Robotik

Programmierung mit LEGO Spike

Los geht`s!

MINTCLUSTERWETZLAR

WISSENSCHAFT • WIRTSCHAFT • WACHSTUM

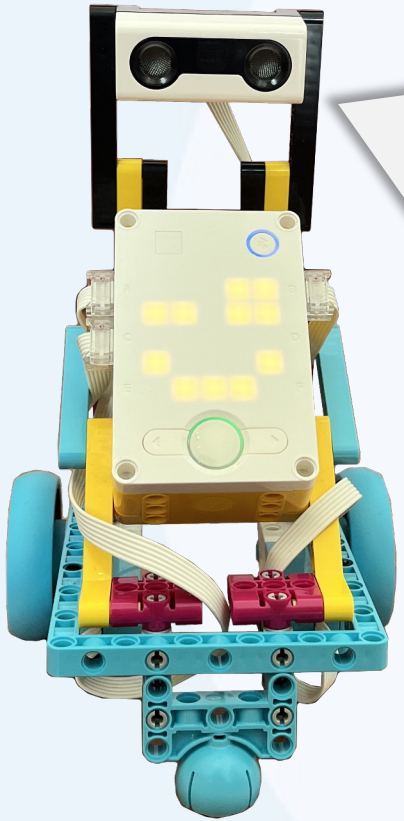
Wir begeistern für MINT und fördern die Berufs- und Studienorientierung

Bundesministerium für Bildung und Forschung

VISEUM Wetzlar – Haus der Optik + Feinmechanik

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN

STADT WETZLAR

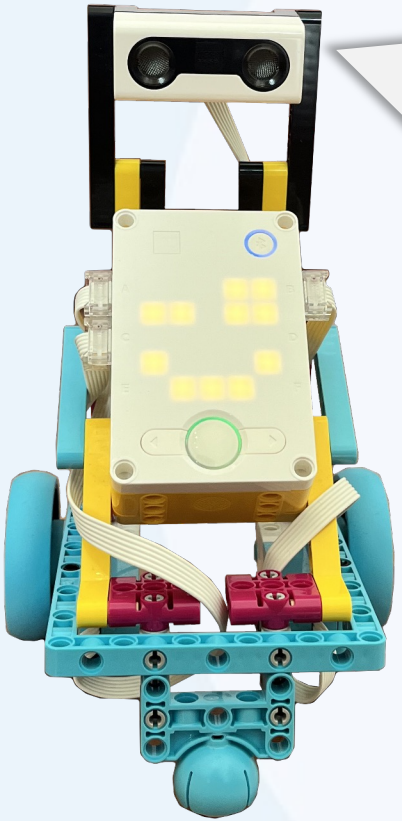


Hallo,
ich bin Spike und begleite euch durch das Programm.

Um durch die Seiten zu blättern, könnt ihr diese Pfeile verwenden.

Es gibt auch Hilfe-Buttons ?, mit denen ihr zu einer Hilfeseite gelangt.

Und jetzt geht`s los. Viel Spaß!



Und noch eine **wichtige** Sache:

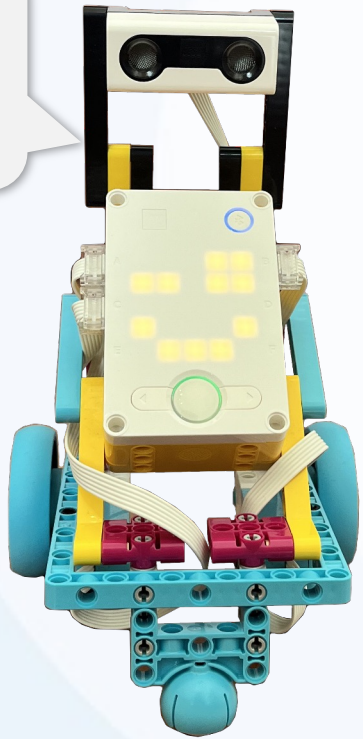
Probiert bei jeder einzelnen Aufgabe aus was sich ändert, wenn ihr die ein oder die andere Sache umstellt oder anders programmiert. Und versucht euch dann gegenseitig zu erklären, warum etwas nicht oder anders funktioniert hat. Solltet ihr es euch gegenseitig einmal nicht erklären können, fragt eine Betreuerin / einen Betreuer um Rat.

Das ist der **beste** Weg, um Robotik und Programmieren zu verstehen.

Roboter zusammenbauen

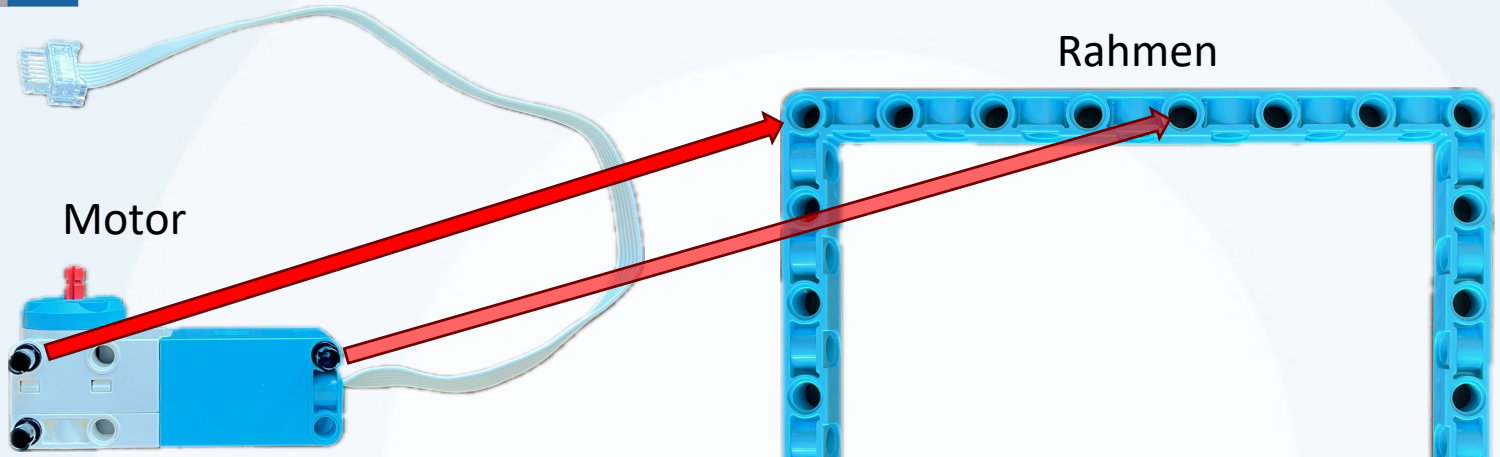
Bevor das Programmieren los gehen kann, müsst ihr zuerst den Roboter zusammenbauen.

Wechselt euch beim Bauen am besten ab!

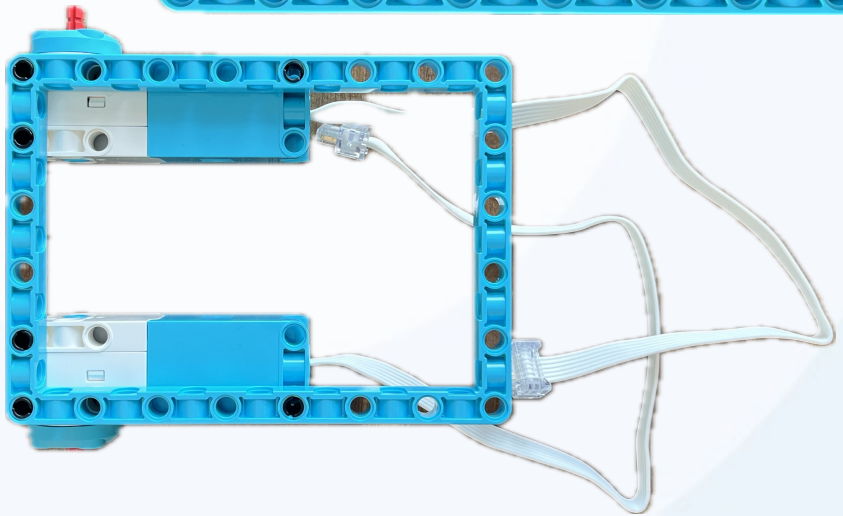


Schritt 1

Benötigte Bauteile:

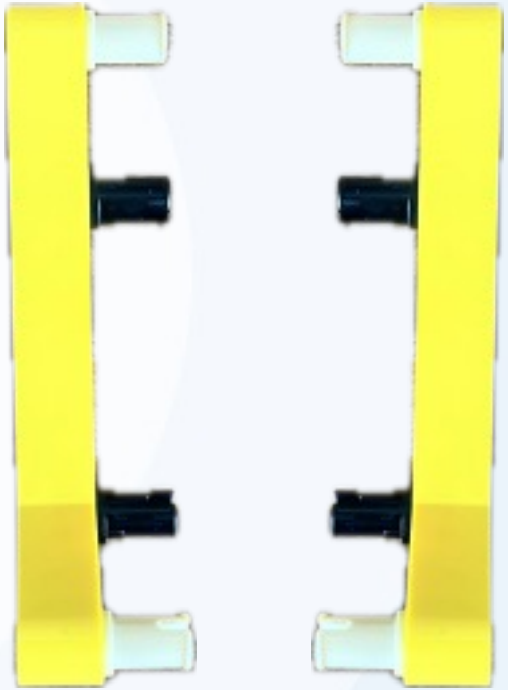


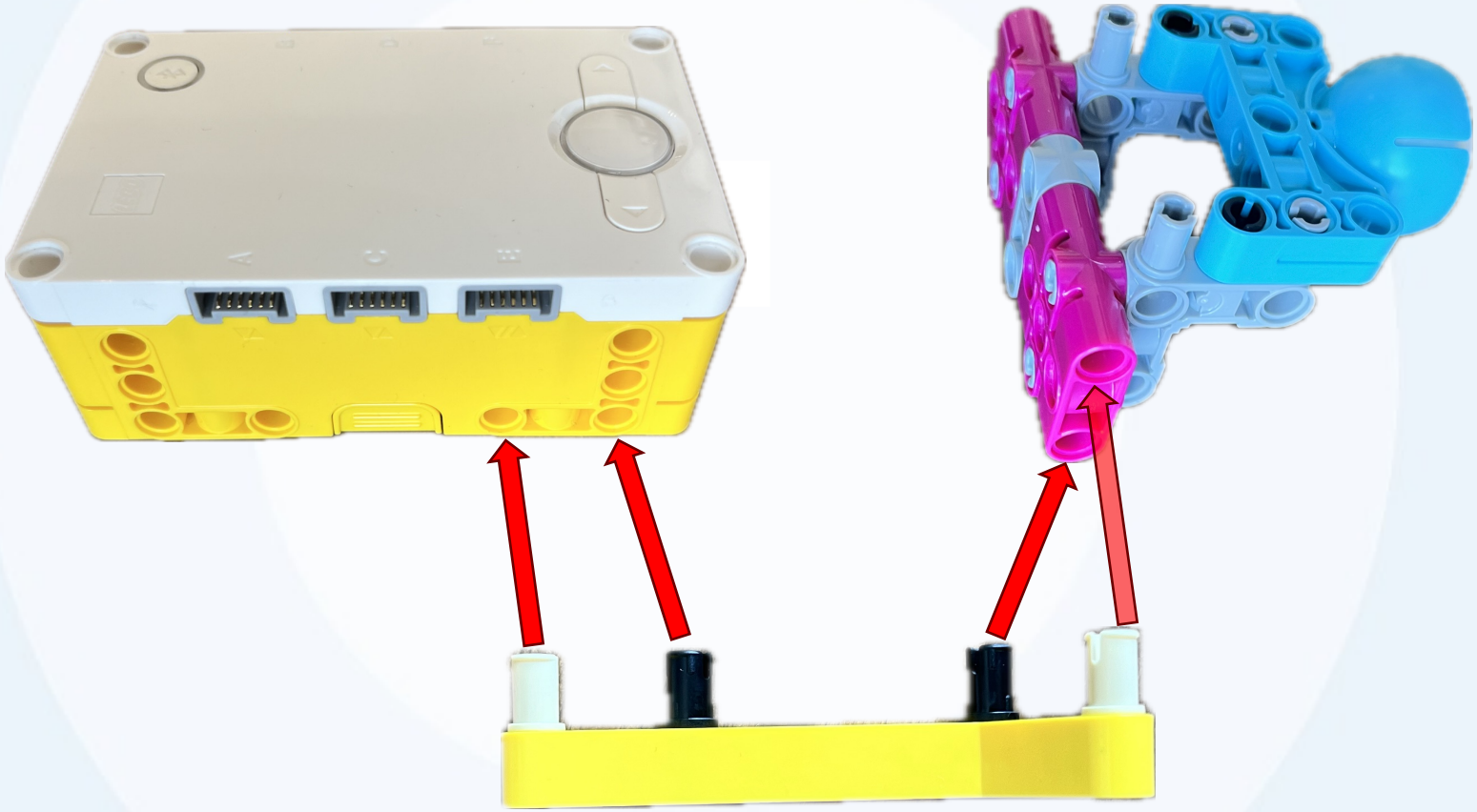
Platziert den Rahmen auf dem Motor, sodass die Stecker am Rahmen mit den Löchern zusammen passen, wie es die Pfeile anzeigen. Macht das auch für den anderen Motor. Danach sollte es so aussehen:



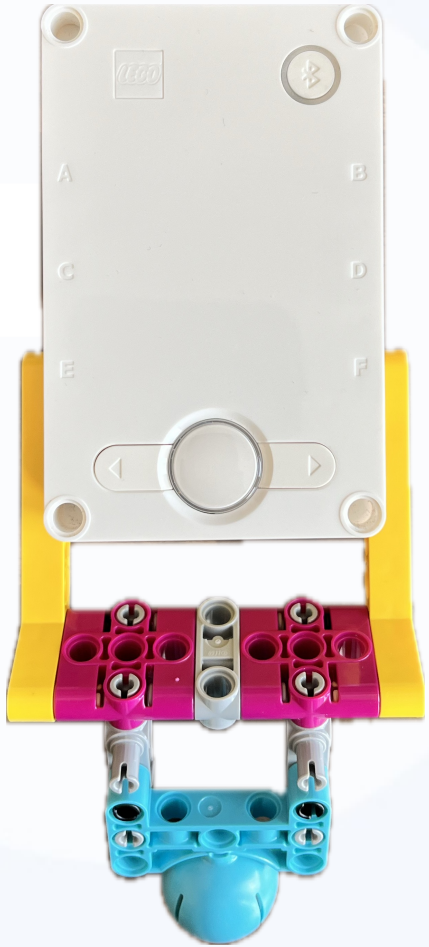
Schritt 2

Benötigte Bauteile:



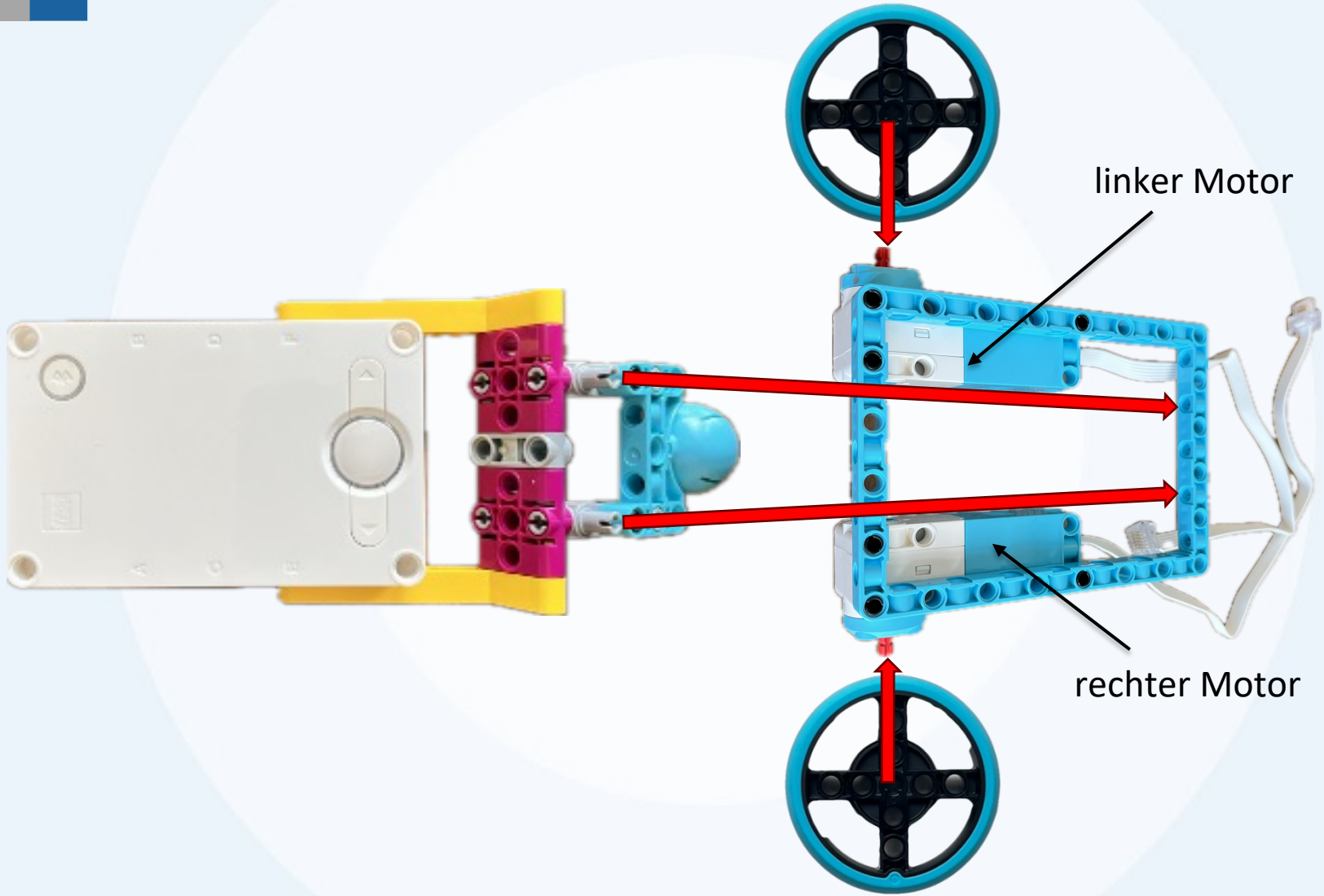


Auf der anderen Seite auch!
Danach sollte es so aussehen:



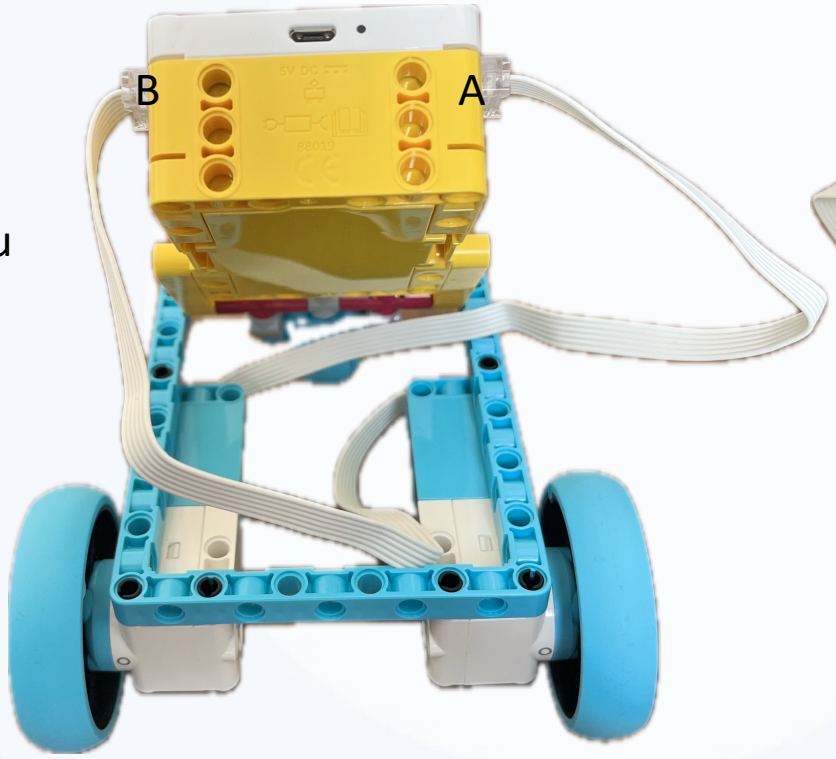
Schritt 3

Benötigte Bauteile:


linker Motor
rechter Motor

Steckt die Motorkabel dabei überkreuz in die Anschlüsse A und B des Hub. (rechter Motor zu B und linker Motor zu A)

