Name: _____ Klasse: ___ Datum: _____

Aufgabe 1

Mecanum-Omniwheels-Fahrzeug mit Hinderniserkennung

Konstruktionsaufgabe

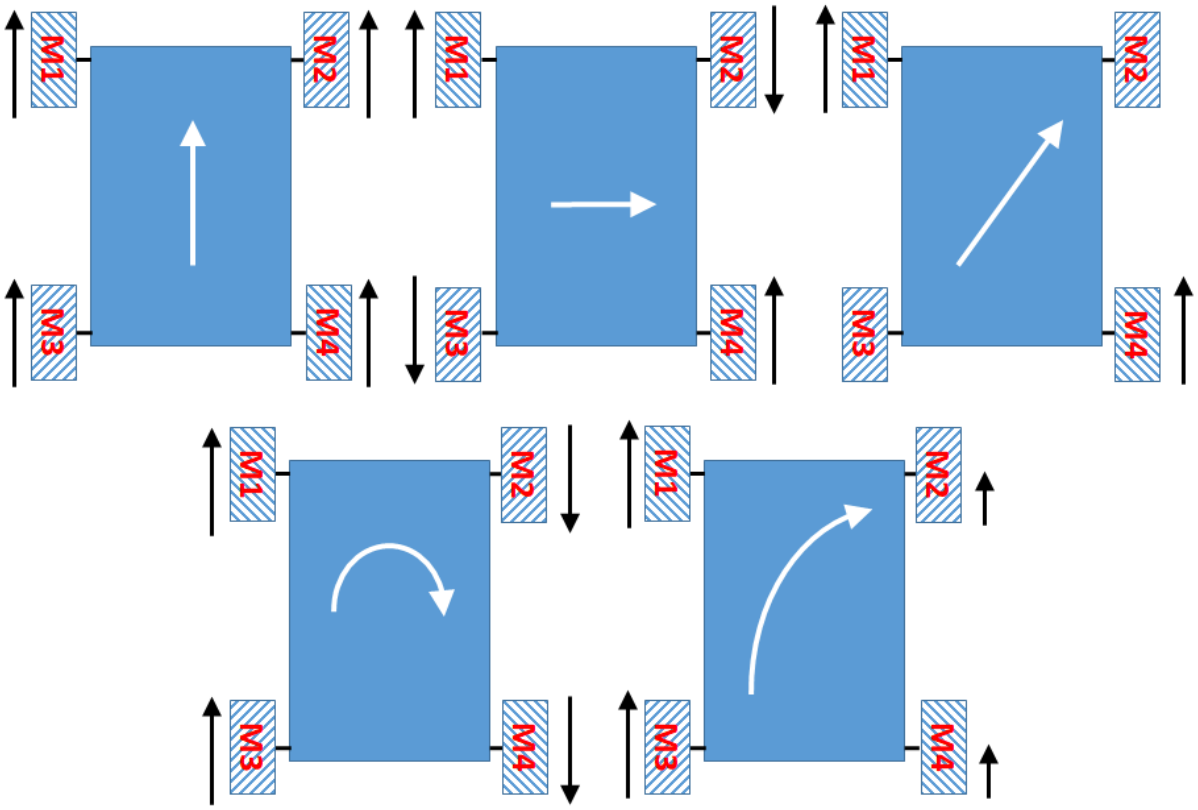
Baue das Mecanum-Omniwheels-Fahrzeug „Basismodell mit Sensoren“ nach der Bauanleitung. Die Naben der Zahnräder Z20 und der Mecanum-Omniwheels müssen fest angezogen werden, damit beim Fahren kein „Schlupf“ entsteht.

In dieser Aufgabe werden der Spursensor (vorne) und der Ultraschall-Abstandssensor zur Hinderniserkennung verwendet.

Der Ultraschall-Abstandssensor wird an I6 (schwarzes Kabel), die beiden Infrarot-(IR-) Sensoren werden an I7 (rechter Sensor, gelb/blaues Kabel) und I8 (linker Sensor, blaues Kabel) angeschlossen. Die IR-Sensoren und der Ultraschallsensor müssen außerdem über den 9V-Spannungsausgang des TXT mit Strom versorgt werden.

Prüfe mit dem Interface-Test, ob die vier Motoren korrekt angeschlossen (vorwärts: Linkslauf, d. h. Drehung gegen den Uhrzeigersinn), die vier Encoder mit den richtigen Zählereingängen verbunden sind (M1: C1, M2: C2, M3: C3 und M4: C4) und die beiden IR-Sensoren die richtigen Werte liefern (weiße Fläche: 1, schwarze Linie: 0).

Programmieraufgaben



Wichtigste Bewegungsarten mit Mecanum-Omniwheels

1. Synchroner Antrieb in alle Richtungen

Die Abbildung veranschaulicht die Steuerung des Mecanum-Omniwheels-Fahrzeugs. Die oberen drei Bewegungsarten zeigen den für die Geradeausfahrt, die Schrägfahrt und die seitliche Bewegung erforderlichen Radantrieb. Jede der drei Bewegungsarten umfasst die Vorwärts- und die Rückwärtsbewegung bzw. die Bewegung nach links und rechts. Ist der schwarze Pfeil neben dem Rad nach oben gerichtet, soll sich das zugehörige Rad vorwärts drehen.

Damit lassen sich die folgenden acht Bewegungen des Mecanum-Omniwheels-Fahrzeugs unterscheiden (siehe auch die Animation in [1]):

- Vorwärts
- Rückwärts
- Seitwärts nach links
- Seitwärts nach rechts
- Schräg nach links vorne

- Schräg nach rechts vorne
- Schräg nach links hinten
- Schräg nach rechts hinten

Entwickle zu jeder dieser Bewegungen ein Blockly-Unterprogramm (eine Funktion), an das du die Geschwindigkeit der Motoren als Parameter übergeben kannst.

Die Funktionen werden in den folgenden Aufgaben immer wieder benötigt. Damit werden die Steuerungsprogramme übersichtlicher und einfacher zu verstehen. Teste deine Funktionen mit einfachen Beispielprogrammen.

Beachte: Bei den Mecanum-Omniwheels ist die Synchronisation der Motoren besonders wichtig (siehe dazu auch Aufgabe 6 des Robotics TXT 4.0 Base Set).

2. Synchrones Drehen

Die unteren beiden Bewegungsarten in der Abbildung zeigen die beiden wichtigsten mit Mecanum-Omniwheels möglichen Drehbewegungen: das Drehen auf der Stelle und das Fahren einer Kurve. Auch darin sind jeweils mehrere Bewegungsrichtungen umfasst, wie das Drehen nach links und rechts oder das Abbiegen nach links oder rechts sowie die Vorwärts- und Rückwärtskurvenfahrt.

Insgesamt ergeben sich so sechs weitere Bewegungen:

- Drehen auf der Stelle nach rechts (mit dem Uhrzeigersinn)
- Drehen auf der Stelle nach links (gegen den Uhrzeigersinn)
- Abbiegen nach rechts
- Abbiegen nach links
- Abbiegen rückwärts nach rechts
- Abbiegen rückwärts nach links

Entwickle zu jeder dieser Bewegungen ein Blockly-Unterprogramm (eine Funktion), an das du die Geschwindigkeit der Motoren als Parameter übergibst. Teste deine Funktionen mit einfachen Beispielprogrammen.

3. Synchroner Antrieb mit Distanz-Vorgabe

Um das Mecanum-Omniwheels-Fahrzeug präzise zu navigieren benötigst du nun einen Ausschnitt des Befehlssatzes aus den Programmieraufgaben 1 und 2, jeweils zuzüglich einer Distanz-Vorgabe – der Angabe der Zahl der Impulse, um die die Motoren sich drehen sollen.

3a. Ergänze die folgenden sechs Bewegungsfunktionen aus den Programmieraufgaben 1 und 2 jeweils um eine Distanz-Vorgabe: eine feste Anzahl von Impulsen

- zur Geradeausfahrt (vorwärts/rückwärts),
- zur Seitwärtsfahrt (links/rechts) und

- zum Drehen um die eigene Achse (links: im Uhrzeigersinn, rechts: gegen den Uhrzeigersinn).

3b. Teste diese sechs Funktionen an einer zuvor ausgemessenen Teststrecke und bestimme experimentell die Faktoren (Impulse je cm bzw. Impulse je °) zur Bestimmung der erforderlichen Impulse aus

- einer in cm angegebenen Distanz bei Geradeausfahrt
- einer in cm angegebenen Distanz bei Seitwärtsfahrt und
- einem in ° angegebenen Drehwinkel

Für den Umrechnungsfaktor bei Geradeausfahrt kannst du (analog Aufgabe 6 des Robotics TXT 4.0 Base Set) als erste Näherung den Umfang eines Mecanum-Omniwheels-Rads messen und daraus den Faktor ableiten.

4. Linienerkennung

Mit dem Spursensor soll das Mecanum-Omniwheels-Fahrzeug nun außerdem Begrenzungslinien und Kanten identifizieren und sie umfahren: Wird eine (schwarze) Begrenzungslinie oder ein Abgrund erkannt, soll das Fahrzeug 10 cm zurücksetzen, sich um eine Achteldrehung (45°) von der Kante abwenden und dann die Fahrt fortsetzen.

4a. Zeichne ein entsprechendes Zustandsübergangsdiagramm.

4b. Schreibe nun unter Verwendung deiner Navigationsfunktionen aus den Programmieraufgaben 1 und 2 ein passendes Blockly-Programm (siehe auch Aufgabe 6 des Robotics TXT 4.0 Base Set).

Experimentieraufgaben

1. Hinderniserkennung mit Ultraschall

Das Mecanum-Omniwheels-Fahrzeug ist mit einem Ultraschallsensor ausgestattet, der dir den Abstand zu einem Objekt in cm liefert (siehe auch Aufgabe 1 des Robotics TXT 4.0 Base Set).

Schreibe ein Blockly-Programm, das verhindert, dass das Fahrzeug einem Hindernis näher als 15 cm kommt. Erkennt es ein Hindernis, soll es diesem durch eine Seitwärtsfahrt ausweichen, bis der Ultraschallsensor kein Hindernis mehr in weniger als 25 cm Entfernung feststellen kann (siehe auch Aufgabe 6 des Robotics TXT 4.0 Base Set), und dann seine Fahrt fortsetzen.

2. Encoder-Navigation

Auch beim Mecanum-Omniwheels-Fahrzeug kannst du aus den Werten des Encoders die zurückgelegte Strecke bestimmen. Damit sollst du es nun zu einem vorgegebenen Ziel navigieren, zu dem ihm die Luftlinien-Entfernung in cm vorgegeben ist. Markiere für deine Tests einen 3 m (Luftlinie) entfernten Zielpunkt auf dem Boden.

Verstelle nun den direkten Weg zum Ziel durch zunächst eines, später auch mehrere Hindernisse. Überlege dir eine Strategie, nach der das Fahrzeug den Hindernissen ausweicht und anschließend die Fahrt zu dem vorgegebenen Zielpunkt auf dem dann kürzesten Weg fortsetzt.

2a. Beschreibe deine Strategie mit einer Skizze.

2b. Erweitere dein Blockly-Programm aus Experimentieraufgabe 1 entsprechend.

Tipp: Die Drehung des Mecanum-Omniwheels-Fahrzeugs erfolgt um die Fahrzeugmitte. Alle Berechnungen zur Fahrstrecke beziehen sich daher immer auf den Mittelpunkt des Fahrzeugs. Soll die Fahrt an einer Linie vor der Stoßstange beginnen und enden, muss das Fahrzeug zum Schluss wieder in Fahrtrichtung gedreht werden, damit es vor der Ziellinie zum Stehen kommt.

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*. github.io

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

Aufgabe 1: Mecanum-Omniwheels-Fahrzeug mit Hinderniserkennung

In dieser Einstiegsaufgabe machen sich die Schülerinnen und Schüler mit der Navigation eines Fahrzeugs mit Mecanum-Omniwheels-Rädern vertraut. Dabei werden Funktionen zur Abstraktion der Bewegungsvorgänge entwickelt. Das Fahrzeug lernt, sich im Raum zu bewegen und Hindernissen und Begrenzungslinien auszuweichen.

Thema

Steuerung eines Fahrzeugs mit Mecanum-Omniwheels-Rädern und Erkennung von Hindernissen.

Lernziele

- Verständnis der Funktionsweise von Mecanum-Omniwheels-Rädern und deren Ansteuerung
- Übersichtliche Programmierung durch Zustandsvariablen und Funktionen
- Erkennung von Linien und Hindernissen mit Sensoren (Infrarot, Ultraschall)
- Navigation über Motorimpulse

Zeitaufwand

Für den Aufbau des Mecanum-Omniwheels-Fahrzeugs „Basismodell mit Sensoren“ anhand der Bauanleitung benötigen die Schülerinnen und Schüler je nach Vorerfahrung 45-90 Minuten (ein bis zwei Unterrichtsstunden).

Für die Entwicklung der Programme zur Lösung der Programmieraufgaben benötigen Schülerinnen und Schüler, Erfahrung mit dem Robotics TXT 4.0 Base Set vorausgesetzt, zwei bis drei Unterrichtsstunden (90-135 Minuten).

Die Bearbeitung der Experimentieraufgaben erfordert weitere 90-135 Minuten.

Bezug Curriculum

Land	Stufe/Fächer	Bezüge
BW		
BY		
BE		
BB		
HB		
HH		
HE		
MV		
NI		
NW		
RP		
SL		
SN		
ST		
SH		
TH		

Anlagen
Aufgabe 1: 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*. github.io

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

Name: _____ Klasse: ___ Datum: _____

Lösungsblatt Aufgabe 1

Mecanum-Omniwheels-Fahrzeug mit Hinderniserkennung

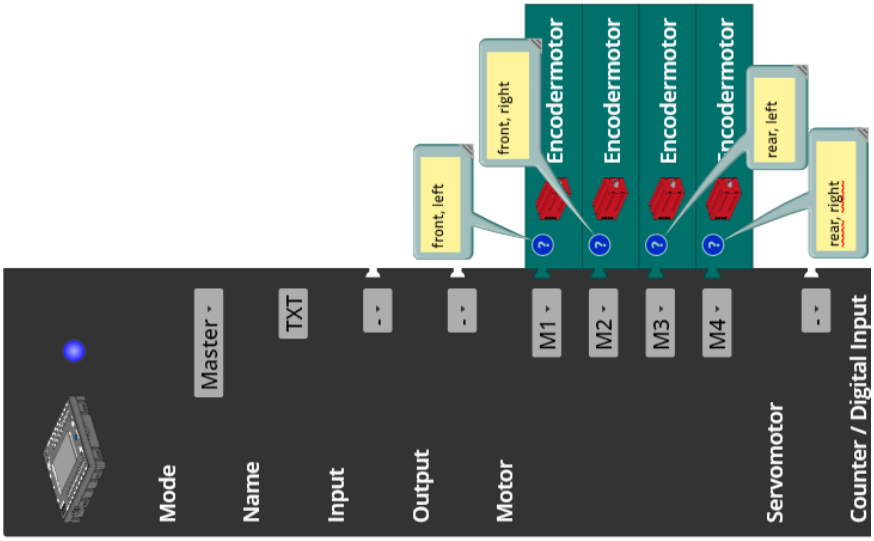
Die in der ersten Programmieraufgabe entwickelten Steuerungsfunktionen bilden die Grundlage für alle folgenden Aufgabenstellungen.

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.

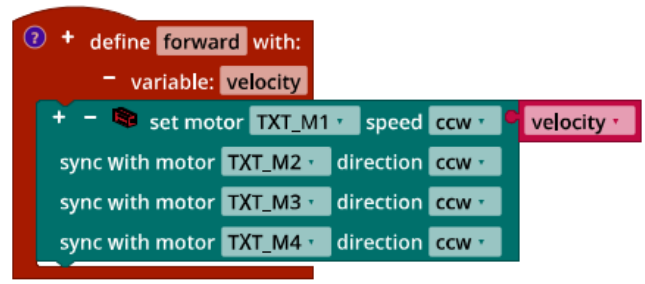
Programmieraufgaben

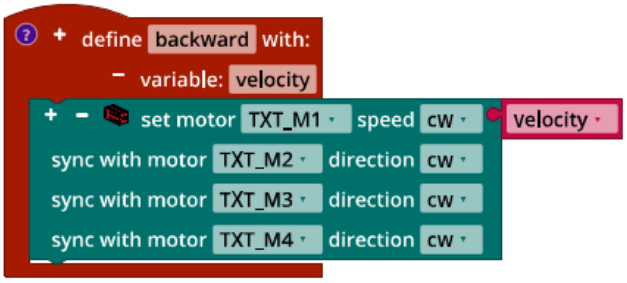
1. Synchroner Antrieb in alle Richtungen

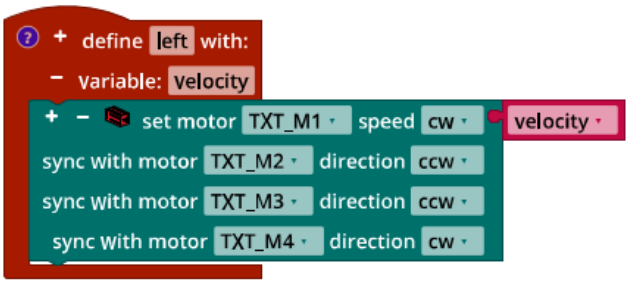
Konfiguration der Sensoren:

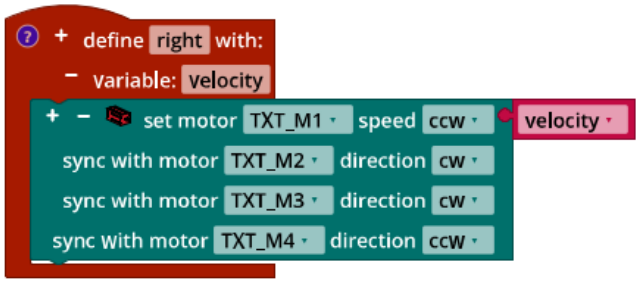


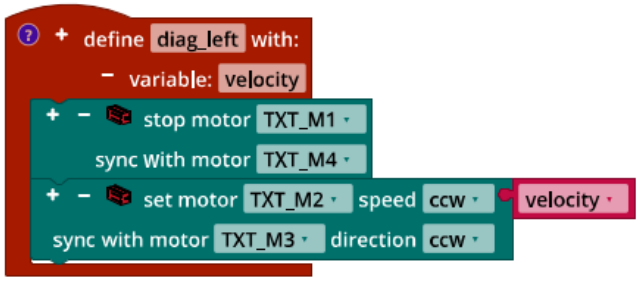
Steuerungsfunktionen (Beispiel):

