Name: \_\_\_\_\_\_\_\_\_\_ Klasse: \_\_\_ Datum: \_\_\_\_\_\_\_\_

# Lösungsblatt Aufgabe 1

# Mecanum-Omniwheels-Fahrzeug mit Hinderniserkennung

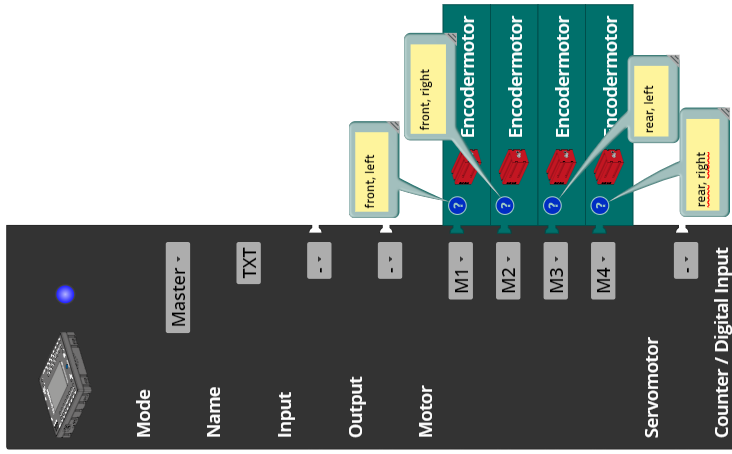
*Die in der ersten Programmieraufgabe entwickelten Steuerungsfunktionen bilden die Grundlage für alle folgenden Aufgabenstellungen.*

*Die Nutzung der Sensoren (Spur­sensor, Ultraschall) ähnelt der in Aufgabe 6 des* *Robotics TXT 4.0 Base Set. Dank der flexibleren Bewegungsmöglichkeiten der Mecanum-Omniwheels ergeben sich zusätzliche, vor allem aber einfachere Lösungen.*

## Programmieraufgaben

**1. Synchroner Antrieb in alle Richtungen**

Konfiguration der Sensoren:



Steuerungsfunktionen (Beispiel):

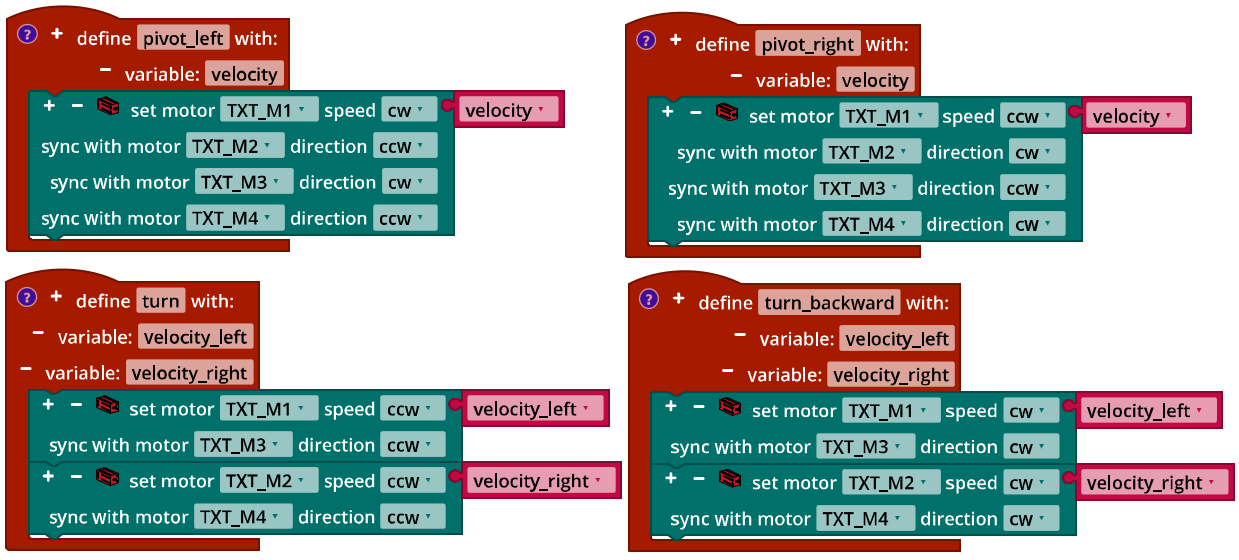
Ein Bild, das Text, Schild, Screenshot, Verkauf enthält.

