***Tu-Ilmenau  
FG System- und Software-Engineering  
Sommersemester 2019***

**Praktikum:**  
Im Fach

**Komplexe Informationstechnische Systeme**

(Dr. -Ing. Ralph Maschotta)

Thema des Praktikums:

„**Kugelfall-Versuch**“

Ahmad Asmandar  
Sabina Zhernovay

[Ziel des Versuches: 4](#_Toc13858826)

[Systemanalyse: 4](#_Toc13858827)

[Sensoren: 4](#_Toc13858828)

[Was ist aber ein Sensor? 4](#_Toc13858829)

[Und wie funktioniert!? 4](#_Toc13858830)

[Hall Sensor: 4](#_Toc13858831)

[Optischer Sensor oder (Optischer Näherungsschalter): 5](#_Toc13858832)

[Aktoren: 6](#_Toc13858833)

[Servo Motor: 6](#_Toc13858834)

[Drehscheibe: 7](#_Toc13858835)

[Die Verschaltung: 7](#_Toc13858836)

[Die Verbindung mit Arduino: 7](#_Toc13858837)

[Sensoren mit Arduino: 8](#_Toc13858838)

[Die Knöpfe (The Buttons): 8](#_Toc13858839)

[Sensor Valuesänderungen: 9](#_Toc13858840)

[Die Genauigkeit der Sensoren: 11](#_Toc13858841)

[Design: 12](#_Toc13858842)

[Die Theorie der Lösung: 12](#_Toc13858843)

[Parameter von Kreisbewegungen: 13](#_Toc13858844)

[Formeln der Kreisbewegung: 13](#_Toc13858845)

[UML Diagramm: 14](#_Toc13858846)

[Main Hauptfunktionen: 16](#_Toc13858847)

[Speed Hauptfunktionen: 16](#_Toc13858848)

[Shoot Hauptfunktionen: 17](#_Toc13858849)

[Simulation 18](#_Toc13858850)

[Das Debug-System: 19](#_Toc13858851)

[Implementierung 21](#_Toc13858852)

[Validierung : 21](#_Toc13858853)

[Bemerkung: 21](#_Toc13858854)

[Fehler Amylase: 22](#_Toc13858855)

[Die Lösung und Optimierung: 24](#_Toc13858856)

[Fazit: 25](#_Toc13858857)

[Empfehlung zur Verbesserung des Versuchs: 25](#_Toc13858858)

[How To: 26](#_Toc13858859)

[Warum VSCODE: 26](#_Toc13858860)

# Ziel des Versuches:

Das Ziel des Versuches ist es mit Hilfe eines Arduino und verschiedenen Sensoren, eine Kugel in einem Loch einer rotierenden Scheibe fallen zu lassen.

# Systemanalyse:

## Sensoren:



Abb 1 Hall Sensor Photo-Sensor

Damit wir unser System richtig analysieren, sollen wir am Anfang die wichtigen Begriffe erklären.

### Was ist aber ein Sensor?

Das Prinzip besteht darin, eine physikalische Größe oder aber einen chemischen Effekt aufzunehmen und in ein analog -elektrisches Signal umzuwandeln. Wesentliche physikalische Größen sind Drehzahl, Gewicht, Lichtstärke, Druck, Temperatur etc.

### Und wie funktioniert!?

Das Prinzip eines Sensors ist leicht zu verstehen, Sensoren sind aber durchaus komplexe elektronische Geräte, denen eine große Bedeutung im Alltag zukommt. Zunächst erfasst der Sensor die physikalische Größe und wandelt diese in eine spezifische elektrische Spannung um, wobei diese in proportionalem Verhältnis zur Eingangsgröße steht.

Der ganze versuche basiert auf zwei wichtigen Sensoren, der Hall-Sensor und der Foto-Sensor.

### Hall Sensor:

Hall Sensoren bestehen aus möglichst dünnen kristallinen dotierten Halbleiterschichten, die seitlich zumeist vier Elektroden besitzen.

Durch die Zwei gegenüberliegenden Elektroden wird ein Strom eingespeist. Die beiden orthogonal dazu liegenden Elektroden dienen der Abnahme der Hall-Spannung. Wird ein solcher Hall-Sensor von einem senkrecht zur Schicht verlaufenden Magnetfeld durchströmt, liefert er eine Ausgangsspannung.

|  |
| --- |
|  |

Bei Kugelfall-Versuch sind unten an der Drehscheibe Zwei Magnete befestigt (Abb2). Die den Hall-Sensor auslösen, der unterhalb der Scheibe angebracht ist. Die beiden Magneten sind gegenüber gelegt und versetzen dabei 180° zueinander. Wenn ein Magnet über den Sensor passt, wird der Sensor seinen Zustand vom 0 zu 1 oder umgekehrt wechseln und dabei diesen Wert an den Arduino ermitteln.

### Optischer Sensor oder (Optischer Näherungsschalter):

Abb 2 Hall-Sensor Position von unten

Ein optischer Sensor besteht aus einem Lichtsender (häufig eine Leuchtdiode oder eine Laserdiode) und einem Lichtempfänger (beispielweise einen lichtempfindlichen Widerstand (LDR) oder eine Fotodiode). Die Empfänger (Auswerteinheit) wertet die Intensität, die Farbe oder Laufzeit des vom Lichtsender empfangenen Lichtes aus und liefert er eine Ausgangsspannung.

Beim optischen Sensor wird der vom Sender ausgesandete Lichtstrahl entweder direkt um Empfänger geschickt (Einweglichtsensor) oder von einem Reflektor reflektiert und dann vom Empfänger aufgenommen. In beiden Fällen wird das Signal ausgelöst, wenn der Licht Strahl unterbrochen wird.

|  |
| --- |
|  |

Der optische Sensor beim Versuch ist unterhalb einer Drehscheibe angebracht. Die in 12 Segmente geteilt wurde (Abb 3). Diese Teile sind gleich groß und abwechselnd schwarz und Weiß gefärbt. Der Wechsel von einem Teil zum Nächsten wird vom Sensor detektiert.

Falls weißer Teil wird 1 an den Arduino ermittelt oder 0 falls schwarz ist und damit ist 30° bestanden.

Abb 3 Photo-Teile unten Ansicht

## Aktoren:

Bei Aktoren oder Aktuatoren handelt es sich um Antriebselemente, die elektrische Signale und Strom in mechanische Bewegung transformieren. Ihr Wirkprinzip ist dabei umgekehrt proportional zu Sensoren. Elektrische Impulse werden durch einen Aktor in Druck, Schall, Temperatur, Bewegung oder andere mechanische Größen umgewandelt.

### Servo Motor:

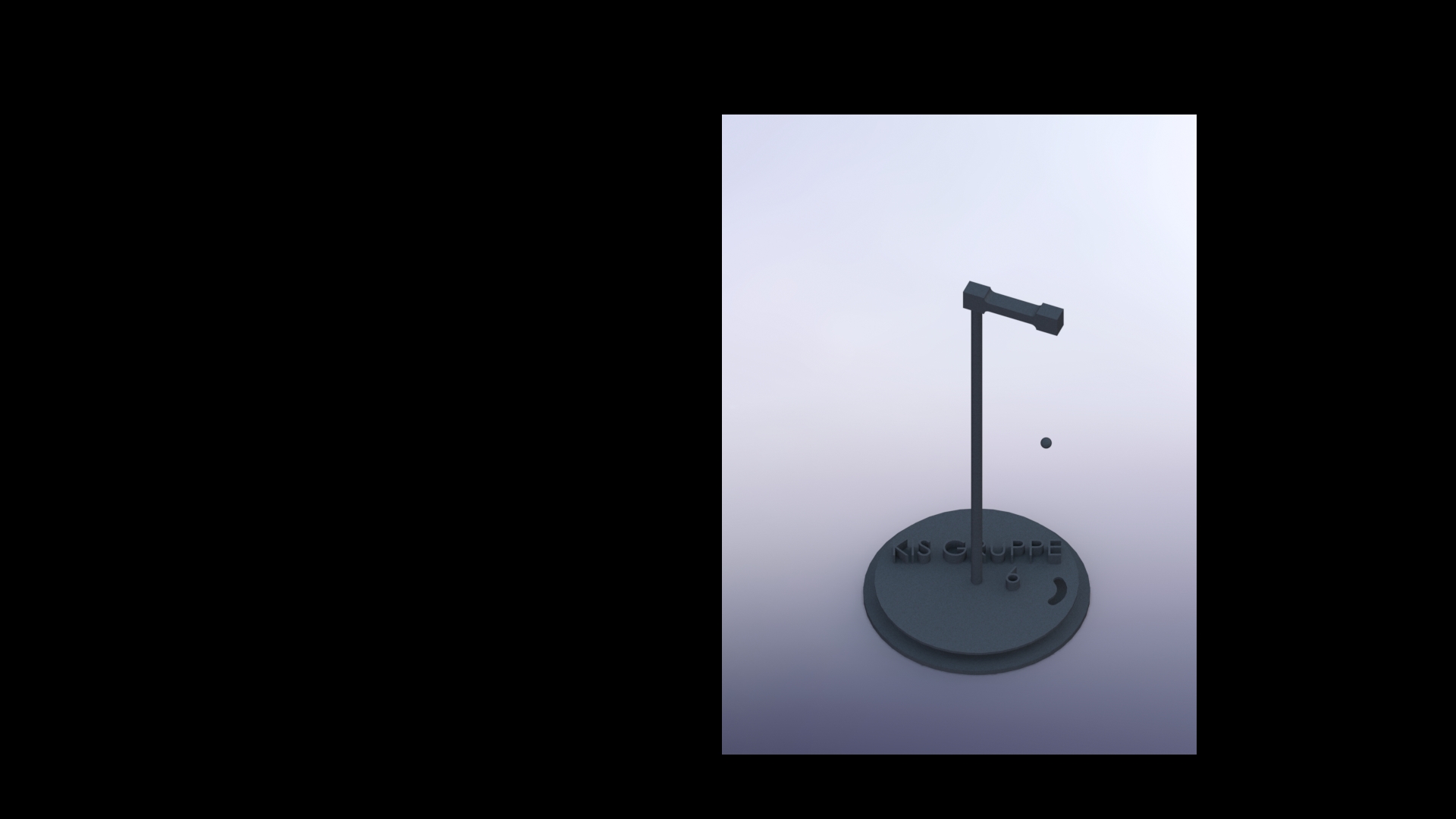
Der einzelne Aktor beim Versuch ist der Servo Motor. Ei Servo Motor besteht aus Motorsteuerung (1), einem Elektromotor (2), einem Getriebe (3) und einem Potentiometer zur Positionsbestimmung (4). Alle Komponenten sind in einem robusten Gehäuse untergebracht.