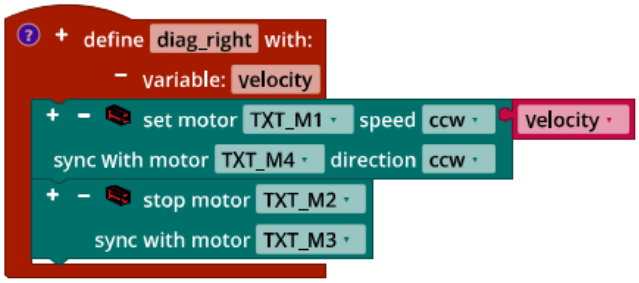


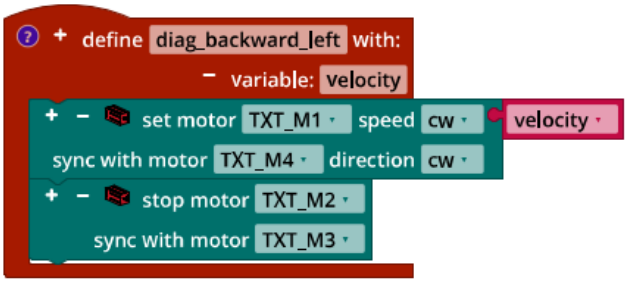


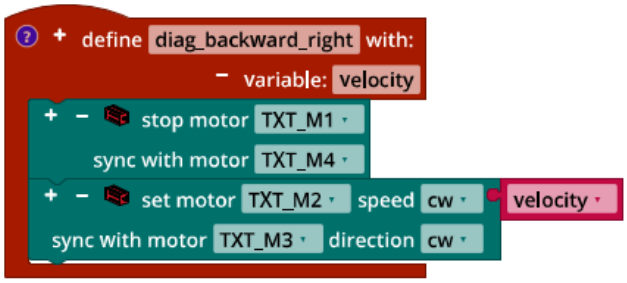








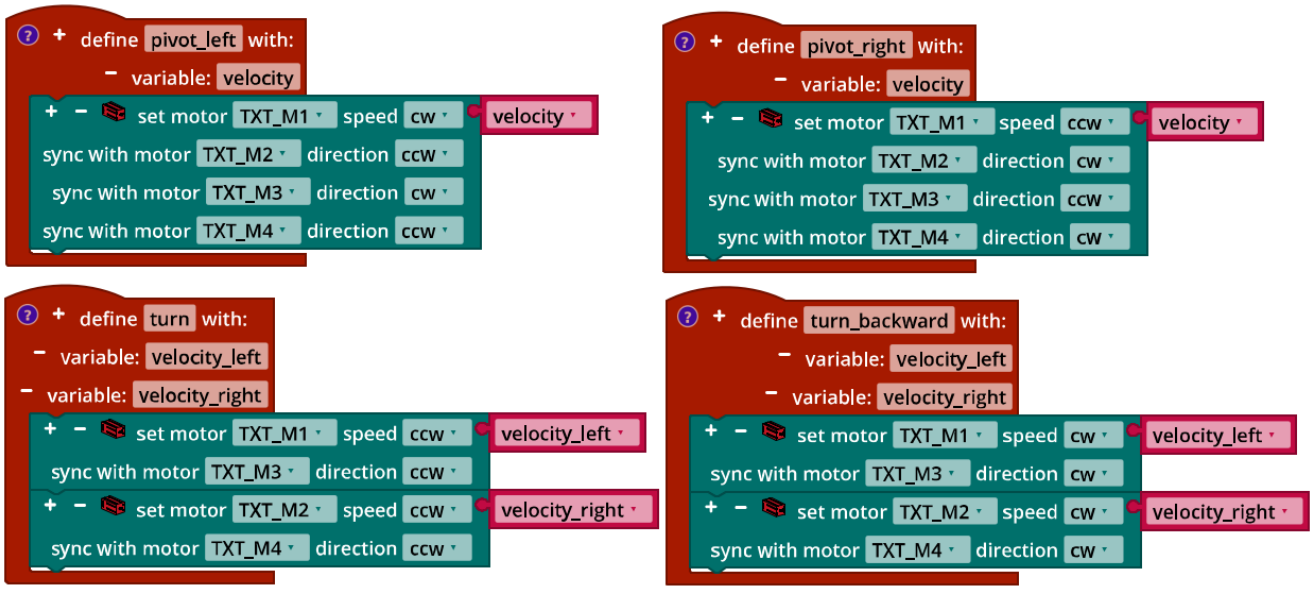




Mecanum_Synchronous_Driving_Funktions.ft

2. Synchrones Drehen

Steuerungsfunktionen (Beispiel):

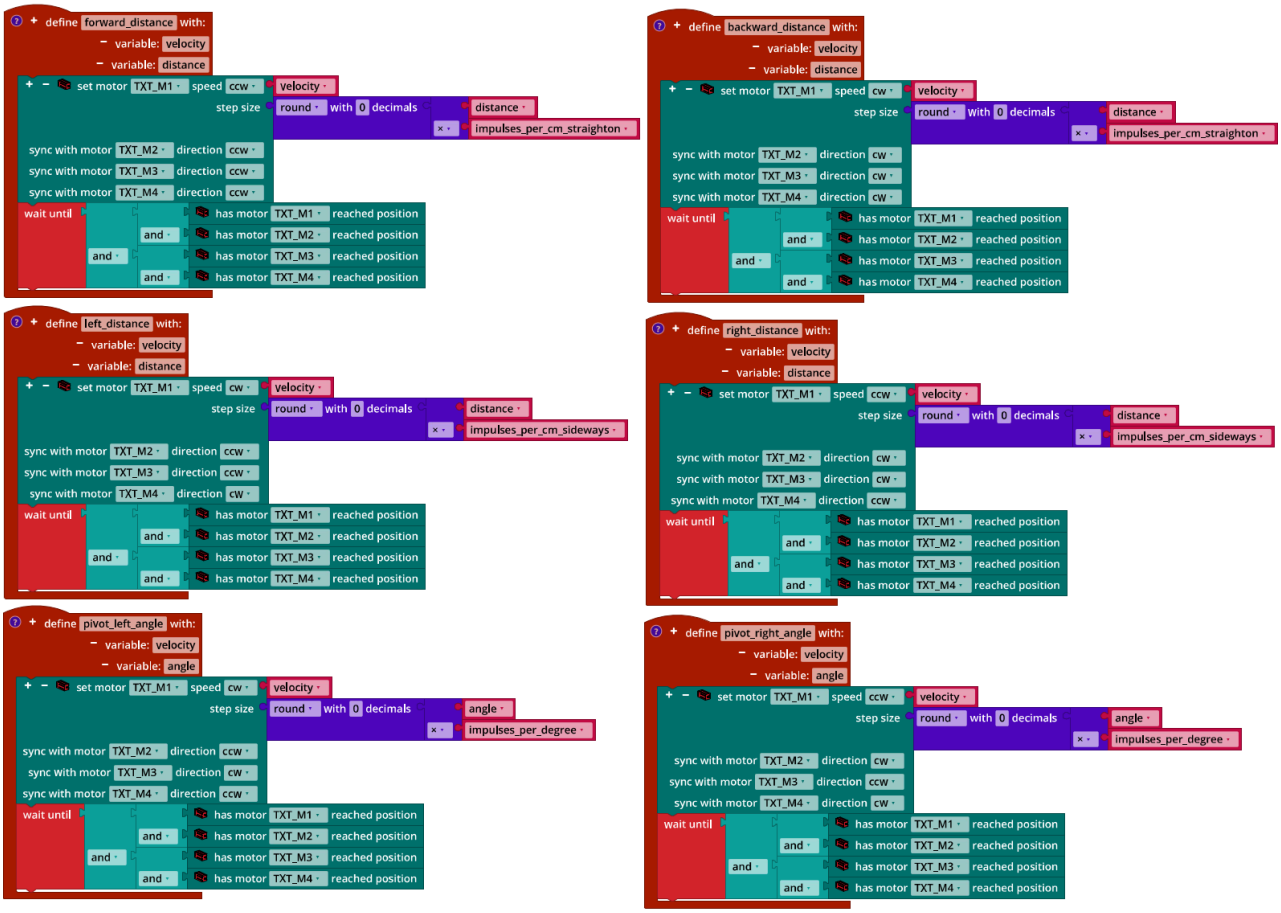


Mecanum_Synchronous_Turning_Functions.ft

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

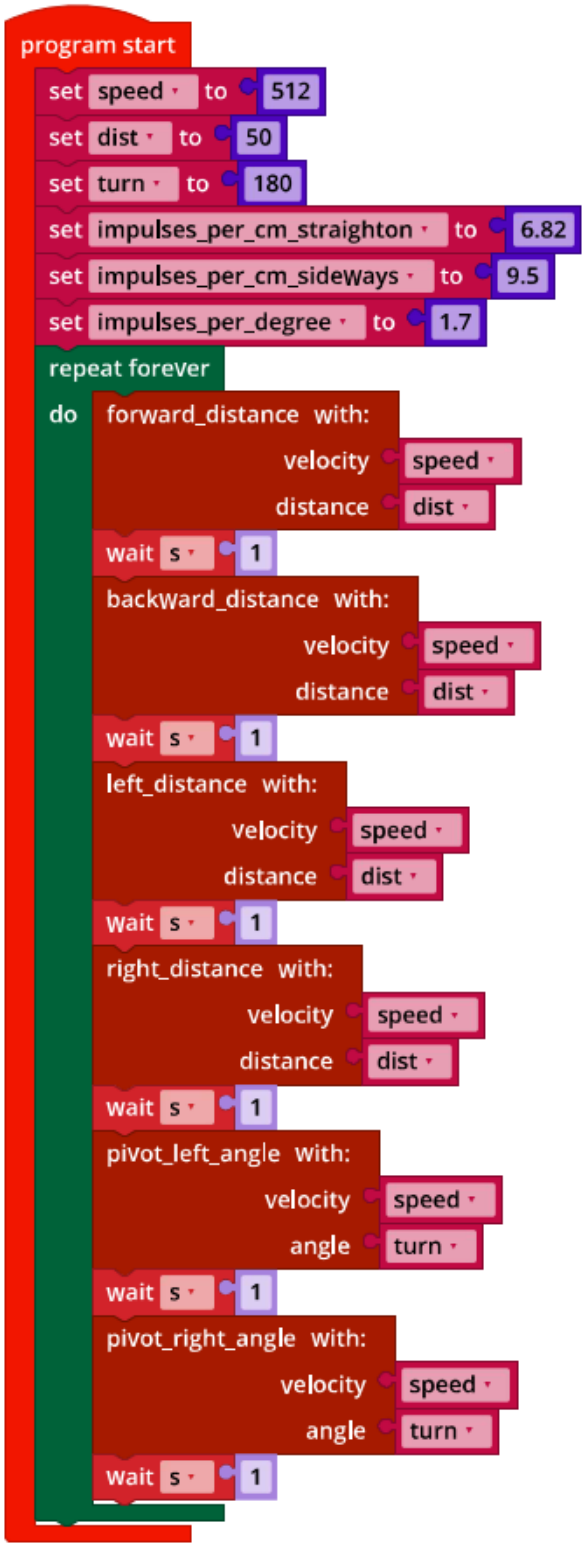
3b. Der Umfang eines Mecanum-Omniwheels beträgt ungefähr 19 cm. Daraus lässt sich die Zahl der Impulse je Umdrehung (wie in Aufgabe 6 des Robotics TXT 4.0 Base Set) leicht berechnen: Da die Räder von den Motoren mit einer Übersetzung von 1:2 ins Langsame angetrieben werden sind es etwa $\frac{63{,}9 \cdot 2}{19} \approx \mathbf{6{,}726}$ **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.

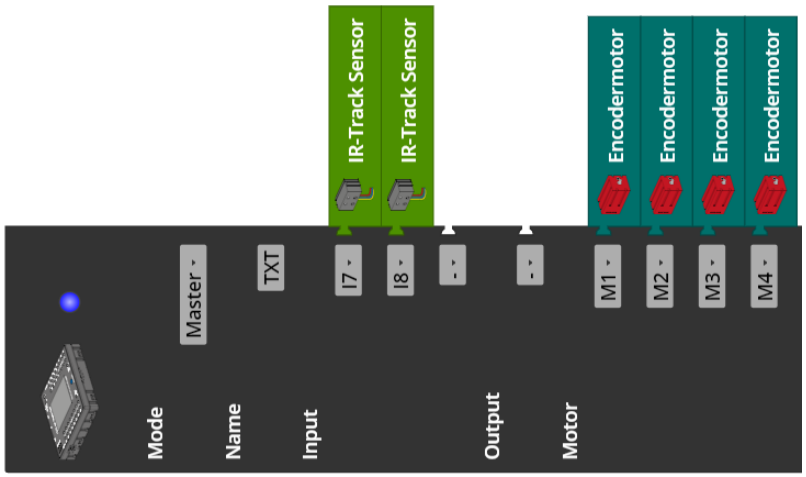
Programm (Beispiel) zum Testen der Umrechnungsfaktoren:



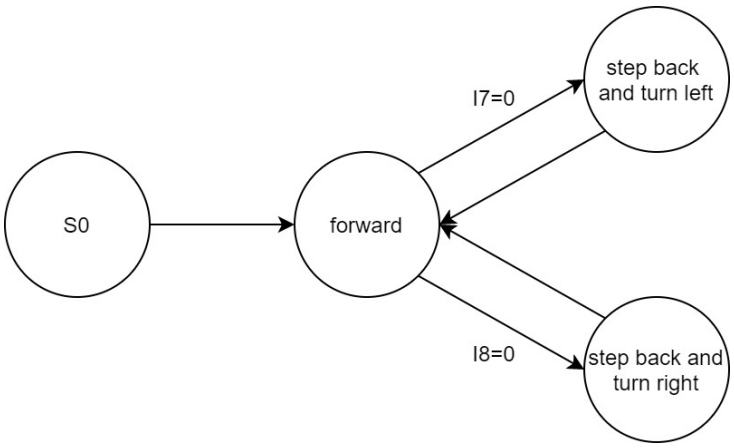
Mecanum_Synchronous_Navigation_Distance.ft

4. Linienerkennung

Konfiguration der Sensoren und Aktoren:

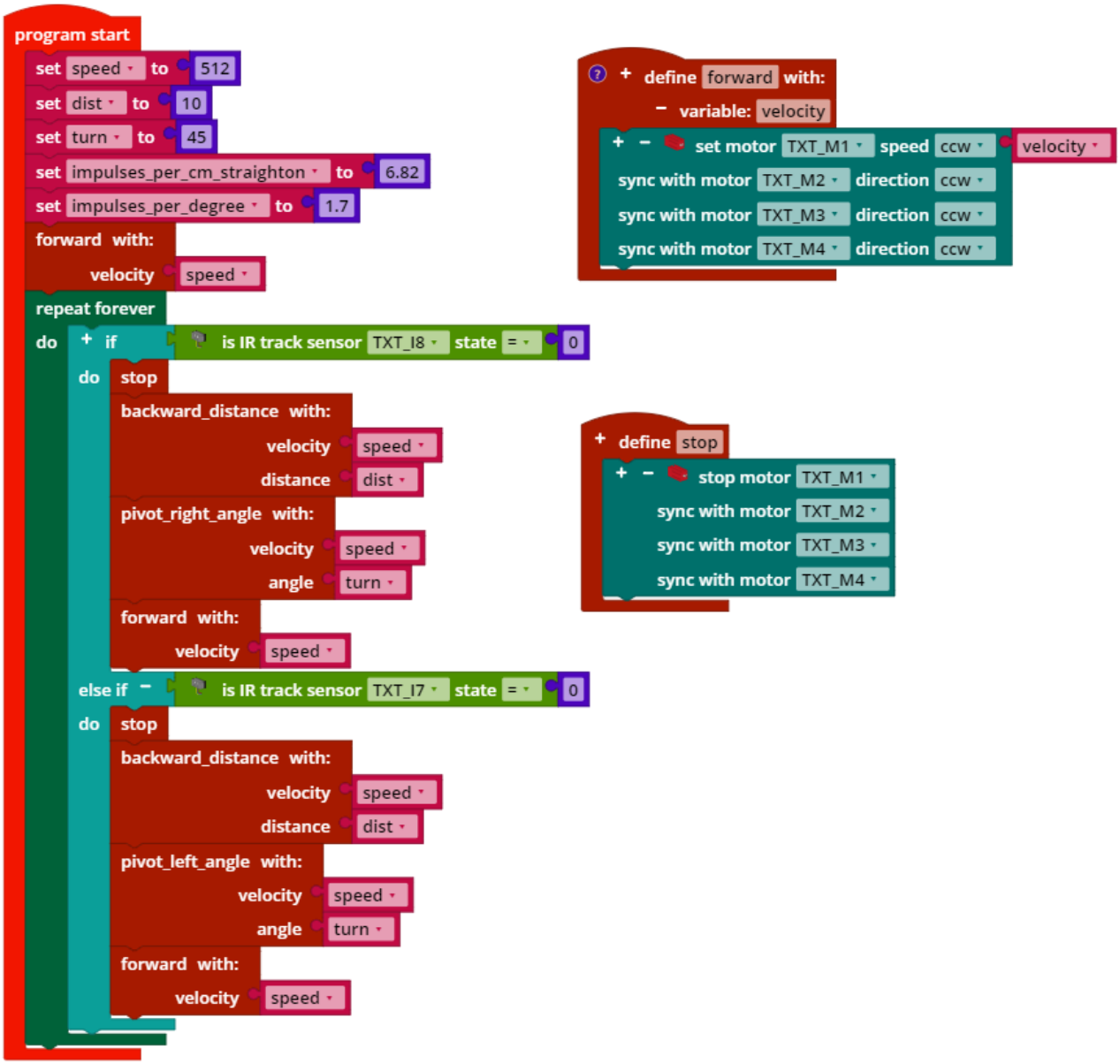


4a. Zustandsübergangsdiagramm:



State-Transition_Diagram_Mecanum_with_IR_Sensors.drawio

4b. Programmauszug (Beispiel):

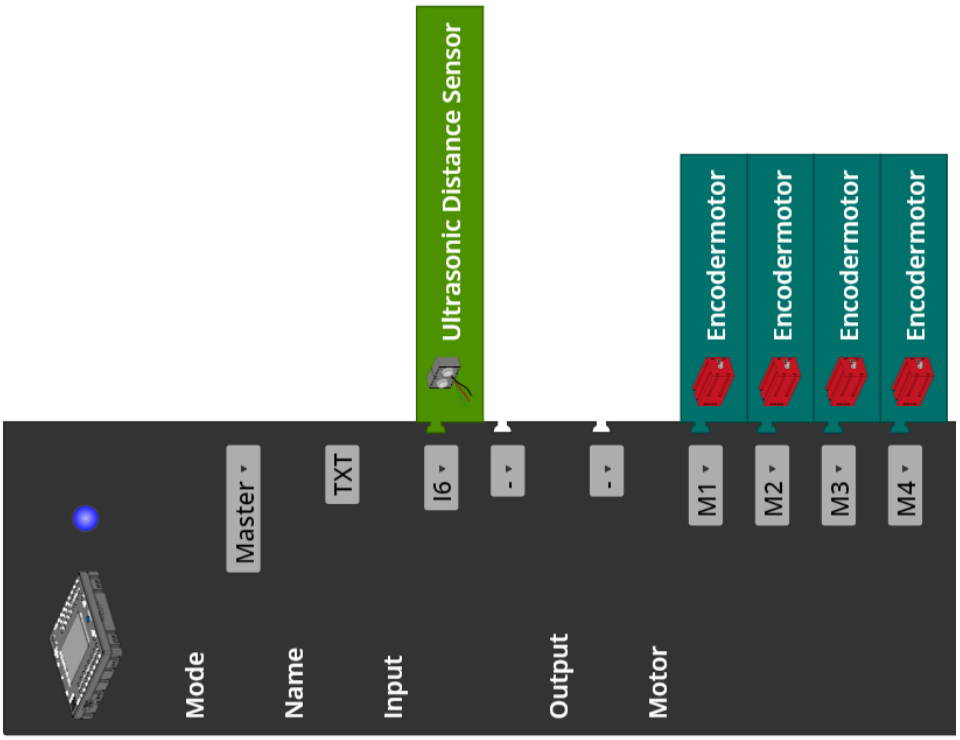


Mecanum_Boundary_Line.ft

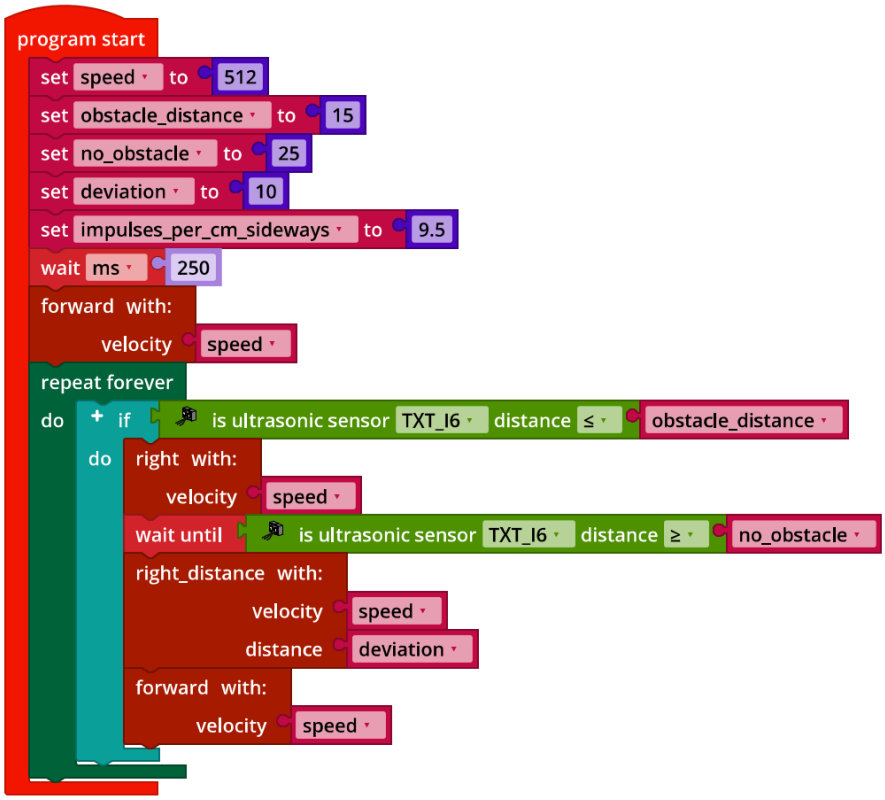
Experimentieraufgaben

1. Hinderniserkennung mit Ultraschall

Konfiguration der Sensoren und Aktoren:



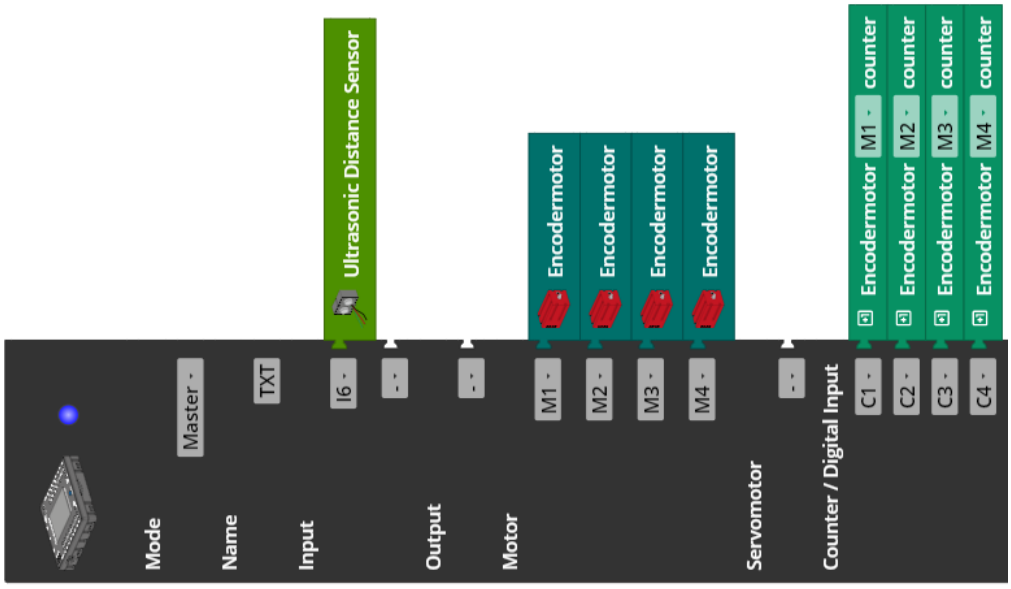
Programmauszug (Beispiel):



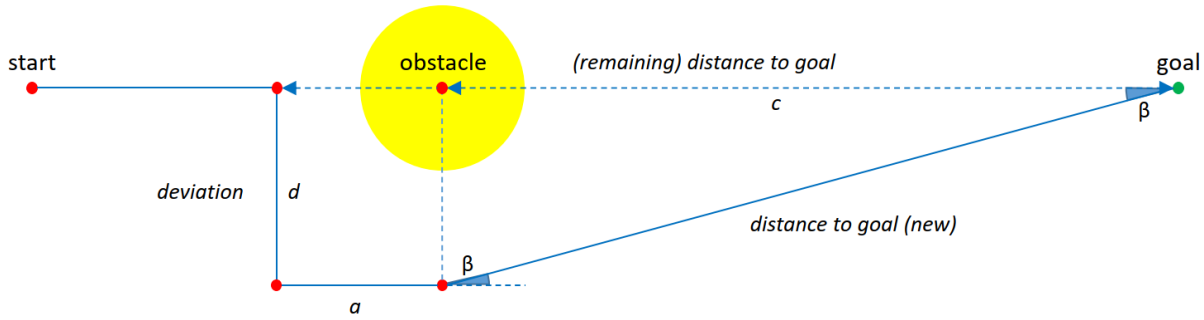
Mecanum_Collision_Prevention.ft

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 (d).

Anschließend fährt es 25 cm (a) geradeaus. Dann werden aus den verbleibenden Impulsen für den (ursprünglichen) direkten Weg zum Ziel (c) und der bei der Umfahrung des Hindernisses in Seitwärtsfahrt zurückgelegten Strecke (d) der neue Abstand zum Ziel (*distance to goal (new)*) und der Drehwinkel β 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:

$$distance\ to\ goal\ (new) = \sqrt{c^2 + d^2}$$

Dabei erhält man die Strecke d aus der Umrechnung der bei der Seitwärtsfahrt gezählten Impulse plus 15 cm; die Länge c erhält man durch Subtraktion der Umfahrungsstrecke a (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 β, um den wir den Roboter nach links drehen müssen, damit er wieder in Richtung Ziel fährt, lässt sich ebenfalls in einem Rechenschritt bestimmen:

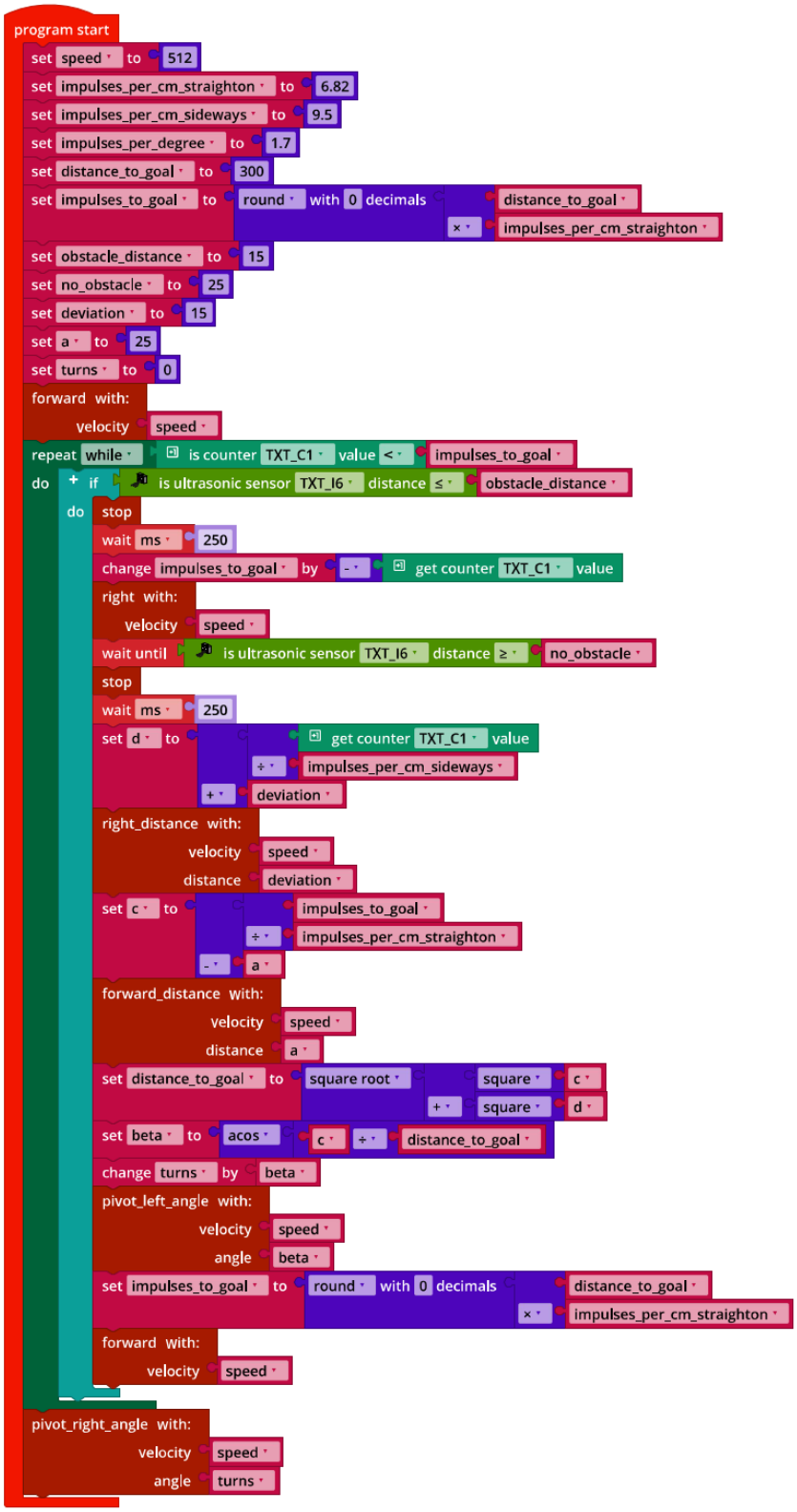
$$\beta = \arccos\left(\frac{c}{distance\ to\ goal\ (new)}\right)$$

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 β, 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 β 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*. github.io

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

Name: _____ Klasse: ___ Datum: _____

Aufgabe 2

Spurfolger

Konstruktionsaufgabe

Für die dritte Programmieraufgabe und die Experimentieraufgaben benötigen wir die Kamera am Mecanum-Omniwheels-Fahrzeug „Basismodell mit Sensoren“. Schließe sie an die USB1-Buchse des TXT an.

Wichtig: Wenn der TXT gestartet ist, stellt sich der Servo automatisch auf „Geradeaus-Stellung“. Stecke den Servo-Hebel anschließend so auf, dass die Kamera in einem Winkel von etwa 45° gegenüber der Senkrechten nach vorne geneigt ist.

Achtung: Wenn du den Servo bei eingeschaltetem TXT mit der Hand bewegst, kannst du ihn beschädigen.

Programmieraufgaben

1. Spurfolger mit Spursensor

Das Mecanum-Omniwheels-Fahrzeug soll nun mit Hilfe des Spursensors der schwarzen, ca. 2 cm breiten Spur des Parcours aus dem Robotics TXT 4.0 Base Set folgen (siehe auch Aufgabe 8 des Robotics TXT 4.0 Base Set).

1a. Erstelle zunächst ein Zustandsübergangsdiagramm, das die möglichen Zustände und das Verhalten des Mecanum-Omniwheels-Fahrzeugs beschreibt.

1b. Setze dein Zustandsübergangsdiagramm in ein Blockly-Programm um. Verwende dafür eine Zustandsvariable, deren Zustandswert du aus den Werten der IR-Sensoren bestimmst.

Teste das Programm an dem kreisförmigen Parcours des Robotics TXT 4.0 Base Set.

1c. Wie lässt sich die Geschwindigkeit des Mecanum-Omniwheels-Spurfolgers erhöhen? Mache Versuche und messe die Zeit, die der Spurfolger jeweils für die Bewältigung des Parcours benötigt.

Lösungsvariante (Anpassungen, Parameter)	Zeit

2. Spurfolger mit Hinderniserkennung

Auf der Spur können sich auch Hindernisse befinden. Wenn der Ultraschallsensor ein Hindernis in maximal 2 cm Entfernung vor dem Fahrzeug entdeckt, soll sich das Fahrzeug seitwärts bewegen, am Hindernis vorbeifahren und dann wieder auf die Spur zurückfinden.

Überlege dir verschiedene Lösungsvarianten für das Finden der Linie. Erweitere dein Blockly-Programm entsprechend und teste die Varianten. Welche Vor- und Nachteile haben sie?

3. Spurfolger mit Farbsteuerung

In den Ecken des Parcours erkennst du vier Farbflächen. Mit diesen Farbflächen kannst du deinem Mecanum-Omniwheels-Fahrzeug Kommandos geben.

3a. Erweitere dein Blockly-Programm um eine Farberkennung. Lass‘ dir zur Kalibrierung der Erkennung den RGB-HEX-Farbcode der erkannten Farbe auf der Konsole ausgeben.

3b. Lege fest, bei welchen Farben (mindestens zwei) das Mecanum-Omniwheels-Fahrzeug wie reagieren soll und passe dein Blockly-Programm entsprechend an.

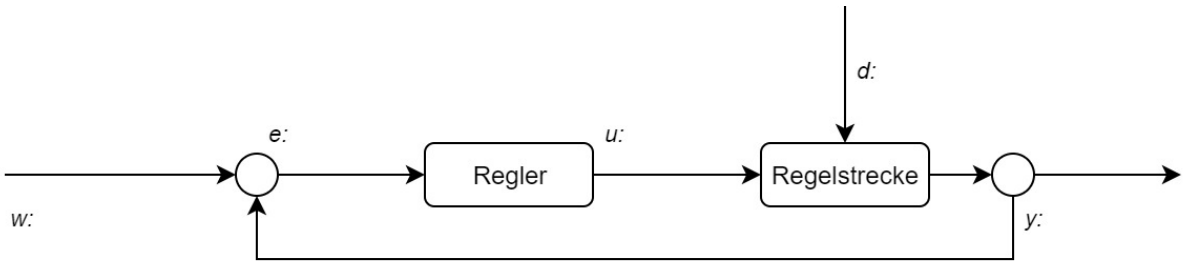
Hinweis: Wundere dich nicht, dass der auf der Konsole angezeigte HEX-Wert stark zu schwanken scheint – der Hue-Wert ist stabil. Du kannst jeden der angezeigten HEX-Farbcodes für die Erkennung verwenden.

Experimentieraufgaben

1. Spurfolger mit Proportionalregler

Der Mecanum-Omniwheels-Spurfolger soll nun wie der Spurfolger-Buggy aus Aufgabe 8 des Robotics TXT 4.0 Base Set mit einem Proportionalregler (P-Regler) ausgestattet werden. Dabei soll die Stärke der Richtungskorrektur von der Größe der Linienabweichung abhängen. Auch hier soll mit der Linienerkennungsfunktion der Kamera die Größe der Abweichung von der Spur bestimmt werden.

1a. Beschrifte zunächst den folgenden Regelkreis mit den entsprechenden Werten deines Mecanum-Omniwheels-Spurfolgers:



1b. Konzipiere und programmiere eine Proportionalregelung (P-Regler) für den Mecanum-Omniwheels-Spurfolger. Dabei kannst du das Erkennungsverfahren für die schwarze Linie aus der Experimentieraufgabe 1 von Aufgabe 8 des Robotics TXT 4.0 Base Set verwenden.

Teste dein Programm an dem einfachen geraden Linien-Parcours, indem du den Mecanum-Omniwheels-Spurfolger parallel zur Spur so startest, dass der Mittelpunt der Spur möglichst weit links im Bild erscheint.

Tipp: Beginne mit dem Proportionalitätsfaktor $k_p = 2$. Erhöhe den Wert schrittweise um 1, bis der Mecanum-Omniwheels-Spurfolger sich zügig „einschwingt“, d. h. die Abweichung von der Spur sich schnell verringert, ohne dass der Regler auf- oder stark überschwingt.

1c. Ergänze in deinem Programm eine Textausgabe, die nach jeder Änderung der Abweichung von der Spur

- die Zeit (in ms), die seit dem Start des Programms verstrichen ist, und
- den Wert der aktuellen Abweichung

durch ein Leerzeichen getrennt auf der Konsole ausgibt.

Kopiere nach jeder Testfahrt mit einem anderen Wert für k_p die Konsolen-Ausgaben in eine Tabellenkalkulation und lass‘ dir die Werte grafisch in einem Diagramm (x: Zeit,

y: Abweichung von der Spur) anzeigen. Passe den Proportionalitätsfaktor k_p so lange an, bis sich der Kurvenverlauf schnell einschwingt.

Teste den Spurfolger mit dem kreisförmigen Parcours aus dem Robotics TXT 4.0 Base Set. Eventuell musst du dafür die Höchstgeschwindigkeit etwas senken.

2. Spurfolger mit PD-Regler

Wie beim Buggy in Aufgabe 8 des Robotics TXT 4.0 Base Set kannst du durch eine Erweiterung des Reglers um ein „D"-Glied (Differential-Anteil), das die Größe der Veränderung der Abweichung von der Spur bei der Geschwindigkeitskorrektur berücksichtigt, das Überschwingen weiter dämpfen.

Erweitere den P-Regler deines Mecanum-Omniwheels-Spurfolgers aus Experimentieraufgabe 1 entsprechend zu einem PD-Regler. Führe Testfahrten mit unterschiedlichen Werten für den Differential-Faktor k_d durch und lass' dir die Daten in einem Tabellenkalkulationsprogramm grafisch anzeigen.

Tipp: Beginne deine Tests mit dem Differential-Faktor $k_d = 0{,}25$ und erhöhe dessen Wert so lange in Schritten von 0,25, bis das Überschwingen des P-Reglers gut gedämpft wird..

Anlagen
Aufgabe 2: Spurfolger

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, gerader Linie
- Parcours-Bogen mit schwarzer, 2 cm breiter, geschlossener Kreislinie (aus Robotics TXT 4.0 Base Set)
- Hindernis (Karton, Dose, …)

Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Endlicher Automat (Zustandsautomat)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

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

[5] Wikipedia: *Regelungstechnik*.

[6] Wikipedia: *Regler*.

[7] RN-Wissen: *Regelungstechnik*.

[8] Tim Wescott: *PID without a PhD*. Embedded Systems Programming, 10/2000, S. 86-108.

Aufgabe 2: Spurfolger

In dieser Aufgabe wird mit Hilfe des Spursensors aus dem Mecanum-Omniwheels-Fahrzeug ein „Spurfolger“: Das Fahrzeug lernt, autonom entlang einer schwarzen Linie zu fahren.

In der Experimentieraufgabe erhält das Fahrzeug eine Kamera mit Linienerkennung: Als analoger Sensor ermöglicht sie einen Spurfolger mit P- und PD-Regler.

Die Aufgabe baut auf Aufgabe 8 des Robotics TXT 4.0 Base Set auf.

Thema

Digitale Steuerung des Fahrzeugs und proportionale (und PD-) Regelung der Fahrt entlang einer schwarzen Linie; Erkennung und Reaktion auf Hindernisse.

Lernziele

- Einfache Dreipunktregelung unter Verwendung digitaler Sensoren
- Ergänzung einer Hinderniserkennung (Ultraschall-Entfernungsmessung)
- Analoge Regelung unter Verwendung der Linienerkennung (Kamera mit Bildauswertung)
- Kalibrierung eines P- und eines PD-Reglers

Zeitaufwand

In dieser Aufgabe wird das in Aufgabe 1 aufgebaute Mecanum-Omniwheels-Fahrzeug Basismodell mit Sensoren verwendet.

Für die Entwicklung der Programme zur Lösung der Programmieraufgaben benötigen Schülerinnen und Schüler, Vorkenntnisse aus dem Robotics TXT 4.0 Base Set (insbesondere Aufgabe 8) vorausgesetzt, 45-90 Minuten (ein bis zwei Unterrichtsstunden).

In den Experimentieraufgaben werden ein P- und ein PD-Regler entwickelt. Für die Programmierung und die Einstellung der Regler sollten mehrere Schulstunden angesetzt werden (jeweils 90-180 Minuten). Die Zusammenarbeit in Gruppen ist zu empfehlen.

Bezug Curriculum

Land	Stufe/Fächer	Bezüge
BW		
BY		
BE		
BB		
HB		
HH		
HE		
MV		
NI		
NW		
RP		
SL		
SN		
ST		
SH		
TH		

Anlagen
Aufgabe 2: Spurfolger

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, gerader Linie
- Parcours-Bogen mit schwarzer, 2 cm breiter, geschlossener Kreislinie (aus Robotics TXT 4.0 Base Set)
- Hindernis (Karton, Dose, …)

Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Endlicher Automat (Zustandsautomat)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

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

[5] Wikipedia: *Regelungstechnik*.

[6] Wikipedia: *Regler*.

[7] RN-Wissen: *Regelungstechnik*.

[8] Tim Wescott: *PID without a PhD*. Embedded Systems Programming, 10/2000, S. 86-108.

Name: _____ Klasse: ___ Datum: _____

Lösungsblatt Aufgabe 2

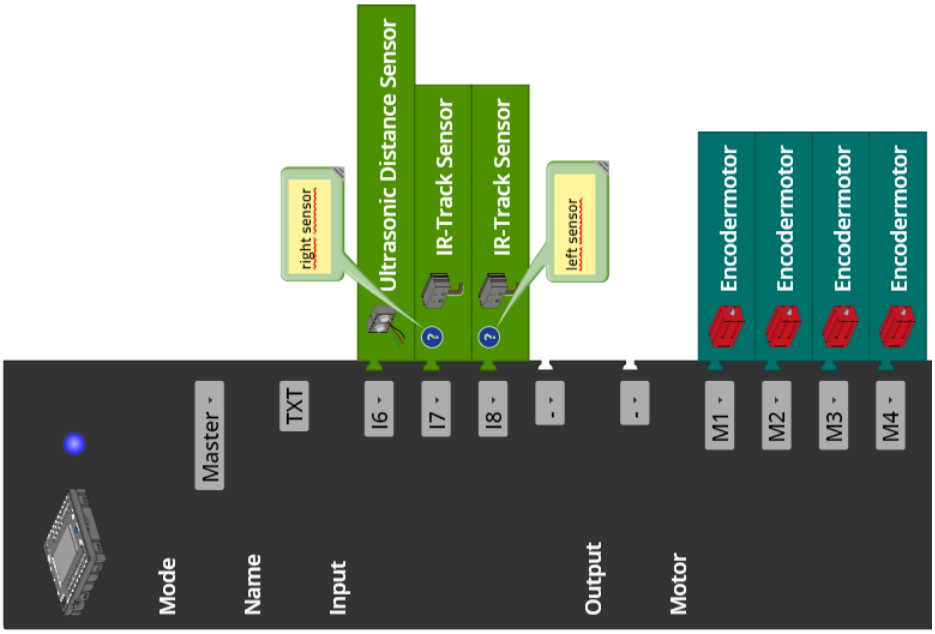
Spurfolger

Die Lösungen der Schülerinnen und Schüler sollen für die Steuerung des Fahrzeugs die Unterfunktionen aus der Aufgabe 1 verwenden. Bei der Programmierung des Spurfolgers soll – wie in der Lösung von Aufgabe 8 des Robotics TXT 4.0 Base Set – eine Zustandsvariable („state“) zur Unterscheidung der Zustände verwendet werden. Damit werden die Programme sehr klar und übersichtlich.

