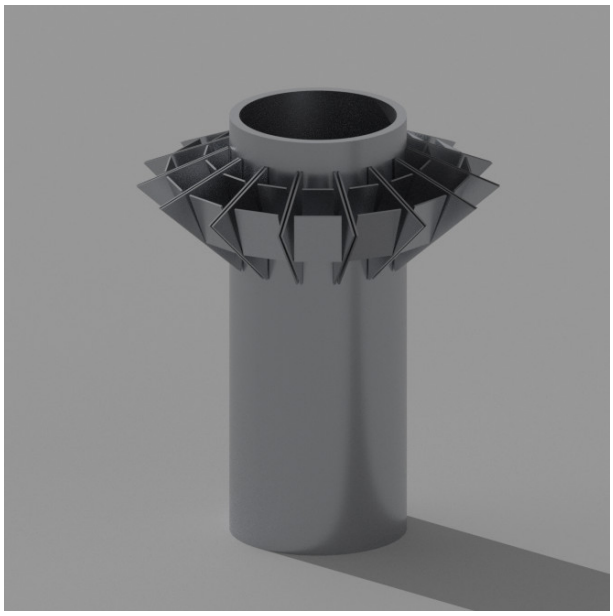
Cross-Innovation-Class 2022
Prof. Hoffmann

Projektbericht Team Frankfurt



TODO

TODO Untertitel

Sven Hülsen, Robin von Berg

13. Juli 2022

Inhaltsverzeichnis

Seite

1 Einleitung

Dieser Projektbericht ist im Rahmen der Cross Innovation Class 2022 entstanden.

1.1 Cross Innovation Class

Die Cross Innovation Class, kurz CIC, ist eine von der Hamburg Kreativ Gesellschaft organisierte Veranstaltung in Kooperation mit Universitäten und Fachhochschulen des Hamburger Umlands. Idee der CIC ist es Studierende verschiedener Fachrichtungen unterschiedlicher Universitäten ein Semester lang in interdisziplinären Teams an Projekten zusammenarbeiten.
Teilnehmen konnten Studierende des Studiengangs Stadtplanung der Hafencity Universität Hamburg, der Studiengänge Produkt- und Interior-Designer der Akademie Mode & Design des Standorts Hamburg und der Studiengänge Informatik, Technische Informatik, Wirtschaftsinformatik, Smart Technology und IT-Ingenieurwesen der Fachhochschule Wedel.
Die Cross Innovation Class lief dabei dieses Jahr unter dem Thema Resilient Cities. In Bezug auf dieses Oberthema wurden fünf Partnerunternehmen ausgesucht, die jeweils eine Fragestellung mit in die CIC gebracht haben.

Resilient Cities

Resilienz, ein wichtiger Faktor in vielen Lebensbereichen. Ein Attribut das Anpassungsfähigkeit und einen standhaften Umgang mit Krisen beschreibt. Neben persönlicher und ökonomischer Resilienz übernimmt Resilienz auch eine immer wichtiger werdende Rolle im Blick auf Gemeinden und Städte. Vorallem im Bezug auf Extremwetterereignisse und dem immer weiter voranschreitenden Klimawandel braucht es neue Ideen und Konzepte.

Daher gibt es viele Bestrebungen auf globaler, europäischer und nationaler Ebene dieses Thema voranzubringen. Eine Institution ist der Urban Resilience Club, der Urbane Resilienz wie folgt definiert:
„Urban Resilience - The measurable ability of any urban system, with its inhabitants, to maintain continuity through all shocks and stresses, while positively adapting and transforming toward sustainability.“[a]

[a] `https://urbanresiliencehub.org/what-is-urban-resilience/`

Partner dieses Jahr waren die Stadt Frankfurt mit der Stabsstelle Digitalisierung, die ACO Gruppe, Hamburg Marketing, Hamburg Institute for Innovation, Climate Protection and Circular Economy (HiiCCE) und das Wald Stadt Labor Iserlohn.

