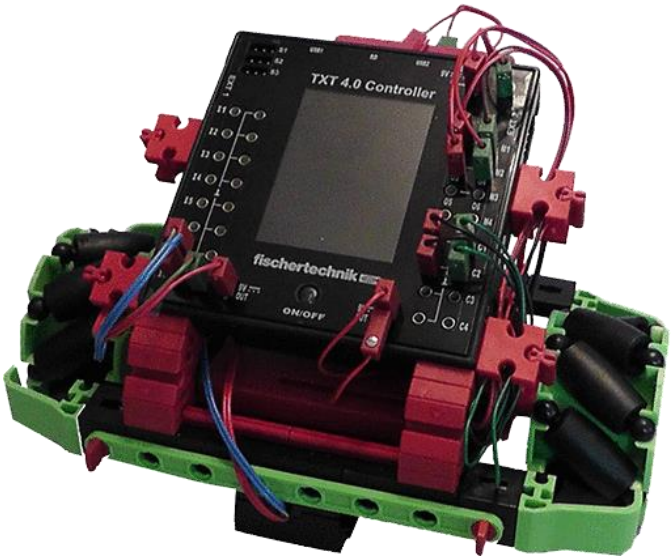
Omniwheels X2



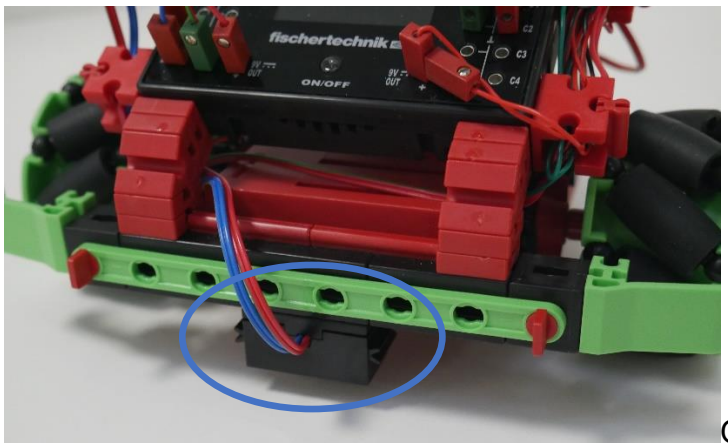
Mit diesem Modell erarbeitest du dir die Grundlagen der Programmierung eines zweirädrigen Fahrzeugs mit Stützrad und der im Modell verbauten Elemente.

Wenn du das Modell anhand der Bauanleitung aufgebaut hast, sind dir sicher neue Bauteile aufgefallen. Du verwendest den Infrarot-Sensor, 2 Encodermotoren als „Aktoren“ und einen neuen Rädertyp – das Omniwheel-Rad.

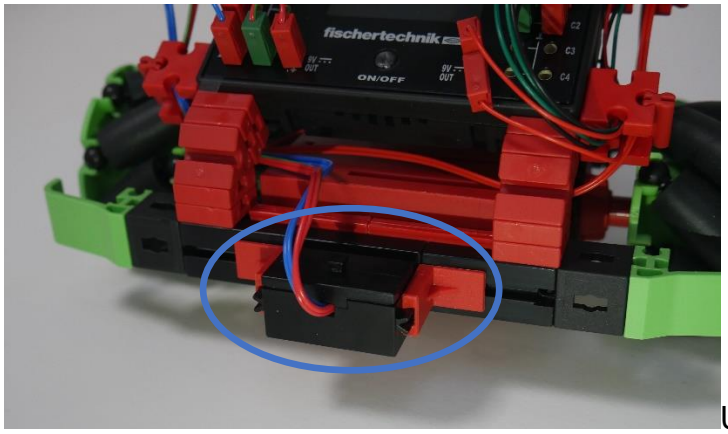
Infrarotsensor:

Der Infrarot-Spursensor ist ein digitaler Sensor zur Erkennung einer schwarzen Spur auf weißem Untergrund mit einem Abstand von 5 – 30 mm. Er besteht aus zwei Sende- und Empfängerelementen.

Achtung! Wichtiger Hinweis!
Ist der IR Spursensor wie in der Bauanleitung abgebildet eingebaut, kann es vorkommen, dass durch zu starke Unebenheiten der Abstand zwischen Sensor und Parcours an manchen Stellen zu gering ist und die Spur nicht mehr richtig erkannt wird. Dieses Problem kann behoben werden durch einen einfachen Umbau: Befestige den Sensor wie in den folgenden Abbildungen gezeigt mit zwei Bausteinen 38423, die im Baukasten enthalten sind. Durch den größeren Abstand zum Parcours wird die Spur auch bei größeren Unebenheiten sicher erkannt.

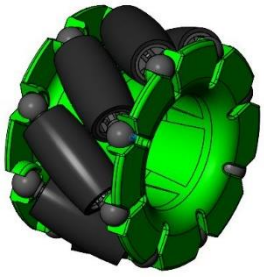


Originaleinbau nach Bauanleitung



Umbau mit größerem Abstand

Omniwheel-Rad:



Wichtig: Wenn du später die Programmierung durchführst, musst du auf die Laufrichtung der Räder achten. Die Drehrichtung nach links bedeutet, die Räder drehen vorwärts, nach rechts bedeutet, die Räder drehen rückwärts.

