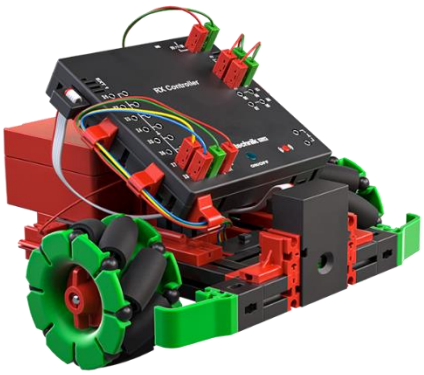
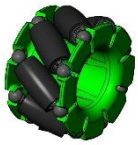
Advanced 2-Radroboter



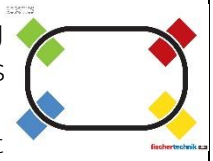
Mit diesem Modell erarbeitest du dir weitere Grundlagen der Programmierung. Die verbauten fischertechnik Bauteile kennst du ja schon von den vorherigen Aufgaben. Neu hinzugekommen ist der IR-Spursensor.

Im Modell verbaute Aktoren, Sensoren und technisches Zubehör:

Mini Motor	Getriebe	Gestensensor	Omniwheelrad	IR-Spursensor
				

Die Erklärung zu den Bauteilen findest du auf der Startseite.

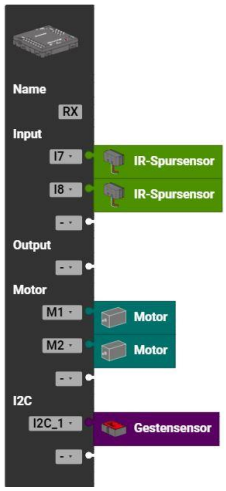
Das Modell „Advanced 2-Radroboter“ gliedert sich in 4 Programmieraufgaben:

Aufgabe 1 Advanced_2-wheeled_robot_1.ft	Programmierlevel 3 Der Fahrroboter fährt geradeaus. Trifft er auf ein Hindernis, soll er diesem ausweichen.
Aufgabe 2 Advanced_2-wheeled_robot_2.ft	Programmierlevel 3 Der Fahrroboter soll entlang einer schwarzen Linie, des vorgegeben Parcours fahren. Verlässt er die schwarze Linie, so bleibt dieser stehen. 
Aufgabe 3 Advanced_2-wheeled_robot_3.ft	Programmierlevel 3 Das Programm soll die Fahrt auf der schwarzen Linie so regeln, dass das Modell die schwarze Linie wieder findet, falls der Roboter die schwarze Linie verlässt.

Aufgabe 4 Advanced_2-wheeled_robot_4.ft	Programmierlevel 3 Der Fahrroboter fährt eine Stecke entlang, erkennt er ein Hindernis, reduziert er die Geschwindigkeit. Über einen Zufallsgenerator wird die Drehrichtung festgelegt, mit der dem Hindernis ausgewichen wird.

Du hast die Vorarbeiten - RoboPro Coding starten, neues Projekt starten, evtl. die Optionen für „neu“ eingestellt und „Leer“ für ein neues Programm durchgeführt.

Wie schon bekannt, musst du als Nächstes den Controller konfigurieren. Schalte dazu im Projektfenster „Controllerkonfiguration“ ein. Lege den RX-Controller als Standartcontroller fest. Füge anschließend die entsprechenden „Aktoren“ und „Sensoren“ ein. Diese kannst du aus der Bauanleitung entnehmen.



Neu hinzugekommen ist der IR-Spursucher. Diesen findest du unter dem Block „Eingang“ – „IR-Spursucher“.

Wichtig: Die Sensoren sind an den Eingängen „I7“ und „I8“ angeschlossen.

Nun zur ersten Aufgabe.

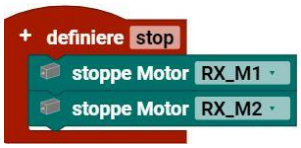
Aufgabe 1

Füge zwischen „Programmstart“ und „dauerhaft wiederholen mache“ den Befehl zum Einbinden des „Gestensensors“ in dein Programm ein.

