Inhalt

[Abbildungsverzeichnis 2](#_Toc44523643)

[Vorwort - Laurin 3](#_Toc44523644)

[Verschiedene Ideen zur Umsetzung - Laurin 3](#_Toc44523645)

[Darts - Can 5](#_Toc44523646)

[Dartscheibe 5](#_Toc44523647)

[Abmessungen der Dartscheibe 5](#_Toc44523648)

[Dartvarianten 6](#_Toc44523649)

[Die Berechnung der Pfeilkoordinaten - Laurin 7](#_Toc44523650)

[Berechnung der Punktzahl aus der Pfeilkoordinate - Laurin 10](#_Toc44523651)

[Raspberry Pi 3B+ - Can 12](#_Toc44523652)

[Was ist ein Raspberry Pi? 12](#_Toc44523653)

[Was kann man mit einem Raspberry Pi machen? 12](#_Toc44523654)

[Aufbau des Raspberry Pi 3B+ 13](#_Toc44523655)

[Arduino - Laurin 15](#_Toc44523656)

[Schnittstelle zwischen dem Arduino und dem Raspberry Pi - Can 16](#_Toc44523657)

[Die Serielle Schnittstelle für unseren Seminarkurs - Can 20](#_Toc44523658)

[Software Dokumentation Python - Can 21](#_Toc44523659)

[Game 21](#_Toc44523660)

[Kontrolle der Spielregeln 24](#_Toc44523661)

[ArduinoInterface 26](#_Toc44523662)

[Location 27](#_Toc44523663)

[Prinzip des Ortens des Dartpfeils 28](#_Toc44523664)

[Die GUI – Pascal 30](#_Toc44523665)

[Programmierung der GUI 32](#_Toc44523666)

[Software Dokumentation Arduino - Laurin 33](#_Toc44523667)

[Materialien des Seminarkurses - Can 37](#_Toc44523668)

[Anbringung der Mikrofone an der Dartscheibe - Laurin 38](#_Toc44523669)

[Der Aufbau und Tests der Dartscheibe - Laurin 40](#_Toc44523670)

[Mögliche Gründe und Verbesserungen für die fehlerhaften Messungen 40](#_Toc44523671)

[Fazit 42](#_Toc44523672)

[Literaturverzeichnis 43](#_Toc44523673)

# Abbildungsverzeichnis

[Abbildung 1: Dartscheibe (https://www.dartshop.de/out/pictures/generated/product/1/310\_310\_100/sol\_em79453\_pic1\_1.jpg) 5](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523674)

[Abbildung 2: Dartscheibe mit eingefärbten Bereichen 5](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523675)

[Abbildung 3: Trilateration in einer Ebene (CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=137860) 7](#_Toc44523676)

[Abbildung 4: Skizzierung der Mikrofone und Dartpfeil in GeoGebra 8](#_Toc44523677)

[Abbildung 5: Dartscheibe mit Beschriftung der Ringe (CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=526241) 10](#_Toc44523678)

[Abbildung 6: Beschrifteter Raspberry Pi 3B+ 13](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523679)

[Abbildung 7: oben: serielle Schnittstelle, unten: parallele Schnittstelle (https://de.ccm.net/contents/274-serieller-anschluss-und-parallelanschluss) 17](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523680)

[Abbildung 8: Arduino Schnittstellenbeispie (https://tutorials-raspberrypi.de/arduino-raspberry-pi-miteinander-kommunizieren-lassen/) 18](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523681)

[Abbildung 9: Python Schnittstellenbeispiel (https://tutorials-raspberrypi.de/arduino-raspberry-pi-miteinander-kommunizieren-lassen/) 19](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523682)

[Abbildung 10: Klasse ArduinoInterface in Python 20](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523683)

[Abbildung 11: Klasse Location, Funktion: findQuadrant 20](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523684)

[Abbildung 12: Leg Anfang Double-In Kontrolle 24](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523685)

[Abbildung 13: Zuweisung des letzten Wurfmultiplikators 24](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523686)

[Abbildung 14: Leg Beenden Double-In Kontrolle 25](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523687)

[Abbildung 15: Wurfkontrolle ohne Sonderregeln 25](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523688)

[Abbildung 16: Dart ID ändern 25](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523689)

[Abbildung 17: Einteilung der Dartscheibe in vier Quadranten 28](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523690)

[Abbildung 18: Dartpfeil Feldkontrolle Beispiel: 20 28](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523691)

[Abbildung 19: Dartpfeil Bull, Bullseye und 0 Punkte Kontrolle 28](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523692)

[Abbildung 20: Skizzierung der Dartscheibe 32](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523693)

[Abbildung 21: Spieloptionen der finalen GUI 32](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523694)

[Abbildung 22: Exemplarischer Programmablaufplan von findStartTime() 35](#_Toc44523695)

[Abbildung 23: Schematische Verkabelung des Arduinos 36](#_Toc44523696)

[Abbildung 24: Logo qnf GmbH (https://image.jimcdn.com/app/cms/image/transf/dimension=218x10000:format=jpg/path/s597f50f05eb4c73f/image/i43ff55bbcd251b31/version/1454243925/image.jpg) 37](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523697)

[Abbildung 25: Verwendete Mikrofone (https://asset.conrad.com/media10/isa/160267/c1/-/de/1485297\_LB\_00\_FB/iduino-1485297-mikrofon-schallsensor.jpg?) 37](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523698)

[Abbildung 26: Logo Empire Dart (https://www.empire-dart.de/?gclid=Cj0KCQjwoub3BRC6ARIsABGhnyaGZDRS2rP3GHney740kgq-4-Dpm2cNR3gca3gV\_qFdNAXWZEjrSXEaAoNFEALw\_wcB) 37](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523699)

[Abbildung 27: Logo Gewerbliche Schulen Waldshut (https://www.gs-wt.de/files/theme/images/logo.png) 37](https://schuelergswtde-my.sharepoint.com/personal/laurin_farner_schueler_gs-wt_de/Documents/Seminarkurs/Ausarbeitung/Seminarkurs%20Dokumentation%20Ortung%20eines%20Dartpfeils%20mit%20akustischer%20Triangulation.docx#_Toc44523700)

[Abbildung 28: Aus 3-d Modell generierte technische Zeichnung der Mikrofonhalter 38](#_Toc44523701)

[Abbildung 29: Aufbau und Test der Dartscheibe 39](#_Toc44523702)

# Vorwort - Laurin

Dart ist in Deutschland hauptsächlich als Kneipensport bekannt. Während die meisten Leute die groben Spielzüge sicherlich ohne Probleme nennen können, gibt es auch vieles, was Dart-fernen Menschen nicht bewusst ist. Zum Beispiel, dass nicht die Mitte die meisten Punkte gibt, wie man intuitiv denken würde, sondern ein unscheinbares rotes Feld oberhalb der Mitte. Grund dafür ist, dass bestimmte Felder ihre zugehörige Zahl verdreifachen. Und da die Größte dieser Zahlen die 20 ist, kann man mit diesem Feld 60 statt der 50 Punkte in der Mitte erzielen. Auch, dass Leute immer noch mit den so genannten Steeldarts spielen, welche auf eine korkähnliche Scheibe geworfen werden, ist vielen vermutlich gar nicht bewusst. Dabei ist die Frage nach einer klassischen oder einer elektronischen Dartscheibe für viele aktive Dart Spieler elementar: Steeldarts können im Vergleich zu E-Darts deutlich schwerer sein, was einen ruhigeren Flug ermöglicht, sie haben für viele ein deutlich angenehmeres Aufprallgeräusch als E-Darts und die Spitzen können nicht so leicht abbrechen und in der Dartscheibe stecken bleiben. Zu guter Letzt ist Steeldarts bei professionellen Turnieren deutlich beliebter.

Allerdings haben Steeldarts einen entscheidenden Nachteil: Im Gegensatz zu E-Darts muss man alle Punkte selbst mitzählen.

Deshalb sind wir als bekennende Steeldartspieler auf die Idee gekommen, ein günstiges System zu entwickeln, welches selbständig erkennt, ob und wo auf einer Steeldartscheibe ein Pfeil gelandet ist, ohne dass wir selbst zählen müssen aber auch ohne die Vorzüge des Steeldarts einzuschränken.

# Verschiedene Ideen zur Umsetzung - Laurin

Unsere erste Idee war, die Darterkennung mit Kameras zu realisieren. Hierzu wird jeweils eine Kamera auf der vertikalen und eine auf der horizontalen Achse neben der Dartscheibe angebracht. Jede Kamera überwacht die Position des Pfeils entlang einer Achse. Die Positionen auf den beiden Achsen des Dartpfeils können zu einer Koordinate des Dartpfeils in einem Kartesischen Koordinatensystem zusammengesetzt werden. Die Positionserkennung des Pfeils im Bild wird mittels der Programmierbibliothek „OpenCV“ programmiert. Vorteil hierbei ist, dass die Genauigkeit der Messung theoretisch nur durch die Auflösung der Kamera beschränkt wird. Allerdings spielen hier konstante Verfälschungen der Ergebnisse, wie Verzeichnung[[1]](#footnote-1), welche herausgerechnet werden müssen, eine Rolle. Eine weitere Limitierung stellen variierende Lichtverhältnisse dar.

Eine andere Idee war, Druckplatten hinter der Dartscheibe zu montieren. Es ist aber fraglich, ob diese ein sicheres und präzises Auslösen garantieren können. Zudem wäre diese Lösung mit vielen Kosten und Aufwand, aber wenig Programmieranspruch verbunden.

Auch optische Abstandssensoren waren in der Überlegung. Allerdings können diese entweder nur Abstände auf einem bestimmten Punkt messen, oder kosten mehrere Hundert Euro.

Letztendlich sind wir dann auf den Ansatz aufmerksam geworden, mit Hilfe von Mikrofonen über Laufzeitendifferenz des Aufprallgeräuschs des Dartpfeils dessen Position zu orten, welche in ähnlicher Form bereits von Studenten des MITs in Angriff genommen wurde[[2]](#footnote-2). Dazu ist zu sagen, dass fast alle oben genannten Methoden bereits zu vorherigen Zeitpunkten anderweitig realisiert oder versucht wurden. Der große Vorteil dieser Methode im Vergleich zu einer Kamera oder optischen Abstandssensoren ist, dass eine ganze Partie Dart mit 3 Würfen pro Runde gespielt werden kann, ohne dass nach jedem Wurf der Pfeil wieder aus der Scheibe gezogen werden muss. Bei einer Kamera besteht bei mehreren Pfeilen auf der Scheibe die Gefahr, dass ein Pfeil einen anderen überdeckt und somit das Messergebnis verfälscht.

# Darts - Can

Darts ist ein Geschicklichkeitsspiel und ein Präzisionssport, bei dem Dartpfeile auf eine Dartscheibe geworfen werden, um so Punkte zu erzielen.

## Dartscheibe

Die Dartscheibe ist in 20 Segmente eingeteilt. Die einzelnen Segmente verteilen sich um den Mittelpunkt im Uhrzeigersinn (wie folgt: 20-1-18-4-13-6-10-15-2-17-3-19-7-16-8-11-14-9-12-5).

Abbildung 1: Dartscheibe (https://www.dartshop.de/out/pictures/generated/product/1/310\_310\_100/sol\_em79453\_pic1\_1.jpg)

Die Zahlen sind mit Absicht so „unordentlich“ eingeordnet worden, sodass man weniger von Zufall sprechen kann, wenn man eine hohe Zahl trifft. Beim Dart ist die Treffsicherheit sehr wichtig und um die Treffsicherheit zu beweisen ist diese Zahlenanordnung recht nützlich.

Die Mitte der Dartscheibe ist in zwei Felder eingeteilt. Es gibt einmal das Bullseye (auch Double-Bull oder inner-Bull genannt), der den roten Kreis in der Mitte darstellt und 50 Punkte zählt. Der grüne Kreis um den Bullseye heißt Bull (auch Outer-Bull oder Single-Bull genannt), der 25 Punkte zählt. Der im zweiten Bild blau gefärbte Bereich, ist der Bereich, in dem die Segment-Ziffer genau einmal zählt. Also kann man im blauen Bereich jede Zahl von 1 bis 20 erreichen. Der gelb gefärbte Bereich ist der Triple Bereich. Wenn ein Dartpfeil diesen Bereich trifft, zählt das Segment, welches man trifft, dreifach. Also wenn man den Triple Bereich im Segment der Ziffer 3 trifft, hat man 9 Punkte erreicht. Der äußere pink gefärbte Bereich ist der Double Bereich. Wenn man diesen Bereich mit dem Dartpfeil trifft, wird der getroffene Punkt verdoppelt. Wenn man jedoch den schwarzen Bereich außerhalb des Farbbereiches trifft, bekommt man keine Punkte.