Abb 4 Servo-Motor

|  |
| --- |
|  |

Der Servo dient dem Abwurf und dem Nachladen der Kugel. Er wird anhand der Arduino intern Servo Bibliothek gesteuert. Er ist am Fallroher befestigt und bewegt zwei klein Metallplättchen, damit schließt er das Rohr und hindert den Fall der Kugeln.

### 3D Modul des Versuches:



## Drehscheibe:

|  |  |
| --- | --- |
|  |  |

Abb 6 Drehscheibe und Iso Ansicht für den Versuch (3d Modell)

Die Drehscheibe ist über eine Achse gelagert und hat ein Loch an einer Stelle, wodurch die Kugel fallen soll. Wie erklärt ist diese scheibe auf der Unterseite in weiß und schwarz Flächen geteilt. Jeder Flache versetzt 30°.  
Während des Versuches werden diese Teile benutzt, um die Position des Lohes in jedem Zeitpunkt wie genau wie möglich zu erfassen.

## Die Verschaltung:

### Die Verbindung mit Arduino:

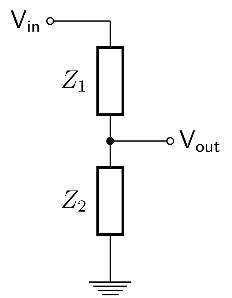
Die Komponenten, die bereits beschrieben sind und zusätzlich die Knöpfe, die zum Steuern den Verlauf des Programms benutzt werden, werden von einem Arduino ausgelesen und angesteuert. Die Pinbelegung des Arduinos wird im Tabelle 1 dargestellt.

Tabelle 1 Pinbelegung

|  |  |  |
| --- | --- | --- |
| **Arduino Pin** | **Mode** | **Funktion** |
| Pin 2 | Input | Photo-Sensor |
| Pin 3 | Input | Hall-Sensor |
| Pin 4 | Input | Trigger |
| Pin 5 | Input | Switch |
| Pin 7 | Output | Blackbox Led gelb |
| Pin 9 | Attached to Servo | Servo |
| Pin 10/11 | Input | Butt1/Butt2 |
| Pin 12/13 | Output | LED1/LED2 |

### Sensoren mit Arduino:

Der Zustand von den Sensoren wird anhand des „DigitalRead“ Befehls ermittelt: Das hat die Bedeutung, dass wir mit Digital- Logik arbeiten müssen also entweder (0 ,1 oder HIGH, LOW) oder mit dem entsprechenden elektrischen Signal (5 V, 0V). Allerdings arbeiten die beiden Sensoren mit 12V Spannung und liefern auch Ausgangsspannung von 12V, deswegen wurde Voltage-Divider (siehe Abbildung) für 5v Stufe benutzt.



### Die Knöpfe (The Buttons):

Damit die Knöpfe richtig funktionieren sollen wir den sogenannten Pullup- oder Pulldown- Widerstand benutzen.

Ein Pullup- oder Pulldown-Widerstand wird dazu verwendet, einen logischen Eingang auf einen definierten Wert zu "ziehen". Normalerweise befindet sich der Eingang im Zustand "schwebend/hochohmig", welcher sich irgendwo zwischen High und Low befindet. Nun sind Schaltungen leider nicht komplett ohne Störsignale, und durch Einstrahlungen von Signalen kann es nun passieren, dass kurzzeitig mal ein Wert über- oder unterschritten wird und der Eingang plötzlich ein High- oder Lowsignal bekommt. Dies führt dann zu unerklärlichen und unregelmäßig auftretenden Fehlern, die einen Neuling zur Verzweiflung treiben können.

|  |
| --- |
|  |

### Sensor Valuesänderungen:

#### Signalwechsel am Photosensor:

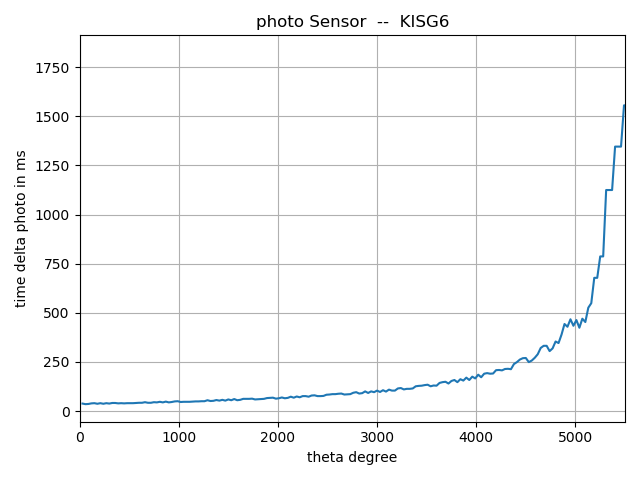
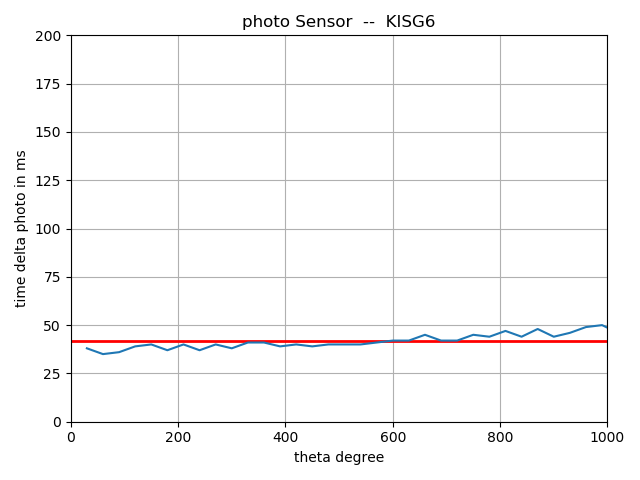
Am Anfang haben wir die Zeit anhand des Photosensors gemessen, die die Drehscheibe braucht, um 30° zu bestehen. Diese Zeiten wurden von 30° bis 5500° gemessen und mit aller möglichste schnelle Geschwindigkeit ca. 3.5 U/s.

Abb7 1 time delta photo sensor

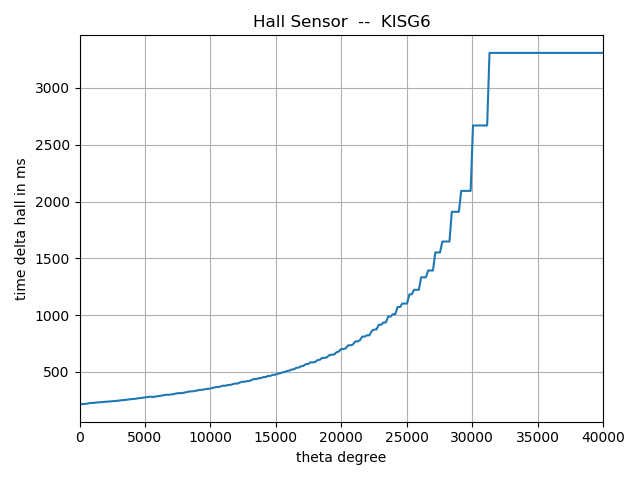
Danach haben wir andre Betrachtung gemacht bis 1000°, wo sich ein Mittelwert von ca. 41.5 ms ergibt.



*Abb7 2 time delta photo sensor mit Average*

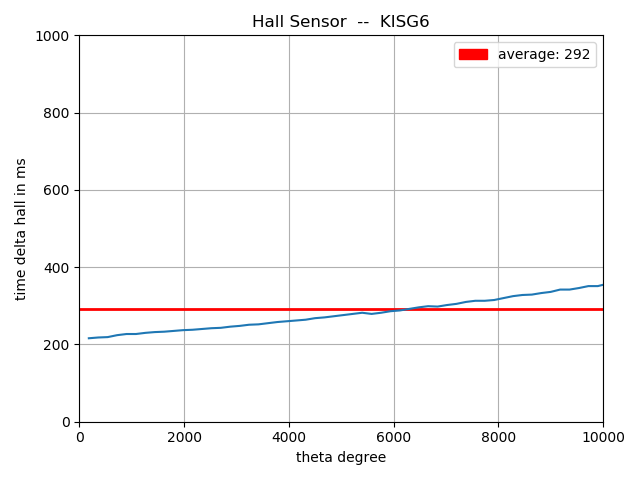
#### Signalwechsel am Hall-Sensor:

Wie bevor aber die Betrachtung hier von (180° bis 40000°) mit der gleichen Anfangsgeschwindigkeit 3.5U/s.



*Abb 8 1 time delta hall sensor*

Bis 10000° haben wir neue Betrachtung gemacht um einen Mittelwert zu bekommen, die ca. 292 ms war.

