Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Laborbericht Kugelfall

Laborpraktikum zum Fach Komplexe informationstechnische Systeme

Maximilian Engelhardt (II)

Rabea Sennlaub (II)

Abgegeben am xx.xx.2020

Inhaltsverzeichnis

1 Versuchsbeschreibung

In diesem Kapitel wird der physische Versuchsaufbau inklusive Sensoren und Aktuatoren beschrieben. Ferner wird die konkrete Aufgabenstellung erläutert.

1.1 Versuchsaufbau

Der Versuchsaufbau des Kugelfallversuches ist in Abbildung 1 zu sehen. Er besteht aus einer drehbaren Scheibe mit Aussparung und einem Fallrohr mit Vereinzeler, welcher durch einen Servomotor realisiert wird. Zur Steuerung wird ein Arduino Uno verwendet, welcher über ein Flachbandkabel an eine Blackbox angeschlossen ist.
Unter der Drehscheibe sind sowohl ein Hallsensor als auch ein Photosensor angebracht. Mit dem Hallsensor können zwei Magnete detektiert werden, welche an den gegenüberligenden Seiten der Drehscheibe angebracht sind (siehe Abbildung 2). Der Photosensor dient zur Detektion der Übergänge zwischen den schwarzen und weißen Segmenten auf der Unterseite der Drehscheibe. Die Drehscheibe hat einen Durchmesser von 40 cm und eine Aussparung von ungefähr 6cm x 2cm. Jedes der Farbsegmente auf der Unterseite hat eine Breite von 30°.
Am oberen Ende des Fallrohres ist ein Trichter angebracht, in welchen Kugeln vom Durchmesser 1 cm eingefüllt werden können. Diese können mit Hilfe des Vereinzelers einzeln fallen gelassen werden. Die Fallhöhe vom unteren Ende des Vereinzelers bis zur Drehscheibe beträgt etwa 72 cm.
Als weitere Elemente stehen ein Trigger und zwei LEDs zur Verfügung. Auch sind eine Kamera und ein Mikrofon angebracht, welche mit einem speziellen Programm zur Analyse verwendet werden können.

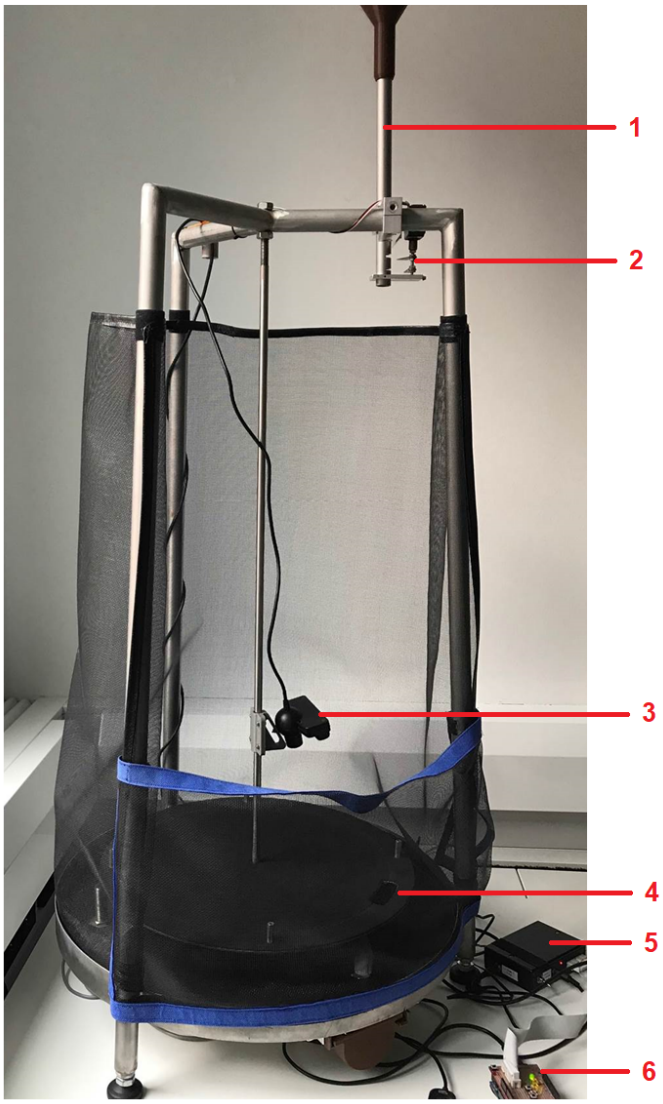

Abbildung 1: (1) Fallrohr mit Trichter (2) Vereinzeler (3) Kamera und Mikrofon (4) Drehscheibe mit Aussparung (5) Blackbox (6) Arduino Uno
Quelle: https://moodle2.tu-ilmenau.de/pluginfile.php/54328/course/section/13111/KIS.png

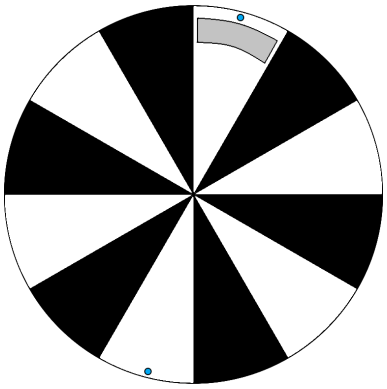

Abbildung 2: Schematische Darstellung der Unterseite der Drehscheibe. In Blau sind die vom Hallsensor detektierbaren Magnete eingezeichnet.

1.2 Aufgabenstellung

Ziel des Versuches ist es, Kugeln in einem definierten Rhythmus in die Aussparung der Drehscheibe fallen zu lassen.

Hierbei werden 3 Drehgeschwindigkeitsmodi (schnell, mittel und langsam) definiert. Die jeweiligen Geschwindigkeitsbereiche können von uns im Bereich von ca. 3 U/s bis 5 s/U festgelegt werden. Auf die konkreten Bereiche wird in Kapitel 3 eingegangen.
Für jeden Drehgeschwindigkeitsmodus sind feste Rhythmen definiert:

- **Schnell:** (Insgesamt 3 Kugeln)

  1 Kugel in das Loch Fallen lassen, ein Loch frei lassen, 1 Kugel durch das Loch fallen lassen, ein Loch frei lassen, 1 Kugel in das Loch fallen lassen
- **Mittel:** (Insgesamt 6 Kugeln)

  1 Kugel in das Loch fallen lassen, ein Loch frei lassen, zwei Kugeln nacheinander die zwei folgenden Löcher fallen lassen, zwei Löcher frei lassen, drei Kugeln in die folgenden drei Löcher fallen lassen
- **Langsam:** (Insgesamt 3 Kugeln)

  3 Kugeln nacheinander in ein Loch fallen lassen. Kein Loch auslassen

Hierzu ist es notwendig die Drehbewegung der Scheibe kontinuierlich zu bestimmen. Ferner sollen untypische Änderungen der Drehbewegung, wie z.B. Andrehen oder Bremsen, erkannt werden und ein fehlerhaftes Auslösen der Kugel verhindert werden. Sobald ein Modi sicher erkannt ist, wird die entsprechende Kugelabfolge gestartet. Sie wird bei untypischen Drehbewegungen unterbrochen und wieder fortgesetzt, sobald die Geschwindigkeit wieder sicher erkannt wird. Dieser Ablauf ist in Abbildung 3 dargestellt.
Der Geschwindigkeitsmodus soll mittels der LED optisch signalisiert werden. Zusätzlich soll signalisiert werden, wenn die Geschwindigkeit nicht sicher ermittelt werden kann. Die konkreten optischen Signale werden in Kapitel 3 näher beschrieben.