Abbildung 2: Dartscheibe mit eingefärbten Bereichen

## Abmessungen der Dartscheibe

Die Spielfläche der Dartscheibe ist 34cm breit. Der Bullseye hat einen Durchmesser von 12.7mm. Der äußere Bull ist mit dem Bullseye 31.8mm breit. Das erste Single Feld, welches sich zwischen dem Single Bull und dem Triple Bereich befindet, ist 8,31cm breit. Das zweite Single Feld ist 5,5cm breit. Jedes Segment hat einen Winkel von 18°.

Diese Maße sind sehr wichtig für unser Projekt, da wir mit den Daten der Mikrofone die Winkel, und die Entfernung des Dartpfeils vom Mittelpunkt aus berechnen und so zuordnen welche Punkte der Spieler gerade erzielt hat.

## Dartvarianten

Leg anfangen / beenden

Ein Leg ist, wenn man von seinem Anfangsscore (meist 501) auf exakt 0 kommt. Um ein Leg anzufangen gibt es auch verschiedene Sonderregeln, wie zum Beispiel Straight-In, Double-In und Triple-In. Beim Straight-In ist es egal mit welchem Dartfeld man ins Spiel einsteigt. Beim Double-In muss der erste Dart auf einem Double-Feld landen. Wenn nicht werden die Würfe nicht gezählt, bis man mit einem Double Wurf einsteigt. So ist es auch mit dem Triple-In, aber hier muss man zum Einstieg ein Triple-Feld treffen. Bei Straight-Out, Double-Out und Master-Out sind es eigentlich dieselben Prinzipien wie bei den vorherigen genannten Einstiegsmöglichkeiten. Nur geht es hier nicht um den ersten, sondern den letzten Wurf. Bei Straight-Out darf man mit einem beliebigen Dartfeld das Spiel beenden, bei Double-Out muss man ein Double-Feld treffen mit seinem letzten Dart und bei Master-Out ein Triple-Feld. Man muss bei Straight-Out, Double-Out und Triple-Out jedoch beachten, dass man genau auf 0 kommt. Wenn man weniger als Null hat, werden die letzten Würfe zurückgesetzt.

Spielmodus 501 / 301 / X01

Diese Variante ist die häufigste gespielte Variante im Dart. Hier hat jeder Spieler 501 (oder 301, 401, …) Punkte und das Ziel ist es auf exakt 0 zu kommen. Jeder hat 3 Darts und man wirft diese. Die Punkte, die man erziehlt hat mit den 3 Darts werden zusammenaddiert und diese Summe wird dann von der eigenen Punktzahl abgezogen. Danach startet der nächste Spieler seinen Wurf. Beim klassischen 501 kann man frühstens mit 9 Darts das Spiel beenden.

Beispiel: Wurf 1: Triple 20 = 60

Wurf 2: Triple 20 = 60

Wurf 3: Triple 20 = 60

Wurf 4: Triple 20 = 60

Wurf 5: Triple 20 = 60

Wurf 6: Triple 20 = 60

Wurf 7: Triple 20 = 60

Wurf 8: Triple 19 = 57

Wurf 9: Double 12 = 24

**Summe = 501**

Neben dem X01-Spielmodus gibt es noch weitere Varianten, Dart zu spielen wie zum Beispiel „Around the Clock“ oder „Fuchsjagd“. Aber diese Modi werden nicht bei Turnieren gespielten und werden eher zu Hause oder in Kneipen gespielt.

# Die Berechnung der Pfeilkoordinaten - Laurin

Das Wort „Triangulation“ bezieht sich in der Mathematik auf verschiedene Verfahren, welche als Gemeinsamkeit alle mit Dreiecken arbeiten.

Eine Form von Triangulation ist das Erzeugen eines Dreiecksnetzes aus einer Punktmenge. Dies dient zum Beispiel der Visualisierung von Flächen und Körpern im Raum, was besonders bei 3-dimensionalen Computerspielen zum Einsatz kommt.[[3]](#footnote-3)

In der Geodäsie bezeichnet die Triangulation wiederum die Aufteilung einer Landfläche in Dreiecke, um diese zu vermessen[[4]](#footnote-4).

Die weitläufigste als „Triangulation“ bekannte Anwendung ist allerdings das Ermitteln einer Strecke über die Winkel eines konstruierten Dreiecks. Hierzu muss lediglich eine Seitenlänge des Dreiecks bekannt sein. Die anderen beiden Seiten werden mithilfe trigonometrischer Formel über die Winkel errechnet[[5]](#footnote-5).

Obwohl unsere Ortung des Dartpfeils der letzteren Triangulationsdefinition ähnelt, weist sie mehr Ähnlichkeit zu der unter anderem bei GPS verwendeten Trilateration auf. Im Gegensatz zur Triangulation wird hierbei nicht mit Winkeln, sondern ausschließlich mit Hilfe von Strecken gearbeitet.

Das Grundprinzip hierbei ist, dass die zu ortende Quelle ein Signal an drei Punkte mit bekannten Koordinaten sendet. Die Zeit zwischen Abschicken und Ankunft des Signals ergibt, unter Einberechnung der Geschwindigkeit des Signals, den Radius der Entfernung zum jeweiligen Punkt an. Die Schnittpunkte dieser Kreise ergeben so die Position der Quelle[[6]](#footnote-6).

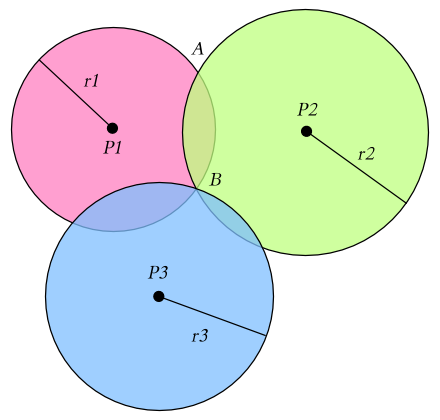


Abbildung 3: Trilateration in einer Ebene (CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=137860)

Das Signal, welches die Quelle, also konkret der Dartpfeil aussendet, ist in unserem Fall das Aufprallgeräusch des Pfeils auf der Dartscheibe. Mittels der Schallgeschwindigkeit lassen sich so die Abstände zwischen den Mikrofonen und dem Dartpfeil ermitteln.

Der Unterschied zu der Klassischen Trilateration ist bei uns, dass sich der genaue Zeitpunkt des Aufpralls nicht messen lässt, da das Mikrofon, welches dem Dart am nächsten ist, den Aufprall erst erkennt, nachdem die Strecke zwischen ihnen überwunden ist. Somit bleibt uns am Anfang der Rechnung auch immer eine unbekannte Strecke (d0) zwischen dem Pfeil und dem nächsten Mikrofon.

Die drei Mikrofone M1(0|0), M2(x2|0) und M3(0| y3) sind wie folgt in gleicher Entfernung um die Dartscheibe angeordnet:

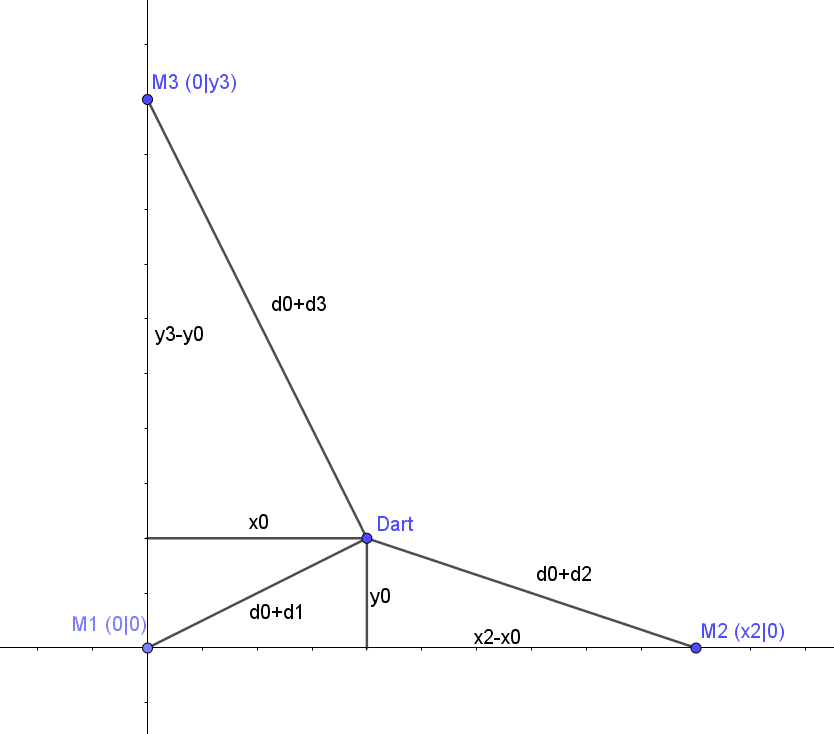


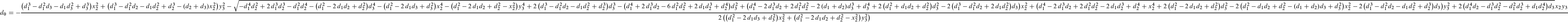
Abbildung 4: Skizzierung der Mikrofone und Dartpfeil in GeoGebra

Um die zu lösenden Gleichungen möglichst simpel zu halten, ist ein Mikrofon im Nullpunkt installiert und die anderen jeweils auf einer der Koordinatenachsen. Im Vergleich zur Modellierung mit Nullpunkt im Mittelpunkt der Dartscheibe werden so vier Konstanten weniger benötigt, was die Gleichungen deutlich vereinfacht.

Mit folgendem Gleichungssystem lassen sich die unbekannte Strecke d0 und die Pfeilkoordinaten x0 und y0 ausrechnen:

Jede der drei Gleichungen beruht auf dem Satz des Pythagoras und steht für konstruiertes Dreieck zwischen einem der Mikrofone und dem Dartpfeil mit der Gesamtlänge zwischen Pfeil und jeweiligen Mikrofon als Hypotenuse. Dieses Gleichungssystem lässt sich nun nach den drei Unbekannten d0, x0 und y0 umstellen, um diese mittels der konstanten Koordinaten der Mikrofone und der gemessenen Strecken zu berechnen.

Obwohl die einzelnen Gleichungen für sich auf den ersten Blick sehr simpel wirken, gestaltet sich das Umstellen nach d0 ohne technische Hilfe als sehr komplex und unübersichtlich. Deshalb haben wir uns für diese Aufgabe die Software „SageMath“ zu Hilfe genommen, um das Gleichungssystem nach d0 aufzulösen:

…

d0 hat noch eine zweite Lösung, welche allerdings nur für den vierten Quadranten des Koordinatensystems gilt und somit für uns uniteressant ist, da die Dartscheibe vollständig im ersten Quadranten liegt.

Mit der Lösung von d0 lassen sich die anderen unbekannten deutlich unkomplizierter von Hand lösen:

Diese Gleichungen wurden dann in Python Syntax übersetzt und mittels einer Modellierung in GeoGebra auf ihre Richtigkeit überprüft.

# Berechnung der Punktzahl aus der Pfeilkoordinate - Laurin

Eine Dartscheibe ist aus insgesamt 82 Feldern aufgebaut, von denen jedes eine ihm fest zugewiesene Punktzahl liefert. Die weißen bzw. schwarzen Felder liefern die jeweils außen angegebene Punktzahl. Der äußere und innere Ring multipliziert diese mal zwei bzw. drei und das innere Bullseye liefert mit 50 genau doppelt so viele Punkte wie das äußere Bullseye.

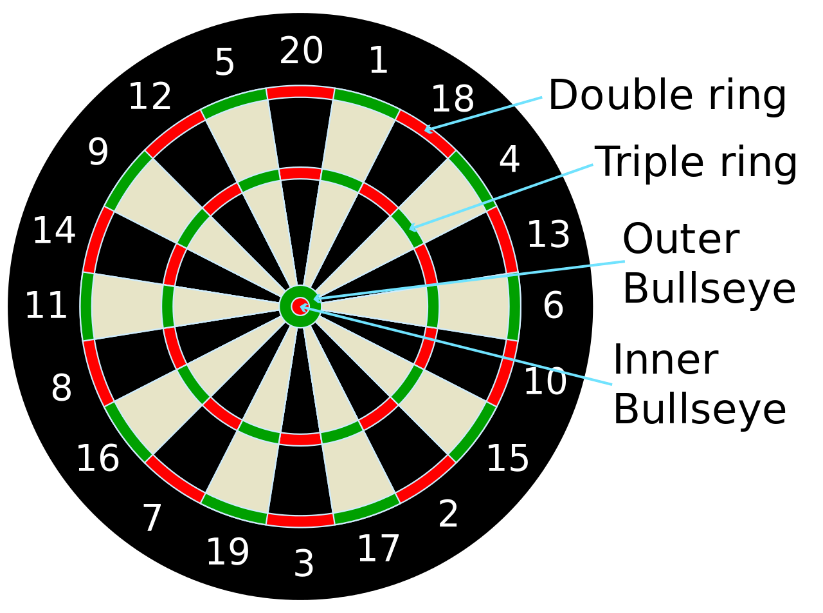


Abbildung 5: Dartscheibe mit Beschriftung der Ringe (CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=526241)

Da die Felder einer Dartscheibe rund angeordnet sind, lassen sich nicht einfach bestimmte Bereiche im Koordinatensystem festlegen, welche auf die entsprechende Punktzahl schließen lassen. Stattdessen kann man über den Winkel, mit dem der Pfeil zur Mittellinie gelandet ist, die Zahl ermitteln und mit der Strecke zwischen Mittelpunkt und Pfeil den Multiplikator errechnen.