Die beiden Experimentieraufgaben vermitteln das Verständnis für die Entwicklung und Konfiguration eines P- und eines PD-Reglers. Dabei ist vor allem die Ausgabe und grafische Veranschaulichung der Messwerte von besonderer Wichtigkeit.

Konstruktionsaufgabe

Anschluss der Sensoren:

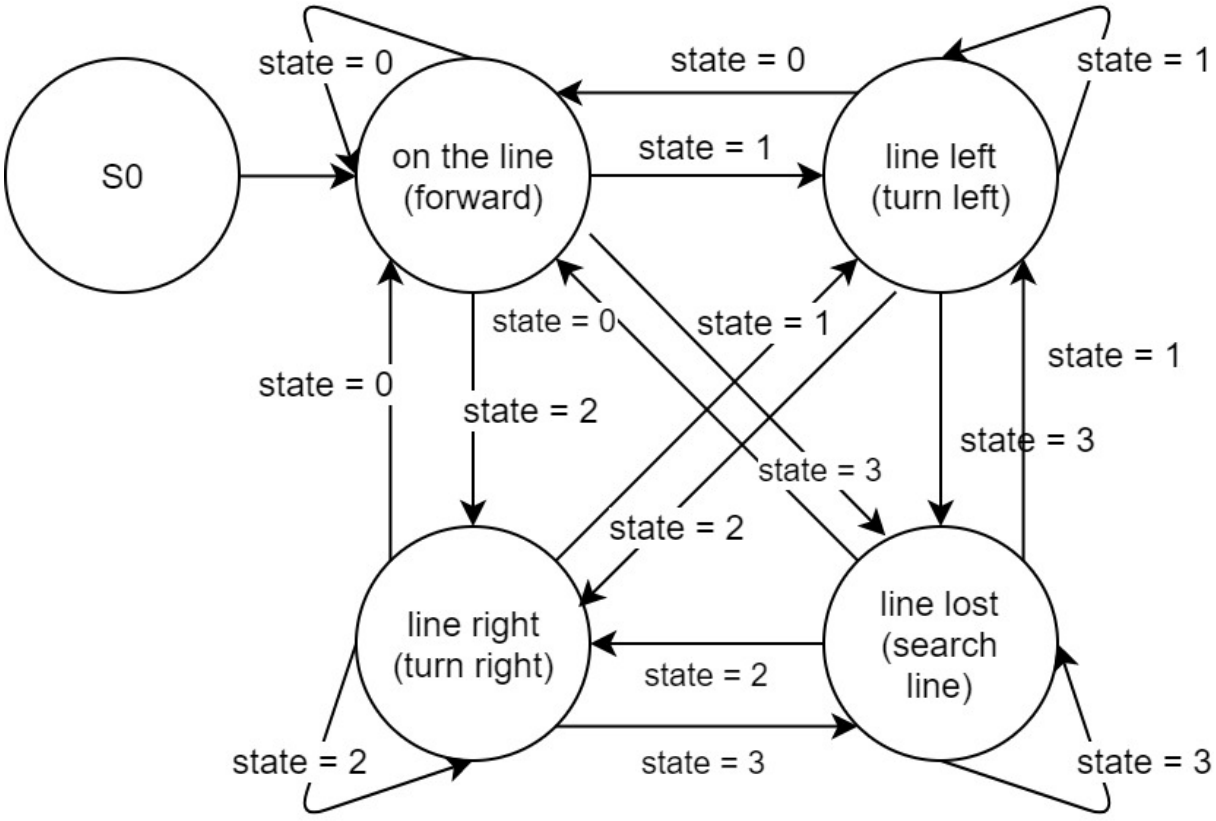


Programmieraufgaben

1. Spurfolger mit Spursensor

Wie der Buggy in Aufgabe 8 des Robotics TXT 4.0 Base Set soll das Mecanum-Omniwheels-Fahrzeug so gesteuert werden, dass die beiden IR-Sensoren des Spursensors mittig über der schwarzen Linie liegen, also beide den Wert 0 liefern.

1a. Zustandsübergangsdiagramm des (digitalen) Spurfolgers:



State-Transition_Diagram_Line_Follower.drawio

1b. Abhängig von den Werten des linken und des rechten IR-Sensors wird die Zustandsvariable *state* mit den folgenden Werten belegt:

- *state* = 0 → auf der Spur (beide Sensoren liefern den Wert 0)
- *state* = 1 → Spur liegt links (nur der linke Sensor hat den Wert 0)
- *state* = 2 → Spur liegt rechts (nur der rechte Sensor hat den Wert 0)
- *state* = 3 → Spur verloren (beide Sensoren liefern den Wert 1)

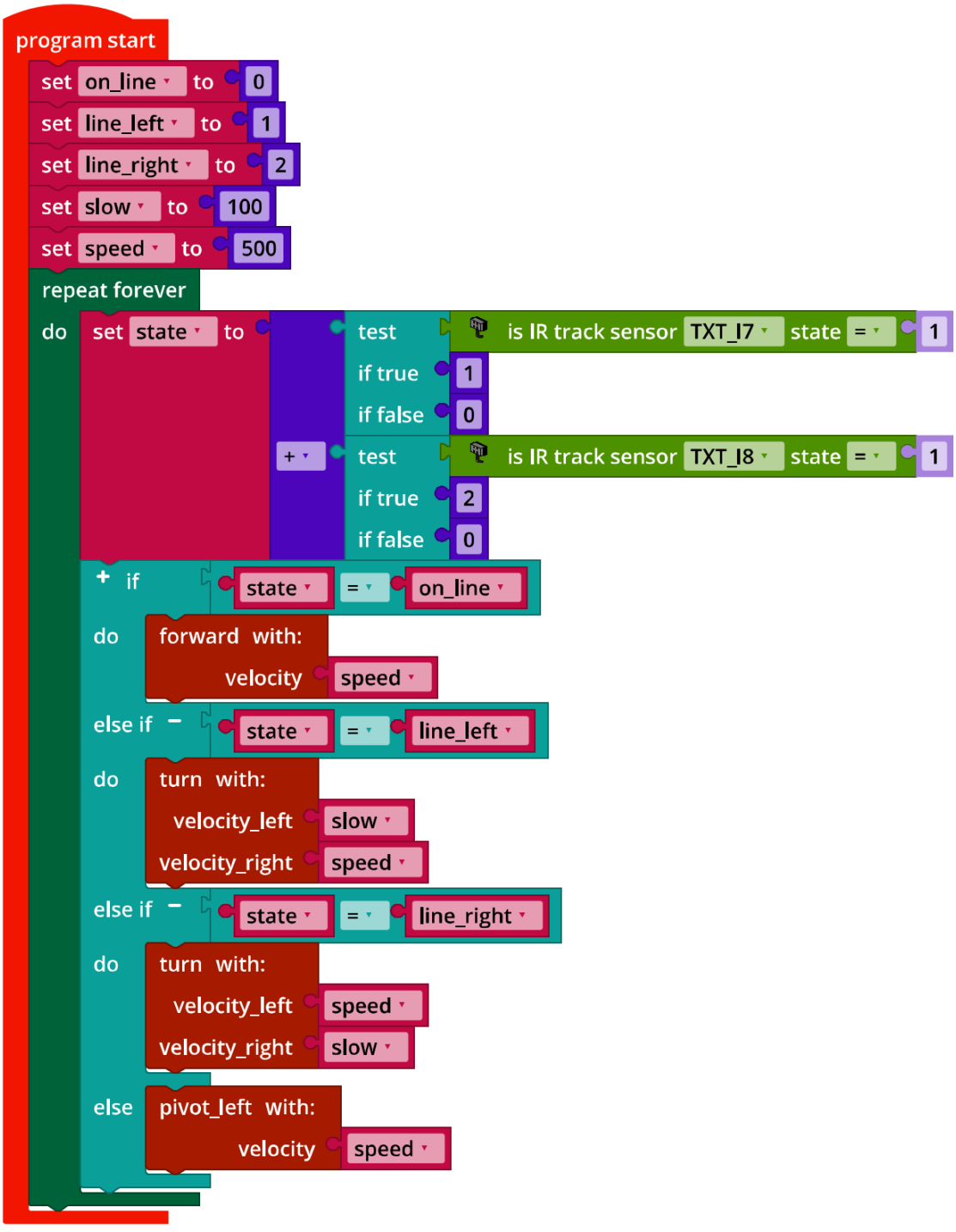
	I8 (linker Sensor)	I7 (rechter Sensor)
Schwarze Linie (Spur)	0 → *state* += 0	0 → *state* += 0
Helle Fläche	1 → *state* += 2	1 → *state* += 1

Lesebeispiel: Liefert der linke Sensor (I8) den Wert 0 und der rechte (I7) den Wert 1, dann liegt die Spur links von der Mitte des Fahrzeugs. Die Zustandsvariable *state* erhält den Wert 1, und in diesem Zustand muss das Fahrzeug nach links gesteuert werden, damit der Sensor wieder mittig über der Spur liegt.

Die folgende Beispiellösung verwendet die in Aufgabe 1 programmierten Navigations-Funktionen des Mecanum-Omniwheels-Fahrzeugs. Dadurch wird das Programm sehr übersichtlich.

Zu Beginn der Schleife wird die Zustandsvariable „state“ abhängig von den Werten der beiden IR-Sensoren auf einen Zustandswert aus {0, 1, 2, 3} gesetzt.

Programmauszug (Beispiel):



Mecanum_Line_Follower_digital.ft

1c. Die Geschwindigkeit des Spurfolgers lässt sich erhöhen, indem die Hauptgeschwindigkeit („speed“) möglichst groß gewählt und der Geschwindigkeitsunterschied

der Motoren beim Lenken verringert, also die Geschwindigkeit „slow“ so angepasst wird, dass der Spurfolger in Kurven gerade nicht die Spur verliert.

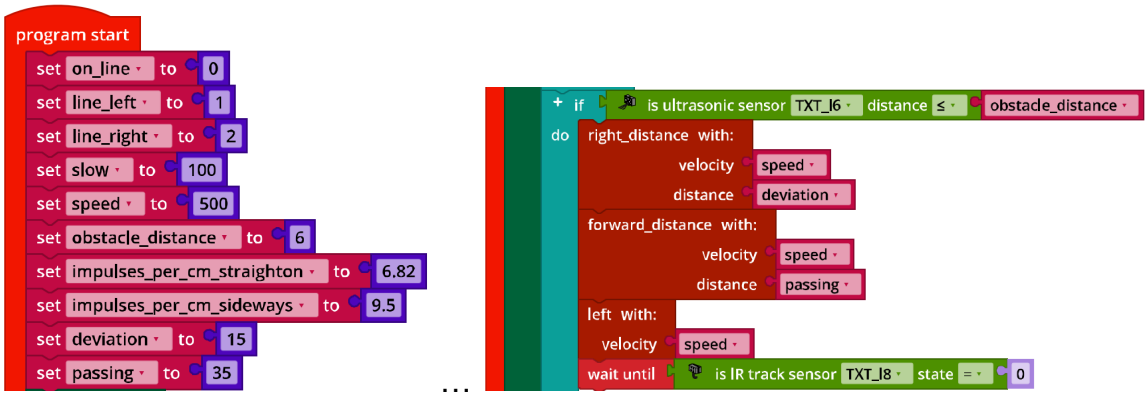
Hinweis: Den digitalen Spurfolger kann man auch als Dreipunktregler verstehen, der abhängig von den drei Zuständen „links von der Spur“, „auf der Spur“ und „rechts von der Spur“ die Geschwindigkeit der Motoren regelt. Der Bereich, in dem beide IR-Sensoren den Sollwert „0“ liefern (beide also direkt über der Spur liegen), wird auch als *Hysterese* bezeichnet.

2. Spurfolger mit Hinderniserkennung

Nach dem Ausweichen kann die Linie hinter dem Hindernis wiedergefunden werden durch eine

- erneute seitliche Fahrt (Nachteil: die Tiefe des Hindernisses ist nicht bekannt, daher kann das Hindernis gestreift oder aber die Linie verfehlt werden, wenn direkt dahinter eine enge Kurve folgt; Vorteil: gerade Linie wird in richtiger Position gefunden)
- Diagonalfahrt (Nachteil: das Hindernis kann gestreift werden; die Linie wird mit höherer Wahrscheinlichkeit verfehlt; Vorteil: eine gerade Linie wird in richtiger Ausrichtung für die Weiterfahrt gefunden)
- Drehung um 45°-90° mit anschließender Geradeausfahrt (Nachteil: das Hindernis kann gestreift, die Linie verfehlt oder auch eine Fortsetzung der Fahrt in der falschen Richtung erfolgen, wenn die Linie gefunden ist)
- Kurvenfahrt um das Hindernis (Nachteil: es kann eine Fortsetzung der Fahrt in der falschen Richtung erfolgen, wenn die Linie gefunden ist)

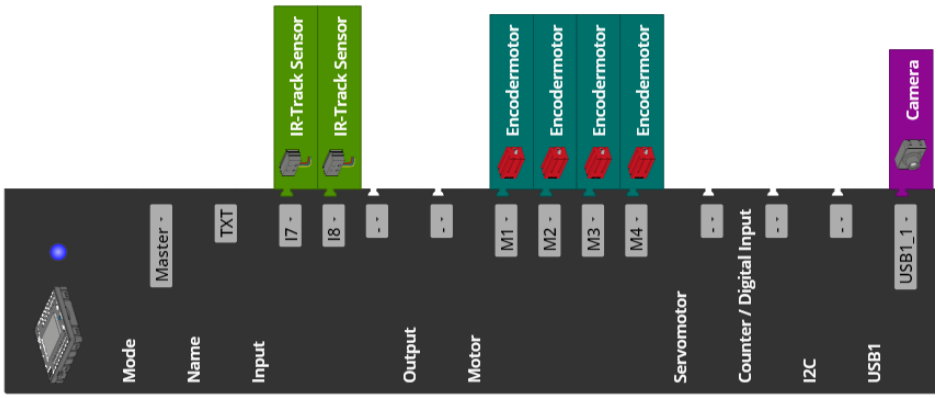
Programmauszug (Beispiel):



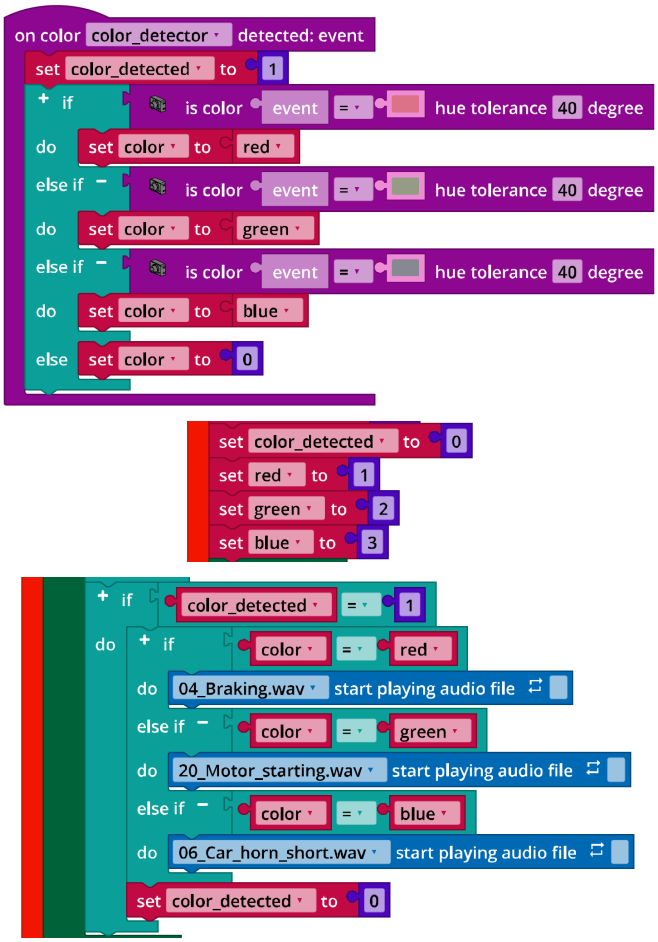
Mecanum_Line_Follower_Obstacle_digital.ft

3. Spurfolger mit Farbsteuerung

Anschluss der Kamera:



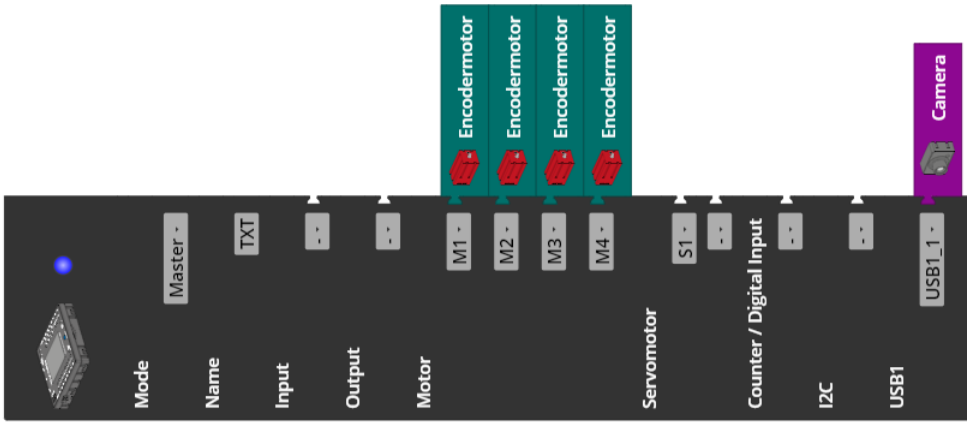
3b. Programmauszug (Beispiel):



Mecanum_Line_Follower_with_Color_Detection_digital.ft

Experimentieraufgaben

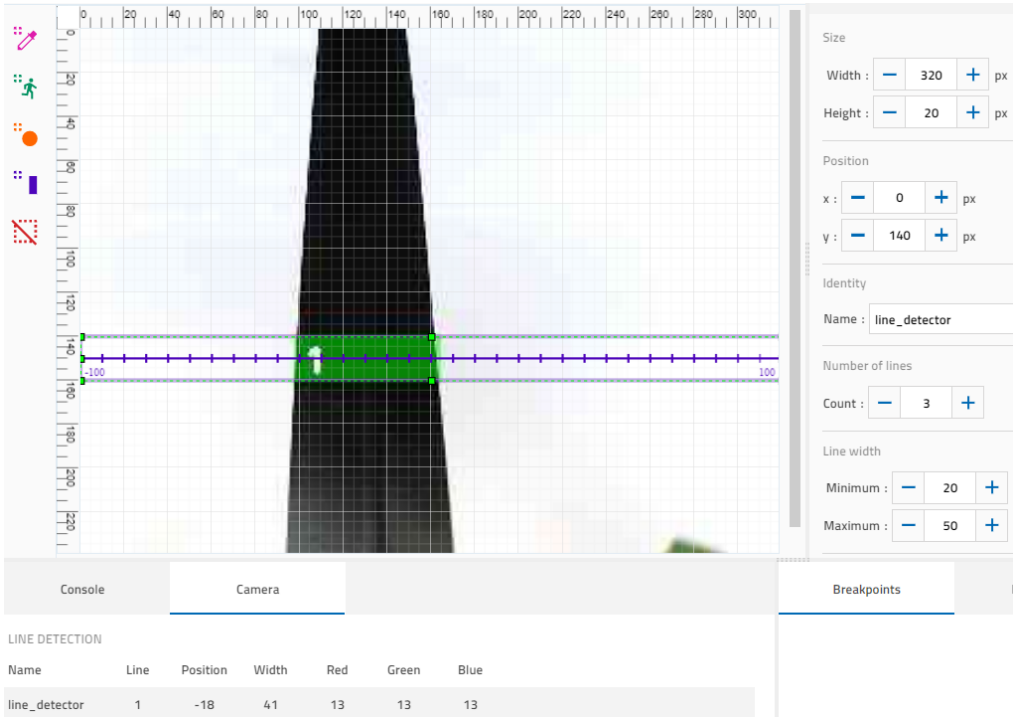
Anschluss der Sensoren:



1. Spurfolger mit Proportionalregler

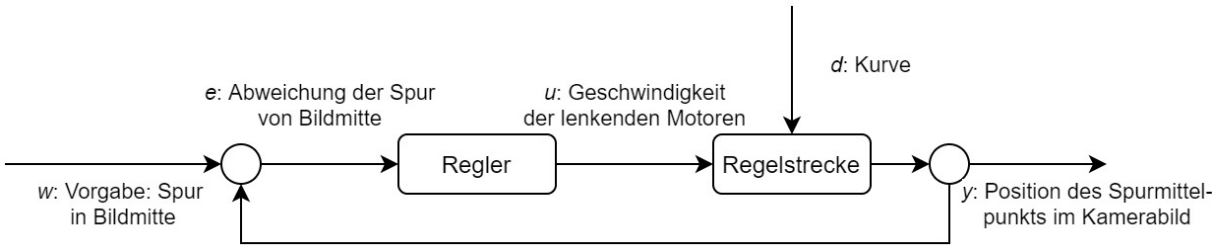
Der Liniendetektor der Kamera liefert die Abweichung von der Mitte des Erkennungsfensters.