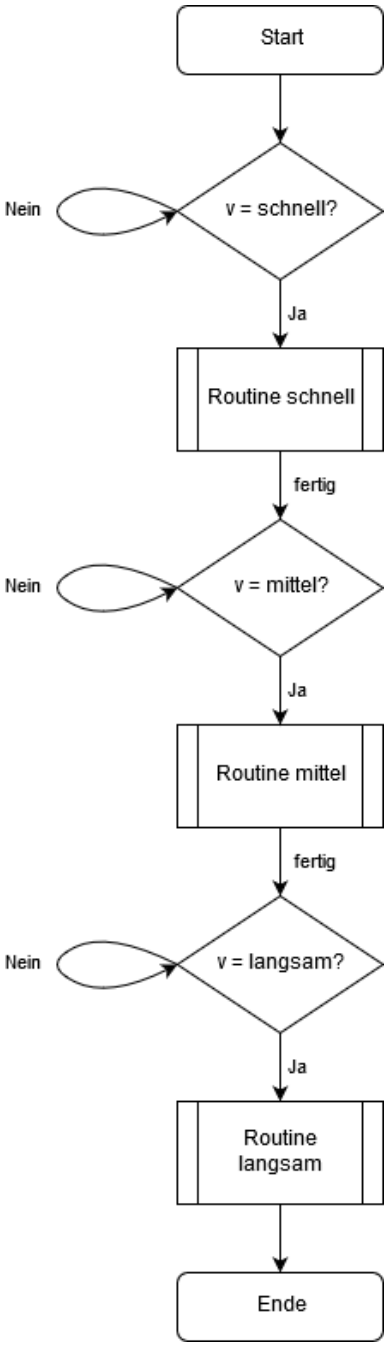

Abbildung 3: Programmablaufplan zur Darstellung der Aufgabenstellung

2 Systemanalyse

2.1 Untersuchung der Sensoren

In einem ersten Schritt haben wir ein sehr einfaches Testprogramm entwickelt, welches jede Änderung eines Sensorwertes gemeinsam mit einem Zeitstempel mit Hilfe der seriellen Schnittstelle an den Steuerrechner sendet. Anschließend wurde die Scheibe mehrmals kräftig angedreht und die Messwerte bis zum Stillstand aufgenommen. Auf diese Daten stützt sich unsere weitere Analyse.

Zunächst wird in Abbildung 4 ein kurzer Ausschnitt aus einer solchen Messung dargestellt. Als wichtigste Erkenntnis zeigt sich, dass die Sensoren nicht prellen, was die Verwendung ihrer Daten vereinfacht. Der Photosensor tastet eine Scheibe mit 12 Segmenten ab, sodass er eine deutlich höhere Auflösung als der Hallsensor bietet, welcher von nur zwei Magneten angesteuert wird. Der Hallsensor bietet aber einen anderen Vorteil: Wir konnten beobachten, dass jeder der Magnete einen spezifischen Pegelwechsel verursacht, sodass über ihn die Absolutposition ermittelt werden kann.

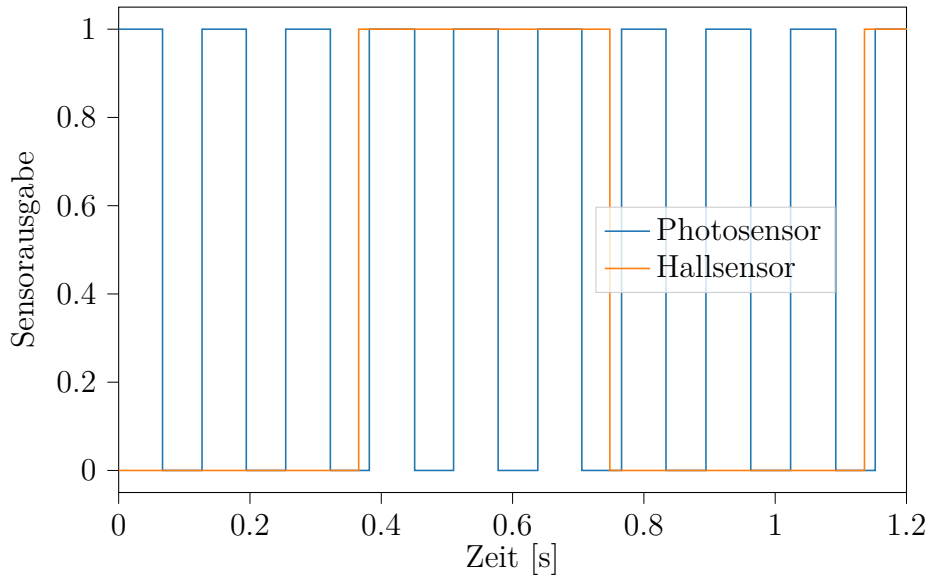

Abbildung 4: Ausgabe der Sensoren, während sich die Scheibe dreht. Es zeigt sich, dass keiner der Sensoren prellt.

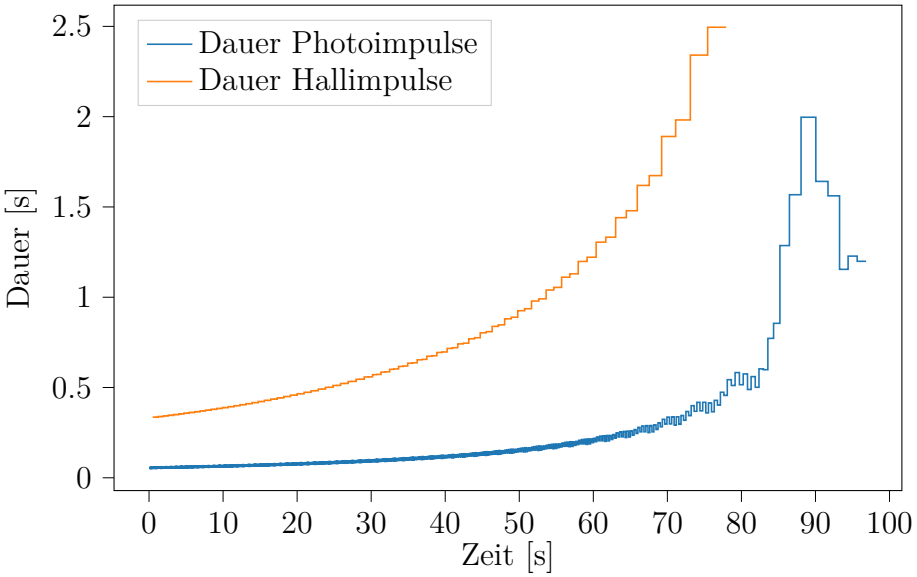

Abbildung 5: Dauer der Impulse der Sensoren im Verlauf eines Durchganges (Andrehen bis Stillstand). Kurz vor dem Stillstand (Umlaufzeiten > 5s) werden beide Sensoren unzuverlässig.

In Abbildung 5 ist die Dauer der Sensorimpulse im Verlauf eines Durchlaufes dargestellt. Wie zu erwarten ist, zeigen beide Kurven zunächst einen Trend nach oben (Verlangsamung der Scheibe). Später reißen die Messwerte des Hallsensors ab und auch der Photosensor liefert stark schwankende Werte. Die Dauer des letzten vom Hallsensor gemeldeten Impuls ist knapp 2,5s, was einer Umlaufzeit von 5s entspricht (2 Impulse pro Umdrehung). Da genau dieser Wert auch als untere Grenze des Geschwindigkeitsbereiches in der Aufgabenstellung angegeben ist, sollten damit keine Probleme für die Implementierung verbunden sein.

Des Weiteren soll untersucht werden, ob die Winkel der einzelnen Segmente, die die Sensoren erfassen, gleich groß sind. Hierfür analysieren wir den Verlauf der Durchgangszeiten der einzelnen Segmente. Natürlich ist hier keine exakte Gleichheit möglich, da sich die Drehung der Scheibe durch Reibung verlangsamt. Umgekehrt kann aber daraus geschlossen werden, dass wenn ein Segment eine kürzere Durchgangszeit als das vorherige hat, dieses einen kleineren Winkelbereich erfassen muss. Hierbei gehen wir

davon aus, dass die Scheibe im Verlauf der Drehung nicht schneller geworden sein kann.