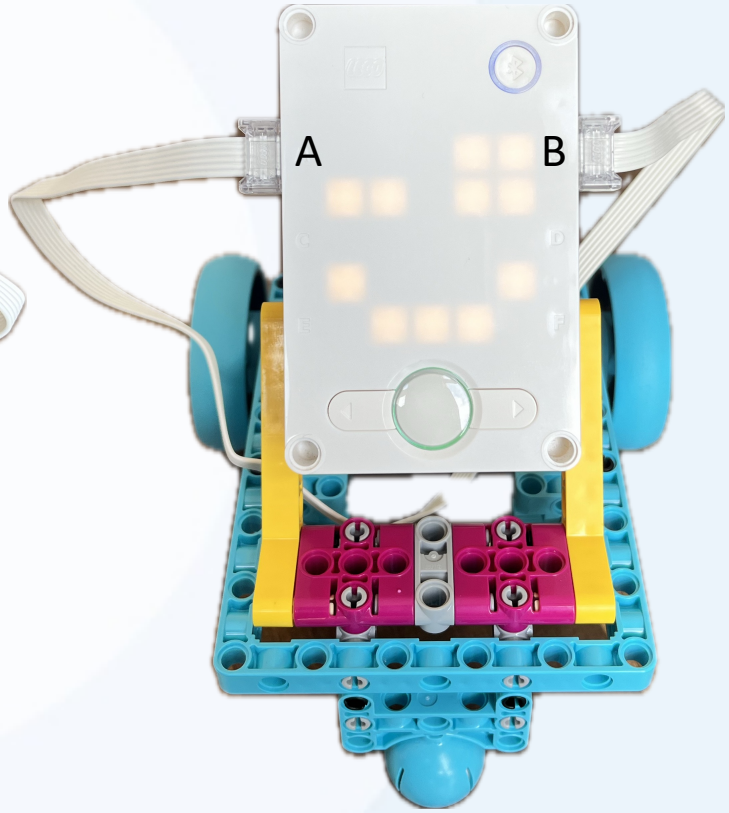
Rückseite



linker Motor

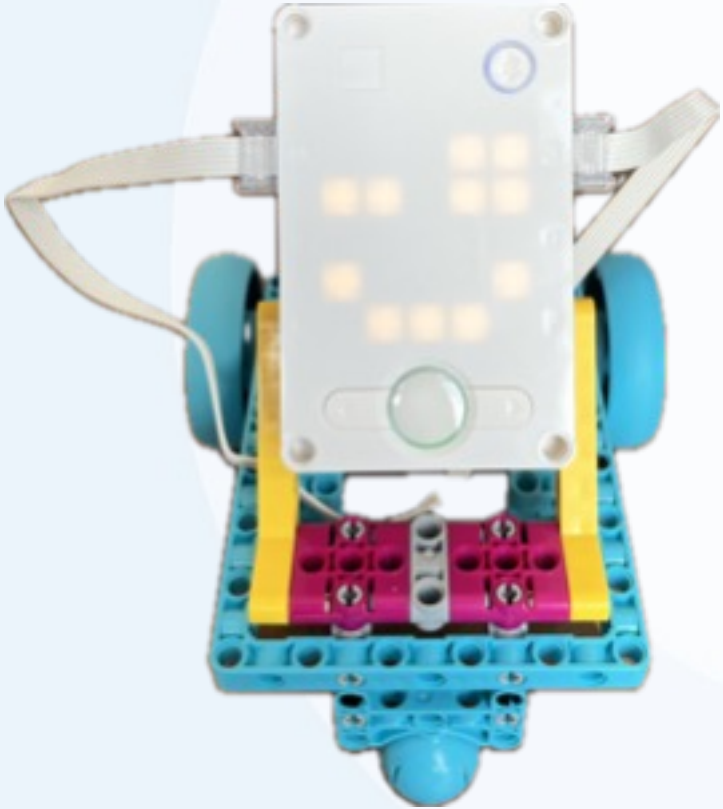
rechter Motor

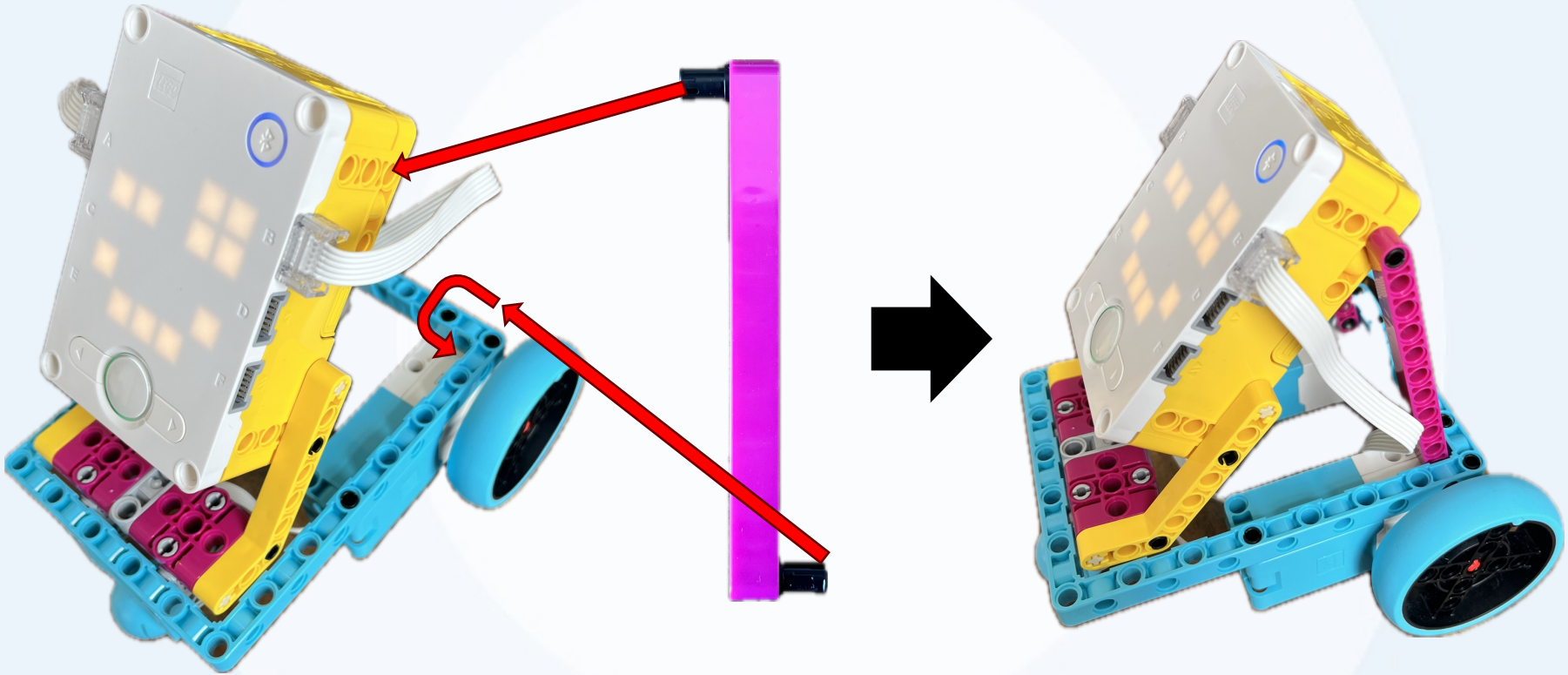
Vorderseite

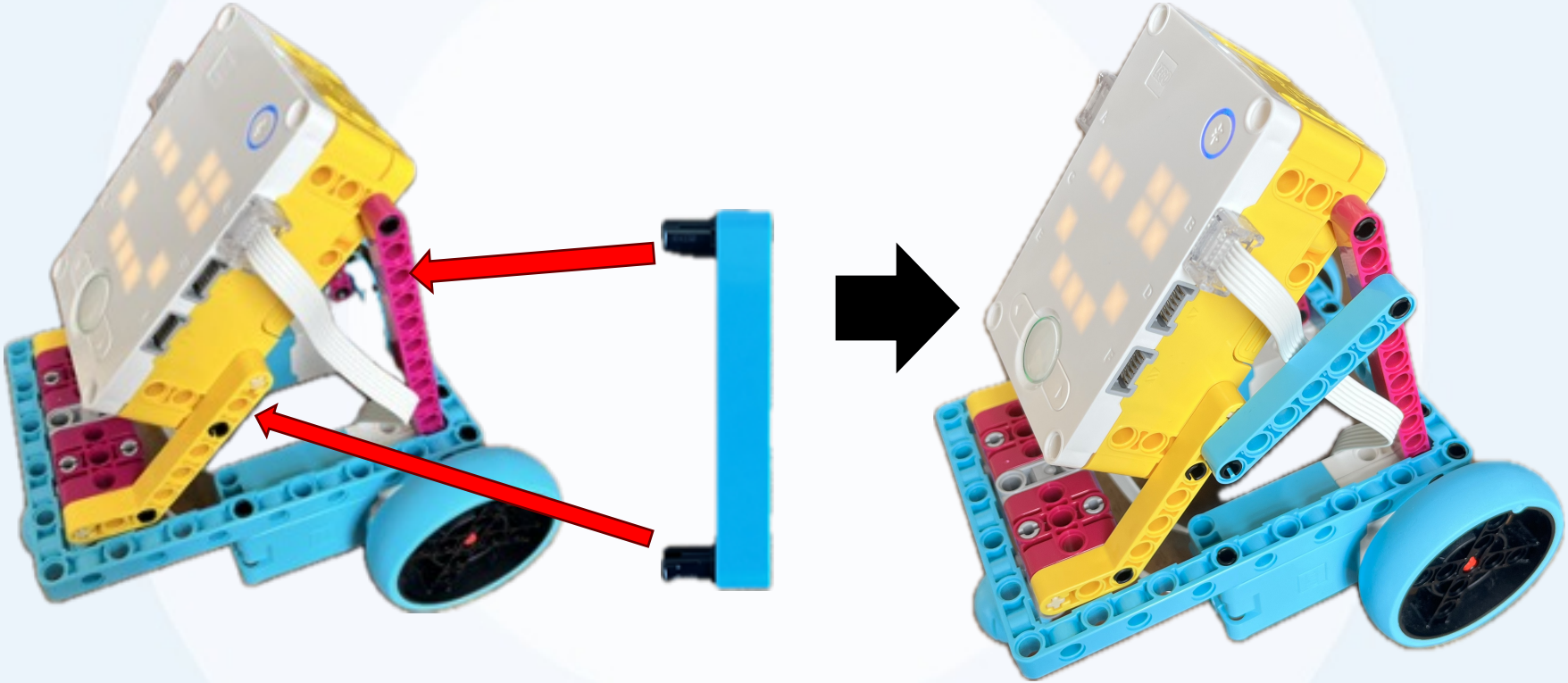


Schritt 4

Benötigte Bauteile:

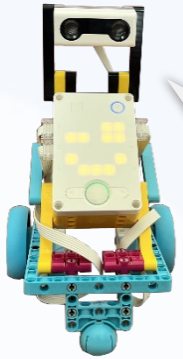






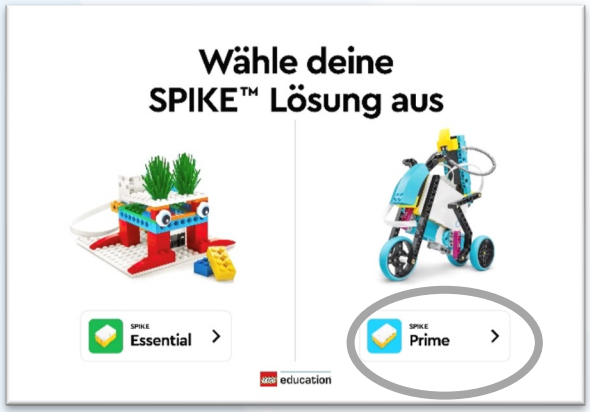
Auf der anderen Seite auch!

Start des Programms

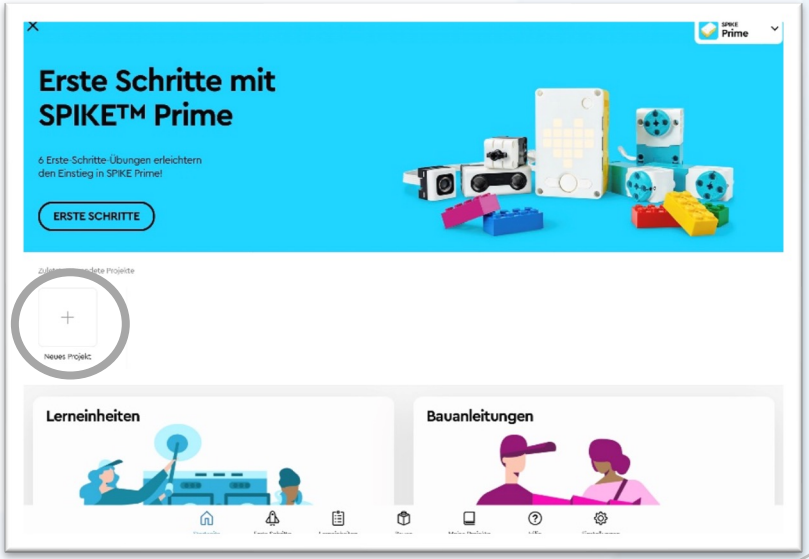


Los geht`s!

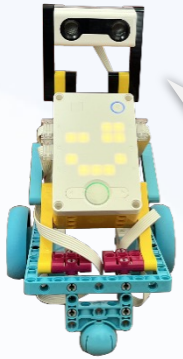
Öffnet das Programm und wählt *Spike Prime* aus.



Klickt dann auf *Neues Projekt*.

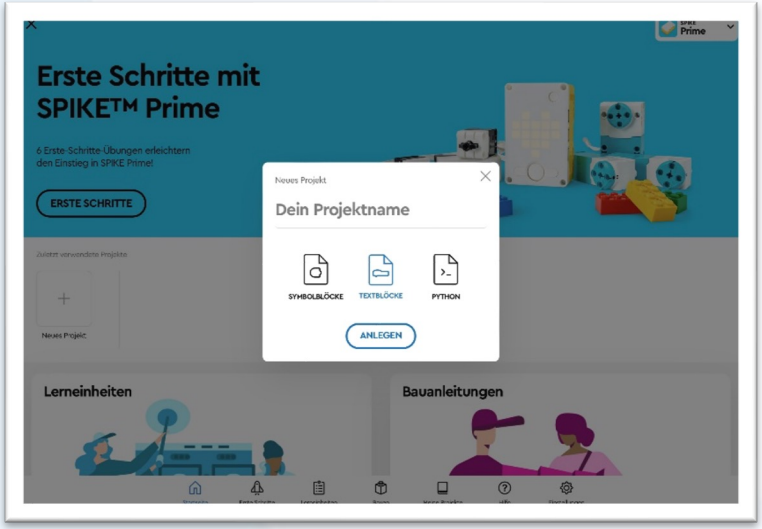


Start des Programms

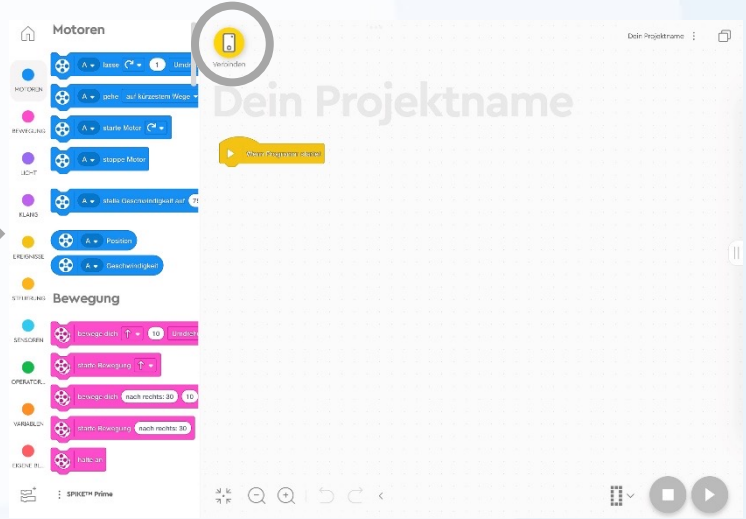


Los geht`s!

Jetzt gebt eurem Projekt einen selbstgewählten Namen und wählt die *Textblöcke* Option aus.



Klickt auf der nächsten Seite auf *Verbinden*.

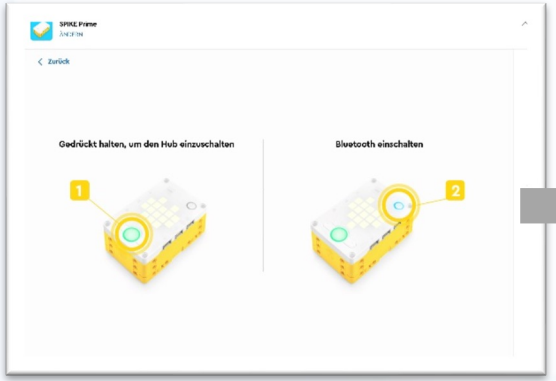


Spike-Hub verbinden

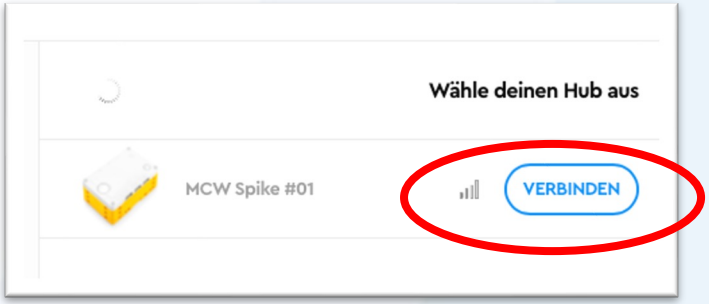
Der Spike-Hub kann über Bluetooth mit dem Programm verbunden werden.

Startet den Hub mit dem großen runden Knopf. (kurz gedrückt halten)

Folgt den Anweisungen, um den Stein zu verbinden.



Am rechten Rand findet ihr alle Namen der Hubs, die in Reichweite sind. Wählt das passende aus und klickt auf verbinden.



Anleitung: Verbinden über Bluetooth

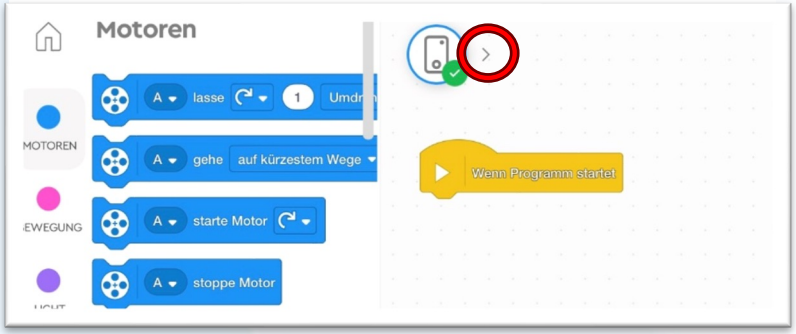
Auf der Unterseite des Hubs findet ihr die Hub-Nummer.

Zurück

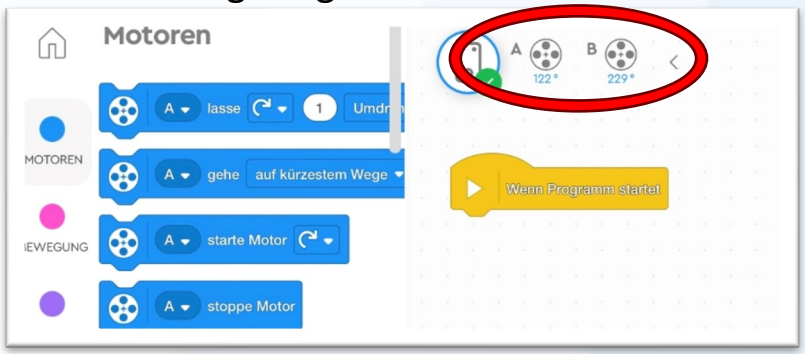
Spike-Hub Statusanzeige

Nach dem Verbinden seht ihr oben links einen grünen Haken neben dem Hub-Symbol.

Klickt auf den Pfeil neben dem Hub-Symbol, um die Statusanzeige auszuklappen.



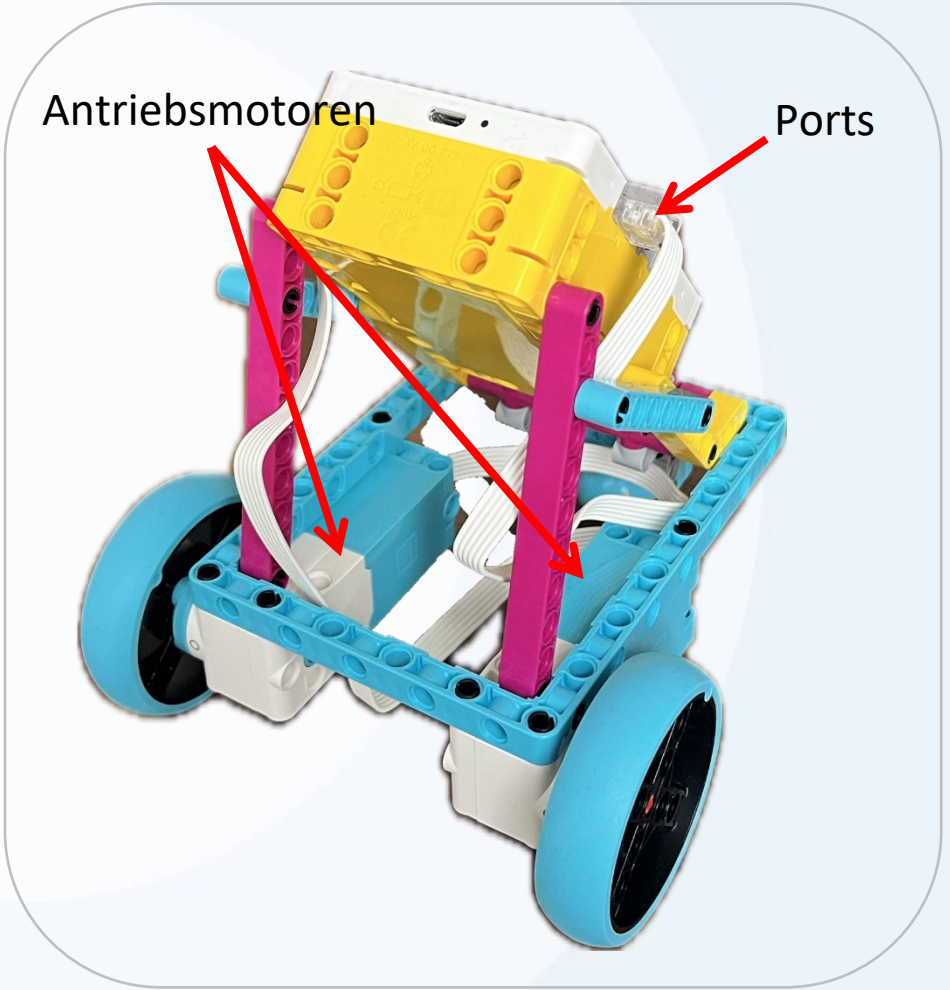
Hier wird euch für die angeschlossen Motoren oder Sensoren der aktuelle Zustand angezeigt.



