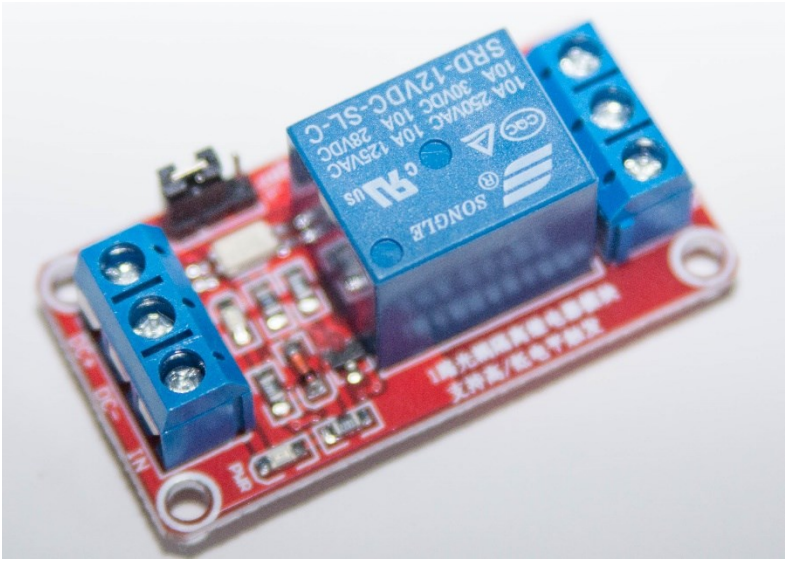

Abbildung 19:Schaltrelaise mit eingebautem Optokoppler.

Die Belegung der Pins auf der linken Seite von oben nach unten ist:

1. DC+ (positiver Pol der Schaltspannung, 12 V)
2. DC- (negativer Pol der Schaltspannung 12 V)
3. IN (Signal, welches das Schalten des Relais steuert)

Die Pins auf der rechten Seite sind von oben nach unten wie folgt belegt:

1. NC (normal closed)
2. COM (Signal)
3. NO (normal open)

COM wird immer mit dem Ausgangssignal verbunden. Als zweites wird NC oder NO genutzt. Diese beiden Pins geben an, ob der Stromkreis auf der rechten Seite im nicht geschalteten Normalzustand offen oder geschlossen ist.

3.4.5 Blitzgerät

Für die Tests werden vier Blitzgeräte der Firma Yongnuo benutzt. Sie haben die Typenbezeichnung Speedlite YN560 Mark III. Diese Blitzgeräte besitzen einen ISO-Schuh, wodurch sie sich für alle Kameras der Marken Canon, Nikon, Pentax und für die neuen Sony Kameras eignen. Sie sind als manuelle Blitze konzipiert und benötigen deswegen nur das Auslösesignal der Kamera. Es findet ansonsten kein Informationsaustausch zwischen Kamera und Blitz statt. Aus diesem Grund ist es auch nicht notwendig für jedes Kameramodell einen eigenen Blitz zu verwenden. (Yongnuo, 2014)

Der Blitz besitzt einen Zoommotor für die Brennweitenverstellung. Da aber, wie bereits erwähnt, kein Informationsaustausch bezüglich der Brennweite erfolgt, muss diese Information manuell eingegeben werden. Durch das Einstellen der Blitzbrennweite kann bis zu einer Blendenstufe Licht gewonnen werden, ohne dass die Blitzstärke erhöht werden muss. (Yongnuo, 2014)

Die YN560 Mark III Blitzgeräte werden mit einem Standfuß ausgeliefert und besitzt einen integrierten Funkempfänger. Deswegen bieten sie ideale Voraussetzungen zum entfesselten Blitzen, was bei mehreren Blitzgeräten unbedingt notwendig ist. Entfesseltes Blitzen wird in der Fotografie das Blitzen mit einem Blitzgerät genannt, welches nicht auf der Kamera montiert ist. Dies ist immer dann der Fall, wenn der Fotograf das Bild durch eine spezielle Belichtung künstlerisch gestalten will. Dies kann das Blitzen gegen die Kamera sein oder die Verwendung von mehreren Blitzgeräten, welche ein Objekt von verschiedenen Seiten und zu verschiedenen Augenblicken beleuchten. (Yongnuo, 2014) (Jorns, 2012)

Zusätzlich zur mitgelieferten Ausstattung, die dieses Blitzgerät für entfesseltes Blitzen ideal macht, hat es eine Brenndauer t0.1 von 1/20000 Sekunde bei einer Stärke von 1/128. Blitzgeräte, die eine kürzere t0.1 Brenndauer haben, sind nur sehr schwer zu finden. Zum einen werden oftmals t0.5 Werte angegeben, welche zwar augenscheinlich die bessere Wahl darstellen, aber nicht aussagen, ob sie auch wirklich kürzer leuchten. Zum anderen wird die Abbrenndauer oftmals sogar ganz verschwiegen. Aus diesem Grund eignen sich die verwendeten Yongnuo-Geräte sehr gut für Versuche in der Highspeed-Fotografie. (Yongnuo, 2014) (Peterson, 2011)

3.4.6 Funkauslöser

Die Blitzgeräte können entweder durch einen Funkauslöser oder durch ihr PC-Syncport ausgelöst werden. Das PC-Syncport ist ein spezieller Stecker, der auf den Blitzgeräten angebracht ist. Wenn die Blitzgeräte dadurch ausgelöst werden sollen, muss von der Schaltung zu jedem Blitzgerät ein eigenes Kabel gelegt werden, wodurch es zu Einschränkungen bei der Positionierung der Blitzgeräte kommt. Diese Einschränkungen ergeben sich aus der Kabellänge und der Signalstärke, die notwendig ist, um das Signal zu übertragen. Ein klarer Vorteil beim kabelgeführten Auslösen ist, dass jeder Blitz extra ausgelöst werden kann, falls dies für ein spezielles Setting notwendig ist. (Yongnuo, 2014)

Ein Funkauslöser kann die Blitzgeräte, unabhängig vom Ort ihrer Montage, auslösen. Über verschiedene Kanaleinstellungen können Blitze auch einzeln gesteuert werden, jedoch sind dafür

mehrere Auslöser notwendig. Für diese Tests wurde ein Yongnuo RF603C II Blitzauslöser gekauft. Dieser Blitzauslöser ist sehr günstig in der Anschaffung und arbeitet perfekt mit den verwendeten Blitzgeräten zusammen. Weiter werden diese Auslöser paarweise verkauft. Dadurch ist es möglich direkt 2 verschiedene Auslöserkonfigurationen zu nutzen. Der Funkauslöser benutzt, gleich wie die Blitzgeräte, einen ISO-Blitzschuh, welcher nur für neue Sony-Geräte geeignet ist und nicht für die hier genutzten Kameras, die einen Minolta-Blitzschuh besitzen. (Yongnuo, 2014) (Sony, 2011) (Sony, 2011)

Das Blitzgerät besitzt auf der Oberseite noch einen Blitzschuh, damit auf der Kamera zusätzlich zum Funkauslöser auch noch ein Blitzgerät montiert werden kann. Der Funkauslöser besitzt in weiterer Folge einen PC-Sync-Anschluss, der aber lediglich als Ausgang dient, um Blitze auszulösen. Ein eingehendes Signal wird nicht verarbeitet. Damit ist es nicht möglich den Funkauslöser über diesen Anschluss zu steuern. (Yongnuo, 2014)

Damit der Funkauslöser ein Signal sendet, gibt es 2 Möglichkeiten: das Signal zum Auslösen kann über den unteren Blitzschuh kommen oder ein Knopf auf der Oberseite des Auslösers muss gedrückt werden. Wenn der Auslöser zerlegt wird, kann dort zum Beispiel ein Kabel angeschlossen werden, welches mit den Steuergeräten verbunden wird. (Nimmervoll, 2014) (Yongnuo, 2014)

3.4.7 Blitzadapter

Da, wie bereits erwähnt, die genutzten Kameras keinen ISO-Blitzschuh verwenden, ist ein Adapter notwendig, um diese Blitzgeräte zu nutzen. Es wurde ein Pixel TF-325 Adapter gekauft. Dieser Adapter besitzt auf der Unterseite einen Minolta-Blitzschuh und auf der Oberseite einen ISO-Blitzschuh. Auf der Vorderseite ist ein PC-Sync-Anschluss angebracht, der eingehende Signale verarbeitet und auf den ISO-Blitzschuh weitergibt. (Pixel, 2010)

Dieser Adapter soll benutzt werden, um direkt am Funkauslöser montiert, das Signal von den Steuergeräten an den Funkauslöser weiterzugeben, damit die Blitzgeräte auslösen.

3.4.8 Mariotte Flasche

Die Mariotte Flasche dient dazu einen gleichmäßigen Druck zu erzeugen. Diese Flasche wird bei den Tests als Flüssigkeitsbehälter eingesetzt. Sie dient dazu den Druck, der auf den Ventilen anliegt, einzustellen und konstant zu halten. Der Druck ist dabei unabhängig vom genauen Flüssigkeitsstand in der Flasche. (Grimvall, 2007)

Abbildung 20 zeigt eine Mariotte Flasche mit seitlichem Auslass. Diese Darstellung wurde gewählt, da die Funktionsweise so leichter erklärbar ist. Bei den Testfällen wird eine Flasche genutzt, bei der sich die Auslassöffnung (D) am Boden der Flasche befindet.

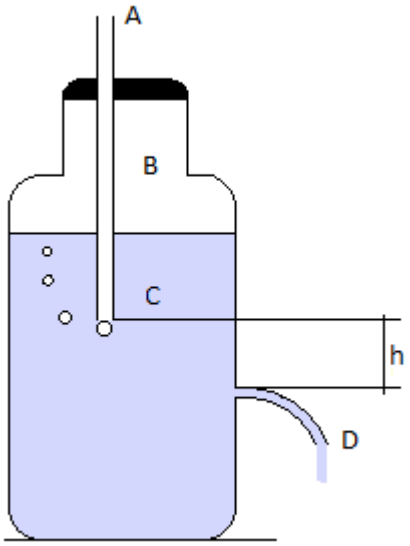

Abbildung 20: Skizze einer Mariotteschen Flasche in Anlehnung an (Grimvall, 2007).

Die in Abbildung 20 gezeigte Flasche besitzt zwei Öffnungen. Eine ganz oben, die mit einem Rohr versehen in die Flüssigkeit hineinsteht (A) und eine zweite, die seitlich an der Flasche hervorsteht (D). Öffnung A dient als Lufteinlass und Öffnung D als Flüssigkeitsauslass. Die Flasche ist mit Ausnahme der zwei Öffnungen luftdicht verschlossen. Beim Lufteinlass liegt ein Druck von einer Atmosphäre an (normaler Umgebungsdruck). Dieser Druck liegt auch am Ende des Lufteinlasses, der unter Wasser steht, an. Da die Flüssigkeit eine höhere Dichte hat als die Luft, würde die Flüssigkeit (C) normalerweise von unten in das Rohr des Lufteinlasses eindringen und das Lufteinlassrohr hätte innen den gleichen Wasserstand wie in der restlichen Flasche. Wenn die Flüssigkeit in das Rohr steigt, sinkt der Flüssigkeitsstand in der Flasche und der eingeschlossene Luftraum (B) wird größer. Da keine Luft von außen in den Luftraum (B) eindringen kann, sinkt der Luftdruck in der Kammer (B). Dies bedeutet, dass Der Druck in der Kammer (B) kleiner wird als der Druck, der am Boden des Lufteinlasses (A) anliegt. Deswegen wird das Wasser im Rohr soweit vom Umgebungsluftdruck in die Flasche zurück gepresst, bis der Druck in der Luftkammer (B) genau eine Atmosphäre beträgt. Das Lufteinlassrohr wird also nicht mit Flüssigkeit geflutet und der Druck am unteren Ende des Rohres beträgt immer genau eine Atmosphäre. (Grimvall, 2007)

Der Druck, der am Auslass (D) anliegt, setzt sich aus dem kumulierten Druck in der Flasche zusammen. Dies ist eine Atmosphäre Druck, der am unteren Ende des Rohres (A) anliegt und der Flüssigkeitsdruck, der sich vom unteren Ende des Einlasses (A) bis zum Auslass (D) aufbaut. Zur Berechnung des Flüssigkeitsdrucks wird die Formel $p(h) = p.g.h$ verwendet.

p ist die Dichte der Flüssigkeit, *g* ist die Erdbeschleunigung und *h* entspricht der Höhe der Flüssigkeit. Im Fall dieser Flasche ist das die Höhe zwischen dem unteren Ende des Lufteinlassrohrs (A) und dem Auslass (D). Da die Dichte der Flüssigkeit und die Erdbeschleunigung nicht verändert werden können, ist die einzige Variable die Höhe. Dies wird auch als hydrostatischer Druck bezeichnet. (Grimvall, 2007)

Die Höhe kann verändert werden, indem das Rohr (A) weiter in die Flasche hineingeschoben oder aus dieser herausgezogen wird. Wenn Flüssigkeit aus dem Auslass (D) austritt, wird Luft durch den Einlass (A) angesaugt und steigt im Inneren des Behälters in die Luftkammer (B) auf. Sobald das Rohr (A) soweit in die Flasche geschoben wird, dass es genau auf der gleichen Höhe ist wie der Auslass (D), tritt keine Flüssigkeit mehr aus dem Auslass (D) aus, da der Druck beim Einlass (A) gleich groß ist wie beim Auslass (D). Wenn das Rohr (A) unter die Höhe des Auslasses (D) hineingeschoben wird, entsteht ein negativer Druck im Inneren der Flasche. Auf das untere Ende des Luftrohres (A) drückt in diesem Fall der Luftdruck, der im Rohr (A) anliegt und als Gegendruck der Luftdruck, der beim Auslass (D) anliegt. Zusätzlich drückt die Flüssigkeit mit der Formel $p(h) = p.g.h$ auf die untere Öffnung (A). Die Flüssigkeit wird wegen dieses Drucks in das Rohr gepresst und Luft strömt beim Auslass (D) ein und steigt in die Luftkammer (B) auf. Dies geschieht so lange, bis ein Druckausgleich erreicht wird. Das ist der Fall, wenn die Flüssigkeit im Rohr die gleiche Höhe erreicht wie Auslass (D). (Grimvall, 2007)

3.4.9 Magnetventil

Für die Tests werden ODE 21JN1R0V23 Magnetventile genutzt. Dies sind die von (Nimmervoll, 2014) empfohlenen Ventile für die Tropfenfotografie. Diese Ventile besitzen eine nicht polarisierte Spule, wodurch es nicht notwendig ist auf eine spezielle Polung beim Anschluss an den Stromkreis zu achten.

Das Ventil besitzt an der engsten Stelle einen Durchmesser von 2,3 mm für den Wasserdurchlass und benötigt keinen Vordruck zum Öffnen. Dies ist sehr wichtig, da nur der hydrostatische Druck der Tropfflüssigkeit als Vordruck zur Verfügung steht. Dieser Druck müsste nicht nur das Ventil öffnen, wenn es einen Vordruck benötigt, sondern ist auch noch für die Geschwindigkeit verantwortlich, mit der die Tropfen versorgt werden. Wird also ein Ventil genutzt, das einen Vordruck benötigt, ist es auch notwendig, dass der Druck vor dem Ventil erhöht wird. Dies könnte beispielsweise durch ein Höherhängen der Flüssigkeitsbehälter oder durch eine Pumpe erreicht werden. Da in diesem Fall aber der Vorteil der Mariotteschen Flasche verloren gehen würde, werden Ventile genutzt, die keinen Vordruck benötigen.

Die Magnetventile sind im stromlosen Zustand geschlossen. Dies bedeutet, dass die Ventile nur dann eine Flüssigkeit durchlassen, wenn Strom anliegt. Aus diesem Grund müssen die Relaise für die Ventile im Normalzustand einen offenen Stromkreis besitzen, da sich ansonsten die Ventile öffnen würden, wenn die Schaltung stromlos ist.

4 THEORETISCHE BESCHREIBUNG EINES ECHTZEITFÄHIGEN RASPBERRY PI/ARDUINO

Wie bereits in Kapitel 3.1 beschrieben, kann ein System nur dann als Echtzeitsystem angesehen werden, wenn es innerhalb gewisser Parameter läuft. Da es nicht möglich ist zu garantieren, dass der Raspberry Pi immer innerhalb gewisser Grenzen bleibt, fällt die Option der harten Echtzeit weg und es wird versucht das Gerät soweit zu optimieren, dass eine weiche Echtzeit, welche in Kapitel 3.1 definiert wurde, erreicht werden kann. Damit dies gelingen kann, muss ein Betriebssystem gewählt werden, welches dem Anspruch der weichen Echtzeit genügt.

Weiter muss ein Betriebssystem genutzt werden, das die Einschränkungen des normalen Schedulers umgeht und sicherstellen kann, dass ein ausgeführtes Programm innerhalb eines möglichst geringen Zeitfensters reagiert. Das System muss dabei nicht zwingend schnell reagieren, sondern möglichst gleichmäßig. Dies bedeutet, dass die maximale Abweichung der Ausführungszeit vom arithmetischen Mittel möglichst gering sein muss. (Jeffay & Zhao, 1996) (Buttazzo, Lipari, Abeni, & Caccamo, 2005)

Um dies zu testen wird mit dem Betriebssystem ein sogenannter Cylic Test durchgeführt. Bei diesem Test wird die niedrigste und die höchste Reaktionszeit der CPU auf ein Event gemessen.

4.1 Der Cyclic Test

Vereinfacht dargestellt misst der Test die Zeit, die gebraucht wird, um auf ein Ereignis zu reagieren. Das Ergebnis eines einzelnen cyclictest durchlaufs ist die Zeit, die ein System gebraucht hat, um auf ein Ereignis zu reagieren. (Rowand, 2013)

Cyclictest wird genutzt um zu testen ob sich das Verhalten eines Linux-Systems bezüglich der Latenz ändert, wenn das RT-Patch update ausgeführt wird. Weiter dient es zur Beurteilung der Echtzeitfähigkeit der verschiedenen genutzten Systeme. Das Betriebssystem mit der geringsten Abweichung zwischen minimaler und maximaler Latenz wird für die Vergleichstests zwischen Arduino und Raspberry Pi herangezogen.

Wie die meisten Programme muss auch cyclictest mit verschiedenen Parametern durchgeführt werden. Das Parameter -l gibt beispielsweise an wie viele Testdurchläufe gestartet werden. Dies bedeutet mit der einfachen Eingabe von cyclictest -l1 kann der Test gestartet werden und liefert bereits ein Ergebnis. Jedoch gibt es noch weitere Parameter, die das Testergebnis beeinflussen und deswegen wichtig sind. (Rowand, 2013) Alle bei diesen Tests benutzten Parameter werden hier vorgestellt. Eine vollständige Auflistung aller Parameter mit ihrer kurzen Beschreibung kann dem Anhang A entnommen werden.

- -l ist die Anzahl der Testdurchläufe. Wie bereits von (Rowand, 2013) festgestellt wurde ist die Genauigkeit des Tests höher, je öfter er durchlaufen wird. So ist die Wahrscheinlichkeit eine hohe Latenz mit dem Test zu beobachten höher wenn er 10 Millionen Mal durchgeführt wird, als wenn der Test nur ein einziges Mal läuft. Aus diesem Grund werden alle folgenden Tests mit dem Parameter -l10000000 gestartet.
- -m gibt an ob Memorylock genutzt werden soll. Das Memorylocking ist sehr wichtig für RT-Anwendungen. Wenn dies nicht aktiviert ist, kann es vorkommen, dass die Applikation einen anderen Speicher zugewiesen bekommt. Der Kernel muss dann den Adressbereich der Applikation neu mappen. Dadurch kommt es zu höheren Latenzzeiten, was auf jeden Fall vermieden werden muss. In der laufenden Anwendung muss sich der Entwickler darum kümmern, beim Cyclictest kann -m als Parameter angegeben werden. (Rowand, 2013)
- -n Es wurde von (Rowand, 2013) aufgezeigt, dass sich durch die Verwendung dieses Parameters die Latenz erheblich verbessert. Aus Gründen der Nachvollziehbarkeit wird für alle Tests dieser Parameter gesetzt.
- -a gibt an welcher Thread auf welcher CPU laufen soll.
- -p gibt die Priorität an, mit der cyclictest läuft. Wie (Rowand, 2013) bereits beschrieben hat, kann es vorkommen, dass eine Anwendung andere Latenzen erfährt als der Test, wenn sie mit unterschiedlichen Prioritäten laufen. Aus diesem Grund wird für cyclictest die gleiche Priorität eingestellt, wie bei der Echtzeitanwendung später auch.
- -h Wenn dieser Parameter gesetzt wird, gibt der Test vor dem Beenden noch die Daten für ein Histogramm mit aus. Dieser Parameter wird genutzt, um die Daten für die Histogramme in dieser Arbeit vorzubereiten.
- -t gibt die Anzahl der Threads pro CPU an. Wenn der Parameter ohne Zahl angegeben wird, wird für jede CPU 1 Thread erzeugt.
- -q Mit diesem Parameter läuft der Test still durch und erst am Ende des Tests wird eine Zusammenfassung ausgegeben.

Von (Rowand, 2013) wurde die folgende Ausführung des Befehls zur Performance-Beurteilung mit diesen Parametern als typisch bezeichnet:

Cyclictest -l100000000 -m -Sp99 -i200 -h400 -q

Für die Durchführung der eigenen Tests wird der Befehl wie in Listing 1 zu sehen ist geändert:

```
cyclictest –l10000000 –m –n –t1 –p99 –h700 -q
```

Listing 1: Befehl für den Latenztest

Die Änderungen am Befehl sollen sicherstellen, dass alle CPU-Kerne des Raspberry Pi getestet werden. Die Änderungen am Histogramm erweitern die X-Achse bis 700 Mikrosekunden. Falls

ein Testfall eine höhere Latenz aufweist, muss der Datensatz händisch zu den Daten des Histogramms hinzugefügt werden.

Die Tests mit den Betriebssystemen werden zuerst im Leerlauf des Betriebssystems durchgeführt, damit eine möglichst niedrige Latenz erreicht werden kann. Anschließend werden die Tests mit voll belasteten CPU-Kernen durchgeführt, um eine möglichst schlechte Latenz zu erreichen.