Fünf Teams, jeweils bestehend aus Studierenden jeder Universität und einem Praxispartner durchliefen über knapp 12 Wochen ein Design Thinking Prozess, der

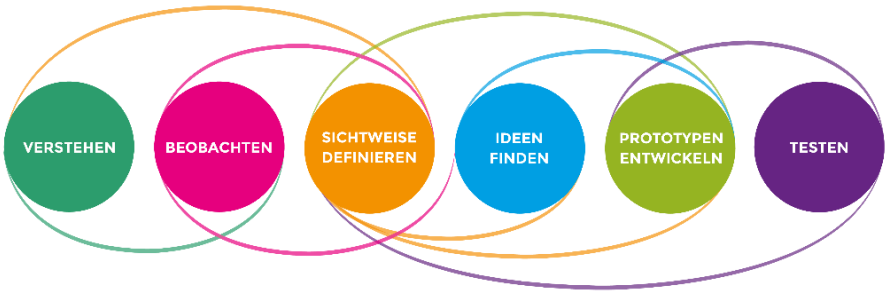


Abbildung 1.1: Der Design-Thinking-Prozess[1]

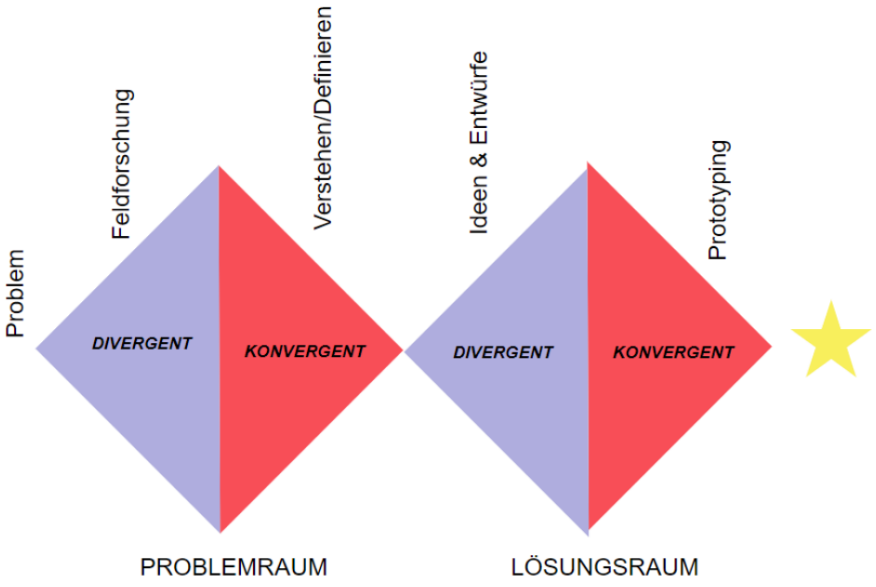


Abbildung 1.2: Denkmodell „Double Diamond“[1]

Design Thinking

Design Thinking ist ein Prozess zur Ideenfindung und -entwicklung. Dabei steht der Mensch im Mittelpunkt der mit einem Problem konfrontiert ist.
Abbildung 1.1 Zeigt die sechs Phasen, sowie die
In vielen Veranstaltungen der CIC ist uns der „Double Diamond“ (Abbildung 1.2) begegnet, denn auch der Ablauf der Cross wurde in einem solchen Rahmen strukturiert.

Dieses Format stellt im Vergleich zu den sonst eher theoretischeren oder fachspezifischeren Veranstaltungen eine willkommene Ergänzung da.

1.1.1 Ablauf der CIC

Das Projekt wurde in drei große Phasen eingeteilt, Konzept, Entwurf und Prototyping. Neben einem KickOff zu Beginn gab es am Ende jeder Phase eine Feedback Runde mit der gesamten Class.
In der KickOff Veranstaltung haben sich die Praxispartner und ihre Fragestellung vorgestellt und wir haben unser Team kennengelernt.
Zusätzlich wurde allen Interessierten vor der Abschlussveranstaltung ein sehr lehrreiches Pitch-Training angeboten.
Wie sind die Teams enstanden, was hatten wir für (Regel)Termine, wie viel Zeit hatten wir für die unterschiedlichen Aufgaben, etc. Skizzierung des CiC-Prozesse.