Die Schaltfläche muss auf „aktiviere“ und „Entfernung“ da die Entfernung zu einem Hindernis gemessen werden soll.

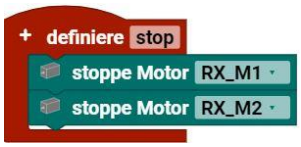
Um dein Programm übersichtlich zu gestalten, verwendest du für immer wiederkehrende Befehlsfolgen „Funktionen“.

Beispiel: Im Programm müssen mehrmals die beiden Motoren gestoppt werden. Diese werden nicht jedes Mal programmiert, sondern in eine „Funktion“ geschrieben. Diese kann an jeder benötigten Stelle im Programm aufgerufen werden.



Wie gehst du vor? Den benötigten Befehl findest du im Block „Verarbeitung“ – „Funktionen“. Ziehe den Befehl in den leeren Arbeitsbildschirm. Klicke auf „etwas tun“ und ändere es im Kontextfenster auf „stop“ ab.

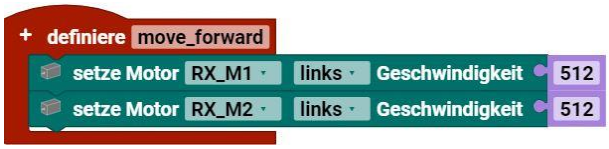
In die Freistelle fügst du die beiden Befehle zum Stoppen der Motoren ein.

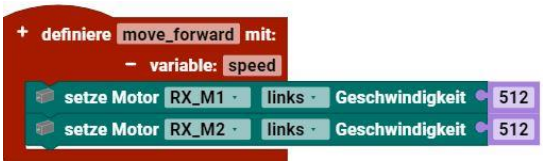


Somit ist diese Funktion fertig definiert.

Und schon folgt die nächste Funktion „move_forward“. Ziehe den neuen Funktionsblock in den Arbeitsbereich. Ändere den Namen auf „move_forward“ ab. Wird diese Funktion später im Hauptprogramm aufgerufen, fährt das Modell vorwärts.

Füge hier zwei Motorbefehle ein. Die Drehrichtung ist „links“, die Geschwindigkeit erstmals „512“.





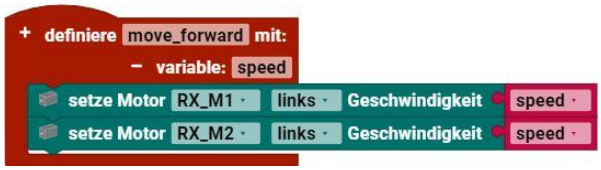
Die Geschwindigkeit soll über eine Variable „speed“ festgelegt werden. Dazu klickst du auf das „+“ vor dem „definiere“. Klicke auf das „x“ und ändere im Kontextfenster auf „speed“ ab.

Erstelle als Nächste die Variable „speed“. Dazu wählst du im Block „Verarbeitung“ – Variablen“ – „Variable erstellen ...“.

Im erscheinenden Kontextfester gibst du „speed“ ein. Die Variable wird im „Variablenfenster“ als Befehlsblock angezeigt.

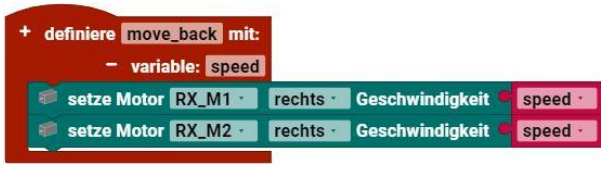
Im Hauptprogramm wird später der Wert „256“ für die Motorgeschwindigkeit festgelegt.

Wichtig: Wird in Funktionen die Variable „speed“ verwendet, wird der später im Hauptprogramm festgelegte Wert von „256“ übernommen.



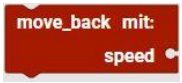
Ändere somit die beiden Geschwindigkeitswerte mit der Variablen „speed“ ab. Dazu ziehst du den Befehl „speed“ aus dem Block „Verarbeitung“ – „Variablen“ über den Zahlenwert der Motore.