Info: Antriebsmotoren

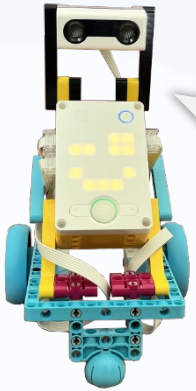
Der Roboter besitzt zwei Antriebsmotoren, über die die Räder und damit Bewegungen gesteuert werden.

Die Antriebsmotoren sind über die Anschlüsse (auch Ports genannt) A und B mit dem Hub verbunden, das dafür zuständig ist die Programmierung an die Antriebsmotoren weiterzugeben.

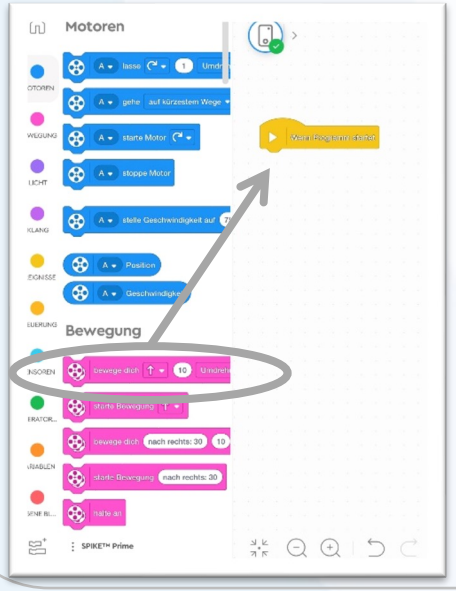


Vorwärts fahren

Programmiert den Roboter so, dass er ein Stück geradeaus fährt.

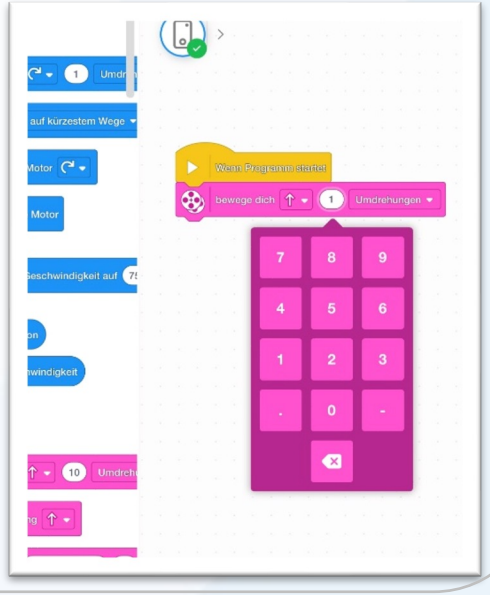


Achtung!
Ich brauche mind.
20 cm Platz, um nicht
herunterzufallen.



Ziehe den Block unter den Start-Block

Stellt die Umdrehungszahl auf 1 und den Pfeil nach oben ein, um euren Roboter vorwärts fahren zu lassen.



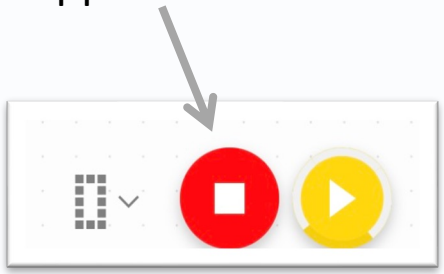
Vorwärts fahren

Programmiert den Roboter so, dass er ein Stück geradeaus fährt.

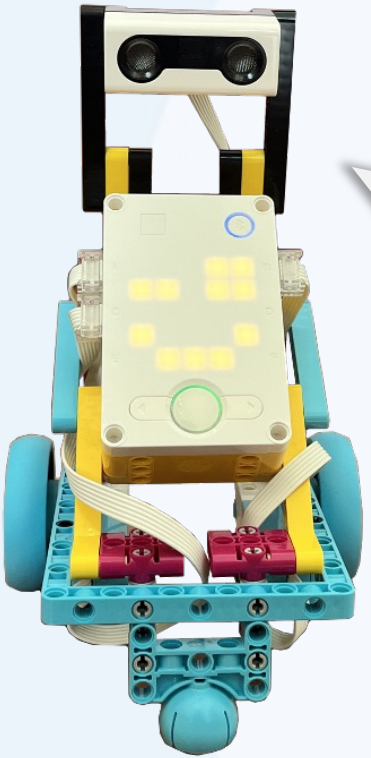
Achtung!
Ich brauche mind. 20 cm Platz, um nicht herunterzufallen.

Testet das Programm, indem ihr auf die Play-Taste klickt.

Stoppen könnt ihr das Programm mit der Stopp-Taste.



Probiert auch gerne selbst etwas herum, z.B. könntet ihr auch statt 1 Umdrehung vorwärts, 10 cm rückwärts einstellen. Wisst ihr wie man das hin bekommen könnte?
Achtet aber darauf, dass der Roboter nicht vom Tisch fällt.



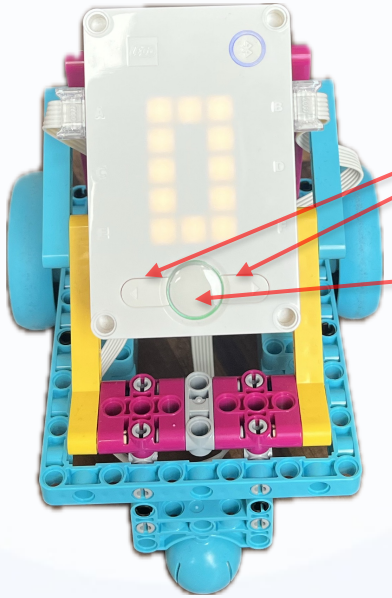
Gut gemacht!
Euer Roboter ist zum ersten Mal gefahren.

Tipps zum Löschen von Blöcken:
Blöcke nach links wegziehen
<u>oder</u>
auf den Block gedrückt halten und “Lösche Block“ wählen.

Info: Programm auf Stein starten/stoppen

Ihr könnt das Programm auch direkt auf dem Hub am Roboter starten.
Dafür muss das Programm einmal auf dem iPad gestartet worden sein.

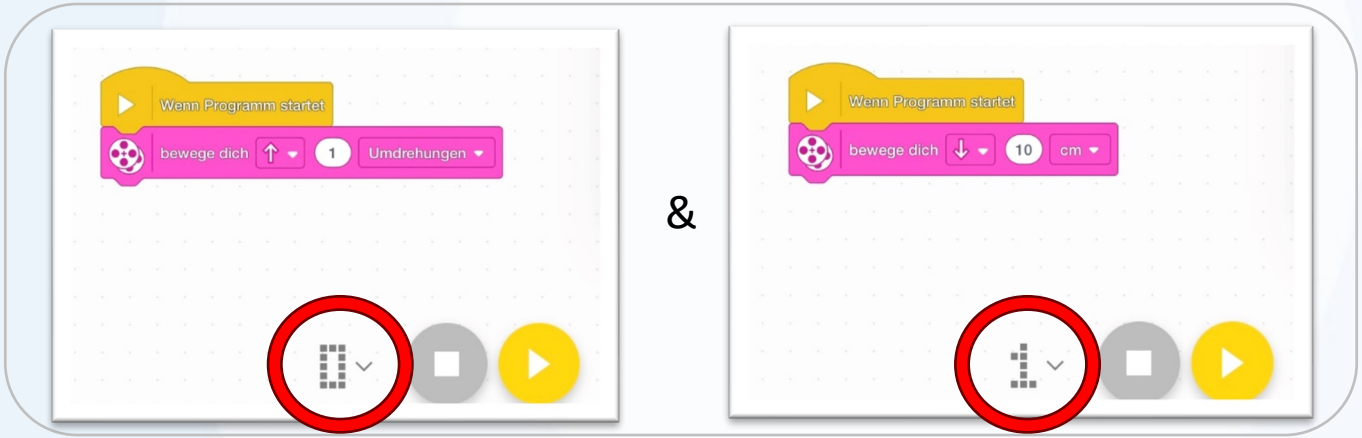
Vielleicht ist euch hier schon die 0 aufgefallen: Damit könnt ihr das Programm auf dem Hub unter dieser Nummer speichern.



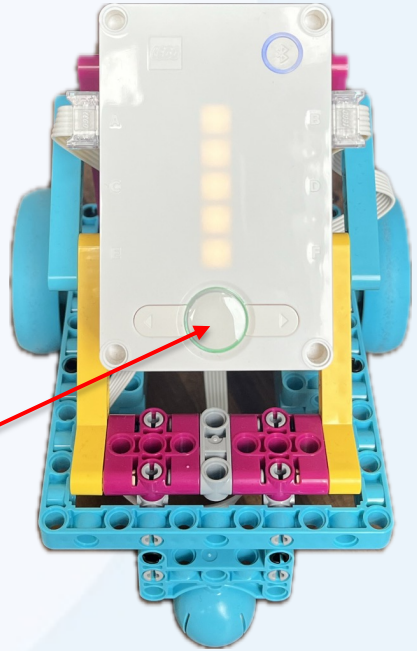
Mit den Pfeiltasten auf dem Hub könnt ihr dann zwischen den Nummern auswählen und mit der Haupttaste das Programm starten oder stoppen.

Info: Programm auf Stein starten/stoppen

Probiert es aus: Erstellt das Programm von eben, mit dem der Roboter eine Umdrehung nach vorne fährt, wählt die Nummer 0 aus und klickt auf die Play-Taste. Danach ändert das Programm, sodass der Roboter 10 cm rückwärts fährt, wählt die Nummer 1 aus und drückt wieder auf die Play-Taste.

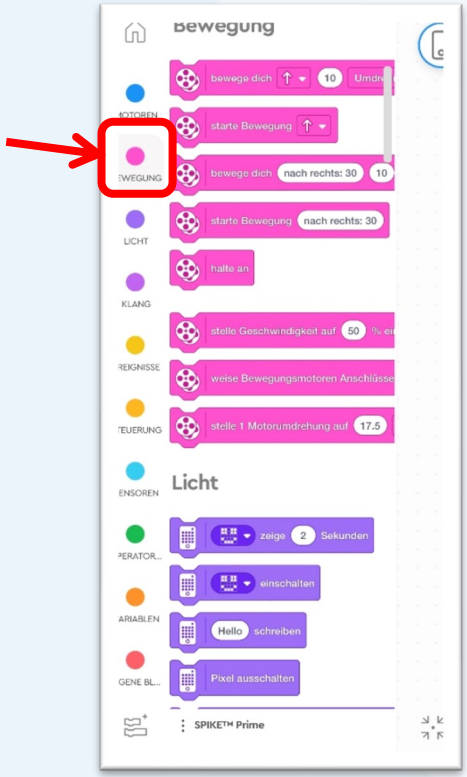


Jetzt könnt ihr die beiden Programme auch über den Hub auswählen und starten.



Fahrtrichtung

Programmiert euren Roboter so, dass er jeweils die Aufgaben in den grauen Kästen ausführt. Nutzt dazu verschiedene Bewegungsblöcke (pink).



Eine weite Linkskurve fahren

Rückwärts fahren

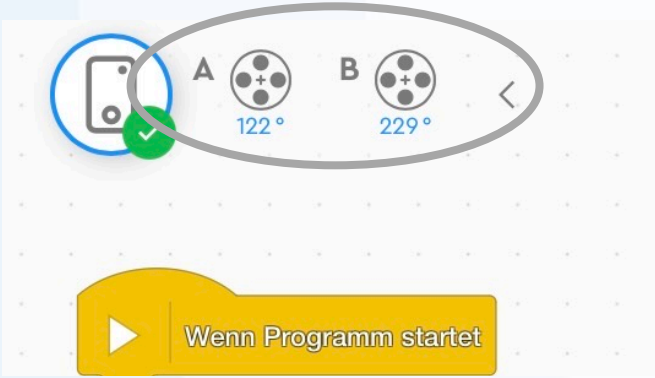
Eine enge Rechtskurve fahren

Testet euer Programm auf dem Boden, damit der Roboter nicht herunterfällt!

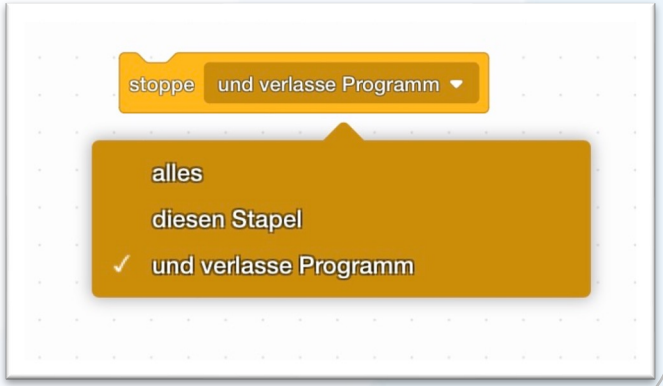
Es gibt diese Blöcke um die Fahrtrichtung zu ändern:

Info: Anzeige der Sensoren und Programm automatisch beenden

Mit dieser Anzeige könnt ihr erkennen, wie weit sich die Motoren gedreht haben.



Ein Programm könnt ihr auch automatisch beenden lassen, wenn ihr zum Schluss diesen Block anfügt.



Geschwindigkeit und Fahrdauer

Programmiert euren Roboter so, dass er die Aufgaben in den grauen Kästen ausführt. Setzt dazu verschiedene Bewegungsblöcke untereinander.

Eine weite Linkskurve und dann eine enge Rechtskurve

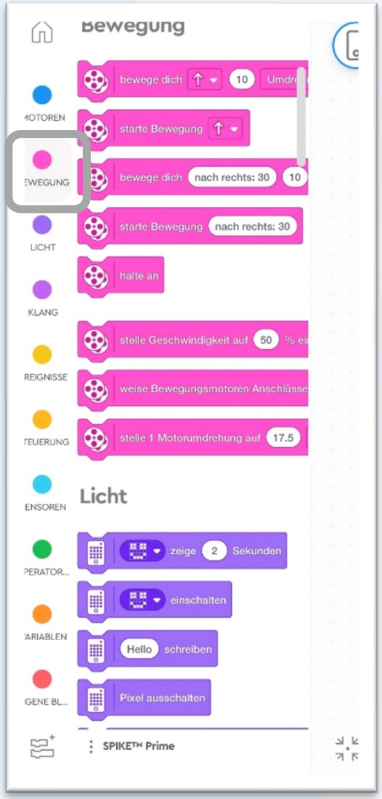
Eine schnelle Linkskurve und dann eine langsame Rechtskurve

Testet euer Programm auf dem Boden, damit der Roboter nicht herunterfällt!

Tipp: Ihr könnt die Geschwindigkeit mit diesem Block einstellen

Unbegrenztes Fahren

Ihr könnte euren Roboter so programmieren, dass er eine unbegrenzte Bewegung ausführt.



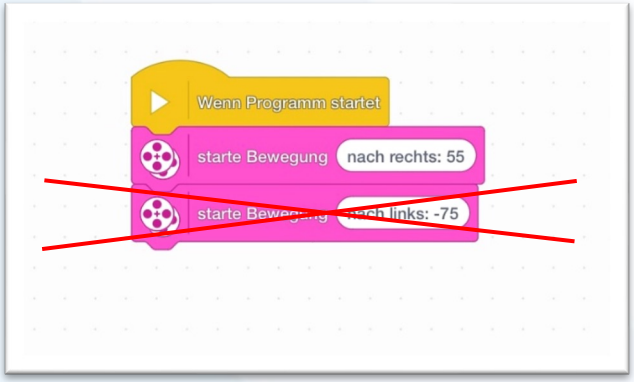
Findet heraus, welche Blöcke euren Roboter unbegrenzt fahren lassen.

Erinnerung:
Ich höre auf zu fahren, wenn ihr unten auf den **runden** Knopf drückt.

Info: Bewegungsblöcke aneinander hängen

Wenn mehrere Bewegungen nacheinander programmiert werden sollen, müsst ihr folgendes beachten:

Auf Bewegungsblöcke, die euren Roboter unbegrenzt fahren lassen, können nicht direkt weitere Bewegungsblöcke folgen.



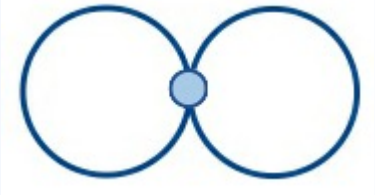
Wenn zwei verschiedene Bewegungen aufeinander folgen sollen, dann nutzt die Blöcke, die den Roboter für eine begrenzte Umdrehungszahl oder Zeitspanne fahren lassen.

Schwierigere Aufgaben

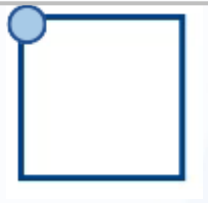
Schaut hier nach, wie ihr ein Programm speichern könnt.

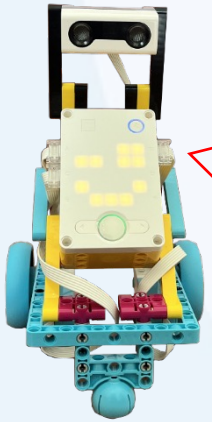
Super! Ihr seid bereit für schwierigere Aufgaben.
Bearbeitet **beide** der Aufgaben in den grauen Kästen.
Speichert beide Aufgaben in zwei separaten Projekten.

Eine Acht fahren



Ein Quadrat fahren





Ihr müsst euer Programm vermutlich mehrmals testen.

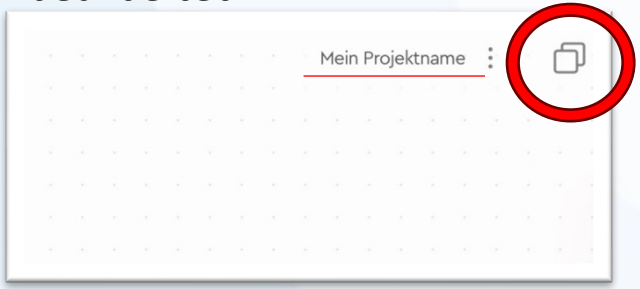
Wenn ihr mit eurem Ergebnis zufrieden seid, zeigt es einem Betreuer / einer Betreuerin.

Info: Speichern

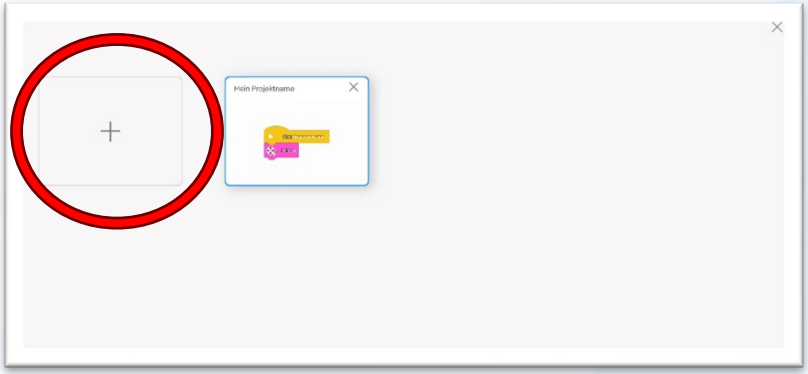
Ihr könnt eure erstellten Programmierungen speichern. Speichert jede Aufgabe in einem separaten Projekt, das ihr entsprechend benennt, z. B. „Quadrat“ oder „Acht“.

Zurück

Klickt vor einer neuen Aufgabe oben rechts auf das Symbol für die Projekteübersicht.
Hier könnt ihr auch sehen, welches Projekt ihr gerade bearbeitet.

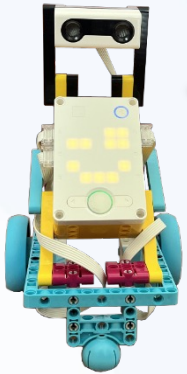


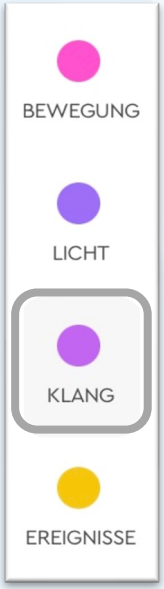
Klickt auf das “+” Symbol und wählt die “Textblöcke” Option und gebt dem Projekt den neuen Namen.



Soundeffekt-Block

Verwendet einen Soundeffekt-Block, damit euer Roboter ein Geräusch oder ein Wort abspielt.





Fügt einen Soundeffekt-Block an beliebigen Stellen in eurem Programm ein.

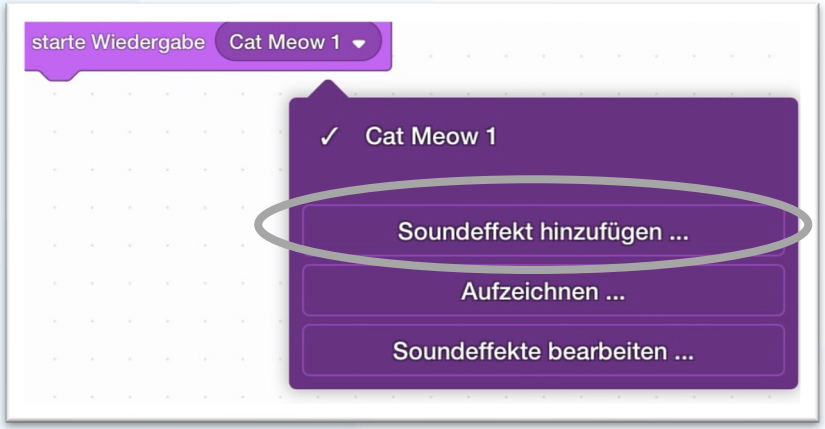
Hinweis: Bei dem oberen Block wartet der Roboter, bis der Soundeffekt ganz abgespielt wurde und setzt dann das Programm fort.