Projektphasen

8. April 2022	Kick-Off
11. April 2022 - 21. April 2022	Analyse & Konzept Phase
25. April 2022 - 6. Mai 2022	Entwurfsphase
9. Mai 2022 - 23. Juni 2022	Prototyping & Modellbau Phase
30. Juni 2022	Abschlussveranstaltung

1.2 Team Frankfurt

Unser Team

1.3 Stadt Frankfurt - Stabstelle Digitalisierung

Beschreibung des Industriepartners

Abbildung 1.3: CIC Team Stadt Frankfurt
v.l.n.r.: Maybritt Braun (AMD), Annika Fröhlich (AMD), Robin von Berg (FHW), Lucas Below (AMD), Sven Hülsen (FHW), Celina Krug (HCU), Moritz Hillen (HCU), Jochen Schmitz (FES). Abwesend: Florian Bucher (HCU), Mechthild Schulze & Karina Mombauer (Stadt Frankfurt Stabstelle Digitalisierung)

1.4 Frankfurter Entsorgungs- und Service GmbH (FES)

Beschreibung des Industriepartners

2 Projektfindung

Design THinking einer aus drei, aber nicht ins Detail der anderen.
Wie hat sich Skizze der Idee und der Realisierung

3 Projektbeschreibung

Eingehendere Beschreibung der Projekt-Idee untermauert mit Skizzen/Zeichnungen

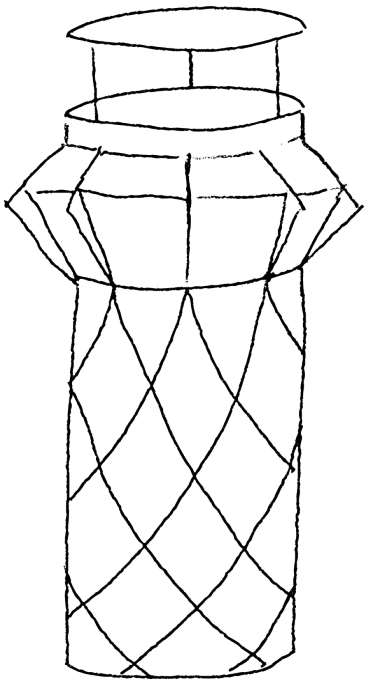


Abbildung 3.1: Skizze des Entwurfs mit Rautenmuster

4 Aufgaben

4.1 Aufgabenverteilung

Darstellung der Aufgabenverteilung innerhalb des Teams, ggf. durch eine Tabelle
Verweise auf das Projekt-Repository in dem weitere Projekt-Artefakte zu finden sind (s.u.).

5 Technische Realisation

5.1 Aufgabenbereiche

Die technischen Aspekte des Gerippten lassen sich grob in folgende Teilbereiche aufgliedern:

- Beleuchtung der Tonnenaußenwand inklusive Animation der Lichteffekte sowie Beleuchtung des Flaschenrings
- Sensorik zur Detektion von Flaschen im Flaschenring
- Detektion von Gegenständen die in den Behälter geworfen werden
- Messung des Füllstandes
- Austausch von Informationen zwischen mehreren Endgeräten
- Verarbeitung der Sensordaten und Ansteuerung der Beleuchtung

In den nachfolgenden Kapiteln werden die zur Realisierung der verschiedenen Teilaspekte genutzten Komponenten beschrieben und diskutiert.