Die Funktion „move_back“ erzeugst du, indem du die Funktion „move_forward“ duplizierst, und die Drehrichtung nach „rechts“ abänderst.

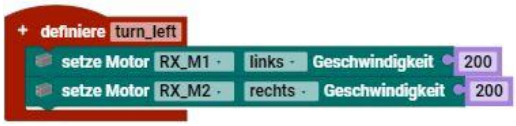


Somit sind die beiden Funktionen fertig definiert und du kannst zwei Funktionen für die Drehung nach „links“ – „turn_left“ und nach „rechts“ – „turn_right“ programmieren.

Aber wo wird ein Wert für die Variable „Speed“ festgelegt? Öffne dazu den Block „Verarbeitung“ und dort „Funktionen“. Im Befehlsfenster findest du den Befehl „move back mit: speed“ und einer freien Andockstelle.



Du benötigst noch zwei weitere Funktionen „turn_right“ und „turn-left“. Erzeuge diese und füge in die Leerstellen jeweils zwei Motorbefehle ein. Ändere in der Funktion „turn_left“ die Parameter von „RX_M1“ auf „links“ und „200“ und für „RX_M2“ auf „rechts“ und ebenfalls auf „200“ ab.



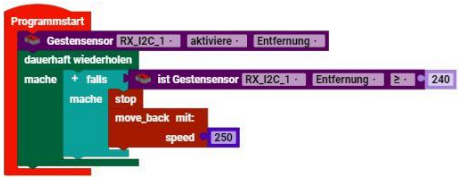
Für die Funktion „turn_right“ müssen die Parameter auf „links“ und ebenfalls auf „200“ gesetzt werden.

Die Aktivierung des Gestensensors hast du schon vorgenommen. Es folgt der Befehl „falls mache“ aus dem Block „Verarbeitung“ – „Logik“.

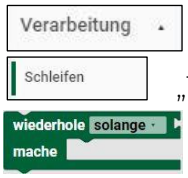
An die Abfrage „falls“ ziehst du aus dem Block „Sensoren“ – „I2C“ den Befehl „ist Gestensensor …“. Ändere auf „Entfernung“, auf „≤“ und „240“ um.

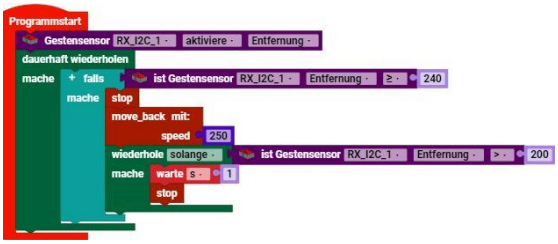
Füge aus dem Block „Verarbeitung“ – „Funktionen“ die Funktion „stop“ ein – d.h.: es werden alle Motoren

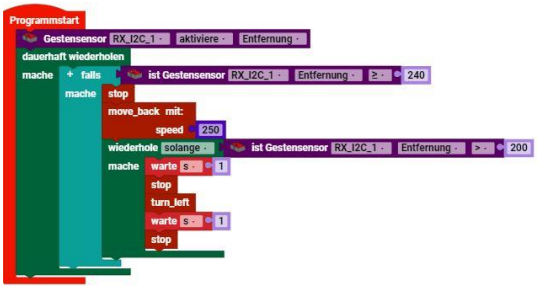
Füge anschließend die Funktion zum Vorwärtsfahren ein. Diese findest du unter „Verarbeitung“ – „Funktionen“. An die freie Andockstelle wird aus dem Block „Verarbeitung“ – „Mathe“ der Befehl „eine Zahl“ eingefügt und von „0“ auf „250“ geändert.



Das Modell fährt jetzt so lange geradeaus bis ein Ereignis „der Abstand zu einem Hindernis wird zu gering“ eintritt. Dies wird in einem „wiederhole solange mache“ und einer Entfernungsabfrage durch den Gestensensor ermittelt. Wenn die Entfernung „>“ – „200“ ist, soll das Modell nach „1“ Sekunde stoppen. Den neuen „Wiederhole solange“ Befehl findest du unter „Verarbeitung“ – „Schleifen“. Den Gestenbefehl kannst du duplizieren und die anschließend die Parameter abändern.