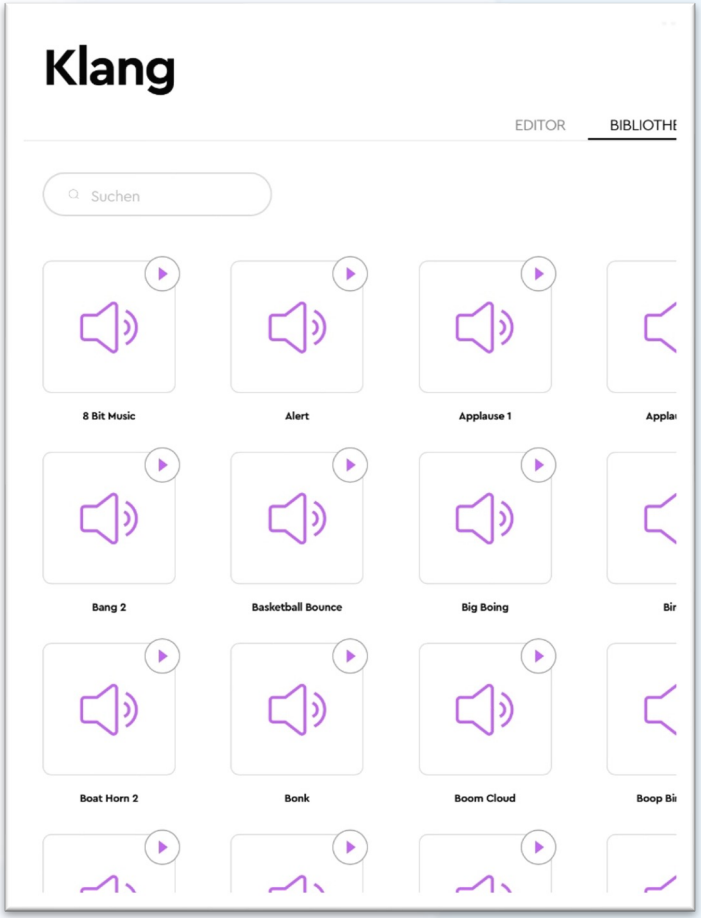
Der untere Block spielt den Soundeffekt während des Programms ab.

Auswahl Soundeffekt

Es gibt eine große Auswahl verschiedener Soundeffekte.

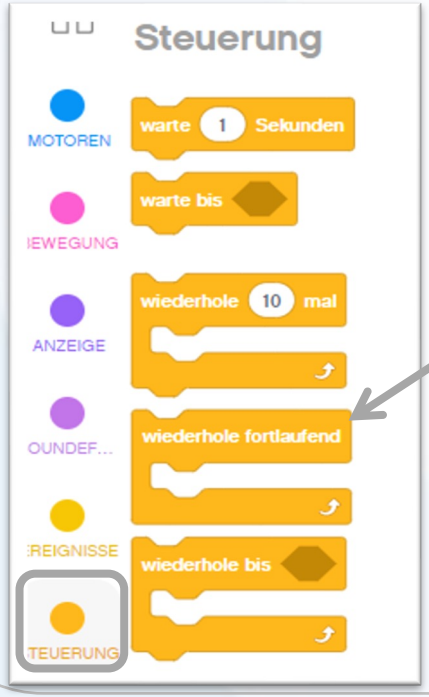
Klickt im Soundeffekt-Block mittig auf das Auswahlfeld und dann auf „Soundeffekt hinzufügen“. Habt ihr euch neue aus der Bibliothek ausgesucht, könnt ihr hier zwischen sie hier auswählen.





Wiederholtes Fahren

Ihr habt bereits programmiert, dass der Roboter eine Acht bzw. ein Quadrat fährt. Öffnet das jeweilige gespeicherte Programm und versucht, euren Roboter zunächst dauerhaft eine Acht fahren zu lassen.

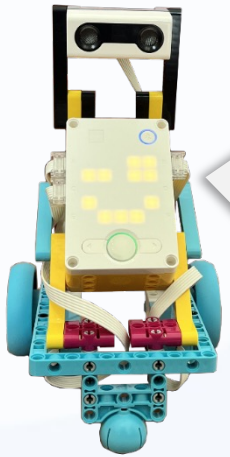


Nutzt für diese Programmierung den „wiederhole fortlaufend“-Block.

Sobald euer Roboter dauerhaft eine Acht fährt, ändert das Programm, so dass er dauerhaft ein Quadrat fährt.

Programm verkürzen

Wenn ihr euch die Programmierung der vorherigen Aufgabe anschaut, fällt euch sicher auf, dass sich einige Blöcke wiederholen. Könnt ihr die Programmierung verkürzen? Probiert es aus.



Es kann sein, dass ihr das schon gemacht habt. In diesem Fall: Direkt weiter zur nächsten Folie!

Info: Iteration

Da Wiederholungsblöcke Programmierungen wiederholen, könnt ihr Dopplungen in der Programmierung geschickt einsparen.

Die Nutzung von Wiederholungsblöcken, die bestimmte Programmierungen wiederholt ausführen, nennt man Iteration.

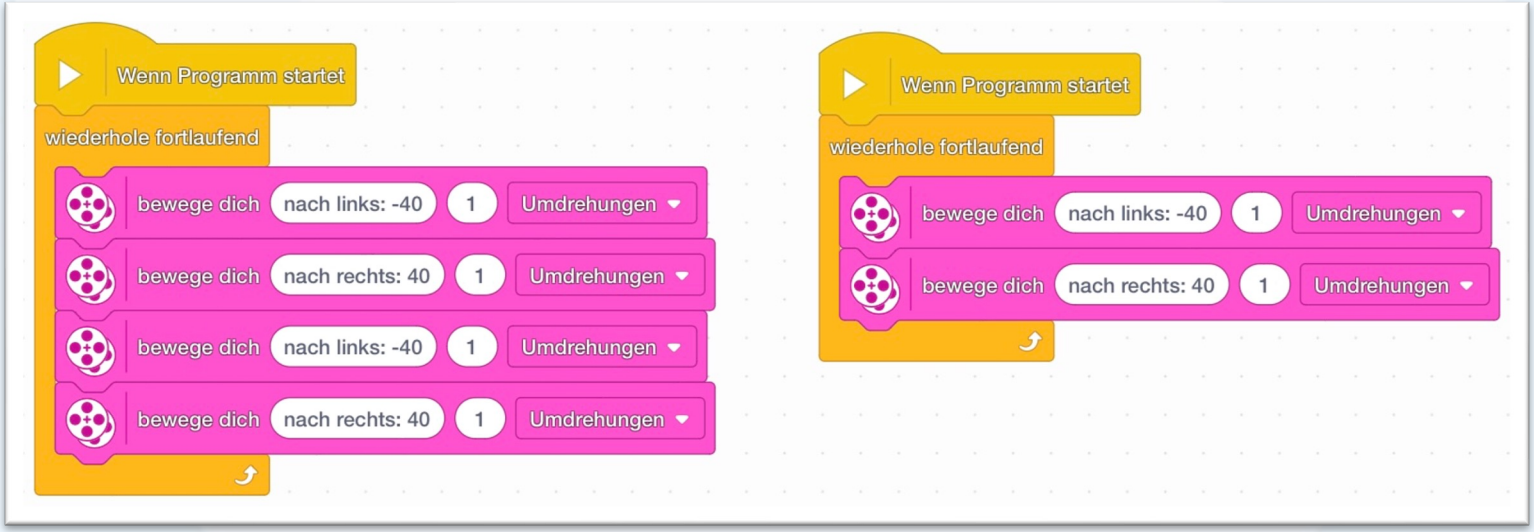
Hinweis: Diese Wiederholungsblöcke werden auch als Schleifen bezeichnet.

Beispiel: Damit der Roboter dauerhaft ein Quadrat fährt, wiederholt man die Blöcke „geradeaus fahren“ und „um 90 Grad um eine Kurve fahren“ mit einem Wiederholungs-Block, anstatt alle Abschnitte mehrfach untereinander zu hängen.

Geschicktes Programmieren

Beim Programmieren gibt es häufig viele verschiedene Möglichkeiten eine Aufgabe umzusetzen. Oft gibt es dabei geschicktere und weniger geschicktere Lösungen.

Unten seht ihr zwei Programme, durch die der Roboter eine Schlangenlinie fährt. Überlegt gemeinsam, welche Programmierung geschickter umgesetzt ist und warum.



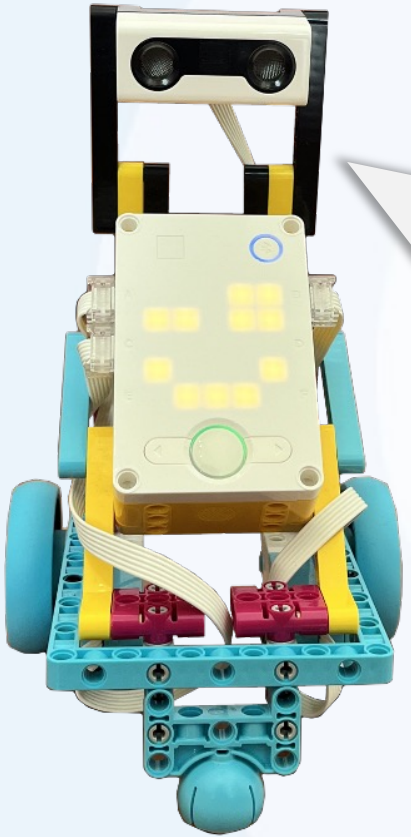
Patrouille

Das Prinzip der Iteration kann für geschicktes Programmieren genutzt werden.

Programmiert den Roboter so, dass dieser sich zwischen zwei Punkten gradlinig hin und her bewegt. Immer, wenn der Roboter ein Ende der Strecke erreicht, soll er einen Piepton ausgeben, sich drehen und zurückfahren.

Hier könnt ihr nochmal das Prinzip der Iteration nachlesen.

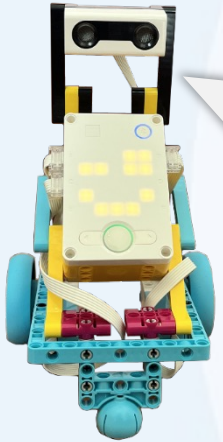
?



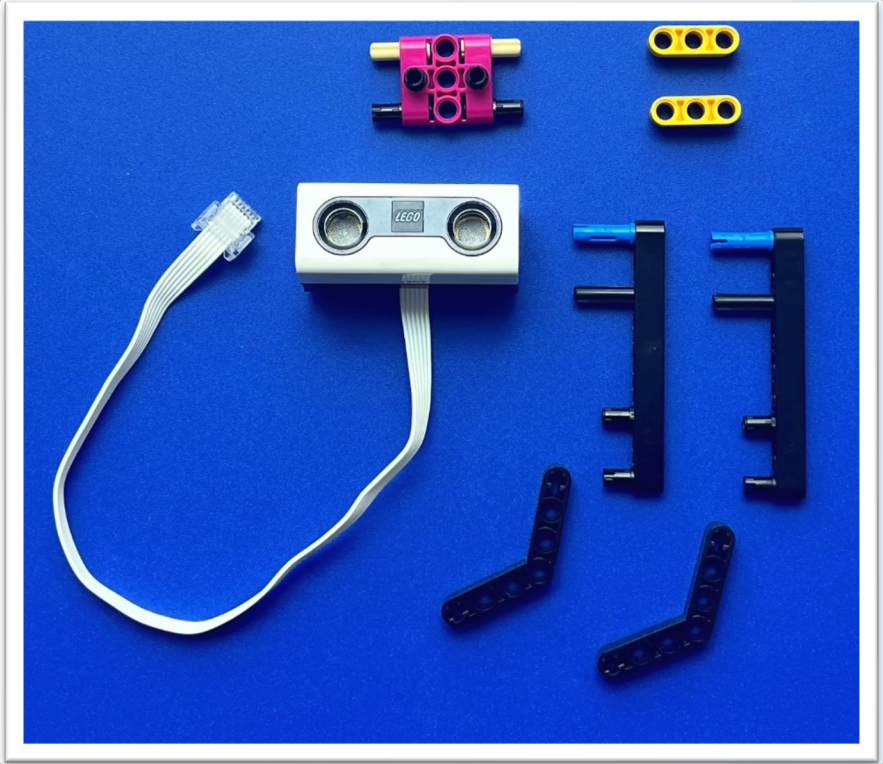
Jetzt wird es Zeit, dass euer Roboter lernt, seine Umgebung wahrzunehmen. Dazu braucht er Sensoren.

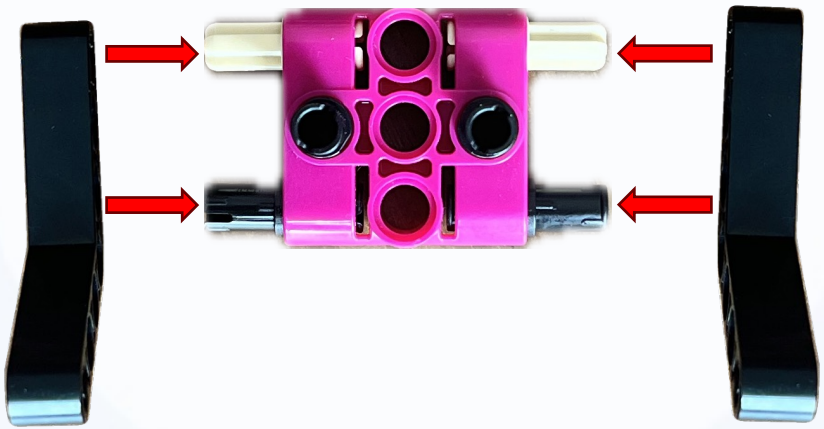
Ein Sensor ist ein Bauteil, das eine Eigenschaft seiner Umwelt (z. B. Helligkeit, Lautstärke, Abstand, Temperatur) erfassen kann.

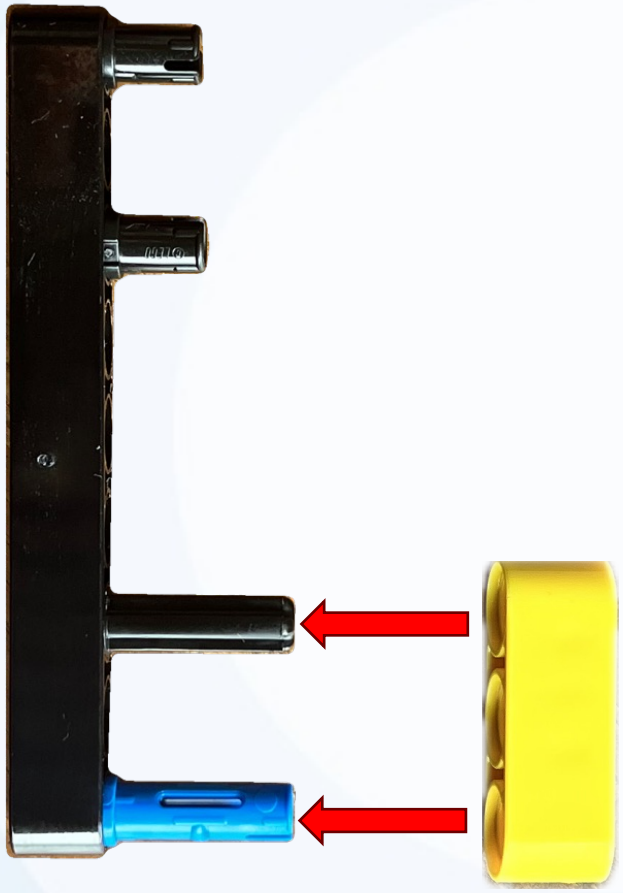
Ultraschallsensor anbauen



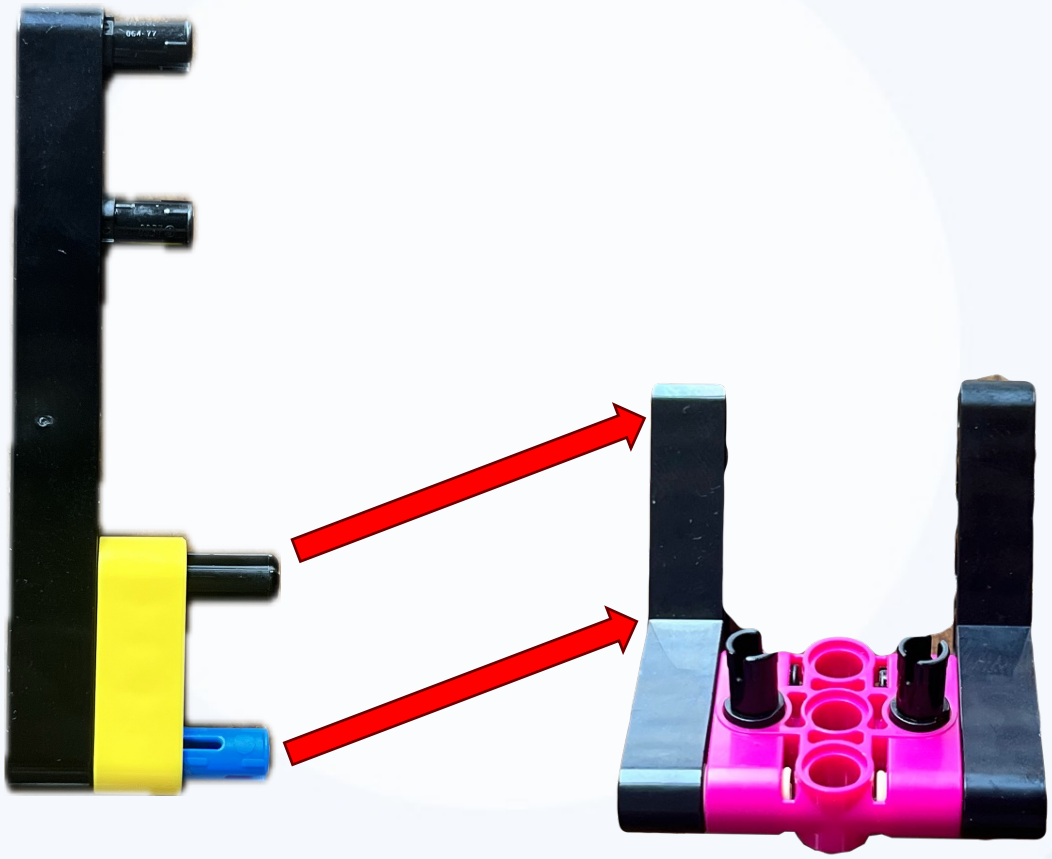
Baut jetzt den Ultraschallsensor zur Abstands-erkennung nach der Anleitung zusammen.

