VOR Sender und Empfänger - Dokumentation

Glossar / Begriffsdefinitionen

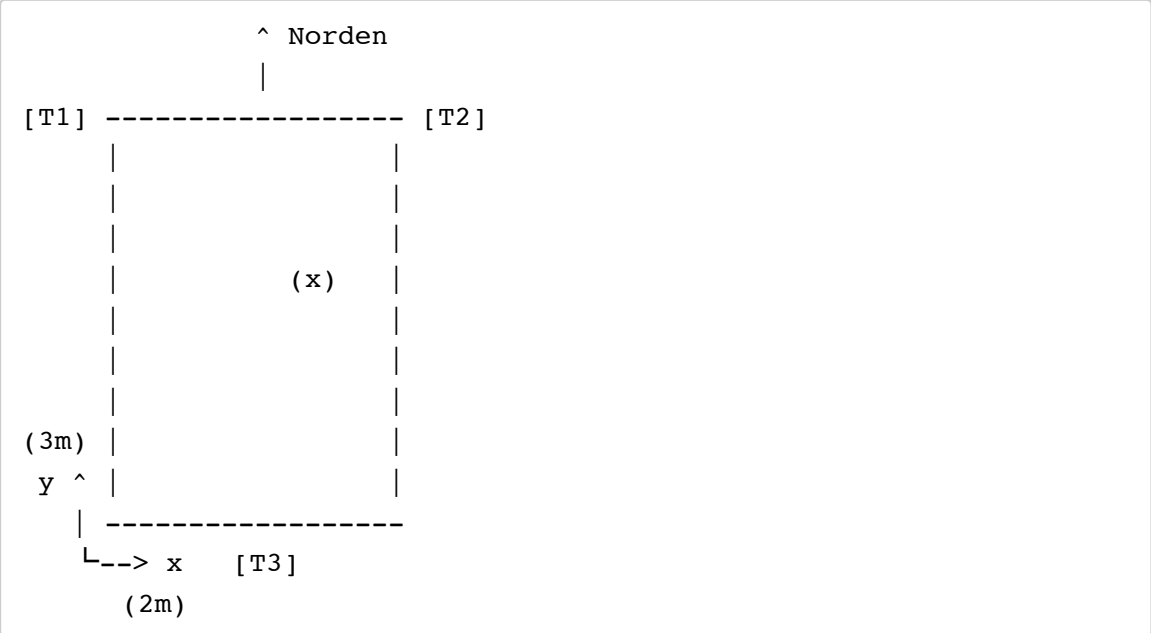
- VOR Quartal (quarter): 90° Segment des VOR
- VOR-Master: VOR-Sender, der den Nordimpuls vorgibt
- VOR-Slave(s): VOR-Sender, die dem Master folgen
- Nordimpuls: Funksignal, das den Beginn einer VOR-Umdrehung anzeigt
- Norden: bezeichnet nicht magnetisch oder geographisch Nord, sonden ist im Spielfeld definiert!
- IR-Strahl (IR beam): Umlaufender Lichtstrahl
- VOR Segmente (segments): Anzahl der umlaufenden Segmente (Anzahl IR-LEDs)

Die VOR-Technologie

Die hier verwendete Technologie zur Positionsbestimmung ist dem sog. VOR aus der Luftfahrt angelehnt. Hierbei werden bekannt Punkte angepeilt, um seine eigene Position berechnen zu können.
Damit nun aber keine Kompasspeilungen vorgenommen werden müssen, gibt es einen Trick: Die Funkfeuer/ Sendetürme senden ein Umlaufendes Signal mit einem relativ kleinen Abstrahlwinkel. In dieser Implementation wird das umlaufende Signal mit IR-LEDs realisiert. Um nun aus diesem umlaufenden Strahl eine Winkelinformation zu erhalten, muss es noch eine Referenz geben. Diese wird durch einen allseitig abgegebenen Nordimpuls realisiert. Nun kann die Zeit zwischen dem Nordimpuls und der Messung den Strahl gemessen werden, die proportional zum Abstrahlwinkel ist. Die Peilung ist also immer zum Empfänger hin.
Da die Positionen der Sendetürme bekannt sind, kann aus mindestens zwei Peilungen eine Positions ermittelt werden.
Damit auch der Fall abgedeckt ist, dass ein Hindernis die Sicht auf einen Sendeturm verdeckt, gibt es drei Sendetürme.