Um die CPU-Last zu simulieren, wird ein CPU-Benchmark gestartet, der den Prozessor belastet. Dazu wird das Programm sysbench genutzt. Der in Listing 2 beschriebene Befehl wird parallel zu den Latenztests ausgeführt:

```
sysbench --test=cpu --cpu-max-prime=20000 run
```

Listing 2: Befehl für das Simulieren der CPU-Last.

4.2 Die Vorauswahl der Betriebssysteme für den Raspberry Pi

Im Internet ist eine große Anzahl von Betriebssystemen erhältlich. Diese lassen sich grob in 3 Kategorien einteilen:

1. Normale Linux Distributionen mit einem RT-Patch. Dabei wird der Linux-Kern voll präemptiv. Ein Vertreter dieser Distributionen ist EMLID, ein Raspberry Pi Build, der auf dem Raspbian wheezy vom 16.02.2015 basiert. (EMLID LIMITED, 2016)
2. POSIX-Systeme: Dies sind eigenständige Betriebssysteme, welche sich am POSIX-Standard orientieren, der von der IEEE spezifiziert wurde. Dies bedeutet, dass es zwar eigenständige Betriebssysteme sind, es aber sehr leicht ist vorhandene Linux-Software auf diese Systeme zu portieren, da sie sich an einen gemeinsamen API-Standard mit Linux halten. (IEEE, 2016)
3. Eigenständige Betriebssysteme: Diese halten sich nicht an den POSIX-Standard, weswegen es oft nur schwer möglich ist benötigte Software zu portieren.

Um einen Vergleichswert zu erhalten, wird der Test als Erstes mit dem Standard OS, dem Raspbian, durchgeführt. Die Werte aus diesem Test dienen dem Vergleich wie sich die Verbesserungen auf den Test auswirken. Als zweites Betriebssystem wird ein Raspbian mit dem RT-Patch herangezogen, um aufzuzeigen in wie weit ein normales Linux angepasst werden kann und ob es notwendig ist auf ein proprietäres System zu setzen. Aus der Riege der POSIX-Systeme wird Xenomai mit dem Cobalt Core getestet (Xenomai, 2016). Als ganz eigenständiges OS wird ChibiOS herangezogen.

4.3 Test und Auswahl der Betriebssysteme

Auf Abbildung 21 sind die Ergebnisse des Latenztests von einem nicht modifizierten Raspbian System zu sehen. Bei der Durchführung der Tests wurde die CPU-Last nicht künstlich erhöht. Die minimale Reaktionszeit betrug 8 Mikrosekunden, die maximale 429 Mikrosekunden.

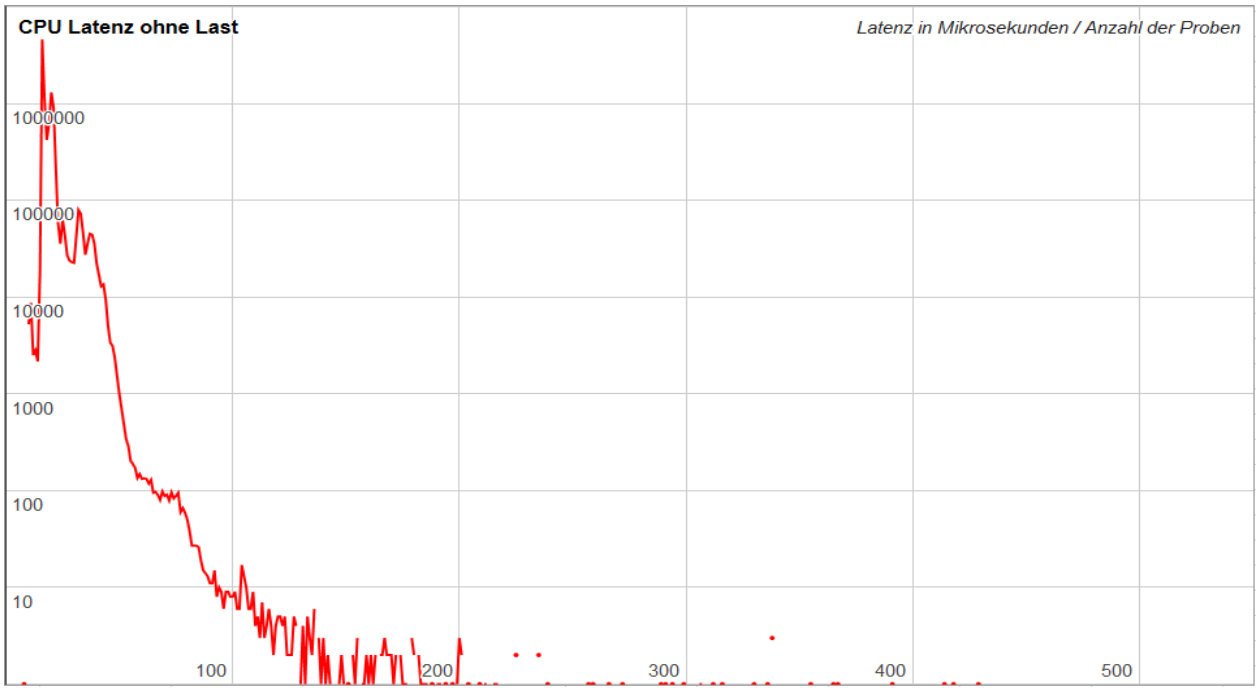

Abbildung 21: CPU-Latenz ohne Last mit einem nicht modifizierten Raspbian.

Die nachfolgende Abbildung 22 zeigt die Ergebnisse des Tests für eine nicht modifizierte Raspbian Distribution. Bei diesem Test wurde die CPU-Last mit sysbench durchgehend auf 100% gehalten. Wie auf dem Chart zu erkennen ist, hat sich das Gesamtverhalten nicht verändert und auch die beste Latenz liegt noch bei 8 Mikrosekunden. Die höchste gemessene Latenz hat sich im Vergleich zum lastfreien Test jedoch auf 518 Mikrosekunden verschlechtert.

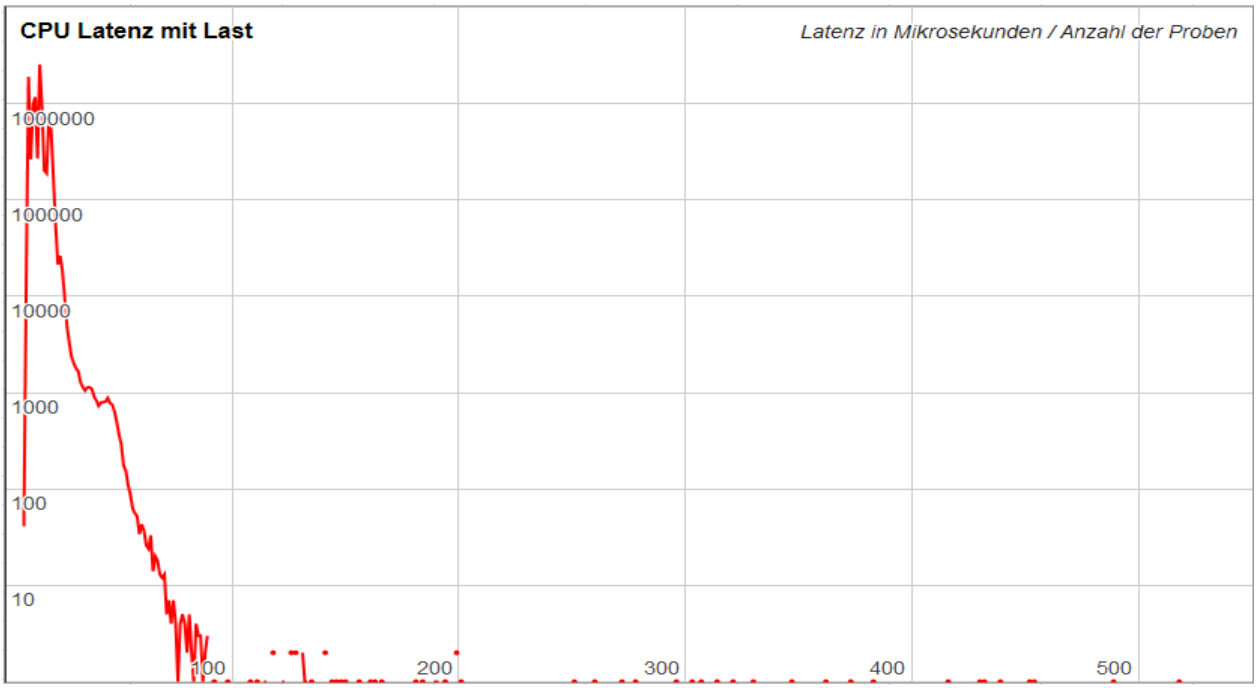

Abbildung 22: CPU-Latenz unter Volllast bei einem nicht modifizierten Raspbian.

Aus diesen zwei Testergebnissen lässt sich errechnen, dass die zu erwartende Latenz eines Systems, das mit einem nicht modifizierten Raspbian läuft, zwischen 8 und 518 Mikrosekunden beträgt. Dadurch beträgt das Zeitfenster für eine Reaktion des Systems 510 Mikrosekunden. Bei einem Testaufbau in der Highspeed-Fotografie kann der Zeitpunkt des Auslösens dadurch um bis zu 510 Mikrosekunden variieren. Dies kann sogar dann noch passieren, wenn alle anderen Komponenten so abgestimmt sind, dass es bei ihnen zu keinen zusätzlichen Abweichungen kommen kann.

Die nachfolgende Abbildung 23 zeigt die Ergebnisse des Tests, die mit EMLID durchgeführt wurden. EMLID ist, wie bereits weiter oben ausgeführt wurde, ein Raspbian Betriebssystem, das durch einen Kernelpatch auf RT-Preemption umgestellt wurde. Der Test wurde, wie bereits der erste Test, dessen Ergebnisse in Abbildung 21 angeführt sind, ohne eine künstlich erzeugte CPU-Last durchgeführt. Wie die Grafik erkennen lässt, ist eine starke Verbesserung zur äquivalenten Abbildung 21 zu erkennen. Die minimale Latenz betrug 10 Mikrosekunden und die maximale 58 Mikrosekunden.

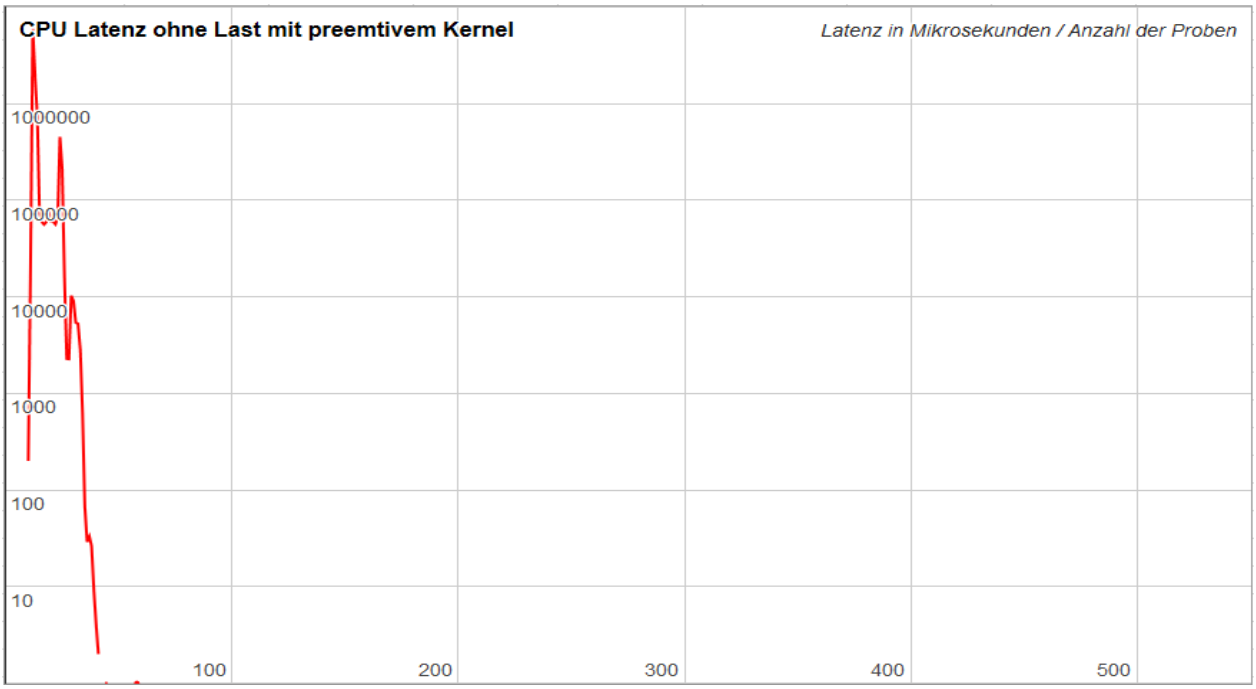

Abbildung 23: CPU-Latenz ohne Last bei einem Raspbian System mit präemptiven Kernel.

Der zweite Test mit Raspbian und einem präemptiven Kernel wurde wieder unter Volllast durchgeführt. Die Ergebnisse sind in Abbildung 24 dargestellt. Wie zu erkennen ist, unterscheiden sich die Ergebnisse nur minimal. Die beste Latenz lag wieder bei 10 Mikrosekunden und die schlechteste bei 70 Mikrosekunden. Als Gesamtergebnis lässt sich ableiten, dass die beste erreichte Latenz somit bei 10 Mikrosekunden lag und die schlechteste bei 70 Mikrosekunden. Daraus folgt, dass die Schwankung bei der Reaktion auf Anfragen, bei diesem System nur 60 Mikrosekunden beträgt. Es gilt auch festzuhalten, dass die etwas höhere beste Latenz von 10 Mikrosekunden, im Vergleich zu den 8 Mikrosekunden bei den Versuchen ohne präemptiven Kernel, nicht schlechter sind, sondern besser. Das System reagiert zwar etwas langsamer, jedoch sind die Schwankungen insgesamt geringer, was das System vorhersehbarer macht.

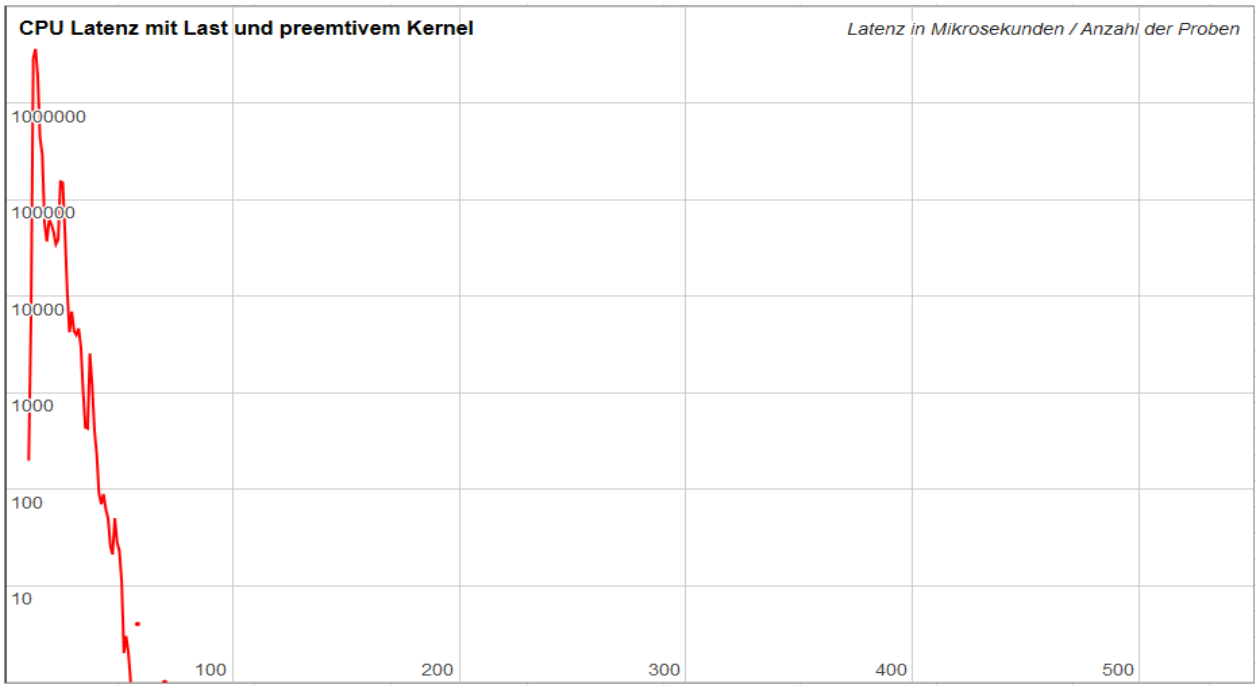

Abbildung 24: CPU-Latenz unter Volllast bei einem Raspbian System mit präemptiven Kernel.

Die nachfolgenden Abbildungen 25 und 26 zeigen die CPU-Latenz eines Raspberry Pi mit dem Xenomai Cobalt Kernel. Auf Abbildung 25 ist die Latenz der unbelasteten CPU und bei 26 mit einer voll ausgelasteten CPU zu sehen.

Wie bereits beschrieben, setzt sich die Zeitspanne der CPU-Latenz aus der Differenz der besten und der schlechtesten gemessenen Zeit zusammen. Der erste Test des unbelasteten Systems hat vielversprechende Ergebnisse hervorgebracht. Die beste Latenz lag bei 10 und die schlechteste bei 41 Mikrosekunden. Mit einer höchsten Latenz von 83 Mikrosekunden. bei einer CPU-Last von 100%, lag das System hinter einem normalen präemptiven Linux-Kernel. Für den Xenomai-Kern bedeutet dies 83 – 10 Mikrosekunden. Die Latenz beträgt also 71 Mikrosekunden.

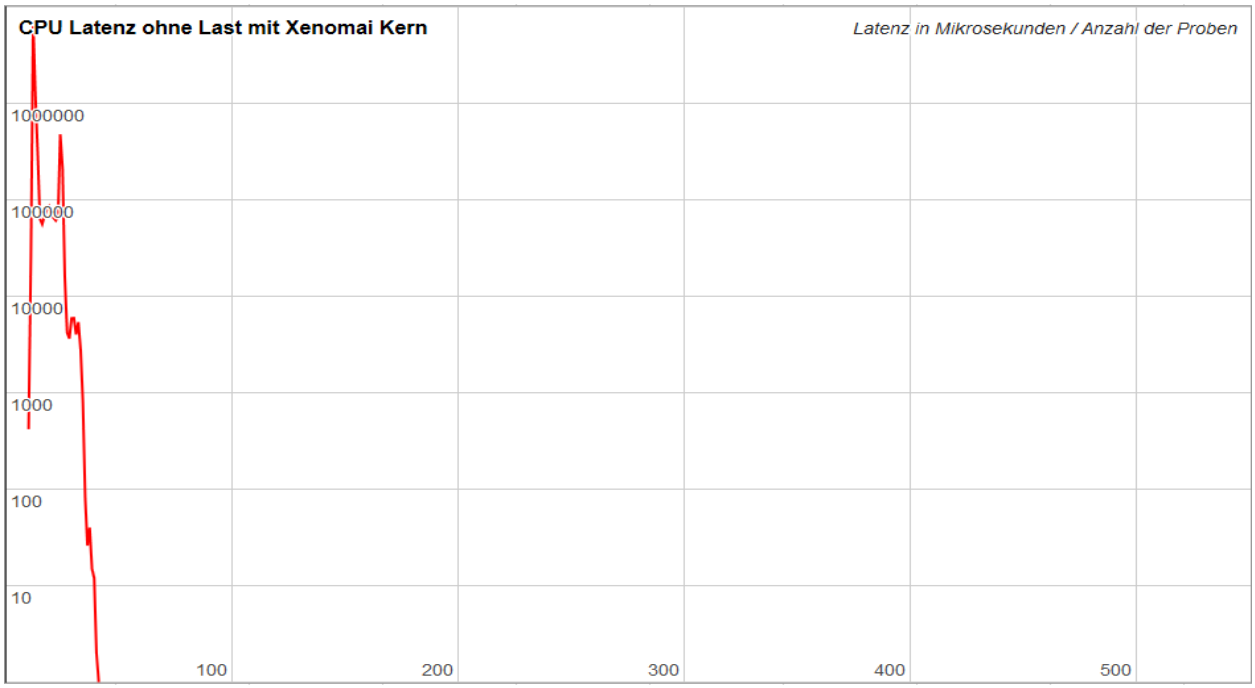

Abbildung 25: CPU-Latenz ohne Last mit einem Xenomai Cobalt Kern.

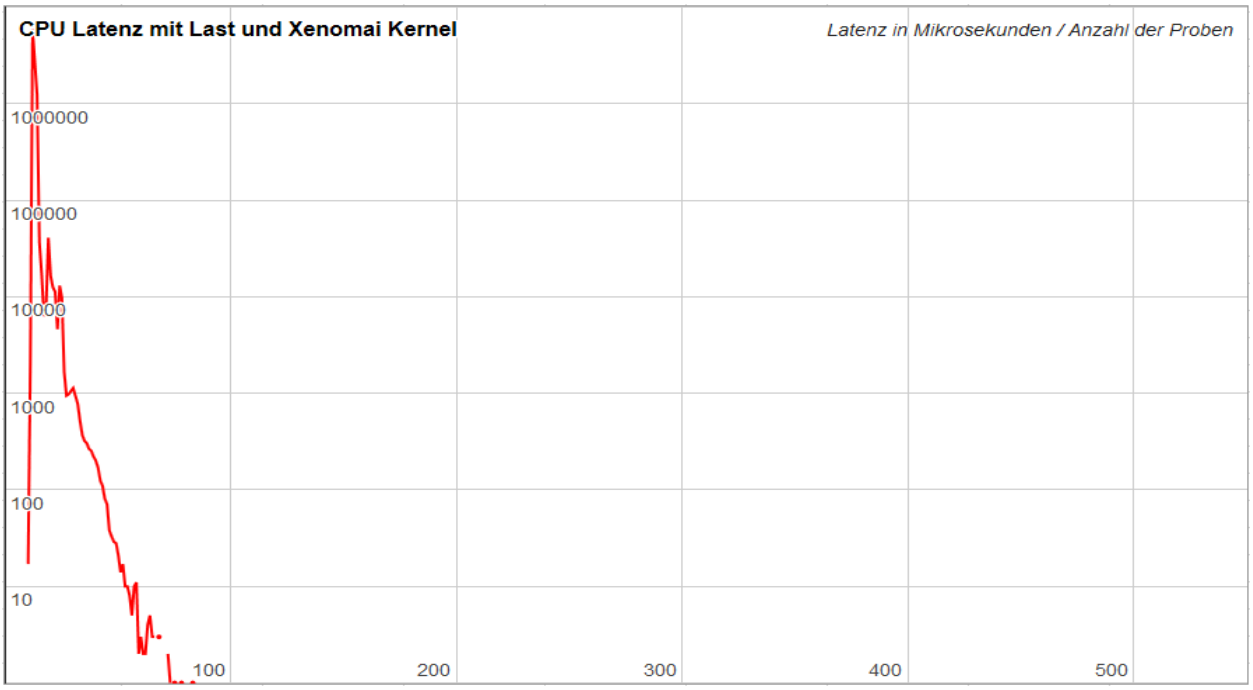

Abbildung 26: CPU-Latenz unter Volllast bei einem Xenomai Kernel.