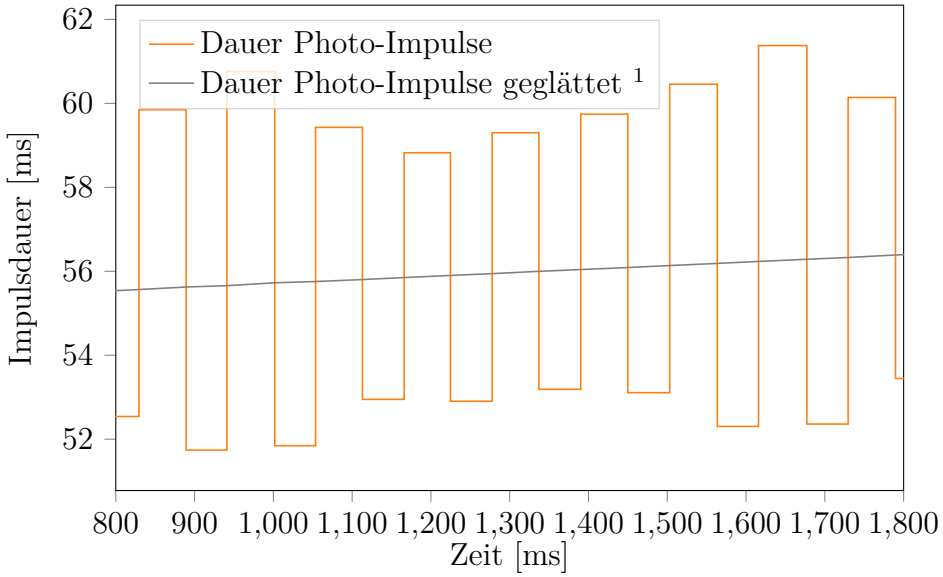

Abbildung 6: Dauer der Impulse des Photosensors (entspricht einem Ausschnitt aus Abbildung 5). Es zeigt sich, dass sich lange und kurze Impulse abwechseln. Zusätzlich ist ein Schwingungsverhalten zu beobachten.

Abbildung 6 zeigt, dass bei dem Photosensor kurze und lange Intervalle abwechselnd aufzutreten scheinen. Dies lässt vermuten, dass die Ursache nicht in der Einteilung der Scheibe liegt, sondern vielmehr der Sensor in eine Richtung die Flanken schneller erkennt als in die andere. Sollte diese Differenz konstant sein, könnte sie einfach korrigiert werden. Auch könnte dieses Problem umgangen werden, indem jeweils nur die fallenden oder nur die steigenden Flanken betrachtet werden.

In Abbildung 7 werden die gemessenen Längendifferenzen zwischen einem schwarzen und dem darauf folgenden weißen Segment über die Messdauer dargestellt. Es zeigt sich zum einen ein langfristiger Trend, was bedeutet, dass die Reaktionszeit des Sensors asymetrisch ist, dieser Effekt aber nicht konstant ist, sondern von der Drehgeschwindigkeit abhängt. Dass dieser Effekt durch die Verlangsamung der Scheibe selbst verursacht wurde, lässt sich ausschließen, indem man die Darstellung umkehrt: Berechnet man die Längendifferenzen zwischen einem schwarzen und dem *davor liegenden* weißen Segment,

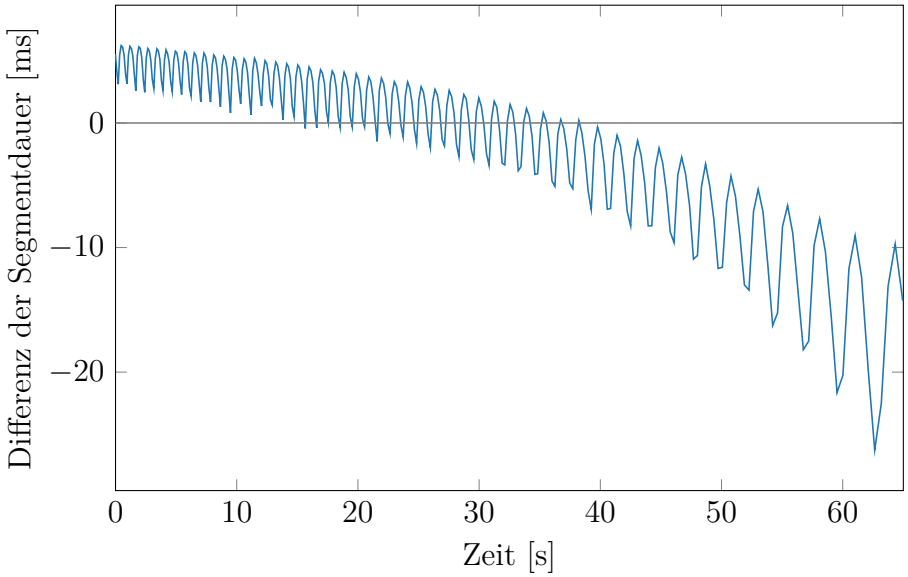

Abbildung 7: Differenz zwischen der Dauer des Durchganges eines schwarzen und dem darauf folgenden weißen Segmentes des Photosensors. Es zeigt sich, dass die Fehler nicht nur durch konstante asymmetrische Reaktionszeit des Sensors entstehen, sondern komplexere Ursachen haben.

ergibt sich das selbe Bild.

Sowohl in Abbildung 6 als auch Abbildung 7 zeigt sich ein Schwingungsverhalten, dessen Periode genau einer Umdrehung (12 Segmente) entspricht. Das deutet darauf hin, dass die Scheibe nicht genau mittig gelagert ist. Dazu passt, dass man mit dem bloßen Auge eine ungleichmäßige Drehung der Scheibe beobachtet kann. Der dadurch entstehende Fehler ist in Abbildung 8 dargestellt. Es zeigt sich, dass mit etwa 10% Abweichung gerechnet werden muss.

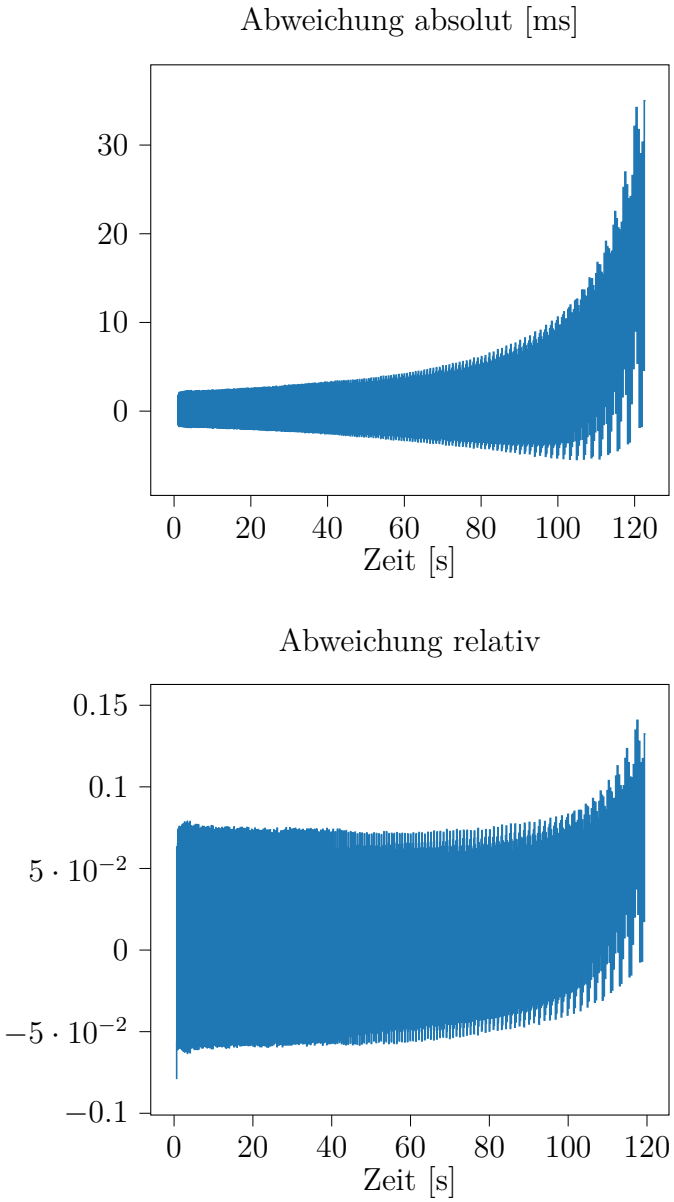

Abbildung 8: Photosensor – Abweichung der Länge der einzelnen Impulse vom über eine Umdrehung geglätteten Wert. Es zeigt sich, dass die Länge einzelner Impulse um etwa 10% abweichen kann.