Um dies zu verwirklichen stellen wir die Strecke zwischen Mittelpunkt und Dart als Vektor der Form dar.

Mit dem Betrag von lässt sich nun der Abstand des Darts zur Mitte berechnen und damit herausfinden, in welchem Ring der Pfeil gelandet ist.

Den Winkel des Vektors in Grad haben wir mittels Tangens eines konstruierten Dreiecks berechnet, in welchem der Vektor die Hypotenuse bildet:

Da der Tangens mit dieser Methode nur Winkel < 90° berechnet, muss noch geprüft werden, in welchem gedachten „Quadranten“, vom Mittelpunkt der Dartscheibe aus, der Dart gelandet ist, um das Ergebnis eindeutig zuordnen zu können.

Eine andere Möglichkeit wäre gewesen, mit folgender Formel den Winkel zwischen und einem anderen Vector zu berechnen:

ist in diesem Fall ein Einheitsvektor, welcher kollinear zur x-Achse ist, wodurch sich die Formel für entsprechend vereinfachen lässt:

Der Vorteil hierbei ist, dass ein Winkel von bis zu 180° berechnet werden kann und nur noch geprüft werden muss, ob der Dart über- oder unterhalb des Mittelpunktes der Scheibe gelandet ist.

Diese Möglichkeit erschloss sich uns allerdings erst mit der Einführung der Vektorgeometrie in Mathematik, nachdem wir bereits die zuerst genannte Berechnung im Programm implementiert hatten.

Da sich nach der Berechnung des Abstandes schon klar sagen lässt, ob die Dartscheibe verfehlt wurde oder das Bullseye getroffen wurde, ist die Berechnung des Winkels in vielen Fällen gar nicht mehr nötig.

# Raspberry Pi 3B+ - Can

## Was ist ein Raspberry Pi?

Der Raspberry Pi ist ein sogenannter Einplatinencomputer (*single-board-computer*). Das bedeutet, dass sich alle wichtigen Komponenten, wie zum Beispiel der Prozessor oder der ROM (*read-only-memory*) auf einer einzigen Platine befinden. Das einzige was sich nicht auf der Platine befindet ist in der Regel das Netzteil.

## Was kann man mit einem Raspberry Pi machen?

Mit einem Rapsberry Pi kann man generell sehr viel machen. Denn der Raspberry Pi ist ein vollwertiges System, was man für eine Vielzahl von Anwendungen einsetzen kann. Theoretisch gesehen, kann man den Raspberry Pi auch als Desktop-Rechner benutzen, er ist zwar nicht zu vergleichen mit einem selbstgebauten Computer aus teuren Komponenten, aber er ist auch als Desktop PC nutzbar. Die meisten Leute benutzen den Raspberry jedoch für kleine, bis zu großen Projekten. Ein sehr beliebtes Projekt ist es, den Raspberry Pi als Heim-Server zu nutzen. Also ein Server auf den man mit der richtigen Verbindung immer zugreifen kann. So kann man sich selbst einfach eine eigene Cloud gebastelt. Aber natürlich gibt es noch viel mehr Möglichkeiten den Raspberry Pi zu nutzen.

Wir benutzen hier den Raspberry Pi 3B+ um die Mikrofon-Messwerte des Arduinos zu verarbeiten und um das Dartspiel auch regelrecht spielen zu können. Also berechnet unser Raspberry Pi die Positionen der Dartpfeile mit den Daten des Arduino, er berechnet die Punktzahl und kontrolliert den Spielablauf.

## Aufbau des Raspberry Pi 3B+

Abbildung 6: Beschrifteter Raspberry Pi 3B+

**GPIO-Pins:** GPIO-Pins (general-purpose input/output), sind Allzweckein- und ausgabe Pins. Die Pins können zum Beispiel zur Ansteuerung von LEDs, oder zur Tonausgabe verwendet werden.

**SoC:** Auf dem SoC (System-on-Chip/System-on-a-Chip) befindet sich der Großteil der Funktionen des Raspberry Pi.

**USB-Port:** USB-Ports sind serielle Bussysteme, die zur Verbindung des Raspberry Pi‘s und einem externen Gerät dienen können(siehe Kapitel Schnittstelle zwischen Arduino und Raspberry Pi).

**Bluetooth/WLAN Chip:** Ein Chip, der dem Raspberry Pi WLAN und Bluetooth Funktionen ermöglicht.

**Micro-SD-Card-Slot:** Slot für Micro SD Karten

**DSI:** Ein DSI (Display Serial Interface) ist ein serieller Bus und eine Kommunikationsmöglichkeit zwischen der Quelle der Bilddaten und dem Ziel dieser Bilddaten. Der DSI ist im Allgemeinen auf LCS Displays ausgerichtet.

**HDMI:** Der HDMI-Port (High Definition Multimedie Interface) ist ein Interface für digitale Bild- und Ton-Übertragung.

**CSI:** Das CSI (Camera Serial Interface) ermöglicht dem Raspberry Pi die Verbindung/Nutzung einer Camera

**Micro-USB:** Der Micro-USB hat eigentlich ähnliche Funktionen wie ein normaler USB-Port. Aber hier dient der Micro-USB Port zur Stromversorgung

**AV-Out:** Der AV-Out (Audio – Video - Output), ermöglicht das anschließen von Köpfhörern oder Lautsprechern mit Klinkensteckern**.**

**Ethernet-Port:** Ethernet ist eine Datenübertragungstechnik, die es ermöglicht Daten zwischen verschiedenen Geräten innerhalb eines Netzwerkes zu übertragen.

# Arduino - Laurin

Die Firma Arduino stellt sowohl Open Source Microcontroller Bords als auch Software zur Verfügung, mit der die Boards programmiert werden können.

Die verschiedenen Boards verwenden eine Vielzahl verschiedener Mikroprozessoren und zeichnen sich durch unterschiedliche Größen, Prozessortakte und Anzahl der so genannten I/O Pins aus. Diese Pins können genutzt werden, um externe Sensoren und Module anzuschließen. Es wird zwischen analogen und digitalen Pins unterschieden.

Arduino Boards werden mit der C-ähnlichen Arduino Programmiersprache programmiert. Ein Programm für den Arduino wird Sketch genannt. Es besteht immer mindestens aus dem setup() Unterprogramm, in welchem Anfangsparameter wie die Modi der einzelnen I/O Pins festgelegt werden und dem loop() Unterprogramm, in welchem der auszuführende Code steht. Zusätzlich kann man den Sketch um eigene Unterprogramme erweitern.

Die offizielle Entwicklungsumgebung zum Programmieren von Arduino Sketches ist die Arduino IDE, welche allerdings nur grundlegende Funktionen zur Verfügung stellt. Für erweiterte Funktionen wie Code Vervollständigung oder Debugging war man bislang auf Software Dritter angewiesen. Um dies zu ändern ist seit Oktober 2019 eine Vorabversion der Arduino Pro IDE verfügbar, welche unter anderem die oben genannten Funktionen sowie Git Implementation beherrscht[[7]](#footnote-7).

Da Arduino Open Source ist, ist eine Vielzahl günstiger, Bords erhältlich, welche baugleich zu den originalen Arduino Boards sind. Das Board, welches wir von der Schule zur Verfügung gestellt bekommen haben ist ebenfalls ein Board eines Drittherstellers. Es ist baugleich mit dem Arduino Mega. Dieser besitzt 54 digitale I/O Pins und 16 analoge Pins[[8]](#footnote-8). Da wir aber insgesamt maximal 4 Pins benötigen, hätte uns auch ein Arduino Nano mit 22 digitalen und 8 analogen Pins[[9]](#footnote-9) gereicht, welcher aufgrund seiner geringen Baugröße wesentlich einfacher an der Dartscheibe anzubringen wäre.

# Schnittstelle zwischen dem Arduino und dem Raspberry Pi - Can

Was ist eine Schnittstelle und wofür werden diese genutzt?

Eine Schnittstelle, oder im englischen auch Interface genannt, ist eine Verbindungsstelle zwischen zwei oder mehreren Prozessen/Komponenten. Die Schnittstelle ist der Teil, der der Kommunikation zwischen den Systemen dient. Schnittstellen werden genutzt, sodass Daten von einem System (Bsp.: Arduino) zu einem anderen System (Bsp.: Raspberry Pi) übertragen werden können, damit dort dann die gerade übertragenen Daten verarbeitet werden können.

Welche Arten von Schnittstellen gibt es?

Es gibt verschiedene Arten, eine Schnittstelle aufzubauen.

**Maschinenschnittstelle**

Maschinenschnittstellen sind die Schnittstellen, an dem ein Gerät mit einem weiteren Gerät interagiert. Die Maschinenschnittstellen sind wichtig für die Nutzung von Hardware mit anderer Hardware.

Beispiele: Tastatur, Maus, Netzwerkanschluss, …

**Softwareschnittstelle**

Softwareschnittstelle sind logische Verbindungspunkte in einer Software. Diese Verbindungspunkte ermöglichen und steuern den Austausch von Daten und Befehlen zwischen verschiedenen Komponenten und Prozessen.

Man kann jedoch auch Softwareschnittstellen nochmals in sich unterscheiden in Schnittstellen für Kommunikation oder Schnittstellen als funktionale Einheiten. Kommunikationsschnittstellen sind immer passiv und enthalten nur die Information, die zwischen zwei oder mehreren Prozessen/Systemen ausgetauscht werden. Schnittstellen als funktionale Einheiten führen eine bestimmte Funktion aus, um die primären Systemteile zu unterstützen oder zu synchronisieren.

Programmierschnittstellen, kurz: API (engl. Application programming interface), sind eben solche Schnittstellen. Programmierschnittstellen kann man aber auch in veschiedene Typen einteilen.

Funktionsorientierte API: Diese Art von API kennt nur Funktionen mit oder ohne Rückgabewert als Kommunikationsmittel.

Dateiorientierte API: Werden über die Standard Dateisystemaufrufe open (Datei öffnen), close (Datei schließen), write (Daten an ein Objekt senden) und read (Daten empfangen) angesprochen.

Objektorientierte API: Hier werden Schnittstellenzeiger verwendet, um fremde Unterprogramme aufzurufen. (Bsp.: IOTools.readInteger()

//IOTools ist hier der Schnittstellenzeiger

**Hardwareschnittstellen**

Hardwareschnittstellen sind Interfaces zwischen physischen Systemen in der Elektrotechnik und Elektronik. Ist ein Gerät mit einer Schnittstelle ausgestattet, dann spricht man von einer Konnektivität.

Hardwareschnittstellen sind in der Computertechnik weit verbreitet. Beispiele für Hardwareschnittstellen in Computern wären:

* Peripheral Component Interconnect (PCI-Bus)
  + Ein Bus-Standard für Peripherieverbindungen mit dem Chipsatz eines Prozessors.
* Universal Serial Bus (USB)
  + Ein serielles Bussystem um einen Computer, mit externen Geräten zu Verbinden (Maus, Tastatur, externe Festplatte, etc.)
* Accelerated Graphics Port (AGP)
  + Eine Anschlussnorm auf dem Mainboard, mit dem eine direkte Verbindung der Grafikkarte mit dem Chipssatz ermöglicht wird.

Bei Hardwareschnittstellen unterscheidet man jedoch nochmnal mit zwei Arten von Schnittstellen. Serielle und Parallele Schnittstellen.

Serielle Schnittstellen

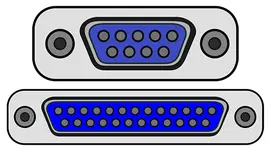
Bei einer seriellen Schnittstelle werden einzelne Bits nacheinander von einem Gerät zum anderen übertragen. Serielle Schnittstellen brauchen weniger Stromkreise und dadurch hat man auch einen geringeren Verkabelungsaufwand.