Positionsberechnung

Spielfeld und Koordinatensystem



Zuordnung Winkel - Sendeturm

Je nach dem, aus welchem Winkel (bezogen auf Norden) der Strahl eintrifft, kann dieser eindeutig zu einem Sendeturm zugeordnet werden.

a) 0°- 90°: Sender 3
b) 90°-180°: Sender 1
c) 180°-270°: Sender 2
d) 270°-360°: Sender 3

Also muss in der Software die aus der Zeit berechnete Winkelangabe dem richtigen Sendetum zugeordnet werden.

Berechnung

Prinzipiell ergibt sich aus der Geometrie erst einmal ein LGS mit drei Gleichungen.

```
Sender 1: g_1(x) = m_1*(x)   + 3   mit: m_1 = tan(90°-α_1)
Sender 2: g_2(x) = m_2*(x-2) + 3   mit: m_2 = tan(90°-α_2)
      ==> g_2(x) = m_2*x - (2*m_2 - 3)
Sender 3: g_3(x) = m_3*(x-1) + 0   mit: m_3 = tan(90°-a_3)
      ==> g_3(x) = m_3*x - (1*m_3)
```

Andererseits kann nicht angenommen werden, dass sich alle drei Strahlen an einem Punkt schneiden, da schon die Anzahl der IR-LEDs nicht für eine hohe Genauigkeit ausreicht.

Daher müssen alle möglichen Schnittpunkte einzeln berechnet werden. Bei drei Sendern also drei Schnittpunkte, wenn ein Sender verdeckt ist, nur zwei Schnittpunkte.

Algorithmus

Zur Berechnung der Position müssen folgende Schritte durchgeführt werden:

- Zeit zwischen Nordimpuls und Empfang des IR-Strahls messen
- aus der Zeit den entsprechenden Winkel ermitteln
- aus dem Winkel (oder der Zeit) auf den Sendeturm zurückschließen
- aus dem Winkel und dem Sendeturm ergibt sich eine Geradengleichung
- mit mind. zwei Geraden lässt sich eine Position errechnen

Ermittelung des Winkels:
Bei einer Umlauffrequenz des IR-Strahls von `frequenz` Hz und einer Timer-Zeit (seit dem Nordimpuls) von `timer` Mikrosekunden ergibt sich ein Winkel `winkel` in Grad nach:

```
winkel = 360 * timer / (1000000/frequenz)
```

Berechnung der Geradenparameter:
Die Geradenparameter ergeben sich aus den oben genannten drei Gleichungen. Allgemein gilt jedoch mit einer Turmposition von `(pos_x | pos_y)` und einem Winkel `alpha` :

```
Bekannt: m (abhängig von alpha)

    g(x) = m*(x - pos_x) + pos_y
==> g(x) = m * x - m * pos_x + pos_y  =  m * x  + (pos_y - m*pos_x)
                                          ^              ^
                                     Steigung     Y-Achsenabschnitt

==> m = tan(90°-alpha)
    b = pos_y - tan(90°-alpha)*pos_x
```

Schnittpunkt zweier Geraden:
Der Schnittpunkt zweier Geraden lässt sich berechnen, wenn die Steigung und der Y-Achsenabschnitt bekannt sind. Die Indizes `_1` und `_2` sollen hier die beiden Geraden unterscheiden.

```
für Geradengleichungen der Form: y(x) = m*x + b

    b_1 - b_2
x = ---------
    m_2 - m_1

    b_1 / m_1  -  b_2 / m_2
y = -----------------------
     1 / m_1   -   1 / m_2
```

Implementierung als C Funktion:

```
void CalcIntersection (float m_1, float m_2, float b_1, float b_2, f
  *p_x = (b_1 - b_2) / (m_2 - m_1);
  *p_y = ((b_1 / m_1) - (b_2 / m_2)) / ((1 / m_1) - (1 / m_2));
```

Mittelung der Werte

Wenn ein Signal von allen drei Türmen empfangen wird, müssen auch drei Schnittpunkte der Geraden berechnet werden. Für die Berechnung des Mittelwertes bzw. -punktes gibt es potenziell zwei Methoden:

- Der Flächenschwerpunkt des Dreiecks
- jeweils der Mittelwert der X- und Y-Kooridnaten der drei Punkte

Wir haben uns für die einfache Methode - Mittelwert errechnen - entschieden. Der Flächenschwerpunkt spiegelt zwar für das ermittelte Dreieck besser den Mittelunkt wieder, hilft jedoch nicht undebingt dabei der realen Position näher zu kommen.