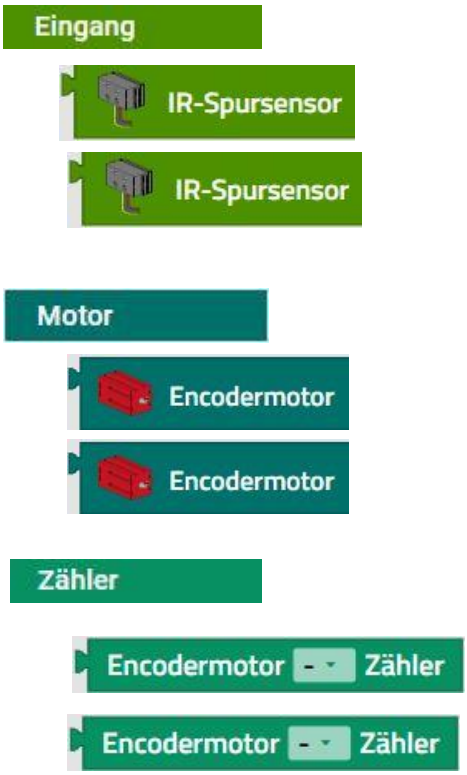
Das Modell „Omniwheels x2“ gliedert sich in 4 Programmieraufgaben:

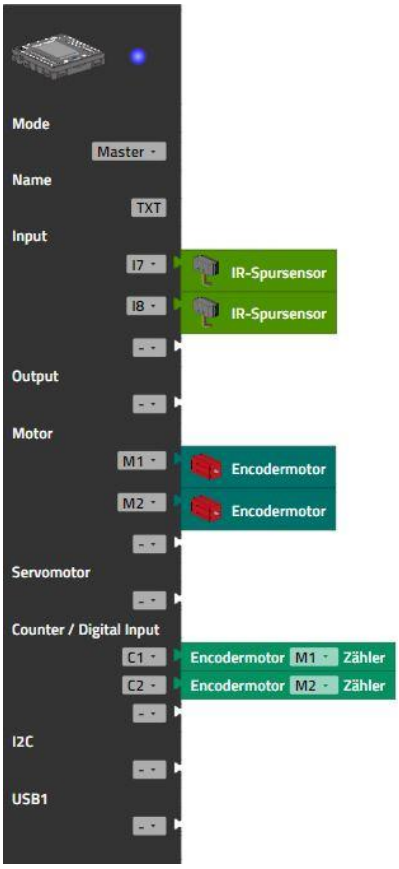
Aufgabe 1	Programmierlevel 2 Fahrbefehl – eine bestimmte Strecke vor- und zurückfahren (zeit- oder impulsgesteuert)
Aufgabe 2	Programmierlevel 2 Fahrbefehl – kurze Strecke geradeaus und dann eine kurze Strecke nach rechts oder links fahren
Aufgabe 3	Programmierlevel 2 Fahrbefehl – Abfahren eines Quadrats (zeit- oder impulsgesteuert)
Aufgabe 4	Programmlevel 2 Fahrbefehl – folgen einer schwarzen Linie

Controllerkonfiguration

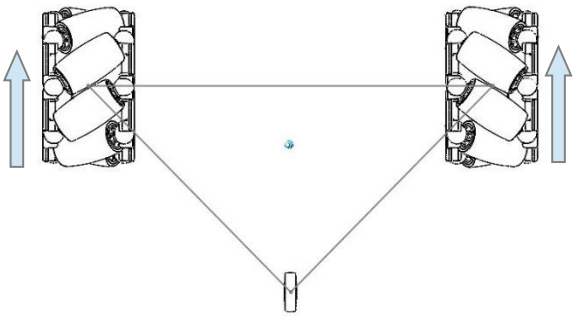
Starte zuerst das Programm „ROBO Pro Coding und führe die Controllerkonfiguration durch.
Die Verdrahtung der Sensoren und Aktoren entnimmst du wieder der Bauanleitung.

Du benötigst folgende Aktoren und Sensoren:





Aufgabe 1

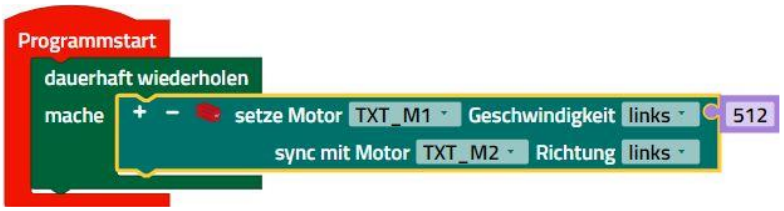


In der ersten Aufgabe soll das Fahrzeug eine bestimmte Zeit geradeaus nach vorne und wieder zurückfahren. Anschließend soll das Programm stoppen. Dazu müssen beide Räder gleichzeitig angesteuert werden. Du befindest übrigens dich in der Lernstufe 2.

Das Programm kannst du in die vorgegebene Endlosschleife einbauen.