Abbildung 7: oben: serielle Schnittstelle, unten: parallele Schnittstelle (https://de.ccm.net/contents/274-serieller-anschluss-und-parallelanschluss)

Parallele Schnittstelle

ei einer parallelen Schnittstelle werden mehrere Bits gleichzeitig auf mehrere Stromkreise übertragen. Parallele Schnittstellen bieten eine höhere Datenübertragungsgeschwindigkeit und sind vorteilhaft, für kurze Distanzen.

**Benutzerschnittstellen**

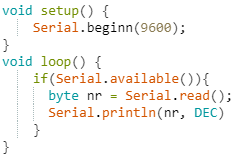
Benutzerschnittstellen sind die Schnittstellen, die ermöglichen, dass ein Mensch (der Benutzer) mit einem Gerät interagiert. Diese sind meist irgendwelche Bedienelemente, wie ein Lichtschalter, oder ein Tastenfeld auf einem Bankautomaten, aber auch die Grafische Benutzeroberfläche (kurz: GUI, vom englischen graphical user interface) eines Computers ist eine Benutzerschnittstelle.

**Wie lässt man einen Arduino und ein Raspberry Pi miteinander kommunizieren?**

**Generelles Beispiel**

Materialliste

* Raspberry Pi (hier: Raspberry Pi 3B+)
* Arduino (hier: Arduino Mega)
* USB – Kabel (Hier: USB Typ B)
* Laptop oder Computer, um auf den Arduino das Programm laden zu können

Vorgehensweise

**Arduino**

Am Arduino muss man eigentlich nicht viel machen, ausser den Code drauf zu laden. Wie der Code aussieht entscheidet jeder eigentlich für sich und da gibt’s auch keine spezifischen Angaben.

In unserem Beispiel, empfängt der Arduino eine Zeichenfolge, die er mit den ASCII Werten der einzelnen Zeichen ausgibt.

Abbildung 8: Arduino Schnittstellenbeispie (https://tutorials-raspberrypi.de/arduino-raspberry-pi-miteinander-kommunizieren-lassen/)

**Raspberry Pi**

Beim Raspberry muss man ein bisschen mehr machen. Dort werden zusätzliche Bibliotheken benötigt, damit man mit der seriellen Schnittstelle des Raspberry etwas anfangen kann. Diese installiert man im command prompt mit: *sudo apt-get install minicom python-serial.* Nun ist das Ziel, den Port herauszufinden, an dem die Daten des Arduino beim Raspberry Pi eingehen. Dazu führt man folgenden Befehl im Command prompt aus: ***ls /dev/tty\****

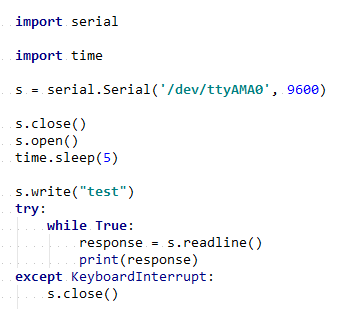
Durch diesen Befehl wird eine Liste mit den Port-Namen des Raspberry Pi’s angezeigt. Nun muss man den Arduino und den Raspberry Pi mit dem USB-Kabel verbinden und den gerade ausgeführten Befehl nochmals ausführen. Wenn man alles richtig gemacht hat, sollte nun am Ende der Liste ein weiterer Port-Name hinzugefügt worden sein. Mit diesem Port-Namen (Im Bsp.: ***/dev/ttyAMA0***) kann man dann den Arduino über die serielle Schnittstelle ansprechen.

Abbildung 9: Python Schnittstellenbeispiel (https://tutorials-raspberrypi.de/arduino-raspberry-pi-miteinander-kommunizieren-lassen/)

Nachdem man den Portnamen dann eingegeben hat, kann man schon losprogrammieren. Bei diesem Beispiel (Bild 2) sendet es sozusagen den String „test“ zum Arduino. Dann wird das was der Raspberry Pi wieder vom Arduino empfängt ausgegeben. Die Ausgabe ist dann eine Anzeige der ASCII Werte von jedem Zeichen im String „test“ (also: t = 116, e = 101, s = 115, t = 116).

# Die Serielle Schnittstelle für unseren Seminarkurs - Can

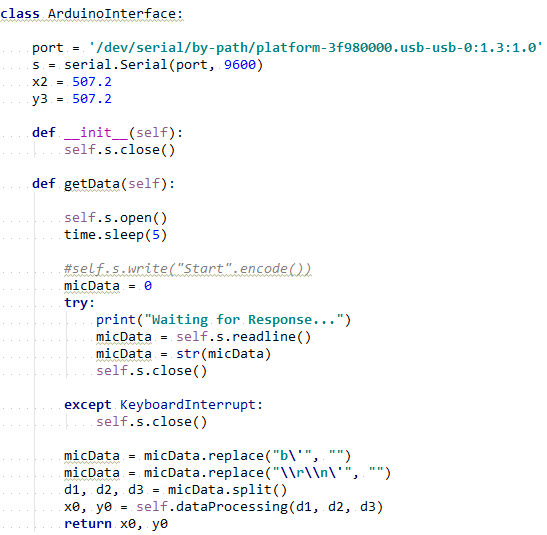
****Unsere Schnittstelle, die vom Raspberry gesteuert wird, mit der Klasse ***ArduinoInterface***, nimmt die Mikrofonmesswerte vom Arduino. Der Arduino gibt direkt alle drei Messwerte weiter. Diese werden dann zunächst zu einem String umgewandelt mit der Funktion ***str()***. Denn die Daten, die wir in dem String bekommen sehen noch ungefähr so aus: „b\‘ x y z\r\n\‘ “.

Abbildung 10: Klasse ArduinoInterface in Python

Aus diesem String benötigen wir nur die Werte x, y und z. Also Entfernen wir mit ***micData.replace()*** das „b\‘“ und das „\r\n\‘“. So bekommen wir einen String, die 3 Mikrofonwerte enthält, die wir dann mit der Funktion ***split()*** trennen, sodass wir jeden einzelnen Wert in eine eigene Variable setzen können. Diese drei Variablen werden dann in die Funktion ***dataProcessing(d1, d2, d3)*** gesteckt. Diese Funktion enthält die Gleichung, die aus den Mikrofon Messwerten die Koordinaten des Dartpfeils berechnet (siehe: Kapitel „Die Berechnung der Pfeilkoordinaten).

Wie wurde die Schnittstelle in das Spiel Programm integriert?

In der Klasse Game, in der der Spielverlauf gesteuert wird, braucht man eigentlich nur wenige Zeilen mehr Quelltext.

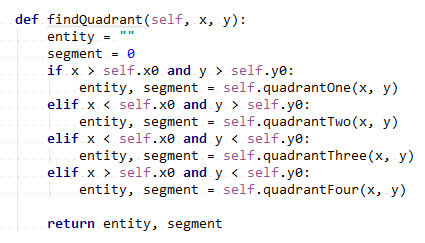
Zunächst muss man die weiteren Klassen in die Klasse Game importieren. Dies kann man machen, indem man *from SeminarKurs import \*Klasse\** eintippt. Dies sollte man dann 2-mal machen, da wir Zugriff auf die Funktionen der Klassen *ArduinoInterface* und *Location* brauchen*.* Dann muss man noch Objekte dieser Klassen kreieren. Und die einzigen Funktionen, auf die wir direkt zugreifen müssen von der Klasse Game aus, sind die Funkitonen getData( ) von der Klasse *ArduinoInterface* (siehe: Bild oben) und der Funktion findQuadrant von der Klasse Location.

Abbildung 11: Klasse Location, Funktion: findQuadrant

# Software Dokumentation Python - Can

## Game

Die Klasse Game steuert den Spielverlauf. Das Programm unterstützt die Modi 301, 501 und 601. Außerdem kann der User die Regeln nach Bedarf anpassen. Wenn keine Sonderregeln aktiviert wurden, wird einfach nur solange gespielt, bis jemand unter 0/ genau 0 Punkte hat.

**Klassenvariablen**

|  |  |  |
| --- | --- | --- |
| **Name** | **Typ** | **Erklärung** |
| players | Liste | Spielernamen/-ID‘s |
| playerPoints | Liste | Punkte von jedem Spieler |
| actPlayer | Integer | Der jetzige Spieler |
| game | Boolean | Ob das Spiel läuft oder nicht |
| dartCount | Integer | Dart-ID (von 1 bis 3) |
| startingEntity | Liste | Kontrolliste für Sonderregeln. Wenn Index (Spielernummer 1) = 0, dann zählen die Würfe nicht. Wenn es 1 ist, dann wird es gezählt. **(Wichtig für Double-In und Master-In).** |
| masterIn | Boolean | Ob mit Master-In gespielt wird oder nicht |
| doubleIn | Boolean | Ob mit Double-In gespielt wird oder nicht |
| straightOut | Boolean | Ob mit Straight-Out gespielt wird oder nicht |
| doubleOut | Boolean | Ob mit Double-Out gespielt wird oder nicht |
| tripleOut | Boolean | Ob mit Triple-Out gespielt wird oder nicht |
| gameContinue | Boolean | Ob weitergespielt wird, wenn ein Spieler gewonnen hat |
| pointsBackUp | Integer | Der Gesamtwert der Dartwürfe pro Runde |
| lastThrow | String | Die Entität vom letzten Dartwurf (Single, Double, Triple) |
| s | Liste | Single Werte der Segmente (1, 2, …, 20) |
| d | Liste | Double Werte der Segmente (2, 4, …, 40) |
| t | Liste | Triple Werte der Segmente (3, 6, …, 60) |
| scoreboard | Liste (verschachtelt) | Liste mit dem Punkteverlauf der Spieler. |
| playersThrows | Liste | Segmentwerte der Dartwürfe von den Spielern |
| playersEntity | Liste | Entitätswerte der Dartwürfe von den Spielern |
| playersWon | Liste | Liste mit Spielern, die schon fertig sind (also bei 0 waren) |
| nameSet | Boolean | Ob man in die Liste players die Spielernamen oder die Spieler-IDs eingetragen werden sollen |

**Unterprogramme der Klasse Game**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Rückgabe-Typ** | **Parameter** | **Erklärung** |
| \_\_init\_\_ | Void | Self | Konstruktor der Klasse Game |
| process | void | Self | Spielablauf |
| fillPointsLists | void | self | Füllt die Punkte Listen mit den dazugehörigen Werten |
| Createplayers | Void | self | Kreiert die Listenplätze für die Spieler |
| Choosemode | void | self | Spielmoduswahl |
| Throwdart | void | Self, n, e | Rechnung für den Dartwurf |
| Checkdart | Integer | Self, t, e | Kontrolliert die Gültigkeit des Segmentwertes und der Entität |
| Checkuserdecision | Boolean | Self, phrase | Kontrolle der User Entscheidung (Ja/Nein) |
| Checkrules | void | Self, n1, e1, n2, e2, n3, e3 | Kontrolle der Spielregeln |
| Changedartcount | void | Self, bool | Dartwurfnummer änderung |
| Changeplayer | void | self | Ändert den Spieler |
| Checkcontinuegame | void | self | Userabfrage, ob weitergespielt werden soll, nachdem jemand 0 erreicht hat |
| Askrulesettings | void | self | Userabfrage, mit welchen Regeln gespielt werden soll |
| Checkstopgame | Boolean | Self, phrase | Kontrolliert, ob das Spiel frühzeitig abgebrochen wurde |
| Showscoreboard | void | self | Scoreboard-Ausgabe |
| Fillscoreboard | Void | self | Füllt Scoreboard mit Spielerwerten |
| Pausescoreboard | void | self | Gibt eine Trennlinie aus, um das Scoreboard lesbarer zu machen |
| Checkoneplayerleft | void | self | Kontrolliert, ob nur noch ein Spieler da ist (also die Punktzahl 0 noch nicht erreicht hat) |
| welcome | void | self | Gibt die Willkommensnachricht in der Konsole aus |
| restartgame | void | self | Resettet die Klassenvariablen, und startet das Programm neu |

## Kontrolle der Spielregeln

**checkDart**

Die Funktion checkDart ist eigentlich dafür da, um die Gültigkeit des Dartwertes zu kontrollieren. Aber es schaut auch, ob die Punkte überhaupt zählen dürfen, denn bei der Spielregel Double-In/Master-In zählen die Punkte erstdann, wenn man ein Double/Triple Feld trifft. Deswegen gibt es die Liste startingEntity, in der für jeden

Spieler eine 0 oder eine 1 eingetragen wird. 0 bedeutet, dass noch kein Double/Triplefeld getroffen wurde und die Punkte nicht gezählt werden. Eine 1 bedeutet dann, dass schon ein Double-/Triplefeld getroffen wurde, und die Punkte dann auch zählen.