2.2 Untersuchung des Vereinzelers

Ein Aktor des Systems ist eine Servo, der die Freigabe der Kugeln steuert. Dessen unmittelbare Ansteuerung ist mit Hilfe der in der Arduino-IDE eingebaute Servo-Bibliothek

einfach erledigt. Im nächsten Schritt musste geklärt werden, welche Winkelpositionen zum Freigeben beziehungsweise Zurücksetzen des Vereinzelers angefahren werden müssen. Dies wurde durch einfaches Ausprobieren gelöst.
Eine weitere interessante Charakteristik ist die Gesamtzeit vom Auslösen bis zum Auftreffen einer Kugel. Diese Verzögerung setzt sich natürlich aus verschiedenen Elementen (elektronische Verzögerung, mechanische Verzögerung, Fallzeit der Kugel) zusammen, deren Bestimmung im Einzelnen aber nicht nötig ist. Zur Ermittelung der Gesamtzeit haben wir ein Programm entwickelt, welches nach Betätigung einer Taste sofort eine Kugel auslöst. Von diesem Prozess haben wir eine Tonaufnahme angefertigt, in der wir die genauen Zeitpunkte des Tastendrucks, der Servobewegung und des Aufschlages der Kugel identifiziert haben (siehe Abbildung 9). Auf diesem Weg ist es möglich, ohne menschlichen Einfluss (Reaktionszeit) zu messen. So wurde eine Fallzeit von näherungsweise 487 ms ermittelt.

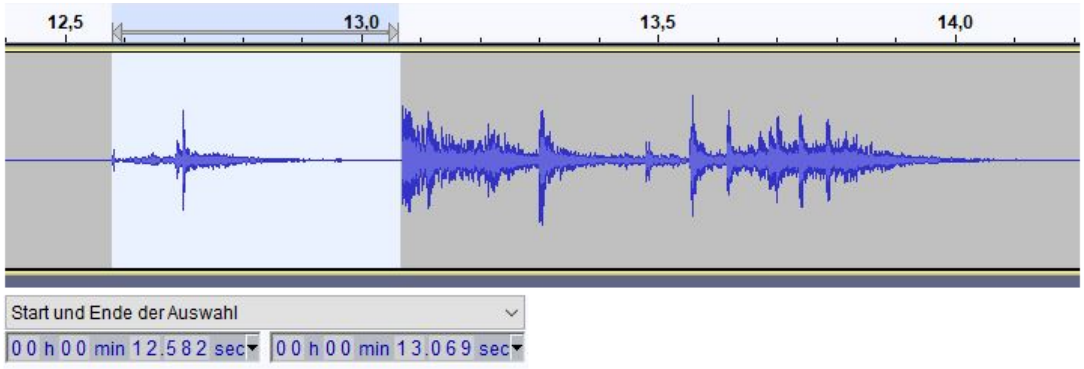

Abbildung 9: Audioaufzeichnung des Auslöseprozesses einer Kugel. Der ausgewählte hellere Bereich beschreibt den Zeitraum von der ersten Servoaktivität bis zum Aufprall der Kugel.

2.3 Analyse des Verhaltens der Drehscheibe

Die Daten des oben beschriebenen Photosensors sollen nun dazu verwendet werden, die Drehung der Scheibe selbst zu analysieren. Abbildung 10 zeigt wie zu erwarten, dass

sich die Scheibe nach dem Andrehen verlangsamt, bis sie zum Stillstand kommt.

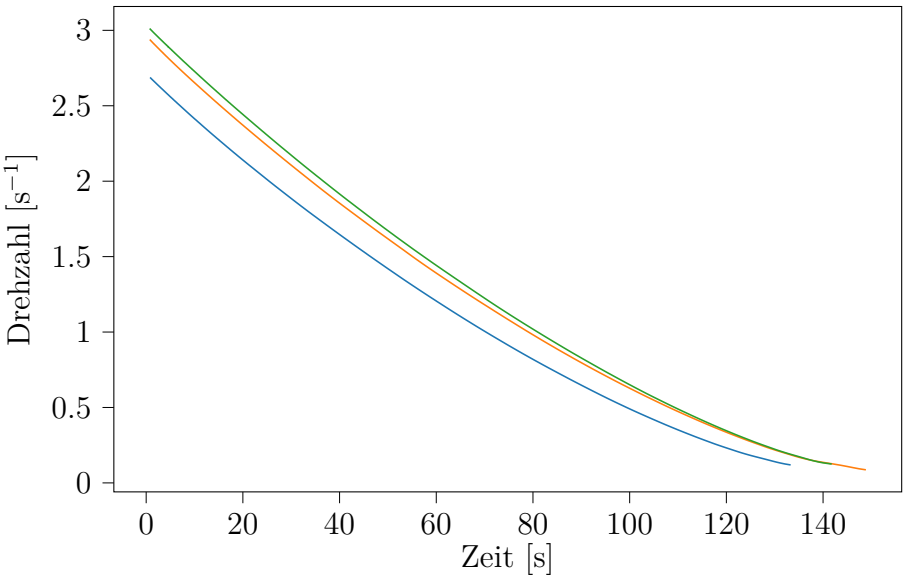

Abbildung 10: Verlauf der Drehgeschwindigkeit ohne äußere Einflüsse für drei Durchgänge. Geglättet mittels gleitendem Mittelwert über 12 Segmente.

Interessant ist nun, wie diese Abbremsung erfolgt. Im häufig verwendeten Modell der Coulombschen Gleitreibung ist die Reibungskraft und damit die Beschleunigung unabhängig von der Geschwindigkeit. Im Falle der Rotation bedeutet dies, dass die Winkelbeschleunigung unabhängig von der Drehzahl ist. Jedoch zeigt sich in Abbildung 11, dass dies hier nicht der Fall ist.