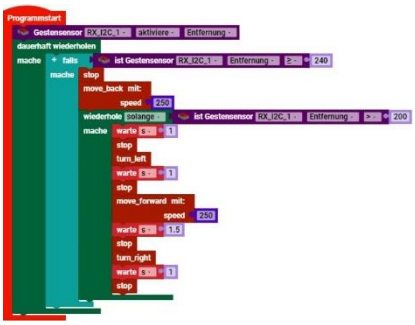


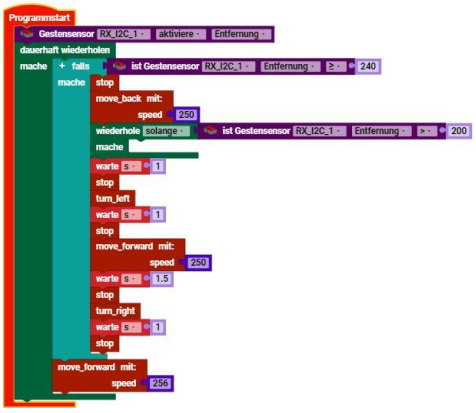




Das Modell soll im nächsten Schritt „1“ Sekunde nach „links“ drehen und dann wieder stoppen. Füge die entsprechenden Funktionen und die Wartezeit ein.

Die letzten Befehle die du in die Schleife einfügen musst, ist die Vorwärtsfahrt mit einer Geschwindigkeit von „250“. Nach einer Pause von „1.5“ Sekunden stoppt das Model und dreht sich „1“ Sekunde nach „rechts“. Dann stoppen nochmals die Motore. Ist die Bedingung nicht erfüllt, d.h. die Entfernung ist immer noch kleiner „200“ wird die Schleife nochmals



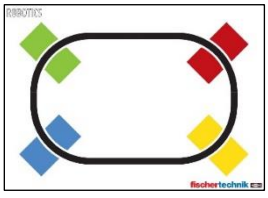


Zum Schluss musst du dem Modell mitteilen, dass es wieder mit einer Geschwindigkeit von „256“ geradeaus fahren soll. Diese Funktion fügst du nach der „falls mache“-Abfrage ein.

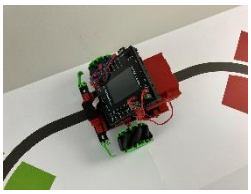
Somit ist das Programm fertig und du kannst es unter dem Namen Advanced_2-wheeled_robot_1 auf deinem Rechner abspeichern. Teste anschließen das Programm zusammen mit dem Modell.

So – und auf zur 2. Aufgabe.

Aufgabe 2



Für die zweite Aufgabe benötigst du, wie du aus der Aufgabenstellung entnehmen kannst, als Fahrstecke die schwarze Linie auf dem im Baukasten beigefügten Parcours.



An der Controllerkonfiguration musst du nichts ändern, da schon alle verwendeten Sensoren und Aktoren definiert wurden.

Als Programmierbasis kannst du die Aufgabe 1 verwenden. Die Funktionen „forward“, „backward“, „stop“, „turn_left“ und „turn_right“ benötigst du auch in der 2. Aufgabe.

Lösche aus dem Hauptprogramm alle Befehle bis auf die Schleife „dauerhaft wiederholen mache“.

Wichtig: Bevor du weitere Programmierung vornimmst, ein technischer Hinweis.

Die beiden IR-Sensoren reagieren auf weiß und schwarz. Anhand einer kleinen Tabelle kann man die IR-Logik darstellen.