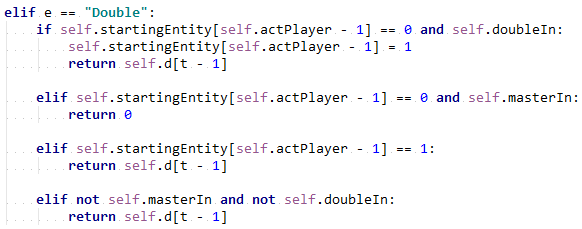


Abbildung 12: Leg Anfang Double-In Kontrolle

**checkrules**

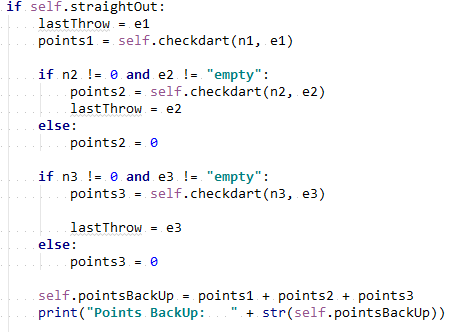
Das Unterprogramm checkRules ist eigentlich nur für die letzten Würfe zuständig. Hier wird geschaut, ob man auch genau null erziehlt (also nicht unter Null geht), oder ob man den letzten Wurf auch mit einem Double-/ Triple Wurf erziehlt hat oder nicht. Dies aber nur wenn diese Regeln auch vom user aktiviert wurden. Um die Kontrolle durchzuführen ist die Varibale „lastThrow“ sehr hilfreich. In dieser Variabel wird die Entität des letzten Pfeils gespeichert (Single, Double, Triple). Wenn dann der Spieler 0 erreicht hat, wird zunächst geschaut, ob einer der Sonderregeln aktiviert wurde (straightOut, doubleOut, TripleOut), falls ja, wird geschaut ob die letzt-geworfene-Entität auch gültig ist. Also wenn der letzte Wurf ein Triple Wurf ist, und die Regel Double-Out aktiviert wurde, dann werden die letzten Würfe nicht gezählt. In solchen Situationen ist die Variable pointsBackUp sehr hilfreich. Diese Variabel enthält die von dem Spieler in der Runde geworfenen Punktewerte, sodass Sie im Falle eines Regelverstoßes wieder zum Punktestand des Spielers zurück addiert werden können.

Abbildung 13: Zuweisung des letzten Wurfmultiplikators

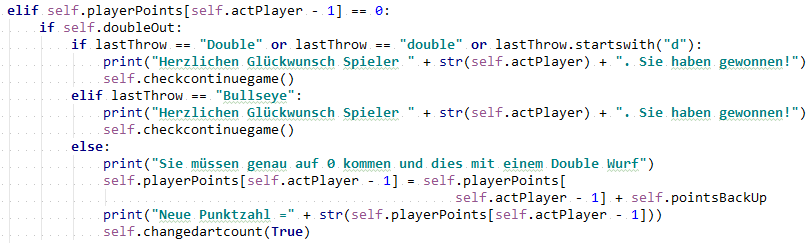


Abbildung 14: Leg Beenden Double-In Kontrolle

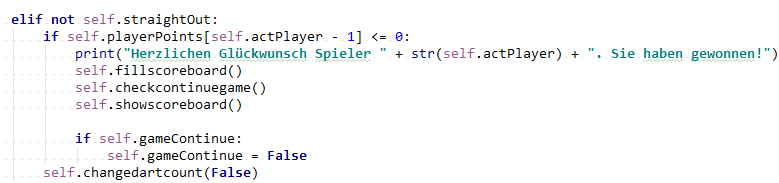
Wenn jedoch gar keine Regel (Double-In, Master-In, Straight-Out, Double-Out, Triple-Out) aktiviert wurde, dann wird nur kontrolliert, ob der Spieler 0 erreicht hat. Wenn nicht, geht es normal weiter.

Abbildung 15: Wurfkontrolle ohne Sonderregeln

**changeDartCount**

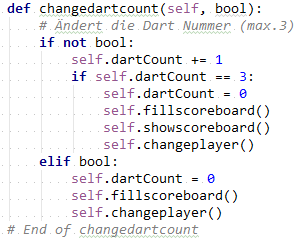
****Das Unterprogramm changeDartCount hat die Aufgabe, die Dart-ID zu ändern und gibt somit vor, wie viele Würfe jeder zur Verfügung hat. Also jedesmal wenn dieses Unterprogramm aufgerufen wird, wird die Dart Anzahl um 1, bis man bei 3 ist. Wenn man bei 3 ist, wird die ID wieder zurück gesetzt und der Spieler wird gewechselt. Aber wenn jemand zum Beispiel überwirft odere auf andere Wege die Regeln verstößt, wird ja direkt der Spieler gewechselt, was dann auch bedeutet, dass die Dart-ID’s zurückgesetzt werden müssen. Um dies zu ermöglichen, muss man beim Aufruf von changeDartCount noch einen Boolean Parameter angeben. Wenn man True eingibt, wird der DartCount direkt auf 0 gesetzt, was nur aufgerufen wird, wenn jemand die Regeln verstößt, und deswegen jemand anderes dran ist. Wenn man False als Parameter angibt, wird ganz normal um 1 erhöht, bis man bei 3 ist, wo es dann zurückgesetzt wird und der Spieler gewechselt wird.

Abbildung 16: Dart ID ändern

## ArduinoInterface

Die Klasse ArduinoInterface ist für das Interface zwischen Arduino und Raspberry Pi zuständig. Durch diese Klasse, werden die Messwerte vom Arduino abgerufen und verarbeitet.

**Klassenvariablen**

|  |  |  |
| --- | --- | --- |
| **Name** | **Typ** | **Erklärung** |
| port | String | Port in dem der Arduino angeschlossen ist |
| S | serial | Objekt der Klasse Serial |
| X2 | float | x-Koordinate des Mikrofons auf der x-Achse |
| Y3 | float | y-Koordinate des Mikrofons auf der y-Achse |

**Unterprogramme**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Rückgabe-Typ** | **Parameter** | **Erklärung** |
| \_\_init\_\_ | void | self | Python Konstruktor der Klasse ArduinoInterface |
| getData | float | self | Ruft Messwerte vom Arduino ab und ruft dataProcessing mit diesen Daten auf |
| dataProcessing | float | Self, d1, d2, d3 | Gleichung, die die Messwerte der Mikrofone zu den Koordinaten des Dartpfeils umrechnet |

**Port Problem**

Wie man den Port herausfindet, um diese in die Schnittstelle hineinzuprogrammieren, kann man auf Seite.(INTERFACE PAGE). Aber während dem Prozess ist und aufgefallen, dass der Port sich ändert, nachdem wir den Arduino aus dem Raspberry ausgesteckt haben, und wieder in denselben USB-Slot eingesteckt haben. Da wir nicht jedesmal die Port-Liste aufrufen möchten und so schauen welcher Port es diesmal ist, mussten wir uns eine weitere Lösung ausdenken. Nach langer Recherche haben wir herausgefunden, dass man auch nach zwei weiteren Möglichkeiten, den Port angeben kann, um die Schnittstelle zu ermöglichen. Entweder by-ID oder by-path. Bei der by-id Methode kann man das Gerät mit der Seriennummer des USB2serial chips beim Arduino mit**: *ls -l /dev/serial/by-id*** finden.Aber uns wurde aus unbekannten Gründen keine ID in der Konsole angezeigt. Also haben wir die by-path Methode verwendet. Bei der by-path Methode, rufen wir den Port auf, in dem der Arduino eingesteckt wird. Die Liste die, die Port-Paths anzeigt, wird mit ***ls -l /dev/serial/by-path*** angezeigt. Dann muss man nur den richtigen USB-Port auswählen und diesen Port-Namen bei s = serial.Serial(‚***portname***‘, 9600) eintragen.

Man muss jedoch darauf achten, dass man dann das Gerät (also den Arduino) immer in denselben Port steckt (hier: USB Port oben rechts am Raspberry Pi).

## Location

Die Klasse Location berechnet mit den Koordinaten des Dartpfeils den Standort auf der Dartscheibe.

**Klassenvariablen**

|  |  |  |
| --- | --- | --- |
| Name | Typ | Erklärung |
| x0 | Integer | x-Koordinate vom Bullseye |
| y0 | Integer | y-Koordinate com Bullseye |

**Unterprograme**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Rückgabe-Typ** | **Parameter** | **Erklärung** |
| findQuadrant | String, Integer | Self, x, y | Berechnet, in welchem Quadranten der Dart sich befindet und ruft ein weiteres Unterprogramm, passend zum Quadranten auf |
| quadrantOne, quadrantTwo, QuadrantThree, QuadrantFour | String, Integer | Self, x, y | Berechnet im zugehörigen Quadranten den Winkel vom Mittelpunkt zum Dartpunkt und er berechnet auch die Reichweite dieser zwei Punkte |
| qOnePoints,  qTwoPoints,  qThreePoints,  qFourPoints | String, Integer | Self, x, y | Findet mit If-else statements das Segment und die Entität (Single, Double, Triple) vom Dartpfeil heraus |
| controlGeneral | String, Integer | Self, x, y | Findet mit If-else statements heraus, ob der Dartpfeil sich im Bull,Bullseye oder sogar ausserhalb der Dartscheibe befindet. |



## Prinzip des Ortens des Dartpfeils

Um den Standpunkt des Dartpfeils auf dem Dartboard zu finden, benötigt man die Entfernung vom Dartpfeil zum Mittelpunkt des Dartboarts und den Winkel zwischen diesen Punkten. (Siehe Kapitel „Berechnung der Punktzahl aus der Pfeilkoordinate“). Jedoch stellt sich dann die Frage, wie man diese Daten auf der Dartscheibe zuordnet. Dafür muss man nur die Maße der Scheibe kennen.

Abbildung 17: Einteilung der Dartscheibe in vier Quadranten

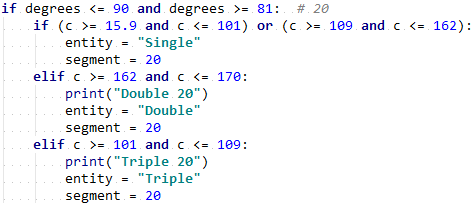
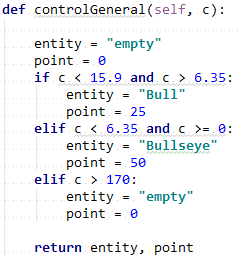
Mit If-Else Statements, vergleichen wir die eben berechneten Daten, mit den Maßen der Dartscheibe. Um diese Vergleiche übersichtlicher zu machen, teilen wir die Dartscheibe in 4 Quadranten (siehe Bild). Dann wird nur geschaut, wie weit der Dartpfeil von der Mitte entfernt ist und (wenn nötig) wird auch geschaut, in welchem Winkel der Pfeil zum Mittelpunkt steht.

Abbildung 18: Dartpfeil Feldkontrolle Beispiel: 20

Abbildung 19: Dartpfeil Bull, Bullseye und 0 Punkte Kontrolle

Aber dies ist nicht nötig, wenn sich der Dart außerhalb des Spielfeldes, oder innerhalb des Bulls befindet. Wenn dies der Fall ist, ist der Winkel überflüssig.

# Die GUI – Pascal

Eine GUI ist eine Grafische Benutzeroberfläche. Ihre Aufgabe ist es grafische Symbole darzustellen und bedienbar zu machen. In diesem Fall wird sie mithilfe von Python und der Directory Tkinter dargestellt. Innerhalb einer GUI gibt es unterschiedliche Steuerelemente wie Buttons, Entries und Scrollbars.

Um all dies verwenden zu können muss die Bibliothek „Tkinter“ allerdings erst importiert werden.

Ein Fenster wird erstellt mit dem Befehl Tkinter.tk(). Diesem Fenster muss aber noch ein Name gegeben werden. In diesem Fall ist der Name des Fensters „root“.

 Hier wird das erste „tk“ als Abkürzung für „Tkinter“ verwendet.

Damit dieses Fenster allerdings sich auch öffnet und offen bleibt muss man den Befehl .mainloop() verwenden. Allerdings wieder auf den Namen des Fensters bezogen.

 Hiermit wird dann ein Fenster erscheinen.