Automatisch generierte Beschreibung

*Mecanum\_Synchronous\_Driving\_Funktions.ft*

**2. Synchrones Drehen**

Steuerungsfunktionen (Beispiel):

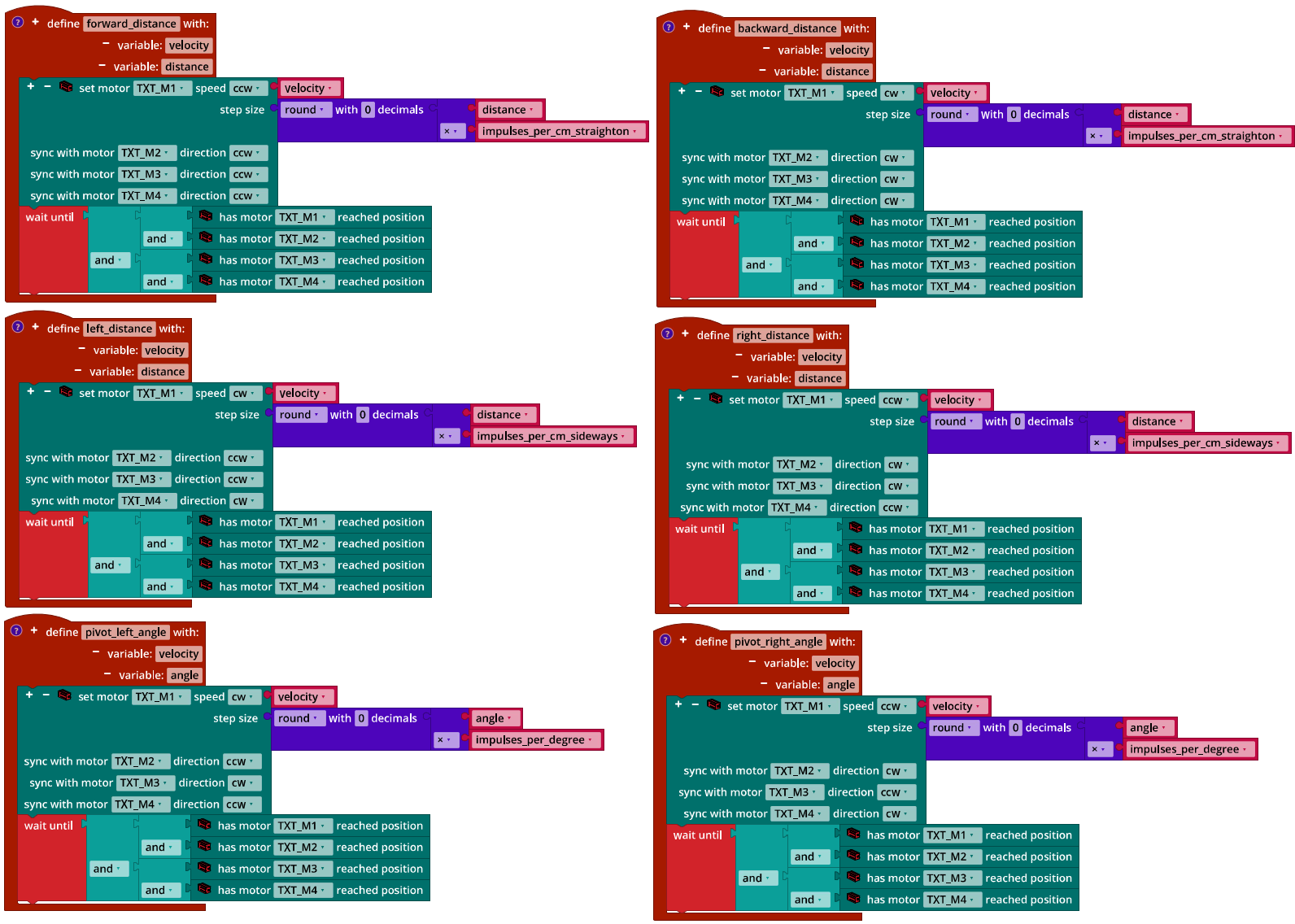


*Mecanum\_Synchronous\_Turning\_Functions.ft*

**3. Synchroner Antrieb mit Distanz-Vorgabe**

3a. Die Umrechnung der Distanz (in cm) in Impulse kann entweder in der Funktion oder beim Aufruf vorgenommen werden.