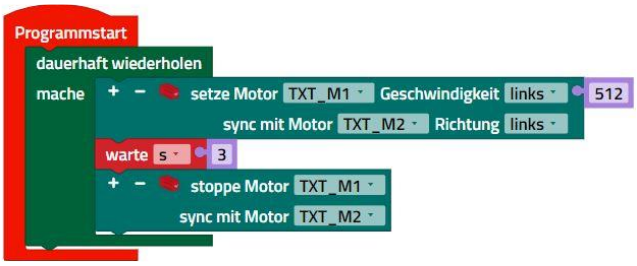
Baue diesen Befehl in die Andockstelle ein.

Klicke mit der Maus das +-Zeichen im Befehl an. Dadurch erweiterst du den Befehl so, dass der Motor am Ausgang TXT_M2 mit dem Motor am Ausgang TXT_M1 synchron und in die gleiche Richtung mit einer Geschwindigkeit von 512 läuft (das ist die volle Geschwindigkeit).

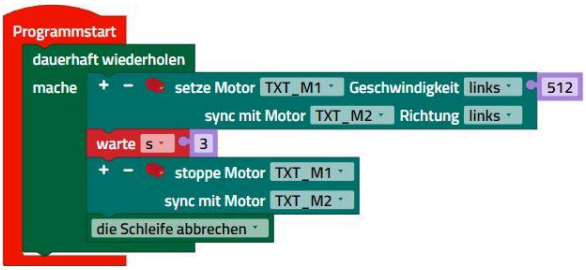


Füge eine Wartezeit von 3 Sekunden ein und dann den Befehl zum Stoppen der beiden Motoren.

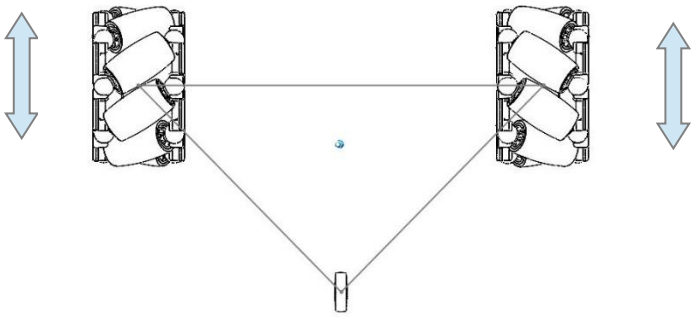
Auch hier klickst du auf das + – Zeichen und der Befehl wird mit TXT_M2 erweitert. Somit wird auch der zweite Motor gestoppt.

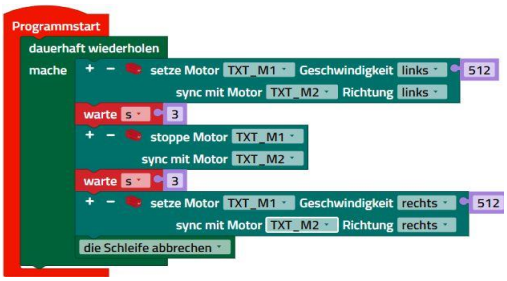


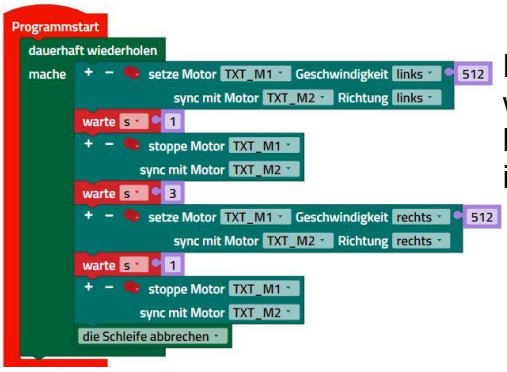
Damit das Programm nicht endlos läuft, baust du noch den Befehl „die Schleife abbrechen“ aus dem Block „Schleifen“ ein. Anschließend kannst du das Programm zum Modell senden und es ausprobieren. Das Modell sollte 3 Sekunden nach vorne fahren.



Damit das Modell nach einer Wartezeit von 3 Sekunden in die andere Richtung fährt, kopierst du einfach den Fahrbefehl und fügst ihn mit einer zusätzlichen Wartezeit von 3 Sekunden nach dem Stoppbefehl für die Motoren ein. Hier musst du nur die Richtung der beiden Motoren ändern.







Das Modell soll nach einer Laufzeit von 3 Sekunden wieder stehen bleiben. Füge den Warte-Befehl ein und kopiere den Stopp-Befehl an die entsprechende Stelle im Programm.

So das wäre geschafft und du kannst das Programm jetzt auf deinen Rechner speichern.

Benenne es einfach mit

„Omniwheels_x2_forward_backward_time“

Im nächsten Beispiel erarbeitest du dir die Grundlagen zum Fahren einer bestimmten Strecke mit der Impulssteuerung. Dazu sind einige Überlegungen nötig.

Wenn du die Informationen zum Encodermotor gelesen hast, wirst du festgestellt haben, dass der Encodermotor ein Impulsrad besitzt.

Bei einer Umdrehung der Motorwelle werden 63,9 Impulse erzeugt. Wir nennen diese Anzahl einfach mal I_{welle} = 63,9. Betrachte als Nächstes einmal das Modell. Auf der Welle sitzt ein Zahnrad mit 10 Zähnen. Dieses treibt ein Zahnrad mit 20 Zähnen auf der Radachse an.