Zu den bereits getesteten Betriebssystemen gibt es noch die Möglichkeit ein spezielles Realtime-Betriebssystem wie ChibiOS zu installieren. Bei diesem geht aber der Vorteil der einfachen Programmierbarkeit und dem Vorhandensein von Treibern für zahlreiche Peripheriegeräte verloren.

Weiter wird derzeit davon ausgegangen, dass die bereits erreichten Latenzzeiten mehr als ausreichend sind, um alle Aspekte der Highspeed-Fotografie abzudecken. Falls sich jedoch

herausstellen sollte, dass die erreichte Performance noch nicht ausreicht, wird bei den Tests der Geräte eine zusätzliche Testreihe mit ChibiOS durchgeführt.

In der nachfolgenden Abbildung 27 ist die Zusammenfassung der Latenztests zu sehen. Wie daraus hervorgeht ist, wurde die beste Echtzeitperformance von einem Raspbian mit präemptiven Kernel erreicht. Dieses System verspricht nicht nur die beste Performance, sondern auch eine sehr hohe Kompatibilität und einen großen Fundus an vorhandener Software. Aus diesem Grund wird für die weiteren Tests und Betrachtungen in erster Linie dieses Betriebssystem herangezogen.

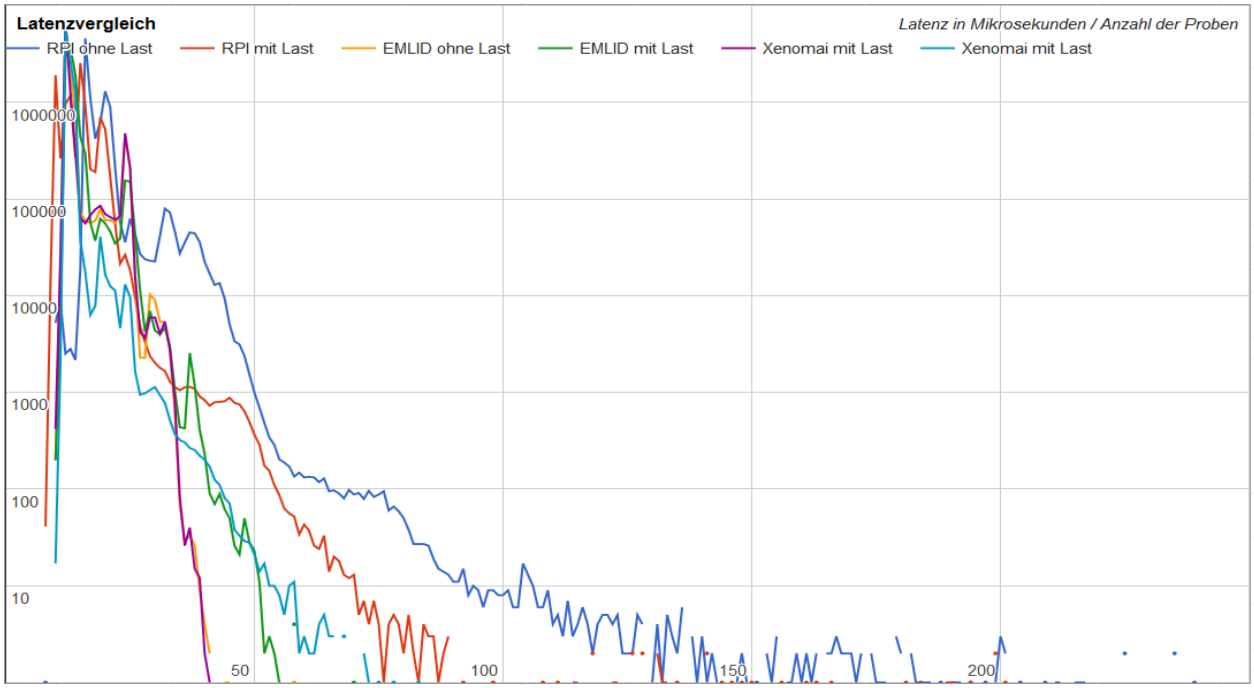

Abbildung 27: Latenz-Zusammenfassung.

System	Minimallatenz ohne Last in [µs]	Minimallatenz mit Last in [µs]	Maximallatenz ohne Last in [µs]	Maximallatenz mit Last in [µs]	Latenz spanne in [µs]
Raspbian	8	8	429	518	510
Raspbian RT	10	10	58	70	60
Xenomai	10	10	41	83	71

Tabelle 2: Ergebnisse aus dem Latenztest der Betriebssysteme.

Tabelle 2 zeigt die Ergebnisse der Betriebssystemtests, die für den Raspberry Pi durchgeführt wurden. Die Tests erfolgten unter Volllast und im Leerlauf, um die bestmöglichen und schlechtmöglichsten Ergebnisse zu erzielen. Die Tabelle listet die festgestellte Minimal- und Maximallatenz auf. Jeder Test wurde 10 Millionen Mal durchgeführt, um ein möglichst aussagekräftiges Ergebnis zu erzielen. Die letzte Spalte der Tabelle enthält die Latenzspanne, die aus der Differenz zwischen dem besten und dem schlechtesten Ergebnis für das jeweilige

Betriebssystem gebildet wurde. Für die Testfälle mit dem Raspberry Pi wird das Raspbian RT Betriebssystem gewählt, da es mit einer Latenz von 60 Mikrosekunden das beste Verhalten unter den getesteten Betriebssystemen aufweist.

5 DER VERGLEICHSTEST

In diesem Kapitel folgt eine theoretische Beschreibung aller geplanten Tests. Es wird der Aufbau und das Zusammenspiel der einzelnen Komponenten aufgezeigt und der Aufbau der Testumgebung erklärt. Weiter werden die verwendeten Schaltungen beschrieben. Danach wird die Inbetriebnahme des Systems erklärt und durchgeführt. Es wird auch aufgezeigt worauf dabei zu achten ist.

Der erste Test dient der Kalibrierung der Komponenten und dem Herstellen eines funktionsfähigen Settings, in dem der nachfolgende Test durchgeführt werden kann.

Im abschließenden Test wird ein normaler Anwendungsfall in der Tropfenfotografie ausprobiert. In diesem Test wird versucht herauszufinden wie zuverlässig der Raspberry Pi im Vergleich zum Arduino ist.

5.1 Aufbau des Test Settings

In diesem Test werden die einzelnen Komponenten getestet. Es wird durch eine Inaugenscheinnahme vorab getestet, ob alle Komponenten funktionieren und ob die Verkabelung richtig ist.

Zuerst wird die Tropfflüssigkeit hergestellt. Als Rezept dient die von (Nimmervoll, 2014) beschriebene und auf Guakernmehl basierende Flüssigkeit. Diese Flüssigkeit wird einen Tag vor den Tests hergestellt. Sie wird wie von (Nimmervoll, 2014) empfohlen 2-mal gefiltert und für die Vortests nicht eingefärbt. Die Viskosität der Flüssigkeit wird von (Nimmervoll, 2014) als milchähnlich beschrieben. Nach einer ersten visuellen Kontrolle scheint sie ein klein wenig dickflüssiger zu sein.

Für den Testaufbau wird ein Holzregal genutzt, auf dem alle Komponenten montiert werden. Da an das Regal keine besonderen Anforderungen gestellt werden, kann jedes beliebige verwendet werden. Um genug Abstand zwischen Tropfflüssigkeit und Auffangbehälter garantieren zu können, wird ein Regal mit einer Höhe von 150 cm verwendet. Als Material wurde auf Holz gesetzt, da dieses sehr leicht zu verarbeiten ist und Änderungen leicht umgesetzt werden können. Es ist damit beispielsweise möglich die Regalböden in jeder beliebigen Höhe zu montieren. Es müssen nicht die vorgefertigten Halterungen benutzt werden. Weiter können auch die Einlegeböden selbst modifiziert und angepasst werden.

Der Aufbau des Regals ist in Abbildung 28 dargestellt. Wie zu sehen ist werden vier Regalböden genutzt. Die oberste Etage dient als Ablage für die Schaltung, den Arduino, den Raspberry Pi und das Relais. Diese Komponenten sind allesamt stromführend. Deswegen werden sie bei diesem Aufbau ganz oben untergebracht. Zwar gibt es stromführende Teile, die weiter unten untergebracht sind, wie beispielsweise die Blitze, diese sind aber nicht auf dem Regalboden, wo sich Flüssigkeiten ansammeln könnten. Weiter sind sie durch ein Gehäuse geschützt, welches zumindest einzelne Wassertropfen abhalten sollte. Die vorhin beschriebenen Komponenten

liegen allesamt frei und auch ein einzelner Wassertropfen kann sofort zu einem Kurzschluss führen.

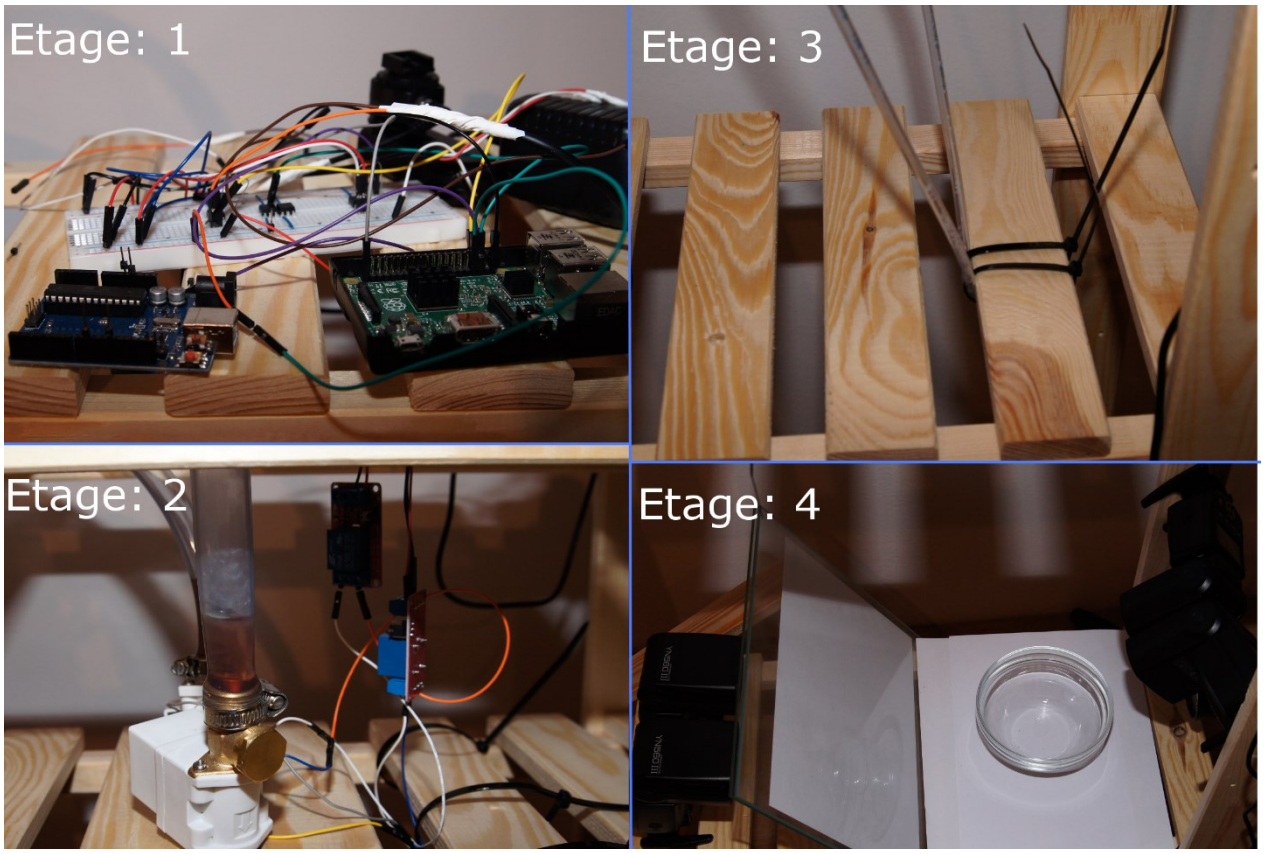

Abbildung 28: Schematischer Aufbau des Testsystems.

Direkt darunter, auf der zweiten Etage, werden die Wasserbehälter befestigt. Diese müssen hoch oben bei dem Aufbau montiert werden, damit genug Abstand zum Wasserbecken gegeben ist. Der minimale Abstand wird von (Nimmervoll, 2014) mit mindestens 50 cm angegeben, damit der Tropfen eine gleichmäßig hohe Wassersäule erzeugt.

Die Wassertanks werden mittels eines Schlauchs und einer Reduzierung an ein Magnetventil angeschlossen. Beim Testaufbau wird Magnetventil 21JN1R0V23 von ODE eingesetzt. Dieses Ventil besitzt eine nicht polarisierte 12V Magnetspule. Dies bedeutet, dass der positive und der negative Pol auch vertauscht angeschlossen werden dürfen. Das Ventil besitzt auf der Eingangs- und Ausgangsseite ein 1/8“ Innen-Gewinde. (ODE, 2007) Beim Eingang wird ein durchsichtiger PE-Schlauch angesetzt, der direkt zum Tank der Tropfflüssigkeit führt. An der Ausgangsseite wird ein Flex-Schlauch angeschraubt, an dessen zweitem Ende ein Schlauchanschluss angeschraubt ist. Der Schlauchanschluss reduziert den Durchmesser auf 6 mm. Es ist ein 20 cm langer 6 mm Schlauch angebracht, der auf der dritten Ebene des Holzgerüsts angebracht ist.

Auf der niedrigsten Etage wird das Auffangbecken angebracht. Dieses Becken wird mit Tropfflüssigkeit gefüllt. Die Wasserlinie des Beckens ist die Wasseroberfläche, auf der die Tropfen einschlagen sollen und fotografiert werden. Das Wasserbecken muss mindestens 20 cm vom Boden entfernt sein, damit die Kamera gut aufgestellt werden kann, um die Fotos zu machen.

Zusätzlich zum Wasserbecken sind auf dieser Ebene noch die Blitzgeräte angebracht, welche das Setting ausleuchten und für die scharfe Abbildung der Tropfen verantwortlich sind. Die Blitzgeräte, die von vorne auf die Kamera blitzen, werden durch eine trübe Plexiglasscheibe abgetrennt, damit das Blitzlicht weicher ist und die Blitzgeräte auf den Bildern nicht sichtbar sind. Für die Tests werden vier bis sechs, der in Kapitel 3.4.5 beschriebenen Yongnuo 560 Mark III Blitzgeräte benutzt. Die Blitzgeräte werden ausschließlich mit der schwächsten einstellbaren Stärke von 1/128 betrieben. Bei dieser Stärke weisen sie, wie bereits beschrieben, ihre kürzeste Abbrenndauer von 1 / 20000 Sekunde auf. Die genaue Anzahl der Blitzgeräte wird direkt bei der Durchführung entschieden. Das Test Setting sieht aber auf jeden Fall eine indirekte Belichtung gegen die Fotografie-Richtung und eine direkte Belichtung des Objekts, zum Aufhellen, von vorne vor.

Die Kamera selbst wird auf einem Stativ montiert, das direkt vor dem Holzregal steht. Es wird ein Rollei C5i Stativ genutzt. Dieses Stativ hat eine minimale Aufstellhöhe von 14 cm vom Boden bis zur Oberkante der Montageplatte.

Für den Arduino gibt der Hersteller eine Stromversorgung von 7 – 12 V an. Es wird auch darauf hingewiesen, dass der Arduino mit weniger als 7 V läuft. Wenn die 7 V unterschritten werden kann es jedoch vorkommen, dass der 5 V Pin bei den GPIO-Ports weniger als 5 V liefert. Da für die Versuche lediglich der 3,3 V Pin genutzt wird, kommt als Stromversorgung auch ein USB 2.0 Port oder höher in Frage. In Anbetracht der Tatsache, dass der Mikrocontroller über das USB-Port programmiert wird und zu erwarten ist, dass das Programm in der Testphase öfter geändert werden muss, wird ein USB 2.0 Port für die Stromversorgung genutzt. (Arduino, 2016) (USB Implementers Forum, Inc., 2000). Der Raspberry Pi braucht eine Versorgungsspannung von 5 V. Der Hersteller gibt, je nach Anwendung und angeschlossenen Geräten, einen typischen Stromverbrauch von 700 mA bis 1A für den Raspberry Pi an. Dieser Strom kann von einem USB-Port nicht geliefert werden. Aus diesem Grund wird er über ein USB-Netzteil mit Strom versorgt (USB Implementers Forum, Inc., 2000) (Raspberry Pi Foundation, 2016). Für den Arduino wird eine Stromversorgung über USB angestrebt, da dieser ohnehin eine USB-Verbindung zum PC benötigt. Für den Raspberry Pi ist dieser Verbindung nicht notwendig, da das Testprogramm auch über das Netzwerk übertragen werden kann.

Alle notwendigen Komponenten werden mittels eines Bread-Boards miteinander verdrahtet. Abbildung 29 zeigt die fertige Schaltung, die vorbereitet wurde, um bis zu 8 unterschiedliche Aktoren zu steuern. Das Board verfügt über zwei separate Lanes zur Stromversorgung. Die untere Lane hat einen Anschluss für zwei Verbindungskabel. Über diese Kabel wird die Stromversorgung vom Arduino bzw. Raspberry Pi zum Board herangeführt. Die nördliche Lane verfügt über einen 12 V DC Stecker, an welchem die 12 V Stromquelle angeschlossen wird. Eine Verbindung zwischen dem oberen und unteren Teil der Platine darf nur mittels eines Optokopplers hergestellt werden, um alle beteiligten Geräte vor einer Überspannung zu schützen.

Die acht Pins, die mittels der gelben Leitung verbunden sind, dienen der Signalgebung von den Steuergeräten. Zwischen der gelben Leitung und dem 4N35-Optokoppler ist jeweils ein 220 Ohm Vorwiderstand geschaltet. Dieser Widerstand verhindert, dass die Spannung des Signals zu groß ist und einen Kurzschluss darstellt. Der Widerstand mündet auf Pin 1 des Optokopplers, welcher

das positive Signal aufnimmt. Der Strom fließt über Pin 2 direkt auf den GND-Anschluss des Steuergeräts zurück. Pin 5 und 6 auf der Oberseite des Optokopplers geben das Signal direkt an die angeschlossenen Geräte weiter.

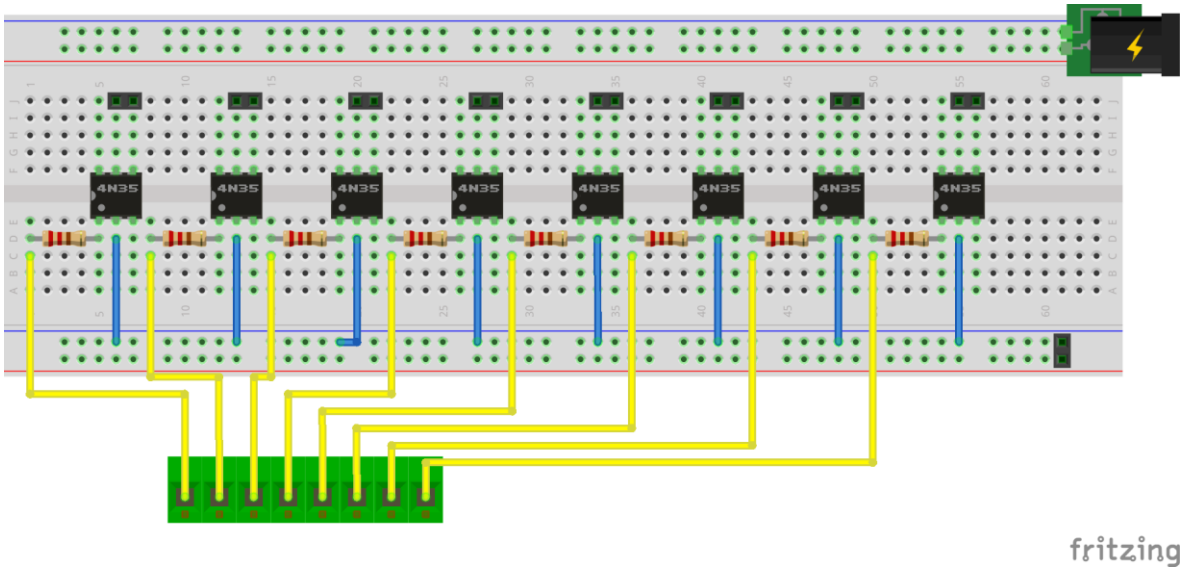

Abbildung 29: Bread-Board zur Steuerung von bis zu acht Aktoren.

Die Stärke des Stroms auf der Ausgangsseite des Optokopplers hängt von der Art des Optokopplers und vom Eingangsstrom ab. Durch die Verwendung eines Photo-Optokopplers und dem Vorwiderstand beim Stromeingang, kommt ein Signal zustande, das sehr schwach ist. Es reicht, um Geräte auszulösen, bei denen bei einem direkten Anschluss ein Kurzschließen der Kontakte reicht, um sie auszulösen.