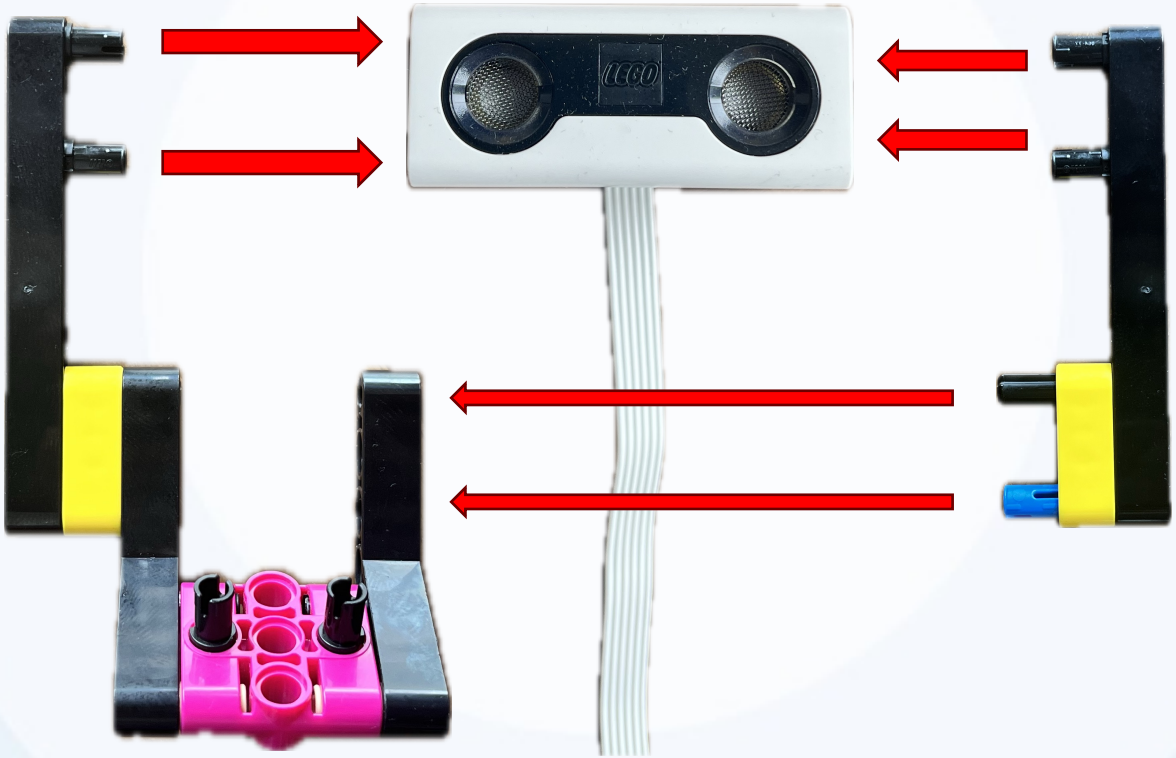


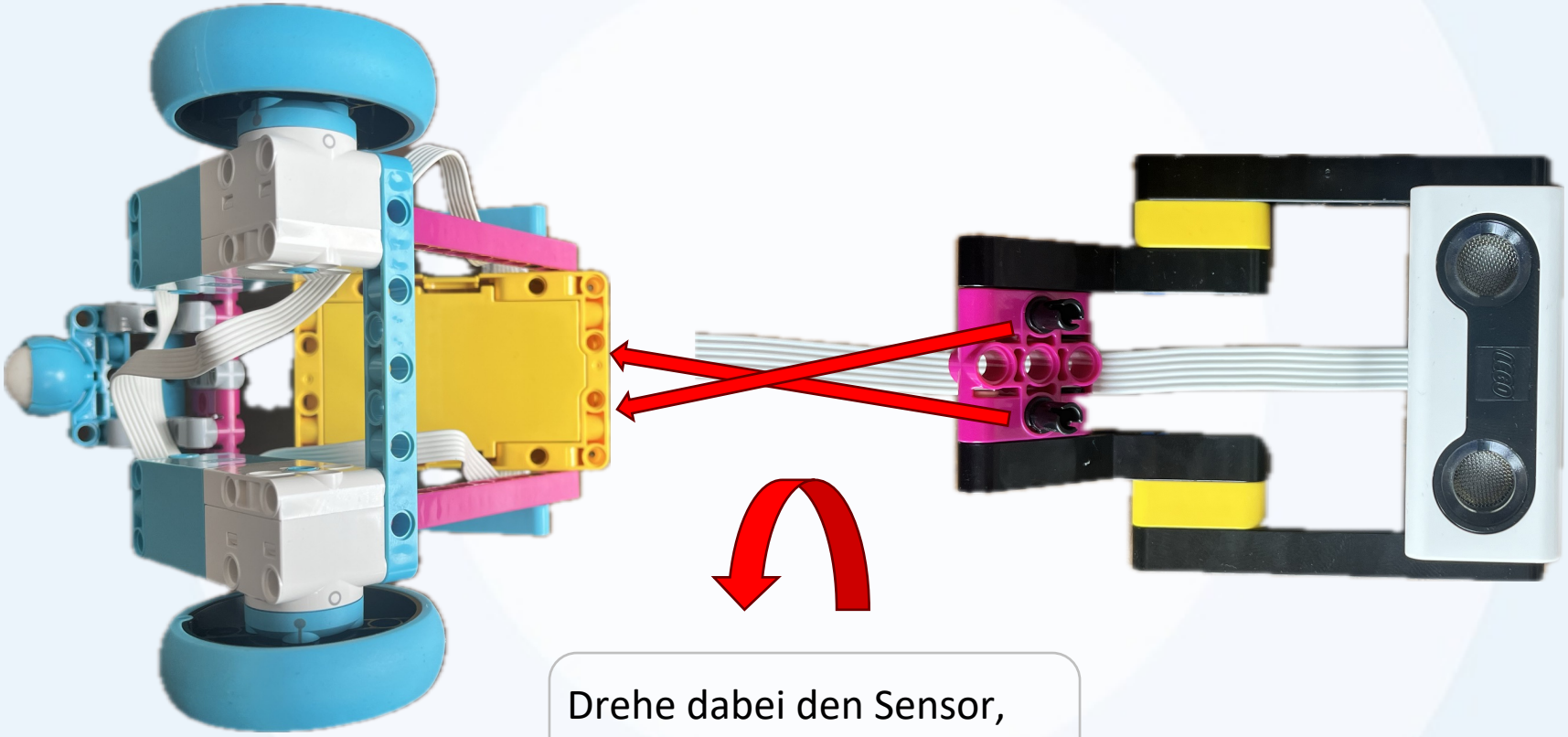




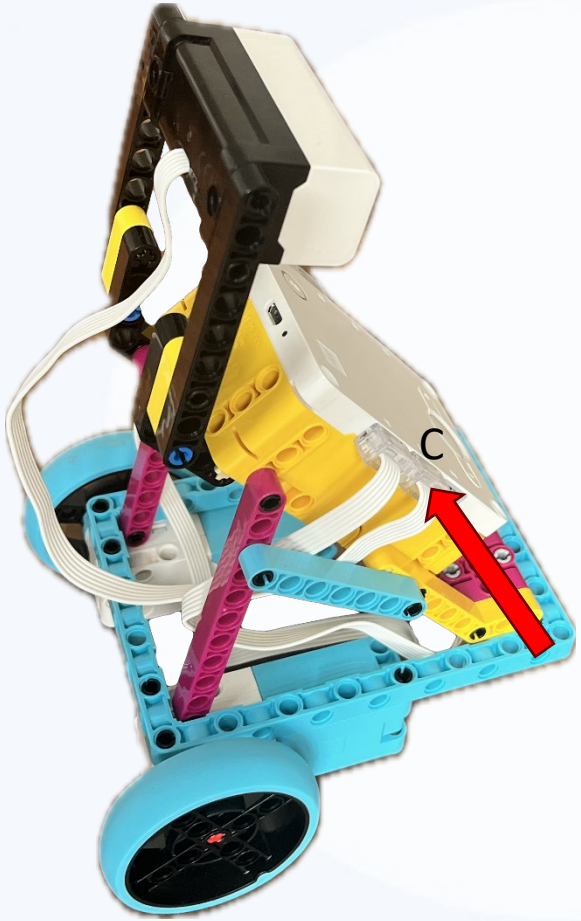
Macht das direkt 2 mal!







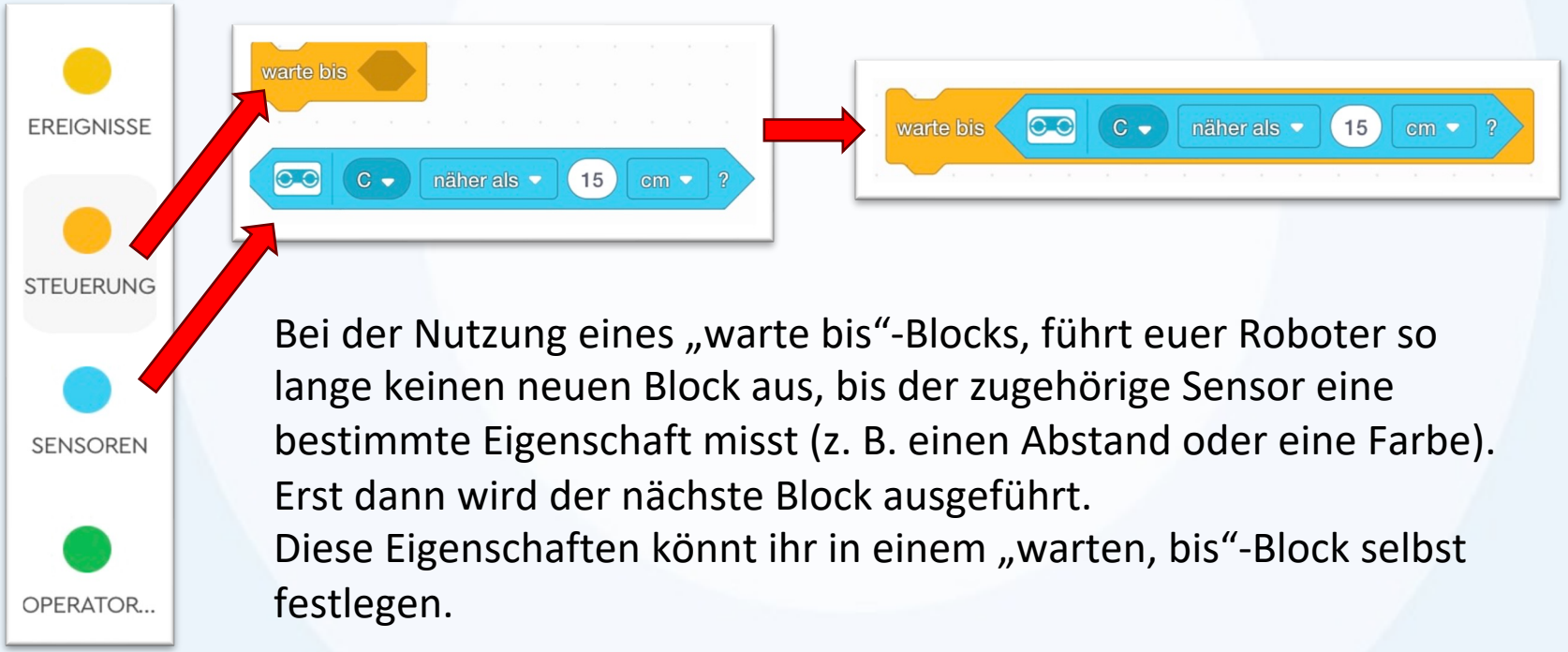
Drehe dabei den Sensor,
sodass er nach forne blickt.



Schließt das Kabel des Sensors an Port C an.

Info: Sensoren und „warte bis“-Block

In dem Tab „Steuerung“ findet ihr den “warte bis“-Block und unter „Sensoren“ könnt ihr für die verschiedenen Sensortypen entweder Bedingungsblöcke (die mit den spitzen Seiten) oder Abfrageblöcke (die mit den abgerundeten Seiten) finden.



Bei der Nutzung eines „warte bis“-Blocks, führt euer Roboter so lange keinen neuen Block aus, bis der zugehörige Sensor eine bestimmte Eigenschaft misst (z. B. einen Abstand oder eine Farbe). Erst dann wird der nächste Block ausgeführt.
Diese Eigenschaften könnt ihr in einem „warten, bis“-Block selbst festlegen.

Vor einem Hindernis anhalten

Programmiert euren Roboter nun so, dass dieser geradeaus fährt und dann stoppt, wenn er 30 cm oder weniger vor einem Hindernis steht.

Überlegt gemeinsam, welche Blöcke ihr dazu nutzen könnt (es gibt verschiedene Möglichkeiten). Schaut dann in der Blockliste nach.

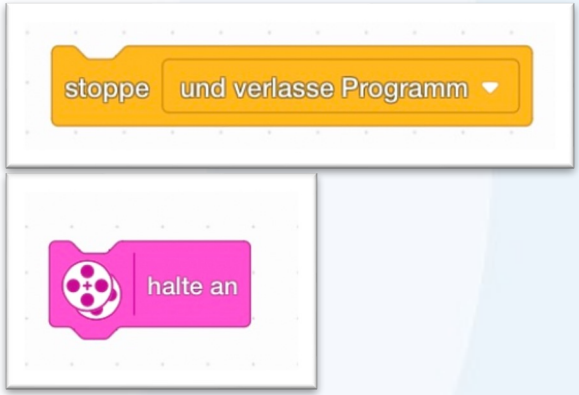
Tipp: Bei den Sensorblöcken stellt dieses Symbol den Ultraschallsensor dar: 

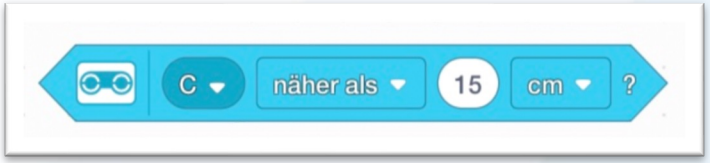
Zur Blockliste

Blockliste – Vor einem Hindernis anhalten

Ihr könnt folgende Blöcke verwenden. Stellt jeweils die entsprechenden Einstellungen ein und überlegt euch eine sinnvolle Reihenfolge.

Entscheidet, welchen dieser beiden Blöcke ihr nutzen wollt.





Start und Stopp bei Hindernissen

Euer Roboter soll nun weiter geradeaus fahren, sobald das Hindernis entfernt wurde.

Tipp: Nutzt dafür wieder den „wiederhole fortlaufend“-Block.

Besprecht untereinander, wie hier das Prinzip der Iteration genutzt wird.

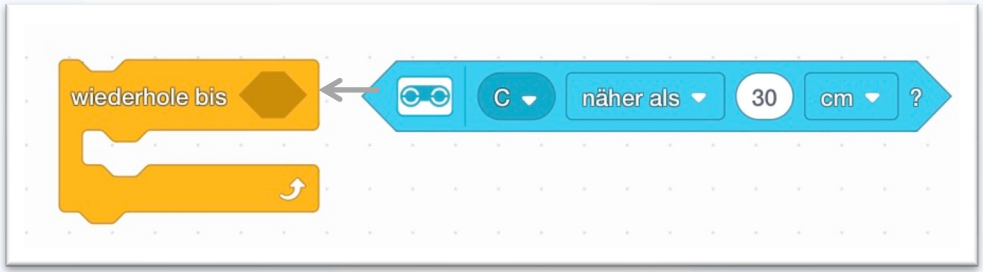
Iteration mit „wiederhole bis“-Block

Neben dem „wiederhole fortlaufend“-Block gibt es noch weitere Blöcke, die das Prinzip der Iteration nutzen. Einer davon ist der „wiederhole bis“-Block.

Dieser Block wird beendet, sobald die Bedingung nach dem „wiederhole bis“ erfüllt ist.

Versucht mithilfe des Blocks euren Roboter so zu programmieren, dass er geradeaus fährt, bis er auf ein Hindernis in 30 cm Abstand trifft und dann stehen bleibt.

Benutzt dafür folgende Blöcke:

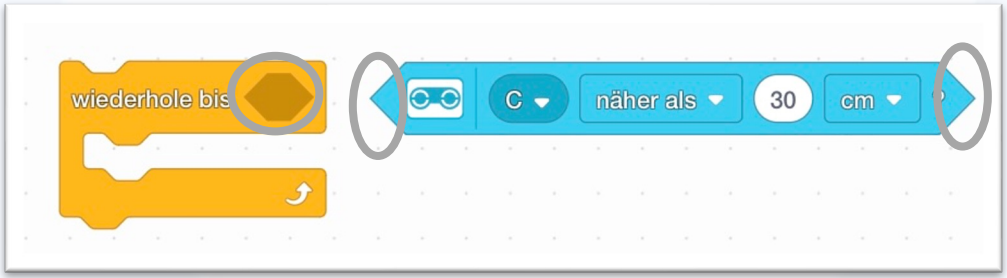


Info: Bedingungen

Ihr habt gerade zum ersten mal eine Bedingung genutzt. Eine solche Bedingung kann entweder erfüllt sein (wenn sie zutrifft) oder nicht erfüllt sein (wenn sie nicht zutrifft).

Bei einem „wiederhole bis“-Block wird vor jedem Durchlauf geprüft, ob die eingesetzte Bedingung erfüllt ist.

Ist die Bedingung erfüllt, dann wird dieser Block nicht mehr ausgeführt.

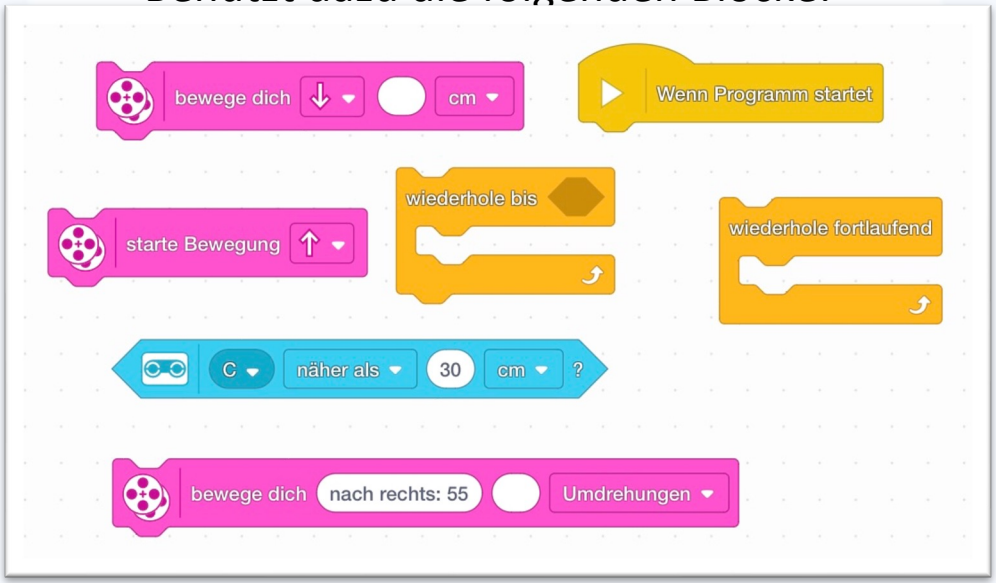


Erinnerung: Eine Bedingung könnt ihr an der spitzen Form erkennen. Diese signalisiert euch, dass ihr sie in eine sechseckige Aussparung einsetzen könnt.

Ein Hindernis umfahren

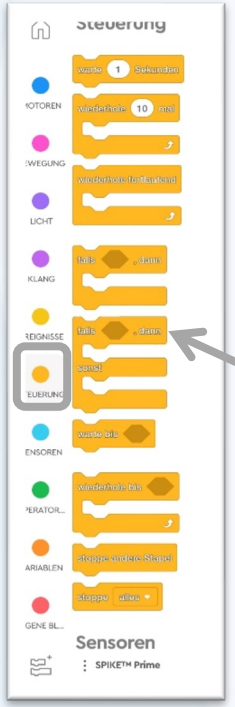
Erweitert eure Programmierung aus der vorherigen Aufgabe so, dass der Roboter so lang geradeaus fährt, bis er auf ein Hindernis trifft. Daraufhin soll er ein kurzes Stück rückwärts und dann eine Rechtskurve fahren.
Dieses Ausweichen soll immer wieder wiederholt werden.

Benutzt dazu die folgenden Blöcke:



Fallunterscheidung - „falls, dann, sonst“

Die letzte Aufgabe habt ihr mithilfe des „wiederhole bis“-Blocks und dem Prinzip der Iteration gelöst. Es gibt jedoch noch eine andere Möglichkeit, diese Programmieraufgabe umzusetzen.



Mit dem „falls, dann, sonst“-Block könnt ihr den Roboter so programmieren, dass er zwei Fälle unterscheidet:

Falls die Distanz kleiner als 30 cm ist, dann soll der Roboter rückwärts und eine Kurve fahren (1. Fall),
sonst soll er geradeaus fahren (2. Fall).

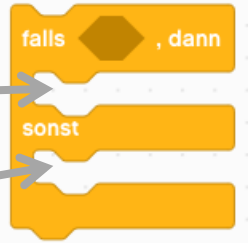
Nutzt den „falls, dann, sonst“-Block, um diese Aufgabe anders zu Programmieren.

Hinweis: Damit der Roboter immer wieder prüft, welcher Fall vorliegt, müsst ihr einen „wiederhole fortlaufend“-Block nutzen.

Info: Fallunterscheidung

Bei dem „falls, dann, sonst“-Block wird eine Fallunterscheidung unter bestimmten Bedingungen vorgenommen.

Der erste Zweig wird genau dann ausgeführt, wenn die Bedingung erfüllt wurde.
Der zweite Zweig wird dann ausgeführt, wenn die Bedingung nicht erfüllt wurde.



Der „falls, dann, sonst“-Block wird auch als Schalter bezeichnet.

Tipp: Es kann helfen, „falls, dann, sonst“-Sätze zu formulieren, die ihr dann im Programm umsetzt.

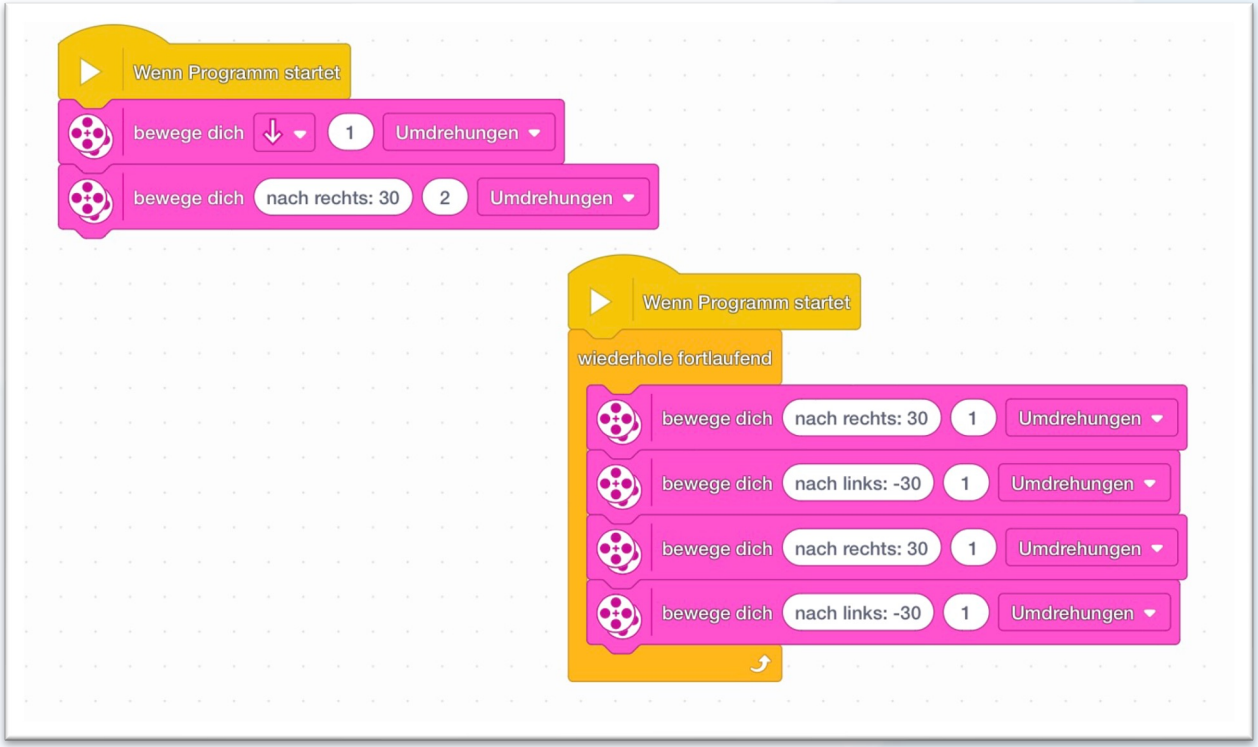
Hier könnt ihr nochmal nachlesen, was eine Bedingung ist.

?

Info: Stapel

Jedes Programm besteht aus mindestens einem Stapel. Ein Stapel besteht aus zusammenhängenden Blöcken, die alle irgendwann nach dem ersten Block von oben nach unten ausgeführt werden.