Das bedeutet, du hast eine Übersetzung vom Schnellen zum Langsamen in einem Verhältnis von 10 Zähnen zu 20 Zähnen. Somit gilt, i für Übersetzungsverhältnis ist das Verhältnis der Zähnezahl Z der Zahnräder. Die Nummerierung erfolgt mit der Richtung des „Kraftflusses“, d. h., das treibende Rad (Motorwelle) ist Z_1 und das getriebene Rad (Radwelle) ist Z_2.

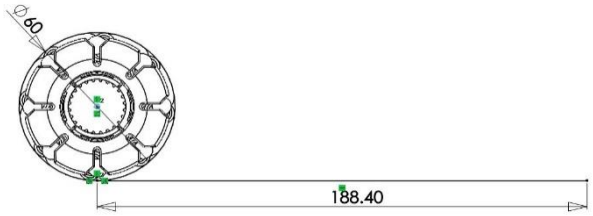
Somit lautet unsere Formel für den Antrieb:

$$i = \frac{Z1}{Z2} \quad \text{oder} \quad i = \frac{10}{20} \quad i = 0{,}5$$

Jetzt musst du diesen Wert auf die Anzahl der Impulse übertragen. Wie vorher schon erwähnt, beträgt die Anzahl der Impulse an der Motorwelle I_{welle} = 63,9.

I_{achse} ist somit I_{welle} x 0,5. Somit werden bei einer Umdrehung 63,9 x 2 = 127,8 Impulse durch den Impulszähler am TXT 4.0 Controller gezählt. Dieser Wert ist bei allen Modellen eine feste Größe, da die Zahnräder nicht verändert werden.

Eine weitere feste Größe ist der Durchmesser des Omniwheelrades. Dieser beträgt 60 mm.



Die Darstellung zeigt die Strecke in mm wenn das Rad eine Umdrehung macht. Wie ermittelt man aber die Strecke? Dazu dient dir die mathematische Formel zur Berechnung der Länge des Umkreises des Rades.

Die Formel lautet: U=d x 3,14 (Pi). Somit ist der Weg, den das Rad bei einer Umdrehung zurücklegt, 60 mm x 3,14 = 188,40 mm. Für die weitere Berechnung 188 mm.

Das Modell soll geradeaus fahren und dann stoppen. Du musst also die Impulse (I_{fahrt}) berechnen, die benötigt werden, um diese Strecke zu fahren. S_{umdr} steht für Strecke einer Umdrehung also 188 mm, I_{umdr} für Impulse pro Umdrehung hier 127,80.

Ermittle wie viel Impulse du für 1 Millimeter Fahrstrecke benötigst.

$$I_{imp/mm} = I_{umdr}/S_{umdr} = 128/188 = 0{,}68 \text{ imp/mm (rein mathematisch)}$$

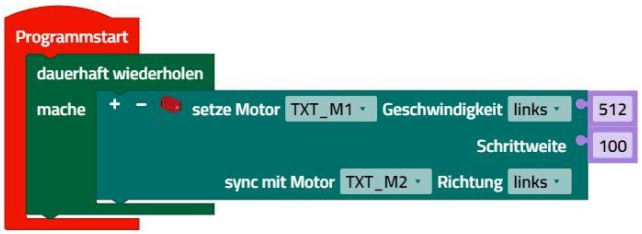
Beachtest du noch Toleranzen und Abmessungen der Räder, dann musst du den Wert auf **0.72** festlegen. Diesen Wert kannst du bei allen Wegstreckenberechnung verwenden.

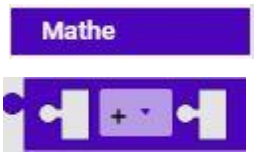
Den Wert multiplizierst du mit der zurückzulegenden Strecke.

$$0{,}72 \text{ imp/mm} \times 200 \text{ mm} = 144 \text{ imp}$$

Lösche aus dem Programm alle Befehle und füge zuerst einen neuen Befehl „setzt Motor – Geschwindigkeit – Schrittweite“ aus dem Block „Motor ein.

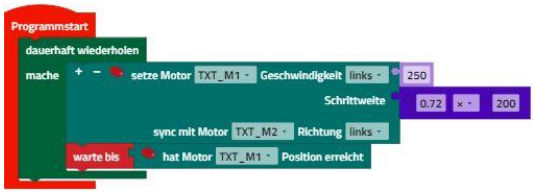
Da beide Motoren gleich drehen sollen, kannst du „+“ aktivieren und somit den Befehl erweitern.



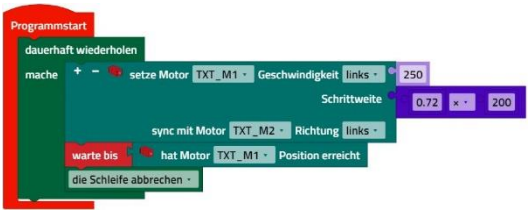


An die Stelle des Wertes der Schrittweite fügst du einen mathematischen Ausdruck ein. Wähle aus dem Block „Mathe“ den Befehl „ist die Summe zweier Zahlen“ aus. In die beiden Leerstellen fügst du aus dem gleichen Block den Befehl „Eine Zahl“ ein.