Für die Testfälle in dieser Arbeit werden vier der acht abgebildeten Optokoppler-Schaltungen benötigt. Der erste Anschluss wird zum Auslösen der Blitze genutzt. Anschluss 2 und 3 werden vorbereitet, um jeweils ein Magnetventil zu steuern. Der vierte Anschluss dient dem Auslösen der Kamera selbst. Sowohl der verwendete Blitzauslöser als auch die Kamera können direkt durch das Überbrücken der Kontakte ausgelöst werden. Deswegen werden ihre Kontakte direkt mit den Kontakten auf dem Bread-Board verbunden. Die Schaltrelaises besitzen nur einen Konnektor für das Signal. Sie leiten das einkommende Signal über den Masseanschluss ihrer Schaltstromquelle ab. Somit muss der Masse-Pin des 5N35 auch auf die Schaltstromquelle abgeleitet werden. Zur Veranschaulichung dient Abbildung 30, bei der Pin 6 des 4N35-Optokopplers direkt mit dem Massepol der Stromquelle verbunden ist.

Zum Schalten der Magnetventile werden die in Kapitel 3.4.4 beschriebenen Relais eingesetzt. Die Relais müssen so zwischen den Magnetventilen und einer 12 V Stromquelle angeschlossen werden, dass sie als Schalter fungieren. Diese Schaltung wird in Abbildung 30 illustriert. Der Strom fließt vom Plus-Pol der Stromquelle zum Anschlusspin des Magnetventils. Dies ist auf der Abbildung durch den roten Draht dargestellt. Der negative Pol-Pin, der vom Ventil zurückkommt, wird direkt an das COM-Pin des Schaltrelais angeschlossen. Vom COM-Pin geht es weiter durch das Relais zum NO-Pin. Das NO-Pin ist direkt mit dem Masseanschluss der Stromquelle verbunden. Damit die Schaltung funktioniert, muss beim Relais der COM-Pin und zusätzlich der NC- oder NO-Pin belegt sein. NC und NO stehen für Normal Closed und Normal Open. Normal

bedeutet, dass das Relais nicht schaltet und somit im stromlosen Zustand ist. Das heißt, NO gibt an, dass der Stromkreis im nicht geschalteten Zustand des Relais offen ist und kein Strom fließen kann.

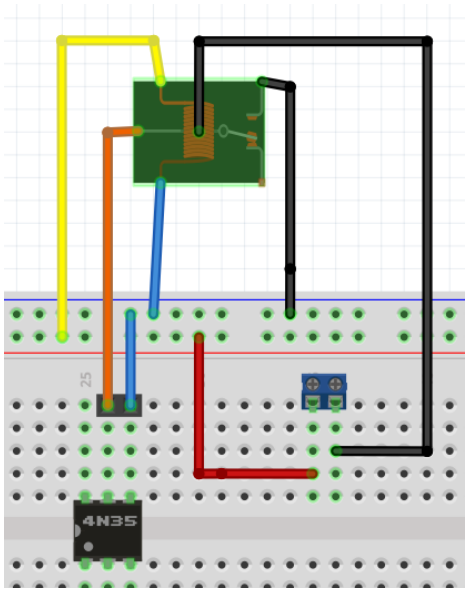

Abbildung 30: Anschlussschema des Schaltrelais vom Optokoppler zum Magnetventil

Als Stromquelle dient ein noName AC/DC Adapter, der einen Input von 230 – 250 V AC benötigt und einen Output von 12 V DC bereitstellt. Dieses Netzteil versorgt die Relaise und die Magnetventile mit Strom. Der Arduino und der Raspberry Pi werden nicht von dieser Stromquelle gespeist. Ein fertiges Produkt kann so ausgeführt werden, dass nur eine Stromquelle benötigt wird.

Die fertige Schaltung besteht somit aus einem Bread-Board, welches acht Optokoppler und zwei Strom-Lanes umfasst. An dieses Board wird je nach Testfall der Arduino oder der Raspberry Pi angeschlossen. Eine der Strom-Lanes wird direkt von einem 12 V DC Netzteil gespeist. Je ein Optokoppler wird an den Funkauslöser und an die Kamera angeschlossen. Die Optokoppler werden genutzt, um zwei Magnetventile zu steuern.

Nachdem das System nach Vorlage von Abbildung 29 aufgebaut und angeschlossen ist, wird das System mit Wasser befüllt. Dazu werden die Magnetventile geöffnet und es wird von unten mit einer Spritze Tropfenflüssigkeit in das Ventil gespritzt, bis im Wasserbehälter keine Luftblasen mehr aufsteigen. Zum Öffnen der Ventile können diese direkt an die Stromquelle angeschlossen werden.

Sobald die Anlage mit Tropfflüssigkeit befüllt worden ist, wird ein Testlauf gestartet, der die Funktionsfähigkeit des Systems testet. Dazu wird ein kleines Programm auf dem Arduino ausgeführt, das alle Funktionen der Schaltung einmal ausführt. Der in Listing 3 angeführte Auszug des Programms zeigt wie die Funktionen getestet werden und wie ein grundsätzlicher Testablauf aussieht. Das gesamte Testprogramm für diesen Test ist in Anhang B angeführt. Das Programm besteht im Grunde nur aus den zwei Funktionen die nacheinander aufgerufen werden. Zuerst wird die setup() Funktion ausgeführt und danach die *loop() Funktion*.

In der *setup()* Funktion wird der Testfall vorbereitet. Es werden grundlegende Funktionen wie das Logging gestartet und die Pins werden für den Test vorbereitet. Jeder verwendete Pin wird auf den benötigten Modus eingestellt und der Startzustand wird definiert. Für diesen Test wird, wie in Listing 3 zu sehen ist, der Modus auf *OUTPUT* gestellt und der Pin mit *LOW* beschrieben. Dies bedeutet nichts Anderes als dass die Pins genutzt werden, um Signale zu senden (OUTPUT) und dass das Signal den Zustand *LOW,* also keine Spannung, aufweist.

In der *loop()* Funktion wird der Pin mit *HIGH* beschrieben und nach einer kurzen Wartezeit wieder mit *LOW* überschrieben. Dies heißt, dass der Pin mit *HIGH* unter Strom gesetzt wird und nach der angegebenen Wartezeit wieder in den stromlosen Zustand gesetzt wird. Dies soll den jeweiligen Aktor auslösen. Die Zeitspanne, in der der Pin Strom führt, bestimmt im Fall der Magnetventile wie lange diese geöffnet sind. Für das Auslösen der Blitze genügt ein kurzes Aktivieren und Deaktivieren. Die Länge des Signals hat bei diesen Aktoren keinen Einfluss darauf wie sich die Geräte verhalten. Die Kamera würde normalerweise, gleich wie die Blitze, unabhängig von der Signalstärke agieren. Es gibt aber einen Bulb-Modus, der die Kamera dazu veranlasst den Verschluss so lange offen zu halten, so lange das Auslösesignal aktiv ist. Dies bedeutet, dass die Länge des Auslösesignals dazu genutzt werden kann, um die Exposure Time, das Zeitfenster, in dem der Verschluss offen ist, zu bestimmen.

```
void setup() {
  // Logoutput on the serial port
  Serial.begin(9600);
  // all used pins are signal output pins
  pinMode(12, OUTPUT);
  // set all pins to low before the test begins
  digitalWrite(12, LOW);
}

void loop() {
// Aktor
digitalWrite(12, HIGH);
delay(50);
digitalWrite(12, LOW);
}
```

Listing 3: Steuerungsprogramm für einen Aktor.

Im letzten Teil des Vortests werden alle Komponenten zeitgleich in einem Ablauf angesteuert, um zu sehen ob alles zeitgleich funktioniert. Nach diesem Test ist das System voll einsatzbereit und der Vortest kann abgeschlossen werden. Nach dem Abschluss kann direkt mit dem Kalibrieren des Systems auf die Testgeräte begonnen werden.

Der Vortest kann als erfolgreich gewertet werden, wenn alle Komponenten funktionieren, das System mit Flüssigkeit befüllt ist und regelmäßige Tropfen erzeugt werden können. Es müssen zwar Fotos erzeugt werden, jedoch müssen diese noch nicht richtig belichtet sein und es muss auch kein Tropfen auf den Fotos sichtbar sein. Es ist ausreichend, wenn alle Komponenten ihre

Funktion ausführen. Die zeitliche Abstimmung, sodass die Bilder belichtet sind und die Tropfen fotografiert werden, erfolgt bei der Kalibrierung des Testsystems.

5.2 Kalibrierung des Testsystems

Nach dem Vortest, bei dem das grundsätzliche Setup aufgebaut wurde, folgt der erste Haupttest. Das Ziel dieses Tests ist es das System so zu kalibrieren, dass ein bestmögliches Ergebnis garantiert werden kann. Im Gegensatz zum Vortest muss dieser Tests mit beiden Geräten durchgeführt werden, um das Testsetting auf den Arduino bzw. Raspberry Pi abzustimmen.

Das Ziel ist es, ein Tropfenfoto mehrmals zu machen und dabei immer das gleiche bzw. ein in engen Schranken liegendes, ähnliches Ergebnis zu erzielen. Um dies zu erreichen, wird ein einzelner Tropfen der Flüssigkeit vom Magnetventil freigegeben. Das System wird so eingestellt, dass der Tropfen nach Möglichkeit mittig auf dem Bild und vor dem Aufschlag auf der Flüssigkeit im Behälter fotografiert wird. Abbildung 31 zeigt die ideale Position des Wassertropfens auf dem Foto. Nach Möglichkeit soll jedes Foto genau den dargestellten Aufbau zeigen. Es kann im gesamten System aber zu geringen Abweichungen kommen, wodurch der Tropfen nicht immer genau mittig auf dem Bild ist. Damit diese Abweichungen möglichst genau bestimmt werden können und das System die bestmögliche Kalibrierung hat, wird der nachfolgend beschriebene Testablauf so lange durchgeführt, bis das System optimal kalibriert ist.

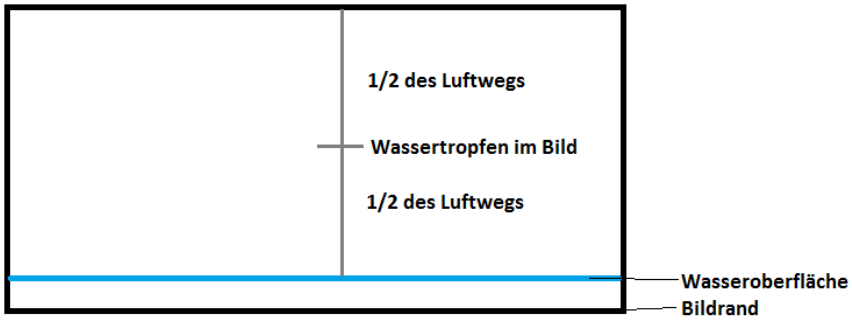

Abbildung 31: Position des Wassertropfens bei einem Kalibrierungsfoto.

Ausgehend davon, dass das System nach der Durchführung des Vortests noch keine Tropfenfotos erzeugt, muss es erst grob eingestellt werden. Dies kann entweder im Trial-and-Error Verfahren durch oftmaliges Probieren geschehen oder durch eine Berechnung.

Für die Berechnung wird das Gewicht eines Tropfens benötigt und ist im Fall der Berechnung des freien Falls ohne Luftwiderstand nur eine Annäherung. (Diehl, et al., 2001) Dies bedeutet, dass selbst bei der Berechnung noch eine Feineinstellung im Trial-and-Error Verfahren notwendig ist. Deswegen wird die erste Grobeinstellung direkt durch Schätzen und Testen gemacht. Nur wenn dies nicht gelingt, wird eine Berechnung durchgeführt, um einen Anhaltspunkt für das gesuchte Zeitfenster zu bekommen.

Nachdem das System so eingestellt ist, dass der Tropfen circa mittig zwischen dem oberen Bildrand und der Wasseroberfläche ist, kann mit der Feinkalibrierung begonnen werden. Dazu werden 20 Bilder mit der Grobeinstellung fotografiert. Bei jedem der Bilder wird die Abweichung

des Wassertropfens zur Bildmitte herangezogen, um das arithmetische Mittel der Tropfen zu bestimmen. Das System wird solange nachkalibriert, bis das arithmetische Mittel genau auf der Bildmitte liegt.

Wenn dies erreicht wurde, werden 50 Bilder erzeugt, mit denen eine Standardabweichung erstellt wird, um zu bestimmen, wie groß die Genauigkeit des Systems ist. Anhand dieser Tests kann bereits ein Eindruck darüber gewonnen werden wie gut sich der Raspberry Pi an die Performance des Arduino annähern kann. Selbst wenn der Raspberry Pi hier eine schlechtere Performance zeigt, als es der Arduino tut, schließt dies die Nutzung des Raspberry Pi in der Highspeed-Fotografie jedoch noch nicht aus. Für die genaue Beurteilung sind noch die weiteren Testfälle heranzuziehen.

5.3 Eine Wassersäule mit einem Tropfeneinschlag auf der Spitze

Bei diesem Test wird erstmals versucht ein typisches Szenario in der Highspeed-Fotografie nachzubilden. Es handelt sich dabei um ein Tropfen-auf-Tropfen (TaT) Szenario. Bei diesen Bildern wird mit zwei Tropfen gearbeitet. Der erste Tropfen fällt in das Auffangbecken und erzeugt eine kleine Flüssigkeitssäule. Mit einer kleinen zeitlichen Verzögerung wird ein zweiter Tropfen fallen gelassen. Dieser zweite Tropfen soll genau zu dem Zeitpunkt, an dem die Flüssigkeitssäule am höchsten ist, auf dieser einschlagen. Beim Zusammentreffen der Säule und des Tropfens platzt die Spitze der Säule und der Tropfen. Die Flüssigkeit wird zur Seite hin verdrängt, wodurch ein kleines Schirmchen entsteht. (Nimmervoll, 2014) Das Ziel dieses Tests ist es diese Schirmchen zu fotografieren.

Es gibt mehrere verschiedene Formen, die diese Schirmchen annehmen können. Auf der nachfolgenden Abbildung 32 sind drei mögliche Schirmchenformen abgebildet. Die erste Form, die ganz links zu sehen ist, sieht von der Form her aus wie ein kleiner Pilz. Das Schirmchen oder auch Kappe bildet eine Spitze, an deren höchstem Punkt die Wassersäule anstößt. Von diesem höchsten Punkt aus fällt die Form in alle Richtungen nach unten. Diese Form wird dann erreicht, wenn die Flüssigkeitssäule zum Zeitpunkt des Aufpralls mit dem Tropfen noch nicht ihre höchste Position erreicht hat. Durch den Aufprall wird die Flüssigkeit seitlich verdrängt und die Säule gewinnt zusätzlich noch an Höhe. Dadurch kommt diese Form zustande. Das mittlere Bild zeigt eine Säule mit einer ganz geraden und flachen Kappe. Um diese Form zu erzeugen, muss der Flüssigkeitstropfen genau mittig auf die Säule auftreffen und die Säule muss zu diesem Zeitpunkt genau an ihrem höchsten Punkt sein. Die rechte und damit letzte vorgestellte Form ist eine schräge Kappe auf der Säule. Diese entsteht, wenn der Wassertropfen etwas seitlich auf die Säule auftrifft. Die Verdrängung der Flüssigkeit verlagert sich dabei auf der Seite der Säule nach oben und auf der Seite des Tropfens nach unten. (Nimmervoll, 2014)

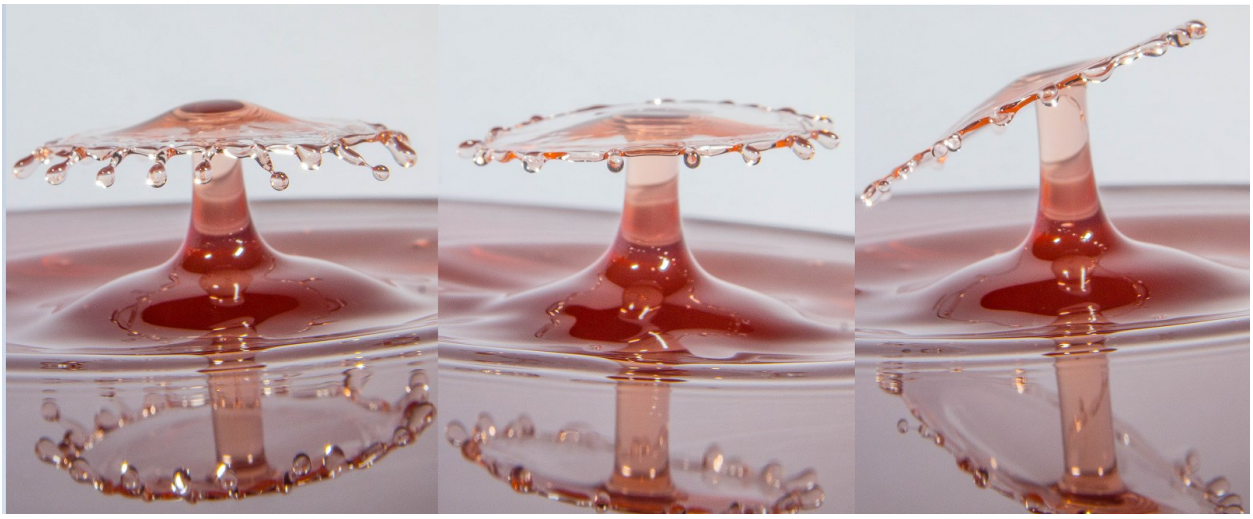

Abbildung 32: Arten der Schirmchen beim TaT-Versuch.

Die seitliche Verschiebung der Tropfen kann durch das Magnetventil hervorgerufen werden. Wenn der Tropfenauslass direkt am Magnetventil angebracht wurde, kann das Schalten des Ventils dazu führen, dass der Tropfen einen Schlag bekommt und nicht mehr ganz gleichförmig ist. Dies führt dazu, dass der Tropfen sich, im Vergleich zur Säule, leicht verschiebt, was eine schräge Kappe auf der Flüssigkeitssäule mit sich bringt. (Nimmervoll, 2014)

Das Testsystem wird so eingestellt, dass sich nur waagerechte Hütchen bilden sollen. Da aber auf eine Entkoppelung der Auslassdüse verzichtet wurde, dürfen auch schräge Kappen auftreten. Dies wird nicht als fehlerhaftes Bild gewertet, weil es sich um eine in Kauf genommene Varianz bei der Schirmchenbildung handelt. Wenn bei einem Bild ein ausgeprägtes Pilzhütchen oder ein gesamtes Verfehlen des Bildes auftritt, wird der Versuch als missglückt betrachtet. Es wird für jedes Testsystem eine Reihe mit 100 Versuchen durchgeführt, um anschließend zu beurteilen ob und wie viele der Bilder funktioniert haben bzw. Fehlschläge waren.

Um dieses Vorhaben umzusetzen werden zwei Dinge benötigt. Als erstes muss eine Wassersäule erzeugt werden. Dazu wird das Programm aus dem Vortest als Ausgangsbasis genommen. Wie bereits bei der Kalibrierung des Testsystems wird das Timing schrittweise angepasst, bis das Timing für die höchstmögliche Flüssigkeitssäule herausgefunden wurde. Dies ist dann der Fall, wenn jede weitere Anpassung am Timing eine kürzere Säule mit sich bringt.

Als zweites wird ein Flüssigkeitstropfen benötigt, der genau auf diese Säule einschlägt. Auch für diesen Teil des Programms wird das Programm zur Kalibrierung herangezogen. Das Timing wird nun soweit verändert, bis der Flüssigkeitstropfen sich auf der Höhe befindet, die dem höchsten Punkt der Säule entspricht.

Anschließend werden beide Programme zu einem zusammengefügt. Es ist genau darauf zu achten, dass sich die Timings der einzelnen Tropfen nicht verändern. Mit diesem Programm werden dann ein paar Probefotos gemacht. Durch kleine Veränderungen am Timing kann versucht werden die Aufprallgeschwindigkeit zu optimieren, damit schöne Formen entstehen. Nachdem das Programm fixiert ist, wird der Testlauf gestartet. Es werden 100 Bilder ohne Unterbrechung gemacht. Nach dem Testlauf werden die Bilder ausgewertet, um aufzuzeigen wie hoch der Ausschuss ist.

6 DURCHFÜHREN DER TESTS UND ERFASSEN DER ERGEBNISSE

Dieses Kapitel behandelt den genau dokumentierten Aufbau der einzelnen Tests. Weiter werden alle Änderungen am Testsystem zum vorhergehenden Test erläutert und aufgezeigt.

Zum Testaufbau selbst wird der verwendete Sourcecode vorgestellt und erläutert. Es wird auch kurz darauf eingegangen auf welche Teile besonders geachtet werden muss und warum dies wichtig ist. Die verwendeten Schaltungen werden erläutert und die einzelnen Komponenten erklärt sofern dies nicht schon in Kapitel 3.4 geschehen ist.

Es werden alle Adaptionen, sofern welche notwendig sind, erfasst und erklärt. Dies beinhaltet bauliche Änderungen am Testaufbau und auch alle Änderungen die am Sourcecode vorgenommen werden müssen damit die Tests funktionieren. Zusätzlich dazu werden auch Änderungen an den Schaltungen dokumentiert und argumentiert sofern dies nötig ist.

Abschließend wird jeder einzelne Testfall kurz zusammengefasst um einen Überblick über den letzten Stand und die Ergebnisse nach der Durchführung festzuhalten. Diese Ergebnisse werden in Kapitel 7 nochmals aufgegriffen und diskutiert.

6.1 Vortest

Dieser Test dient in erster Linie dazu das System aufzubauen. Es wird die Funktionsfähigkeit der einzelnen Komponenten geprüft und das System als Gesamtes erstellt. Falls Adaptionen am System notwendig sind, soll dieser Test deren Notwendigkeit aufzeigen. Das Ziel ist es, dass nach dem Test ein Aufbau besteht, mit dem alle weiterführenden Tests durchgeführt werden können.