Programmfunktionen (Beispiel):



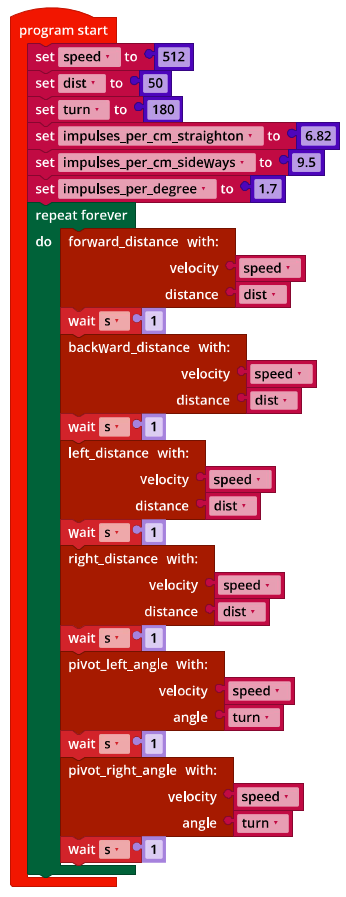
*Mecanum\_Synchronous\_Navigation\_Distance.ft*

3b. Der Umfang eines Mecanum-Omniwheels beträgt ungefähr 19 cm. Daraus lässt sich die Zahl der Impulse je Umdrehung (wie in Aufgabe 6 des Robotics TXT 4.0 Base Set) leicht berechnen: Da die Räder von den Motoren mit einer Übersetzung von 1:2 ins Langsame angetrieben werden sind es etwa  **Impulse/cm**.

Tests über Distanzen von mehreren Metern zeigen, dass der Wert auf etwa **6,82 Impulse/cm** korrigiert werden muss.

Bei der Seitwärtsfahrt und der Drehung helfen für die Bestimmung des Umrechnungs­faktors nur Experimente (siehe nachfolgendes Programm). In Tests konnten der Wert für die Seitwärtsfahrt auf ungefähr **9,5 Impulse/cm** und der Wert für die Drehung auf **1,7 Impulse/Winkelgrad** bestimmt werden.

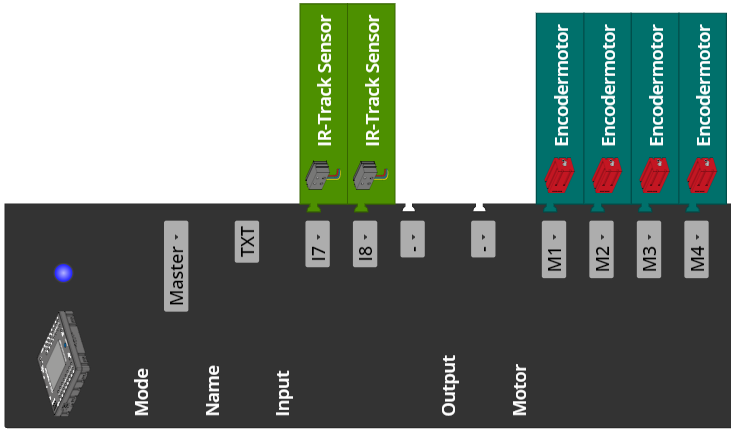
Programm (Beispiel) zum Testen der Umrechnungsfaktoren:



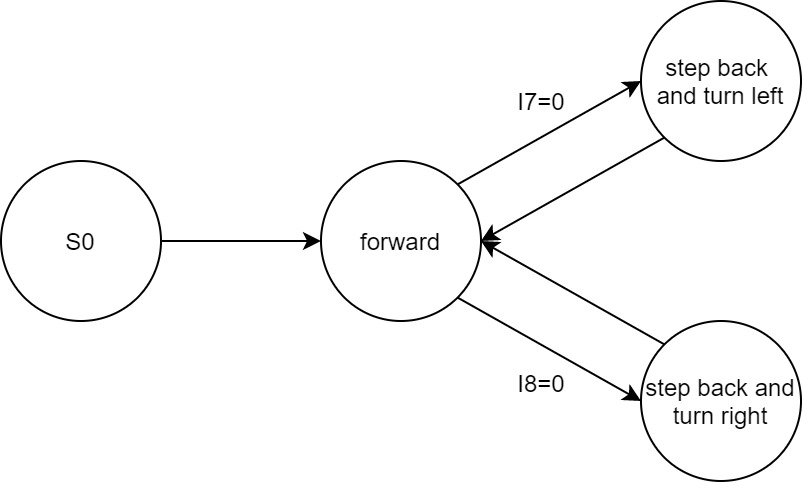
*Mecanum\_Synchronous\_Navigation\_Distance.ft*

**4. Linienerkennung**

Konfiguration der Sensoren und Aktoren:



4a. Zustandsübergangsdiagramm:



*State-Transition\_Diagram\_Mecanum\_with\_IR\_Sensors.drawio*

4b. Programmauszug (Beispiel):

Ein Bild, das Text enthält.