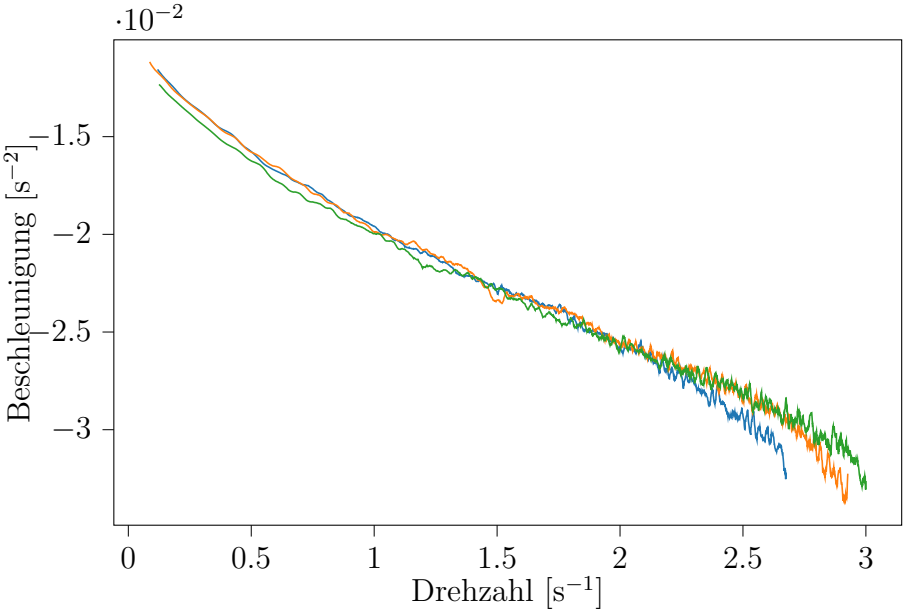

Abbildung 11: Darstellung der Beschleunigung über die Drehzahl. Es zeigt sich, dass die Beschleunigung nicht unabhängig von der Geschwindigkeit ist.

2.4 Nutzung der Sensoren und Aktoren für die spätere Aufgabe

Für die Geschwindigkeits- und Positionsbestimmung der Drehscheibe können der Hallsensor und der Photosensor verwendet werden. Hierbei kann mit dem Hallsensor die absolute Position der Drehscheibe bestimmt werden. Durch die höhere Auflösung des Photosensors (12 statt 2 Flanken pro Umdrehung der Scheibe) ist dieser geeignet, um eine genauere Positionsbestimmung relativ zu der durch den Hallsensor bestimmten Position durchzuführen. Diese wird benötigt, um den Abwurfzeitpunkt der Kugel zu ermitteln.

Der Servomotor wird als Vereinzeler verwendet, um die Kugeln einzeln zum richtigen Zeitpunkt fallen zu lassen.

Mit Hilfe der zwei LEDs sollen die in der Aufgabenstellung geforderten Geschwindigkeitsmodi angezeigt werden. Ferner soll mit ihnen signalisiert werden, wenn eine Störung (z.B. plötzliches Abbremsen der Scheibe) erkannt wurde.

3 Design

Im folgenden Kapitel wird die Machbarkeit der Aufgabenstellung untersucht. Ferner wird unsere Lösungidee und die Implementierung derselbigen beschrieben.

3.1 Machbarkeitsanalyse

Für die Implementierung der Aufgabenstellung sollten sinnvolle Geschwindigkeitsbereiche für die verschiedenen Geschwindigkeitsmodi festgelegt werden. Diese wurden von uns wie folgt verteilt:

- **schnell:** 1,8 U/s - 3,0 U/s
- **mittel:** 1 U/s - 1,8 U/s
- **langsam:** 0,25 U/S - 1 U/s

Im Folgenden soll mit Fokus auf die Rotationsgeschwindigkeit überprüft werden, ob der Versuch unter den genannten Rahmenbedingungen erfolgreich durchführbar ist. Hierbei sind vor allem die untere Systemgrenze und die obere Systemgrenze interessant.

Untere Systemgrenze

Die untere Systemgrenze ist durch die Genauigkeit der Sensoren gegeben. In Abbildung 5 ist gut zu sehen, dass der Hallsensor ab einer Geschwindigkeit von etwa 0,2 U/s keine Werte mehr liefert. Auch der Photosensor liefert ab dieser Geschwindigkeit keine zuverlässigen Werte mehr.
Da der von uns definierte niedrigste Wert für die Geschwindigkeit 0,25 U/s beträgt, sollten sich dadurch keine Probleme ergeben.

Obere Systemgrenze

Die obere Systemgrenze definiert sich durch die Zeit, welche die Kugel braucht, um die Aussparung in der Scheibe zu passieren. Ist diese Zeit größer als die Zeit in der sich die Aussparung unter der Fallgeraden der Kugel befindet, so ist es physikalisch nicht

möglich, die Kugel hindurchfallen zu lassen.
Die Fallgeschwindigkeit v_{Kugel} der Kugel bestimmt sich durch

$$v_{Kugel}(h) = \sqrt{2 * g * h} = 3{,}758\frac{m}{s}$$

wobei $h = 0{,}72m$ die Fallhöhe der Kugel und $g = 9{,}81\frac{m}{s^2}$ die Beschleunigung ist.
Die Dicke der Drehscheibe beträgt $d_{Scheibe} = 0{,}5cm = 0{,}005m$. Um durch die Aussparung hinzurchzufallen benötigt die Kugel mit der Geschwindigkeit v_{Kugel} die Zeit t, welche sich durch

$$t = \frac{d_{Scheibe}}{v_{Kugel}} = 0{,}00133s = 1{,}33ms$$

bestimmt. Die maximal erlaubte Geschwindigkeit $v_{Scheibe}$ der Drehscheibe ist also jene, in welcher sich die Scheibe in $1{,}33ms$ nur maximal $l_{Aussparung} - d_{Kugel} = 6cm - 1cm = 0{,}05m$ weit bewegt. Das entspricht ungefähr $\frac{1}{25}$ U. Hierbei beschreibt $l_{Aussparung}$ die Länge der Aussparung und d_{Kugel} den Durchmesser der Kugel.
$v_{Scheibe}$ kann durch

$$v_{Scheibe} = \frac{l_{Aussparung} - d_{Kugel}}{t} = 30\frac{U}{s}$$

bestimmt werden.
Da durch manuelles Andrehen keine Werte erzielt werden konnten, welche 3,5 U/s überschreiten, ist diese theoretische obere Systemgrenze nicht relevant.

3.2 Der Schätzalgorithmus

Kernkomponente unsere Programms ist der Schätzalgorithmus. Er ermittelt auf Basis der gesammelten Daten das Segment, über welchem die Kugel abgeworfen werden soll. Ferner ermittelt er die Abwurfverzögerung, also die Dauer zwischen Beginn des ermittelten Segmentes und der eigentlichen Auslösung.
Seine grundlegende Funktionsweise soll Abbildung 12 illustrieren: Zunächst wird die Umdrehungsdauer ermittelt, indem die Segmentzeiten der letzten Umdrehung auf-

summiert werden. Dann wird von der Soll-Landestelle (Mitte der Aussparung in der Drehscheibe) ausgehend rückwärts betrachtet, wie weit vorher abgeworfen werden muss. Grundlage hierfür ist eine konstante Verzögerungszeit sowie die Annahme, dass sich die Scheibe noch immer mit der selben Geschwindigkeit wie in der Vorrunde dreht. Als letzter Schritt wird die Abwurfverzögerung ermittelt. Das ist die Zeit, die nach Beginn des Abwurfsegmentes noch gewartet werden soll. Sie wird einfach als Differenz der Verzögerung, die bei Abwurf gleich zu Segmentbeginn entstehen würde und der konstanten Sollverzögerung berechnet. Die Sollverzögerung wurde in Unterkapitel 2.2 bestimmt und experimentell angepasst.

Vorteil dieser Methode ist zum einen ihre Einfachheit, zum Anderen, dass mit der empirisch ermittelten Verzögerungszeit praktisch alle zeitlich konstanten Fehler des Systems kompensiert werden können: So enthält sie neben der reinen Fallzeit auch die elektronische und mechanische Verzögerung des Servos.

Als Nachteil bleibt, dass sowohl die Ungenauigkeit des Photosensors als auch die Verlangsamung der Scheibe durch Reibung vernachlässigt werden. Der praktische Versuch muss zeigen, ob diese Vereinfachungen zulässig sind.