Als Erstes werden das Regal und alle Komponenten, so wie es in Kapitel 5.1 beschrieben, aufgebaut. Nachdem der Aufbau fertiggestellt ist, wird das System mit der vorbereiteten Tropfflüssigkeit befüllt. Bei der ersten Befüllung wird das System nur soweit befüllt, dass der Boden des Flüssigkeitstanks minimal mit Flüssigkeit bedeckt ist. Das System wird dann in Augenschein genommen, um die Dichtheit der Anschlüsse zu beurteilen. Dies ist wichtig, damit die Tropfen beim Test selbst gleichmäßig erzeugt werden.

Nachdem die Dichtheit bestätig ist, wird ein erster Durchlauf mit dem in Kapitel 5.1 beschriebenen Programm unternommen, um die Funktionsfähigkeit aller Systeme zu testen. Als Öffnungszeit für die Ventile wurden 40 Millisekunden gewählt, weil dieser Zeitraum von (Nimmervoll, 2014) als ausreichend beschrieben wurde. Bei diesem Test konnte beobachtet werden, dass die Kamera und die Blitze wie gewollt ausgelöst wurden. Die Relais-Schaltung hat funktioniert und die Ventile haben sich geöffnet und wieder geschlossen. Es hat aber keiner der Flüssigkeitsauslässe einen Tropfen erzeugt, der in die Wasserschale gefallen wäre.

Für den zweiten Durchlauf wird die Öffnungszeit der Ventile verdoppelt, um mit Sicherheit einen Tropfen zu erzeugen. Bei diesem Durchlauf konnte beobachtet werden, dass wieder kein Tropfen erzeugt wurde. Jedoch ist die Entstehung eines Tropfens, wie Abbildung 33 zeigt, zu beobachten.

Abbildung 33: Entstehung eines Tropfens beim Auslassschlauch

Die Öffnungszeit wurde schrittweise erhöht, bis bei jedem Auslösen ein Tropfen in den Auffangbehälter gefallen ist. Dieser Punkt wurde erst bei einer Ventilöffnungszeit von 400 Millisekunden erreicht. Da von (Nimmervoll, 2014) beschrieben wurde, dass die Flüssigkeit eine milchähnliche Konsistenz haben soll und sich bei 56 Millisekunden bereits zwei Tropfen lösen sollen, muss das Testsetting angepasst werden.

Da die Tropfflüssigkeit nach einer rein subjektiven Einschätzung etwas zu dickflüssig ist, wird mit Wasser verdünnt. Dazu wird ein Liter Tropfflüssigkeit mit 500 ml Wasser verdünnt. Nach dem Verdünnen der Tropfflüssigkeit, wird das System entleert und neu befüllt. Bei der Befüllung wird wieder nur ein sehr niedriger Füllstand bei den Tanks ausgewählt, damit die Flüssigkeit leicht ausgetauscht werden kann.

Nach einem neuerlichen Funktionstest konnte beobachtet werden, dass sich bei einer Öffnungszeit von 50 Millisekunden bereits ein Tropfen löst. Leider wurde auch festgestellt, dass etwas zeitversetzt weitere Tropfen in die Auffangschale fallen. Es konnte auch beobachtet werden, dass bei Tests in kurzer Folge schneller Tropfen erzeugt werden, als es der Fall ist wenn eine Minute zwischen den Tests verstreicht. Als Fehlerursache werden Luftblasen in den Schläuchen oder der Luftdruck, der von außen auf die Schläuche wirkt, vermutet. Da Teile der Schläuche nicht durchsichtig sind und der Schlauchaufbau nach dem Ventil sehr lang ist, wird das System so umgebaut, dass beide möglichen Fehlerquellen vermieden werden.

Dazu muss der Aufbau, wie auf Abbildung 34 gezeigt, verändert werden. Die Ventile kommen, wie auf der linken Bildseite zu sehen ist, möglichst nahe an die Auslassstelle für die Tropfen. An der Ausgangsseite wird ein Verbindungsstück angebracht, an dem das Reduktionsteil auf 6 mm angeschraubt wird. Die rechte Seite des Bildes zeigt die Flüssigkeitseingangsseite der Ventile. An ihnen wird ein Schlauch angebracht, der die Flüssigkeit für das Ventil zur Verfügung stellt. Durch den neuen Testaufbau soll die Menge an Flüssigkeit, die sich nach dem Ventil in den Schläuchen und Anschlussteilen aufhält, möglichst geringgehalten werden.

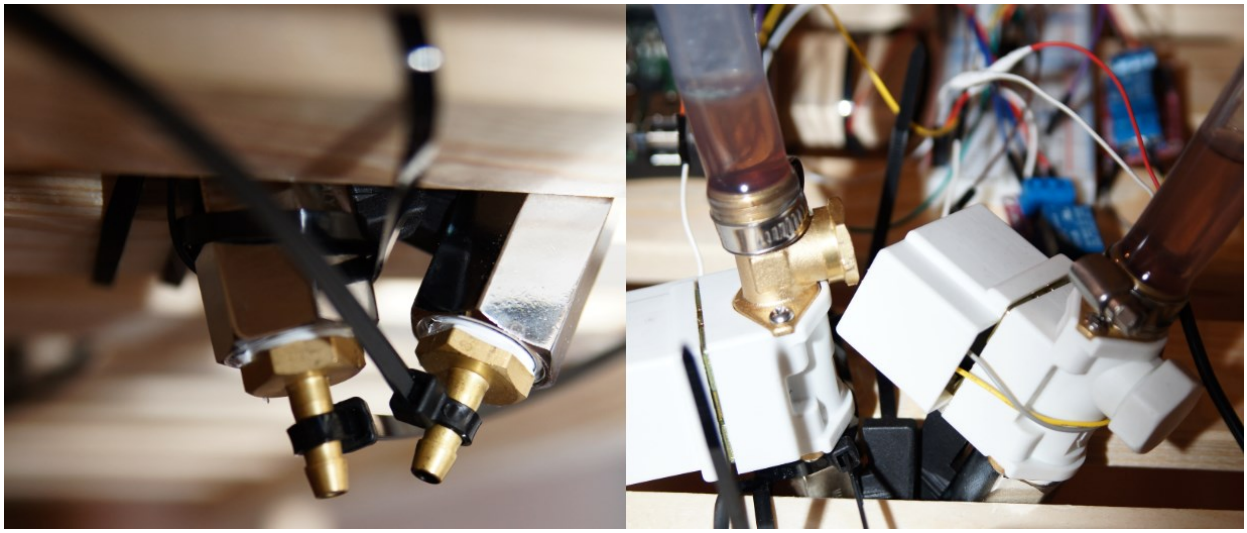

Abbildung 34: Testaufbau nach der Adaption am Ventil.

Der erste Test dieses Settings erzeugte bei 46 Millisekunden zuverlässig einen Tropfen. Da für spätere Testfälle viele Tropfen genutzt werden, wird der Schlauch an der Eingangsseite des Ventils mit Flüssigkeit befüllt und danach nochmals getestet, um Auswirkungen des Füllstands bereits jetzt zu erkennen. Bei einem erneuten Test hat sich gezeigt, dass bei einem vollen Schlauch und einer Öffnungszeit von 46 Millisekunden bereits ~ 4 Wassertropfen erzeugt werden. Damit eine gleichmäßige Tropfenbildung garantiert werden kann, wird eine Mariottesche Flasche als Flüssigkeitsbehälter genutzt. Diese Flasche dient dazu einen gleichmäßigen hydrostatischen Druck aufrecht zu halten.

Der Hydrostatische Druck ist jener Druck der von einer ruhenden Flüssigkeit oder einem ruhenden Gas ausgeübt wird (Grimvall, 2007). Nachdem die Mariottesche Behälter aufgebaut waren hat sich die benötigte Zeit für die Erzeugung eines Wassertropfens auf 220 Millisekunden erhöht. Nach einer kurzen Recherche zur Funktion der Flasche wurde festgestellt das über das Röhrchen am oberen Ende der hydrostatische Druck reguliert werden kann. Der gesamte Druck setzt sich aus der Flüssigkeit in den Rohren zum Ventil und der Flüssigkeit die sich zwischen Lufteinlass und Luftauslass befindet zusammen. Um das Druck erzeugende Volumen in der Flasche zu erhöhen muss das Luftröhrchen nach oben aus der Flasche gezogen werden. Dadurch wird das Flüssigkeitsvolumen zwischen dem Lufteinlass und dem Flüssigkeitsauslass höher und auch der Druck steigt an. Das Röhrchen wurde in diesem Testschritt so positioniert das nach 40 Millisekunden genau ein Tropfen Flüssigkeit aus dem Ventil austrat. Damit diese Flasche funktioniert, muss lediglich darauf geachtet werden, dass die Füllhöhe nicht unter die Höhe des Lufteinlasses sinkt.

6.2 Kalibrierungstest für den Arduino

Die Durchführung dieses Tests setzt voraus, dass bereits der Test aus Kapitel 6.1 abgeschlossen ist und ein funktionsfähiger Testaufbau existiert. Hier wird der Testaufbau so modifiziert und eingestellt, dass er perfekt mit dem Arduino zusammenarbeitet.

Als erstes wird eine grobe Voreinstellung der Software vorgenommen. Es hat sich gezeigt, dass dies am besten durchführbar ist, wenn zuerst mit einer sehr langsamen Verschlusszeit gearbeitet wird, um ein möglichst großes Zeitfenster abzudecken. Wenn der Fotoapparat so eingestellt ist, dass der Tropfen abgebildet ist, wird das Auslösen des Blitzes so lange nachjustiert, bis der Zeitpunkt stimmt und das Bild schön ausgeleuchtet ist. Danach kann damit begonnen werden die Belichtungszeit schrittweise zu reduzieren. Auf diese Weise wurde die Kamera so eingestellt, dass eine Wassersäule fotografiert wurde.

Nachdem das System zuverlässig mehrere Wassersäulen fotografiert hat, wurde der Auslösezeitpunkt der Blitzgeräte, wie es die Abbildung 35 zeigt, angepasst, bis der Tropfen circa mittig auf dem Bild zu sehen war.

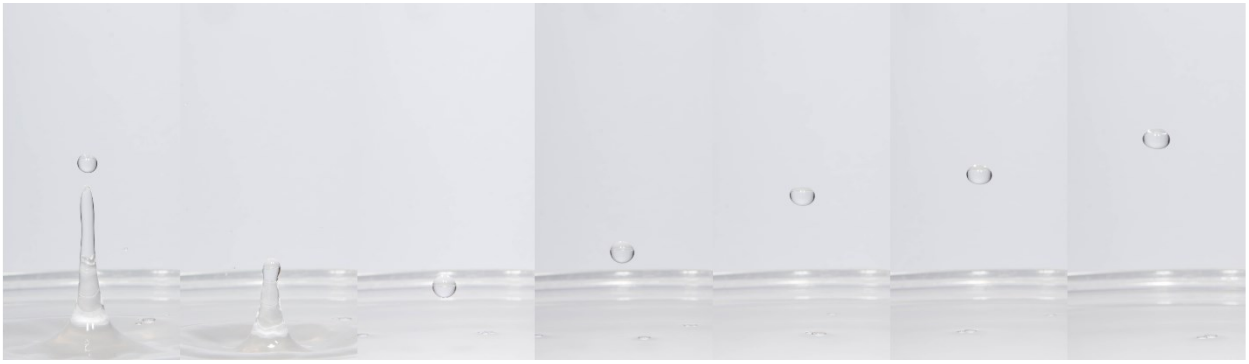

Abbildung 35: Bilderreihe bei der Einstellung des Tropfens.

Die letzte Einstellung, bei der in Abbildung 35 gezeigten Fotoreihe, war bei der Kamera ISO 200 mit einer Belichtungszeit von 1/20 Sekunde und einer Blendenzahl von 16. Die Blitzgeräte sind, wie bereits beschrieben, mit der niedrigsten Stärke von 1/128 eingestellt. Das für die Kalibrierung genutzte Programm für den Arduino ist in Anhang C angeführt.

```
void loop() {
// Relaise 1
digitalWrite(7, HIGH);
delay(40);
digitalWrite(7, LOW);
// wait before shutter
delay(240);
// Shutter
digitalWrite(2, HIGH);
delay(10);
digitalWrite(2, LOW);
// wait before bulb
delay(110);
// Bulb
digitalWrite(12, HIGH);
digitalWrite(12, LOW);
delay(5000);
}
```

Listing 4: Programm für das Erstellen eines Tropfenfotos.

In Listing 4 ist zu erkennen, dass die Kamera 240 Millisekunden nach dem Schließen des Magnetventils auslöst. Nach weiteren 120 Millisekunden werden die Blitzgeräte ausgelöst. Mit diesen Einstellungen wurde die erste Testreihe zur Feineinstellung erzeugt.

In dieser Testreihe wurden 25 Fotos gemacht. Diese erste Bilderreihe hat gezeigt, dass es nur sehr schwer möglich ist die genaue Position des Tropfens auf dem Bild zu bestimmen. Der Grund dafür liegt darin, dass der Tropfen durchsichtig ist und sich nur sehr wenig vom Hintergrund abhebt. Für alle weiteren Tests wird die Tropfenflüssigkeit rot eingefärbt, damit die Tropfen leichter vom Hintergrund unterschieden werden können.

Die zweite Testreihe mit 25 Bildern wurde mit einer rot eingefärbten Tropfflüssigkeit durchgeführt. Die Bilder wurden mit Adobe Lightroom CC nachbearbeitet, um die Schärfe und den Kontrast zu erhöhen. Es wurden keine weiteren Änderungen, wie das Retuschieren von Wasserspritzern auf der Linse des Objektivs, vorgenommen, da dies keinen Einfluss auf die Ergebnisse bei der Auswertung hat und auch zu keiner Verbesserung der Messbarkeit führt. Durch die Nachbearbeitung ist es leichter möglich die Position der Wassertropfen zu vermessen.

In der nachfolgenden Tabelle 3 sind die erfassten Ergebnisse der 25 Bilder aus der letzten Testreihe angeführt. Die erste Spalte gibt den vertikalen Durchmesser der Tropfen an. Es wurde von (Nimmervoll, 2014) beschrieben, dass die Tropfen nicht unbedingt rund sein müssen. Weiter ist auf Abbildung 35 und den Testbildern zu erkennen, dass dies auch auf die Testfotos zutrifft. Damit dies keine Auswirkung auf die Testergebnisse hat, wird die Schnittlinie vertikal durch den Tropfen gezogen. Der Radius des Tropfens wird genutzt, um dessen Mittelpunkt zu ermitteln und damit die Position des Tropfens auf dem Bild zu bestimmen.

Bildname	*Vertikaler Durchmesser der Tropfen in [Pixel]*	*Abstand der Tropfen zum oberen Rand in [Pixel]*	*Höhe von oben zum Tropfenmittelpunkt in [Pixel]*	*Abweichung vom Mittelwert in [Pixel]*
DSC5050	297	1487	1635,50	13,64
DSC5051	297	1510	1658,50	9,36
DSC5052	298	1568	1717,00	67,86
DSC5053	302	1880	2031,00	381,86
DSC5054	296	1232	1380,00	269,14
DSC5055	296	1413	1561,00	88,14
DSC5056	297	1467	1615,50	33,64
DSC5057	294	1168	1315,00	334,14
DSC5058	297	1491	1639,50	9,64
DSC5059	299	1801	1950,50	301,36
DSC5060	298	1564	1713,00	63,86
DSC5061	300	1691	1841,00	191,86
DSC5062	298	1573	1722,00	72,86
DSC5063	297	1688	1836,50	187,36
DSC5064	295	1608	1755,50	106,36
DSC5065	294	1506	1653,00	3,86
DSC5066	294	1494	1641,00	8,14
DSC5067	296	1532	1680,00	30,86
DSC5068	294	1287	1434,00	215,14
DSC5069	295	1305	1452,50	196,64
DSC5070	294	1270	1417,00	232,14
DSC5071	294	1527	1674,00	24,86
DSC5072	293	1372	1518,50	130,64
DSC5073	296	1544	1692,00	42,86
DSC5074	296	1547	1695,00	45,86

Tabelle 3: Testergebnisse der Kalibrierung für den Arduino.

Aus diesen Daten wurden zwei Diagramme erstellt. Als erstes wird die Verteilung der Tropfen auf dem Bild betrachtet. Die Abbildung 36 zeigt eine Verteilungsdichtefunktion für die Position der Tropfen auf dem Bild. Der höchste gemessene Tropfen hatte einen Abstand von 1315 Pixel zum oberen Bildrand und der niedrigste einen von 2031 Pixel. Die höchste Tropfendichte lag bei 1659 Pixel.

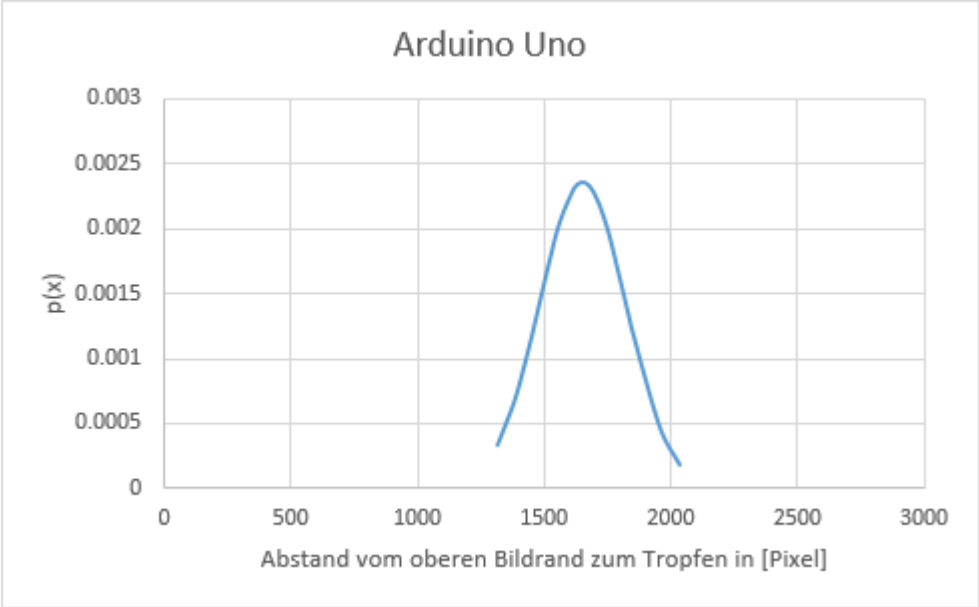

Abbildung 36: Normalverteilung der absoluten Tropfenposition auf dem Bild für den Arduino.

Als zweites wurde die Streuungsbreite der Tropfen erfasst. Abbildung 37 zeigt wie weit sich die einzelnen Tropfen voneinander entfernen. Diese Auswertung wurde gemacht, um aufzuzeigen welche Varianz die einzelnen Tropfen untereinander ausweisen. Dies kann als Maß dafür genommen werden wie zuverlässig das System arbeitet und mit welchen Schwankungen zu rechnen ist. Der Mittelpunkt bei den Messwerten liegt bei knapp 124 Pixel. Die Spannweite der dem Mittelpunkt nächsten 50% der Messwerte entspricht 143 Pixel.

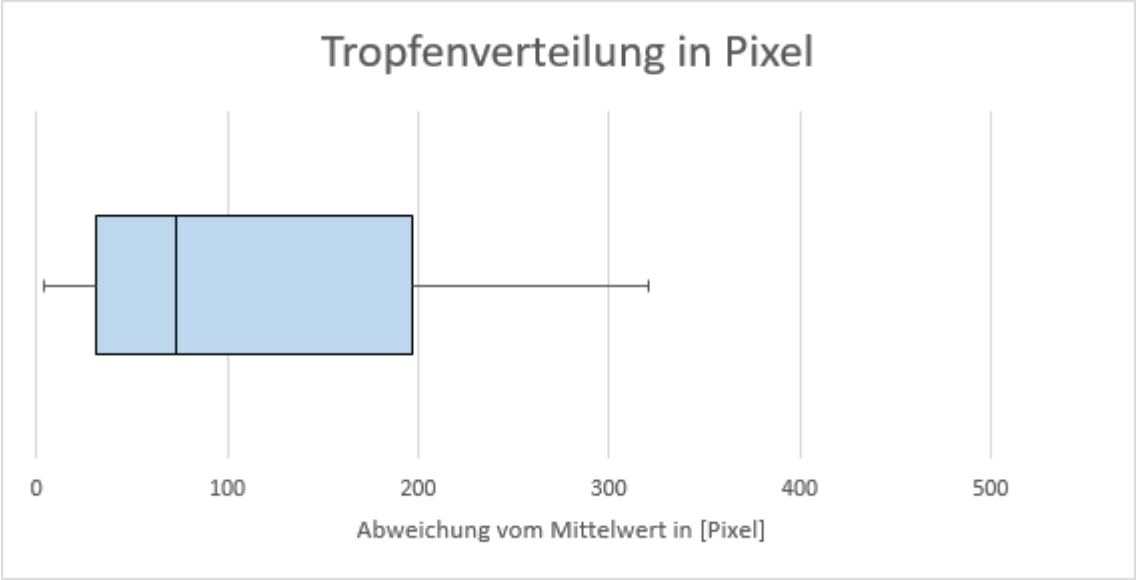

Abbildung 37: Abweichung der Tropfen vom Mittelpunkt beim Arduino.

6.3 Kalibrierungstest für den Raspberry Pi ohne RT-Kernel

Der Kalibrierungstest wurde für den Raspberry Pi zwei Mal durchgeführt, einmal mit dem RT-Patch und einmal ohne RT-Patch. Der Test ohne RT-Patch wurde durchgeführt, um

herauszufinden ob der Raspberry Pi auch ohne diesen Patch bereits in der Lage ist in der Highspeed-Fotografie eingesetzt zu werden.

Die Testreihe hat gezeigt, dass es grundsätzlich möglich ist den Raspberry Pi auch ohne den Patch in der Tropfenfotografie einzusetzen. Wie auf Abbildung 38 zu sehen ist, sind die Schwankungen aber sehr groß und kommen oft vor. Auf manchen Bildern ist noch der Verschluss zu sehen und bei anderen ist der Tropfen noch nicht ganz im Bild. Von den 20 Kalibrierungsbildern haben 4 Bilder Abweichungen, bei denen der Tropfen noch nicht im Bild ist oder bereits der Verschluss zu sehen ist. 16 Bilder zeigen ein Bild, bei dem der Tropfen ca. mittig auf dem Bild zu sehen ist. Da bereits bei diesen Bildern zu sehen ist, dass es zu großen Abweichungen kommt, wird auf weitere Tests mit dem Raspberry Pi ohne RT-Patch verzichtet.