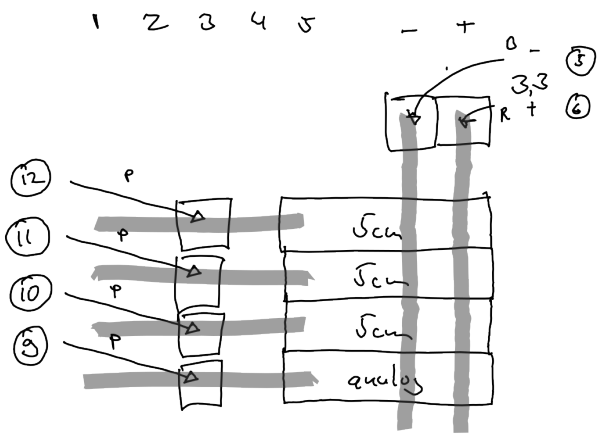


Abbildung 5.1: Caption

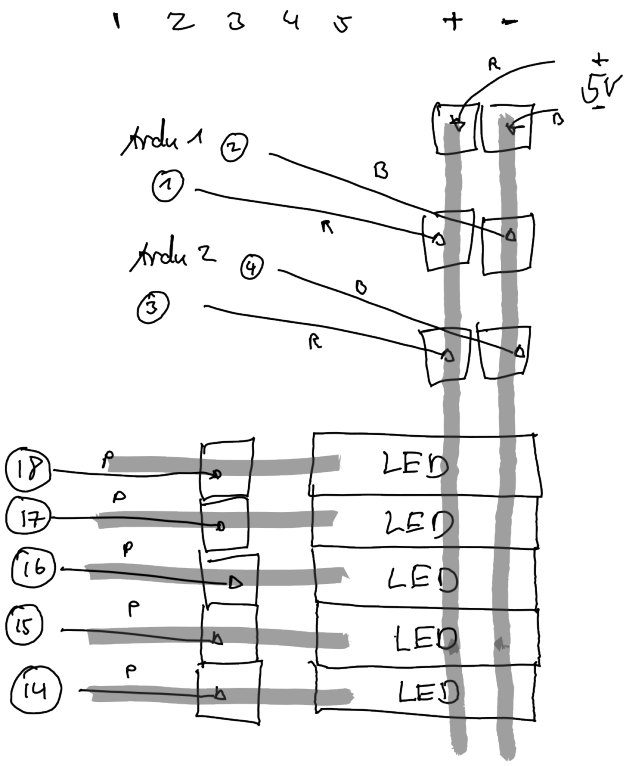


Abbildung 5.2: Caption

5.2 Mikrocontroller

5.2.1 Diskussion

Warum Arduino

5.2.2

C++ & PlatformIO

5.3 Beleuchtung

5.3.1 Diskussion

5.3.2 Implementation

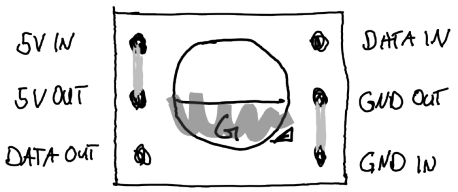


Abbildung 5.3: Zeichnung Pin Belegung der NeoPixel SK6812RGBW

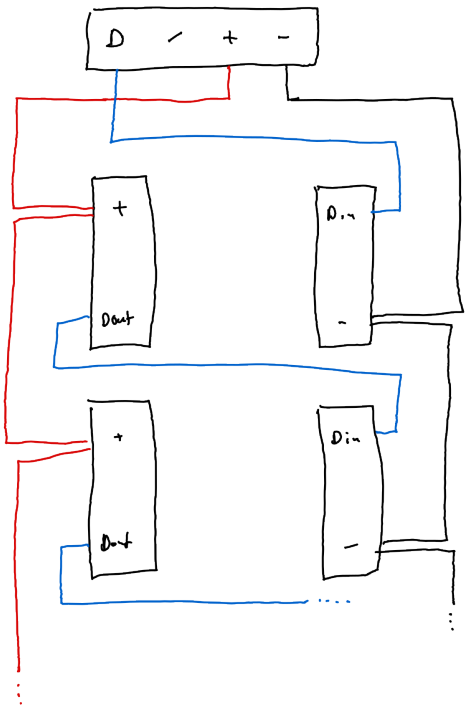


Abbildung 5.4: Caption

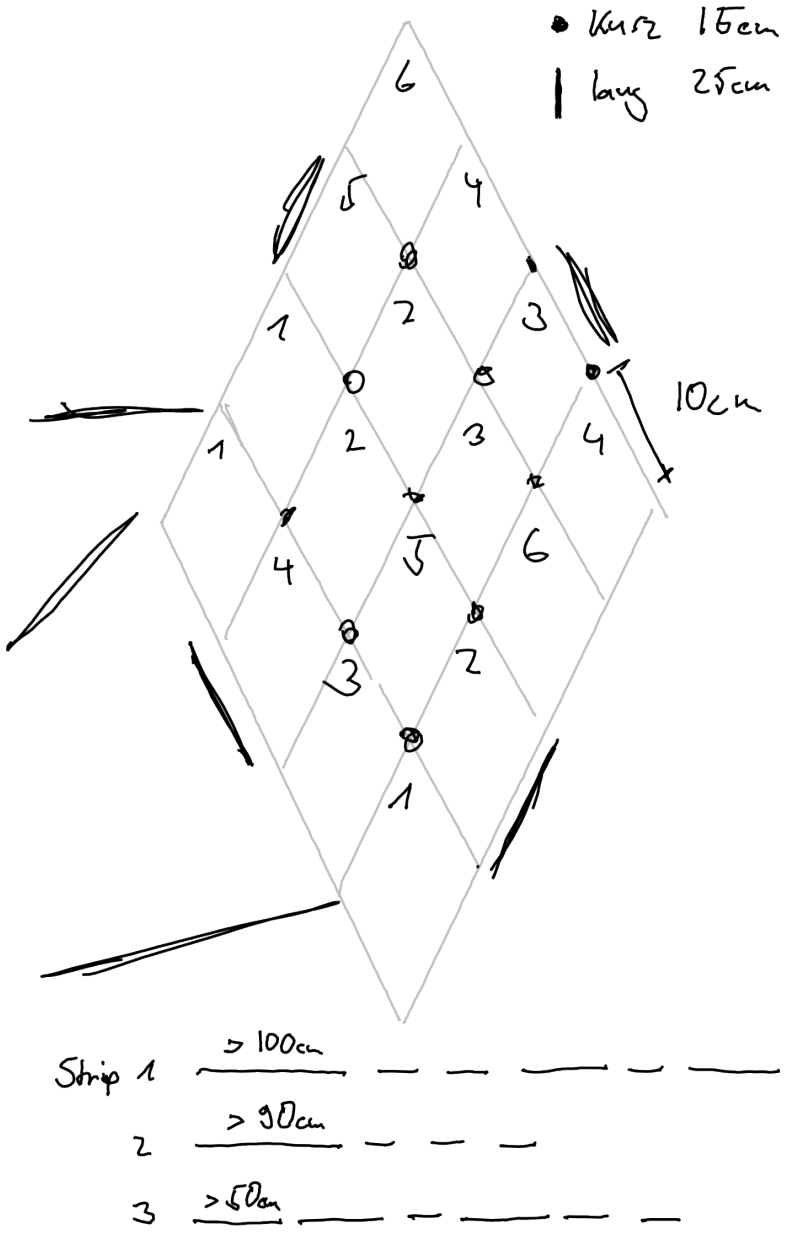


Abbildung 5.5: Caption

5.4 Animation

5.4.1 Diskussion

5.4.2 Implementation

5.5 Einwurfserkennung

5.5.1 Diskussion

Ein wichtiger Punkt bei der Realisierung der Einwurfserkennung ist, dass nur Gegenstände erkannt werden sollen, die gänzlich in den Behälter geworfen wurden und nicht wieder herausgezogen werden, wie es beispielsweise beim Hereinhalten einer Hand in den Behälter der Fall ist. Um dies zu implementieren muss der Gegenstand auf mindestens zwei übereinanderliegenden Ebenen detektiert werden. Wird auf der oberen Ebene der Gegenstand erkannt, wird ein möglicher Einwurf detektiert. Jedoch erst wenn die obere ebene keinen Gegenstand mehr detektiert, während die untere Ebene diesen weiterhin detektiert ist der Einwurf des Gegenstandes abgeschlossen. Sollte der Gegenstand hingegen zuletzt lediglich auf der oberen Ebene detektiert werden, so wurde er wieder aus dem Behälter herausgezogen und somit wird dies nicht als Einwurf gewertet.

Die Realisierung der zwei Detektionsebenen durch zwei einzelne Sensoren oder Lichtschranken hat jedoch das Problem zur Folge, dass die beiden Sensoren miteinander interferieren, wenn sie die gleiche Lichtwellenfrequenz nutzen, was zur Folge hat, dass eine genaue Abgrenzung zwischen den Ebenen nicht möglich ist. Zur Realisierung dieser Funktion ist es deshalb notwendig Sensoren zu verwenden, die unterschiedliche Lichtwellenfrequenzen verwenden, oder es muss ein Sensor verwendet werden, der die Möglichkeit bietet Objekte in einem dreidimensionalen Bereich zu detektieren, statt nur auf einer bestimmten Ebene zu messen.