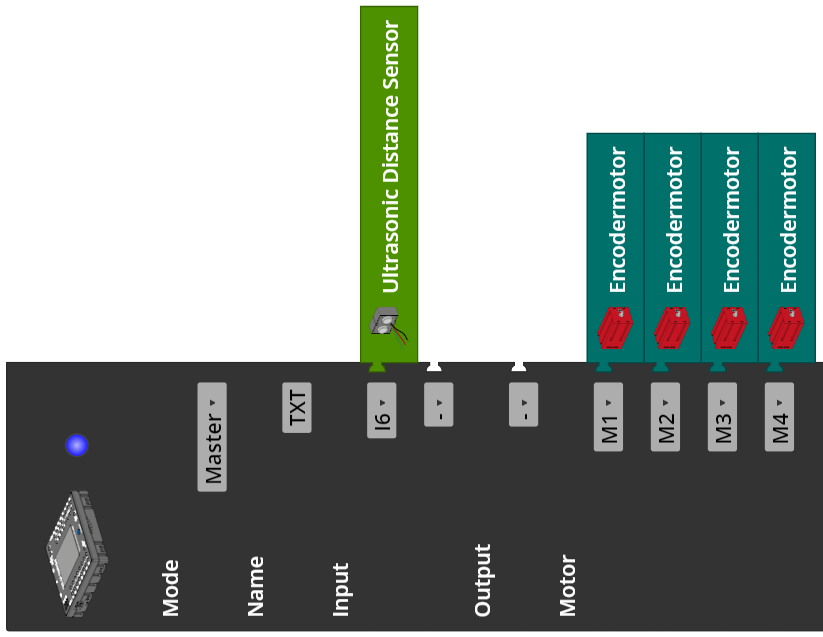
Automatisch generierte Beschreibung

*Mecanum\_Boundary\_Line.ft*

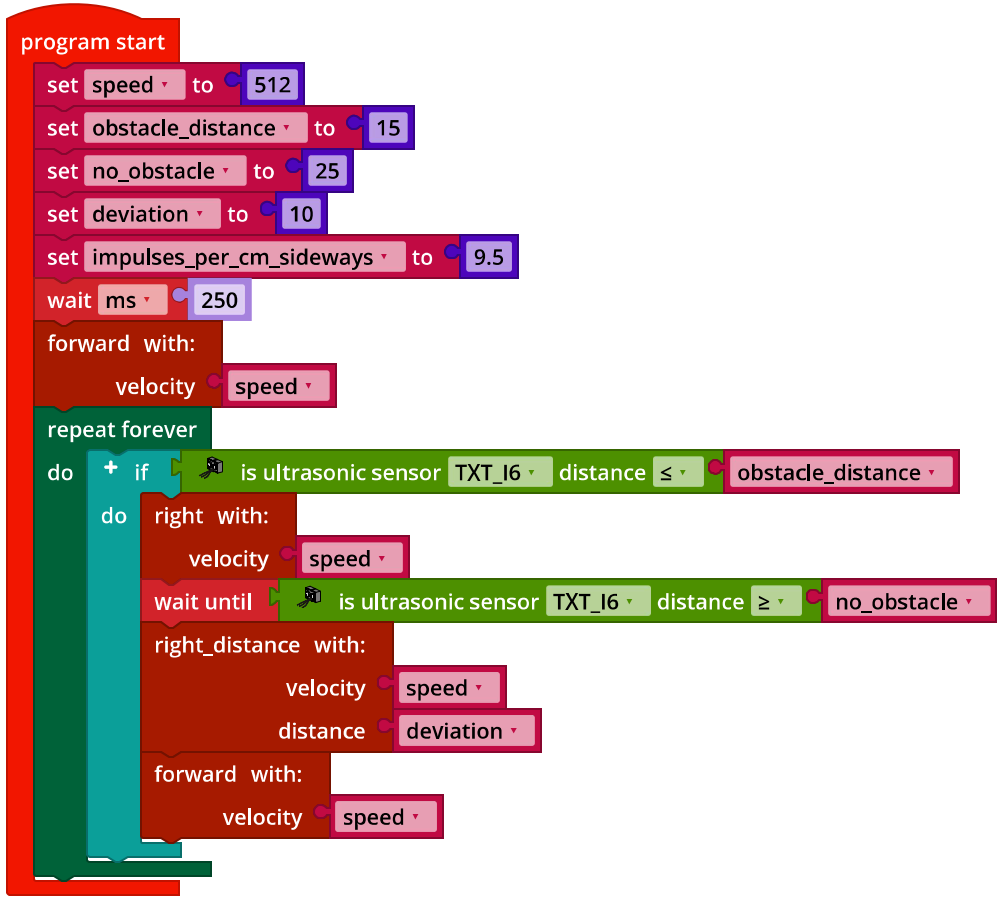
## Experimentieraufgaben

**1. Hinderniserkennung mit Ultraschall**

Konfiguration der Sensoren und Aktoren:



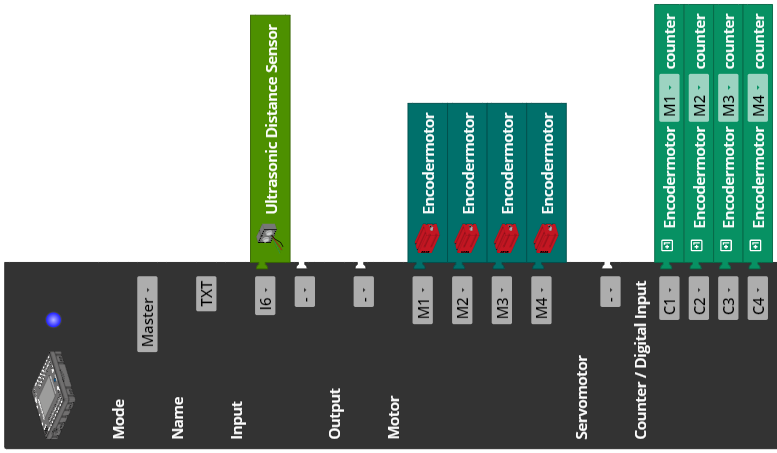
Programmauszug (Beispiel):