Konfiguration der Linienerkennung:



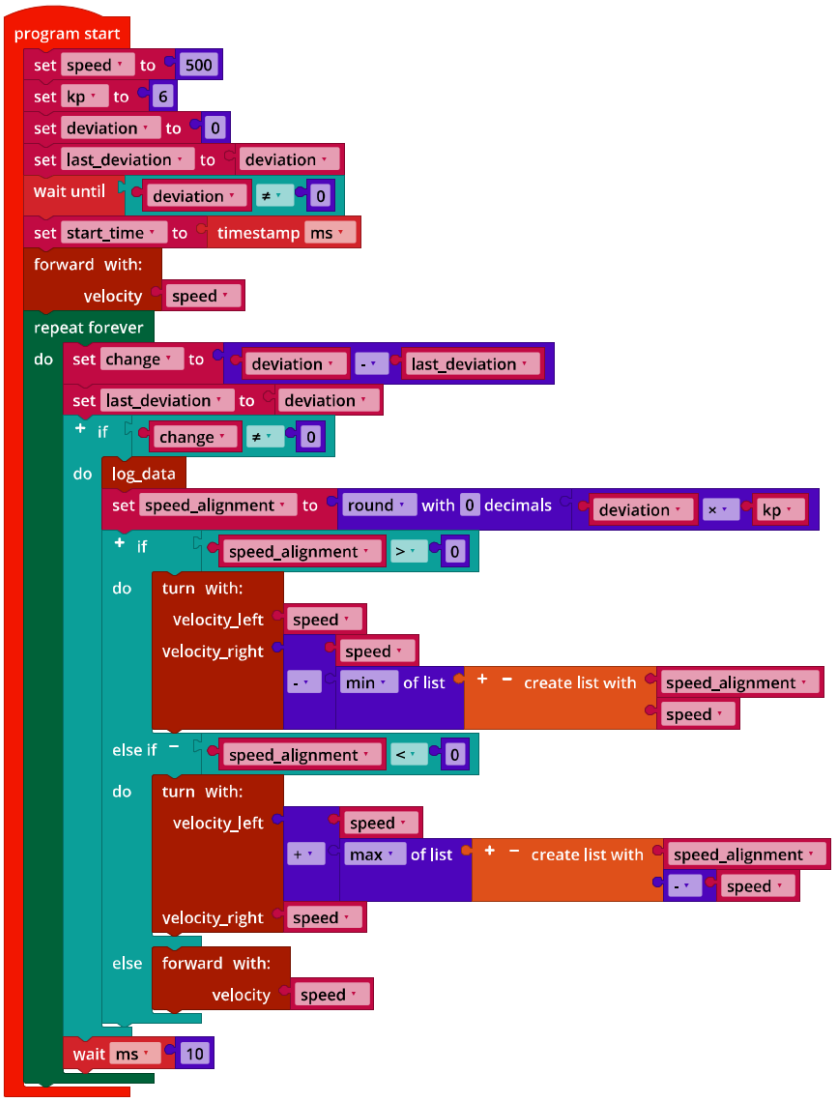
Da Farbflächen als Linie erkannt werden können, sollte die Linienerkennung auf mindestens zwei Linien konfiguriert werden.

1a. Regelkreis:

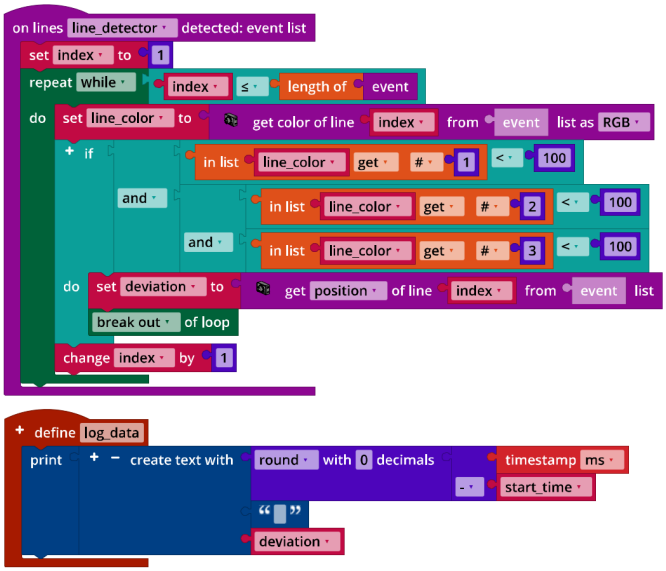


Regelkreis_Mecanum_Spurfolger.drawio

1b. Programmauszug (Beispiel):



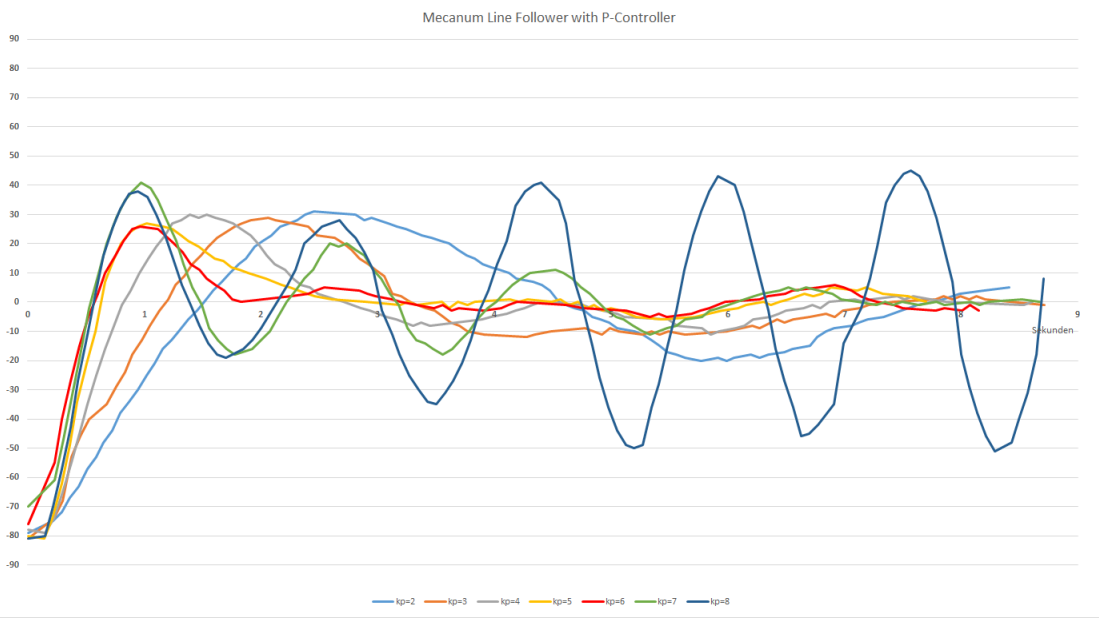
Mecanum_Line_Follower_P_Controller.ft



Mecanum_Line_Follower_P_Controller.ft

Multipliziert mit dem Proportionalitätsfaktor k_p wird die Position so zu der Motorgeschwindigkeit des Links- bzw. Rechtsabbiegens (Funktion *turn*) addiert bzw. subtrahiert, dass mit der Änderung der Fahrtrichtung die Abweichung von der Spur abnimmt. Je nach vertikaler Positionierung des Liniendetektors in der Kamera-Konfiguration müssen ggf. die minimale und maximale Linienbreite im Programm angepasst werden.

1c. Messergebnisse des P-Reglers (mit $k_p \in \{2, 3, 4, 5, 6, 7, 8\}$):



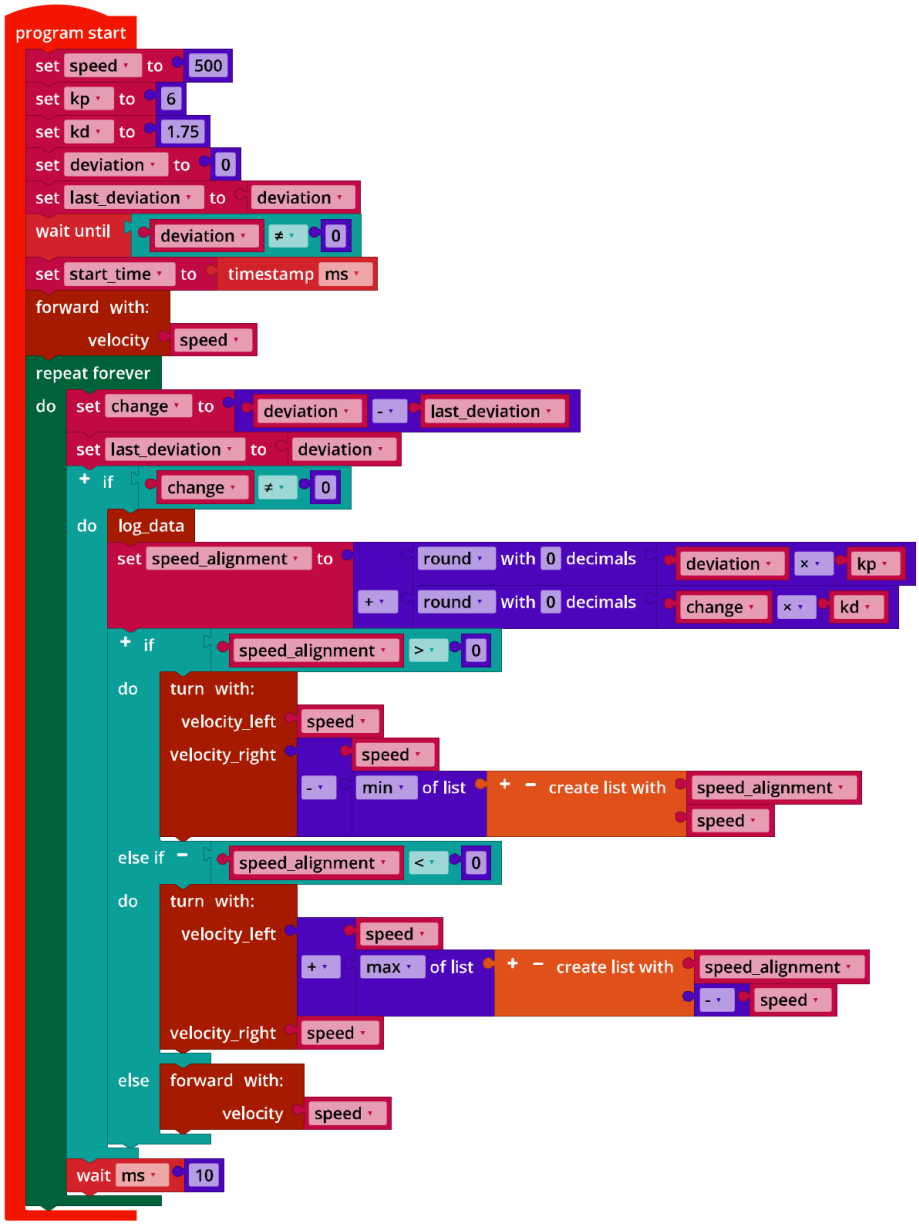
Mecanum_Line_Follower_with_P_Controller_Results.jpg

Die rote Linie mit $k_p = 6$ schwingt nach ca. 1,7 Sekunden am schnellsten ein, ohne ein zweites Mal überzuschwingen. Mit $k_p = 8$ (dunkelblaue Linie) oszilliert der Regler.

2. Spurfolger mit PD-Regler

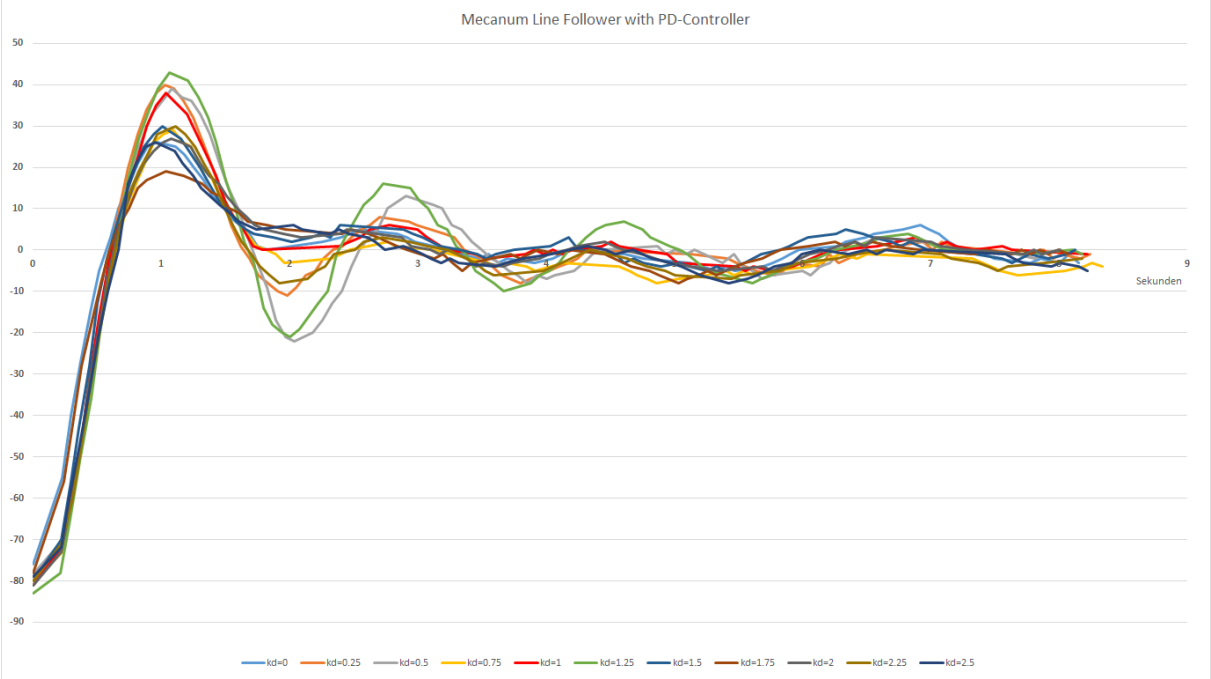
Das D-Glied des PD-Reglers ist die Änderung der Abweichung multipliziert mit dem Differenzial-Faktor k_d.

Programm (Beispiel):



Mecanum_Line_Follower_with_PD_Controller.ft

Messergebnisse des PD-Reglers
(mit $k_d \in \{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5\}$):



Mecanum_Line_Follower_with_PD_Controller_Results.jpg

Mit dem Differential-Faktor $k_d = 1.75$ war die Dämpfung des Überschwingens in den durchgeführten Tests am wirksamsten. Der Wert kann abhängig von der Position der Linienerkennung im Kamerabild und kleinen konstruktiven Unterschieden von Modell zu Modell unterschiedlich ausfallen.

Anlagen
Aufgabe 2: Spurfolger

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, gerader Linie
- Parcours-Bogen mit schwarzer, 2 cm breiter, geschlossener Kreislinie (aus Robotics TXT 4.0 Base Set)
- Hindernis (Karton, Dose, …)

Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Endlicher Automat (Zustandsautomat)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

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

[5] Wikipedia: *Regelungstechnik*.

[6] Wikipedia: *Regler*.

[7] RN-Wissen: *Regelungstechnik*.

[8] Tim Wescott: *PID without a PhD*. Embedded Systems Programming, 10/2000, S. 86-108.

Name: _____ Klasse: ___ Datum: _____

Aufgabe 4

Malroboter

Das Mecanum-Omniwheels-Fahrzeug erhält nun einen Stift – und wird damit zum Malroboter.

Konstruktionsaufgabe

Konstruiere den Malroboter nach der Anleitung bzw. baue das Mecanum-Omniwheels-Fahrzeug aus der Aufgabe 1, 2 oder 3 zum Malroboter um. Schließe die Encoder-Motoren und den Servo-Motor wie im Verkabelungsplan angegeben an.

Prüfe mit dem Interface-Test oder deinen Steuerprogrammen aus Aufgabe 1, ob alle Motoren richtig angeschlossen sind.

Wichtig: Befestige zuerst den Stift (Fineliner, Filzstift) so in der Halterung, dass die Mine auf dem Papier aufsitzt. Starte dann den TXT. Dabei wird der Servo automatisch auf mittlere Stellung eingestellt. Stecke anschließend den Servo-Hebel so auf den Servo, dass er etwa horizontal nach vorne zeigt und den Stift mit der Stifthalterung gerade nicht anhebt.

Teste die Absenkung des Stifts über den Servo mit dem Interface-Test.

Achtung: Wenn du den Servo bei eingeschaltetem TXT mit der Hand bewegst, kannst du ihn beschädigen.

Programmieraufgaben

1. Stiftabsenkung

Damit der Malroboter sich auch bewegen kann, ohne eine Linie zu zeichnen, muss der Stift mit dem Servo-Motor angehoben werden.

Ergänze deine Funktionsbibliothek für das Mecanum-Omniwheels-Fahrzeug um eine Funktion für das Anheben und das Absenken des Stifts. Wähle die Servo-Position mit Hilfe des Interface-Tests.

2. Haus vom Nikolaus

Mit den Navigationsfunktionen aus Aufgabe 1 kannst du den Malroboter nun das „Haus vom Nikolaus“ in einem Zug zeichnen lassen.

Zum Schluss soll der Malroboter den Stift anheben und ein Stück zur Seite fahren.

3. n-Eck

Nun soll der Malroboter (wie in Aufgabe 7 des Robotics TXT 4.0 Base Set) ein beliebiges n-Eck mit fester Kantenlänge zeichnen. Versuche, das Programm so allgemein und – unter Verwendung von Schleifen – so kompakt wie möglich zu gestalten.

Teste das Programm, indem du den Malroboter nacheinander ein Dreieck, ein Viereck, ein Fünfeck, … und schließlich ein 15eck zeichnen lässt.

Tipp: Wähle die Kantenlänge nicht zu groß.

Experimentieraufgaben

1. Zielpunkt ansteuern

Der Malroboter soll nun von seinem Standpunkt aus zu einem durch Koordinaten definierten (Ziel-) Punkt (x, y) fahren. Nimm dafür an, dass sich der Stift beim Starten des Programms im Nullpunkt $(0, 0)$ des Koordianten-Bezugssystems befindet und der Roboter entlang der (positiven) x-Achse ausgerichtet ist.

1a. Welche Bewegungen muss der Malroboter absolvieren, damit er von seiner Position (dem Nullpunkt) auf kürzestem Weg zu einem vorgegebenen Punkt (x, y) fährt? Welche Berechnungen sind dafür erforderlich?

Veranschauliche deine Überlegungen mit einer Zeichnung.

1b. Schreibe ein Programm unter Verwendung der Navigationsfunktionen aus Aufgabe 1, das den Malroboter auf dem kürzesten Weg zu dem vorgegebenen Punkt (x, y) fahren lässt. Teste dein Programm mit zuvor ausgemessenen Punkten.

2. „Malen nach Zahlen“

In dem Programm-Template „*Mecanum_Drawing_Coordinates.ft*“ findest du zwei Listen mit x- und y-Koordinaten sowie eine mit „up/down“-Flags. Die x- und y-Koordinaten bezeichnen jeweils einen Punkt im Koordinatensystem mit einem Punkt in der linken unteren Ecke des Papiers als Ausgangspunkt $(0, 0)$.

