Die Kompassverwendung in Gebäuden

Einführung

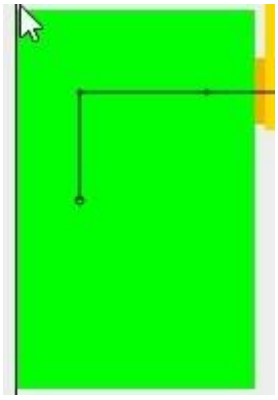
Ich verwende für die Steuerung meines Roboters einen elektronischen Kompass (CMPS11). Das Bauteil wird über einen I2C Bus an einem Arduino ausgelesen. Bei ruhig stehendem Roboter wackelt der Kompasswert im Bereich von +- 0.5 Grad hin und her. Diese Abweichung entsteht aus natürlichen Schwankungen des Erdmagnetfeldes. Nach der Eichung des Kompasses auf den lokalen Standort (Auslieferstandort ist England) soll er angeblich 0.4% genau sein. Die Auflösung der Werte beträgt 0.1 Grad. Der Kompass wird über einen Digitalausgang nach dem Start des mit 5V versorgt. Nach 2 Sekunden warten wird der I2C Bus aktiviert (wire.begin()).

Testergebnisse

Bei meinen Tests bin ich auf verschiedene physikalische Phänomene gestossen. Die einfache Anzeige der Richtung scheint zunächst gut zu klappen. Wenn man dann die Werte nebeneinander hält fällt auf, dass die Abweichung von z.B. Nord nach West nicht 90 Grad beträgt sondern wesentlich davon abweicht z.B. 80 Grad oder 110 Grad. Geht man mit dem Kompass durch einen Raum wird man feststellen, dass diese Abweichung innerhalb des Raumes stark schwankt, was natürlich genauso für unterschiedliche Räume gilt. Bereits bei 1.5 m Abstand zwischen den Messungen haben Pos1, Pos2 und Pos3 abweichende Werte bis zu 43 Grad, was die Tabelle in Beispiel 1deutlich zeigt.

Beispiel 1:

Im Office meines Hauses gewonnene Messergebnisse an 3 unterschiedlichen Positionen. Position 1 ist die unterste.



	A	B	C	D	E	F	G
1					Office		
2	Relative direction	Angle	Deviation	Bearing	Pos1	Pos2	Pos3
3	W	270	110	20	42	36	20
4	N	0	115	106	106	100	96
5	E	90	135	216	240	228	197
6	S	180	158	329	324	329	307
7	Mean value		129				

Der Raum wird gemäss Kartendarstellung mit den relativen Himmelsrichtungen (W/N/E/S) bezeichnet. Der Roboter steht also in der relativen Richtung. Bearing bezeichnet den abgelesenen Kompasswert des Roboters. Deviation ist die Abweichung einerseits des Gebäudes von der Nordrichtung, enthält aber auch den Anzeigefehler durch Gebäudekonstruktion, den Roboter selbst und die Missweisung. Der Mittelwert der Deviation sollte etwa die Abweichung des Gebäudes von der magnetischen Nordrichtung sein, bezogen auf die relative Richtung N.

Diese Anzeigefehler ergeben sich aus Armierung, Stahlträger, Metallzargen in den Türen, Wasserleitungen, Heizkörper, stromdurchflossene Leitungen aber auch Stahlteile an Möbeln oder

Geräte mit magnetisch leitfähigem Gehäuse. Das Magnetfeld wird sozusagen im Gebäude in jedem Raum anders verzerrt.

Was sagt die Wissenschaft

Finnische Wissenschaftler machen aus der Not eine Tugend und entwickeln eine App um aus der Magnetfeldverzerrung des Raumes, der typisch und in keinem Raum eines Gebäudes gleich ist, den Standort eines Handys im Gebäude zu bestimmen.

http://www.golem.de/news/indooratlas-mit-dem-smartphone-kompass-in-gebaeuden-zurechtfinden-1207-93070.html

Aus der Navigation bei Flugzeugen und Schiffen ist bekannt, dass das Fahrzeug selbst den Magnetkompass ablenkt. Das tut er nicht etwa mit einem konstanten Wert sondern in jeder Richtung mit einem anderen. Diese Abweichung entsteht durch die Fahrzeugkonstruktion (Rumpfkomponenten aus Stahl, Motor, Ausrüstung, Werkzeug). In der Praxis wird man für das Fahrzeug eine sogenannte Deviationstabelle erstellen und diese Abweichung dokumentieren. Bei der Kursberechnung fliesst diese Abweichung zusammen mit der Missweisung in das Ergebnis ein.

https://www.segelrevier.ch/tutorial/hochseeausweis-2-mercator-deviation-missweisung-kurse/

Was machen wir nun daraus?