IR1	sw	0	IR1	ws	1	IR1	ws	1	IR1	sw	0
IR2	sw	0	IR2	ws	1	IR2	sw	0	IR2	ws	1

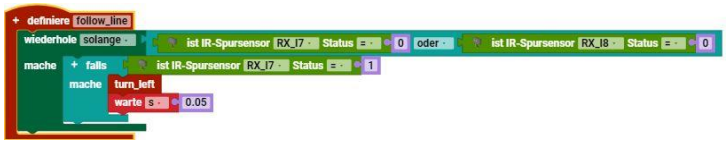
Mit Hilfe des Schnittstellentest kannst du dir die Schaltfunktionen darstellen lassen. Öffne diesen und schiebe dein Modell mit den IR-Sensoren über die schwarze Linie und beobachte die Ausgänge an I7 und I8.

Erzeuge eine neue Funktion mit dem Namen „follow_line“. Füge einen „wiederhole solange mache“-Befehl ein. Anschließend soll überprüft werden, ob IR-Sensor I7 „oder“ IR-Sensor I8 eine „0“ an den Controller übermitteln. Dazu fügst du zuerst aus dem Block „Verarbeitung“ – „Logik“ den Befehl „und“ hinzu. Ändere die Abfrage auf „oder“ ab.

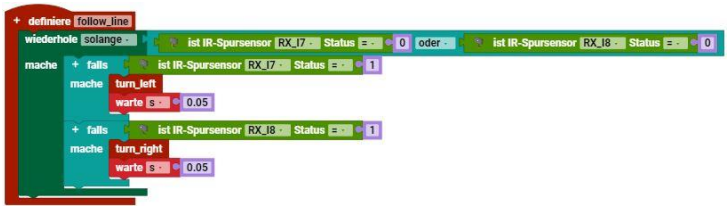
In die beiden freien Stellen fügst du aus dem Block „Sensoren“ – „Eingang“ den Befehl „ist IR-Spursensor …“ ein. Ändere ggf. die Parameter um.

Was geschieht, wenn IR/I7 eine 1 meldet? D.h. IR/I7 hat die schwarze Linie verlassen. Das Modell soll für einen Zeitraum von „0.05“ Sekunden nach „links“ drehen - „turn_left“.

Füge eine „falls mache“ Abfrage neben „mache“ ein. Dupliziere den Befehlsblock „ist IR-Spursucher RX_I7 …“ neben die Abfrage „mache“ und ändere ggf. den Wert auf „1“. Die Korrektur wird so lange ausgeführt, bis der Wert von „RX_I7“ wieder „0“ ist.



In gleicher Weise musst du im nächsten Schritt den Wert von „RX_I8“ abfragen. Dazu duplizierst du die Befehlsblöcke aus der ersten Abfrage und setzt diese unter den „falls mache“-Block. Ändere von „RX_I7“ auf „RX_I8“ um. Tausche die Funktion „turn_left“ mit der Funktion „turn_right“.



Zum Schluss musst du noch eine „falls mache“-Abfrage einbauen in der abgefragt wird, ob beide IR-Sensoren wieder den Wert „0“ melden. Ist dies der Fall, soll das Modell mit der Geschwindigkeit von „250“ vorwärts fahren. Füge also die „falls mache“-Abfrage ein. Dupiziere den „solange“-Block und füge ihn neben „falls“ ein. Ändere „oder“ auf „und“ und die Parameter „0“ auf „1“ um. Es folgt die Funktion „forward mit: speed“ mit dem Wert „250“.

Somit sind alle Funktionen definiert und du kannst das Hauptprogramm fertig stellen. Füge zunächst die Funktion „follow_line“ ein. Laut Aufgabenstellung sollen die Motore stoppen, wenn die Linie verlassen wurde. Dafür fügst du als letzte Funktion „stop“ ein.

Teste das Programm und speichere es unter dem Namen Advanced_2-wheeled_robot_2 auf deinem Rechner ab.