Eine Checkbox



Ein Button

Solche Elemente werden mit dem dazugehörigen Befehl kreiert. Für einen Knopf wird der Befehl  verwendet. And der Stelle des „frame“ wird festgelegt in welchem Fenster der Knopf erscheinen soll. Der „text=“ Befehl deklariert was auf dem Knopf später stehen soll und dass „command =“ sagt welche Funktion ausgeführt werden soll, wenn der knopf gedrückt wird.

Diese Steuerelemente müssen allerdings auch irgendwie platziert werden.

Dies funktioniert mit drei unterschiedlichen Befehlen. Dem .pack() , .place() oder dem .grid() Befehl.

Der .pack() Befehl bringt den Button oder die Checkbox auf den Bildschirm in die linke obere Ecke, da dort der Standardplatz für neue Objekte ist.

Der .place() Befehl ist deutlich genauer, aber auch schwerer zu bedienen.

In die Klammer werden verschiedene Attribute geschrieben, die bestimmen wo genau sich das Objekt befinden soll.

|  |  |
| --- | --- |
| **Name des Attributs** | **Aufgabe** |
| *relheight* | Gibt die Höhe in Relation zum Fenster an. |
| *relwidth* | Gibt die Breite in Relation zum Fenster an. |
| *relx* | Gibt den Abstand zum linken Fensterrand in Relation an. |
| *rely* | Gibt den Abstand zum oberen Fensterrand in Relation an. |

Dann gibt es noch den .grid() Befehl. Dieser ist sehr einfach zu verstehen und zu bedienen, hilft jedoch sehr um grob zu organisieren.

Wie bei dem .place() Befehl müssen auch bei .grid() Attribute in die Klammer.

|  |  |
| --- | --- |
| **Name des Attributs** | **Aufgabe** |
| *row* | Gibt and in welcher Reihe das Objekt sich befinden soll. Es gibt jedoch keine leeren Reihen. |
| *column* | Gibt and in welcher Spalte das Objekt sich befinden soll. Wie bei *row* keine leeren Spalten vorhanden. |

## Programmierung der GUI

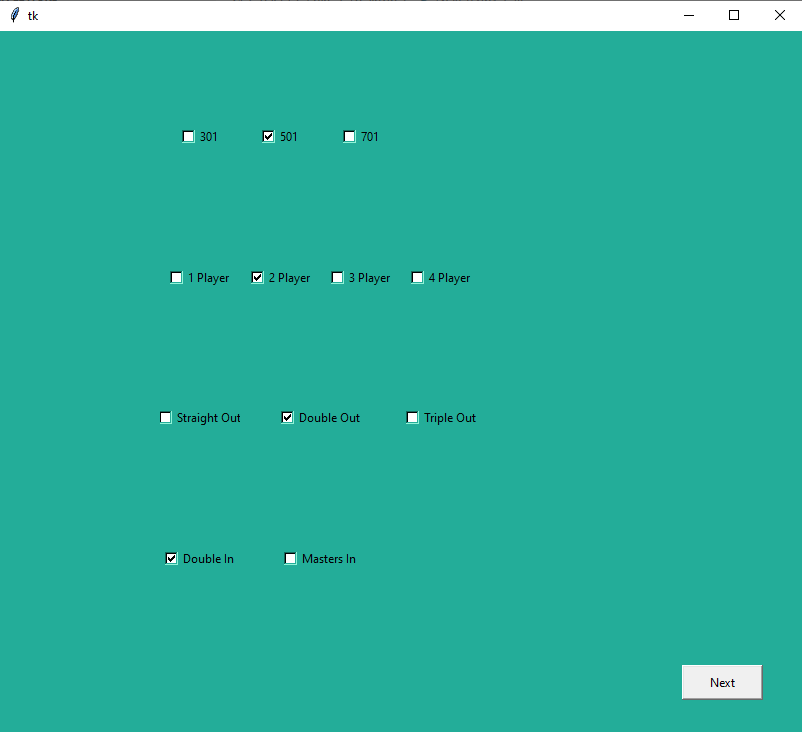
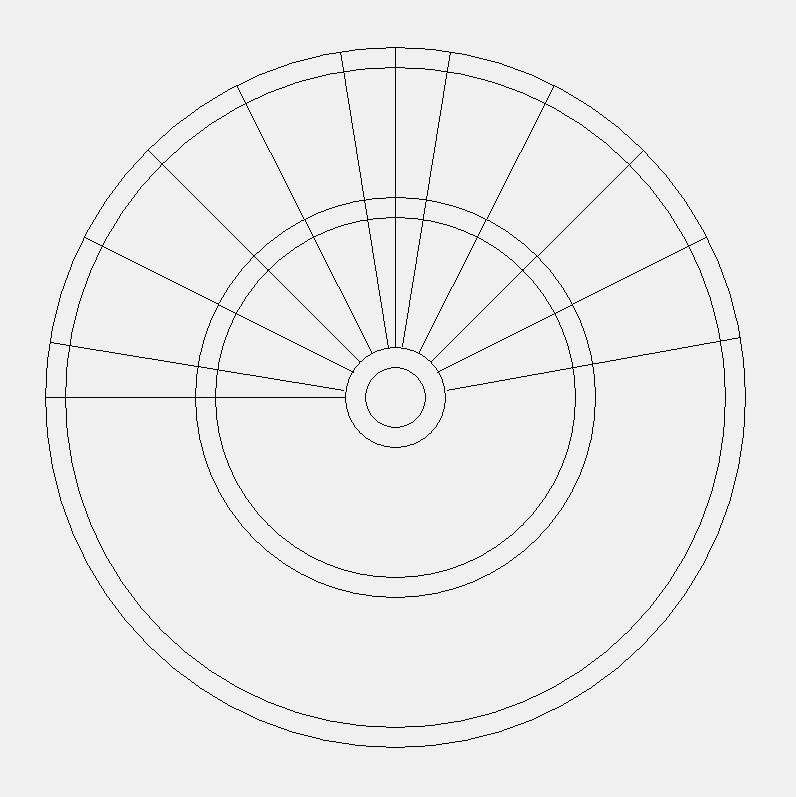
Ich hatte die Aufgabe die passende GUI zum Rest zu programmieren. Meine ursprüngliche Idee war Die GUI in Java zu programmieren, da ich schon Vorwissen in Java hatte. Da der Raspberry Pi allerdings mit Java nicht allzu gut klar kommt habe ich schnell gemerkt, dass ich auch Python benutzen muss. Also habe ich angefangen mir die Python Bibliothek Tkinter anzuschauen. Meine ersten Versuche liefen ganz gut bis ich dann gemerkt hab, dass das nicht genau das ist, was ich will. Nachdem dann diese Scheibe dabei heraus gekommen ist musst ich mir etwas anderes einfallen lassen. Ich dachte mir erst wieder zu Java zu gehen, aber Can und Laurin haben gesagt, dass das zu schwer wäre es mit den restlichen Programmen zu verbinden. Also habe ich mich weiter umgeschaut und bin wieder auf Tkinter gestoßen und hab es mir nochmal neu angeschaut. Ich habe gemerkt, dass ich mir die falschen Sachen angeschaut habe und es mir noch einmal angeschaut. Ich bin auf viele andere Möglichkeiten gestoßen und hab damit ein bisschen rumprobiert. Nach ein bisschen rumprobieren hatte ich den dreh relativ schnell raus. Der Umgang mit Python war anfangs etwas schwerer aber aufgrund des Vorwissens in Java und ein bisschen Übung ging das sehr einfach nach einer gewissen Weile. Ich habe dann dieses Fenster als Spieleinstellungen und als erstes Fenster erstellt.

Abbildung 20: Skizzierung der Dartscheibe

Abbildung 21: Spieloptionen der finalen GUI

# Software Dokumentation Arduino - Laurin

Die Aufgabe des Programms auf dem Microcontroller ist, einen Ausschlag der einzelnen Mikrofone zu erkennen und in diesem Moment den genauen Zeitpunkt festzuhalten. Nachdem alle Mikrofone ausgeschlagen haben, werden die Differenzen dieser Zeiten errechnet und diese mittels serieller Kommunikation an den Raspberry Pi zu übermitteln. Um dies zu verwirklichen stehen verschiedene Unterprogramme zur Verfügung:

|  |  |  |
| --- | --- | --- |
| **Name** | **Rückgabetyp** | **Beschreibung** |
| setup() | void | Legt die Modi der benötigten I/O Pins fest und stellt die serielle Übertragungsrate ein |
| loop() | void | wartet auf einen festgelegten seriellen Input, und ruft detectFirstMic() auf. |
| detectFirstMic() | void | Stellt das eigentliche Hauptprogramm dar und fragt mit Hilfe von micDetection() alle Mikrofone nacheinander ab, ob ein Mikrofon einen Ausschlag registriert. Wenn dies der Fall ist, speichert es die Systemzeit in Mikrosekunden und ruft nacheinander detectSecondMic(), timeToDistance()  und printDistances() auf. |
| detectSecondMic(int firstMic),  detectThirdMic(int secondMic, int firstMic) | void | Messen die Zeit des Ausschlags des jeweils zweiten und dritten Mikrofons zeitlicher Reihenfolge |
| micDetection(int micNumber) | boolean | Gibt zurück, ob das angefragte Mikrofon ausgeschlagen hat |
| timeToDistance() | void | Rechnet die erfassten Zeiten über die Schallgeschwindigkeit in Strecken um. |
| findStartTime() | long | Ermittelt die erste gemessene Zeit und gibt sie zurück |
| printDistances() | void | Übermittelt die umgerechneten Strecken über die serielle Schnittstelle an den Raspberry Pi |

Konkret wartet das Programm am Anfang immer auf das Schlüsselwort „Start“, welches der Raspberry an den Seriellen Eingang sendet, um mit der Messung der Zeit anzufangen. Hierbei wartet der Arduino darauf, dass ein Mikrofon ausschlägt, um die Zeit auf der Systemuhr abzufragen. Wichtig hierbei ist, dass das erste Mikrofon, welches ausschlägt, nicht zwingend das Mikrofon mit dem Index 1 (bzw. im Programm Index 0) sein muss. Deshalb muss der Arduino jedes Mikrofon immer wieder überprüfen. Das Mikrofon erkennt das Aufprallgeräusch des Dartpfeils und sendet so lange ein Signal an den digitalen Pin des Arduinos bis das Aufprallgeräusch verstummt ist. Da dieses Signal mehrere Millisekunden lang ist, das Programm aber nur den Anfang braucht, kann bei jedem Mikrofon nicht einfach auf ein Signal geprüft werden, sondern es muss auf einen Signalwechsel von LOW zu HIGH geprüft werden. Diese Prüfung wird auch Edge Detection genannt. Wenn ein Mikrofon ausgeschlagen hat, wird es für weitere Messungen ignoriert und die anderen Mikrofone werden nach dem gleichen Schema abgefragt, bis alle Mikrofone ausgeschlagen haben.

Aus den erfassten Systemzeiten in Mikrosekunden werden dann die Differenzen zur ersten Zeit gebildet und diese in Millimeter umgewandelt. Da sich Schall bei 20°C Raumtemperatur mit einer Geschwindigkeit von 343,2 m/s gleichmäßig in alle Richtungen ausbreitet[[10]](#footnote-10), muss der Wert in Mikrosekunden mit umgerechnet 0,3432 mm/μs multipliziert werden, um auf die gewünschten Millimeter zu kommen. Durch die Bildung der Differenzen ist die kürzeste Strecke immer 0 Millimeter lang. Die Gesamtstrecke dazu entspricht genau der Strecke d0 zwischen dem Dart und dem am nächsten liegenden Mikrofon, welche nicht gemessen werden kann, da der Arduino erst vom Landen des Darts erfährt, wenn diese Strecke bereits überwunden wurde.

Diese Strecken werden abschließend über serielle Kommunikation an den Raspberry übermittelt und von diesem weiterverarbeitet.

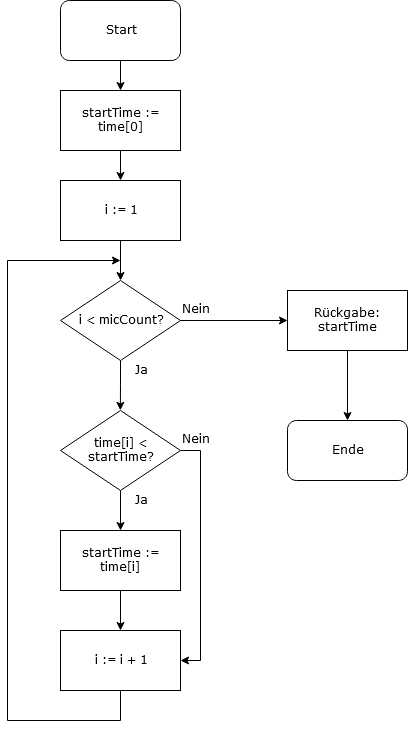


Abbildung 22: Exemplarischer Programmablaufplan von findStartTime()

Ein Bild, das Elektronik, Schaltkreis enthält.

