Rahmenstruktur für das Programm

Wenn die Daten von den einzelnen Sensoren erfasst werden können und auch die Kommunikation mit dem Raspberry Pi funktioniert müssen wir das Programm der Kollisionsabwehr selber schreiben. Davon handelt diese Ausführung.

Sensorfusion von Radar und Lidar:

Derzeit können wir die Daten vom Lidar und Radar einzeln erfassen. Das Lidar erfasst in 361 Bereichen nur statische Ziele als Winkel und Abstand. Das Radar erfasst auch bewegte Ziele direkt vor dem Boot mit Abstand und Geschwindigkeit. Damit bei jedem Programmdurchlauf der richtige Punkt gefunden werden kann, müssen die Ergebnisse beider Sensoren kombiniert werden.

- **Daten von Lidar und Radar müssen in einem Array vorliegen.**

Durch das Lidar wird der Bereich vor dem Boot in 361 Bereiche aufgeteilt. Jeder Bereich, wo ein Objekt erfasst wird (ein Abstand von weniger als 80m zurückgegeben wird) wird gesperrt, wenn der Abstand kleiner ist, als der Abstand zwischen Boot und Punkt (ggf. mit Sicherheitsabstand. Damit kein ‚zu nahes' Vorbeifahren auftritt, wird auch jeweils ein Bereich links und rechts neben jedem gesperrten Bereich ebenfalls gesperrt.

- **Bei gemessenem Abstand, wird der entsprechende Bereich gesperrt.**
- **Jeweils die Bereiche links und rechts daneben werden auch gesperrt.**

Beim Radar spielt der Datensatz in jedem Durchlauf nur dann eine Rolle, wenn sich das Objekt auf das Boot zubewegt (relative Geschwindigkeit zum Boot ist negativ). In Abhängigkeit des vom Radar angegebenen Abstandes werden dann verschieden viele Bereiche gesperrt (im Programm wird das mit ‚angularrange' beschrieben).

- **Nur sich auf das Boot zubewegende Objekte spielen eine Rolle.**
- **In Abhängigkeit des Radar-Abstands werden unterschiedlich viele Bereiche gesperrt.**

Das Radar kann weiter schauen, als das Lidar. Der Abstand eines Objektes, ab welchem Bereiche gesperrt werden müssen hängt auch von der Geschwindigkeit des Objektes ab. Um auszurechnen, ob ein Objekt wichtig ist, muss ermittelt werden: Wenn sich das Objekt mit seiner gemessenen Geschwindigkeit weiter auf uns zu bewegt, wird es in fünf Sekunden (Richtwert) näher sein, als der Abstand des Zielpunktes?

- **Aus der Objektgeschwindigkeit wird ein neuer Abstandswert berechnet.**
- **Die berechnete Abstandswert bestimmt, ob Bereiche gesperrt werden müssen.**

Danach werden die Ergebnisse vom Radar mit denen des Lidar kombiniert. D.h., all jene Bereich, die wegen der Radardaten noch gesperrt werden müssen, werden gesperrt. Es ergibt sich mit den Sperrbereichen des Lidar, denen des Radars und den Sicherheitsbereichen ein Array aus Booles, durch welche Bereiche gefahren werden darf und durch welche nicht.

- **Am Ende verbinden sich alle Datenteile zu einem Booleschen Array.**

Position der Bahnpunkte relativ zum Boot:

Aufgrund der Punkteliste für die Bahnplanung und der GPS-Daten, sind die komplette Bahn und die derzeitige Position des Bootes bekannt. Zunächst wird der erste Punkt der Liste angefahren. Ist man diesem nahe genug, wird er als erreicht betrachtet und der nächste Punkt in der Liste wird ausgewählt. Dadurch ist in jedem Programmdurchlauf ein Zielpunkt vorhanden. Außerdem sind der Kurswinkel des Bootes zur Nordrichtung und der Winkel des Zielpunktes zum Boot bekannt.