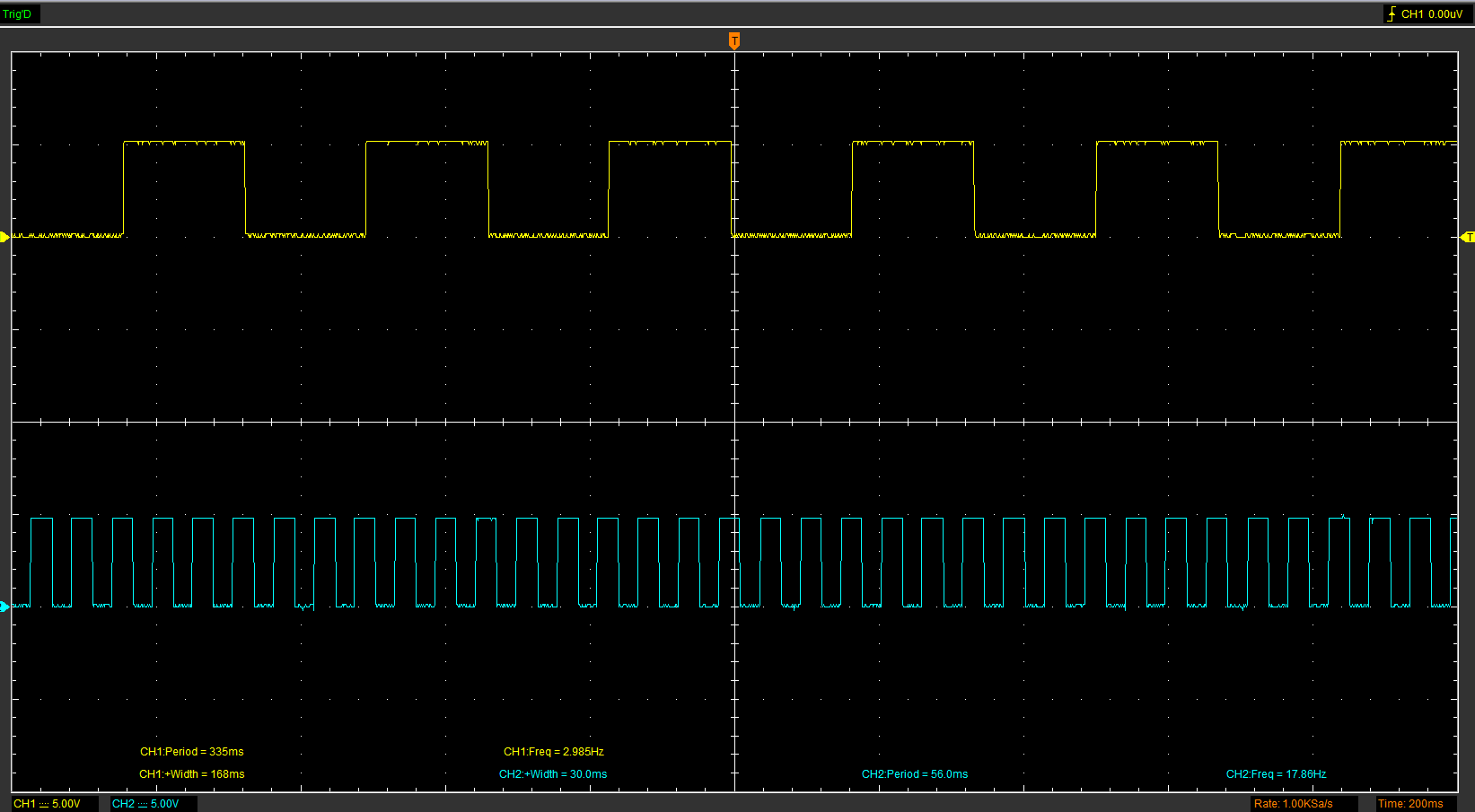


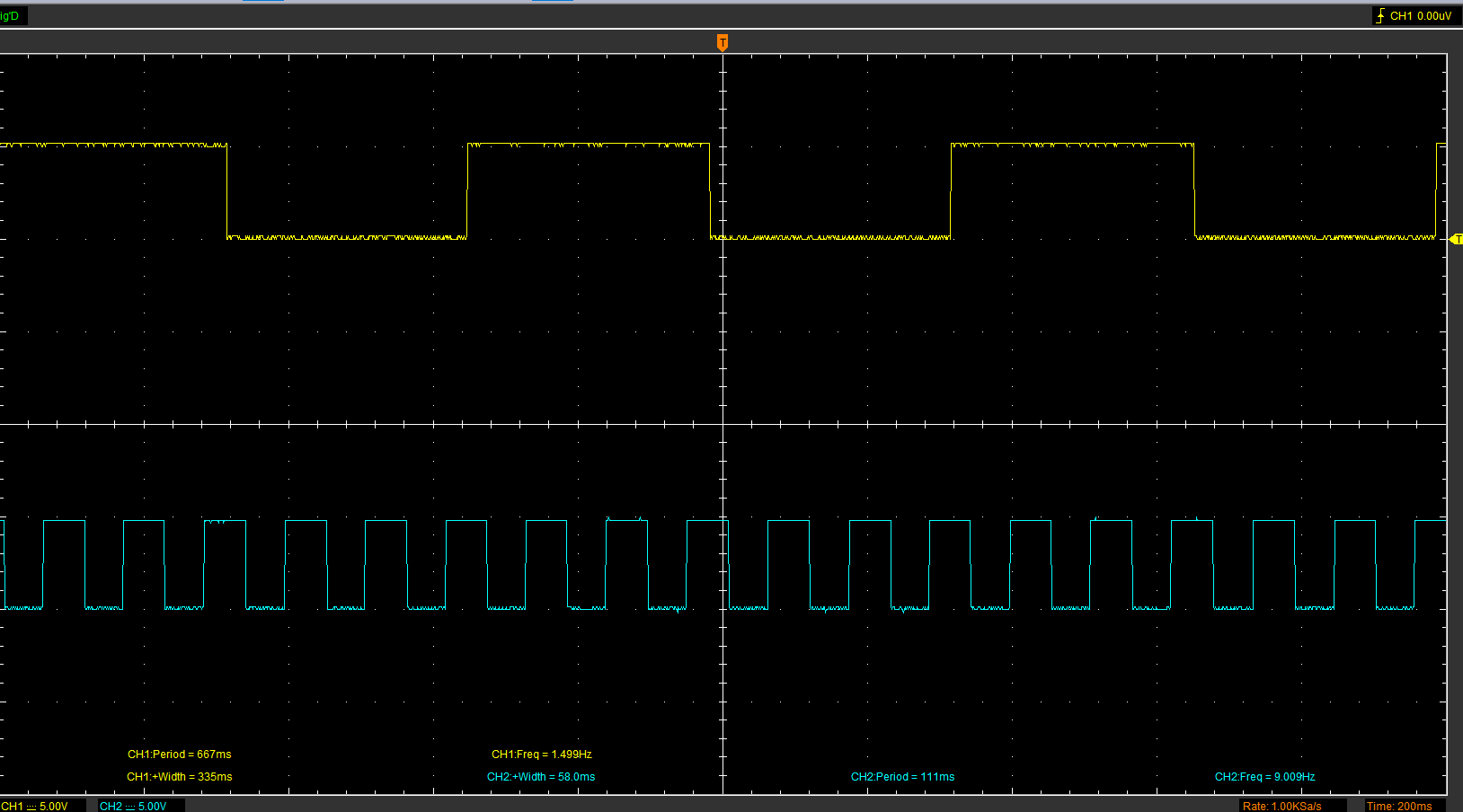
*Abb 8 2 time delta hall sensor mit Average*

### Die Genauigkeit der Sensoren:

Um die Genauigkeit der beiden Sensoren zu messen, wurde einen Hantek (IDSO1070 A Digital Oszilloskope 2CH 70MHZ/250MS) Digital Oszilloskope benutzt.

Die folgenden Werte beweisen, dass die beide Sensoren Fehlerfrei und ganz genau arbeiten, aber das bedeutet nicht, dass der Arduino diese Signale genau bearbeiten und erfassen kann!





Hantek 1 Fast Speed

Hantek 2 Medium Speed

# Design:

## Die Theorie der Lösung:

Es handelt sich hier um ein physikalisches Problem, die die Zeit, Geschwindigkeit, Beschleunigung, Winkel und Strecke verbindet. Dieses Problem wurde als gleichförmig beschleunigtes Kreisbewegungsproblem behandelt, deswegen wurden die Kreisbewegungsgleichungen benutzt.

Zuerst wurde die Zeit, die die Kugel bis zum Loch braucht, anhand des Gesetzes des freien Falls.

Der freie Fall ist eine spezielle gleichmäßige beschleunigte geradlinige Bewegung ohne Anfangsweg und Anfangsgeschwindigkeit. Als beschleunigende Kraft wirkt nur die Gewichtskraft des Körpers. Die Beschleunigung wird Fallbeschleunigung genannt.