Gebe anstelle der „0“ den Wert I= 0.72 ein. Ändere anschließend das +-Zeichen auf x um und ändere die 2. Eingabe in die Strecke, die gefahren werden soll (200 für 20 cm). **Verwende einen Punkt als Dezimalzeichen!**



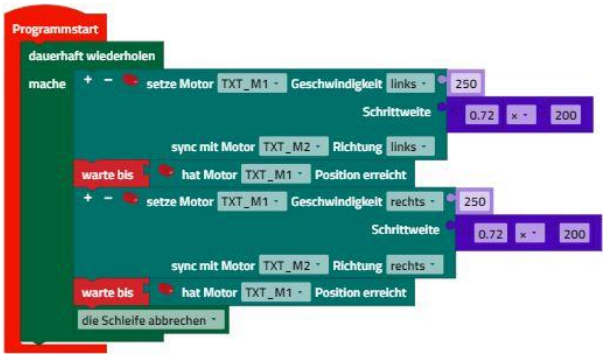
Füge nach dem Block noch einen Befehl aus dem Block „Util“ „warte bis“ und aus dem Block „Motor“ den Befehl „hat Motor --- Position erreicht“ ein. Danach soll die Schleife abgebrochen werden.



Teste das Teilprogramm.

Sicher wirst du feststellen, dass das Fahrzeug je nach Beschaffenheit der Fahrbahn nicht geradeaus fährt. Das liegt zum einen an einem Rutschfaktor der Fahrbahn und weil beim Start das komplette Drehmoment auf die Räder gelangt.

Gehe hier einfach mit der Geschwindigkeit auf 200 herunter.



Erweitere das Programm für eine impulsgesteuerte Rückfahrt. Kopiere dazu die entsprechenden Befehle in das Programm. Ändere hier die Laufrichtung der Räder von „links“ nach „rechts.

Speichere das Programm noch ab und teste die Funktion des Programms. Funktioniert alles, fährt das Fahrzeug 200 mm nach vorne und dann wieder 200 mm zurück und bleibt dann stehen.

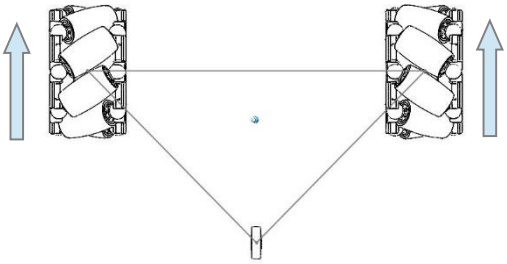
OmniwheelsX2_forward_backward_dist

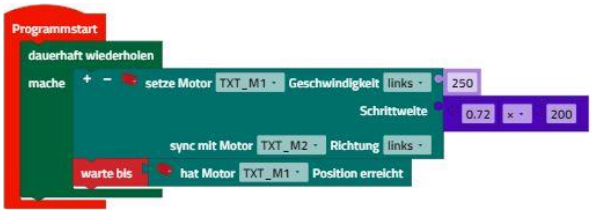
Aufgabe 2

Für die nächste Aufgabe kannst du die Controllereinstellung übernehmen. Starte ein neues Projekt und verwende einen neuen Projektnamen, z.B.:

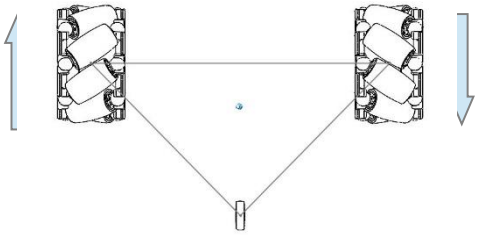
Omniwheels_x2_forward_rot90_dist

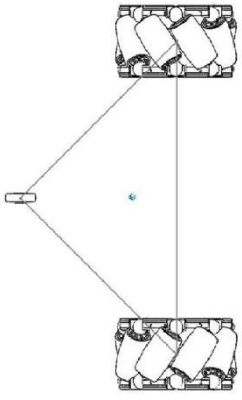
Zuerst soll das Fahrzeug eine bestimmte Strecke (200mm) nach vorne fahren. Diese Befehlsfolge kennst du aus der ersten Aufgabe.





Das Fahrzeug stoppt und soll sich dann um 90° nach rechts drehen.





Dazu muss der linke Motor die Laufrichtung beibehalten, der rechte Motor die Laufrichtung ändern.

Der Drehpunkt der Räder ist der Mittelpunkt des Stützrades.

Auch hier musst du die Räder über eine bestimmte Impulseingabe steuern.

Für eine Umdrehung des Modells werden **640 Impulse**

benötigt.

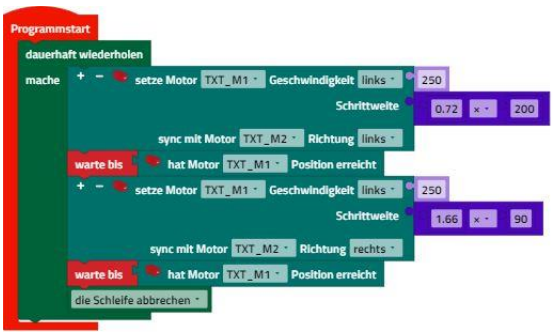
Im nächsten Rechenschritt setzt du die Impulse in Relation zu einem Winkel.

Dazu ermittelst du zuerst, wie viele Impulse man benötigt, wenn das Fahrzeug 1° drehen soll.