Aufgabe 3

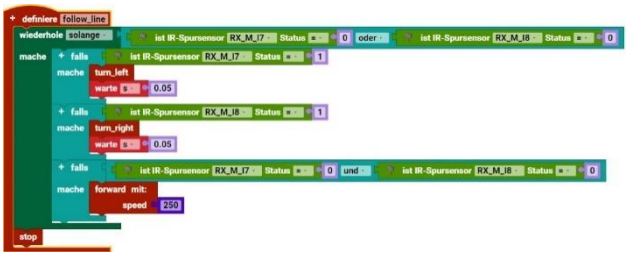
Jetzt geht’s an die 3. Aufgabe. Diese ähnelt der vorherigen Aufgabe – nur, dass zuerst eine schwarze Linie gesucht werden soll. Auch hier vorab das Hauptprogramm mit seinen Unterprogrammen. Neu hinzugekommen ist das Unterprogramm „find_line“.

Mit diesem Unterprogramm wird die schwarze Linie gesucht. Haben beide IR-Sensoren den Wert „0“, stoppen die beiden Motoren und das Unterprogramm wird verlassen. Ansonsten dreht das Modell über das Unterprogramm „turn_left“.

Du benötigst eine Funktion mit dem Namen „find_line“. Füge zuerst den Schleifenbefehl „wiederhole solange mache“ ein. Dupliziere aus der Funktion die IR Sensorabfrage und füge sie nach „solange“ ein. Ändere die Abfrage von „0“ auf „1“ um.

Füge aus den Funktionen noch die Funktion „turn_left“ ein – Drehbewegung nach links.

Bevor die die beide Funktionen „find_line“ und „follow_line“ ins Hauptprogramm einbindest, musst du in der Funktion „follow_line“ nach der Schleife die Funktion „stop“ einbinden.



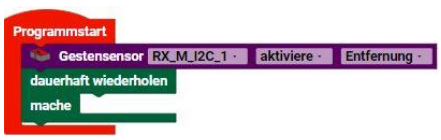
Teste das Programm und speichere es unter dem Namen Advanced_2-wheeled-robot_3 auf deinem Rechner ab.

Aufgabe 4

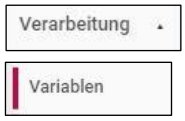
Jetzt aber noch zur 4. Aufgabe. Hier geht es nochmals um die Möglichkeit, der schwarzen Spur zu folgen und evtl. einem Hindernis auszuweichen.

Verwende „Advanced_2-wheeled-robot_3“ als Basisprogramm in dem du verschiedene Veränderungen und neue Funktionen erstellen und einbinden musst.

Beginne mit dem Hauptprogramm. Lösche alle Befehlsblöcke bis auf den Schleifenbefehl. Da in diesem Steuerprogramm auch mit dem Gestensensor gearbeitet wird, kannst du diesen schon einmal vor der Schleife initialisieren.

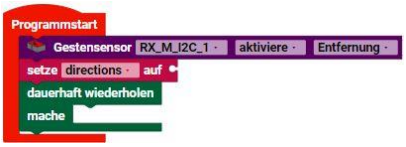


Erzeuge eine „Variable“ mit dem Namen „directions“ – „Wegbeschreibung“ für die „Variable erstellen“ und im Kontextfenster den Variablennamen eingeben.



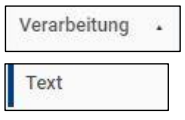
Füge aus „Verarbeitung“ – Variablen“ den Befehl „setze directions auf“ unter den

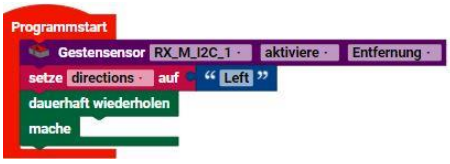
Gestensensor.



Fülle die freie Stelle von „directions“ bzw. definiere auf was die Variable gesetzt werden soll.

Dazu benötigst du den Befehl „ein Buchstabe …“ aus dem Block „Verarbeitung“ – „Text“. Füge diesen ein und klicke auf den Freiraum und gebe im Kontextfenster „Left“ ein.