Das Fallgesetz lautet:

*Gleichung (1)*

g: die Erdbeschleunigung und beträgt 9.81

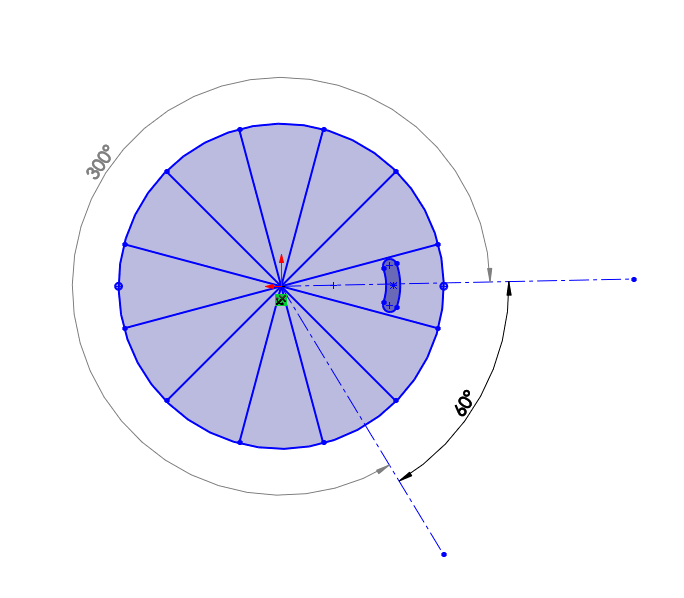
s: die Länge der Strecke und beträgt 75cm=0.75m

Davon ergibt sich eine Fallzeit von ca. 391ms.

Zunächst wurde die Position des Loches anhand der Sensoren erkannt und ermittelt, damit wir rechnen können, wie lang braucht es bis zum 0 Position zurückzukommen und die totale Umdrehungszeit. Wir haben zwei Zähler, ein ist für die Photo\_Sections und der andre ist für Hall\_Section.

Photo\_Sections Zähler zahlt von 0 bis 11 und damit werden 12 Teile gezahlt. Das ist wichtig um den Winkel zu bekommen alle unsere Berechnungen basieren auf dem Winkel.

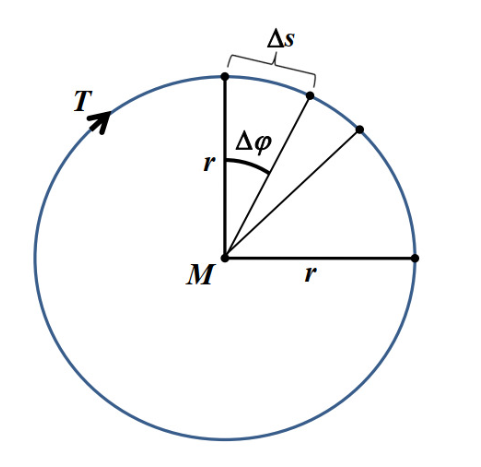
Es wurde immer mit Radianten gearbeitet, damit wir physikalische Gesetzte problemlos benutzen können.

 z.B. wenn photo section = 2 dann und der verbleibende Winkel bis zum 0 ist ().

rest to null

*Abb 9 Winkel und Winkel-Rest*

### Parameter von Kreisbewegungen:



**r** Radius des vom Bezugspunkt (z. B. Massepunkt) beschriebenen Kreises um Mittelpunkt **M**

**T** **Umlaufzeit** [s]

**φ** **Umlaufwinkel** (1 Umlauf = 2π)

**Δs** **Weg** Δs = r • Δ φ

**Drehfrequenz**

**ω** **Kreisfrequenz**

*Abb 10 Parameter von Kreisbewegungen*

Die **Kreisfrequenz** **ω** ist eine Geschwindigkeit, die **Winkelgeschwindigkeit**.

Sie gibt die **Änderung des Umlaufwinkels** an: *Gleichung (2)*

### Formeln der Kreisbewegung:

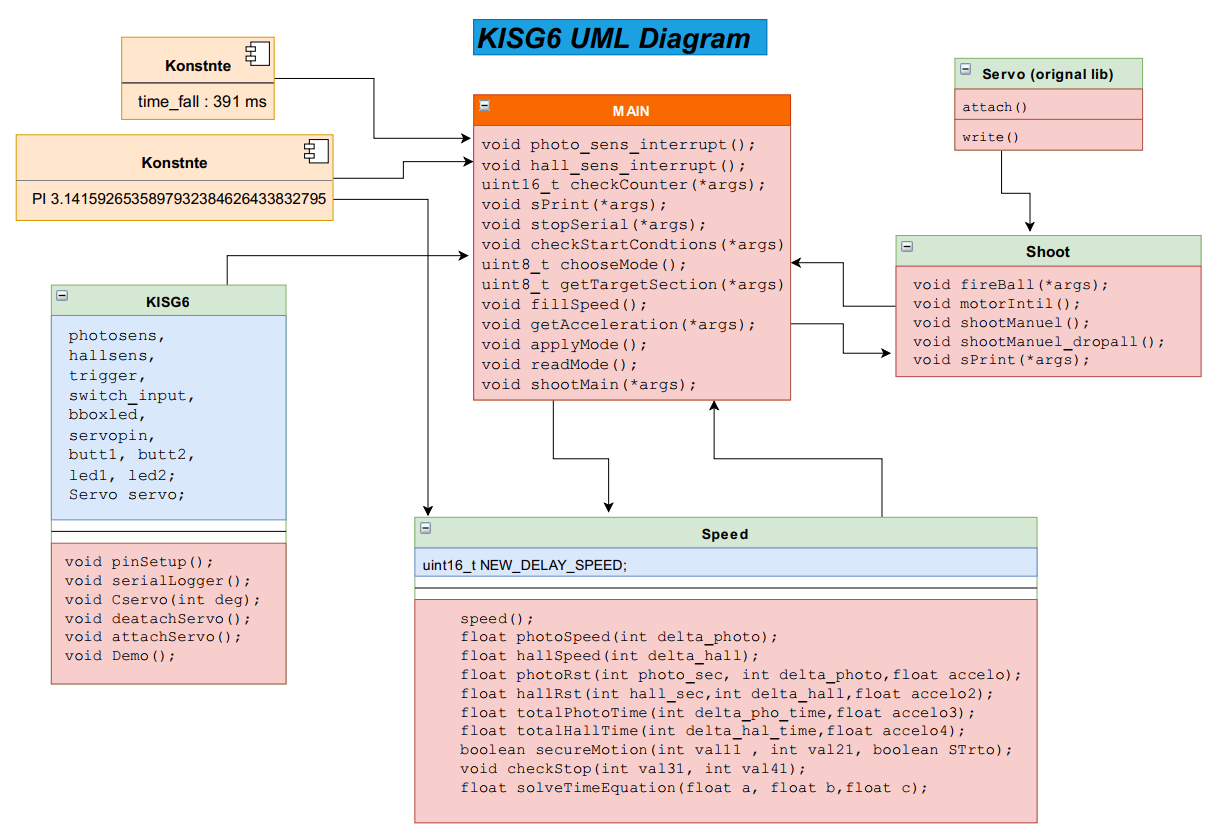
*Gleichung (3)*

*Gleichung (4)*

Die Gleichungen (2,3,4) spielen die Hauptrolle in dieser Lösung, so dass 2 der Rechnung der , 3 der Rechnung der Zeiten (total, Rest), 4 der Rechnung der Geschwindigkeit dienen.

**Bemerkung**: obwohl es sich hier um eine regelmäßige beschleunigte Bewegung handelt, weil keine andren oder äußeren Kräfte auf die Drehung bewirken sollten, ist aber nicht der Fall und wir haben große Änderungen beim Beschleunigung betrachten. Der Grund dafür ist große Reibungskraft.

## Klassen Diagramm:



*Abb 11 Klassendiagramm*

## Workflow Diagramm:

*Abb 12 Workflowdiagramm*

Das Programm basiert auf drei wichtigen Klassen:

1. KISG6 Klasse: in dieser Klasse wurde alle Versuchsparameter definiert und vorbereitet, damit wir sie einfach im Programm benutzen-
2. Speed: in Speed werden alle wichtigen Berechnungen gemacht, um die Zeit und Geschwindigkeit zu bekommen. Zusätzlich hat sie die Securemotionfunktion. Diese Funktion sorgt dafür, dass sie jede plötzliche unerwartete Bewegung erkennt und das Programm sperrt.
3. Shoot: sie hat zwei Aufgaben. Erste ist den Motor zu initialisieren und Zweite ist den Ball zu schießen.

Die drei Klassen sind durch die Main-Block verbunden. In Main-Block werden die Informationen von die Sensoren-Interrupts erfasst, z.B. Photo\_Section, Hall\_Section, Time\_Delta\_Photo, Time\_Delta \_Hall und werden die aktuelle Geschwindigkeit und Beschleunigung berechnet. Diese Werte werden als Anfangsbedingungen in Speed benutzt. Die Hauptfunktionen werden zunächst erkläret.

## Main Hauptfunktionen:

1. Die Interrupt-Funktionen:

*void* photo\_sens\_interrupt();

*void* hall\_sens\_interrupt();

Die beiden dienen der Berechnung der Zeit zwischen zwei nacheinander Signale

vom Sensor und damit die Position des Lochs in jeder Zeit.

1. Checkstart:

*void* checkStartCondtions(*uint8\_t* *hall\_seco*, *uint8\_t* *hoto\_cso*);

hier wird die Position der beiden Sensoren geprüft. Wenn die beide in 0 Sind dann können wir die Scheibe drehen und anfangen.

1. Get-Acceleration:

*void* getAcceleration(*char* *x\_sens*);

je nachdem welcher Sensor wollen wir die Beschleunigung berechnen “h“ steht für Hall Sensor und “P“ für photo sensor.

1. Mode-Funktionen:

*uint8\_t* chooseMode();

*void* applyMode();

*void* readMode();

Sie werden benutzt um ein von 4 Mode zu wählen:

1. Hall Mode: nur Hall Sensor.
2. Photo Mode: nur Photo Sensor.
3. Hall-Photo: die Beide Sensoren werden benutzt.
4. Manual Mode: hier wird einfach einen Ball geschossen.