Da die Positionspunkte für den Roboter dynamisch errechnet werden, ist eine manuelle Bestückung der Positionspunkte mit einer Deviationstabelle sehr aufwändig. Deshalb habe ich untersucht ob es genügt eine solche Tabelle pro Raum (Region) zu erstellen.

Die Navigation meines Roboters ist auf rechtwinklige Bewegungen ausgelegt. Die Navigationspunkte des Ziels liegen also immer in einer der relativen Kompassrichtungen nach Karte. Wir markieren uns nun die Messergebnisse einer Region gelb, falls sie überhaupt benötigt werden, um einen benachbarten Punkt zu erreichen.

	A	B	C	D	E	F	G
1					Office		
2	Relative direction	Angle	Deviation	Bearing	Pos1	Pos2	Pos3
3	W	270	110	20	42	36	20
4	N	0	115	106	106	100	96
5	E	90	135	216	240	228	197
6	S	180	158	329	324	329	307
7	Mean value		129				

Jetzt fällt auf dass die Winkel innerhalb der Region Office bis auf die beiden in Richtung E nur von einer einzigen Position benötigt werden. Das ist praktisch, denn das heisst ich benötige die übrigen Werte nicht, sondern kann getrost für den ganzen Raum einen einzigen Wert pro Richtung übernehmen.

Die beiden Werte E für Pos2 und Pos3 schauen wir noch genauer an. Die Abweichung ist mit 31 Grad enorm. Eine einfache Mittelwertbildung wäre das Naheliegende. Aber Vorsicht, der Abstand zwischen Pos2 und Pos3 beträgt 155cm und zwischen Pos3 und Pos4 100cm. Wir müssen also die Winkel gewichten mit (228*155+197*100)/255 = 216. Dieser Wert 216 Grad führt in der Bewegung von Pos2 nach Pos3 zu einer Richtungsverfälschung. In der Bewegung von Pos3 nach Pos4 wird diese

Richtungsverfälschung wieder korrigiert. Wir müssen natürlich dafür sorgen, dass der Roboter den nötigen Raum für die Abweichung hat.

Pos2 nach Pos3:

-tan(Winkeldifferenz)*155cm

-tan(12)*155cm = -33cm

Pos3 nach Pos4:

tan(Winkeldifferenz)*100cm

tan(19)*100cm = 34 cm

Die 1 cm Abweichung ist den Rundungsfehlern der Winkel geschuldet.

Im Beispiel ist die Tür Durchgangsbreite 75cm die Roboterbreite beträgt <24 cm. Der Fahrweg ist in Türmitte bei 37cm. An der Wand stehen Möbel die den ganzen restlichen Raum (60cm) ausfüllen.

Unser Spielraum muss grösser sein als die Abweichung + halbe Roboterbreite

33cm + 12cm = 45cm der freie Raum beträgt nur 37cm und genügt damit nicht

Wir korrigieren den Deviationswinkel von 216 auf 219 Grad.

Pos2 nach Pos3:

-tan(Winkeldifferenz)*155cm

-tan(9)*155cm = -24.5cm

Pos3 nach Pos4:

tan(Winkeldifferenz)*100cm

tan(22)*100cm = 40 cm

24.5cm + 12cm = 36.5cm

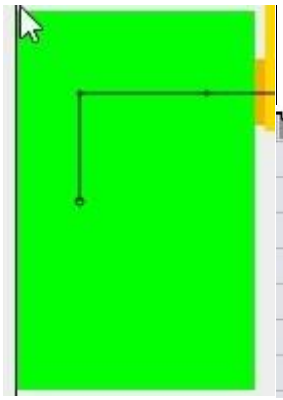
Dieser Weg ist leider auch nicht so ganz was wir brauchen, denn er führt zu einer Abweichung der erreichten Zielposition um 15.5 cm. Der Fehler würde uns später Ärger machen wenn der Roboter zu früh abbiegt.