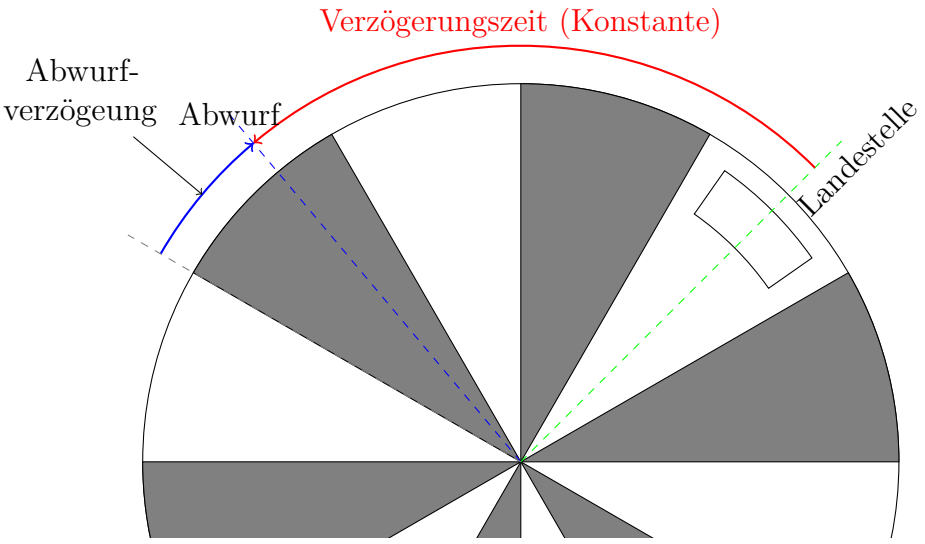

Abbildung 12: Funktionsweise des Schätzalgorithmus. Er ermittelt das Segment, in dem abgeworfen werden soll sowie die Abwurfverzögerung innerhalb des Segmentes.

3.3 Programmatische Umsetzung

Da es sich hier um ein System mit harter Echtzeitanforderung handelt, muss vor der Implementierung geplant werden, wann welche Berechnungen ausgeführt werden sollen. Hier bietet der Arduino fast die volle Kontrolle: Es gibt kein Betriebssystem, dessen Zeitverhalten beachtet werden müsste und Hardwareinterrupts stehen zur Verfügung. Der Ablauf, für den wir uns entschieden haben, ist in Abbildung 13 dargestellt: Die Überwachung des Photosensors erfolgt mittels eines Interrupts. Immer, wenn sich der erkannte Zustand ändert, wird die zugehörige ISR (Interrupt Service Routine) aufgerufen, welche die aktuelle Zeit ausliest, die Dauer des vorherigen Segmentes speichert und den Segmentzähler weiterschaltet. Komplexe Berechnungen oder auch Debugausgaben erfolgen hier nicht, um die Ausführungszeit kurz zu halten.

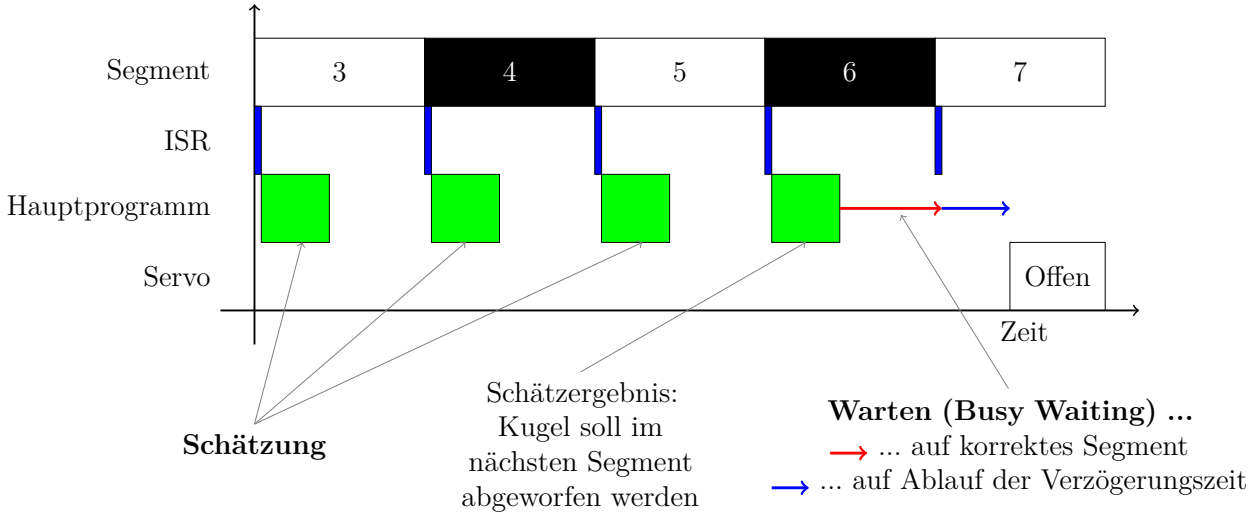

Abbildung 13: Zeitlicher Ablauf des Programms

Währenddessen wartet das Hauptprogramm im Normalzustand darauf, dass der Segmentzähler weiter geschaltet wird, um dann die oben beschriebene Schätzroutine aufzurufen. Kommt diese zu dem Ergebnis, dass der optimale Abwurfzeitpunkt erst im übernächsten Segment oder noch später liegt, wird die Berechnung verworfen und im nächsten Segment erneut durchgeführt. Dies hat den Vorteil, dass dann aktuellere Daten zur Verfügung stehen, die die Schätzung eventuell verbessern.

Stellt die Schätzung dagegen fest, dass der Abwurf im aktuellen oder nächsten Segment erfolgen soll, wird zunächst gewartet, bis das Segment beginnt (entfällt, wenn aktuelles Segment), und dann weiter gewartete, bis seit Segmentbeginn (hier wird der von der ISR aufgezeichnete Wert verwendet) die berechnete Auslöseverzögerung vergangen ist. Dieses Warten wird als *busy wait* implementiert, da es hier keine anderen Berechnungen zu tun gibt. Der Regelfall ist, wie in der Grafik dargestellt, dass im nächsten Segment ausgelöst wird. Natürlich kann sich aber, wenn sich die Schätzung verändert hat, der Auslösezeitpunkt nach vorne verschieben, sodass auch ein Auslösen im selben Segment möglich ist.
Diese Aufteilung stellt sicher, dass der Aufruf der Schätzmethode relativ zeitunkritisch ist. Das ermöglicht es, hier mit Gleitkommazahlen zu arbeiten und viele Debugausgaben einzubauen. Nebenaufgaben, insbesondere das Blinken der LEDs, werden vom Hauptprogramm zu unkritischen Zeitpunkten per Polling erledigt. Dies vereinfacht die Implementierung und stellt sicher, dass sie nicht mit zeitkritischen Berechnungen in Konflikt kommen.

3.4 Objektorientierte Implementierung

Wir haben unser Programm in folgende fünf Komponenten geteilt (siehe Abbildung 14):

- Der *DiskMonitor* verfolgt die Drehung der Scheibe und enthält den Schätzalgorithmus
- Die *Routine* überwacht den Ablauf der Abwurfsequenzen in den verschiedenen Geschwindigkeiten. Sie entscheidet jeweils, ob eine sich ergebende Abwurfgelegenheit genutzt oder ausgelassen wird.
- Der *LEDController* steuer die optische Ausgabe. Er ermöglicht es auch, die LEDs blinken zu lassen.
- Der *ServoController* sendet die Befehle an den Servo
- Das Hauptprogramm steuert das Zusammenspiel der Komponenten