## Speed Hauptfunktionen:

1. Aktuelle Speed:

*float* photoSpeed(*int* *delta\_photo*);

*float* hallSpeed(*int* *delta\_hall*);

Anhand der Gleichung (2) und der Deltazeit wird die Geschwindigkeit berechnet.

Die Einheit hier Rad/s.

1. Die verbleibende Zeit bis zum 0:

*float* photoRst(*int* *photo\_sec*, *int* *delta\_photo*,*float* *accelo*);

*float* hallRst(*int* *hall\_sec*,*int* *delta\_hall*,*float* *accelo2*);

die Photo\_Section und Hall\_Section ergeben wie viel Winkel hat das Loch bis jetzt  
 besteht, dann können wir einfach der verbleibende Winkel bis zum 0 berechnen,

indem wir der aktuelle Winkel vom 2PI subtrahieren. Die Zeit wird anhand der solveTimeEquation-Funktion ermittelt.

1. Die Totalzeit:

*float* totalPhotoTime(*int* *delta\_pho\_time*,*float* *accelo3*);

*float* totalHallTime(*int* *delta\_hal\_time*,*float* *accelo4*);

Die Zeit um 2PI mit der aktuellen Geschwindigkeit zu bestehen. Die Zeit wird anhand der solveTimeEquation-Funktion ermittelt.

1. Lösung der Zeitgleichung:

*float* solveTimeEquation(*float* *a*, *float* *b*,*float* *c*);

hier wird die Gleichung (3) benutzt und bekommen wir nur die kleinste Zeitlösung,

aber wenn die Zeitlösung großer als 15 s bekommen nur eine Rückmeldung, dass Lösung ignoriert wurde.

1. Untypische Bewegungen:

boolean secureMotion(*int* *val11* , *int* *val21*, boolean *STrto*);

hier wird die time\_delta von den beiden Sensoren immer geprüft ob sie plötzlich sehr groß geändert wird oder normal und bekommen wir entweder True für normal oder False für untypische Bewegung.

## Shoot Hauptfunktionen:

Ball Schießen:

*void* fireBall(*float* *time\_resto*,*uint8\_t* *section*, *float* *total\_time* , *uint16\_t* *window*,*uint16\_t* *target\_time*);

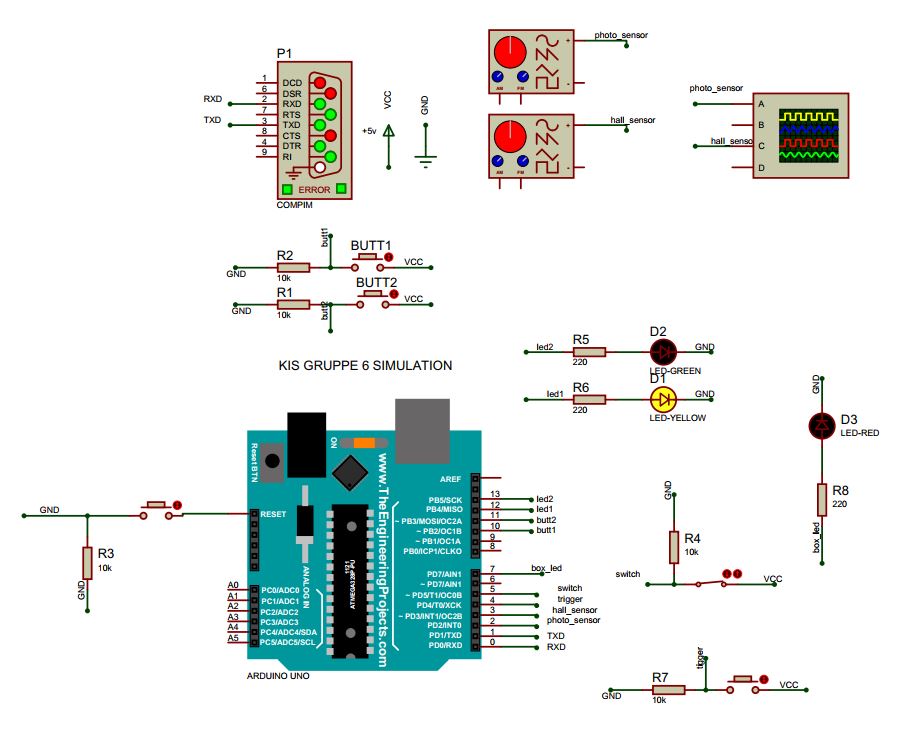
Diese Funktion wird die Zeitlösung benutzen um einen Ball zu schießen. Würde die Zeitrest großer als die Fallzeit würde ein Delay erstellt. Aber wenn sie kleiner als die Fallzeit wurde die Totalzeit benutzt um ein Delay zu erstellen.

# Simulation

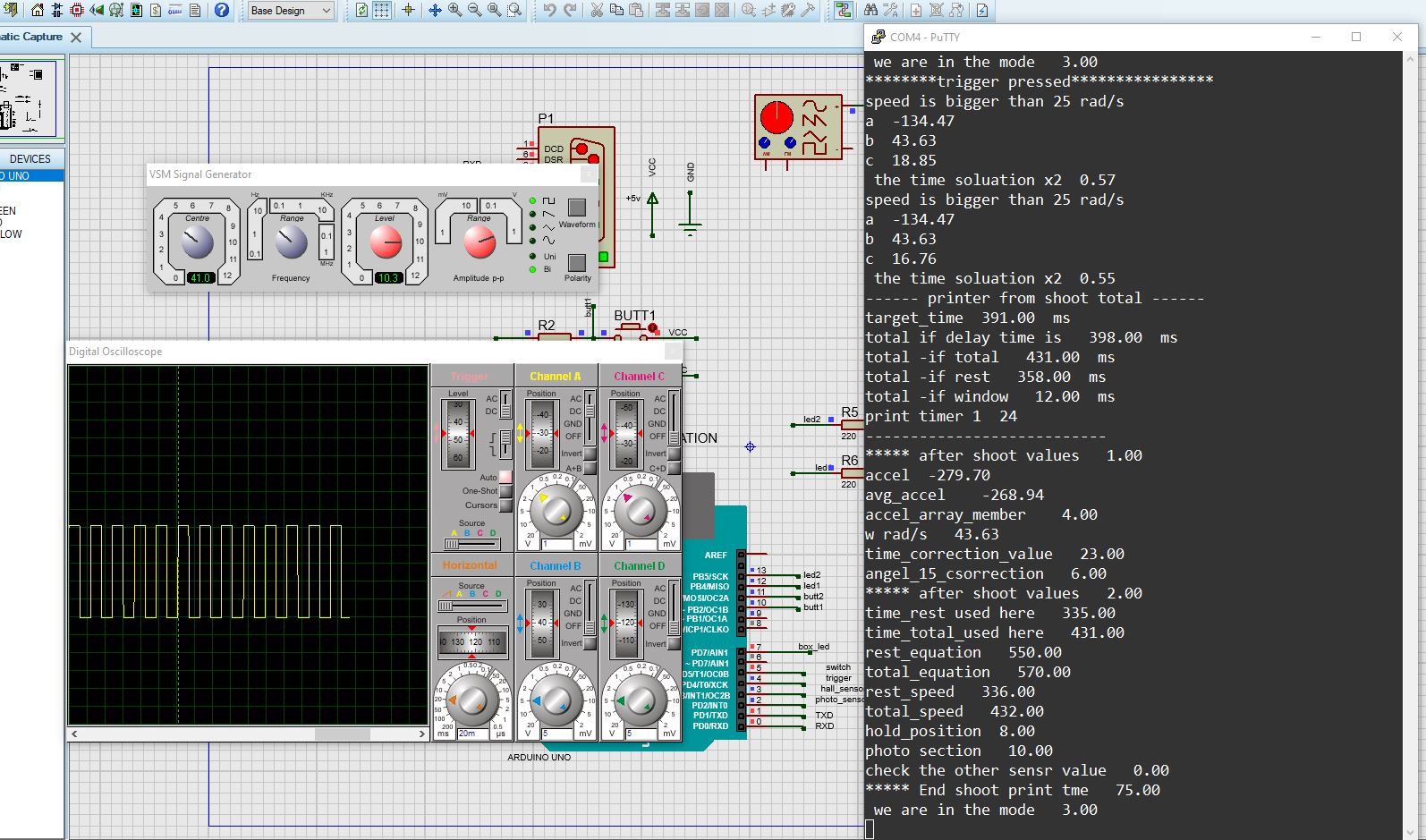
Die benutzte Software:

1. Proteus
2. VSCODE (Visual Studio Code Editor)
3. PlatformIDE Plugin für VSCDOE
4. VSPE (Visual Serial Port Emulator)
5. PUTTY

Diese Simulation wurde gemacht, um den Code unabhängig von Labor Terminen zu validieren und verbessern. Damit konnten wir unsere Lösung mit Geschwindigkeiten von ca. 40 rad/s probieren (6.3 U/s).



*Abb 13 Simulation Schematic*



Hier können wir ein Beispiel sehen, wo die Geschwindigkeit betrug 43 rad/s und das Programm hat sie problemlos behandelt. Wir haben ein Debug-System entwickelt, was sehr hilfreich war und hat uns sehr gut bedienen. Das wurde zunächst erklärt.