Platziere den Malroboter so, dass der Stift direkt über dem Punkt $(0, 0)$ zu liegen kommt und richte den Roboter parallel zur unteren Papierkante nach rechts aus. Das Zeichenpapier sollte mindestens 60 cm breit und 40 cm hoch sein.

2a. Erweitere dein Steuerungsprogramm für den Malroboter aus Experimentier-aufgabe 1 so, dass die Punkte (Koordinaten) der Liste nacheinander angefahren werden. Falls das zur Koordinate gehörige „up/down“-Flag gleich 0 ist, soll der Malroboter die Strecke mit angehobenem Stift fahren; ist es gleich 1, dann soll er eine Linie dorthin zeichnen. Die Größe des Bildes kannst du einstellen, indem du die Koordinaten mit einem festen Faktor multiplizierst.

2b. Mache ein Foto von dem gezeichneten Bild. Was stellt es dar?

2c. Nun kannst du dem Malroboter eigene Koordinatenlisten für ein Bild vorgeben.

Anlagen
Aufgabe 4: Malroboter

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.
- Stift (Fineliner, Filzstift), großes weißes Blatt Papier
- Programm-Template „*Mecanum_Drawing_Coordinates.ft*“

Weiterführende Informationen

[1] Suche nach „Coordinate Grid Picture“ im Internet.

[2] Oliver Boorman: *Cartesian Grid Image Generator*.

Aufgabe 4: Malroboter

Aus dem Mecanum-Omniwheels-Fahrzeug wird nun ein Malroboter. Er lernt, geometrische Formen nach Vorgabe mit einem Filzstift oder Fineliner zu zeichnen, und ihm wird das „Malen nach Zahlen“ beigebracht.

Die Aufgabe baut auf Aufgabe 7 des Robotics TXT 4.0 Base Set auf.

Thema

Steuerung des Mecanum-Omniwheels-Fahrzeugs zum Zeichnen geometrischer Formen und vorgegebener Linien.

Lernziele

- Nutzung von Funktionen zur übersichtlichen Programmgestaltung
- Modellbildung und Berechnung der Fahrzeugsteuerung (Trigonometrie)
- Gestaltung von Datenstrukturen

Zeitaufwand

Für den Umbau des Mecanum-Omniwheels-Fahrzeugs aus Aufgabe 1 und 2 in einen Malroboter benötigen die Schülerinnen und Schüler ca. 45 Minuten, bei einer kompletten Neukonstruktion nach Bauanleitung bis zu 75 Minuten (Erfahrung mit fischertechnik vorausgesetzt).

Für die Entwicklung des Steuerungsprogramms zur Lösung der Programmieraufgaben benötigen Schülerinnen und Schüler, Vorkenntnisse aus dem Robotics TXT 4.0 Base Set (insbesondere Aufgabe 7) vorausgesetzt, etwa 90 Minuten. Der Zeitaufwand für die Lösung der Experimentieraufgaben liegt – abhängig von Alter und Erfahrung – bei 135-180 Minuten.

Bezug Curriculum

Land	Stufe/Fächer	Bezüge
BW		
BY		
BE		
BB		
HB		
HH		
HE		
MV		
NI		
NW		
RP		
SL		
SN		
ST		
SH		
TH		

Anlagen
Aufgabe 4: Malroboter

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.
- Stift (Fineliner, Filzstift), großes weißes Blatt Papier
- Programm-Template „*Mecanum_Drawing_Coordinates.ft*“

Weiterführende Informationen

[1] Suche nach „Coordinate Grid Picture“ im Internet.

[2] Oliver Boorman: *Cartesian Grid Image Generator*.

Name: _____ Klasse: ___ Datum: _____

Lösungsblatt Aufgabe 4

Malroboter

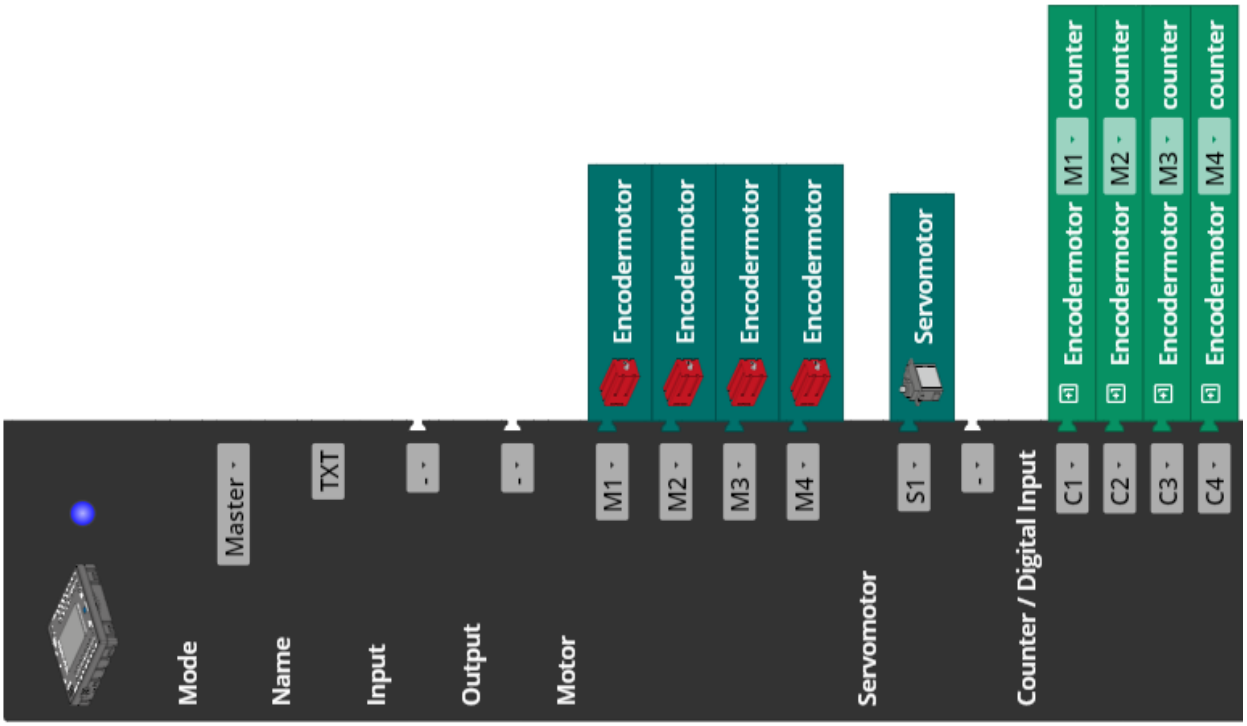
Die Aufgaben sind eine Übung zur Nutzung von Schleifen und Funktionen mit dem Ziel, ein übersichtliches, kompaktes und verständliches Programm zu erhalten. Die Ergebnisse der Schülerinnen und Schüler sollten diesbezüglich miteinander verglichen werden.

Konstruktionsaufgabe

Siehe Bauanleitung.

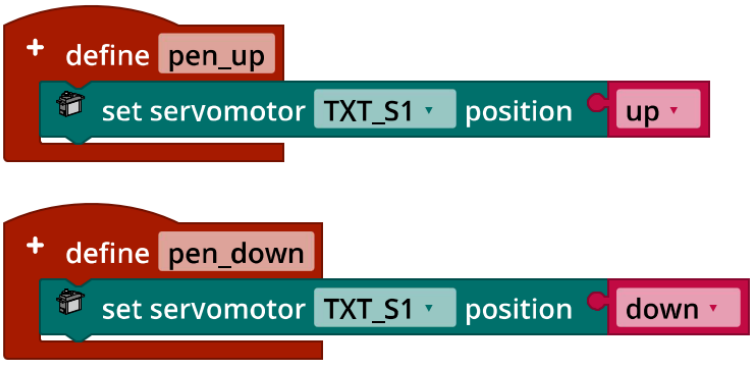
Programmieraufgaben

Anschluss der Aktoren:



1. Stiftabsenkung

Programmauszug (Beispiel):



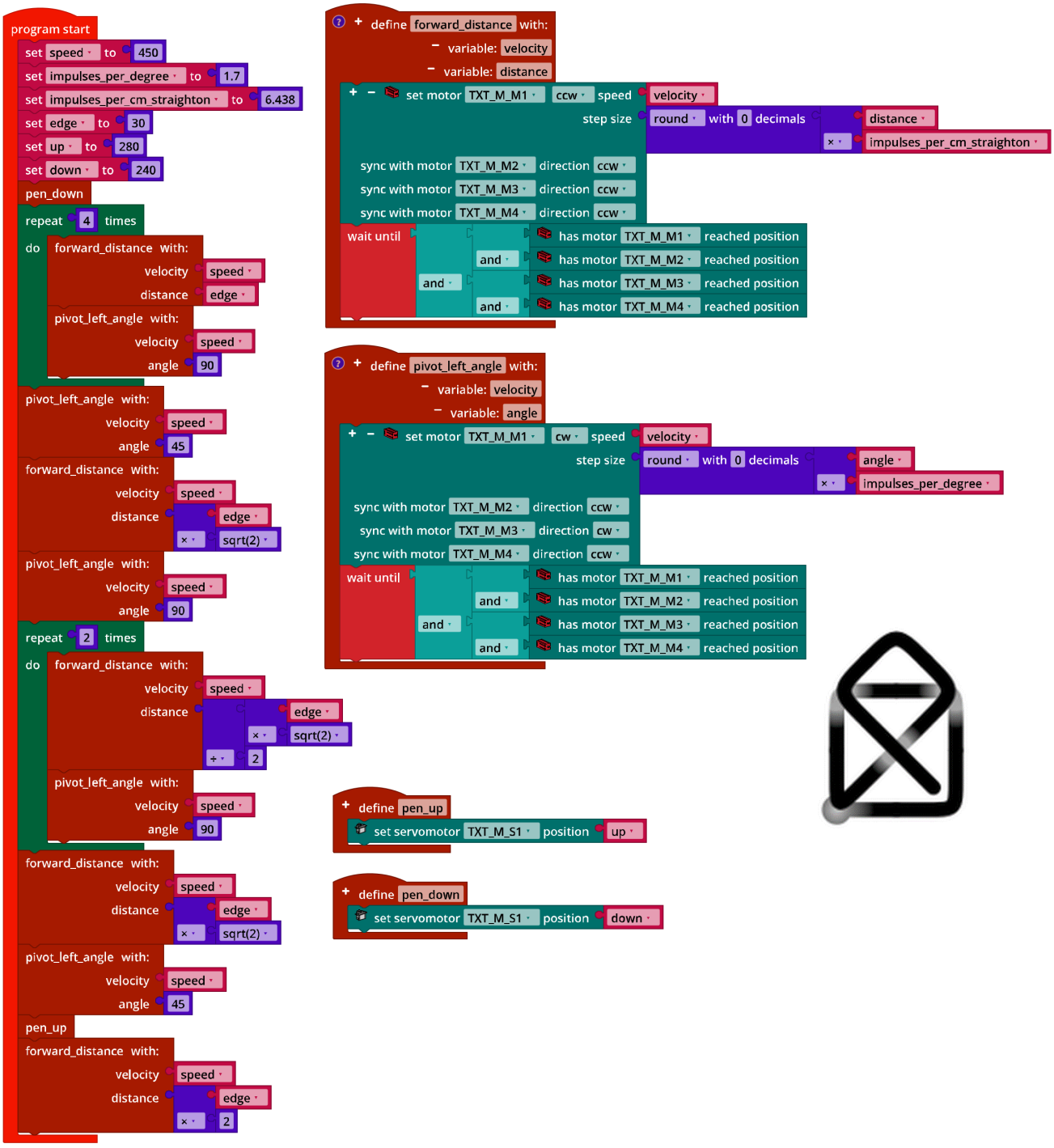
Die Werte für die Variablen „up“ und „down“ können mit dem Interface-Test bestimmt werden. Sie sind modellabhängig und hängen insbesondere von der Position des aufgesteckten Servo-Arms ab.

2. Haus vom Nikolaus

Das „Haus vom Nikolaus“ lässt sich in acht Strichen zeichnen, ohne den Stift vom Papier abzuheben. Tatsächlich gibt es sogar 88 verschiedene korrekte Möglichkeiten dafür.

Das folgende Programmbeispiel zeichnet es mit einer Kantenlänge von 30 cm und verwendet die Funktionen „forward_distance“ und „pivot_left_angle“ aus Aufgabe 1.

Programm (Beispiel):

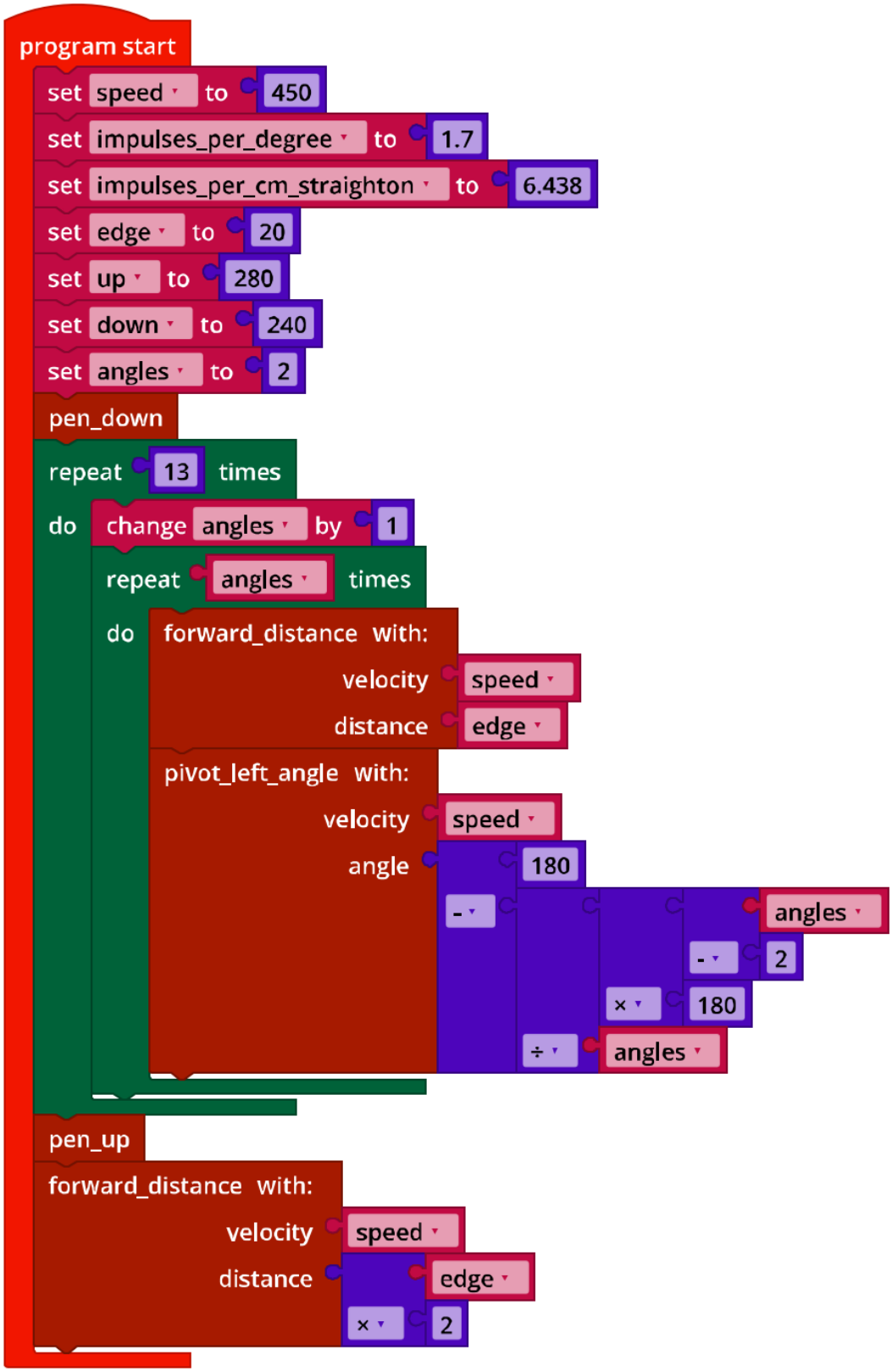


Mecanum_House_of_Santa_Claus.ft

3. n-Eck

Die Summe der Innenwinkel eines n-Ecks beträgt $(n-1)\cdot 180°$. Um ein n-Eck zu zeichnen muss das Mecanum-Omniwheels-Fahrzeug nach jeder Kante also um $180° - \frac{(n-1)\cdot 180°}{n}$ drehen. Die Beispiellösung verallgemeinert die Aufgabenstellung zur Zeichnung von n-Ecken und zeichnet zunächst ein Dreieck, dann ein Viereck bis zu einem 15-Eck mit einer Kantenlänge von jeweils 20 cm. Durch zwei geschachtelte Schleifen wird das Programm sehr kompakt.

Programmauszug (Beispiel):

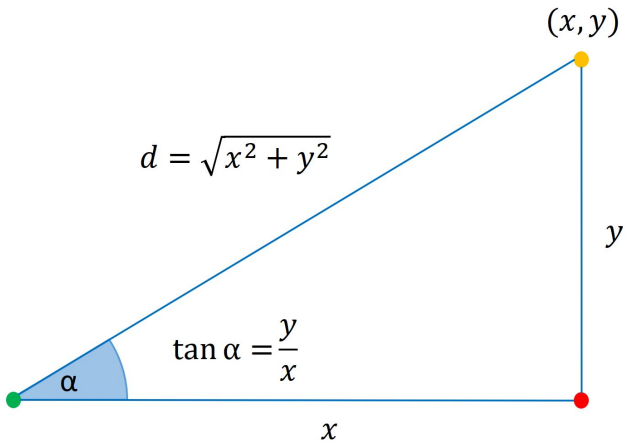


Mecanum_Drawing_Polygons.ft

Experimentieraufgaben

1. Zielpunkt ansteuern

1a. Um vom Punkt $(0, 0)$ zu einem Punkt (x, y) zu gelangen muss sich der Malroboter entlang der Hypotenuse eines rechtwinkligen Dreiecks mit den Kathetenlängen x und y bewegen.



Mathematical_Model_Coordinates_Drawing.jpg

Die Länge der Hypotenuse erhält man mit dem Satz des Pythagoras:

$$d = \sqrt{x^2 + y^2}$$

Der Innenwinkel α des rechtwinkligen Dreiecks lässt sich mit ein wenig Trigonometrie leicht bestimmen:

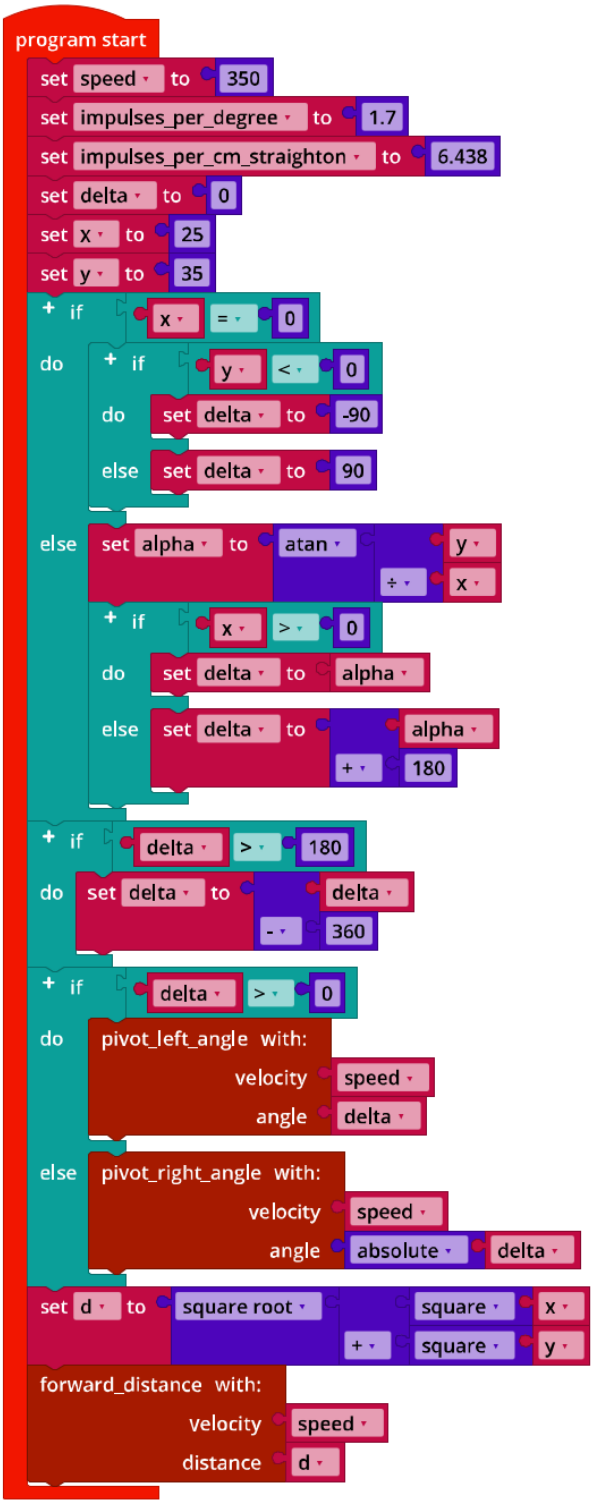
$$\tan \alpha = \frac{y}{x}$$

Daraus muss noch der Drehwinkel δ – bezogen auf die x-Achse – abgeleitet werden. Das geht am einfachsten mit einer Fallunterscheidung:

- Sofern $x > 0$: $\delta = \alpha$
- Falls $x < 0$: $\delta = 180° + \alpha$

Der Fall $x = 0$ muss gesondert behandelt werden, damit keine Division durch 0 erfolgt. Für $y > 0$ ist in diesem Fall $\delta = 90°$, anderenfalls gilt $\delta = -90°$.