Abbildung 38: Kalibrierungstest mit dem Raspberry Pi ohne RT-Patch.

6.4 Kalibrierungstest für den Raspberry Pi mit RT-Kernel

Es ist möglich Software für den Raspberry Pi in vielen verschiedenen Sprachen wie Java, Python, C, C++, uvm. zu programmieren. Der Minicomputer wurde ursprünglich für die Programmiersprache Python konzipiert. Dies hat sich auch im Namen des Geräts niedergeschlagen. Das Pi im Raspberry Pi steht für die Sprache Python. (Weigend, 2016) (Saint-Andre, Smith, & Troncon, 2009) (Courington & Collins, 2016) Aus diesem Grund wird für die Steuerung ein Python-Skript eingesetzt.

Damit über Python auf die GPIO-Pins leicht zugegriffen werden kann, muss erst eine Library installiert werden. Für diese Testfälle wurde auf dem Raspberry Pi RPi.GPIO in der derzeit aktuellsten Version 0.6.3 installiert.

Das nachfolgende Listing 5 zeigt das Programm vor dem Test. Da sich am Testsystem nur das Steuergerät ändert, alle anderen Komponenten aber gleichbleiben, werden die Timings vom Arduino-Sketch übernommen. Im Unterschied zum Arduino ist die Zeiteinheit in diesem Programm eine Sekunde und nicht eine Millisekunde. Damit die Timings trotzdem auf eine Millisekunde genau abgebildet werden können, kann die Zeit mit Kommastellen angegeben

werden. Zum Beispiel werden die 40 Millisekunden zum Erzeugen eines Tropfens als 0.04 angegeben.

Zuerst importiert das Skript die benötigte RPi.GPIO Library und initialisiert alle benötigten GPIO-Pins. Direkt danach folgt eine Endlosschleife, die nach dem Start so lange wiederholt wird, bis das Programm vom Benutzer unterbrochen wird. Für das in Listing 5 gezeigte Skript gelten die nachfolgenden Timings. Direkt zum Start der Schleife pausiert das Programm für 10 Sekunden. Dies ist notwendig, um der Flüssigkeit im Behälter die notwendige Zeit zu geben sich zu beruhigen. Nach der Pause wird zuerst der Tropfen ausgelöst, indem das Ventil 0.04 Sekunden geöffnet wird. Nach einer Wartezeit von 0.24 Sekunden erfolgt das Auslösen der Kamera mit einem 0.01 Sekunden langen Puls. Nachdem noch einmal 0.11 Sekunden gewartet wurde, wird zum Abschluss der Blitz ausgelöst.

```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(35, GPIO.OUT) #valve1
GPIO.setup(36, GPIO.OUT) #valve2
GPIO.setup(37, GPIO.OUT) #shutter
GPIO.setup(38, GPIO.OUT) #bulb
while 1:
    GPIO.output(35, False)
    GPIO.output(36, False)
    GPIO.output(37, False)
    GPIO.output(38, False)
    sleep(10)
    GPIO.output(35, True)
    sleep(0.04)
    GPIO.output(35, False)
    sleep(0.24)
    GPIO.output(37, True)
    sleep(0.01)
    GPIO.output(37, False)
    sleep(0.11)
    GPIO.output(38, True)
    GPIO.output(38, False)
```

Listing 5: Python Programm für die Kalibrierung des Raspberry Pi.

Da die Tropfenflüssigkeit bei den Arduino-Tests bereits eingefärbt wurde, sind die Fotos für das Kalibrieren hier bereits eingefärbt. Wie auf Abbildung 39 zu erkennen ist, hat der Raspberry Pi mit den vom Arduino übernommenen Einstellungen bereits die Tropfen in der Luft fotografiert. Durch die Anpassung der Timings ist es möglich die Tropfen nach oben oder unten zu verschieben. Die Unterschiede in der Höhe der einzelnen Tropfen, die auf der Abbildung zu sehen sind, können damit aber nicht verbessert werden. Deswegen wird auf eine Anpassung der

Timings verzichtet. Falls es dennoch notwendig wird, kann dies jederzeit noch nachgeholt werden.

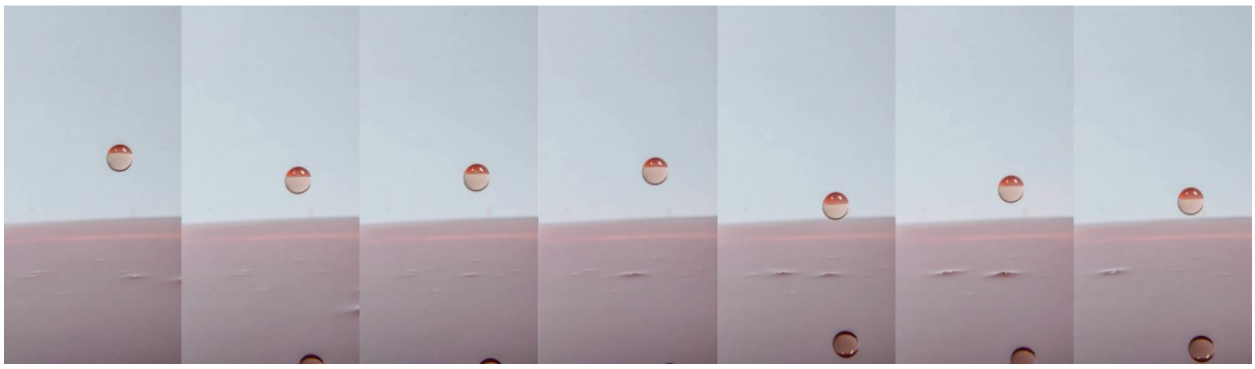

Abbildung 39: Tropfen-Kalibrierung mit dem Raspberry Pi

Für den Raspberry Pi wurde gleich wie für den Arduino eine Testreihe mit 25 Bildern erstellt. Die Testergebnisse sind in Tabelle 4 aufgeführt. Aus der Ergebnistabelle ist zu erkennen, dass ein Datensatz mit dem Namen DSC05348 fehlt. Dieses Bild wurde nicht korrekt belichtet. Es wurde aus der Wertung aber ausgeschlossen, da es kein Fehler des Steuergeräts war. Das Bild wurde nicht belichtet, da eines der Blitzgeräte ausgefallen ist. Der Versuch wurde mit einem Ersatzgerät fortgesetzt und das Bild ausgeschieden. Die aufgeführten Daten entsprechen ansonsten dem gleichen Schema wie bei der Testdurchführung mit dem Arduino.

Bildname	*Vertikaler Durchmesser der Tropfen in [Pixel]*	*Abstand des Tropfens zum oberen Rand in [Pixel]*	*Höhe von oben zum Tropfenmittelpunkt in [Pixel]*	*Abweichung vom Mittelwert in [Pixel]*
DSC05339	321	1537	1697,50	403,02
DSC05340	325	1783	1945,50	155,02
DSC05341	327	1752	1915,50	185,02
DSC05342	325	1686	1848,50	252,02
DSC05343	330	2069	2234,00	133,48
DSC05344	323	1882	2043,50	57,02
DSC05345	324	2017	2179,00	78,48
DSC05346	326	1776	1939,00	161,52
DSC05347	326	1969	2132,00	31,48
DSC05349	328	2302	2466,00	365,48
DSC05350	326	2069	2232,00	131,48
DSC05351	326	2144	2307,00	206,48
DSC05352	327	2028	2191,50	90,98
DSC05353	325	2042	2204,50	103,98
DSC05354	324	1993	2155,00	54,48
DSC05355	324	2006	2168,00	67,48
DSC05356	326	1991	2154,00	53,48
DSC05357	324	1933	2095,00	5,52
DSC05358	325	1929	2091,50	9,02
DSC05359	326	1852	2015,00	85,52
DSC05360	326	1985	2148,00	47,48
DSC05361	327	1957	2120,50	19,98
DSC05362	326	1905	2068,00	32,52
DSC05363	325	1977	2139,50	38,98
DSC05364	326	1860	2023,00	77,52

Tabelle 4: Testergebnisse der Kllibrierung für den Raspberry Pi.

Als Datenauswertung wurden erneut 2 Diagramme erstellt. Das erste Diagramm, welches auf Abbildung 40 zu sehen ist, entspricht wieder einer Verteilungsdichtefunktion. Für dieses Diagramm gelten die gleichen Voraussetzungen wie für das Diagramm, das für den Arduino erstellt wurde. Es umfasst die 25 in Tabelle 4 aufgeführten Datensätze. Der höchste gemessene Tropfen, bei dem das System am schnellsten reagiert hat, liegt bei 1698 Pixel und der niedrigste, bei dem die Reaktionszeit am langsamsten war, liegt bei 2466 Pixel. Die höchste Tropfendichte liegt bei 2121 Pixel, vom oberen Bildrand gemessen.

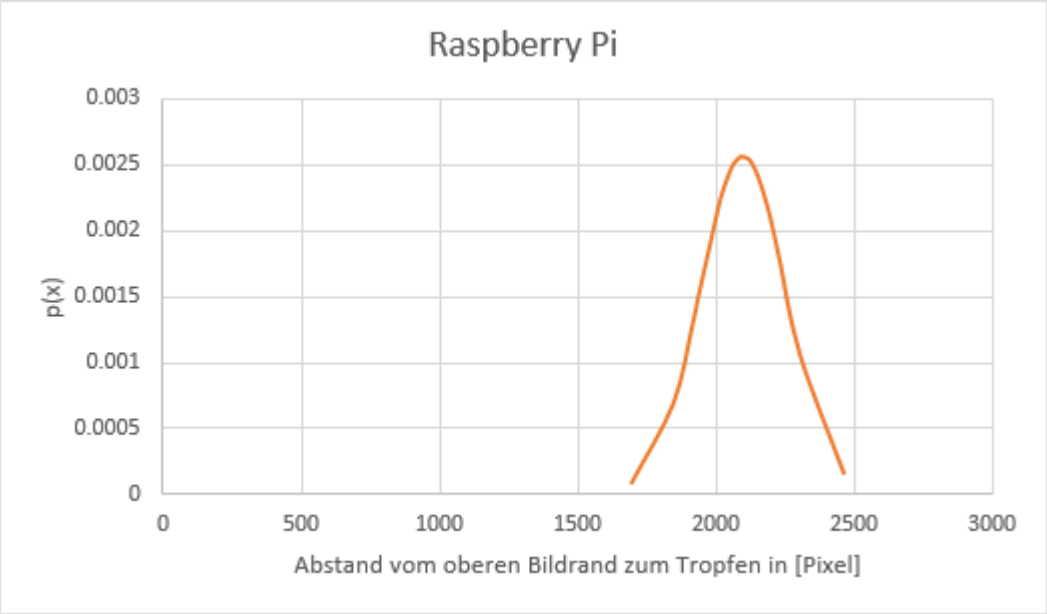

Abbildung 40: Normalverteilung der absoluten Tropfenposition auf dem Bild für den Raspberry Pi.

Die nachfolgende Abbildung 41 zeigt den Whisker-Box-Plot für die Testreihe mit dem Raspberry Pi. Es ist zu erkennen, dass die Spannweite bei 50% der Tropfen nicht mehr als circa 110 Pixel betrug. Weiter ist zu erkennen, dass sich der Großteil davon im Bereich zwischen 50 und 100 Pixel aufhält. Bei 76 Pixel liegt der Mittelpunkt der Messdaten.

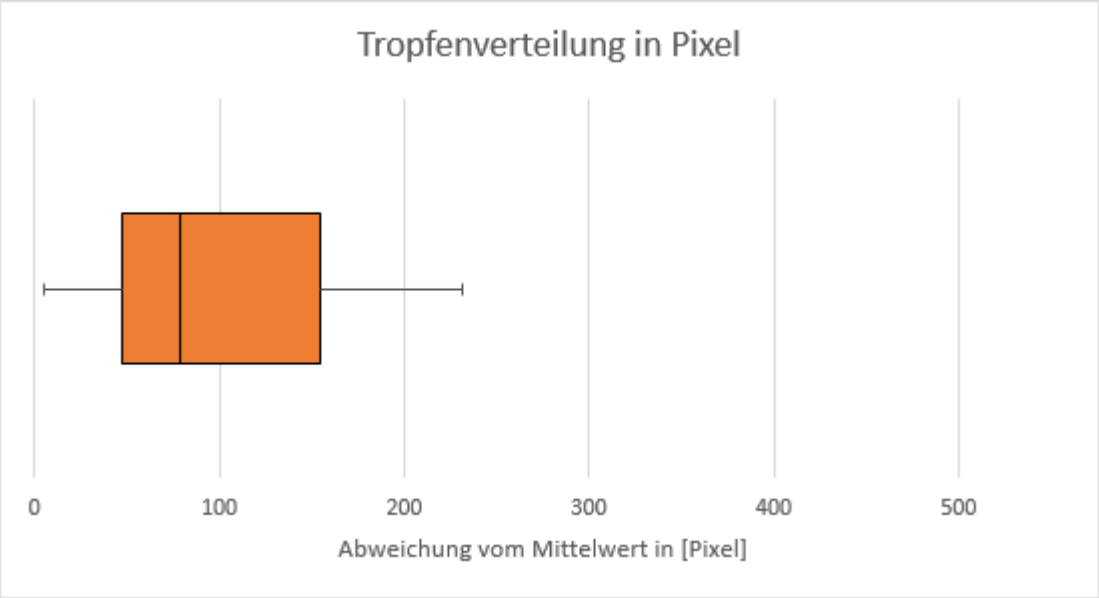

Abbildung 41: Abweichung der Tropfen vom Mittelpunkt beim Raspberry Pi.

6.5 Wassersäule mit Tropfeneinschlag unter Verwendung des Arduino

Zur Erstellung von TaT-Bildern wird, wie bereits in Kapitel 5.3 beschrieben, eine Kalibrierung der Wassersäule und des Tropfens, der darauf einschlägt, durchgeführt. Dies ist notwendig, um die

Timings zu ermitteln, die notwendig sind, um den Test durchzuführen. Für die Kalibrierung der Wassersäule wird das bereits in Kapitel 6.2 beschriebene Programm für den Arduino genutzt. Ausgehend von diesem Programm werden die Timings ermittelt und ein Programm entwickelt, welches den Test durchführt.

Zuerst werden die Timings für die Flüssigkeitssäule ermittelt. In dem in Listing 6 gezeigten Code ist in Rot das *delay()* hervorgehoben, das geändert werden muss, damit sich der Flüssigkeitstropfen auf dem Bild verschiebt. Eine Erhöhung des Werts bewirkt, dass das System länger wartet und der Tropfen mehr Zeit bekommt, um einzutauchen und eine Säule zu bilden, bevor das Bild gemacht wird.

```
void set1() {
// Relaise 1
fireDrop();
// wait before shutter
delay(285);
// Shutter
digitalWrite(2, HIGH);
delay(10);
digitalWrite(2, LOW);
// wait before bulb
delay(110);
// Bulb
digitalWrite(12, HIGH);
digitalWrite(12, LOW);
}
```

Listing 6: Funktion zur Erzeugung einer Flüssigkeitssäule mit dem Arduino.

Durch eine schrittweise Erhöhung der Wartezeit auf 10 Millisekunden, im Vergleich zum Kalibrierungstest, wurden die in Abbildung 42 gezeigten Bilder erzeugt. Für das Durchführen des Tests wird zur Erzeugung der Säule eine gesamte Verzögerung von 285 Millisekunden bis zum Auslösen der Kamera und 405 Millisekunden bis zum Auslösen der Blitze angestrebt. Zur Erzeugung dieser Bilder wurden vier Blitzgeräte genutzt. Zwei haben die Bilder indirekt von hinten und zwei weitere haben die Motive direkt beleuchtet.

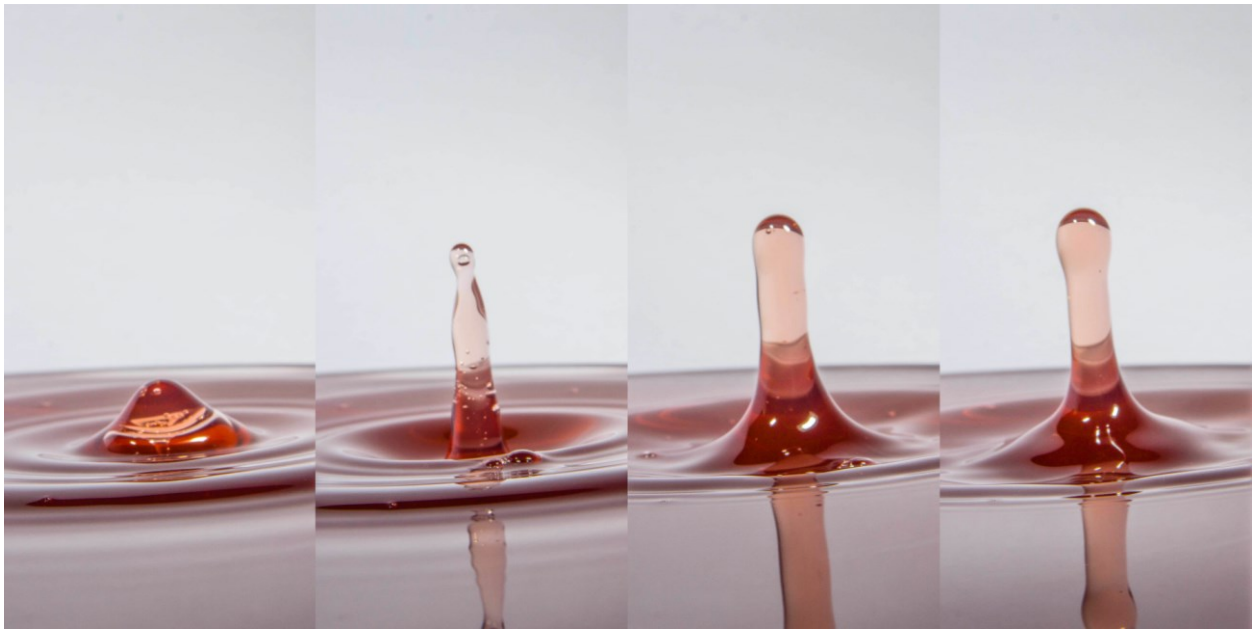

Abbildung 42: Flüssigkeitssäule mit dem Arduino.

Nachdem die Timings für die Flüssigkeitssäule fixiert sind, wird das in Listing 6 hervorgehobene *delay()* erneut angepasst, um einen Flüssigkeitstropfen zu erzeugen, der circa den Abstand der Säulenhöhe zur Flüssigkeitsoberfläche hat. Als Startwert wird 240 Millisekunden genommen, da dies bereits beim Vortest einen Tropfen erzeugt hat, der sich circa auf dieser Höhe befindet. Die Bilderreihe in Abbildung 43 zeigt die dabei entstandenen Bilder mit einer schrittweisen Verlängerung der Wartezeit, bis zum Auslösen der Blitze von links nach rechts. Für den Tropfen wurde eine ideale Verzögerung von 245 Millisekunden festgestellt. Damit liegt zwischen den beiden Tropfen eine Gesamtverzögerung von 105 Millisekunden.

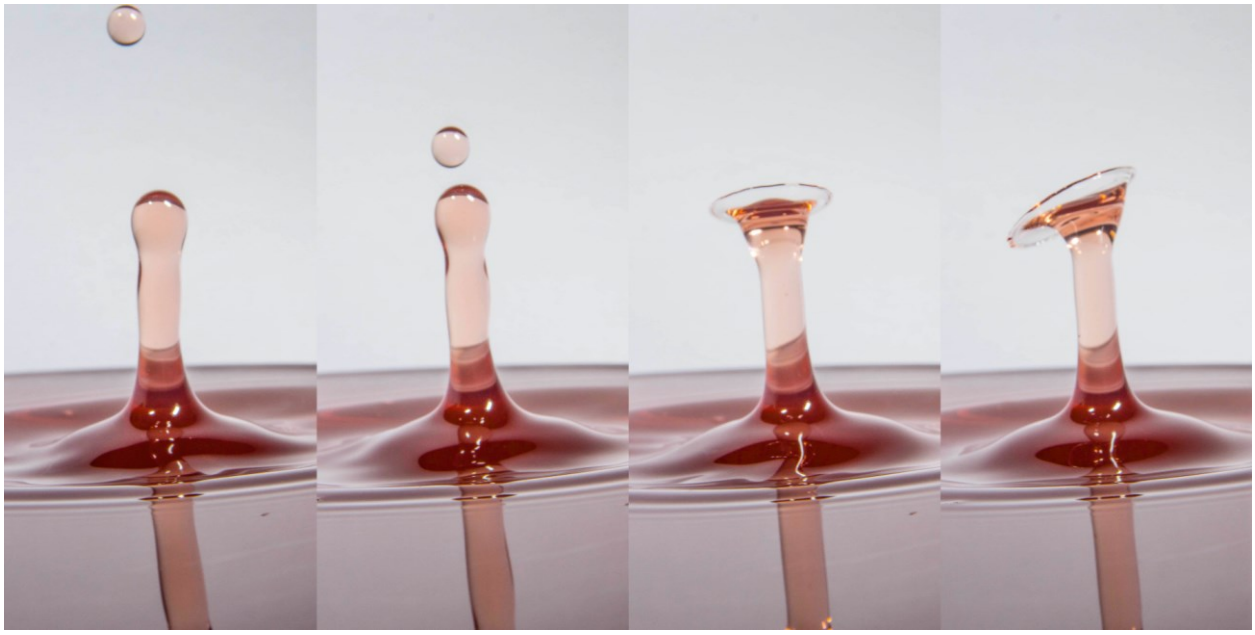

Abbildung 43: Tropfen-Säule Kalibrierung mit dem Arduino

Abbildung 44 zeigt ein Bild mit dem Endergebnis der Kalibrierung. Das Bild zeigt eine leichte Pilzform, da die Flüssigkeitssäule zum Zeitpunkt des Aufpralls noch nicht ihre maximale Größe

erreicht hatte. Da dieses Ergebnis sehr zufriedenstellend ist, werden die dafür genutzten Einstellungen mit einer Verzögerung von 105 Millisekunden zwischen den beiden Tropfen für die nachfolgende Testreihe genutzt.