- **Die derzeitige Position und die Zielposition müssen bekannt sein.**
- **Der Kurswinkel und der relative Winkel vom Boot zum Punkt müssen bekannt sein.**

Das Winkelsystem des Bootes stimmt mit der Orientierung des Kompasses nicht überein und ändert sich während der Fahrt. Während jedes Durchlaufes muss daher ermittelt werden, welcher der 361 Bereiche (immer nur einer) in Richtung des Zieles zeigt. Dazu müssen die zwei vorhandenen Winkel miteinander verrechnet werden.

- **Aus den Winkeln ergibt sich die Nummer eines Bereiches, wo der Zielpunkt ist.**

Korrekturpunkt berechnen:

Sind die Daten der Sensoren miteinander fusioniert und ist auch der Bereich bekannt, welcher in Richtung des Zielpunktes zeigt, kann festgestellt werden, ob ein alternativer Punkt zu ermitteln ist. Dabei geht aus der Sensorfusion hervor, dass jeder Bereich gesperrt wird, bei dem ein Abstand kleiner dem Zielpunktabstand ermittelt wurde. An der Stelle im Programm, wo es zu einer Korrektur des Zielpunktes kommt, ist also schon absolut bekannt, welche Bereiche endgültig gesperrt werden. Um herauszubekommen, ob eine Korrektur erforderlich ist, muss man vergleichen, ob im booleschen Sensorarray (siehe Sensorfusion) der Wert ‚False' für den Bereich angegeben wurde, der auf den Zielpunkt zeigt.

- **Aus der Sensorfusion gehen alle gesperrten Bereiche hervor.**
- **Aus dem Zielpunktbereich und dem booleschen Sensorarray ergibt sich die Notwendigkeit einer Korrektur.**

Ist der direkte Weg zum Zielpunkt gesperrt, wird ein Korrekturpunkt ermittelt, welcher von den freien Bereichen neben dem Sperrbereich abhängt. Jeweils links und rechts neben dem gesperrten Bereich liegen zwei Freibereiche. Der Korrekturpunkt soll jeweils im größeren der beiden Bereiche liegen. Es kann sein, dass es nur auf einer Seite einen freien Bereich, wobei dann der einzig freie gewählt wird. Gibt es keinen freien Bereich muss ein Wendemanöver durchgeführt werden.

- **Der Korrekturpunkt liegt im größeren der beiden freien Bereiche links und rechts.**
- **Gibt es nur einen freien Bereich, liegt der Punkt dort.**
- **Gibt es keinen freien Bereich, gibt es ein Wendemanöver.**

Der Korrekturpunkt im Falle eines freien Bereiches wird so nahe, wie möglich an den bisherigen Zielpunkt gesetzt. D.h. er wird in den ersten freien Bereich gelegt, welcher gefunden wird. Der Abstand des Punktes wird auf die maximale Reichweite des Lidar gesetzt.

Anschließend muss der Korrekturpunkt anhand des Kurswinkels in einen absoluten Punkt in GPS-Datenformat zurückgewandelt werden, welcher an den Regler gesendet wird.

- **Der neue Korrekturpunkt muss in einen passenden Bereich gelegt werden.**
- **Der Abstand des Punktes wird auf die maximale Reichweite des Lidar gesetzt. Wenn der Abstand ohnehin kleiner war, dann bleibt er gleich.**

Eine kritische Situation beschreibt einen Zustand, bei dem Hindernisse so nah am Boot sind, dass angehalten werden muss. Diese wird bereits bei der Sensorfusion ermittelt. Kommt es also zu einer Korrekturnotwendigkeit, wo alle Bereiche gesperrt sind, ist ein Wendemanöver in jedem Fall möglich. Das sollen die Sicherheitsabstände gewährleisten. In diesem Fall soll der Korrekturpunkt hinter das Boot in einen Abstand von z.B. fünf Metern (Richtwert) gelegt werden. Ein vollständiges Wendemanöver braucht dann noch weitere Schritte, die erstmal nicht weiter betrachtet werden!