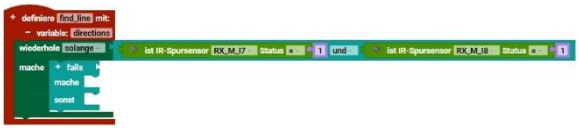


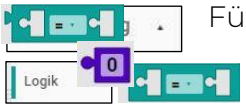


In den Funktionen „turn_left“, „turn_right”, “backward”, “forward” und “stop” müssen keine Veränderungen vorgenommen werden, sodass du diese bei Bedarf in das Programm binden kannst.

In der Funktion „find_line“ musst du mehrere Veränderung vornehmen. Dazu klickst du auf das „+“ vor „definiere“. Der Befehl wird erweitert mit „Variable“. Ändere das „x“ mit „directions“ ab.

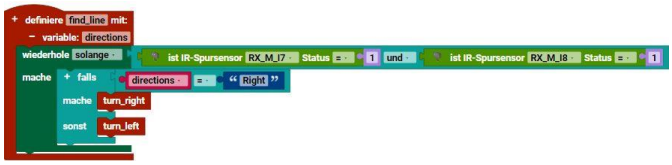
Ziehe die Blöcke aus der Schleife in den rechten Bildschirm und füge aus „Verarbeitung“, „Logik“ den Befehl „falls mache sonst“ ein. Die Abfrage der beiden IR-Sensoren bleiben unverändert.





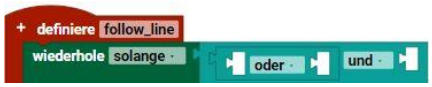
Für „falls“ musst du eine Abfrage eingeben „directions = Right“. Dazu ziehst du aus dem Block „Verarbeitung“ – „Logik“ den Befehl „=“ an die freie Andockstelle.

An die erste freie Stelle fügst du die Variable „directions“ ein. An die zweite Stelle eine „Textvariable“ mit dem Namen „Right“. Wenn die Bedingung erfüllt ist, wird unter „mache“ die Funktion „turn_right“ aufgerufen. Ist die Bedingung nicht erfüllt, wird unter „sonst“ die Funktion „turn_left“ aufgerufen.

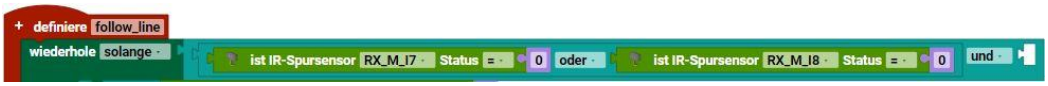


Bevor du das Hauptprogramm vervollständigst, musst du noch die Funktion „follow_line“ anpassen.

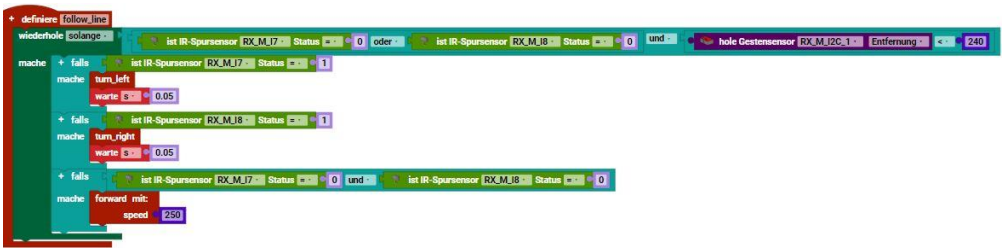
Lösche die Stopfunktion. Erweitere die Abfrage der Wiederholschleife mit „... und Gestensensor ...“ Dazu ziehst du die beiden IR-Sensorabfragen in den leeren Bildschirm. Zuerst benötigst du einen „oder“ Befehl. An die zweite Stelle fügst du einen „und“ Befehl ein.



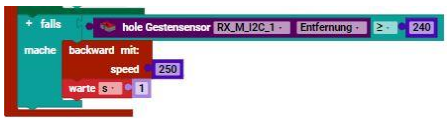
Die ausgelagerten IR-Sensorabfragen kannst du wieder an die beiden „oder“ Andockstellen einbinden.



An die „und“ Position fügst du zuerst den Logik-Befehl „=“ ein. Setze den Entfernungswert auf „240“. Somit ist der erste Teil von „follow_line“ abgeändert bzw. ergänzt.