Abbildung 44: Ergebnis der Kalibrierung für den TaT-Test mit dem Arduino.

Zur Erfassung der Zuverlässigkeit bei der Erstellung der TaT-Fotos wurde mit dem Arduino eine Testreihe mit der Größe von 100 Bildern erzeugt. Zur Durchführung der Testfälle wird ein Programm erstellt, das die zuvor festgelegten Timings nutzt, um die Testbilder zu erstellen.

6.6 Wassersäule mit Tropfeneinschlag unter Verwendung des Raspberry Pi

Für die Erzeugung der TaT-Bilder mit dem Raspberry Pi wird die gleiche Vorgehensweise verfolgt, wie mit dem Arduino. Zuerst wird das Timing für die Flüssigkeitssäule ermittelt. Dazu wird, ausgehend vom Kalibrierungsprogramm für den Raspberry Pi, das Timing des Tropfenfalls schrittweise angepasst, bis die richtige Zeitverzögerung für die Säule feststeht. Auf der nachfolgenden Abbildung 45 ist von links nach rechts die Anpassung der Timings zu sehen. Das ganz rechte Bild wurde mit einer Verzögerung von 105 Millisekunden aufgenommen. Dies entspricht derselben zeitlichen Verzögerung, die auch für das Arduino-Board genutzt wird. Dass die zeitliche Verzögerung für beide Geräte gleich ist, korrespondiert auch mit der Erwartung für die Timings, da die benötigte Fallzeit der Tropfen unabhängig von den benutzten Geräten ist.

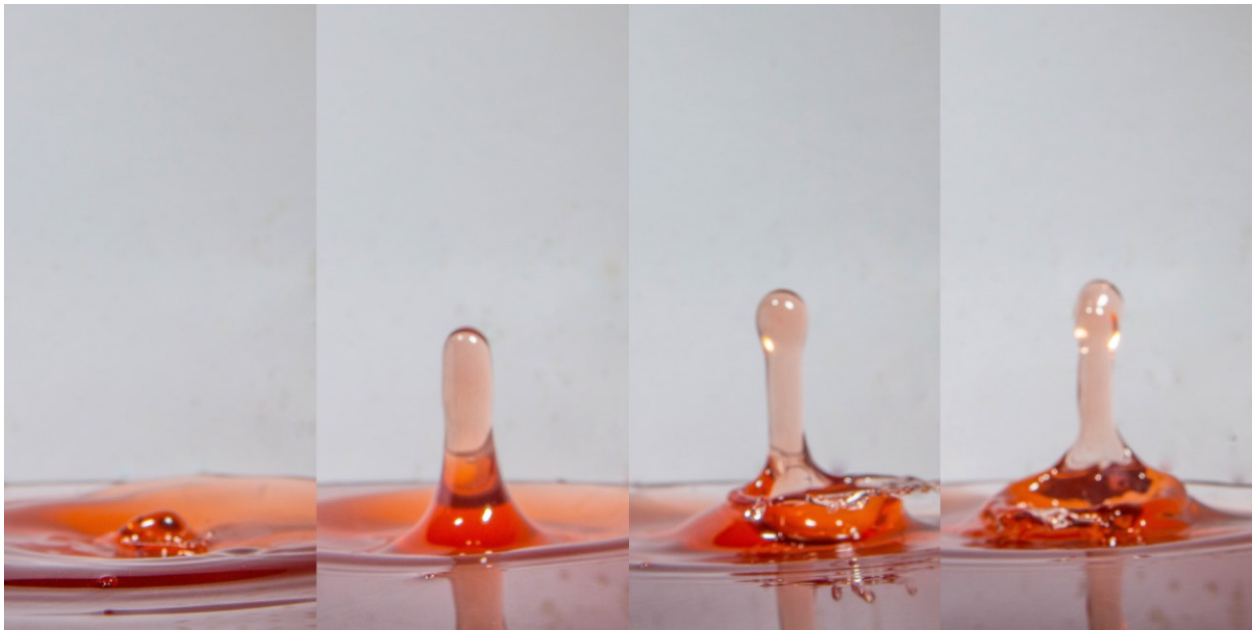

Abbildung 45: Flüssigkeitssäule mit einem Raspberry Pi.

Ausgehend von den bisher gewonnenen Erkenntnissen werden die Timings für das Raspberry Pi Programm bereits so eingestellt, dass im Idealfall keine weitere Kalibrierung des Systems notwendig sein sollte.

Für die Kalibrierung werden deswegen Timing-Werte von 245 Millisekunden voreingestellt und danach schrittweise nachjustiert, falls dies notwendig sein sollte. Die in Abbildung 46 gezeigten Fotos wurden direkt mit den voreingestellten Timings gemacht. Da bereits auf allen Kalibrierungsbildern ein TaT zu sehen ist, werden die Timings nicht weiter angepasst und es wird direkt mit der Testreihe begonnen. Das Programm mit den finalen Einstellungen für den Raspberry Pi ist in Anhang D angeführt.

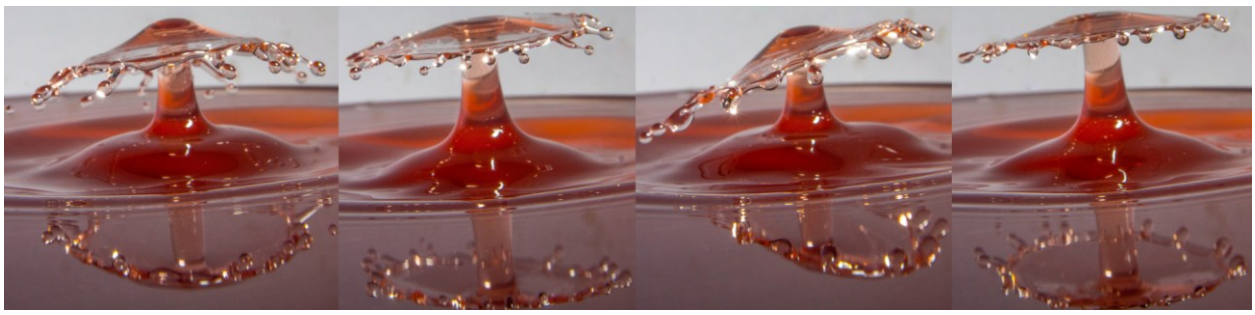

Abbildung 46: Kalibrieren des Tropfeneinschlags auf der Flüssigkeitssäule.

Für die Testreihe wird, gleich wie für den Arduino, ein eigenes kleines Programm geschrieben, das diesen Test durchführt. Die Testreihe besteht ebenfalls aus 100 Bildern. Die Bilder werden im Anschluss an die Testreihe manuell gesichtet, um fehlgeschlagene Bilder zu identifizieren. Die Bilder, die nicht das gewünschte Ergebnis ausweisen, können denen aus der Testreihe mit dem Arduino gegenübergestellt werden, um zu bewerten wie sich der Raspberry Pi im Vergleich zum Arduino verhält. Weiter kann beurteilt werden, ob die Limits für eine weiche Echtzeit eingehalten werden.

7 VERGLEICHEN DER ERGEBNISSE UND DISKUSSION DES ERKENNTNISGEWINNS

Für jeden der durchgeführten Tests wurden bereits die Ergebnisse in Kapitel 6 erfasst und dokumentiert. Hier werden diese Ergebnisse noch einmal aufgegriffen, zusammengefasst, eingehend analysiert, gegenübergestellt und interpretiert. Aus den Ergebnissen wird eine Schlussfolgerung gezogen, mit der eine der Hypothesen aus Kapitel 1.2 verworfen wird. Im Anschluss daran werden die gewonnenen Erkenntnisse genutzt, um die Forschungsfrage zu beantworten und einen Ausblick darauf zu geben, wie diese Ergebnisse weiter genutzt werden könnten.

7.1 Gegenüberstellung der Ergebnisse aus der Kalibrierung der Systeme

Dieser Test wurde für beide Systeme durchgeführt, um einerseits die Performance der Systeme zu testen und andererseits die Reproduzierbarkeit und Gleichmäßigkeit der Performance zu bestimmen. Weiter dient dieser Test als Grundlage für den zweiten Test, der diesen als Grundlage nutzt. Um die höchstmögliche Performance zu ermitteln, wurde versucht einen Tropfen im freien Fall zu fotografieren.

Die Bilder wurden mit einer Auflösung von 24 Megapixel gemacht. Dies entspricht einer Auflösung von 6000 x 4000 Pixel im Querformat. Das genutzte Tamron Objektiv hat bei einem Abstand von 33 cm zum Objekt bei einer maximalen Brennweite von 75 mm im Kleinbildformat einen Abbildungsmaßstab von 1:3,9. Durch die Verwendung eines APS-C Sensors ist der Bildausschnitt nochmals um den Faktor 1,5 vergrößert. Damit wird ein Abbildungsmaßstab von 5,85 erreicht. Durch diesen sehr großen Abbildungsmaßstab sind bereits geringfügige zeitliche Verzögerungen zu erkennen, da der Tropfen sofort eine um ein paar Pixel verschobene Position hat. Für die Tropfen wurde, wie in Kapitel 5.2 beschrieben, eine mittige Position zwischen der Wasseroberfläche und der oberen Bildkante gewählt. Die Messung der Bilder hat ergeben, dass sich die Wasseroberfläche in dem Bereich, in dem der Tropfen aufschlägt, 3324 Pixel unter der Bildschirmoberkante befindet. Bei einer mittigen Kalibrierung wird also eine Position von 3324 / 2 = 1662 Pixel angestrebt. Dies bedeutet, dass die höchste Dichte im Idealfall bei 1662 Pixel auftreten sollte.

Zur Feststellung des Verhaltens wurde für jedes System eine Testreihe durchgeführt. Die Ergebnisbilder wurden vermessen und in Abbildung 47 angeführt. Das daraus erstellte Box-Plot Diagramm zeigt, dass die Tropfen, die mit dem Arduino fotografiert wurden, unregelmäßiger und weiter verstreut sind, als mit dem Raspberry Pi. Der Arduino hatte aber nur einen Ausreißer mit einer Abweichung von 381 Pixel zum arithmetischen Mittel. Der Raspberry Pi hatte, sowohl nach unten also auch nach oben hin, eine geringere Streuung. Jedoch gibt es beim Raspberry Pi mehr Ausreißer und diese sind zum Teil auch weiter vom arithmetischen Mittel entfernt. Der Mittelpunkt

selbst liegt beim Raspberry Pi etwas höher, als beim Arduino. Dies bedeutet, dass der Raspberry Pi im Durchschnitt etwas langsamer reagiert hat, als der Arduino.

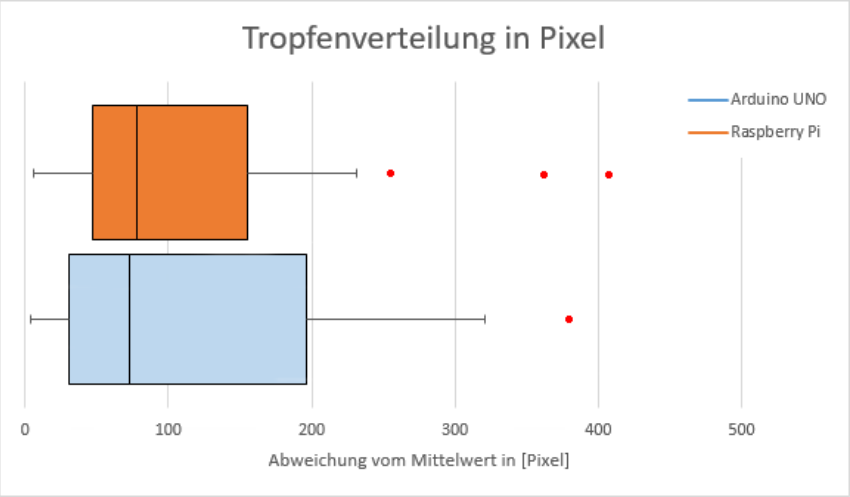

Abbildung 47: Vergleich der Latenzstreuung zwischen Arduino und Raspberry Pi.

Insgesamt lässt sich aus diesem Diagramm schlussfolgern, dass beide Systeme eine sehr gute Performance aufweisen. Beim Raspberry Pi ist zwar mit einer höheren Ausreißer-Rate zu rechnen, dies spielt aber für Systeme, die eine weiche Echtzeit anstreben, keine Rolle. Beim Arduino ist die Schwankungsbreite etwas größer, aber die Ergebnisse sind dadurch auch besser verteilt. Dies wiederum lässt derzeit die Schlussfolgerung zu, dass der Raspberry Pi unter Verwendung eines entsprechenden Betriebssystems sogar besser geeignet ist als der Arduino weiche Echtzeitanwendungen auszuführen.

Die zweite Betrachtung, die bei diesem Test angestellt wurde, betrifft die absolute Performance. Auch wenn eine Echtzeitanwendung lediglich ein möglichst gleichmäßiges Ergebnis voraussetzt, so geht es bei der Highspeed-Fotografie zusätzlich noch um Geschwindigkeit. Es ist nicht nur notwendig, dass der Blitz schnell arbeitet, um ein scharfes Foto zu erstellen. Auch die Steuergeräte müssen ein Mindestmaß an Geschwindigkeit aufweisen, damit beispielsweise der Blitz zur richtigen Zeit ausgelöst wird.

Das in Abbildung 48 gezeigte Diagramm ist eine Überlappung der Diagramme aus den TaT-Tests mit dem Arduino und dem Raspberry Pi. Dieses Diagramm zeigt, dass die Spitzen der Kurven eine Verschiebung um circa 550 Pixel aufweisen. Wie bereits bei den Testfällen selbst erklärt wurde, sind die absolut gleichen Einstellungen für beide Systeme benutzt worden. Die Verschiebung dieser Kurven zeigt deswegen, dass die Tropfen, die mit dem Raspberry Pi gemacht wurden, im Schnitt circa 550 Pixel mehr Abstand zum oberen Bildrand besitzen. Dies bedeutet, dass der Raspberry Pi im Vergleich zum Arduino meistens langsamer war. Die absolute Bestmarke, die der Raspberry Pi zustande gebracht hat, liegt performancetechnisch gerade einmal im Durchschnitt des Arduino.

Damit ist der Arduino immer dann im Vorteil, wenn es besonders zeitkritische Anwendungsfälle gibt, wie beispielsweise ein TaT-Foto, bei dem es nur möglich ist mit einer Lichtschranke zu arbeiten. Wenn dieser Lichtschranken zu kurz vor dem Aufschlag positioniert ist, kann es sein, dass die Reaktionsfähigkeit des Raspberry Pi nicht ausreichend ist.

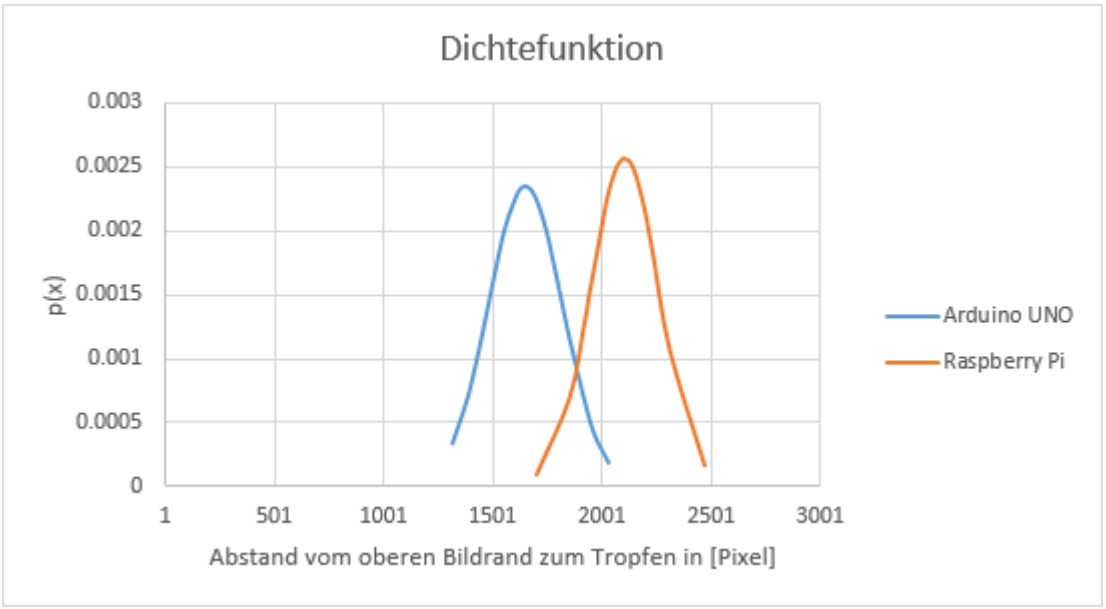

Abbildung 48: Vergleich der Tropfenverteilung zwischen Arduino und Raspberry Pi.

Abschließend lässt sich zu diesem Testfall festhalten, dass es mit dem Raspberry Pi besser möglich ist gleichmäßige Ergebnisse zu produzieren, so lange genug Zeit vorhanden ist, damit er reagieren kann. In extrem zeitkritischen Anwendungsfällen kann es hingegen vorkommen, dass der Raspberry Pi nicht schnell genug reagiert, was wiederum dafürspricht, in solchen Situationen auf einen Microcontroller wie den Arduino zu setzen.

7.2 Gegenüberstellung der Ergebnisse aus dem TaT-Test

Der TaT-Test wurde durchgeführt, um etwas mehr Dynamik in das Test Setting zu bekommen. Bei diesem Test geht es darum mehrere zeitkritische Aktoren miteinander so zu timen, dass ein fallender Tropfen genau auf einer stehenden Flüssigkeitssäule auftrifft. Dies ist ein typisches und sehr oft fotografiertes Szenario. Zusätzlich lässt sich festhalten, dass dieses Setting ausreichend ist, um ein Gesamturteil für die Tropfenfotografie abzugeben, da alle weiteren Szenarien, die beispielsweise von (Nimmervoll, 2014) auch schon besprochen wurden, nur mehr eine Aneinanderreihung von zusätzlichen Aktoren sind. Für das System ist es irrelevant ob es ein, zwei oder noch mehr Ventile zeitgleich steuern muss. Wenn die Latenz für den Tat-Test ausreichend ist, reicht sie auch für komplexere Szenarien wie XXL-TaT's oder ähnliche.

Für den TaT-Test wurde eine Testreihe mit 100 Bildern je Gerät gefertigt. Die Auswertung wurde manuell vorgenommen. Es wurde jedes Bild in Augenschein genommen und geprüft. Zur Bewertung der Bilder wurde die Form der TaT's zu Grunde gelegt. Die Schärfe wurde nicht beurteilt, da ein unscharfes Bild nicht als Fehler des Kontrollers zu werten ist. Unscharfe Bilder können bei TaT's beispielsweise entstehen, wenn das Objektiv nicht richtig eingestellt ist. Da dies aber manuell erfolgt, ist es kein fehlerhaftes Verhalten des Systems.

Was als Fehler in den Bildern gewertet wird sind fehlerhaftes Auslösen der Blitze, ein Vorhandensein eines Teils des Verschlusses auf dem Foto und auch ein gänzliches Verfehlen des Motivs, wenn beispielsweise kein TaT auf dem Bild zu sehen ist. Weiter wird es auch als

Fehler gewertet, wenn der Tropfen noch nicht oder nur so leicht auf die Säule aufgetroffen ist, dass noch kein umlaufendes Schirmchen zu erkennen ist. Wenn ein Schirmchen bereits die Wasseroberfläche berührt, wird dies als zu langsames Auslösen und damit Verfehlen des Bildes gewertet.

Der Arduino hat es geschafft, dass 100% aller gemachten Bilder einen schönen TaT aufweisen. Er kann also uneingeschränkt für die Highspeed-Fotografie genutzt werden. Beim Raspberry Pi gibt es drei Bilder, die nicht den Anforderungen entsprechen. Er hat also ein Ergebnis von 97% abgeliefert. In Kapitel 3.3.5 wurde festgehalten, dass keine harte Echtzeit mit dem Raspberry Pi möglich ist und da auch die Anwendungsfälle in der Highspeed-Fotografie nicht kritisch sind, wurde das Erreichen einer weichen Echtzeitfähigkeit als ausreichend bezeichnet. Mit 97% liegt der Raspberry Pi eindeutig über den geforderten 95% und hat somit den Test ausreichend gut geschafft.

7.3 Verwerfen einer Hypothese und Beantworten der Forschungsfrage

Die Nullhypothese wurde in Kapitel 1.2 wie folgt definiert.

> *Die Nullhypothese besagt, dass es trotz Optimierung eines Embedded Systems nicht möglich ist, mit diesem im hoch zeitkritischen Anwendungsbereich der Highspeed Fotografie alle Sensoren und Aktoren in Echtzeit auszulesen und anzusteuern.*

Bereits die Tests zu den Betriebssystemen in Kapitel 4.3 haben gezeigt, dass sich das Verhalten des Systems durch die Optimierungen stark verändert hat. Der Test in Kapitel 6.3 hat gezeigt, dass es bereits mit einem Raspberry Pi ohne RT-Kernel möglich ist Tropfen zu fotografieren, jedoch haben lediglich 80% der Fotos den vorher definierten Anforderungen entsprochen. Somit wird die Schlussfolgerung gezogen, dass es nicht möglich ist einen Raspberry Pi ohne RT-Patch sinnvoll in der Highspeed-Fotografie einzusetzen. Wie in den Kapiteln 7.1 und 7.2 aufgezeigt, ist es mit einem Raspberry Pi, nach einer Optimierung möglich, Fotos in der Highspeed-Fotografie zu machen. Aus diesem Grund wird die Nullhypothese verworfen.