*Abb 14 Simulation Output*

### Das Debug-System:

Anhand des (Serial.print) Befehls können Daten während Des Laufs des Programms und damit viele wichtige Informationen bekommen. Natürlich ist die Nutzung von vielen Serial.print Befehlen sehr Speicher aufwändig und Ausführungszeit – ca. 1ms pro Befehl- braucht, aber alles wurde berücksichtigt und gerechnet, damit es keinen Einfluss auf das Programm haben kann.

|  |
| --- |
|  |

* Programm Mode (1,2,3,4)
* Zeitgleichungsparameter und Lösung für Ttotal\_Time und Rest\_Time.
* Welche Shoot\_If benutzt wurde und mit welchen Parametern.
* Es gibt Total\_If und Rest\_If
* Hier können wir alle wichtigen Parameter sehen und prüfen.
* Beschleunigung, Winkelgeschwindigkeit und die Sensorensections.
* Time\_correction\_Value und Angel 15\_correction Value.
* Section bevor der Rechnung und Section nach Der Rechnung.
* Andere Sensor Value.

*Abb 15 Debugger Output*

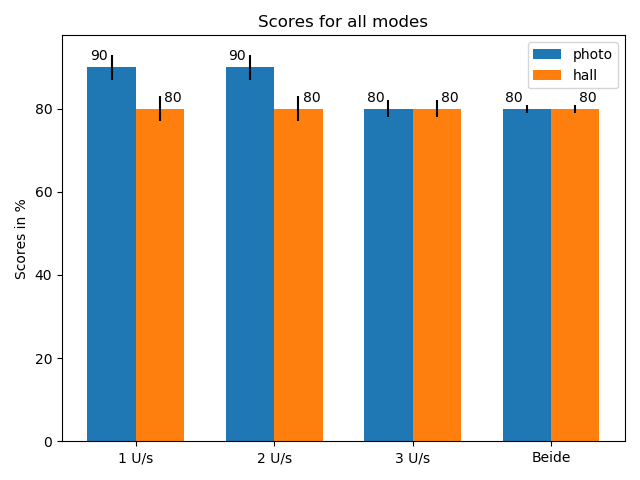
Die Datei der Simulation befindet sich mit dem Quellencode.

# Implementierung

Um den Code zu nutzen, werden VSCODE mit PlatformIDE statt ArduioIDE gebraucht.

## Validierung :

In einen Test über mehrere Geschwindigkeiten konnte eine Treff Quote von (100%-90%) mit Geschwindigkeit unter 17 rad/s nachgewiesen werden. Dieses Ergebnis ändert sich jedes Mal wir es versuchen und manchmal bekommen wir 10% Treff Quote obwohl es vorher 90% war. Es konnte daranlegen, dass es Veränderungen am Versuchaufbau geben könnte.



*Abb 16 Score for all Modes*

Bis diesem Zeit Punkt 12.07.2019 befindet sich den Code in Beta-Version und es wird verbessern um besseres Ergebnis mit hohen Geschwindigkeiten zu bekommen.

### Bemerkung:

Diese Lösung wurde an den beiden Versuchen (Fenster, Tür) probiert und wir hatten manchmal das gleiche Ergebnis. Trotzdem wir haben uns für Kugellfalltür entschieden., woran unsere Lösung getestet werden soll.

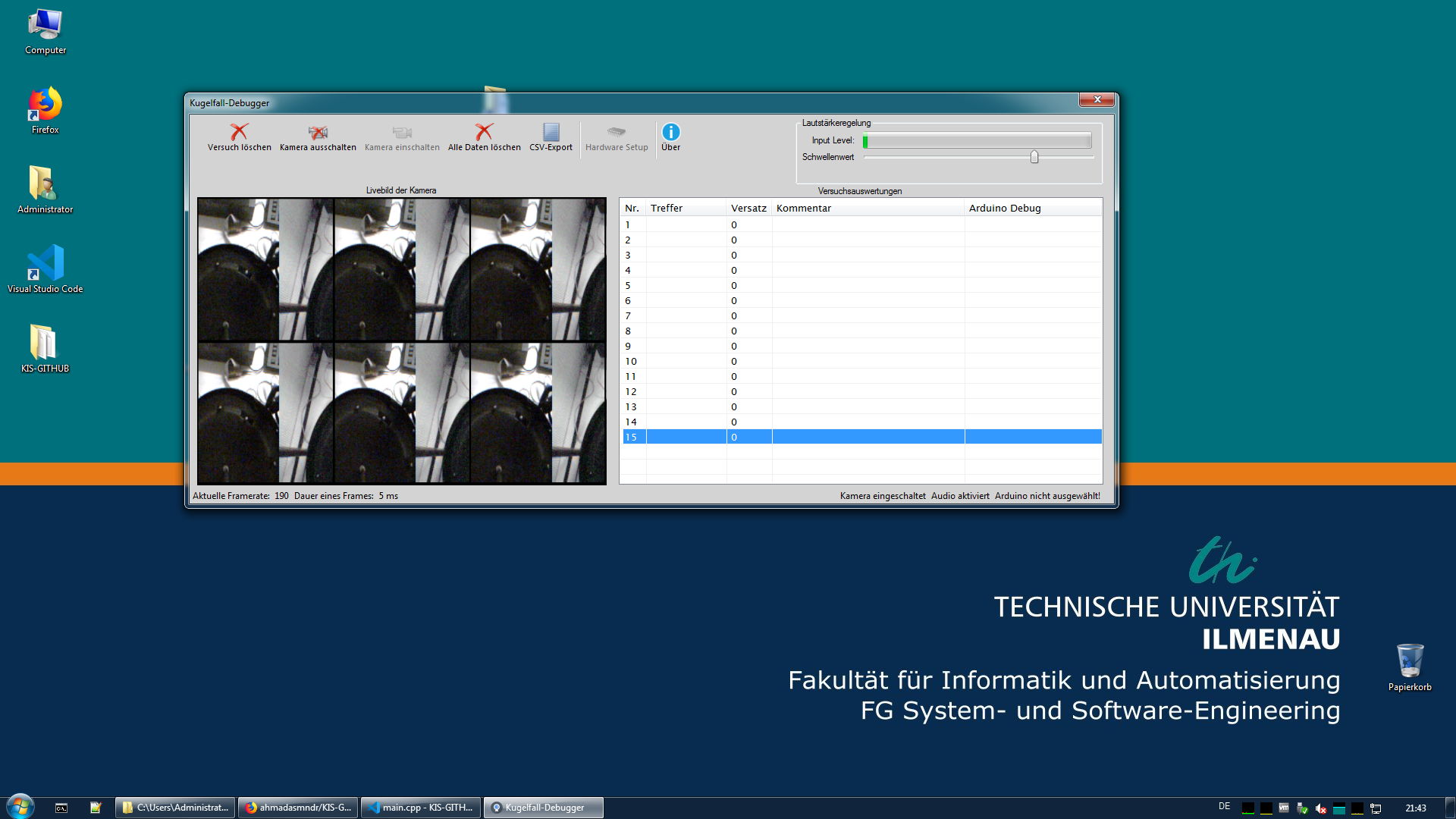
Bevor dem Andrehen soll das Loch so justiert, dass der Ball genau drin durchfallen muss. Die Rückmeldung von der Checkstart- Funktion soll berücksichtigt werden d.h. wenn die grüne Led leuchtet soll die Drehscheibe justiert und den Arduino neugestartet werden.

Um bessere Ergebnis zu bekommen könnte den Black\_Box aus und wieder angeschaltet werden, sodass wir versichern, dass das Hall-Sensor seinen Wert nicht sofort ändert.

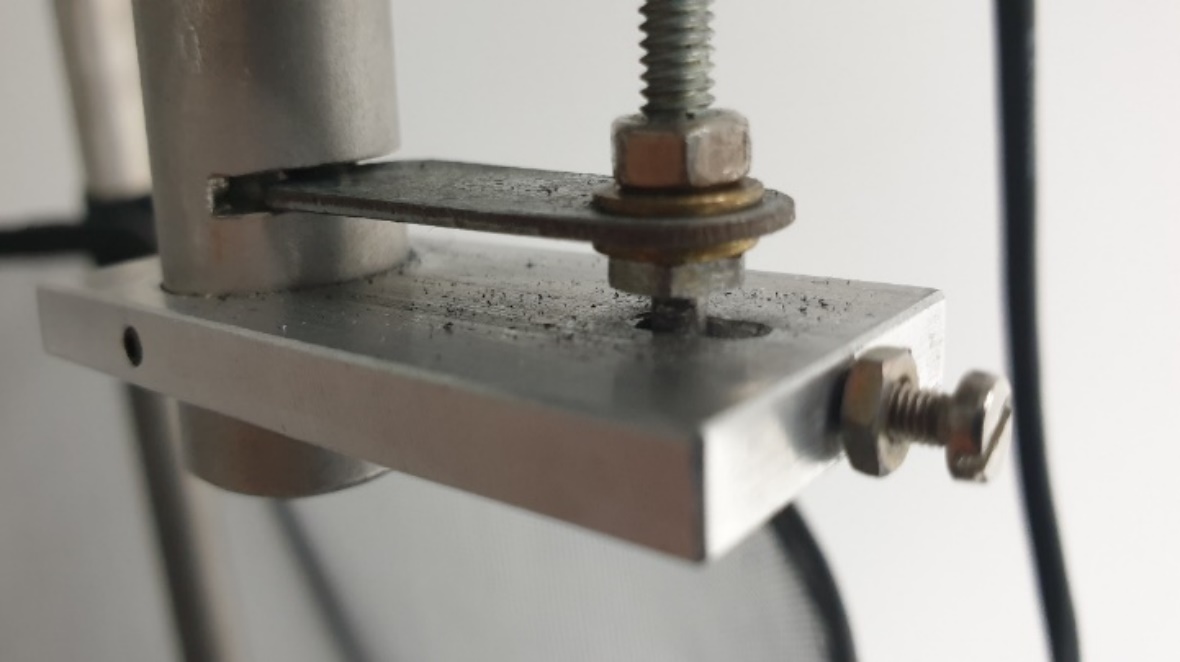
## Fehler Amylase:

Während unserer Arbeit am Versuch haben wir viele Fehler gemerkt:

1. die Drehscheibe dreht sich unregelmäßig und dadurch ergibt sich Schwingungen, die den Durchfall von Ballen verhindert.
2. Den Ball fällt an den Runden des Lochs. (Abb 17)

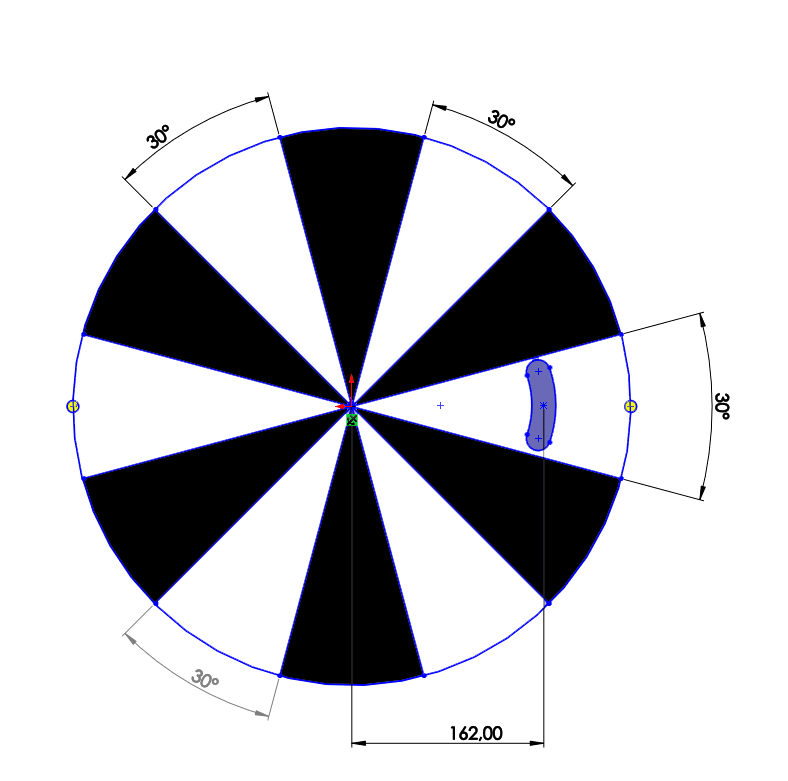


*Abb 17 Kugel trefft den Rand*

1. Es gibt große unregelmäßig veränderte Reibungskräfte, die der Änderung der Geschwindigkeit dient.
2. Stoßt ein Ball die Drehscheibe, ergibt sich ein schiefer unelastischer Stoß, der die mechanische Energie ändert und damit die Geschwindigkeit.
3. Die Reaktionszeit des Motors wird durch Reibungskraft verlängert, die sich am Treffpunkt von Blächenachse und dem Blatte. (Abb 18) 

*Abb 18 Reibungskraft am Motor*

1. Die Fallzeit ist zwar bewusst, aber es ist nicht genug um einen Durchfall zu erreichen. Wir haben Target\_Time von ca. 500ms manchmal benutzt um einen Durchfall zu erreichen. Diese Werte besteht aus Drei Parameter:
2. Motor\_Dealy: der in unserer Lösung beträgt 100ms und das ist die Zeit, die der Ball braucht um ihr Trägheitsmoment zu überwinden und fällt.
3. Angel\_15\_: der Versuch wurde so aufgebaut, dass die beiden Sensoren nicht gleichzeitig ausgelöst werde können, dadurch dass der Photosensor den Hall bei 15° überholt.



15°

1. Time \_delta: die Zeit zwischen zwei nacheinander Signalen vom Sensor und sie beträgt im Fall von Photosensor die ganze time\_delta aber im Fall vom Hall (time\_delta/6). Das wurde als Zeitfenster betrachtet.

### Die Lösung und Optimierung:

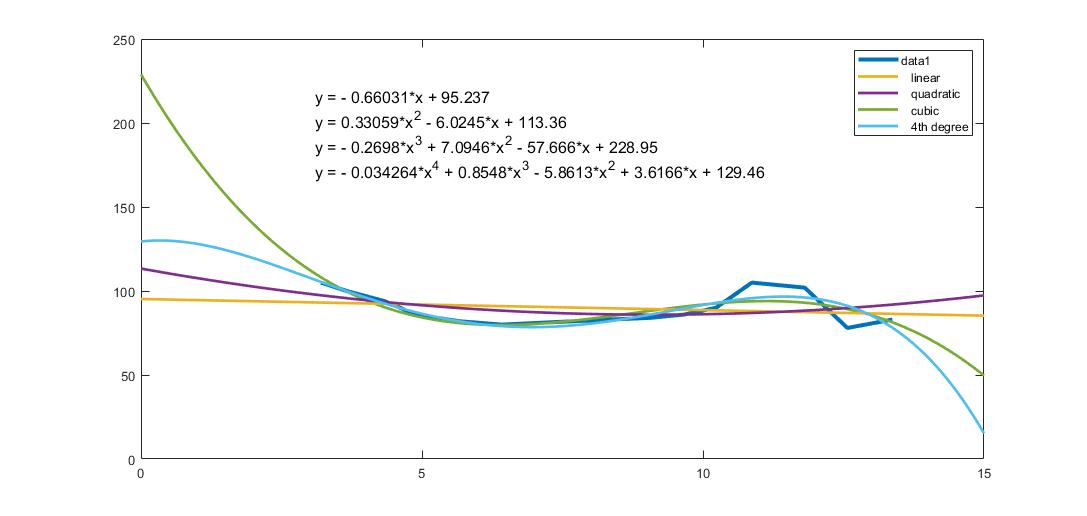
Alle Probleme außer (2 und 5) lassen sich teilweise durch Programmierung lösen.

Die äußeren Kräfte lassen sich anhand des Newtonschen-Gesetzes bestimmen:

Gleichung (6)

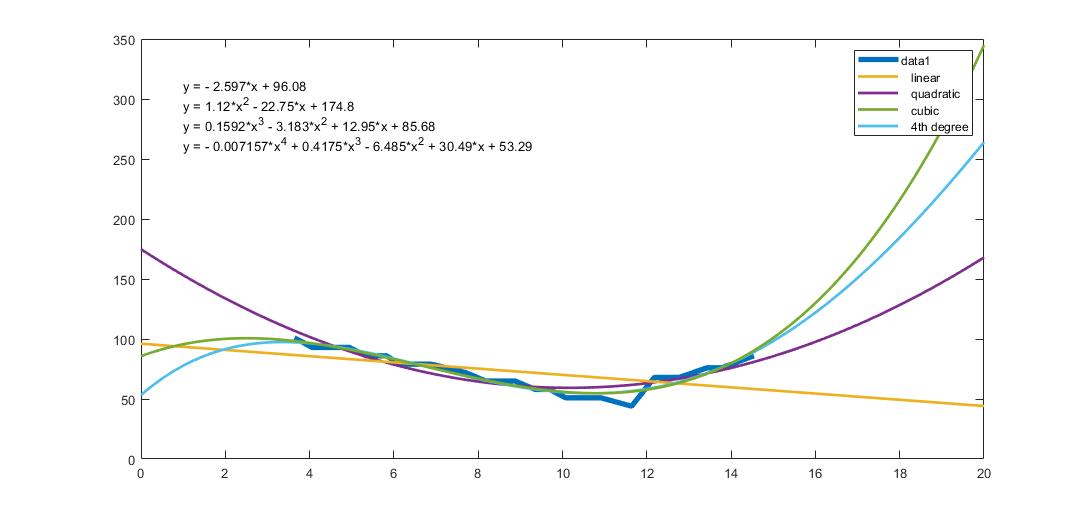
Durch die Berechnung der Beschleunigung in jedem Zeitpunkt können wir diese Kräfte bestimmen.

Es bleibt aber 6, die zusätzliche Zeit die wir addieren müssen um einen Durchfall zu erreichen. Um dieses Problem zu lösen, haben wir alle Durchfälle-Daten, die wir schon gesammelt haben (alle Daten befinden sich mit dem Quellen-Code), anhand Python und Matlab analysieren. Dadurch haben wir ein Verhältnis zwischen der Geschwindigkeit und der zusätzlichen Zeit bekommen.



*Angular\_speed rad/s*

*Addition\_time\_ ms*

ANA 1 Hall\_Time\_Correction \_Functions

*Addition\_time\_ ms*

*Angular\_speed rad/s*

ANA 2 Photo\_Time\_Correction \_Functions

### Fazit:

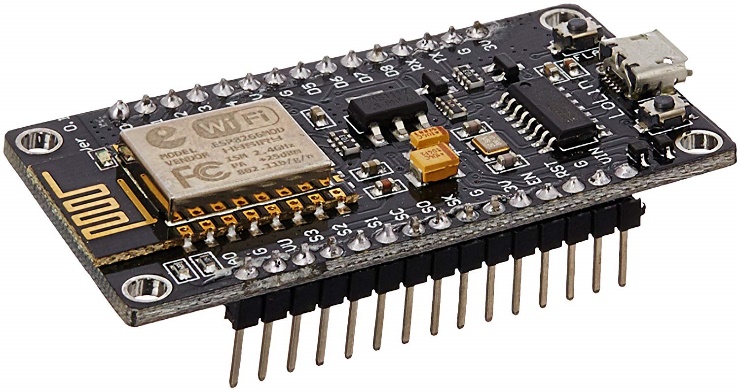
Die Lösung erfüllt die Anforderungen, braucht trotzdem eine Verbesserung, damit sie mit hohen Geschwindigkeiten arbeiten kann.