Wir probieren als nächstes den Algorithmus für das Fahren zu manipulieren. Drehen soll nur durchgeführt werden, wenn die Richtung wirklich geändert werden muss. Bei Geradeausfahrt wird an einer neuen Position nicht gedreht. Da die Kompassrichtung nur währen des Drehens angeschaut wird, ersetzen wir den Deviationswert im Office Beispiel für E durch die richtigen 228 Grad! Das hilft uns vielleicht noch in ähnlichen Situationen bei anderen Regionen.

Nun messen wir, wie die gefundene Lösung für die übrigen Regionen funktioniert!

Deviation Office

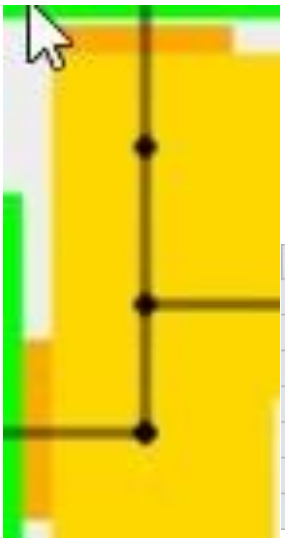
Die Zählung der Positionen beginnt links unten mit Pos1. Der Bearingwert für W wird aus Corridor1 übernommen.



	A	B	C	D	E	F	G
1					Office		
2	Relative direction	Angle	Deviation	Bearing	Pos1	Pos2	Pos3
3	W	270	129	39	42	36	20
4	N	0	115	106	106	100	96
5	E	90	147	228	240	228	197
6	S	180	158	329	324	329	307
7	Mean value		137				

Deviation Corridor1

Die Zählung der Positionen beginnt links unten mit Pos4.

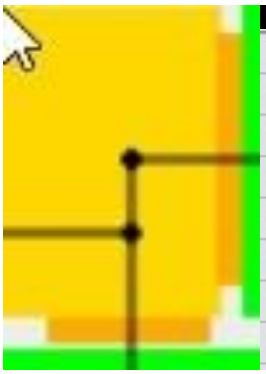


	A	B	C	D	E	F	G
9					Corridor1		
10	Relative direction	Angle	Deviation	Bearing	Pos4	Pos5	Pos6
11	W	270	129	39	39		
12	N	0	118	109	109	96	75
13	E	90	131	212	227	212	
14	S	180	149	320	326	320	301
15	Mean value		132				

Deviation Corridor 4

Die Zählung der Positionen beginnt unten mit Pos10.

Hier haben wir ein deutliches Problem! Der Bearingwert für S weicht für die Positionen 10 und 11 leider um 33 Grad ab und das obwohl die beiden Punkte nur 32cm auseinander liegen. Das können wir durch nichts ausgleichen, denn je nachdem von wo der Roboter kommt muss der eine oder der andere Wert angewandt werden.



	A	B	C	D	E	F
33					Corridor4	
34	Relative direction	Angle	Deviation	Bearing	Pos10	Pos11
35	W	270	33	33	33	
36	N	0	78	78	78	
37	E	90	202	202		202
38	S	180	330	348	348	315
39	Mean value		161			

Fazit

Somit haben wir keine Methode gefunden, die es uns erlaubt pro Region und relativer Richtung einen einzigen sinnvollen Wert für die Deviation zu hinterlegen.

Die Verwendung des Kompasses in Gebäuden ist eigentlich nicht wirklich sinnvoll möglich, ausser man definiert für jede Position an der er angewandt werden soll die Deviation!

Da wir uns auf die Positionen und die vier Richtungen beschränken können, rückt das Ziel mit dem Kompass zu arbeiten wieder etwas näher.