Die in Kapitel 1.2 definierte Alternativhypothese lautet wie folgt:

> *Die Alternativhypothese besagt, dass es durch die Optimierung eines Embedded Systems möglich ist, mit diesem im hoch zeitkritischen Anwendungsbereich der Highspeed Fotografie alle Sensoren und Aktoren in Echtzeit auszulesen und anzusteuern.*

Da es mit dem Abschluss dieser Arbeit nicht gelungen ist diese zu widerlegen, bleibt sie aufrecht und hat weiterhin ihre Gültigkeit.

Damit es möglich ist mit einem Raspberry Pi verschiedene Sensoren und Aktoren über digital In- und Outputs in Echtzeit auszulesen und anzusteuern, um damit zeitkritische Echtzeitanwendungen auszuführen, sind die nachfolgenden Restriktionen ausschlaggebend.

Um die Aktoren auszulesen und anzusteuern, muss eine Sprache genutzt werden, in der entweder bereits Libraries zu deren Steuerung zur Verfügung stehen oder es muss selbst eine Software dafür geschrieben werden. Die Sprache Python bietet sich an, um auf die digitalen GPIO-Pins zuzugreifen, da dafür Libraries zur Verfügung stehen.

Weiter ist es notwendig, dass der Kernel des Systems einen Echtzeitpatch erhält, damit mögliche Schwankungen in der Reaktionszeit des Systems möglichst klein gehalten werden. Für die meisten Linux Kernels stehen sogenannte RT-Patches zur Verfügung. Für den Raspberry Pi stehen außerdem fertige Images im Internet zur Verfügung, die auf Raspbian basieren und bereits einen Kernel mit RT-Patch enthalten. Wenn diese Voraussetzungen erfüllt sind, kann der Raspberry Pi für RT-Anwendungen genutzt werden. Es bestehen aber weiterhin zwei Einschränkungen:

Die erste Einschränkung betrifft die Art der Echtzeitfähigkeit. Es kann nicht garantiert werden, dass das System immer innerhalb der notwendigen Parameter läuft. Deswegen ist keine harte Echtzeit möglich. Es kann nur eine weiche Echtzeit erreicht werden, bei der gelegentliche Ausreißer in Kauf genommen werden müssen.

Die zweite Einschränkung betrifft die maximale Latenz. Wie bereits die Auswertung des Tests in Kapitel 7.1 gezeigt hat, ist der Raspberry Pi langsamer als der Arduino. Wenn die maximal zumutbare Latenz für eine Anwendung sehr niedrig ist, kann es sein, dass der Raspberry Pi nicht in der Lage, ist innerhalb der notwendigen Parameter zu laufen. In diesem Fall muss weiterhin ein Mikrocontroller zur Steuerung genutzt werden.

7.4 Ausblick

Weiterführend können die verwendeten Programme so umgebaut werden, sodass sie durch einen Zustandsautomaten (state machine) abgebildet werden. Bei einem endlichen Automaten handelt es sich um ein System, das sich immer in einem bestimmten Zustand befindet. Der Automat kann seinen Zustand verändern. Dazu werden Zustandsübergänge, sogenannte Transitions, genutzt. In diesen Transitions können verschiedene Aktionen durchgeführt werden. Diese Automaten haben den Vorteil, dass sie leicht zu warten sind und dass es damit möglich ist ein System zu kreieren, das niemals wartet oder stehen bleibt. (Hopcroft & Ullman, 1190)

Die Zustände des Automaten könnten wie folgt aussehen:

1. Ready
2. Wait
3. ActorOneExecuted
4. Wait
5. ActorTwoExectued
6. Wait
7. ActorThreeExecuted

Die Übergänge zu diesen Status würden wie folgt aussehen:

Status zu Beginn	*Aktion*	*Nächster Status*
1	fireDrop(1)	2
2	Wait(Time t1)	3
3	fireDrop(2)	4
4	Wait(Time t2)	5
5	executeShutter()	6
6	Wait(Time t3)	7
7	executeBulb()	1

Tabelle 5: Übergänge des potentiellen Zustandsautomaten.

Die in Tabelle 5 angeführten Übergänge können genutzt werden, um zwischen den einzelnen Status zu wechseln und dabei Aktionen durchzuführen. Direkt nach dem Start befindet sich der Automat im Status eins. Es wird der Befehl zum Fallenlassen eines Tropfens aufgerufen und ein Statuswechsel durchgeführt. Beim nächsten Durchlauf befindet sich der Automat in Status zwei in diesem Status wird keine Aktion durchgeführt. Der Automat bleibt in diesem Status bis eine vorher definierte Zeitspanne verstrichen ist. Nach dem Verstreichen der Zeitspanne wechselt der Automat in den Status drei und lässt erneut einen Tropfen fallen. Es wird ein Wechsel in den vierten Status durchgeführt in dem wieder gewartet wird bis ein Wechsel in den nächsten Status erfolgt. Beim fünften Durchlauf der Schleife wird die Aktion zum Auslösen des Verschlusses der Kamera aufgerufen und nach einer neuerlichen Wartephase wird in den letzten Status gewechselt. Nach einem weiteren Durchlauf wird der Blitz ausgelöst und der Automat kehrt wieder in seinen Ursprungsstatus zurück. Dieser Ablauf hat einen gesamten Durchlauf des Automaten gezeigt.

Der Zustandsautomat befindet sich bei der Ausführung ständig im Durchlauf seiner Hauptschleife. Er bleibt zu keinem Zeitpunkt stehen oder wartet auf ein Event. Bei dieser Darlegung des Automaten würde er ohne zeitliche Verzögerung alle Befehle und Statuswechsel schnellstmöglich durchführen. Damit die Timings eingehalten werden, gibt es die wait Stati. In diesen wird geprüft wie viel Zeit seit dem Eintritt in den Status verstrichen ist. Wenn eine vordefinierte Zeitspanne erreicht wurde erfolgt ein Statuswechsel in den nächsten Status.

Damit kann beispielsweise nach dem Erreichen von Status 2 gewartet werden, bis die 10 Millisekunden, die in den vorangegangenen Tests zwischen den Tropfen gewartet wurde, bereits vergangen sind, um dann den zweiten Tropfen auszulösen und in den nächsten Status zu wechseln. Damit können *delays()* umgesetzt werden ohne, dass das System jemals stehen bleiben muss und den Zugriff auf andere Aktoren verhindert. Nach dem Umbau der Programme kann damit begonnen werden einen funktionsfähigen Prototyp zu bauen.

Der Raspberry Pi kann mit der Einschränkung auf weiche Echtzeitanwendungen im Bereich der Highspeed-Fotografie eingesetzt werden. Im Vergleich zum Arduino ist er etwas langsamer in der Ausführung von Anweisungen. Aus diesem Grund kann es vorkommen, dass er für Settings bei

denen es auf eine sehr kurze Latenzzeit ankommt, nicht geeignet ist. Durch seine Vorteile im Bereich der Konnektivität und der einfacheren Bedienbarkeit mit einer grafischen Oberfläche eignet er sich besser als der Arduino für Anwender ohne technischem Hintergrund. Wenn Adaptionen an den Einstellungen notwendig sind, kann dies direkt am Arduino durchgeführt werden, sofern er ein Eingabe- und Anzeigegerät, wie einen Touch Screen, besitzt. Weiter hat der Raspberry Pi die Möglichkeit viele verschiedene Programme auf einmal zu speichern. Der Benutzer kann im Internet nach Programmen suchen und diese dann ausprobieren, ohne dass ein anderes Programm überschrieben werden muss. Der Arduino kann immer nur einen Programmablauf in seinem Speicher halten. Wenn verschiedene Programme genutzt werden sollen oder wenn Anpassungen am Programm notwendig sind, muss der Arduino jedes Mal mit einer speziellen Software neu programmiert werden.

Bei einem technisch versierten Benutzer ist es egal welches Gerät benutzt wird. Es können beide Geräte im Bereich der Highspeed-Fotografie eingesetzt werden. Leichter in der Anwendung ist der Raspberry Pi und ist für Nutzer ohne technische Vorkenntnisse die bessere Wahl. Wen es nicht stört, dass das genutzte Gerät für jede Änderung an ein Notebook angesteckt werden muss und dass Änderungen nur durch Programmieren möglich sind, sollte zum Arduino greifen. Dieser bietet die bessere Latenz, ist für harte Echtzeitanwendungen geeignet und ist in der Anschaffung das kostengünstigere Gerät.

ANHANG A - Cyclic Test Manpage

pi@raspberrypi ~/rt-tests $ sudo ./cyclictest --help
cyclictest V 0.42
Usage:
cyclictest <options>

-a	[NUM]	--affinity	run thread #N on processor #N, if possible
			with NUM pin all threads to the processor NUM
-b	USEC	--breaktrace=USEC	send break trace command when latency > USEC
-B		--preemptirqs	both preempt and irqsoff tracing (used with -b)
-c	CLOCK	--clock=CLOCK	select clock
			0 = CLOCK_MONOTONIC (default)
			1 = CLOCK_REALTIME
-C		--context	context switch tracing (used with -b)
-d	DIST	--distance=DIST	distance of thread intervals in us default=500
-E		--event	event tracing (used with -b)
-f		--ftrace	function trace (when -b is active)
-i	INTV	--interval=INTV	base interval of thread in us default=1000
-I		--irqsoff	Irqsoff tracing (used with -b)
-l	LOOPS	--loops=LOOPS	number of loops: default=0(endless)
-m		--mlockall	lock current and future memory allocations
-n		--nanosleep	use clock_nanosleep
-N		--nsecs	print results in ns instead of ms (default ms)
-o	RED	--oscope=RED	oscilloscope mode, reduce verbose output by RED
-O	TOPT	--traceopt=TOPT	trace option
-p	PRIO	--prio=PRIO	priority of highest prio thread
-P		--preemptoff	Preempt off tracing (used with -b)
-q		--quiet	print only a summary on exit
-r		--relative	use relative timer instead of absolute
-s		--system	use sys_nanosleep and sys_setitimer
-T	TRACE	--tracer=TRACER	set tracing function
			configured tracers: unavailable (debugfs not mounted)
-t		--threads	one thread per available processor
-t	[NUM]	--threads=NUM	number of threads:
			without NUM, threads = max_cpus
			without -t default = 1
-v		--verbose	output values on stdout for statistics
			format: n:c:v n=tasknum c=count v=value in us
-D		--duration=t	specify a length for the test run

		default is in seconds, but 'm', 'h', or 'd' maybe added to modify value to minutes, hours or days
-h	--histogram=US	dump a latency histogram to stdout after the run US is the max time to be be tracked in microseconds
-w	--wakeup	task wakeup tracing (used with -b)
-W	--wakeuprt	rt task wakeup tracing (used with -b)

ANHANG B - Test Setting build-up Programm

```
void setup() {
  // Logoutput on the serial port
  Serial.begin(9600);

  // all used pins are signal output pins
  pinMode(12, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(2, OUTPUT);

  // set all pins to low befopre the test begins
  digitalWrite(12, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(2, LOW);
}

void loop() {
// Relaise 1
digitalWrite(8, HIGH);
delay(50);
digitalWrite(8, LOW);

// Relaise 2
digitalWrite(7, HIGH);
delay(50);
digitalWrite(7, LOW);

// Shutter
digitalWrite(2, HIGH);
digitalWrite(2, LOW);

// Bulb
digitalWrite(12, HIGH);
digitalWrite(12, LOW);

 delay(5000);
}
```

ANHANG C - Programm zum Kalibrieren des Arduino

```
void setup() {
  // Logoutput on the serial port
  Serial.begin(9600);

  // all used pins are signal output pins
  pinMode(12, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(2, OUTPUT);

  // set all pins to low befopre the test begins
  digitalWrite(12, LOW);
  digitalWrite(7, LOW);
  digitalWrite(2, LOW);
}

void loop() {
// Relaise 1
digitalWrite(7, HIGH);
delay(40);
digitalWrite(7, LOW);

// wait before shutter
delay(240);
// Shutter
digitalWrite(2, HIGH);
delay(10);
digitalWrite(2, LOW);
// wait before bulb
delay(110);
// Bulb
digitalWrite(12, HIGH);
digitalWrite(12, LOW);

delay(5000);
}
```

ANHANG D - Programm zum Kalibrieren des Raspberry Pi

```
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(35, GPIO.OUT) #valve1
GPIO.setup(36, GPIO.OUT) #valve2
GPIO.setup(37, GPIO.OUT) #shutter
GPIO.setup(38, GPIO.OUT) #bulb
while 1:
      GPIO.output(35, False)
      GPIO.output(36, False)
      GPIO.output(37, False)
      GPIO.output(38, False)
      sleep(10)
      GPIO.output(35, True)
      sleep(0.04)
      GPIO.output(35, False)
      sleep(0.245)
      GPIO.output(37, True)
      sleep(0.01)
      GPIO.output(37, False)
      sleep(0.11)
      GPIO.output(38, True)
      GPIO.output(38, False)
```

ABKÜRZUNGSVERZEICHNIS

APS-C – Active Pixel Sensor – Crop
CMOS – Complementary metal oxide semiconductor
DSLR – Digital Single Lense Reflex
GPIO - General Purpose Input/Output
HSS – High Speed Synchronisation
Hz - Herz
IC – Integrated Circuit
Kb – Kilobyte
MIPS – Million instructions per secont
OS – Operating System
PWM – Pulse Width Modulation
RISC - Reduced Instruction Set Computer
RTC – Real time clock
RTOS – Real time operating system
TaT – Tropfen auf Tropfen
USB - Universal Serial Bus

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGVERZEICHNIS

LITERATURVERZEICHNIS

Arduino. (2016, 11 20). *Arduino - Delay Microseconds*. Retrieved from Arduino: https://www.arduino.cc/en/Reference/DelayMicroseconds

Arduino. (2016, 11 03). *Arduino Mega - Technical Specs*. Retrieved from Arduino: https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Arduino. (2016, 11 02). *Arduino UNO - Technical Specs*. Retrieved from Arduino: https://www.arduino.cc/en/Main/ArduinoBoardUno

Arduino S.R.L. (2016, 12 02). *www.arduino.org*. Retrieved from Relayshield : http://www.arduino.org/products/shields/arduino-4-relays-shield

Artmann, U. (2013, 06 26). Blende, Lichtstärke und Schärfentiefe erklärt. *Colorfoto*.

Atmel Corporation. (2016, 11 02). 8-bit AVR Microcontrollers ATmega328/p. *Datasheet Complete*. San Jose, CA, USA: Atmel Corporation.

Barrett, S. (2010). *Arduino Microcontroller: Processing for Everyone! Part I.* Morgan & Claypool.

Bartmann, E. (2011). *Die elektronische Welt mit Arduino entdecken.* Köln: O'REILLY.

Brühlmann, T. (2015). *Arduino Praxiseinstieg.* Heidelberg: mitp.

Buttazzo, G., Lipari, G., Abeni, L., & Caccamo, M. (2005). *Soft Real-Time Systems - Predictability vs. Efficiency.* New York, USA: Springer.

Chang, H.-Y. (1987). Dynamic scheduling algorithms for distributed soft real-time systems. *Band 728 von Computer sciences technical report*. University of Wisconsin - Madison.

Courington, B., & Collins, G. (2016, 10 16). *Getting Started with Java SE Embedded on the Raspberry Pi*. Retrieved from Oracle.com: http://www.oracle.com/technetwork/articles/java/raspberrypi-1704896.html

Di Sirio, G. (2016, 11 29). *ChibiOS*. Retrieved from www.chibios.org: http://www.chibios.org/dokuwiki/doku.php?id=chibios:product:rt:features

Diehl, B., Erb, R., Lindner, K., Schmalhofer, C., Schön, L.-H., Tillmanns, P., & Winter, R. (2001). *Physik Oberstufe Gesamtband.* Mecklenburg-Vorpommern: Cornelsen.

DIN ISO 517:2009-05. (2009, 05). *Photographie - Öffnungsverhältnisse und verwandte Größen bei Photoobjektiven - Bezeichnungen und Messungen*.

EMLID LIMITED. (2016, 11 23). *EMLID*. Retrieved from Docs: https://docs.emlid.com/navio/Downloads/Real-time-Linux-RPi2/

Furht, B., Grostick, D., Gluch, D., Rabbat, G., Parker, J., & McRoberts, M. (1991). *Real-Time UNIX SYSTEMS - Design and Application Guide.* New York: Springer Science + Business Media, LLC.

George, C. (2008). *Blitzlicht-Fotografie.* Markt + Technik.

Gockel, T. (2014). *Entfesseltes Blitzen.* Bonn: Galileo Design.

Grau, O., & Pansiot, J. (2012). Motion and velocity estimation of rolling shutter cameras. *CVMP '12 Proceedings of the 9th European Conference on Visual Media Production* (pp. 94 - 98). New York: ACM.

Grimvall, G. (2007). *Brainteaser Physics Puzzlers.* Baltimore, Maryland: Johns Hopkins University Press.

Gross, S. (2009). *Das Profi-Handbuch zur Canon EOS 500D.* Düsseldorf: Data Becker.

Gwinn, J. (2005, October). Quality-of-Service versus Realtime. *ACM SIGOPS Operating Systems Review*, p. Volume 39 Issue 4.

Haasz, C. (2012). *Profibuch Sony - SLT-a65 SLT-a77.* Haar bei München: Franzis.

Hay, A., & Pritschow, K. (1931). *Handbuch der Wissenschaftlichen und Angewandten Photographie.* Wien: Julius Springer.

Hogl, M. (2016). *Digitale Fotografie.* Köln: Vierfarben (Rheinwerk).

Hopcroft, J., & Ullman, J. (1190). *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie.* Bonn: Addison-Weseley.

IEEE. (2016). IEEE Std 1003.1-2008, 2016 Edition. *IEEE Std 1003.1-2008, 2016 Edition*. IEEE.

Jeffay, K., & Zhao, W. (1996). IEEE Real-Time Technology and Applications Symposium. *1996 IEEE Real-Time Technology and Applications Symposium: proceedings, June 10-12* (p. 263). Brookline, Massachusetts: IEEE.

Jorns, A. (2012). *Das Blitzkochbuch.* Heidelberg: dpunkt.

Kappel, B. (2016). *Arduino.* Bonn: Rheinwerk Computing.

Kelby, S. (2007). *Digitale Fotografie - Das große Buch.* München: Addison-Wesley.

Kofler, M., Kühnast, C., & Scherbeck, C. (2016). *Raspberry Pi - Das umfassende Handbuch.* Bonn: Rheinwerk Computing.

Kuehnel, C. (2015). *Raspberry Pi: Erfassen von Umweltdaten.* Altendorf: Skript Verlag.

Laplante, P. (February 2006). It Isn't Your Father's Realtime Anymore. *ACM QUEUE*, 63, 64.

M. Kuo, S., & H. Lee, B. (2001). *Real-Time Digital Signal Processing.* Illinois, USA: Wiley.

Monaghan, D., O'Connor, N., Cleary, A., & Connolly, D. (2015). The Real Time Rolling Shutter. *MM '15Proceedings of the 23rd ACM international conference on Multimedia* (pp. 731 - 734). New York: ACM.

Nimmervoll, D. (2014). *Highspeedfotografie Kunstvolle Tropfenfotos in Perfektion.* Heidelberg: mitp.

ODE. (2007, 11 07). Datasheet. *Solenoid valve 2/2 way N.C. Direct acting*. ODE.

Odendahl, M., Finn, J., & Wenger, A. (2009). *Arduino - Physical Computing für Bastler, Designer und Geeks.* Köln: O'REILLY.

Otto, B., & Hillebrand, R. (2013). *Carl Zeiss Kamera-Register 1902-2012.* Verlag Rudolf Hillebrand.

Peterson, B. (2011). *Petersons Fotoschule - Der Einstieg.* München: Markt + Technik.

Pixel. (2010). User Manual. Pixel.

Ras, J. (2016). *Real-Time Embedded Systems.* Lulu.com.

Raspberry Pi Foundation. (2016, 11 27). *Documentation*. Retrieved from www.raspberrypi.org: https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md

Raspberry Pi Foundation. (2016, 11 29). *Downloads*. Retrieved from www.raspberrypi.org: https://www.raspberrypi.org/downloads/

Raspberry Pi Foundation. (2016, 12 06). *Power Supply*. Retrieved from www.raspberrypi.org: https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

Rowand, F. (2013). Using and Understanding the Real-Time Cyclictest Benchmark. *Embedded Linux Conference.* San Francisco, CA.

Saint-Andre, P., Smith, K., & Troncon, R. (2009). *XMPP The definitive Guide - Building Real-Time Applications with Jabber Technologies.* O'REILLY.

Schnabel, P. (2007). *Elektronik-Fibel.* Ludwigsburg: Books on Demand GmbH.

Silberschatz, A., Galvin, P., & Gagne, G. (2005). *Operating System Concepts.* Hoboken, NJ: John Wiley & Sons INC.

Sony. (2011). *a35 Digitalkamera mit Wechselobjektiv.* Thailand: Sony Corporation.

Sony. (2011). *a65 Digitalkamera mit Wechselobjektiv.* Thailand: Sony Corporation.

Steeg, H.-C. (2014). *Highspeed: Kurzzeitfotografie in Natur und Studio.* Heidelberg: dpunkt.

Tanenbaum, A., & Bos, H. (2014). *Modern Operating Systems.* Prentice Hall.

Technical Committee ISO/TC 42. (2015, 1 13). ISO 2827:1988(en). *Photography - Electronic flash equipment - Determination of light output and performance*. International Organization for Standardization.

The University of North Carolina at Chapel Hill. (2006). Soft Real-time Scheduling on Multiprocessors. *Soft Real-time Scheduling on Multiprocessors*. The University of North Carolina at Chapel Hill.

Uhlig, M. (2007). *Manual der Filmkameratechnik.* Warschau: Camera Obscura Verlag.

USB Implementers Forum, Inc. (2000, 4 27). Universal Serial Bus Specification. *Universal Serial Bus Specification*. USB Implementers Forum, Inc.

Weigend, M. (2016). *Raspberry Pi programmieren mit Python.* Frechen: mitp.

Westphalen, C. (2013). *Die große Fotoschule: Digitale Fotopraxis.* Bonn: Galileo Design.

Xenomai. (2016, 11 23). *Xenomai*. Retrieved from User Guide: https://xenomai.org/installing-xenomai-3-x/#Installing_the_Cobalt_core

Yongnuo. (2014). YN560-III User Manual. Yongnuo.