Beispiel für zwei verschiedene Stapel:

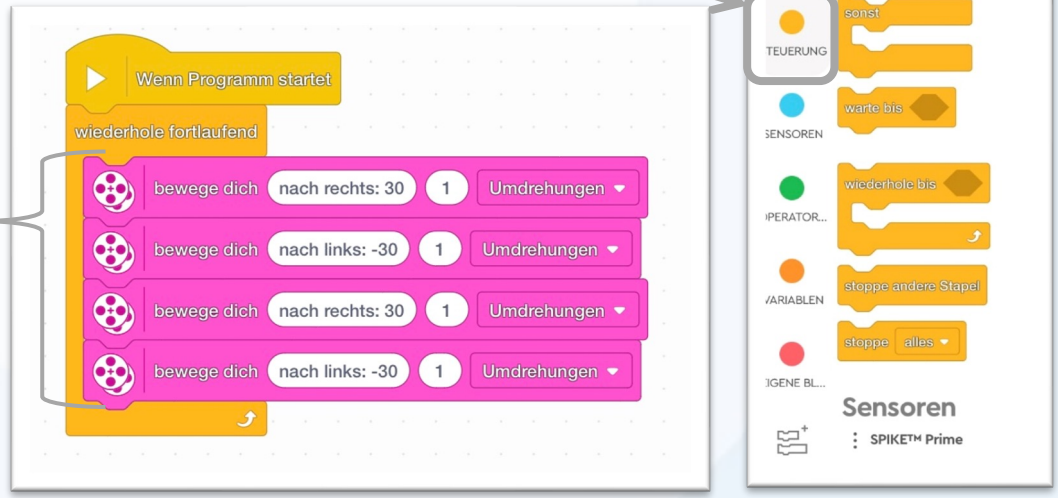


Info: Steuerungsblöcke und Unterstapel

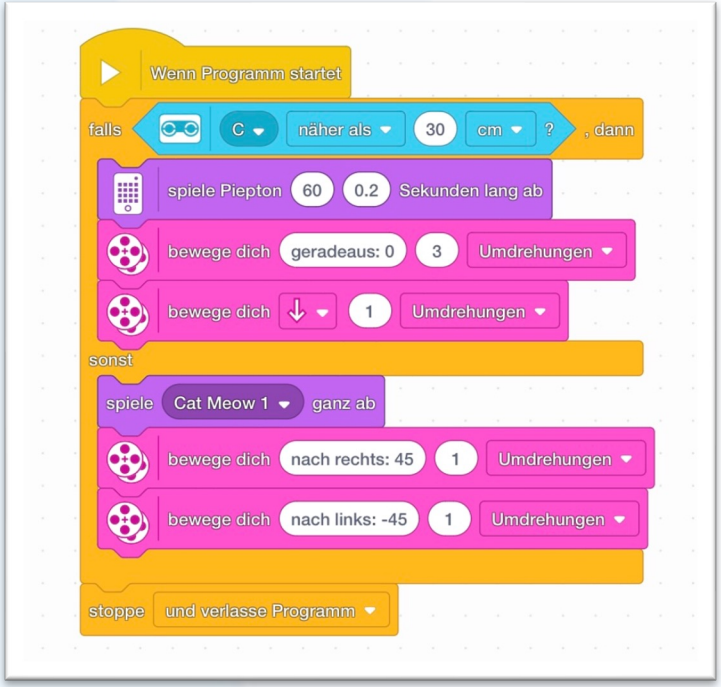
Die eben verwendeten Blöcke zur Iteration und Fallunterscheidung werden zusammengefasst als Steuerungsblöcke bezeichnet.

Es gibt Steuerungsblöcke, die unter bestimmten Bedingungen ausgeführt werden.

Dieser Abschnitt wird als Unterstapel bezeichnet. Ein Unterstapel ist ein Stapel, der von einem Steuerungsblock eingefasst wird.



Programmierung Fallunterscheidung

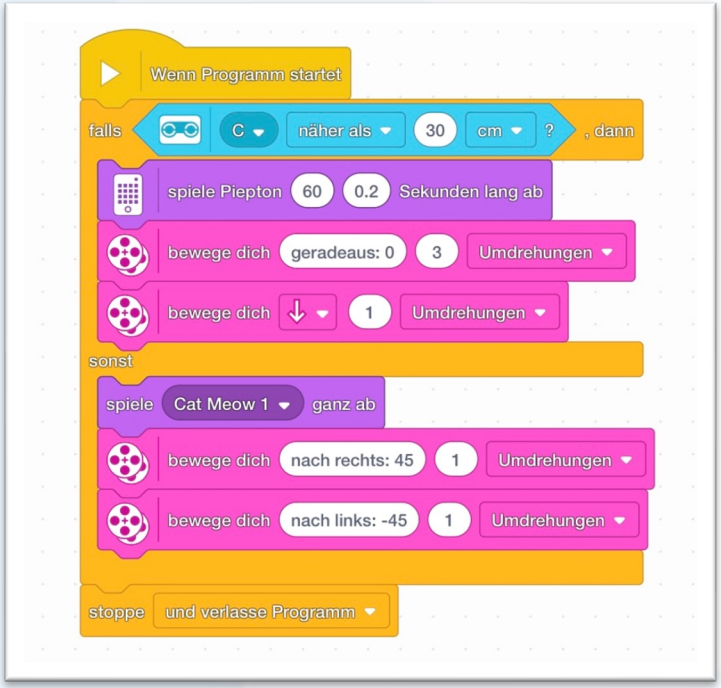


Sucht in der Programmierung links die Unterstapel.
Was passiert bei der Programmierung?

Beschreibt, was der Roboter bei der Ausführung des gesamten Stapels tun würde.
Wann wird welcher Unterstapel ausgeführt?
Versucht, mithilfe eines „falls, dann, sonst“-Satzes zur Lösung zu kommen.

Wenn ihr euch nicht sicher seid, dann probiert es aus.

Programmierung Fallunterscheidung - Lösung



Falls der Ultraschallsensor einen Wert kleiner als 30 cm misst, dann gibt er einen Piepton aus, fährt dann drei Umdrehungen geradeaus und eine rückwärts. Danach endet das Programm.

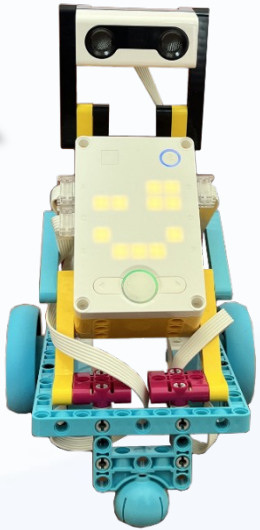
Für den Fall das der Ultraschallsensor einen Wert misst, der größer als 30 cm beträgt, dann hört man das Miau einer Katze, der Roboter fährt eine Rechts- und anschließend eine Linkskurve. Dann endet das Programm.

Fallunterscheidung - Abstandserkennung

Nutzt den „falls, dann, sonst“-Block, um euren Roboter so zu programmieren, dass dieser auf der Stelle steht und immer dann zu piepen beginnt, wenn ein Hindernis weniger als 40 cm entfernt ist.

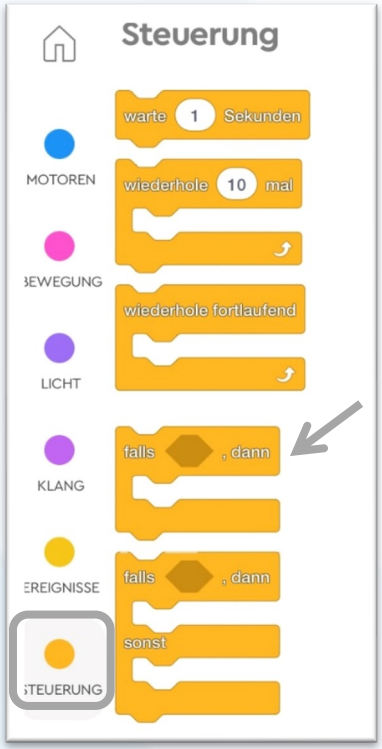
Hinweis: Den unteren Zweig könnt ihr leer lassen, da der Roboter in diesem Fall nichts macht.

Denkt daran, einen „falls, dann, sonst“-Satz zu formulieren.



Fallunterscheidung - „falls, dann“

In der letzten Aufgabe habt ihr den unteren Zweig leer gelassen. Es gibt einen anderen Block, der geschickter für diese Programmierung gewählt werden kann.



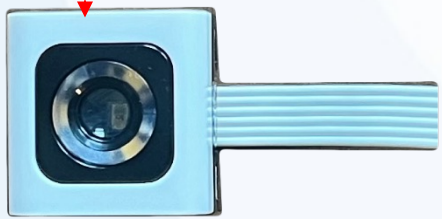
Ersetzt den „falls, dann, sonst“- Block in der Programmierung durch den „falls, dann“-Block.

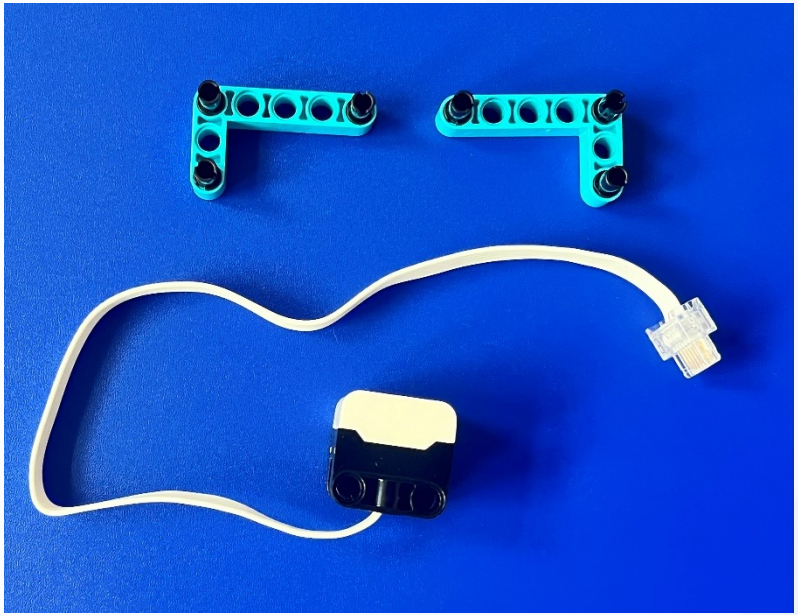
Als Hilfe könnt ihr einen „falls, dann“-Satz formulieren.

Probiert aus, ob euer Roboter die gleiche Aufgabe ausführt?

Farbsensor anbauen

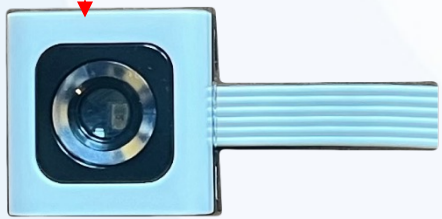
Ihr könnt jetzt den Farbsensor als weiteren Sensor nutzen.

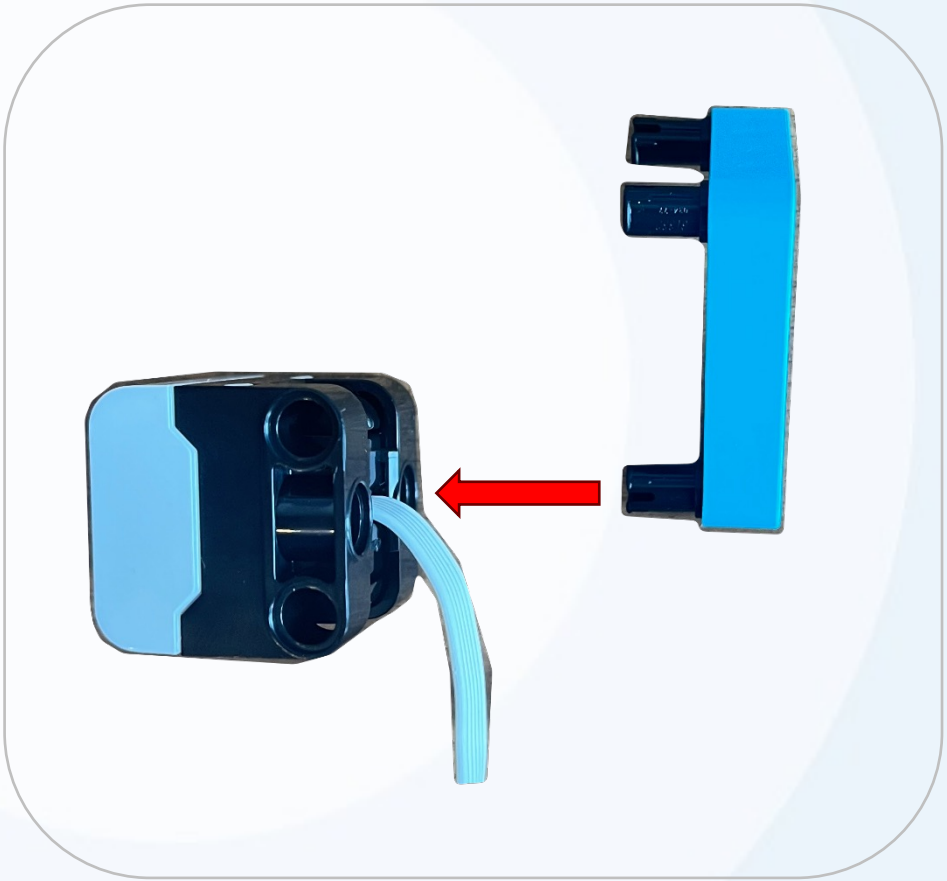




Farbsensor anbauen

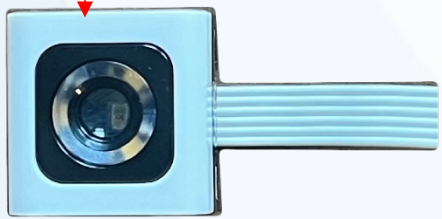
Ihr könnt jetzt den Farbsensor als weiteren Sensor nutzen.





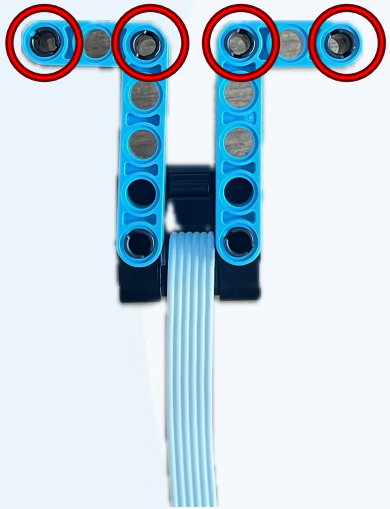
Farbsensor anbauen

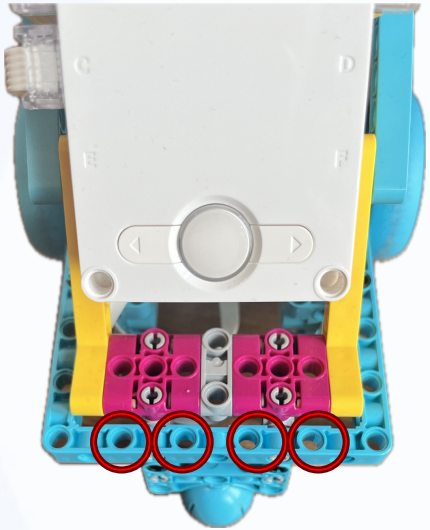
Ihr könnt jetzt den Farbsensor als weiteren Sensor nutzen.

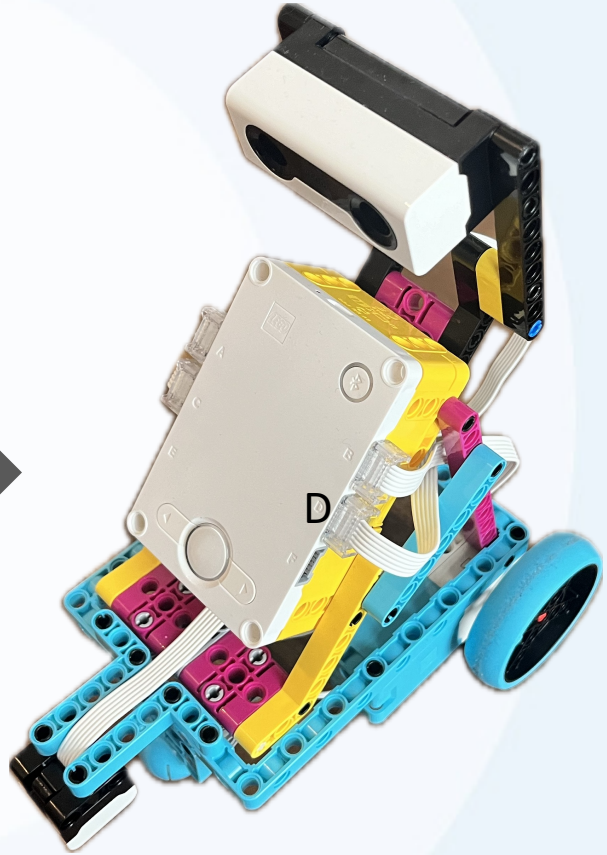


Auf der anderen Seite auch. Dann sollte es so aussehen:

Farbsensor anbauen



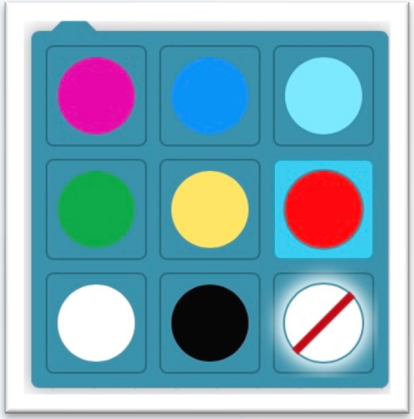


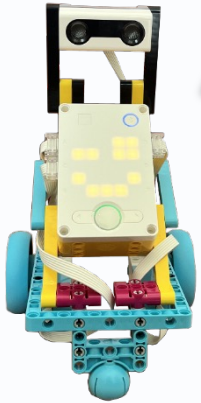


Steckt die Halterung vom Farbsensor an den markierten Stellen mit dem Spike zusammen. Schließt dann noch das Kabel an Port D an.

Farbsensor

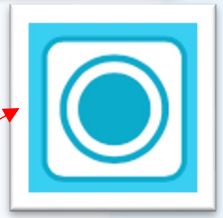
Der Farbsensor kann die unten angezeigten sieben Farben unterscheiden.
Bearbeitet <u>mindestens eine</u> der vier untenstehenden Aufgaben zum Farbsensor.





Achtung!
Um Farben gut erkennen zu können, sollte euer Roboter höchstens mit einer Geschwindigkeit von 15 % fahren.

Hinweis: So sieht das Symbol des Farbsensors aus



Farbe als Signal verwenden (z. B. wie bei einer Ampel)

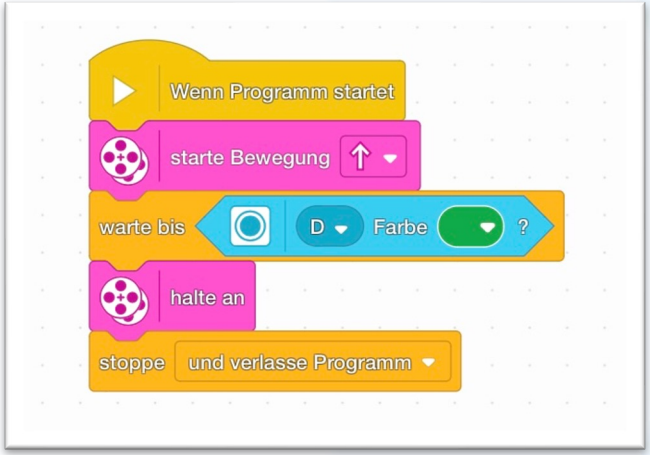
Richtungsänderung durch Farbe

Farben als Signal verwenden

Weitere Farbsensoraufgabe

Euer Roboter kann mittels Farbsensor auf unterschiedliche Farben mit einem bestimmten Verhalten reagieren.

Überlegt euch, was euer Roboter machen würde, wenn ihr ihn mit dem abgebildeten Programm über Farbkarten aus eurer Kiste fahren lassen würdet.

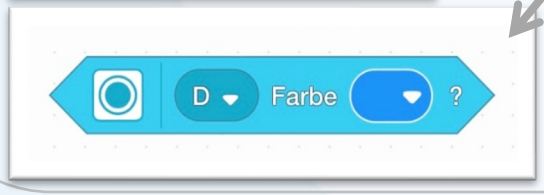


Nutzt die Farbkarten wie eine Ampel. Programmiert euren Roboter so, dass er bei einer grünen Ampel weiterfährt und danach an einer roten Ampel stoppt.
Hinweis: Achtet darauf, dass der Roboter nicht zu schnell fährt.

Nach Farben sortieren

Weitere Farbsensoraufgabe

Euer Roboter kann Gegenstände in acht unterschiedlichen Farben unterscheiden.



Programmiert ihn so, dass er für jede Farbe der Farbkarte , die ihr ihm hinhaltet, einen anderen Ton ausgibt.
Verwendet dazu mehrere „falls“- Blöcke (jeweils für eine Farbe).

Damit die Farben immer wieder benannt werden, muss alles wiederholt werden.

In jedem „falls“-Block müsst ihr die richtige Bedingung einstellen und den passenden Soundeffekt zuweisen.

Hinweis: Achtet darauf, dass der Roboter nicht zu schnell fährt.

Richtungsänderung durch Farbe

Weitere Farbsensoraufgabe

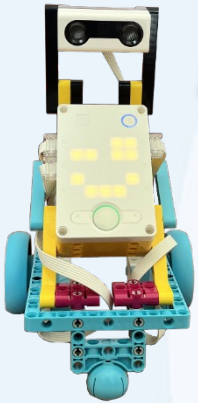
Euer Roboter kann mittels Farbsensor auf unterschiedliche Farben mit einem bestimmten Verhalten reagieren.

Programmiert euren Roboter so, dass er bei blauem Untergrund eine Rechtskurve fährt, sonst aber geradeaus.
Tipp: Nutzt für diese Aufgabe den “falls, dann, sonst“-Block und diesen Sensorblock:

An seiner spitzen Form könnt ihr erkennen, dass es sich um eine Bedingung handelt.

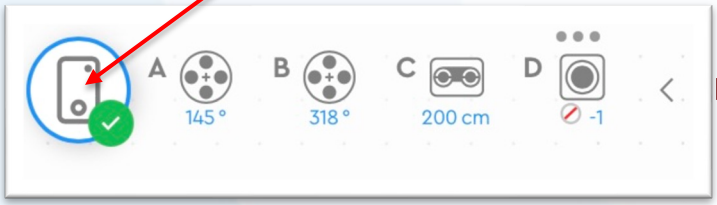
Hinweis: Achtet darauf, dass der Roboter nicht zu schnell fährt.