Die Formel lautet: $I_{360°}/360° = 640/360 = 1.77$. In Bezug auf Toleranzen und Abmessungen der Omniwheel-Räder verwendest du den **Wert 1.66**.

Mit dieser Zahl, multipliziert mit dem benötigten Winkel, erhältst du den Wert für die Impulse.

Beispiel: Das Fahrzeug soll sich um 90° drehen – das entspricht 1,66 x 90 = 149,4 oder 149 Impulsen.



Baue in dein Programm die Befehlsfolge zur Drehbewegung 90° nach rechts ein. Setze die Geschwindigkeit des Motors auf 250.

Du kannst die gleiche Formel aus dem Block „Mathe“ verwenden. Der Fixwert ist aber „1.66“.
Du kannst das Teilprogramm schon einmal testen. Hier ging es darum, dass du eine Kurve nach rechts oder links programmierst.

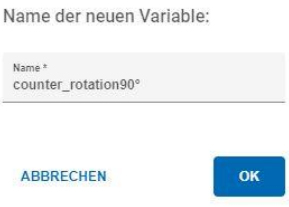
Aufgabe 3

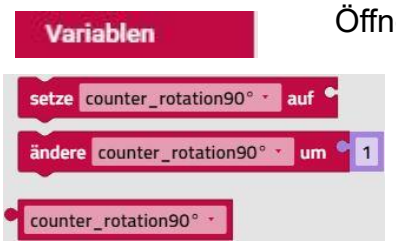
Omniwheels_x2_square

In dieser Aufgabenstellung soll das Fahrzeug ein Rechteck abfahren. Lösche nicht die Befehle aus dem Programm, sondern ziehe sie nach rechts aus dem Programm. Diese werden später wieder benötigt.

Für die Aufgabe musst du eine sogenannte „Variable“ erstellen die du „counter_rotation90°“ nennst. Dazu öffnest du aus dem Block „Variablen“ den Befehl „Variable erstellen“.

Es erscheint ein Eingabefenster, in das du den entsprechenden Namen einträgst. Die Eingabe beendest du mit „OK“.





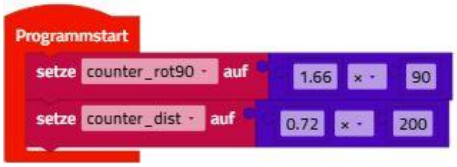
Öffne anschließend den Block „Variablen“. ROBO Pro Coding stellt dir jetzt 3 Befehle zur Verfügung

Ziehe zuerst den Befehl „setze counter_rotation90° in den Programmstart. Aus dem Block „Mathe“ holst du dir die schon bekannten Befehle „ist die Summe ...“ und zweimal den Befehl eine Zahl“ und fügst sie an die Andockstelle ein.

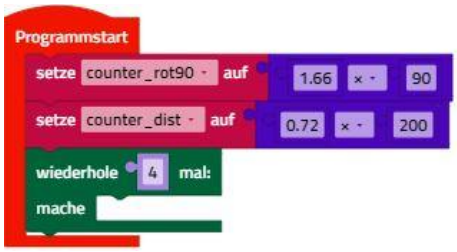
Ändere das „+-Zeichen“ in das „x-Zeichen“ um. Ersetze die erste 0 mit dem Wert „1.66“ und die zweite 0 mit dem Winkelwert „90°“.

Wichtig ist, dass du als Dezimalzeichen einen Punkt eingibst.

Erzeuge noch eine zweite Variable für die Strecke, die das Fahrzeug geradeaus fährt. Benenne sie „counter_dist“.

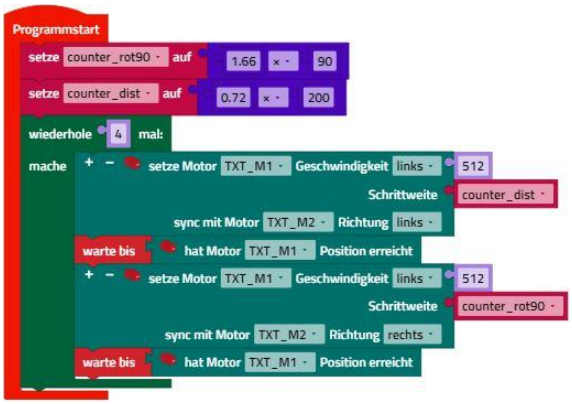


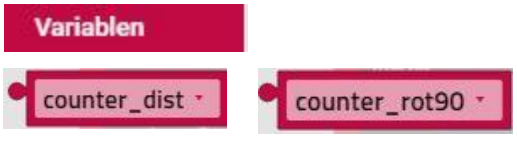
Jetzt folgt ein neuer Befehl aus der Gruppe „Schleifen“. Da ein Quadrat aus 4 Seiten besteht, kannst du einen Schleifenbefehl verwenden, der viermal durchlaufen wird. Diesen fügst du in dein Programm ein. Ändere die Wiederholung auf „4“ um.



Füge anschließend die Befehlsfolge für die Motorsteuerung ein. Die Befehle ziehst du einfach wieder zurück in dein Programm.

Im letzten Programmierschritt fügst du die beiden Variablen „counter_dist“ und „counter_rotation90°“ aus dem Block „Variablen“ ein. Diese ersetzen die beiden Schrittweitenwerte.