Neuer Ansatz

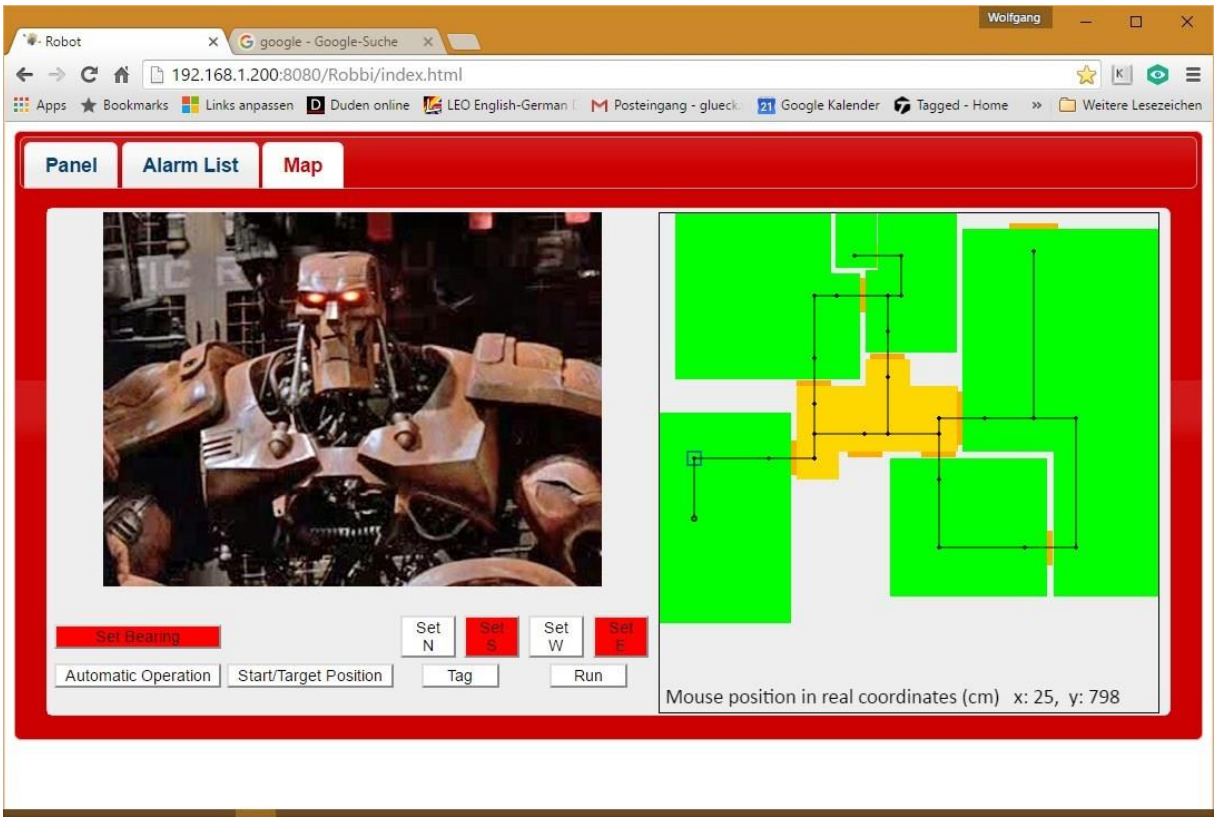
Wir haben die Vektoren (Edges) zwischen den Positionen, an denen derzeit die relativen Zielwinkel für jeden Vektor hinterlegt sind. Wir können zusätzlich die absolute Richtung (bearing) in jedem Vektor speichern.

Wie kommen wir an die Werte?

Ganz klar manuell! Aber wir werden den Roboter dazu verwenden, um uns die Erfassungsarbeit zu erleichtern.

Eine weitere Betriebsart „Set Bearing“ soll das Markieren einer Position erlauben „Robot Position“. Für alle gefundenen Edges mit „fromP“ gleich der markierten Position wird eine Taste aktiviert (rot)„Set N“, „Set S“, „Set W“, „Set E“. Der Roboter muss nun manuell auf die ausgewählte Position in die betreffende Richtung gestellt werden. Nach betätigen der entsprechenden Set Taste wird Bearing als absoluter Winkel in der Edge gespeichert. Die betätigte Taste wird wieder neutral eingefärbt.

Die Daten werden beim Verlassen der Betriebsart in einem File gespeichert. Falls das File existiert werden die Daten beim Programmstart eingelesen.



Der Algorithmus für das Drehen auf Ebene Arduino muss nun von relativer Abbruchbedingung auf eine Abbruchbedingung mit absolutem Winkel umgestellt werden.

Hier sind je nachdem in welchem Quadranten der Start Winkel (startAngle) und der Zielwinkel (turnAngle) liegen unterschiedliche Bedingungen zu beachten:

	A	B	C	D	E	F	G	H	I
1		A1 startAngle [0..90)	A2 startAngle [90..180)	A3 startAngle [180..270)	A4 startAngle [270..360)				
2	T1 turnAngle [0..90)	Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle || angle16 > startAngle &turnedAngle >0	Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle	Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle		T1 ||			T1
3		Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle		Rechtsdrehung ende: angle16>=turnAngle &angle16<startAngle	Rechtsdrehung ende: angle16>=turnAngle &angle16<startAngle		T1&A1 || T1&A2	T1&A3 || T1&A4	
4	T2 turnAngle [90..180)		Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle	Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle	Linksdrehung ende: angle16 <= turnAngle & angle16 < startAngle	T2 ||			
5		Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle	Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle		Rechtsdrehung ende: angle16>=turnAngle &angle16<startAngle		T2&A1 || T2&A2	T2&A3 || T2&A4	
6	T3 turnAngle [180..270)	Linksdrehung ende: angle16 <= turnAngle & angle16 > startAngle		Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle	Linksdrehung ende: angle16 <= turnAngle & angle16 < startAngle		T3&A1 || T3&A2	T3&A3 || T3&A4	
7		Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle	Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle	Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle		T3 ||			
8		Linksdrehung ende: angle16 <= turnAngle & angle16 > startAngle	Linksdrehung ende: angle16 <= turnAngle & angle16 > startAngle		Linksdrehung ende: angle16 <= turnAngle &angle16<startAngle		T4&A1 || T4&A2	T4&A3 || T4&A4	
9	T4 turnAngle [270..360)		Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle	Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle	Rechtsdrehung ende: angle16>=turnAngle &angle16>startAngle || angle16 < startangle & turnedAngle > 0	T4 ||			T4

Bei genauerer Betrachtung kann man die Varianten auf 2 pro Drehrichtung reduzieren. Die reduzierten Varianten sind am rechten Rand farblich unterschieden. Für die Übergänge am Nullpunkt sind die beiden Varianten rot und gelb zusätzlich notwendig.

Ein weiteres Problem stellt der nicht stetige Verlauf der Messung der Kompassrichtung dar. Der Arduino Zyklus kann nicht beliebig kurz sein. Um die Kompassrichtung während des Drehens möglichst schnell abzutasten, unterdrücken wir alle Funktionen die nicht benötigt werden. Mit einem kleinen Toleranzwert von 2 Grad können wir ausserdem den Abschaltzeitpunkt vorverlegen. Damit erreichen wir eine Genauigkeit der Drehrichtung von +- 1 Grad. Dieser Wert bedeutet eine Richtungsabweichung von +- 1.7cm / m.

Selbst kleine Abweichungen, deren Ursache in den Messfehlern und in den mechanischen Unzulänglichkeiten des Chassis liegt, können sich auf dem Weg durch die Räume kumulieren. Deshalb sehen wir einen Algorithmus zur Korrektur vor.

Messung der Kursabweichung bei der Türdurchfahrt und ihre Verwendung

Fährt der Roboter durch eine Tür, ändert sich die Distanzmessung oben von ca. 235cm auf 200cm. Wir müssen also im Hauptzyklus (200ms) des Raspberry diesen Wert verfolgen. Wenn die Durchfahrt erkannt wird, schauen wir uns den Seitenabstand der beiden seitlichen US Detektoren an. Die Differenz zwischen linkem und rechtem Detektor entspricht der zweifachen Richtungsabweichung.

Diese Abweichung wird nun als Korrekturwert gespeichert und beim nächsten Positionspunkt auf zwei verschiedene Arten als Korrektur verwendet:

1. Falls der Roboter geradeaus fährt wird aus der Abweichung/2 der Korrekturwinkel bis zum nächsten Positionspunkt berechnet:

   D = Distanz zum nächsten Positionspunkt (aus edge)

   A = Abweichung/2

   K = Korrekturwinkel

   K = arctan(D/A)

   Je nach Vorzeichen der Abweichung muss der Korrekturwinkel zu Bearing addiert oder subtrahiert werden. Ist die Abweichung zwischen dem errechneten Wert und der aktuellen Kompassrichtung >=2 Grad wird eine Korrekturdrehung ausgeführt.

2. Falls der Roboter nach links oder rechts abbiegt wird die Abweichung in die Roboterposition hineingerechnet und beim weiteren Fahren bis zum nächsten Punkt als Verkürzung bzw. Verlängerung der Strecke mit berücksichtigt.

Erwartete und tatsächliche Türdurchfahrt

Die Strecke zwischen den Positionen beidseits der Türen ist bekannt. Davon kann also auch die Strecke abgeleitet werden nach der eine Türdurchfahrt erwartet wird. Die Differenz zwischen der errechneten Strecke und der tatsächlich gemessenen Strecke dient uns wiederum als Korrekturwert

für die restlich zu fahrende Strecke. Falls aus der Kursabweichung ebenfalls ein Korrekturwert existiert wird dieser nun durch den neu gemessenen ersetzt.