Bei der Recherche nach zur Verfügung stehenden Komponenten fiel die Wahl letztlich auf den Mini dToF Imager TMF8821 von SparkFun. Dabei handelt es sich um einen so genannten direct Time of Flight Sensor. Damit ist ein Sensor gemeint, welcher viele kurze Lichtimpulse aussendet und die reflektierten Lichtimpulse wieder detektiert. Um die Entfernung zum Objekt zu bestimmen, welches den Lichtimpuls reflektiert hat, wird die Zeit gemessen, die zwischen der Aussendung des Lichtimpulses und der Detektion des reflektierten Lichtimpulses verstreicht. Eine Besonderheit des TMF8821 ist, dass dieser einen in mehrere Felder aufgeteilten Messbereich besitzt. So kann die Entfernung zu einem Gegenstand beispielsweise in 9 verschiedenen Messbereichen, aufgeteilt auf ein 3x3 Quadrat, gemessen werden. Der Messbereich kann Softwareseitig angepasst werden und somit sind verschiedene Messwinkel und Submessbereiche in den Ausprägungen 3x3, 4x4 und 3x6 möglich. Das bedeutet, dass der Messbereich des Sensors bereits in mehrere Ebenen aufgeteilt ist, die getrennt voneinander betrachtet werden können. Technisch realisiert wird dies durch Single Photon Avalanche Photodioden (SPAD), welche hinter einer speziellen Linse angebracht sind, die den Messbereich auf die Fotodioden fokussiert.

Die einzelnen Photodioden werden dann den verschiedenen Messzellen zugeordnet und somit kann ein Photon, dass von einer bestimmten SPAD empfangen wird einem Bereich im Feld der Messung zugeordnet werden. Der im folgenden verwendete Begriff 'SPAD Map' bezeichnet die Konfiguration des Messfeldes durch Aufteilung in Unterbereiche mit jeweils fest zugeordneten SPADs.

5.5.2 Implementation

Um den TMF8821 Softwareseitig zu implementieren wird die von SparkFun zur Verfügung gestellte Bibliothek verwendet. Diese beinhaltet alle notwendigen Methoden um den Sensor zu konfigurieren, Messungen durchzuführen und die Messergebnisse auszuwerten.

Die Voreingestellte SPAD Map des Sensors besitzt eine Größe von 3x3 Messbereichen und die Messwinkel betragen in der Horizontalen 33° und in der Vertikalen 32°. Die ersten Messungen, mit dem in der Bibliothek zur Verfügung gestellten Codebeispiel Example-01_Basic, welches alle Messewerte auf dem Seriellen Monitor ausgibt, lieferten sehr zufriedenstellende Ergebnisse. Dabei wurde eine Hand in unterschiedlich abgemessenen Abständen und an unterschiedlichen Positionen vor den Sensor gehalten und die Werte auf dem Seriellen Monitor mit Position und Entfernung der Hand abgeglichen.

Für die geplante Einwurföffnung des Gerippten ist ein Messwinkel von maximal 33° jedoch nicht ausreichend, weshalb die beschriebenen Probemessungen ebenfalls mit anderen SPAD Map Konfigurationen durchgeführt wurden. Dabei war zu beobachten, dass die Konfigurationen, mit einem 4x4 oder 3x6 Messbereich besonders im Randbereich viele unplausible Ergebnisse lieferten. Da die Fehlerursache für dieses Verhalten nicht in kurzer Zeit ausgemacht werden konnte, wurde sich aufgrund der wenigen zur Verfügung stehenden Zeit auf die Verwendung einer SPAD Map mit einem 3x3 Messbereich konzentriert. Die vorkonfigurierte SPAD Map mit der ID 6 bietet mit 52° in der Vertikalen den größten Messwinkel aller vorkonfigurierten SPAD Maps, weshalb diese letztendlich im Prototypen Verwendung fand. Das Schema dieser SPAD MAP wird in Abbildung 5.6 dargestellt.