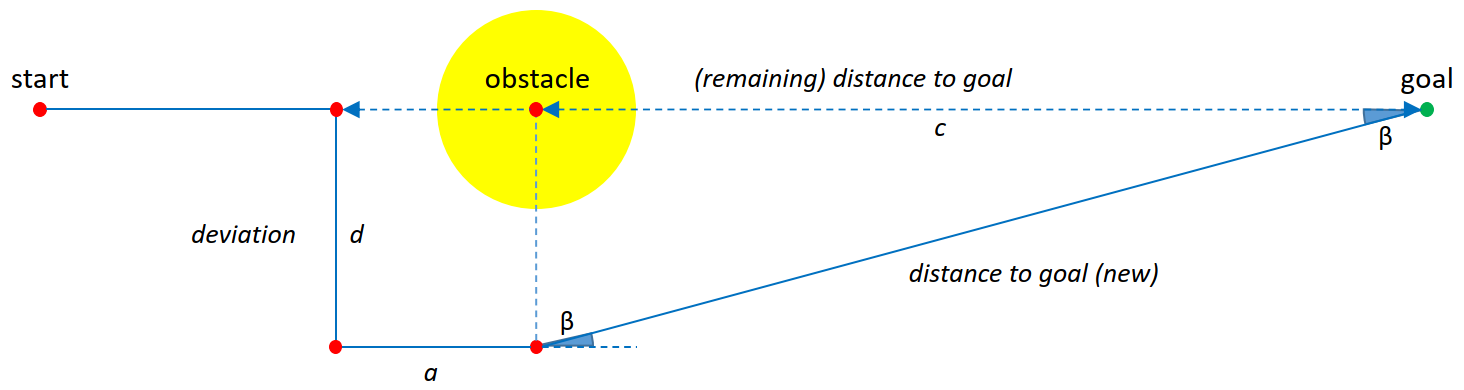
*Mecanum\_Collision\_Prevention.ft*

**2. Encoder-Navigation**

Konfiguration der Sensoren und Aktoren:



2a. Für diese Aufgabe gibt es mehrere Lösungsstrategien. Die im Folgenden dargestellte verwendet eine durch die Möglichkeiten der Mecanum-Omniwheels vereinfachte Version des mathematischen Modells, das als Lösungsbeispiel zur Experimentieraufgabe in Aufgabe 6 des Robotics TXT 4.0 Base Set beschrieben wurde:



*Mecanum\_Model\_Target\_Reacher.jpg*

**Lösungsstrategie:**

Das Mecanum-Omniwheels-Fahrzeug wird in Richtung des Ziels (*goal*) ausgerichtet und in Geradeausfahrt gestartet. Erkennt es auf der Fahrt ein Hindernis (*obstac­le*), so weicht es seitlich nach rechts aus, bis der Ultraschallsensor kein Hindernis mehr in weniger als 25 cm Abstand erkennen kann. Damit die Räder beim Vorbeifahren das Hindernis nicht streifen fährt es weitere 15 cm nach rechts ().

Anschließend fährt es 25 cm () gerade­aus. Dann werden aus den verbleibenden Impulsen für den (ursprüng­lichen) direkten Weg zum Ziel () und der bei der Umfahrung des Hindernisses in Seitwärtsfahrt zurückge­legten Strecke () der neue Abstand zum Ziel (*distance to goal (new)*) und der Drehwinkel für die Korrektur der Ausrichtung auf den Zielpunkt berech­net.

Die Berechnung des neuen Abstands zum Ziel ist einfach; wir erhalten ihn nach dem Satz des Pytha­goras:

Dabei erhält man die Strecke aus der Umrechnung der bei der Seitwärts­fahrt gezählten Impulse plus 15 cm; die Länge erhält man durch Subtraktion der Umfahrungsstrecke (25 cm) von der aus den zu Beginn des Ausweichmanövers noch zurückzulegenden Impulszahl (*remaining impulses to goal*) berechneten Strecke (*remaining distance to goal*).