Automatisch generierte Beschreibung

Abbildung 23: Schematische Verkabelung des Arduinos

# Materialien des Seminarkurses - Can

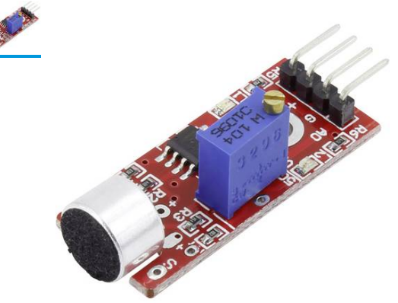
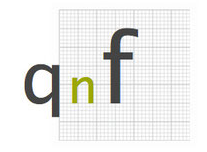
* Dartscheibe
  + Die Dartscheibe brauchten wir als Vorführungsstück, an dem wir auch die Mikrofone testen. Die Dartscheibe haben wir freundlicherweise von dem Dart Shop „Empire-Dart“ gesponsert bekommen.
* Raspberry Pi
  + Den Raspberry Pi haben wir von der qnf GmbH gesponsert bekommen. Neben dem Raspberry Pi, haben wir auch ein Gehäuse und verschiedene Kabel von der qnf erhalten
* Mikrofone
  + Die Mikrofone wurden uns freundlicherweise ebenfalls von der qnf GmbH zur Verfügung gestellt.
* Arduino
  + Den Arduino haben wir von den Gewerblichen Schulen Waldshut geliehen bekommen.
* Befestigung der Mikrofone
  + Für die Befestigung der Mikrofone an der Dartscheibe, haben wir einen 3D-Drucker benutzt, um die Halterungen zu drucken.
* Kabel
  + Die Kabel waren größtenteils beim Raspberry Pi oder beim Arduino dabei.

Abbildung 24: Logo qnf GmbH (https://image.jimcdn.com/app/cms/image/transf/dimension=218x10000:format=jpg/path/s597f50f05eb4c73f/image/i43ff55bbcd251b31/version/1454243925/image.jpg)

Abbildung 25: Verwendete Mikrofone (https://asset.conrad.com/media10/isa/160267/c1/-/de/1485297\_LB\_00\_FB/iduino-1485297-mikrofon-schallsensor.jpg?)

Abbildung 26: Logo Empire Dart (https://www.empire-dart.de/?gclid=Cj0KCQjwoub3BRC6ARIsABGhnyaGZDRS2rP3GHney740kgq-4-Dpm2cNR3gca3gV\_qFdNAXWZEjrSXEaAoNFEALw\_wcB)



Abbildung 27: Logo Gewerbliche Schulen Waldshut (https://www.gs-wt.de/files/theme/images/logo.png)

# Anbringung der Mikrofone an der Dartscheibe - Laurin

Unsere Idee war anfangs, die Dartscheibe auf einer Holzplatte zu befestigen und die Mikrofone so ringsum die Dartscheibe auf der Platte anzubringen. Aber da wir seit Ende letzten Jahres uneingeschränkten Zugang zu einem 3D-Drucker haben und die Baumärkte am Anfang der Corona Pandemie geschlossen waren, haben wir uns dazu entschieden, die Mikrofone mittels 3D-gedruckter Bauteile zu befestigen.

Teile aus dem 3D-Drucker werden Schicht für Schicht aus einem Kunststoff gefertigt, der in dünnen Ebenen übereinandergelegt wird. Als Vorlage dient ein digitales 3D-Modell. Dieses Modell wird mittels einer so genannten Slicer Software in für den Drucker verständlichen Code übersetzt. Im Slicer können auch verschiedene Druckparameter wie die Auflösung des Modells oder das Drucken von Stützmaterial für komplexe Formen vorgenommen werden.

Die 3D Modelle lassen sich mit verschiedener Software modellieren oder aus dem Internet laden. Für die Konstruktion der Mikrofonhalter haben wir auf das CAD Programm „Fusion 360“ von Autodesk zurückgegriffen. Wir haben die Halterungen mit dem Kunststoff PLA gedruckt, welcher sich besonders einfach verarbeiten lässt. Die Halterungen haben in der Mitte eine Aussparung, um dem Schall möglichst wenig Reflexionsfläche zu bieten. Grundlage für ein solches Modell ist immer eine 2-dimensionale Zeichnung. Aus dieser wird Extrusion in ein 3-dimensionales Objekt erstellt. Details wie Bohrungen und Fasen können am Ende des Modellierungsprozesses eingefügt werden.

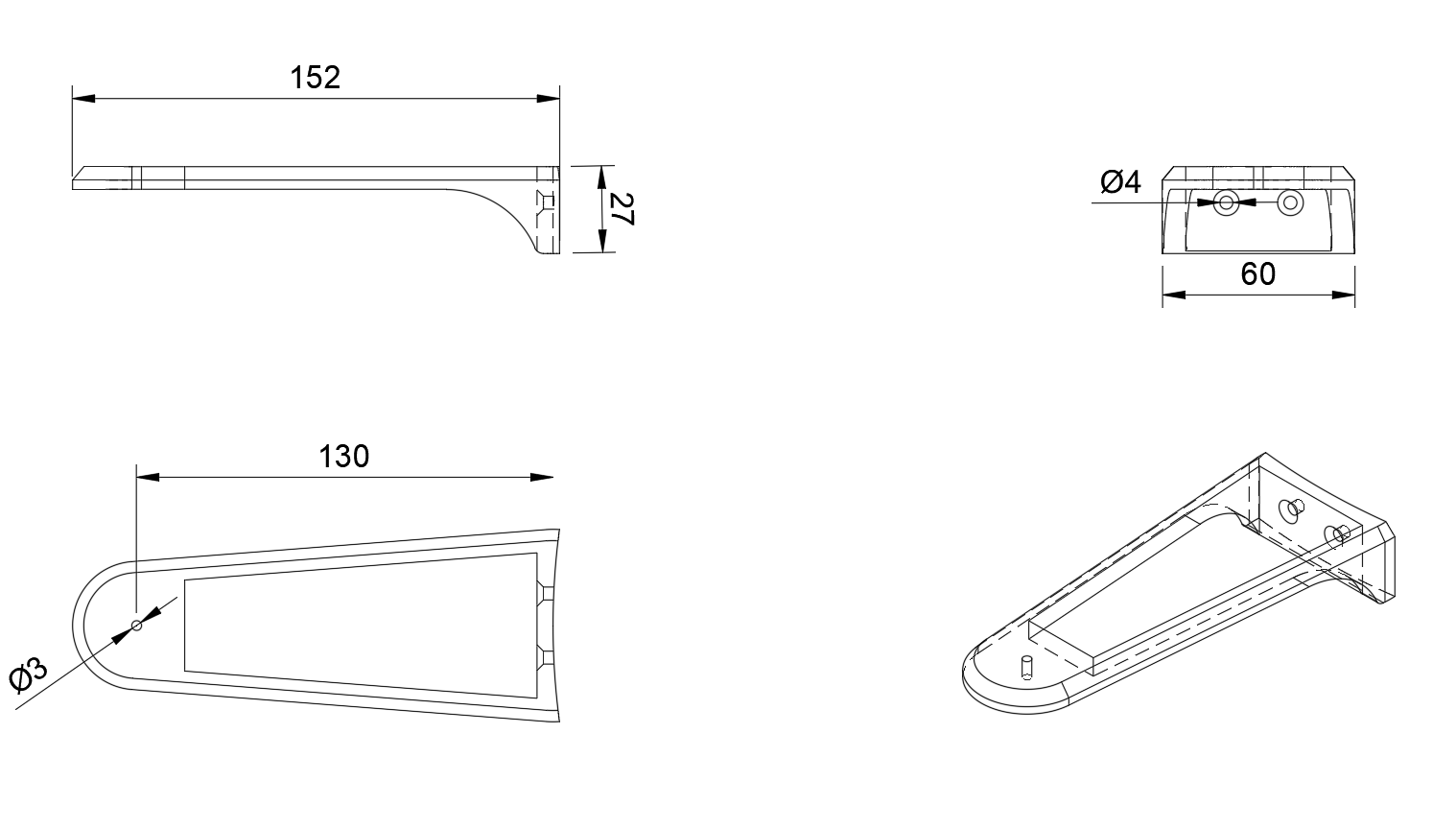


Abbildung 28: Aus 3-d Modell generierte technische Zeichnung der Mikrofonhalter

Der Vorteil dieser Fertigungstechnik ist, dass neben dem Drucker kein besonderes Werkzeug und auch keine Holz-, oder Metallbearbeitungskenntnisse benötigt werden. Auch lässt sich ein einmal konstruiertes Objekt beliebig oft fertigen, ohne dass sich signifikante Qualitätsunterschiede feststellen lassen. Auch lassen sich Änderungen und Korrekturen in der CAD Software schnell realisieren, ohne das gesamte Objekt neu zu modellieren. Da auch alle Löcher für die Schrauben im Druck berücksichtigt wurden, beschränkte sich die Nachbearbeitung lediglich auf das Schleifen einiger Oberflächen.

Ein Nachteil ist, dass die Mikrofone so direkt mit der Dartscheibe verbunden sind und eventuelle Erschütterungen der Scheibe bei Auftreffen des Darts das Ergebnis verfälschen könnten.