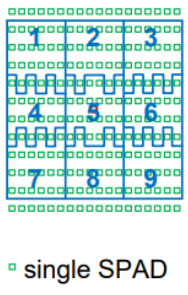


Abbildung 5.6: Schema der verwendeten SPAD Map

Da bei dieser SPAD Map Konfiguration der vertikale Messwinkel größer ist als der horizontale Messwinkel muss der Sensor für einen maximal großen Messbereich um 90° versetzt eingebaut werden, sodass die Messfelder 1, 4 und 7 die obere Messreihe bilden und die Messfelder 3, 6 und 9 die untere. Da jedoch auch der horizontale Messwinkel mit 41° einen großen Messbereich abdeckt, der nach der Rotation in der Vertikalen aufgespannt ist, wird die daraufhin obere Messebene mit den Feldern 1, 4 und 7 bei der Messung nicht ausgewertet, da dieser Messbereich zu großen Teilen oberhalb der Einwurföffnung gelegen ist. Die zwei geforderten Messebenen sind somit durch die Messbereiche 2, 5 und 8 für die obere Messebene sowie 3, 6 und 9 für die untere gegeben.

Die Implementation der abstrahierten Interaktion mit dem Sensor ist in der Datei tmf8821.cpp realisiert, die entsprechende Schnittstelle dazu durch die Header Datei tmf8821.hpp gegeben. Durch die dort definierte Methode init wird der Sensor konfiguriert. Das bedeutet, die SPAD Map wird auf die vordefinierte SPAD Map mit der ID 6 festgelegt und der zeitliche Abstand zwischen zwei Messungen wird auf 50 ms gesetzt. Dieser Wert hat sich aus Testläufen mit unterschiedlichen Werten ergeben. Das Ziel ist diesen Wert so gering zu halten wie möglich, damit ein eingeworfener Gegenstand auf jeden Fall detektiert wird und nicht genau zwischen zwei Messungen eingeworfen werden kann. Die durchgeführten Tests haben ergeben, dass Messungen mit einem zeitlichen Abstand von unter 50 ms jedoch ungenauere Messwerte hervorbrachten. Die init Methode sollte typischerweise in der Setup Methode der main Klasse aufgerufen werden.

Die Methode start lässt den Sensor eine einzelne Messung durchführen und interpretiert zugleich das Ergebnis. Dazu werden die vom Sensor zurückgelieferten Messwert zu einem Status ausgewertet, welcher in der aktuellen Instanz des Sensorobjekts gespeichert wird. Dabei kann das Messergebnis einem von drei möglichen Status zugeordnet werden. Die Status und ihre Bedeutungen werden in folgender Tabelle erläutert.

NONE	In keiner der beiden auszuwertenden Messreihen wird ein Objekt detektiert.
INCOMING	In der oberen Messebene wurde ein Objekt detektiert.
DETECTED	Der vorherige Zustand ist INCOMING und in der unteren Messebene wird ein Objekt detektiert, während in der oberen Messebene keines mehr detektiert wird. Dieser Zustand kann nicht mehr durch weitere Messungen geändert werden, sondern wird beim Auslesen des Sensorzustandes zurückgesetzt.

Um den Zustand des Sensors auszulesen, kann die Methode getState aufgerufen werden. Diese gibt immer den aktuell gespeicherten Zustand zurück. Ein Parameter vom Typ bool gibt an, ob der Zustand auf NONE zurückgesetzt werden soll, wenn der aktuelle Zustand DETECTED ist. Diese Option dient lediglich Debug Zwecken und im Produktivsystem sollte die Methode grundsätzlich nur mit dem bool-Wert true aufgerufen werden, da ein Aufruf der getState Methode mit einem Parameterwert von true die einzige Möglichkeit

ist, den DETECTED Zustand zurückzusetzen.

Während des Verlaufs der technischen Umsetzung wurde deutlich, dass die Bedienung des Einwurfsensors durch den Arduino nur dann in entsprechend schneller Taktung erfolgen kann, wenn der Arduino keine weiteren Aufgaben erfüllen muss. Somit musste für die Steuerung und Auswertung des Einwurfsensors ein eigenständiger Arduino, im folgenden Sensor-Arduino genannt, eingesetzt werden, welcher über zwei digitale Leitungen mit einem zweiten Arduino, im folgenden Haupt-Arduino genannt, kommuniziert, der die restlichen Funktionen des Gerippten steuert. Eine der Leitungen wird vom Sensor-Arduino auf HIGH gesetzt, sobald der Status DETECTED ausgelesen wird. Wenn der Haupt-Arduino diesen Status erkannt hat, setzt dieser die zweite Leitung für eine Iteration seiner main loop auf HIGH, um die Meldung zu bestätigen. Diese Bestätigung wird wiederum vom Sensor-Arduino gelesen, woraufhin dieser die erste Leitung wieder auf LOW setzt. Somit dient der Sensor-Arduino dem Einwurfsensor als Steuergerät, welches dem Haupt-Arduino lediglich mitteilt, wann ein Einwurf detektiert wurde.