Der Versuch ist alt und die Komponenten sind stark ausgenutzt. Arduino kann nur ca. 10.000 Mal programmiert werden dann muss ausgetauscht werden.

### Empfehlung zur Verbesserung des Versuchs:

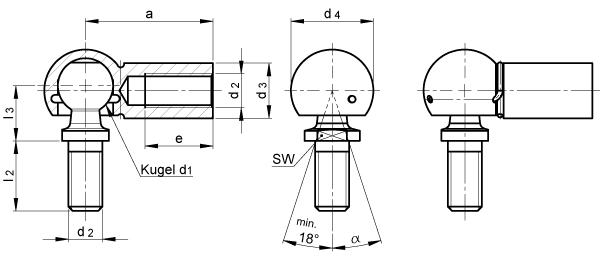
1. Esp8266 statt Arduino ,weil :
2. 32-Bit-Mikrocontroller statt 8-bit !!
3. 80Mhz -160Mhz statt 16mhz Arduino
4. 4Mb Flash Memory statt 32Kb an Arduino!!!!
5. Ist günstiger als Arduino 2€ statt 5€ für Arduino (ohne Ethernet-Shield oder Bluetooth)
6. 8 Interrupts Statt 2
7. Eingebettete WiFi und Bluetooth Modul

Die Esp8266 (Nodmcu) benutzt auch Arduino-Plattform und wird gleich wie Arduino programmiert.



*Abb 19 Nodmcu Esp8266*

1. Kugelgelenk nach DIN 71802 um die Reibung am Motorachse zu verkleinern und damit die Reaktionszeit zu verbessern.



*Abb 20 Kugelgelenk DIN 71802*

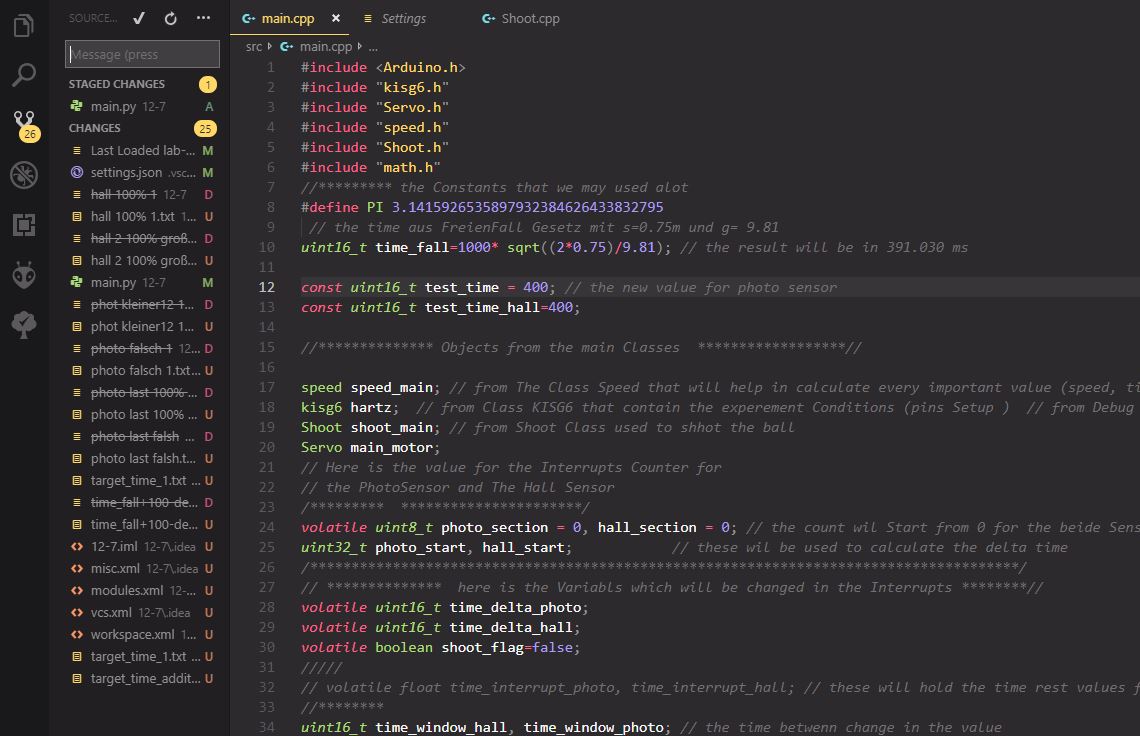
# How To:

Um diese Lösung zu nutzen, brauchen Sie:

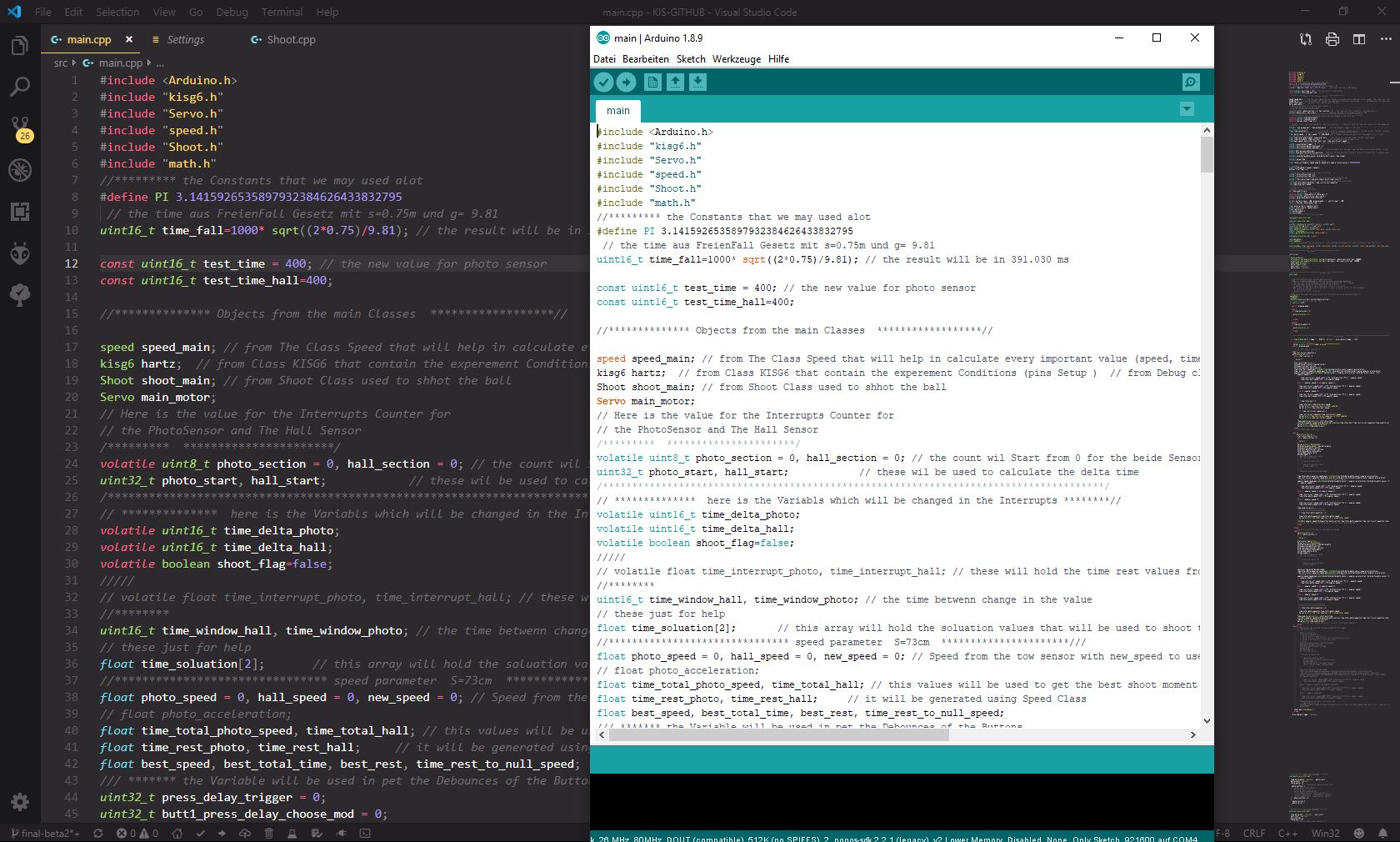
1. Vscode die (nur die Editor Link befindet sich unten 5)
2. PlatformIDE Plugin für Vscode (einfach durch vscode installiert werden kann)
3. Git
4. <https://github.com/ahmadasmndr/KIS-GITHUB.git>
5. https://code.visualstudio.com/download

Der Quellen-Code befindet sich in GitHub unter dem Link in 4.

### Warum VSCODE:

Visual Studio Code (VS Code) heißt das neueste Entwicklerwerkzeug von Microsoft. Dem Namen nach könnte man denken, dass es sich um die neueste Edition von Visual Studio handelt, allerdings ist VS Code eher der flexible kleine Bruder: Ein schlanker Codeeditor der Software-Entwickler nicht nur beim Editieren, sondern auch beim Verwalten und Testen von Code unterstützt. Insbesondere lassen sich Codedateien außerhalb von Projekten schnell mit VS Code bearbeiten, ohne dabei auf die gewohnte Unterstützung seitens einer IDE verzichten zu müssen. Dabei wird nicht nur die Syntax farblich hervorgehoben, sondern auch andere von IDEs bekannte Funktionen zur Verfügung gestellt. Der Clou bei VS Code ist allerdings, dass es nicht nur unter Windows, sondern auch unter MacOS und Linux läuft.

*Abb 21 VSCODE IDE*



*Abb 21 VSCODE vs. Arduino IDE*