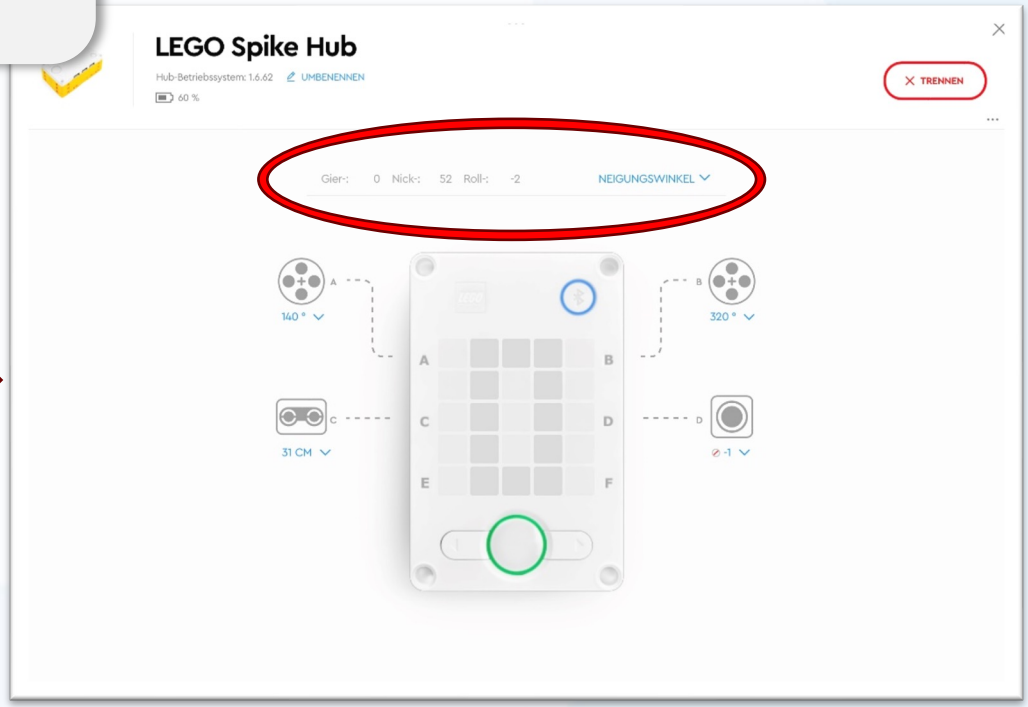
Kreiselsensor und Operatoren



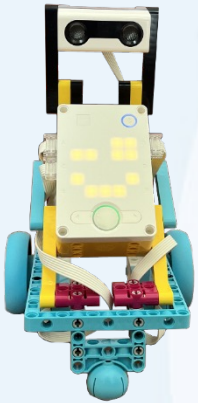
Der Kreiselsensor ist direkt im Hub verbaut. Klickt auf das Hub-Symbol in der Statusleiste, um zur Info-Seite des Hubs zu gelangen.

Hier seht ihr die verschiedenen Neigungswinkel des Hubs. Kippt das Hub in unterschiedliche Richtung und achtet dabei auf die Winkelanzeigen





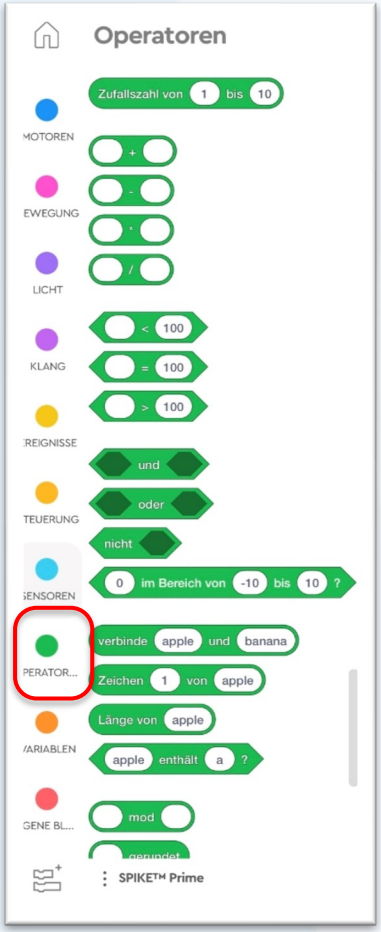
Kreiselsensor und Operatoren



Das Symbol für den Kreiselsensor sieht übrigens so aus:

Ihr könnt den Winkel mit dem abgerundeten Block auslesen. Stellt den “Gier-Winkel” ein, denn dieser beschreibt die Links-Rechts-Drehung des Hubs.

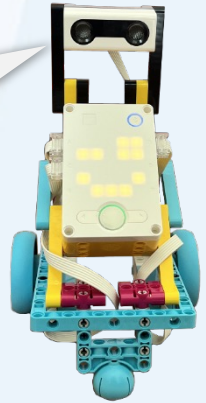
Kreiselsensor und Operatoren



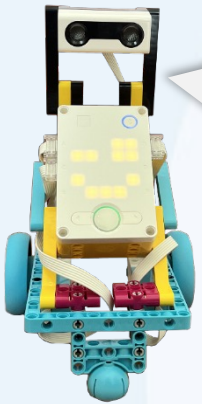
Nun benötigt ihr noch einen Bedinungsblock (zur Erinnerung: das sind die mit den spitzen Seiten). Benutzt zunächst den ”=” –Block:

Setzt dort den Gier-Winkel Block ein und stellt den Wert auf 90 ein.

Versucht nun die Aufgabe „Quadrat fahren“ von vorhin erneut zu lösen, mit Hilfe des Kreiselsensors, sodass jede Kurve ein rechter Winkel ist.



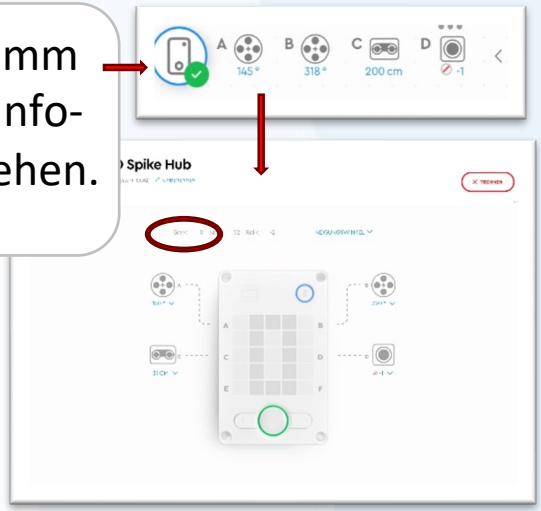
Kreiselsensor und Operatoren: Quadrat fahren



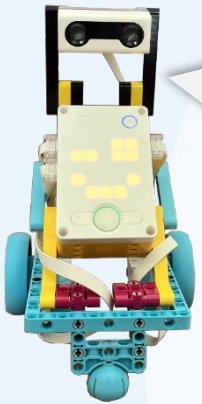
Hier könnt ihr euch Tipps für die benötigten Blöcke anschauen:

?

Während Spike euer Programm ausführt, könnt ihr auf der Info-Seite live den Sensorwert sehen.



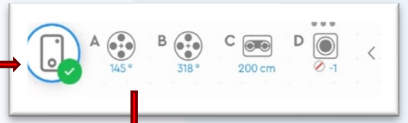
Kreiselsensor und Operatoren: Quadrat fahren

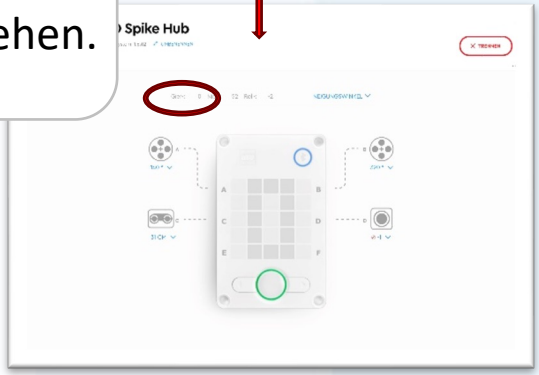


Hier könnt ihr euch Tipps für die benötigten Blöcke anschauen:

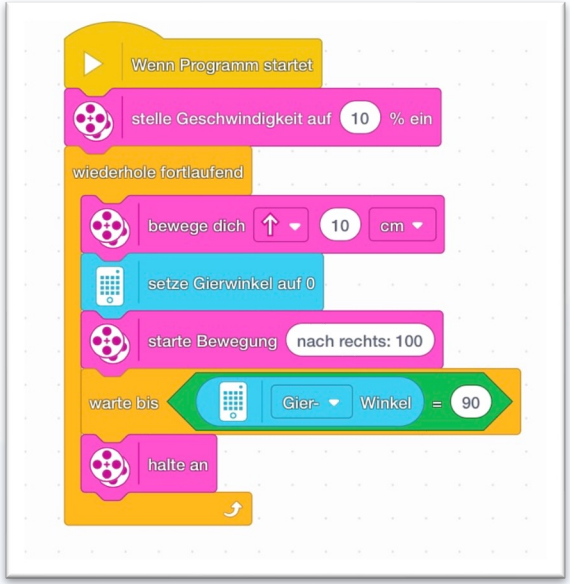
?

Während Spike euer Programm ausführt, könnt ihr auf der Info-Seite live den Sensorwert sehen.





Eine mögliche Lösung könnte so aussehen:

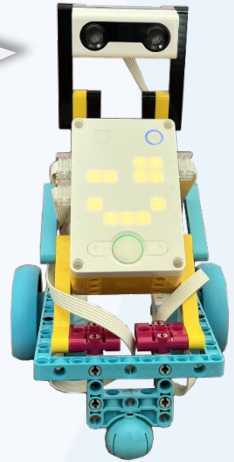


Diskutiert auch, warum ihr den Block “setze Gierwinkel auf 0” benötigt.

Info: Nebenläufigkeit

Ihr habt bereits erfahren, was Stapel sind. Der Roboter kann mehrere dieser Stapel gleichzeitig ausführen, was als Nebenläufigkeit bezeichnet wird.

Beispiel: Der Roboter fährt ein Quadrat und zeigt dabei gleichzeitig verschiedene Bilder auf der LED-Matrian.



Hier könnt ihr nochmal nachlesen, was ein Stapel ist.

?

Zwei Stapel

Programmiert zwei Stapel. Ein Stapel soll aus verschiedenen Bewegungsblöcken bestehen, der andere Stapel aus Soundeffektblöcken. Trefft dazu eine eigene Auswahl an Blöcken!

Tipp: Jeder Stapel sollte mit diesem Block beginnen. So startet der jeweilige Stapel bei Programmstart.

Könnt ihr erklären, wie das zum Prinzip der Nebenläufigkeit passt? Falls ihr euch unsicher seid, schaut nochmal auf die vorherige Folie.

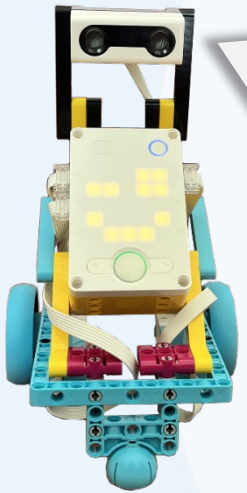
Doppelte Bewegungen

Passt nun das vorherige Programm an und entfernt die Soundeffekte im zweiten Stapel.
Benutzt stattdessen auch im zweiten Stapel beliebige Bewegungsblöcke.
Überlegt gemeinsam, bevor ihr das Programm startet, was passieren wird.

Diskutiert, warum die Programmierung so nicht funktioniert.

Auf der nächsten Folie findet ihr die Lösung.

Nebenläufigkeit und Konflikte



Wenn der Roboter gleichzeitig widersprüchliche Programmierungen ausführen soll, funktioniert das nicht. Die Nebenläufigkeit kann nur dann verwendet werden, wenn keine Konflikte zwischen den Stapeln bestehen.

Ein solcher Konflikt tritt zum Beispiel auf, wenn ein Stapel den Roboter nach links fahren lässt, und ein anderer Stapel den Roboter nach rechts fahren lässt.

Patrouille II

Ihr habt den Roboter bereits einmal so programmiert, dass dieser eine gerade Strecke auf- und abgefahren ist und vor der Drehung gepiept hat. Dieses Programm sollt ihr jetzt abändern.

Programmiert den Roboter nun so, dass er eine solche Strecke auf- und abfährt, aber ohne vor der Drehung zu piepen.

Der Roboter soll währenddessen mithilfe der Nebenläufigkeit einen Ton ausgeben, wenn sich etwas weniger als 30 cm vor ihm befindet.

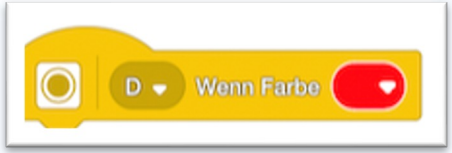
Info: Ereignisse

Bisher begann ein Stapel immer mit dem Stein „wenn das Programm startet“. Ihr habt dazu das Programm gestartet und das war das Ereignis, dass zum Ablauf des Stapels geführt hat.

Neben dem Programmstart gibt es auch andere Ereignisse, die einen Stapel ablaufen lassen können. Jedes Mal, wenn dieses Ereignis eintritt, dann wird der Stapel erneut ausgeführt.

Dieses Prinzip bezeichnet man als Ereignishandling und die zugehörigen Blöcke als Ereignisblöcke.

Beispiel: Der Ereignisblock „Wenn Farbe“ startet einen Stapel, wenn das Ereignis: *der Farbsensor gibt die Farbe blau aus* eintritt.



Abstandsereignis

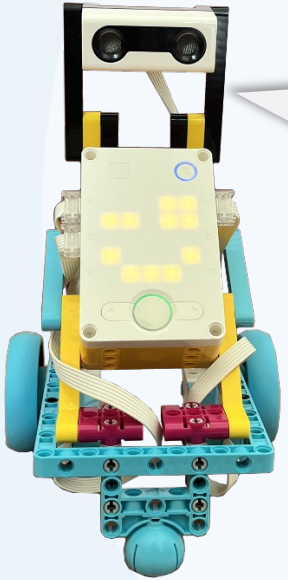
Programmiert einen Stapel, der mit dem „wenn näher als“-Block aus den Ereignisblöcken mit den passenden Einstellungen beginnt:

Euer Roboter soll ausgelöst durch dieses Ereignis geradeaus fahren.

Startet das Programm. Überlegt euch: Was passiert, wenn ihr die Hand vor den Abstandssensor haltet?

Was ist das Ereignis, das den Stapel startet?

Abstandsereignis - Lösung



Wenn ihr eine Hand dicht vor den Abstandssensor haltet, beginnt der Roboter geradeaus zu fahren.

Das auslösende Ereignis ist der ausreichend niedrige Abstand zum Sensor.

Farberkennung mit Ereignissen

Programmiert einen Stapel, bei dem der Roboter die vom Farbsensor ermittelte Farbe mithilfe des passenden Soundeffekts ausgibt. Nutzt dafür die „bei Farbe“-Blöcke aus den Ereignisblöcken.

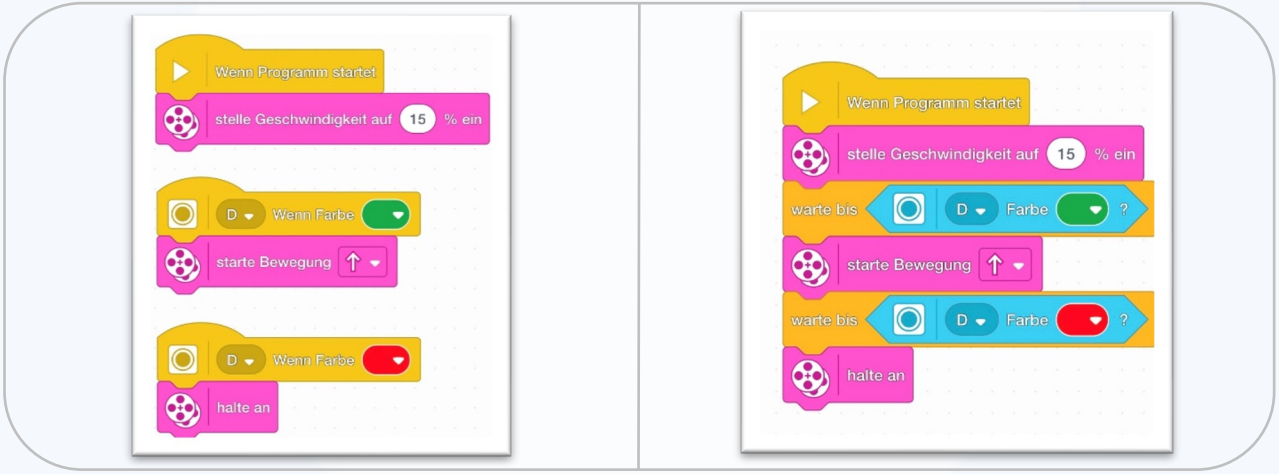
Nutzt die Nebenläufigkeit, um den Roboter zusätzlich geradeaus fahren zu lassen.

Erinnerung: Stellt die Geschwindigkeit des Roboters auf maximal 15 % ein, da dieser die Farben sonst nicht gut erkennen kann!

Hinweis: Ihr könnt auch eigene Soundeffekts aufzeichnen, indem unter „Soundeffekt hinzufügen“ auf „Aufzeichnen“ klickt. Damit könnt ihr die benötigten Farbnamen einsprechen.

Über eine Ampel fahren

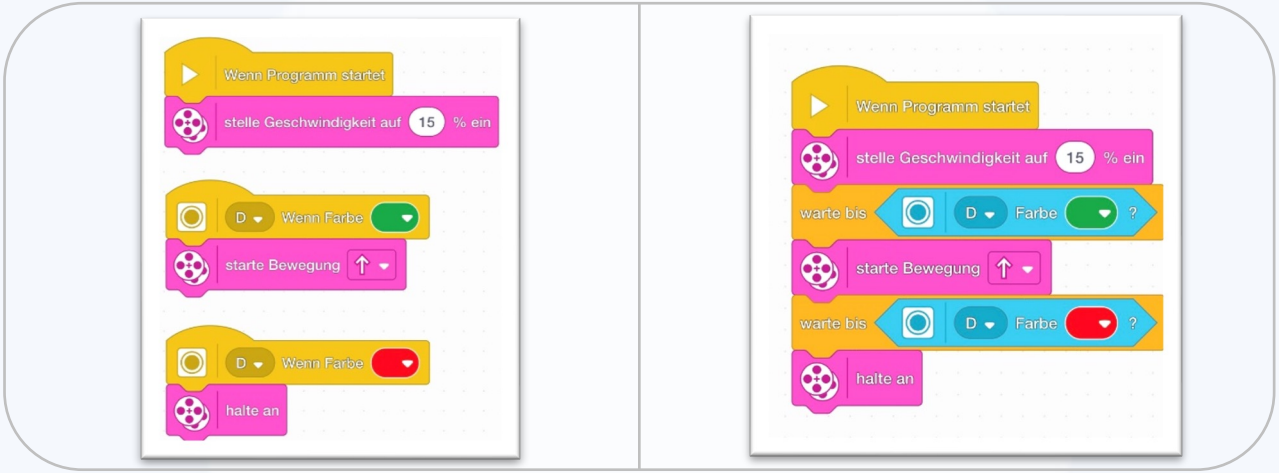
Die linke und rechte Programmierung sind beide gedacht, den Roboter (weiter-)fahren zu lassen, wenn der Farbsensor die Farbe grün ausgibt. Außerdem würde der Roboter stoppen, wenn der Farbsensor die Farbe rot ausgibt.
Überlegt gemeinsam: Welche Variante würde im Straßenverkehr mehr Sinn machen?



Hinweis: Ein Stapel wird immer dann gestartet, wenn das entsprechende Ereignis ausgelöst wird.

Über eine Ampel fahren – geschickte Lösung

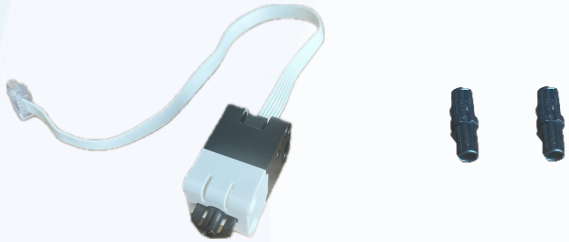
Auch in diesem Fall gibt es wieder eine Programmierung, die die geschicktere Wahl ist.



Die linke Programmierung wird bei jeder grünen und roten Ampel erneut gestartet.
Die rechte Programmierung startet nur bei Beginn des Programms. Außerdem müsste erst eine grüne und dann eine rote Ampel aufeinander folgen. Andersherum würde es nicht funktionieren.
Somit ist die linke Programmierung mit der Nutzung des Prinzips der Ereignisblöcke geschickter.

Berührungssensor anbauen

Der Berührungssensor reagiert auf Druck. Wird er gedrückt, kann ein Ereignis ausgelöst werden.



Baut den Berührungs-sensor hinten an den Roboter an.

Steckt das Kabel in **Anschluss E**.

Berührungssensor als Start-Taste verwenden

Überlegt euch ein kurzes Programm, dass ihr mit dem Berührungssensor als Start-Taste auslösen möchtet. Nutzt dazu das Prinzip des Ereignishandlings.