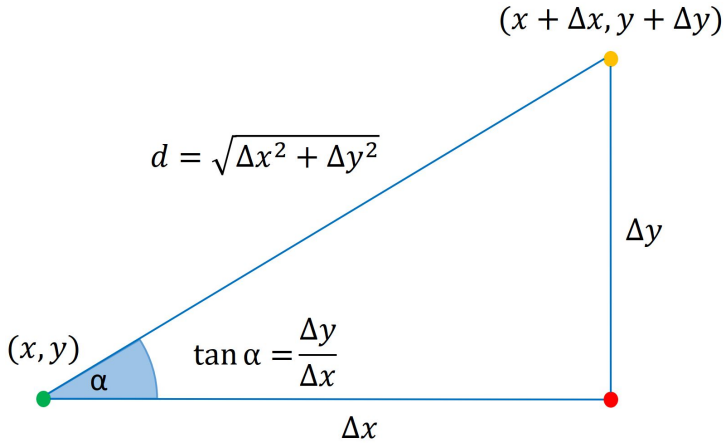
1b. Programm (Beispiel):



Mecanum_Move_2_Point.ft

2. „Malen nach Zahlen“

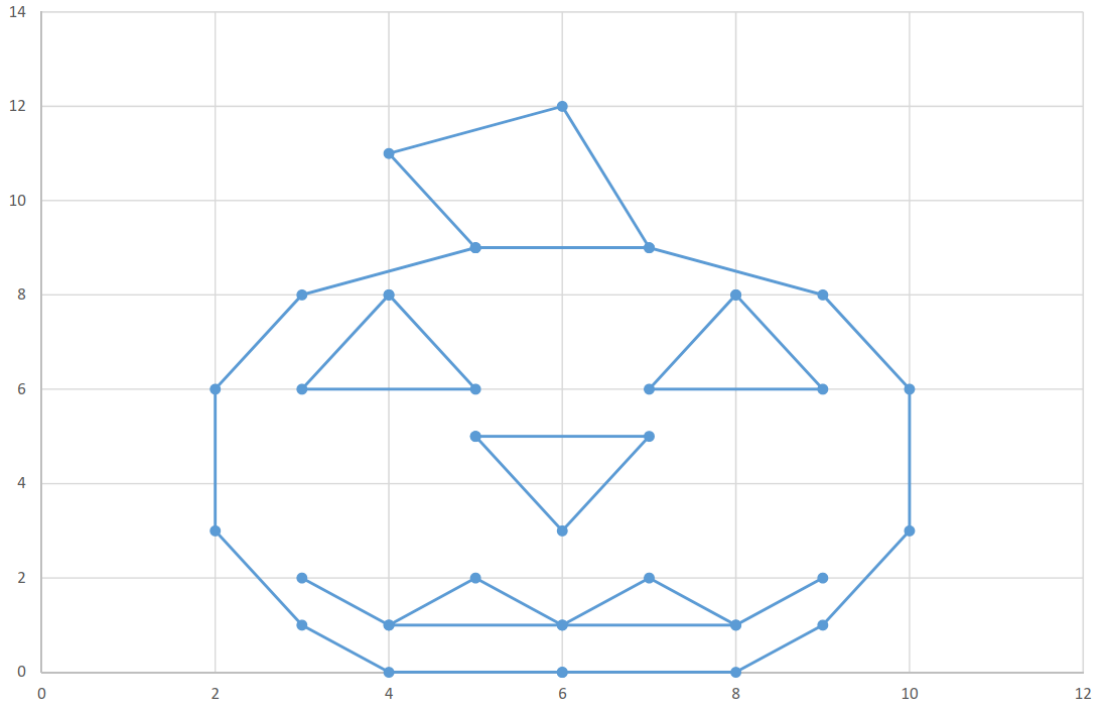
Im allgemeinen Fall ist der Malroboter nicht entlang der x-Achse ausgerichtet, sondern steht in einem Winkel γ zur x-Achse; der Drehwinkel δ ist zu diesem Winkel in Beziehung zu setzen. Der Abstand zum (nächsten) Zielpunkt muss zudem in Relation zur Position des Roboters berechnet werden ($(\Delta x, \Delta y)$).



Mathematical_Model_Coordinates_Drawing_generalised.jpg

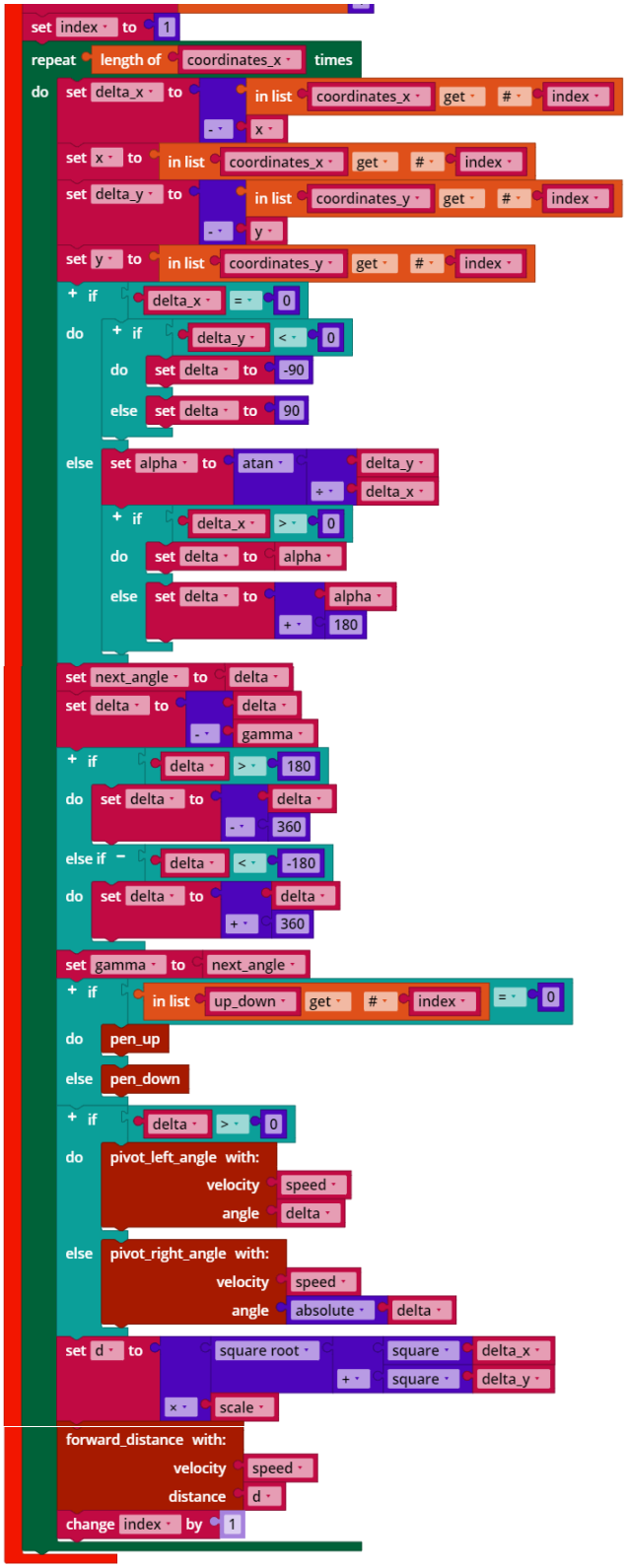
Die abzufahrenden Punkte sind in drei Listen mit den jeweiligen x- und y-Koordinaten und einem „up/down“-Flag gespeichert, das angibt, ob der Stift auf dem Weg zu dem von den Koordinaten bezeichneten Punkt gesenkt (1) oder angehoben (0) sein soll.

Die Koordinaten ergeben die folgende Zeichnung:



Pumpkin.jpg

Programmauszug (Beispiel):



Mecanum_Coordinates_Drawing.ft

Anlagen

Aufgabe 4: Malroboter

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.
- Stift (Fineliner, Filzstift), großes weißes Blatt Papier
- Programm-Template „*Mecanum_Drawing_Coordinates.ft*“

Weiterführende Informationen

[1] Suche nach „Coordinate Grid Picture“ im Internet.

[2] Oliver Boorman: *Cartesian Grid Image Generator*.

Name: _____ Klasse: ___ Datum: _____

Aufgabe 5

Ballroboter

Das Mecanum-Omniwheels-Fahrzeug wird in dieser Aufgabe mit einer Abschussvorrichtung für Kunststoffhohlkugeln ausgestattet, erhält eine Sprachsteuerung und wird mit Hilfe der Kamera um eine autonome Zielausrichtung ergänzt.

Konstruktionsaufgabe

Konstruiere den Ballroboter nach der Bauanleitung bzw. baue das Mecanum-Omniwheels-Fahrzeug aus einer der anderen Aufgaben entsprechend um. Schließe die Encoder-Motoren, die Kamera und den Servo-Motor wie im Verkabelungsplan angegeben an.

Prüfe mit dem Interface-Test oder deinen Steuerprogrammen aus Aufgabe 1, ob alle Motoren richtig angeschlossen sind.

Wichtig: Befestige den Servo-Hebel erst, nachdem du den TXT gestartet hast. Dabei wird der Servo automatisch auf eine mittlere Position eingestellt. Stecke anschließend den Servo-Hebel so auf den Servo, dass er etwa „geradeaus“ (im Modell also nach links) zeigt und hinter der grünen Auswurf-Strebe zu liegen kommt.

Achtung: Wenn du den Servo-Hebel bei eingeschaltetem TXT mit der Hand bewegst, kann der Servo irreparabel beschädigt werden!

Programmieraufgaben

1. Auswurfmechanismus

Der Auswurf einer Kunststoffhohlkugel wird ausgelöst, indem der Servo-Hebel so weit nach hinten bewegt wird, bis er die gespannte grüne Schuss-Strebe freigibt. Dabei rutscht zugleich automatisch eine Kunststoffhohlkugel aus dem Magazin in die Auswurfposition. Anschließend muss der Servo-Hebel wieder vor die Auswurf-Strebe bewegt werden.

1a. Bestimme mit dem Interface-Test geeignete Positionen für den Servo-Hebel zum Abschuss einer Kunststoffhohlkugel und zum erneuten „Vorspannen“ der Auswurf-Strebe.

1b. Ergänze deine Funktionsbibliothek für das Mecanum-Omniwheels-Fahrzeug um eine Funktion für das Auswerfen einer Kunststoffhohlkugel.

1c. Befestige seitlich an deinem Fahrzeug einen Taster und schließe ihn an I1 an. Schreibe ein Blockly-Programm, das bei Betätigung des Tasters eine Kunststoffhohlkugel auswirft.

1d. Erweitere dein Programm um eine Füllstandsanzeige des Magazins auf dem Display des TXT. Fordere zum Nachladen auf, wenn alle Kunststoffhohlkugeln ausgeworfen sind, und lass‘ das erfolgte Nachladen über den Taster bestätigen.

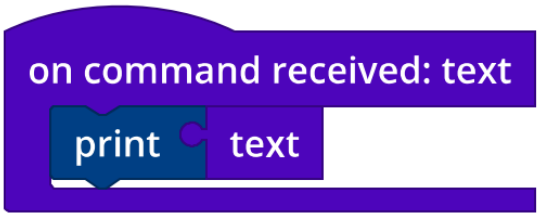
2. Sprachsteuerung

Du kannst dein Mecanum-Omniwheel-Fahrzeug auch über Sprach-Kommandos steuern. Lade dazu die App „Voice Control“ aus dem Apple-App-Store (für iOS) oder dem Google-Play-Store (für Android) herunter und verbinde sie mit dem TXT 4.0.

- Verbindung über WLAN: Der TXT 4.0 Controller und das Gerät (Smartphone oder Tablet) müssen mit demselben WLAN-Router verbunden werden. Der Router muss außerdem die Kommunikation der Geräte untereinander erlauben. Die IP-Adresse des TXT 4.0, mit der die App verbunden werden muss, ist dann über das Menu des Touch-Screen unter „Info“ / „WLAN“ abfragbar.

- Verbindung mit WLAN AP: Am TXT 4.0 kann statt „WLAN“ die Option „Access Point“ unter „Einstellungen“ / „Netzwerk“ aktiviert werden. Dann kann das Smartphone direkt mit dem Controller verbunden werden. Der für die WLAN-Verbindung benötigte WPA2-Key kann im TXT-Menu unter „Access Point“ abgelesen (oder geändert bzw. deaktiviert) werden.

Wenn du die App mit dem Controller verbunden hast, werden die Sprachkommandos als Text an den Controller übertragen und du kannst sie mit der folgenden Event-Funktion auswerten:



Ersetze in deinem Programm aus Programmieraufgabe 1 die Funktion des Tasters zum Auslösen eines Schusses durch ein geeignetes Sprachkommando, das dein Smartphone gut erkennt.

Experimentieraufgaben

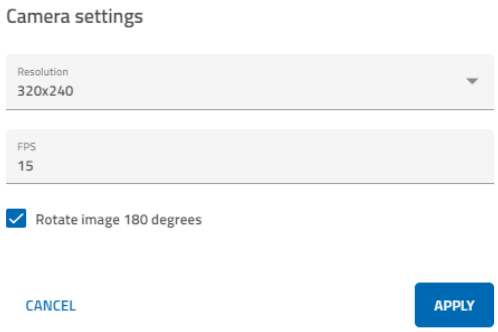
In den folgenden vier Aufgaben wird das Mecanum-Omniwheel-Fahrzeug Schritt für Schritt mit einer automatischen Zielsuche ausgestattet.

Baue dafür zunächst eine farbige Zielscheibe nach der Bauanleitung und stelle sie in einer Entfernung von rund 50 cm vor deinem Mecanum-Omniwheel-Fahrzeug auf.

1. Zielscheibe

1a. Aktiviere die Kamera in der Kamera-Konfiguration. Konfiguriere die Ball-Erkennung so, dass die Mitte der Zielscheibe genau dann in der Mitte des Erkennungsfensters liegt, wenn die Schusseinrichtung möglichst zuverlässig die Zielscheibe trifft. Teste die Einstellung mit deinem Programm aus Programmieraufgabe 1.

Hinweis: Die Kamera steht auf dem Kopf, daher muss das Bild in den Kamera-Einstellungen um 180° gedreht werden:



1b. Wenn du das Fahrzeug näher an die Zielscheibe heranbewegst oder weiter von ihr entfernst, ändert sich die y-Koordinate der Ball-Erkennung. Trage in die folgende Tabelle zum Abstand von der idealen Position die zugehörige y-Koordinate ein.

Abstand von idealer Distanz	y-Koordinate
15 cm	
10 cm	
5 cm	
0 cm	
-5 cm	
-10 cm	
-15 cm	

1c. Leite aus dieser Messung eine einfache Näherungsformel ab, mit der du ausrechnen kannst, um wie viele cm das Mecanum-Omniwheel-Fahrzeug vor oder zurück fahren muss, damit er den idealen Abstand zur Zielscheibe hat.

2. Zielabstandskorrektur

Schreibe ein Blockly-Programm, das das Mecanum-Omniwheel-Fahrzeug die exakt richtige Entfernung zur Zielscheibe einnehmen lässt. Verwende dazu deine Ergebnisse aus Experimentieraufgabe 1.

Skizziere zunächst ein Zustandsübergangsdiagramm.

3. Zielausrichtung

Schreibe ein Blockly-Programm, das das Mecanum-Omniwheel-Fahrzeug so dreht, dass die Zielscheibe genau in der Mitte des Erkennungsfensters liegt.

Zeichne zunächst ein Zustandsübergangsdiagramm für deine Lösung.

Tipp: Die Kamera hat einen Öffnungswinkel von 60°.

4. Zielsuche

Jetzt sollen deine in den vorausgegangenen Teilaufgaben entwickelten Lösungen zusammengeführt werden. Schreibe ein Blockly-Programm, das das Mecanum-Omniwheel-Fahrzeug zunächst nach einer Zielscheibe suchen lässt. Hat es sie entdeckt, soll es die Zielscheibe exakt in die Mitte des Fadenkreuzes nehmen und mit den drei Kunststoffhohlkugeln aus dem Magazin die Zielscheibe treffen.

Veranschauliche dein Lösungskonzept zunächst mit einem Zustandsübergangsdiagramm.

Zum Abschluss kannst du den TXT 4.0 einen Sound abspielen lassen.

.

Anlagen
Aufgabe 5: Ballroboter

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.
- Styroporkugeln

Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Endlicher Automat (Zustandsautomat)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

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

Aufgabe 5: Ballroboter

Das Mecanum-Omniwheels-Fahrzeug erhält einen Abschussmechanismus für Styroporbälle, eine Sprachsteuerung und eine Kamera zur eigenständigen Zielaus-richtung.

Thema

Bildauswertung zur Abstandsbestimmung und Zielausrichtung.

Lernziele

- Experimentelle Analyse einer Aufgabenstellung
- „Computational Thinking“: Zerlegung einer komplexen Aufgabenstellung in überschaubare und lösbare Teilaufgaben, die abschließend zusammengeführt werden (Prinzip „teile und herrsche“)

Zeitaufwand

Für den Umbau des Mecanum-Omniwheels-Fahrzeugs aus den vorausgegangenen Aufgaben in einen Ballroboter benötigen die Schülerinnen und Schüler ca. 90 Minuten, bei einer kom¬pletten Neukon¬struktion nach Bauanleitung bis zu 120 Minuten (Erfahrung mit fischertechnik voraus¬gesetzt).

Für die Entwicklung des Steuerungsprogramms zur Lösung der Programmieraufgaben benötigen Schülerinnen und Schüler etwa 60-90 Minuten. Der Zeitaufwand für die Lösung der Experimentieraufgaben liegt – abhängig von Alter und Erfahrung – bei 135-240 Minuten.

Bezug Curriculum

Land	Stufe/Fächer	Bezüge
BW		
BY		
BE		
BB		
HB		
HH		
HE		
MV		
NI		
NW		
RP		
SL		
SN		
ST		
SH		
TH		

Anlagen
Aufgabe 5: Ballroboter

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.
- Styroporkugeln

Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Endlicher Automat (Zustandsautomat)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

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

Name: _____ Klasse: ___ Datum: _____

Lösungsblatt Aufgabe 5
Balllroboter

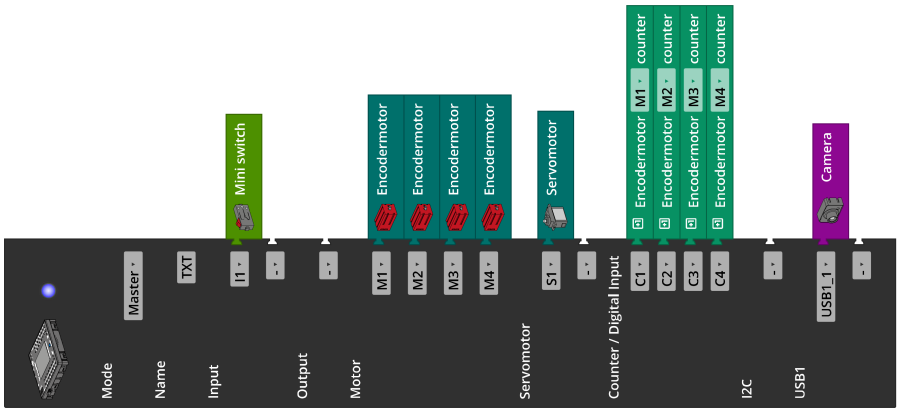
Die Aufgaben sind eine Übung zur Nutzung von Schleifen und Funktionen mit dem Ziel, ein übersichtliches, kompaktes und verständliches Programm zu erhalten. Die Ergebnisse der Schülerinnen und Schüler sollten diesbezüglich miteinander verglichen werden.