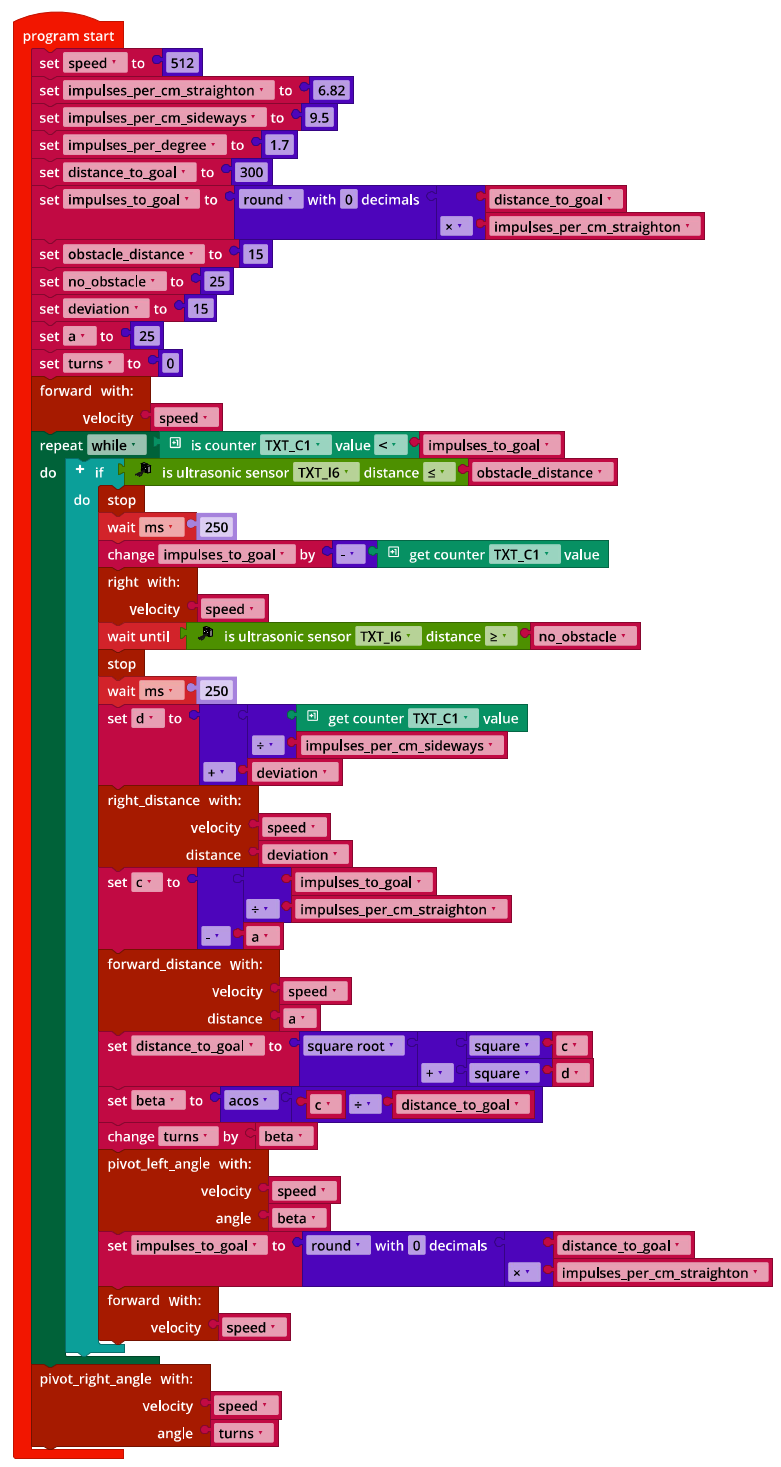
Der Winkel , um den wir den Roboter nach links drehen müssen, damit er wieder in Richtung Ziel fährt, lässt sich ebenfalls in einem Rechenschritt bestimmen:

Die Lösungsstrategie funktioniert auch mit mehreren Hindernissen in Folge.

Nach Erreichen des Zielpunkts wird das Fahrzeug wieder nach rechts in die ursprüngliche Fahrtrichtung gedreht. Dazu werden im Fahrtverlauf alle Winkel , um die das Mecanum-Omniwheels-Fahrzeug nach den Ausweichmanövern seine Richtung ändern musste, addiert.

**Wichtig**: Da sich die Umrechnungswerte der Impulse pro cm bei Geradeausfahrt und Seitwärtsfahrt unterscheiden, sollten die Berechnungen des neuen Abstands zum Ziel und des Drehwinkels in cm-Werten vorgenommen werden.

2b. Programmauszug (Beispiel):

*Mecanum\_Target\_Reacher.ft*

Anlagen

# Mecanum-Omniwheels-Fahrzeug mit Hinderniserkennung

## Erforderliches Material

* PC für Programmentwicklung, lokal oder über Web-Schnittstelle.
* USB-Kabel oder BLE- bzw. WLAN-Verbindung für die Übertragung des Programms auf den TXT4.0.
* Parcours-Bogen mit schwarzer, 2 cm breiter, geschlossener Kreislinie (aus Robotics TXT 4.0 Base Set).

## Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): [*How a Mecanum Drive Works*](https://seamonsters-2605.github.io/archive/mecanum/). github.io

[2] Online-Diagrammeditor zur Erstellung von Zustandsübergangsdiagrammen (Format drawio): <https://www.diagrammeditor.de/>