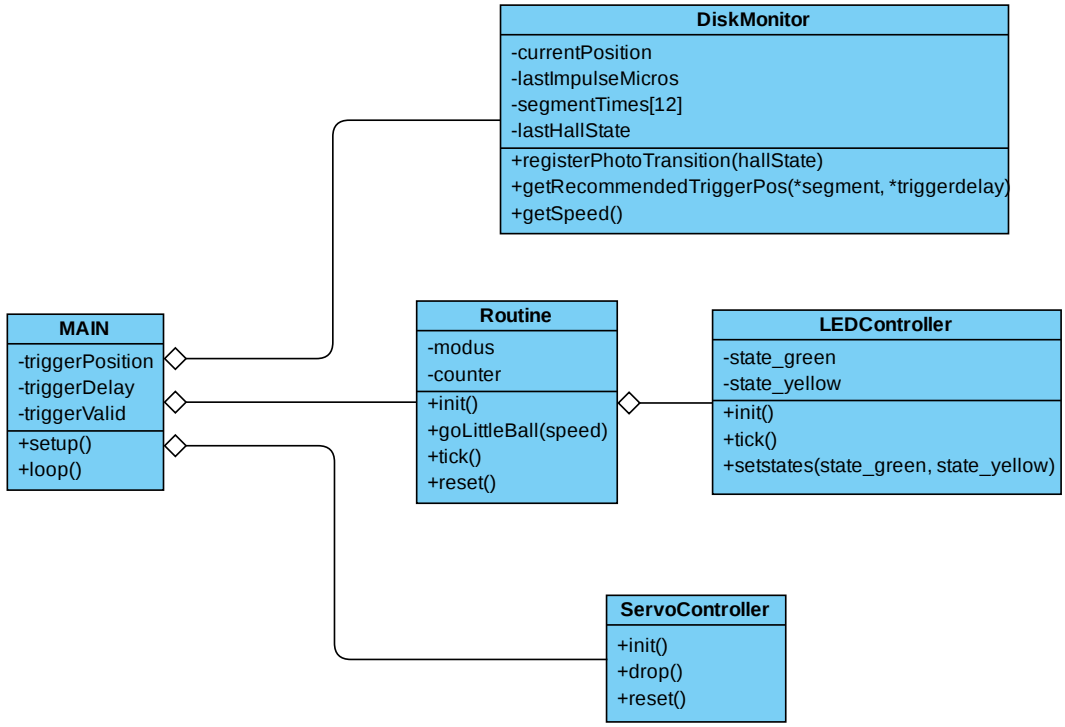

Abbildung 14: UML Klassendiagramm unserer Lösung

Durch diese Aufteilung soll eine geringe Kopplung zwischen und eine hohe Kohäsion innerhalb der einzelnen Klassen gewährleistet werden.

3.5 Fehlererkennung

Eine weitere Anforderung war die Erkennung von unüblicher Beschleunigung oder Abbremsung der Drehscheibe und das Verhindern des fehlerhaften Kugelabwurfes. Dies wird in der Funktion *DiscMonitor::getSpeed()* realisiert. In dieser wird die aktuelle Geschwindigkeit zurückgegeben, gemittelt über 12 Segmente. Bei vermutetem Fehler (Abbremsen etc.) wird 0 zurückgegeben. Hierzu wird die Abweichung der Segmentzeit der letzten beiden gleichfarbigen Segmente berechnet. Wenn diese einen festgelegten Threshold überschreitet, kann von einer unüblichen Bewegung der Drehscheibe ausgegangen werden. Die *Routine* fragt die aktuelle Geschwindigkeit der Drehscheibe ab. Wenn also 0 zurückgegeben wird, so wird die aktuelle Abwurfgelegenheit nicht genutzt.

3.6 Designentwurf LED

Laut Aufgabenstellung soll der aktuelle Geschwindigkeitsmodus mittels der LED optisch signalisiert werden. Zusätzlich soll signalisiert werden, wenn die Geschwindigkeit nicht sicher ermittelt werden kann.
In unserer Lösung haben wir uns für die in Tabelle 1 dargestellte Aufteilung für die Signalisierung der drei Modi entschieden.

Modus	**LED grün**	**LED gelb**
schnell	x	x
mittel	x	
langsam		x

Tabelle 1: Entwurf der optischen Darstellung der Geschwindigkeitsmodi

Eine jeweilige Abweichung der Geschwindigkeit bzw. eine nicht ermittelbare Geschwindigkeit wird über ein Blinken der jeweiligen LEDs dargestellt.
Zur Verdeutlichung drei Beispiele:

- Aktueller Modus: *schnell*, gemessene Geschwindigkeit: schnell → beide LEDs leuchten
- Aktueller Modus: *mittel*, gemessene Geschwindigkeit: schnell → grüne LED blinkt
- Aktueller Modus: *schnell*, gemessene Geschwindigkeit: nicht ermittelbar → beide LEDs blinken

4 Fehler- und Ergebnisanalyse

4.1 Simulation „Software in the Loop“

Um unsere Implementierung des Schätzalgorithmus verifizieren zu können, haben wir folgendes Testverfahren entwickelt: Die Klasse DiskMonitor wird als dynamische Bibliothek für die Entwicklungsplatform gebaut (also nicht wie üblich cross-compiliert) und anschließend von einem Pythonprogramm geladen, was die im Schritt Systemanalyse aufgenommenen Sensordaten einliest und in den Algorithmus hinein gibt. Dann wird die auch im Kapitel Systemanalyse abgeschätzte Fallzeit herangezogen, um zu simulieren, wo die Kugel aufkommen würde.

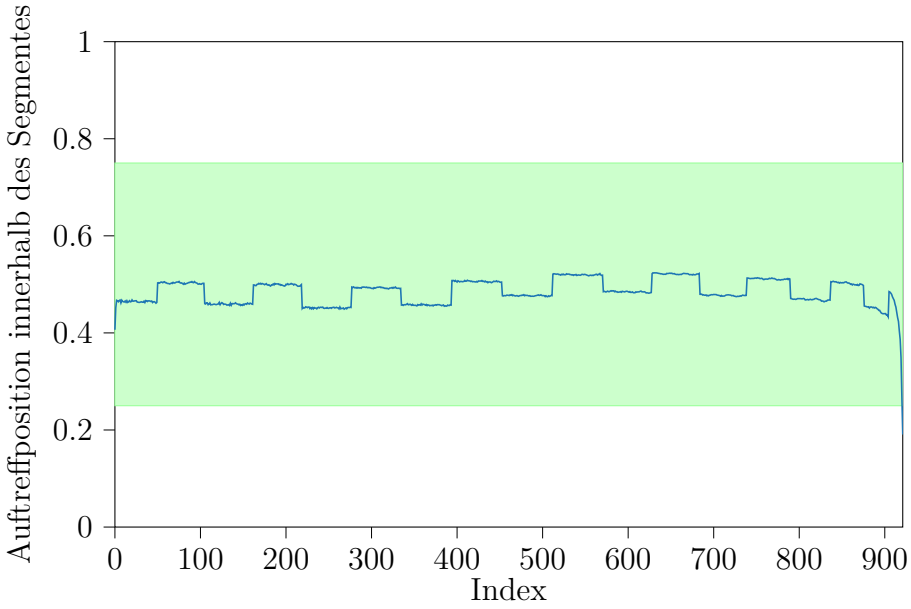

Abbildung 15: Ergebnis der Simulation: Bis auf 2 Ausreißer am Rand sollte zuverlässig in die Mitte des Loches getroffen werden.

Das Ergebnis ist in Abbildung 15 dargestellt: Bis auf 2 Ausreißer an den Rändern treffen alle Kugeln relativ genau in die Mitte des Loches, was andeutet, dass die Schätzung korrekt implementiert ist. Die Auswirkungen unserer Vereinfachungen (alle Segmente gleich groß, Vernachlässigung der Reibung) sind zwar zu erkennen, aber ausreichend klein, um sie zu vernachlässigen.

Natürlich muss man auch die Grenzen dieser „Software in the Loop“-Simulation bedenken: Dem Verfahren inhärent ist, dass das Hardwareverhalten nur so gut simuliert wird, wie man es modelliert hat: Auf Seite der Sensoren sollte dies keine Einschränkung sein, da wir hier real aufgenommene Messdaten verwendet haben. Demgegenüber wurde das Abwurfverhalten nur durch eine konstante Verzögerung abgebildet, was möglicherweise eine starke Vereinfachung ist. Hinzu kommt, dass sich die Entwicklungsplatform vom eingebetteten System unterscheidet, z.B. breitere Datentypen verwendet. Das könnte Fehler wie etwa numerische Überläufe maskieren.