Im 2. Teil musst du mit „falls mache“ ermitteln, ob die Entfernung des Gestensensors „≥ 240“ ist. Ist dies der Fall, soll das Modell mit einer Geschwindigkeit von „240“ zurück fahren und „1“ Sekunde warten. Füge dazu den Befehlsblock unter dem „wiederhole solange mache“ ein.

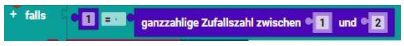


Jetzt noch etwas Besonders. Ein Zufallsgenerator soll vorgeben, in welcher Richtung das Hindernis umfahren werden soll. Den Block der benötigten Befehle findest du im rechte Programmteil.

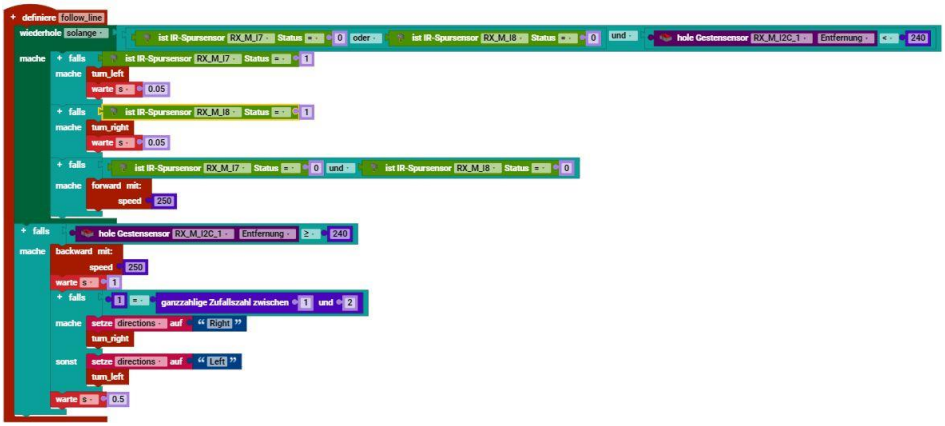
Zahl“ aus dem Block „Verarbeitung“ – „Logik“ eingefügt. Der Wert wird auf „1“ gesetzt.

In den zweiten Abfragebereich setzt du den Befehl „ganzzahlige Zufallszahl …“. Hier musst du vorgegebenen Werte auf „1“ und auf „2“ setzen. Den Befehl findest du im 

Block „Verarbeitung“ – „Mathe“.

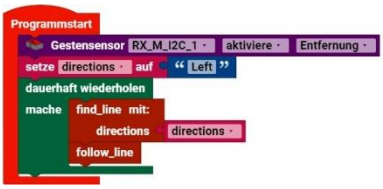
Wenn der Vergleich „1“ mit der Zufallszahl 

übereinstimmt, wird unter „mache“ die Variable „directions“ auf „Right“ gesetzt. Anschließend wird der Befehl „turn_right“ ausgeführt. Die Befehlsfolge kannst du aus dem Programmstart duplizieren und den Text auf „Right“ abändern. Dupliziere den beiden „mache“ Befehl in den Bereich „sonst“. Ändere auf „Left“ ab und tausche „turn_right“ mit „turn_left“. Zum Schluss fügst du noch eine Wartezeit von „0.5“ Sekunde ein.



Jetzt muss noch das Hauptprogramm im Bereich „mache“ ergänzt werden. Hier fügst du zuerst aus dem Block “Verarbeitung“ – „Funktionen“ die Funktion „find_line mit: …“ ein. Als Variable fügst du „direction“ ein. Als letzten Befehl setzt du „follow_line“ darunter.

Stelle auf dem Parcours einfach ein/zwei Hindernisse auf und starte das Programm. Wenn alles funktioniert, weicht der Roboter problemlos diesen aus.



Speichere das Programm unter dem Namen Advanced_2-wheeled_robot_4 auf deinem Rechner ab.

Demontiere anschließend das Modell und widme dich der nächsten Aufgabe.