Während der letzten Praxistests, im Endzustand der Modellbauphase wurde deutlich, dass die Messgeschwindigkeit für die Einwurfdetektion noch nicht ausreichend ist, um Einwürfe zuverlässig zu detektieren. Ein großer Teil der Einwürfe wurde nicht erkannt. Jedoch wurden 'falsche Einwürfe', bei denen beispielsweise die Hand hineingesteckt und wieder herausgezogen wurde, zuverlässig als nicht vollständige Einwürfe erkannt. Da es für die Vorstellung des Prototypen von Vorteil war eine zuverlässige Einwurfdetektion zu präsentieren, statt des zuverlässigen Ausschließens von nicht vollständigen Einwürfen wurde, die Bedingung für das Setzen des Detektionssignals angepasst, sodass jedes Mal ein Detektionssignal gesetzt wird, sobald der Sensorstatus nicht mehr NONE entspricht, was der Funktionsweise einer einzelnen Lichtschranke entspricht. Durch weitere Analyse des Quellcodes im Anschluss an die Präsentationsveranstaltung fiel auf, dass in der Methode start, zu sehen in Listing 1, alle Messwerte mindestens doppelt und maximal vierfach durchlaufen werden, wenn der Sensorstatus vor der Messung den Wert INCOMING aufweist, da jeder Aufruf von checkMiddleRow oder checkBottomRow jeweils einmal über alle Messwerte der letzten Messung iteriert.

Die Auswertung der Messwerte zum neuen Status kann jedoch auch mit einer einzelnen Iteration über die Messdaten realisiert werden, indem die Methoden checkMiddleRow und checkBottomRow zu einer Methode zusammengefasst werden, die einmal durch die Messwerte iteriert und dabei zurückgibt in welchen Messebenen etwas detektiert oder eben nichts detektiert wurde. Diese Verbesserung bietet somit eine Reduktion der Laufzeit, welche möglicherweise ausreichend ist, um die Einwurfdetektion in ihrem geplanten Zustand funktionsfähig zu implementieren.

```
void TMF8821::start(void)
{
    sensor.startMeasuring(results);
    IntakeState curr = state;
    switch (curr)
    {
        case NONE:
            if (checkMiddleRow())
            {
                state = INCOMING;
            }
            break;
        case INCOMING:
            if (checkBottomRow() && !checkMiddleRow())
            {
                state = DETECTED;
            }
            else if (!checkBottomRow() && !checkMiddleRow())
            {
                state = NONE;
            }
            break;
        default:
            break;
    }
}
```

Listing 1: start Methode aus tmf8821.cpp ohne Kommentare und Logging Ausgaben

5.6 Flaschenerkennung (Working Title)

5.6.1 Diskussion

5.6.2 Implementation

5.7 Kommunikation zwischen Endgeräten

5.7.1 Diskussion

5.7.2 Implementation

5.8 Füllstandsmessung

5.8.1 Diskussion

5.8.2 Implementation

5.9 Aufgabenbereiche

5.10 Stückliste

siehe Technische Daten.

5.11 Zusammenführung von Modell und Technik

Der ganze technische Krams und ein bisschen Modellbau

6 Ergebnis

6.1 Der Gerippte

Vorstellung des realisierten Prototyps und Beschreibung seines Funktionsumfangs

6.2 Abschlussveranstaltung

7 Fazit

7.1 Was lief gut

7.2 Was haben wir gelernt

Herausforderung bei der Realisierung Herausforderungen im Projektmanagement (Zeiten, Inhalte, Aufgaben) Herausforderungen in der technischen Umsetzung (Elektronik, Mechanik, Programmierung) Herausforderungen in der interdisziplinären Zusammenarbeit

8 Bewertung

9 Zukünftige Entwicklungsmöglichkeiten

Zusammenfassende Bewertung und Blick in zukünftige

Abbildungsverzeichnis

[1] Quelle: Design Thinking Workshop CrossInnovationClass