Nachdem nun nur noch ein Punkt zur Verfügung steht, wird dieser noch einmal über 5 Werte gemittelt bevor die Koordinaten an das Fahrzeug übergeben werden. So wird eine Datenrate von 2 Hz erreicht. Weitere Erklärungen dazu im nächsten Kapitel "Programmablauf".

Programmablauf

Um eine vorhersehbare Ausführungszeit zu haben, wird folgender Ablauf verwendet:

- 4 Nordimpuls-Perioden Daten sammeln
- danach eine Nordimpuls-Periode die Daten verarbeiten

So werden Schwankungen in der Ausführungszeit, während die Daten gesammelt werden, vermieden - die Abtastzeit der IR-Signale bleibt also gleich.

Hauptprogramm

- Wenn Nordimpuls
  - dann Timer starten
  - Counter++
- Wenn Counter im Intervall [1, 4]
  - Wenn irgendeine IR-LED empfangen
    - aktuellen Timerwert auslesen
    - Winkel aus Timer berechnen
    - aus dem Winkelbereich den korrekten Turm ermitteln
    - Winkel in die Turminstanz speichern
- Wenn Counter = 5
  - Wenn mind. 2 Türme Werte haben
    - Geradenparameter von der Turminstanz erhalten
    - Schnittpunkt(e) errechnen
  - Werte übertragen/ ausgeben
  - Counter = 0

Werteausgabe

Aufgrund der Nordimpuls-Frequenz von 50 Hz werden auch mit dieser Frequenz neue Daten gesammelt. Zur Glättung werden jeweils 4 Werte (in 5 Zyklen) zusammengefasst. Es verbleibt also eine Datenrate von 10 Hz.
Da eine Datenrate von 2 Hz für das Fahrzeug ausreicht (und spezifiziert wurde), sodass immer noch einmal über 5 Koordinaten-Werte gemittelt wird, bevor diese ausgegeben werden.

Die Serielle Datenausgabe (bzw. Datenübergabe) wurde von dem Team für das Fahrzeug entwickelt und getestet.

C++-Klasse für die VOR-Sender (Türme)

Die Turm-Klasse mittelt alle erhaltenen Winkel, wenn ein neuer Winkel eingegeben wird.
Erst beim Abrauf der Geradenparameter werden diese aus dem gemittelten Winkelwert errechnet. Dann wird auch der Winkel wieder zurückgesetzt

Hardware

Dieses VOR-System besteht aus drei Sendern, und zwei Empfängern. Die Sender sind dabei miteinander gekoppelt und decken die 360° nur ein Mal ab. Es können beliebig weitere Empfänger hinzugefügt werden.

Sender Hardware

Der Sender besteht aus:

- Arduino nano
- 433 MHz Sender
- (ein oder zwei) 8-Bit Schieberegister 74HC595 ??
- (8 oder 16) IR-LEDs mit 40 kHz Modulation
- (8 oder 16) Transistoren für die LEDs
- TODO: weitere Bauelemente auflisten, Schaltplan hinzufügen

Empfänger Hardware

Der Empfänger besteht aus

- Arduino nano
- 433 MHz Empfänger
- (8 oder 16) IR-Empfänger mit 40 kHz Modulation
- TODO: weitere Bauelemente auflisten, Schaltplan hinzufügen

433MHz Strecke

Der Nordimpuls wird über eine 433 MHz Funkstecke übertragen. Diese hat eine Verzögerung zwischen Sendereinganz und Empfängerausgang von 40µs.

Schnittstelle zum Roboter

Dem Roboter werden die Positionsdaten über die Serielle Schnittstelle (UART) des Arduino übergeben.
Das Datenformat ist folgendes:

- Startzeichen `@`
- X-Koordinate (in cm) als 16 Bit unsigned int (zwei Bytes nacheiander)
- Y-Koordinate (in cm) als 16 Bit unsigned int (zwei Bytes nacheiander)

```
Serial.print("@");
Serial.write((char)x>>8); Serial.write((char)x);
Serial.write((char)y>>8); Serial.write((char)y);
```