Somit ist auch diese Aufgabe gelöst. Speichere das Programm noch einmal ab.

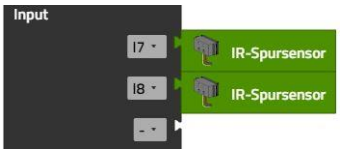
Es wird immer interessanter und du entwickelst dich langsam zu einem Programmierprofi. Nachdem du dein Fahrzeug schon bestimmte Stecken und Kurven fahren lassen kannst, kommt jetzt eine Aufgabe bei der du den Infrarot-Sensor einsetzt. Dieser ermöglicht, ins Programm eingebunden, einer schwarzen Linie zu folgen.

Aufgabe 4

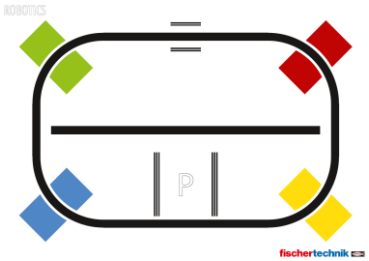
Beginne ein neues Programm und nenne es

Omniwheels_x2_linefollower

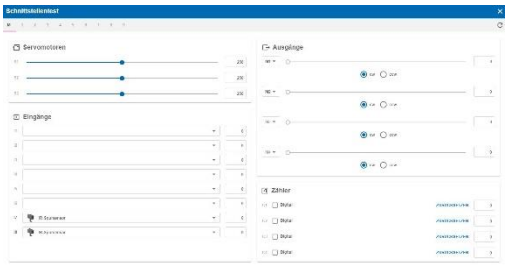
In der Controllerkonfiguration hast du den IR-Sensor schon eingesetzt.



Achte darauf, dass du die Anschlüsse „I7“ und „I8“ verwendest.



Für die spätere Testfahrten verwendest du den im Baukasten mitgelieferten Parcours.
Zuerst eine wichtige Information zum IR-Sensor.
Wie ist das Schaltverhalten? Das kannst du ganz einfach über den Schnittstellentest ermitteln.



Starte den Test mit dem Befehl 

Es öffnet sich das Testfenster.
Setze dein Modell mit dem IR-Sensor so auf die schwarze Linie, sodass beide Sensoren auf der Linie liegen.

Im Schnittstellentest werden die Eingangwerte mit „0“ angezeigt.

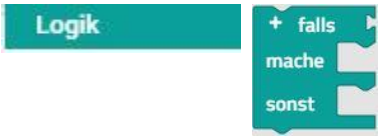
Verschiebe jetzt dein Modell so, dass ein IR-Sensor außerhalb der Linie liegt. Verschiebe nach links. Der Wert von „I8“ schaltet auf „1“ um.

Versuche das auch mit dem zweiten IR-Sensor. Verschiebe dazu das Modell nach rechts. Der Wert von „I7“ schaltet auf „1“ um.

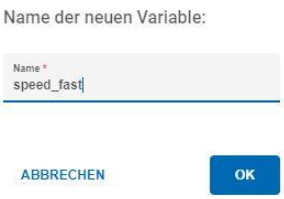
Liegen beide IR-Sensoren außerhalb der Linie sind beide Eingangswerte „1“.

Füge zuerst in den Programmstart eine Schleife „dauerhaft wiederholen“ ein.

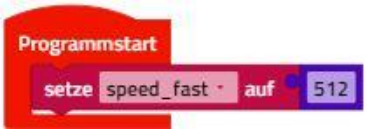
Füge aus dem Block „Logik“ den Befehl „falls – mache – sonst“ ein.



Die Geschwindigkeit legst du in einer Variablen mit dem Namen „speed_fast“ ab. Erzeuge die Variable.



Füge aus der Gruppe „Variablen“ den Befehl „setze speed_fast auf“ ein. Füge aus dem Block „Mathe“ noch eine „Zahl“ ein und ändere den Inhalt auf „512“ um.



Erzeuge eine zweite Variable mit dem Namen „speed_slow“ und füge sie nach dem letzten Befehl ein. Ändere den Wert auf „350“ um.

Erzeuge eine dritte Variable mit dem Namen „state“ und füge sie nach dem letzten Befehl ein. Ändere den Wert auf „0“ um.

Die Setze-Blöcke für Variablen fügst du vor der Endlosschleife ein. Damit werden alle Variablen auf einen bestimmten Wert initialisiert.

Wenn du auf den Pfeil neben dem Variablennamen klickst, öffnet sich ein Auswahlfenster. In diesem werden dir die aktuellen Variablen angezeigt. Gleichzeitig kannst du hier eine Variable auswählen.

Die Variable „state“ soll so gesetzt werden, dass diese alle Zustände der IR-Spursensoren bestehend aus I7 und I8 abbildet. Es gibt 4 Kombinationen für die beiden Sensoreingänge.

state	I7==0	I8==0
0	falsch	falsch
1	wahr	falsch
2	falsch	wahr
3	wahr	wahr

Wenn beide Werte von I7 und I8 den Wert 0 (schwarze Fläche) haben, so ist der Wert von state=3. Wenn beide Sensoren I7 und I8 den Wert 1 (weiße Fläche) zurückgeben, so hat die Variable state=0. Wenn nur der linke oder der rechte Sensoreingang den Wert 0 (Kante schwarze Linie) erkennt, so soll state=1 oder state=2 sein.