- **Ist ein Wendemanöver nötig, liegt der Korrekturpunkt hinter dem Boot.**

Senden der Korrekturpunkte

Für die Bewegung des Bootes ist der Regler zuständig. Die Kollisionsabwehr stellt nur die Punkte zu Verfügung, zu welchen der Regler fahren soll. Da der Regler diese Punkte vom Bussystem hernimmt, muss die Kollisionsabwehr diese auch auf den Bus schreiben.

0x100	8	Länge, Breite	Führungsgröße Punkt A (4 Byte Länge + 4 Byte Breite)	Q9.17
0x108	8	Länge, Breite	Führungsgröße Punkt B (4 Byte Länge + 4 Byte Breite)	Q9.17

Dabei gibt es zwei Punkte. Der Führungspunkt A gibt die derzeitige Position des Bootes an. Der Führungspunkt B gibt die Zielposition an. Wenn das Programm zum ersten mal abläuft, muss der Führungspunkt A in den Bus geschrieben werden, damit der Regler seine Position kennt. Auch der Führungspunkt B muss auf den Bus geschrieben werden, damit der Regler weiß, wohin er fahren soll. Das ist aber nur beim ersten Ablaufen des Programmes nötig.

Bei allen folgenden Abläufen muss nur der Führungspunkt B übertragen werden. Immer wenn dieser sich ändert, z.B. durch eine Korrektur oder weil der Zielpunkt erreicht wurde und man einen neuen Punkt anfährt, muss man diesen aktualisieren. Der aktuelle Punkt, der in unserem Programm ermittelt wurde, wird dann an den Regler als Führungspunkt gesendet.

Aufgabe ist es, zu realisieren, dass im ersten Ablauf beide Führungspunkte A und B gesendet werden. In allen folgenden Abläufen muss dann nur der Führungspunkt B aktualisiert werden, wenn wir ihn in unserem Programm ändern.

Main-Programm erstellen

Der gesamte Programmablauf benötigt eine Rahmenstruktur, welche die Funktionen aufruft und miteinander verknüpft. Das soll im Main-Programm passieren.

Zunächst sollen bei jedem Durchlauf die Daten aller Sensoren ermittelt werden, damit diese für das weitere Programm zur Verfügung stehen. Danach folgen die Prozesse der

Sensorfusion, der Korrekturpunktermittlung und des Versendens der Korrekturdaten in dieser Reihenfolge. Damit das reibungslos ablaufen kann, sind weitere Einsätze nötig.

Ggf. sollten globale Variablen eingeführt werden. Beispielsweise braucht das Senden der Korrekturdaten einen Zähler dafür, wie viele Programmabläufe bereits durchgelaufen sind. Im Main-Programm wird auch festgestellt, ob das Senden und Empfangen von Daten erfolgreich durchgeführt wurde. Wenn ein Fehler aufgetreten ist, muss das Programm pausiert und das Empfangen der Daten weiderholt werden, bis es funktioniert hat. Bzw. muss auch das Senden der Daten bei einem Fehlschlag wiederholt werden. Eventuell sind hier Fehlerstatusvariablen global sinnvoll, die zur Sicherheit andere Programmteile an der (womöglich falschen) Ausführung hindern.

Das Main-Programm soll auch den Wechsel zwischen Punkten in der Bahnpunkteliste definieren. Bei jedem Durchlauf muss geprüft werden, an welcher Stelle in der Liste man sich befindet und dafür eine Indikatorvariable gesetzt werden.

Wenn bestimmte Programmteile, hauptsächlich die Sensorfusion, eine Anhalten des Bootes erfordern, muss im Main-Programm der Stoppbefehl erfolgen. In diesem Fall wird die Zielposition mit der aktuellen Position gleichgesetzt, damit ein Anhalten erfolgt.

Eine weitere Aufgabe des Main-Programms soll es laut letztem Meeting sein, die Adressen der einzelnen Busteilnehmer aus einer ‚json-Datei' zu importieren, so dass sie von Herrn Franke effizienter angepasst werden können. Da das womöglich aufwändiger wird, behalten wir uns das als ‚Extra' vor, wenn dafür noch Zeit bleibt.