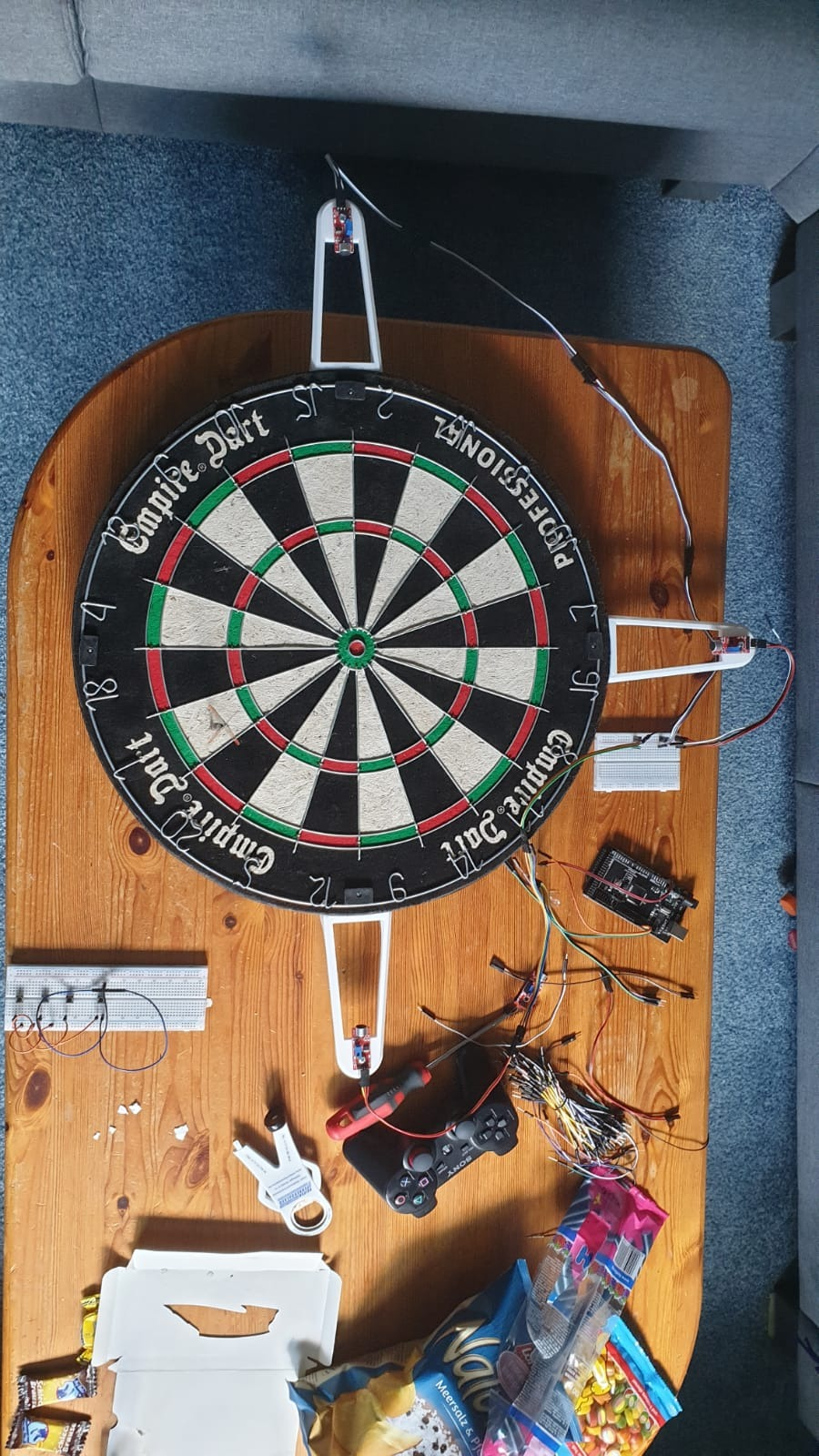


Abbildung 29: Aufbau und Test der Dartscheibe

# Der Aufbau und Tests der Dartscheibe - Laurin

Da Teile unserer Materialien während der Schulschließung noch in der Schule lagerten konnten wir die fertige Dartscheibe erst nach der Schulöffnung zusammenbauen und testen. Das Anbringen der Mikrofonhalter an der Richtigen Position gestaltete sich als Herausforderung, da sich die genauen Positionen der Mikrofone nur schwer ausmessen und anzeichnen ließen und der Akkubohrer, auf den wir gezwungenermaßen angewiesen waren, um die Löcher für die Schrauben in der Dartscheib vorzubohren, auf dem runden Metallrand sehr leicht verrutschte. Problematisch ist hierbei, dass eine kleine Verschiebung der Positionen der Mikrofonhalter durch den verlängerten Arm eine deutlich größere Abweichung der Mikrofone von ihren eigentlichen Plätzen zur Folge hat. Beim Festschrauben der Mikrofone auf den Haltern fiel uns auf, dass diese ohne Unterlegscheibe durch Lötstellen an der Unterseite schief auf den Halterungen saßen. Glücklicherweise konnten wir diese in wenigen Minuten konstruieren und Drucken. Die Mikrofone haben wir über eine Steckplatine mit dem Arduino verkabelt. Die Mikrofone besitzen eine Stellschraube, um ihre Sensibilität einzustellen. Die Herausforderung hierbei ist, alle Mikrofone so gut es geht gleich sensibel so einzustellen, dass jedes Mikrofon den Aufprall des Darts erkennt, ohne bei leiseren Störgeräuschen, beispielsweise durch Bewegungen in der Nähe der Dartscheibe, auszuschlagen. Zudem variierte die Sensitivität je nachdem, ob der Arduino mit dem Laptop, mit dem wir das Programm aufgespielt und getestet haben, oder mit dem Raspberry Pi verbunden war. Da wir noch keine Anbringungsmöglichkeit für den Arduino und die Steckplatine an der Dartscheibe haben, konnten wir die Scheibe noch nicht an der Wand testen, sondern haben sie nur auf einen Tisch gelegt und die Darts von oben herauffallen lassen.

Bei den ersten Tests stellten wir fest, dass die errechnete Position der Dartpfeile teils um mehrere Meter daneben lag oder sich gar nicht erst errechnen ließ. Daraufhin haben wir noch einmal das Python Programm überprüft, das die Position des Pfeils und die Punktzahl errechnet und haben dort keine Fehler finden können. Allerdings haben wir im Arduino Code Unstimmigkeiten bei den Datentypen behoben. Der Test danach lieferte deutlich bessere, wenn auch immer noch keine zufrieden stellenden Ergebnisse. Trotz Treffen der Dartscheibe lag das Ergebnis fast immer bei 0 Punkten. Das richtige oder ein benachbartes Feld wurde sehr selten ausgegeben. In unregelmäßigen Abständen waren die gemessenen Längen immer noch so falsch, dass sich die Koordinaten nicht errechnen ließen.

## Mögliche Gründe und Verbesserungen für die fehlerhaften Messungen

Da wir alle Programme noch einmal manuell überprüft und verbessert haben, vermuten wir die Ursache für die hohen Abweichungen der Messergebnisse in den Mikrofonen. Genauer ist unsere Vermutung, dass die Mikrofone durch die unterschiedlichen Sensibilitäten nicht am gleichen Punkt des Schallsignals ausschlagen. Wenn man sich die Schallwelle des Aufpralls stark vergrößert anschaut, dann erkennt man, dass diese ihren lautesten Punkt nicht schlagartig erreicht, sondern sich bis dort gewissermaßen aufbaut. Wenn die Mikrofone unterschiedlich kalibriert sind, kann es sein, dass ein Mikrofon schon auf halber Lautstärke ausschlägt, während ein anderes erst auf voller Lautstärke aufschlägt.

Eine Lösung hierfür könnte das analoge Aufnehmen des Schalls und anschließende manuelle Weiterverarbeitung beispielsweise mittels Kreuzkorrelation mit dem Arduino oder über Python sein. Damit lässt sich die Zeitverzögerung zweier sonst identischer Signale ermitteln[[11]](#footnote-11). Ein Problem hierbei ist, dass die Signale nicht von jedem Mikrofon exakt gleich erkannt werden, da allein schon die Lautstärke je nach Entfernung von Dartpfeil zu Mikrofon variiert. Eine andere Art der Weiterverarbeitung des analogen Signals wäre, nur den Zeitpunkt des höchsten Ausschlags zu bestimmen, welcher bei jedem Mikrofon an der gleichen Stelle des Signals sein sollte und so genau die z1eitliche Differenz zwischen den Aufnahmen darstellt. Da dieser auch bei jedem Mikrofon relativ zum Rest des Schallsignals ist, sind Lautstärkeunterschiede zwischen den einzelnen Aufnahmen irrelevant.

# Fazit

Abschließend können wir sagen, dass der Seminarkurs erfolgreich war. Obwohl die Erkennung der Dartpfeile am Ende nicht so präzise und zuverlässig funktioniert hat wie wir es uns erhofft hatten, konnten wir dennoch vieles dazulernen. Durch das Zusammenspiel vieler technischer Disziplinen wie die Programmierung einer GUI, objektorientierter Programmierung, dem Einsatz von Mikrocontrollern in Verbindung mit Sensoren und Eingabegeräten oder serieller Kommunikation, haben wir viel neues technisches Wissen erlangen können.

Allerdings haben wir auch gemerkt, dass wir den Umfang und die Komplexität des Projekts anfangs unterschätzt haben. Besonders die Unterschiede zwischen den mathematischen Modellen unter perfekten Bedingungen und der realen Applikation, welche durch Faktoren wie die Genauigkeit der Mikrofone oder Störgeräusche beeinflusst wird, haben wir erst spät im Projekt berücksichtigt. Da wir unsere Materialien erst nach den coronabedingten Schulschließungen aus der Schule holen konnten, fiel die reale Testphase unerwartet kurz aus, weshalb uns leider keine Zeit mehr blieb, um die erwähnten Verbesserungsansätze in die Tat umzusetzen.

Trotzdem war der Seminarkurs eine spannende und lehrreiche Erfahrung.

# Literaturverzeichnis

(2019). Abgerufen am 27. Juni 2020 von Wikipedia: Display Serial Interface: https://de.wikipedia.org/wiki/Display\_Serial\_Interface

*Arduino Store*. (2020). Abgerufen am 28. Juni 2020 von https://store.arduino.cc/arduino-nano

*Arduino Store*. (2020). Abgerufen am 28. Juni 2020 von https://store.arduino.cc/arduino-mega-2560-rev3

Chaves, D. H. (kein Datum). *Die Kreuzkorrelation.* TU Bergakademie Freiberg.

Cipriani, L. (2020). *GitHub*. Abgerufen am 26. Juni 2020 von https://github.com/arduino/arduino-pro-ide/releases

*Dart Club Verzeichnis*. (kein Datum). Abgerufen am 27. Juni 2020 von https://www.dart-club.de/masse-hoehe-entfernung/

*Dartscheibentest*. (kein Datum). Abgerufen am 27. Juni 2020 von https://www.dartscheiben-test.de/dartscheiben-masse-durchmesser-profi-dartscheibe-material.html

Ehrenberg, M., & Patel, A. (2005). *The Design and Implementation of an Automated Dartboard.*

*Elektronik Kompendium: Raspberry Pi*. (2020). Abgerufen am 27. Juni 2020 von https://www.elektronik-kompendium.de/sites/com/1904221.htm

*GisGeography*. (2020). Abgerufen am 11. Juni 2020 von https://gisgeography.com/trilateration-triangulation-gps/

Hofferbert, B. (2019). *Heise Online*. Abgerufen am 27. Juni 2020 von https://www.heise.de/tipps-tricks/Raspberry-Pi-Das-koennen-die-GPIO-Pins-4583823.html

*MAKER PRo*. (2018). Abgerufen am 27. Juni 2020 von https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino

*pySerial documentation*. (kein Datum). Abgerufen am 27. Juni 2020 von https://pythonhosted.org/pyserial/

*Raspberry Forum*. (2017). Abgerufen am 27. Juni 2020 von https://www.raspberrypi.org/forums/viewtopic.php?t=196024

*Raspberry Pi GPIO*. (kein Datum). Abgerufen am 27. Juni 2020 von https://www.raspberrypi.org/documentation/usage/gpio/

*Raspberry Pi Tutorial: Arduino und Raspberry Pi miteinander kommunizieren lassen*. (kein Datum). Abgerufen am 27. Juni 2020

*Raspberry.org*. (kein Datum). Abgerufen am 27. Juni 2020 von https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Riedl, D. S. (2018). *IT-Business*. Abgerufen am 27. Juni 2020 von https://www.it-business.de/was-ist-eine-schnittstelle-a-714831/

*SoftSelect*. (kein Datum). Abgerufen am 27. Juni 2020 von http://www.softselect.de/business-software-glossar/interface

*Toptarif*. (27. Juni 2020). Von https://www.toptarif.de/internet/wissen/schnittstelle/ abgerufen

*Wikipeda: Programmierschnittstelle*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Programmierschnittstelle

*Wikipedia: Camera Serial Interface*. (2020). Von https://en.wikipedia.org/wiki/Camera\_Serial\_Interface abgerufen

*Wikipedia: Darts*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Darts

*Wikipedia: Einplatinencomputer*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Einplatinencomputer

*Wikipedia: Ethernet*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Ethernet

*Wikipedia: https://de.wikipedia.org/wiki/Accelerated\_Graphics\_Port*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Accelerated\_Graphics\_Port

*Wikipedia: Micro-USB-Standard*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Micro-USB-Standard

*Wikipedia: Peripheral Component Interconnect*. (2018). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Peripheral\_Component\_Interconnect

*Wikipedia: Raspberry Pi*. (kein Datum). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Raspberry\_Pi

*Wikipedia: Schallgeschwindigkeit*. (2020). Abgerufen am 26. Juni 2020

*Wikipedia: Schnittstelle*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Schnittstelle#Maschinenschnittstelle

*Wikipedia: Serielle Schnittstelle*. (2019). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Serielle\_Schnittstelle

*Wikipedia: Triangulaion (Geodäsie)*. (2019). Abgerufen am 11. Juni 2020 von https://de.wikipedia.org/wiki/Triangulation\_(Geod%C3%A4sie)

*Wikipedia: Triangulation (Fläche)*. (2018). Abgerufen am 11. Juni 2020 von https://de.wikipedia.org/wiki/Triangulation\_(Fl%C3%A4che)

*Wikipedia: Triangulation (Messtechnik)*. (2020). Abgerufen am 11. Juni 2020

*Wikipedia: Universal Serial Bus*. (2020). Abgerufen am 27. Juni 2020 von https://de.wikipedia.org/wiki/Universal\_Serial\_Bus

*Wikipedia: Verzeichnung*. (2020). Abgerufen am 02. Juni 2020 von https://de.wikipedia.org/wiki/Verzeichnung

1. (Wikipedia: Verzeichnung, 2020) [↑](#footnote-ref-1)
2. (Ehrenberg & Patel, 2005) [↑](#footnote-ref-2)
3. (Wikipedia: Triangulation (Fläche), 2018) [↑](#footnote-ref-3)
4. (Wikipedia: Triangulaion (Geodäsie), 2019) [↑](#footnote-ref-4)
5. (Wikipedia: Triangulation (Messtechnik), 2020) [↑](#footnote-ref-5)
6. (GisGeography, 2020) [↑](#footnote-ref-6)
7. (Cipriani, 2020) [↑](#footnote-ref-7)
8. (Arduino Store, 2020) [↑](#footnote-ref-8)
9. (Arduino Store, 2020) [↑](#footnote-ref-9)
10. (Wikipedia: Schallgeschwindigkeit, 2020) [↑](#footnote-ref-10)
11. (Chaves) [↑](#footnote-ref-11)