Konstruktionsaufgabe

Siehe Bauanleitung.

Programmieraufgaben

Konfiguration der Sensoren und Aktoren:



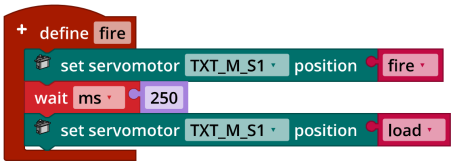
Für die Lösung von Programmieraufgabe 2 wird die „Voice Control“-App (für iOS oder Android) benötigt. Die App muss für die Spracherkennung mit dem Internet verbunden sein und (über Bluetooth oder WLAN) mit dem Controller verbunden werden.

1. Auswurfmechanismus

Kommentiert [KJ1]: evtl. anderen Begriff verwenden

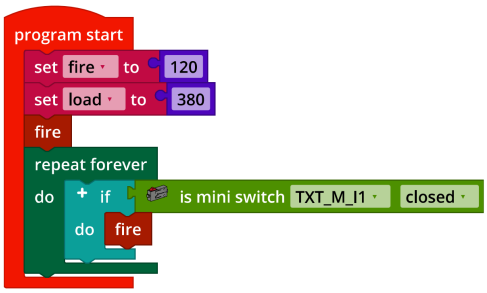
1a. Geeignete Werte für die Variablen „fire“ und „load“ (die gesuchten Positionen des Servo-Hebels) können leicht mit dem Interface-Test bestimmt werden. Sie sind modellabhängig und hängen insbesondere von der Position des aufgesteckten Servo-Hebels ab. Im hier verwendeten Modell liegen sie bei 120 (fire) und 380 (load).

1b. Funktion „fire“ (Beispiel):



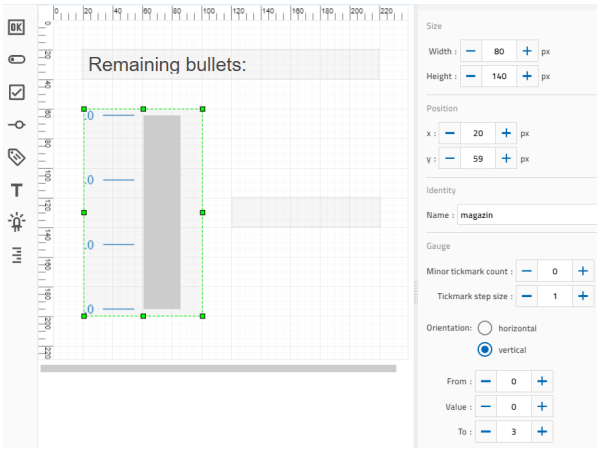
Mecanum_Fire_and_Load.ft

1c. Programmauszug (Beispiel):

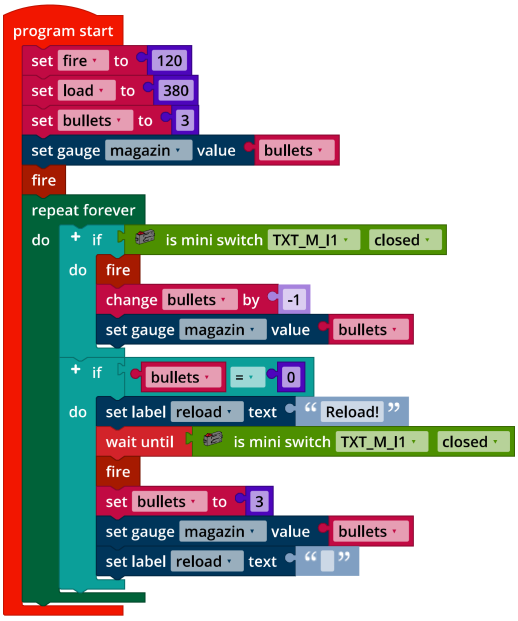


Mecanum_Fire_and_Load.ft

1d. Konfiguration des TXT-Displays (Beispiel):



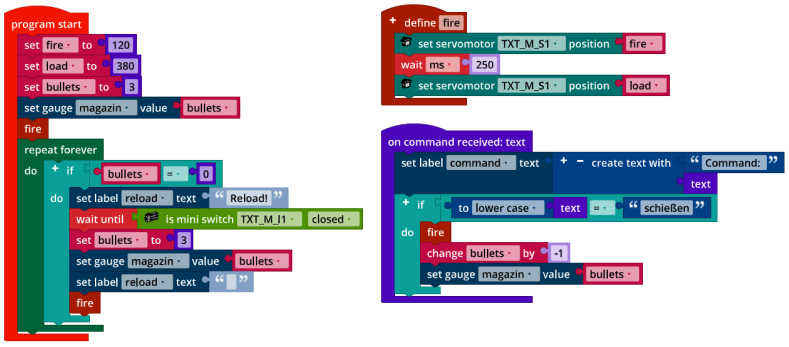
Programm (Beispiel):



Mecanum_Fire_and_Load_extended.ft

2. Sprachsteuerung

Programm (Beispiel):



Mecanum_Fire_and_Load_Voice_Command.ft

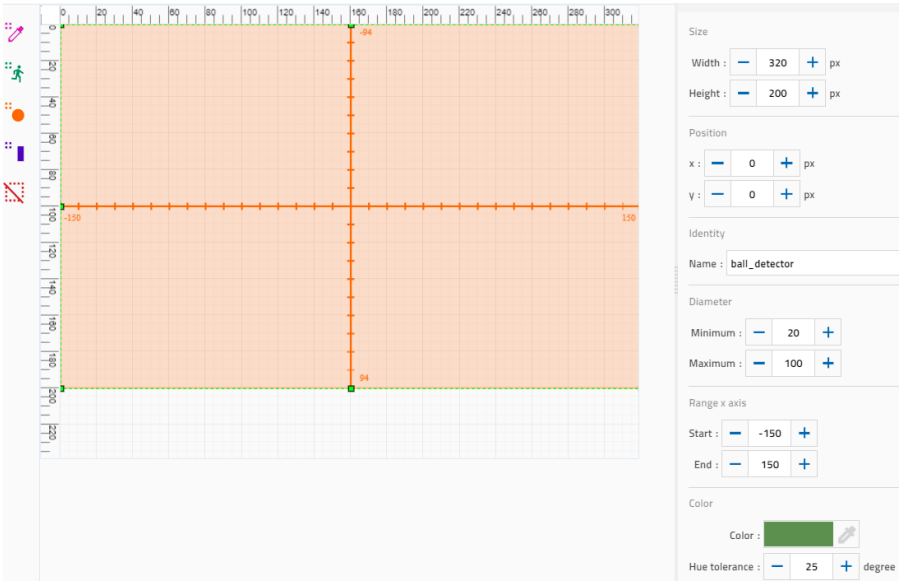
Experimentieraufgaben

1. Zielscheibe

1a. Die Zielscheibe sollte idealerweise so weit entfernt stehen, dass die Kunststoffhohlkugel die Zielscheibe etwas oberhalb der Mitte treffen.

Im verwendeten Beispielmodell liegt dieser ideale Abstand bei etwa 35 cm. Dieser Wert kann von Modell zu Modell leicht variieren.

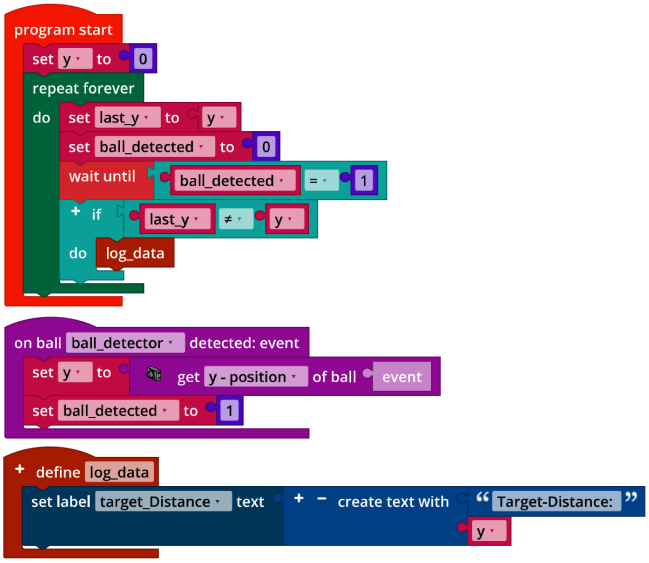
Konfiguration des Erkennungsfensters (Ball-Erkennung):



Je größer das Erkennungsfenster und der Bereich des zu erkennenden Ball-Durchmessers, desto größer die maximale Abweichung vom idealen Abstand, die erkannt werden kann. Als Range für die X-Achse ist das Intervall [-150, 150] zu empfehlen: Es nutzt fast die gesamte Auflösung (320 Pixel) und lässt sich leicht in einen Winkel umrechnen, da der Öffnungswinkel der Kamera 60° beträgt.

Die gewählte Farbe und die Hue-Toleranz müssen an die jeweiligen Lichtverhältnisse angepasst werden.

1b. Programm zur Abstandsmessung (Beispiel):

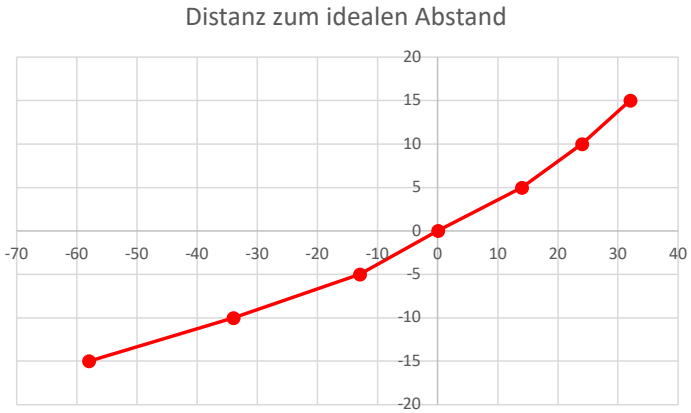


Test_Target_Distance.ft

Beispielmessung zum Verhältnis des Abstands der Zielscheibe zum y-Wert des Ballmittelpunkts, den die Ball-Erkennung liefert:

Abstand von idealer Distanz	y-Koordinate
+15 cm	32
+10 cm	24
+5 cm	14
0 cm	0
-5 cm	-13
-10 cm	-34
-15 cm	-58

Die von der Ball-Erkennung zurückgelieferten Werte der y-Koordinate des Ballmittelpunkts hängen von der gewählten Skalierung und der Höhe des Erkennungsfensters ab. Bis auf einen Skalierungsfaktor sollten die Werte jedoch mit den in obiger Tabelle angegebenen übereinstimmen.



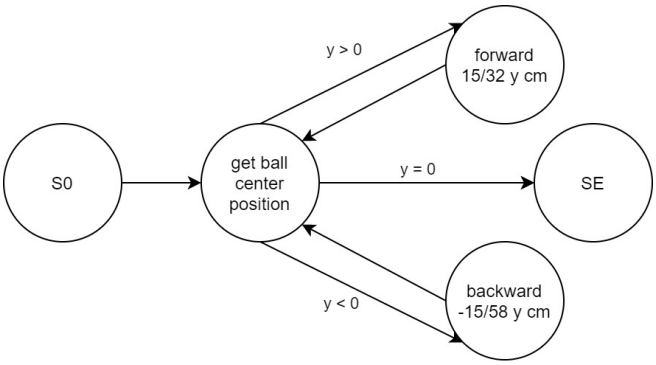
1c. Die Distanz d, um die das Fahrzeug vom idealen Abstand zur Zielscheibe entfernt ist, lässt sich leicht mit zwei linearen Funktionen annähern:

$$d = -\begin{cases} y > 0: \frac{15}{32}y \\ y = 0: 0 \\ y < 0: \frac{15}{58}y \end{cases} \text{ cm}$$

Auch hier hängt der Steigungsfaktor der Geraden von der Höhe des Erkennungsfensters und der in der Ball-Erkennung eingestellten Skalierung ab und kann um einen Faktor abweichen.

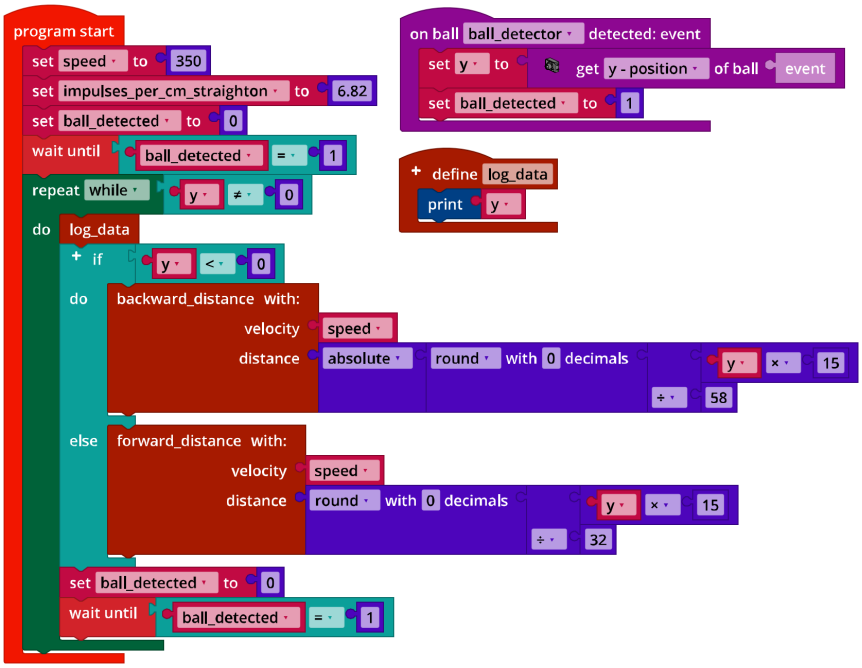
2. Zielabstandskorrektur

2a. Zustandsübergangsdiagramm:



State_Transition_Diagram_Correct_Position.drawio

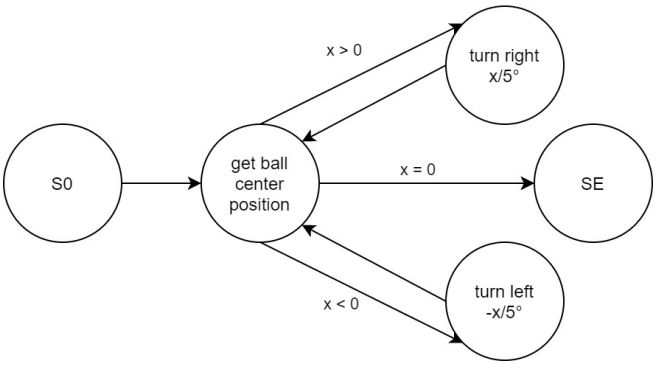
2b. Programmauszug (Beispiel):



Mecanum_Correct_Distance.ft

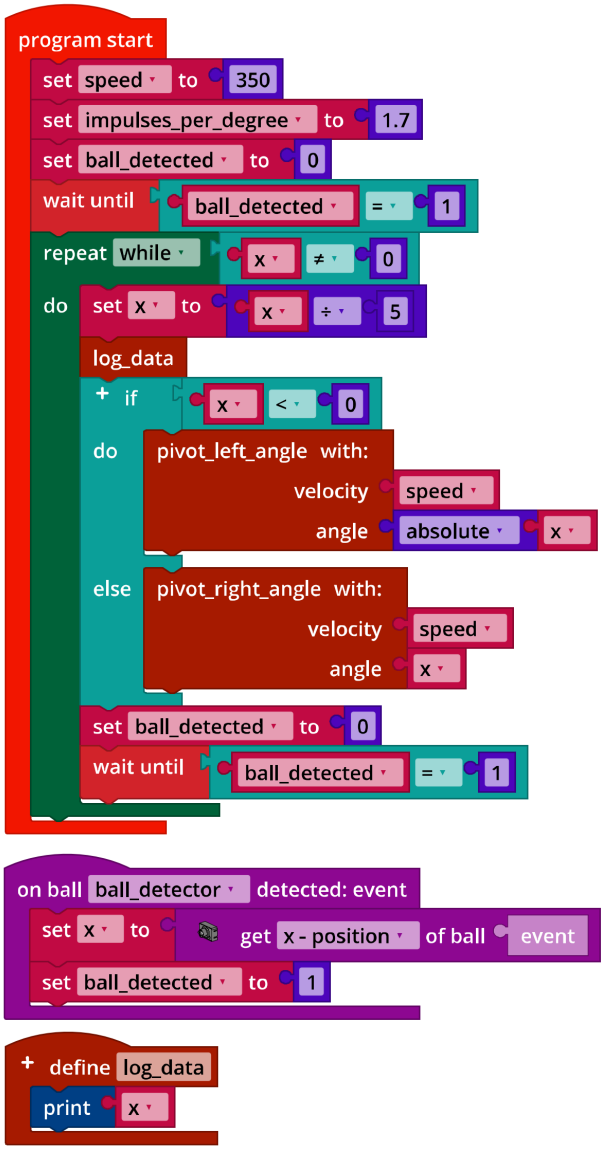
3. Zielausrichtung

3a. Zustandsübergangsdiagramm:



State_Transition_Diagram_Correct_Position.drawio

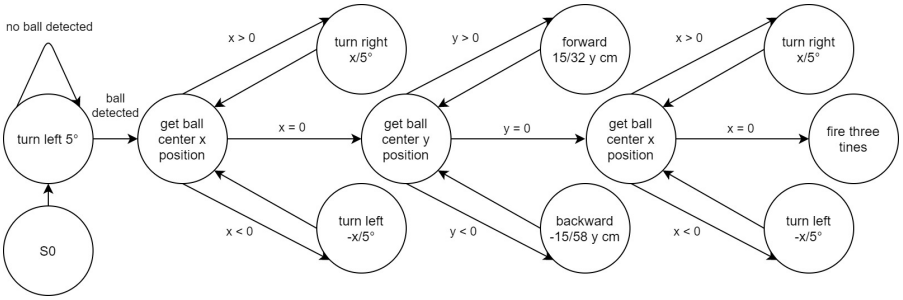
3b. Programmauszug (Beispiel):



Mecanum_Turn2Target.ft

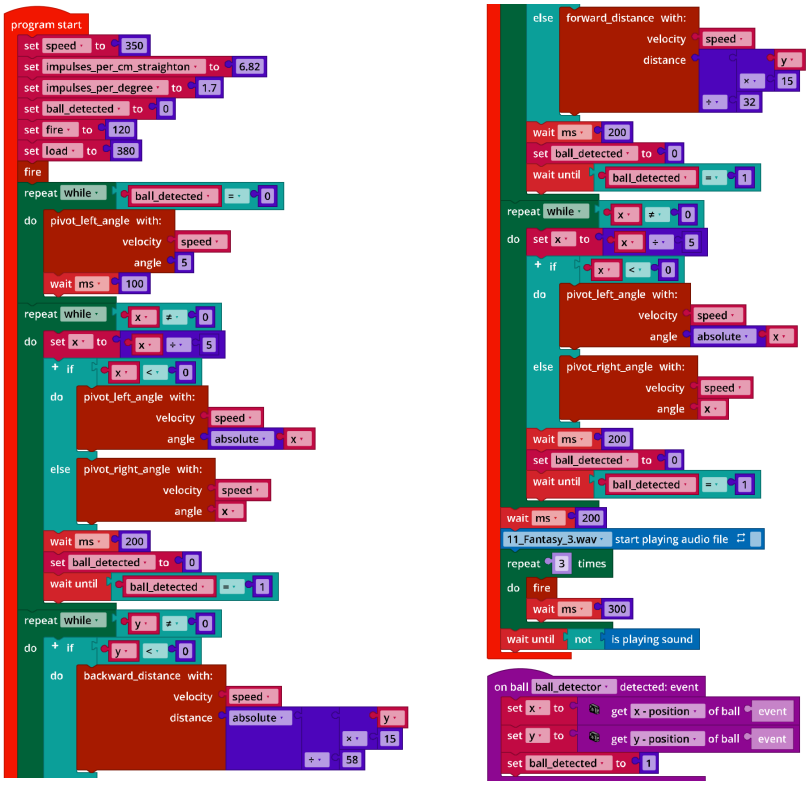
4. Zielsuche

4a. Zustandsübergangsdiagramm:



State-Transition_Diagram_Find_Target.drawio

4b. Programmauszug (Beispiel):



Mecanum_Find_Target.ft

Anlagen

Aufgabe 5: Ballroboter

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.
- Styroporkugeln

Weiterführende Informationen

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Endlicher Automat (Zustandsautomat)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

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