4.2 Probleme und Problemlösungen

Ein Problem, welches sich beim Testen ergeben hat war, dass die Kugel beim Fallenlassen leicht angedreht wurde, sodass ihre Fallbahn abgelenkt wurde und seitlich neben der Aussparung in der Drehscheibe aufkam. Dieses Problem haben wir durch Anpassung der Servoposition behoben. Hierbei haben wir die Winkelposition bestimmt, bei welcher die Kugel gerade so nicht fallen gelassen wird. Sobald die Kugel dann fallengelassen werden soll muss der Servo nur noch eine sehr kleine Bewegung ausführen und das Andrehen der Kugel wird vermieden.

Ein weiteres Problem ist kurz vor Fertigstellung der Software aufgetreten. Wir konnten beobachten, dass eine vorher reibungslos funktionierende Version des Programms auf einmal Kugeln fehlerhaft abgeworfen hat. Durch Beobachtungen konnten wir feststellen, dass jedes Mal wenn sich der Servomotor bewegte, die berechnete Position von der gemessenen abwich. Aufzeichnungen der Sensorsignale ergaben, dass zu den fraglichen Zeitpunkten irreguläre Peaks im Verlauf des Photosensors zu verzeichnen waren (vgl. Abbildung 16).

Um die korrekte Funktionsweise der Sensorik zu überprüfen haben wir ein einfaches Testprogramm geschrieben. In diesem wird der Photosensor durch Interrupts ausgelesen. Bei jedem dieser Interrupt wird der aktuelle Wert des Photosensors ausgegeben.

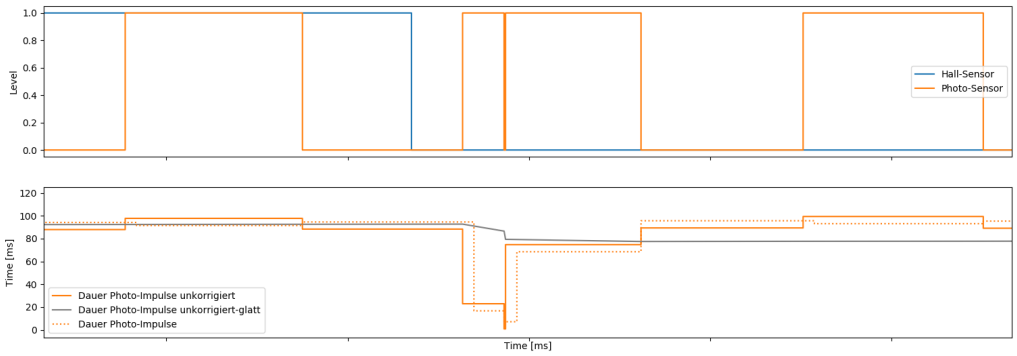

Abbildung 16: Diagramm der fehlerhaften Flanken des Photosensors

Mit Drücken des Triggers kann das Auslösen einer Kugel (also eine Bewegung des Servos) bewirkt werden. Nun haben wir die Drehscheibe so positioniert, dass sich der Photosensor mittig eines weißen Segments befindet und haben die Scheibe an dieser Position fixiert. Beim wiederholten Auslösen des Triggers konnten keine Flanken bzw. Interrupts des Photosensors verzeichnet werden. Danach haben wir die Drehscheibe so positioniert, dass sich der Photosensor mittig eines schwarzen Segments befindet und haben die Scheibe erneut an dieser Position fixiert. Beim Auslösen des Triggers konnten wir jeweils zwei Interrupts beobachten (vgl. Abbildung 17). Mit diesem Versuch konnten wir nachweisen, dass der Fehler bei der Sensorik liegt und seinen Ursprung nicht im Programmcode hat.

Eine mögliche Fehlerbehandlung wäre, die fehlerhaften Peaks (und somit fehlerhaften Segmente) zu erkennen und auszusortieren bzw. zu ignorieren. Da die fehlerhaften Interrupts sehr nah hintereinander stattfinden könnte dies über die Segmentzeit realisiert werden. Da unser Algorithmus zur Fallzeitschätzung jedoch darauf angewiesen ist, am Anfang eines Segmentes zu wissen, dass das Segment gültig ist, konnten wir diese Variante der Fehlerbehandlung nicht ohne weiteres umsetzen. Auf Grund der weit fortgeschrittenen Phase des Projektes und der Einhaltung von Deadlines war die Entwicklung eines neuen Algorithmus zur Fallzeitschätzung nicht möglich.

Die zweite Möglichkeit wäre die Behebung des Problems durch Reparieren oder Austau-

schen der Sensorik. Da diese Möglichkeit ein besseres Kosten-Nutzen-Verhältnis hatte haben wir uns für diese entschieden und durch einen Wechsel des Versuchsaufbaus realisiert.

4.3 Probleme der Wiederholbarkeit

Es hat sich herausgestellt, dass der Parameter der Fallzeit *dropdelay* bei jedem Labortermin etwas angepasst werden musste. Diese Anpassungen beliefen sich auf etwa 0,5-1 ms. Dieser Umstand kann verschiedene Gründe haben. Am Wahrscheinlichsten sind kleinste Änderungen des Versuchsaufbaus, wie zum Beispiel das Verschieben desselbigen beim Andrehen oder minimale Gradänderungen oder Ungenauigkeiten des Servos.

4.4 Fazit

Die in der Aufgabenstellung vorgegebene Abwurfroutine wird zuverlässig ausgeführt. Die aktuellen Modi werden mit Hilfe der LEDs signalisiert. Ferner können unübliche Bewegungen wie Abbremsen oder Beschleunigen erkannt werden und ein fehlerhaftes Abwerfen der Kugel wird verhindert.

```
Hier steht die Drehscheibe so, dass der Phosensor mittig
eines weißen Segmentes ist. Die Scheibe wird in dieser
Position festgehalten

15:18:00.079 -> Photosensor ist jetzt 0
15:18:02.792 -> Servo auf
15:18:03.788 -> Servo zu
15:18:04.306 -> Servo auf
15:18:05.305 -> Servo zu
15:18:05.650 -> Servo auf
15:18:06.679 -> Servo zu

Nun haben wir die Scheibe so gedreht, dass der Photosensor
mittig von einem schwarzen Segment ist. Auch hier bewegt
sich die Scheibe nicht.

15:18:10.654 -> Photosensor ist jetzt 1
15:18:14.221 -> Servo auf
15:18:15.248 -> Servo zu
15:18:16.415 -> Servo auf
15:18:16.553 -> Photosensor ist jetzt 0
15:18:16.553 -> Photosensor ist jetzt 1
15:18:17.407 -> Servo zu
15:18:18.711 -> Servo auf
15:18:18.848 -> Photosensor ist jetzt 0
15:18:18.848 -> Photosensor ist jetzt 1
15:18:19.706 -> Servo zu
15:18:19.842 -> Photosensor ist jetzt 0
15:18:19.874 -> Photosensor ist jetzt 1
15:18:21.178 -> Servo auf
15:18:21.316 -> Photosensor ist jetzt 1
15:18:21.316 -> Photosensor ist jetzt 1
15:18:22.173 -> Servo zu
15:18:23.101 -> Servo auf
15:18:23.238 -> Photosensor ist jetzt 0
15:18:23.272 -> Photosensor ist jetzt 1
15:18:24.097 -> Servo zu
15:18:24.270 -> Photosensor ist jetzt 0
15:18:24.270 -> Photosensor ist jetzt 1
```

Abbildung 17: Kommentierte Ausgabe des Testprogramms zur Fehleranalyse des Photosensors

Abbildungsverzeichnis