Die verschiedenen Zustände von I7 und I8 können am einfachsten durch eine Summe berechnet werden. Bei I7 wird eine 1 und bei I8 wird eine 2 hinzuaddiert. Die Summe ist je nach Sensorwert dann zwischen 0 und 3.

Füge aus der Gruppe „Variablen“ den Befehl „setze state auf“ ein.

Füge aus der Gruppe „Mathe“ den Addiere-Block hinzu. Wenn Du mit Rechtsklick auf den Block gehst und „externe Eingänge“ auswählst, so werden die beiden Platzhalter übereinander dargestellt.

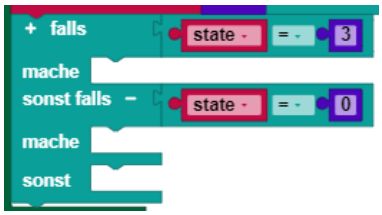
Füge in die beiden Platzhalter Bedingungen zur Abfrage der beiden IR-Sensoren ein. Dazu benötigst du zuerst aus der Gruppe „Logik“ den Befehl „prüfe … falls wahr … falls falsch. Bei dem Platzhalter „wahr“ und „falsch“ wird die entsprechende Zahl hinzugefügt, die zurückgegeben werden soll.

In die beiden Leerstellen hinter „prüfe …“ fügst du aus der Gruppe „Eingang“ den Befehl „ist IR-Spursensor ...“ ein.

Füge 4 Zahlen bei „wahr“ und „falsch“ wie folgt hinzu.

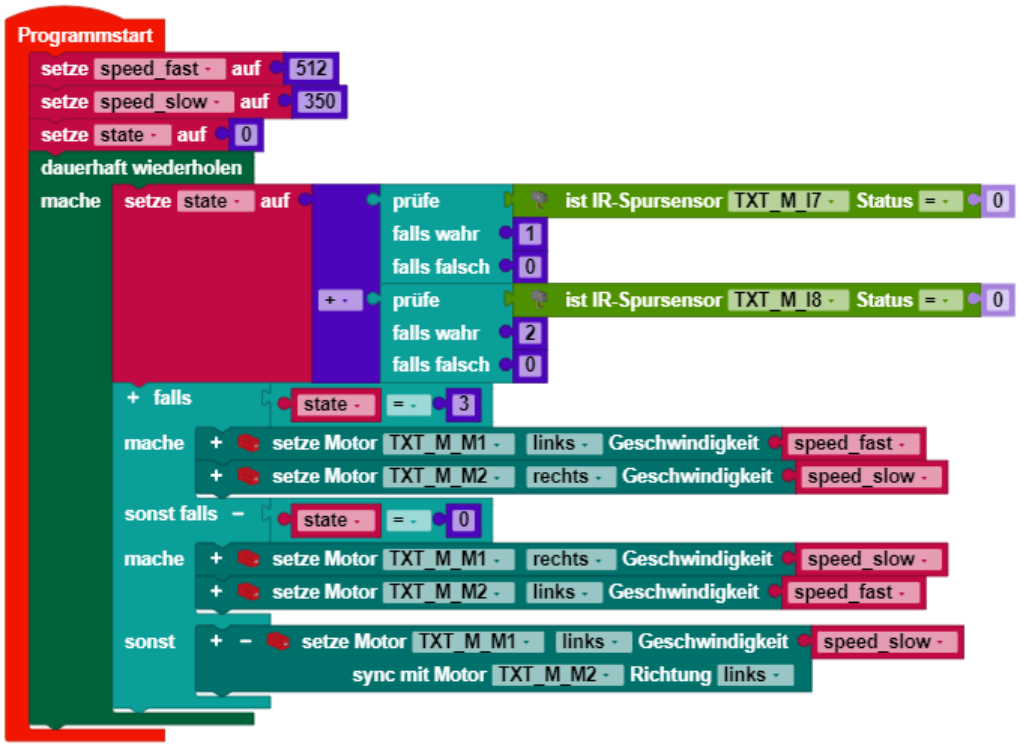
So das wäre schon mal geschafft. Der Zustand state wird in Abhängigkeit der beiden IR-Sensoren berechnet. Jetzt müssen die Motoren M1 und M2 entsprechend gesetzt werden.

Am besten funktioniert der Linienfolger, wenn dieser der Kante der Linie entlang folgt. D.h. wenn beide IR-Sensoren schwarz oder weiß sind, so dreht der Roboter in die entsprechende Richtung. Wenn beide IR-Sensoren unterschiedliche Werte haben, so soll der Roboter geradeaus auf der Linienkante fahren.
Erstelle zuerst die verschiedenen Fallunterscheidungen für state=3 und state=0 und sonst.



In den Bereich „mache“ ziehst du aus der Gruppe „Motor“ den Befehl

jeweils entsprechend in das Programm. Ändere die Geschwindigkeit in die Variablen „speed_fast“ und „speed_slow“ um. Ändere auch die Drehrichtung und die Bezeichnung der Motoren.



Speichere dein Programm und teste es.
Somit sind alle Aufgaben programmiert und du kannst dich an das nächste Modell wagen.