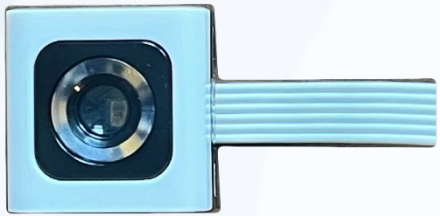
Tipp: Dieses Symbol stellt den Berührungssensor dar: 

Abstandserkennung

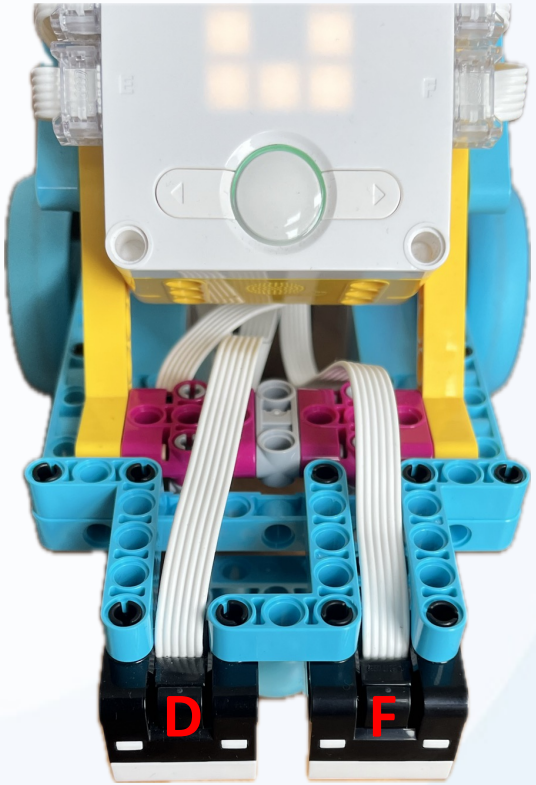
Programmiert den Roboter so, dass er geradeaus fährt. Nutzt das Ereignishandling, sodass der Roboter anhält, sobald der Abstand geringer als 30 cm ist. Danach soll sich Spike um 180 Grad drehen und ganz langsam rückwärts fahren bis der Berührungssensor aus löst. Nutzt auch hierfür das Ereignishandling.

Entlang der Linie fahren

Dazu benötigt ihr einen zweiten Farbsensor:

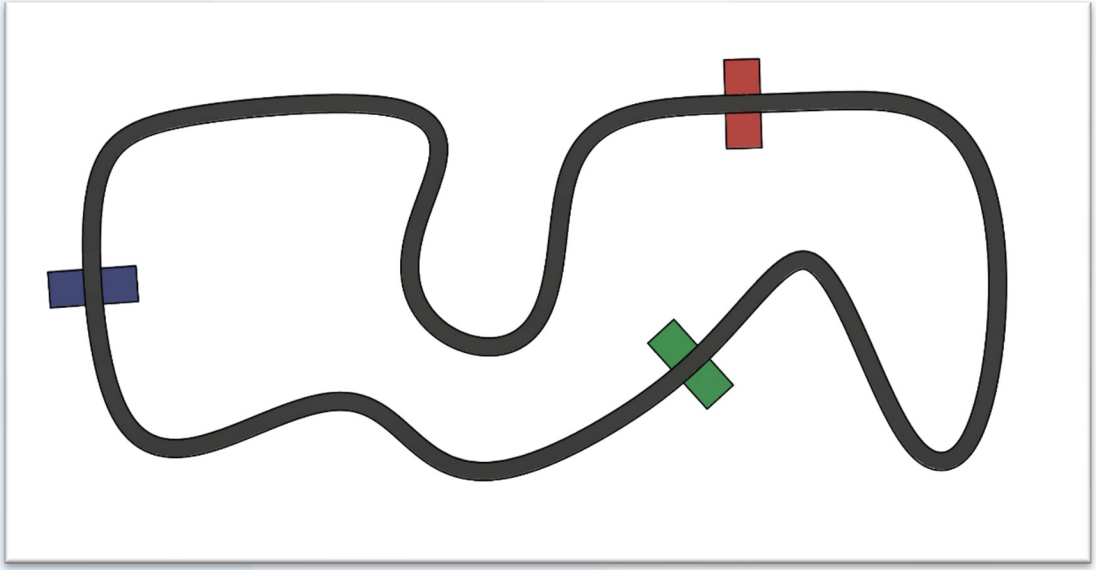


Versetzt den ersten Farbsensor auf die rechte Seite des Roboters (auf dem Bild hier links). Baut den zweiten Sensor wie rechts zu sehen ist daneben an und verbindet ihn mit Port F.



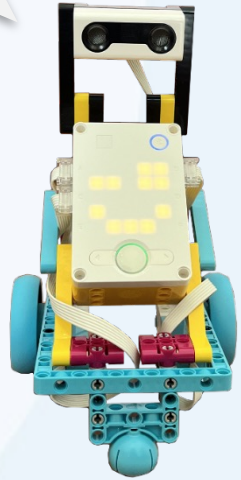
Entlang der Linie fahren

Überlegt euch, wie euch die vorgestellten Prinzipien helfen können, um durch das entlang der Linie zu fahren.



Prinzipien:

- Iteration
- Fallunterscheidung
- Nebenläufigkeit
- Ereignishandling



Diese Bahn bekommt ihr von eurer Betreuerin / eurem Betreuer. Ignoriert zunächst die farbigen Kästchen und programmiert euren Roboter so, dass er der Linie folgt.

Entlang der Linie fahren

Nun da euer Spike der Linie folgen kann. Kommen die Farbfelder ins Spiel.

Fügt folgende Abläufe hinzu:

- Bei Blau:

  Der Spike spielt einen Sound euerer Wahl ab, solange bis er eine andere Farbe findet

- Bei Rot:

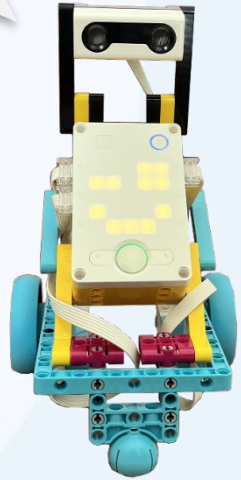
  Der Spike hält an und fährt erst weiter, wenn der Berührungssensor gedrückt wird

- Bei Grün:

  Der Spike dreht sich um 180° und fährt die Runde andersherum

Prinzipien:

- Iteration
- Fallunterscheidung
- Nebenläufigkeit
- Ereignishandling



Abschluss: Kreative Programmieraufgabe

Überlegt euch gemeinsam eine kreative Programmieraufgabe, die ihr dann auch umsetzen könnt. Was soll eurer Roboter gerne können?

Denkt dabei an die Prinzipien und Sensoren, die ihr kennengelernt habt. Zum Beispiel könntet ihr den dritten Motor aus dem Set verwenden, um einen Arm zu bauen oder den Kopf des Roboters zu drehen. Oder baut den Roboter um.

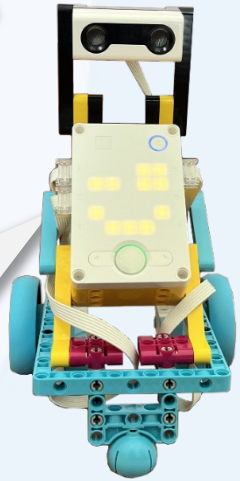
Notiert die Programmieraufgabe auf einem Zettel.

Sensoren:

- Ultraschallsensor
- Farbsensor
- Kreiselsensor
- Berührungssensor

Prinzipien:

- Iteration
- Fallunterscheidung
- Nebenläufigkeit
- Ereignishandling



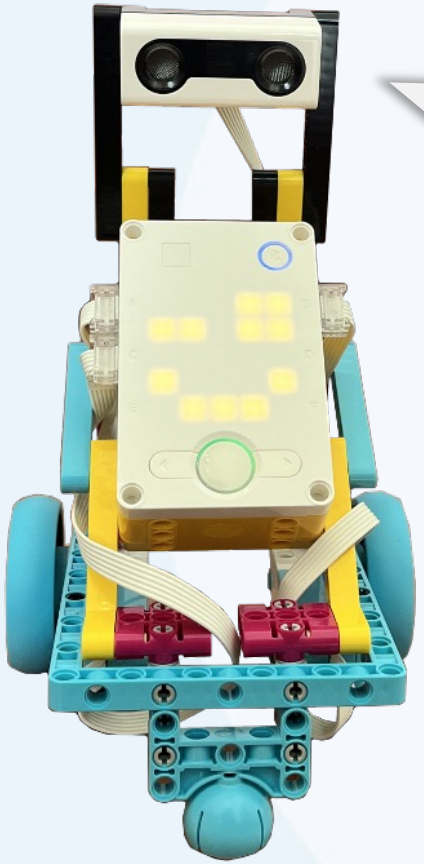
Hilfekarte - Unbegrenztes Fahren

unbegrenzt fahren

Vorherige Bewegung(en) werden gestoppt

Zurück

Hilfekarte - Vor einem Hindernis anhalten



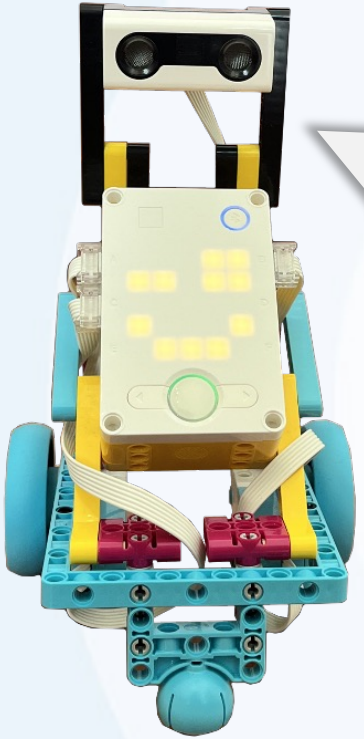
Sind alle Einstellungen korrekt eingestellt?

Stimmt eure Reihenfolge?

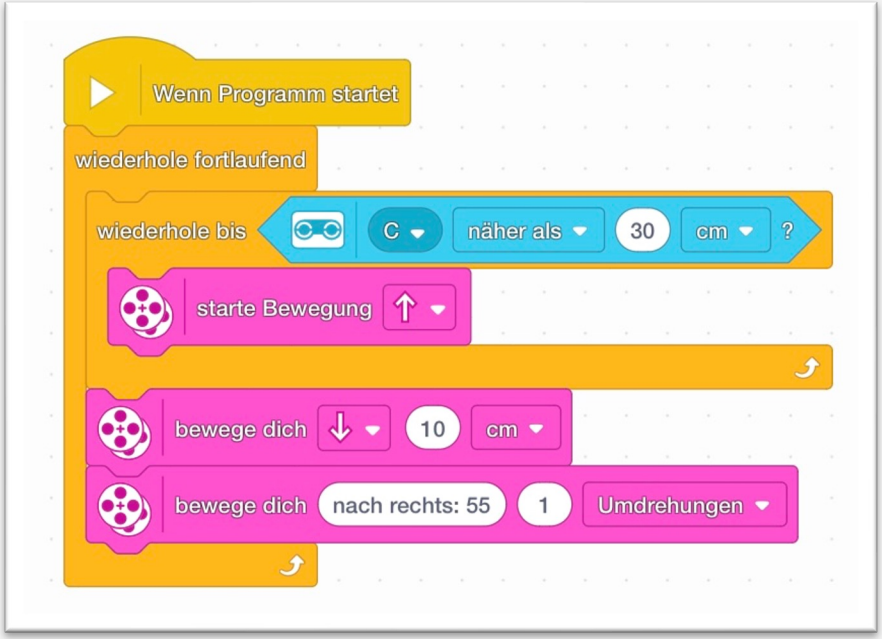
1) Startblock
    2) Unbegrenztes Fahren (geradeaus)
        3) Warten bis der Ultraschallsensor kleiner als 30 cm misst
            4) Bewegung anhalten
                5) Programm verlassen

Zurück

Hilfekarte - „wiederhole bis“

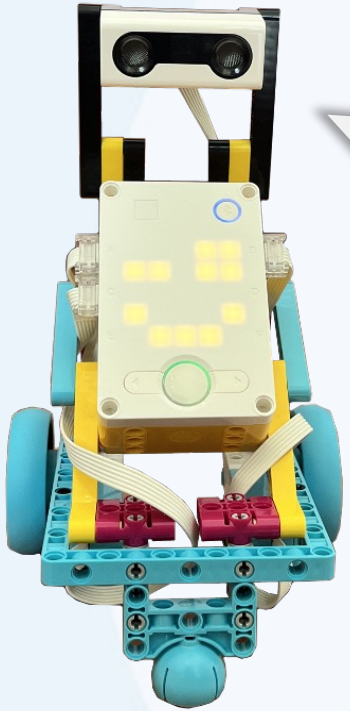


So könnte das fertige Programm zum dauerhaften Ausweichen eures Roboters aussehen:

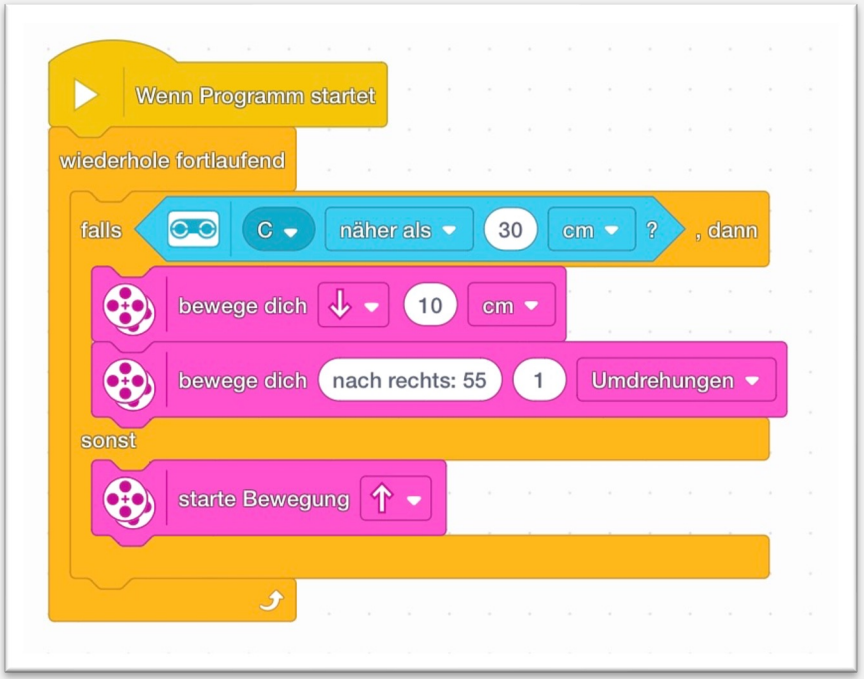


Zurück

Hilfekarte - „falls, dann, sonst“



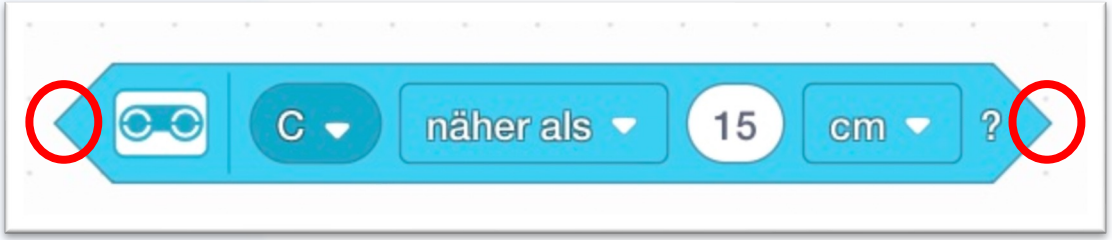
So könnte euer fertiges Programm aussehen:



Zurück

Hilfekarte - Info zur Fallunterscheidung

Eine Bedingung kann entweder erfüllt sein (wenn sie zutrifft) oder nicht erfüllt sein (wenn sie nicht zutrifft).
Dieser Zustand wird immer dann geprüft, wenn ein Block mit einer eingesetzten Bedingung ausgeführt werden soll.



Eine Bedingung könnt ihr an der spitzen Form erkennen. Dies signalisiert euch, dass ihr sie in eine sechseckige Aussparung einsetzen könnt.

Zurück

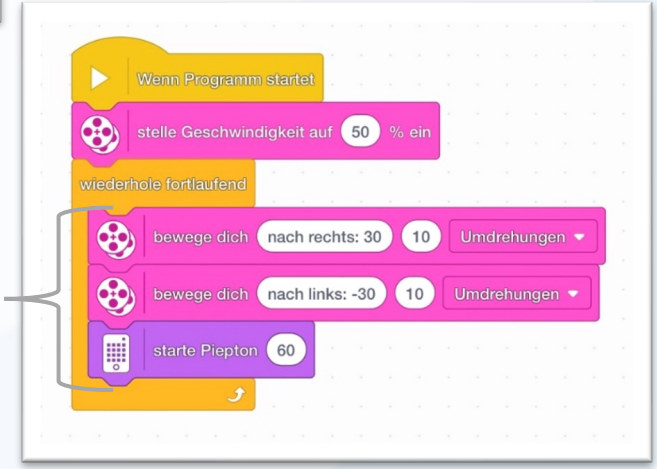
Hilfekarte - Info Stapel

Zusammenhängende Blöcke werden Stapel oder Unterstapel genannt. Zu einem (Unter-)Stapel gehören alle Blöcke, die irgendwann nach dem ersten Block ausgeführt werden.

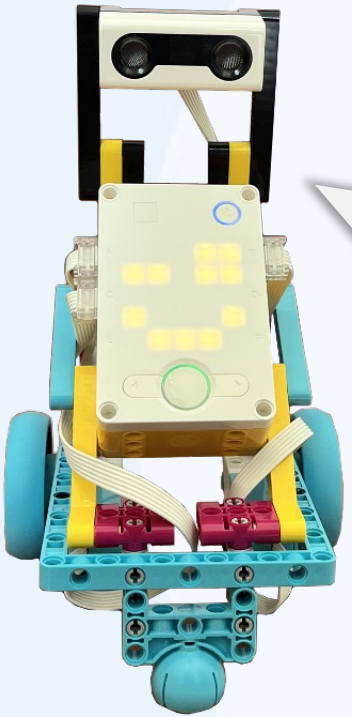
Beispiel für zwei verschiedene Stapel:

Zurück

Dieser Abschnitt wird als Unterstapel bezeichnet.



Hilfekarte - Prinzip der Iteration



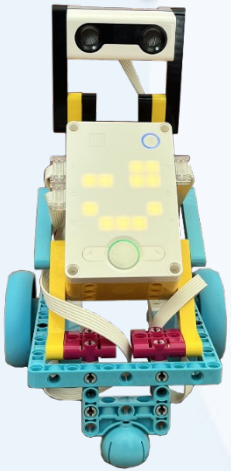
Da Wiederholungsblöcke alles wiederholen, könnt ihr Dopplungen in der Programmierung geschickt einsparen.

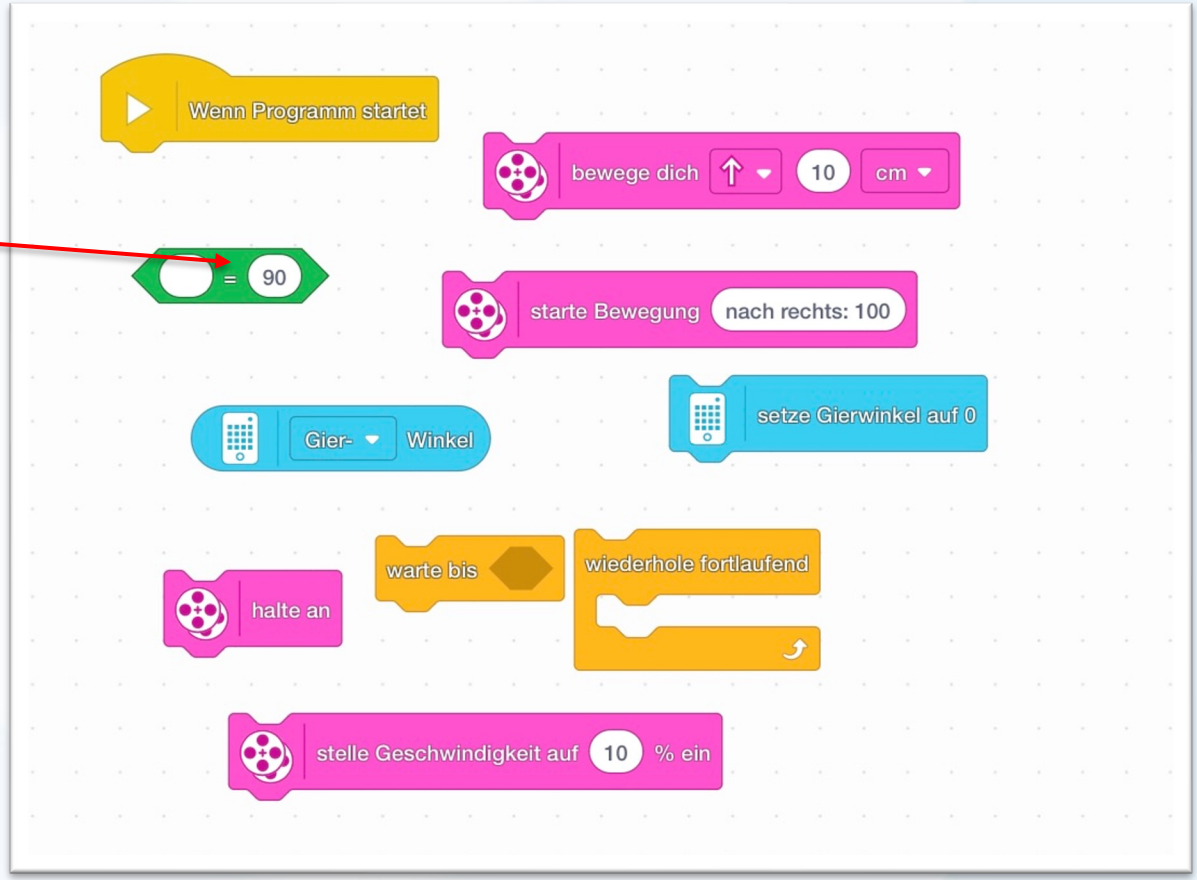
Die Nutzung von Wiederholungsblöcken, die bestimmte Programmierungen wiederholt ausführen, nennt man Iteration.

Zurück

Hilfekarte - Blöcke zum Lösen von “Quadrat fahren“ mit Kreiselsensor

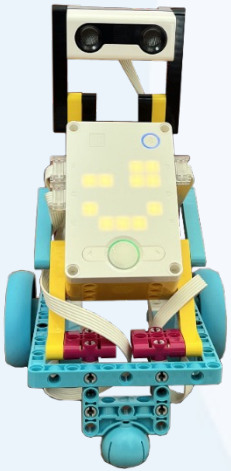
Für Drehungen links herum müsst ihr zusätzlich die Winkelmessung auf ”-90” stellen.

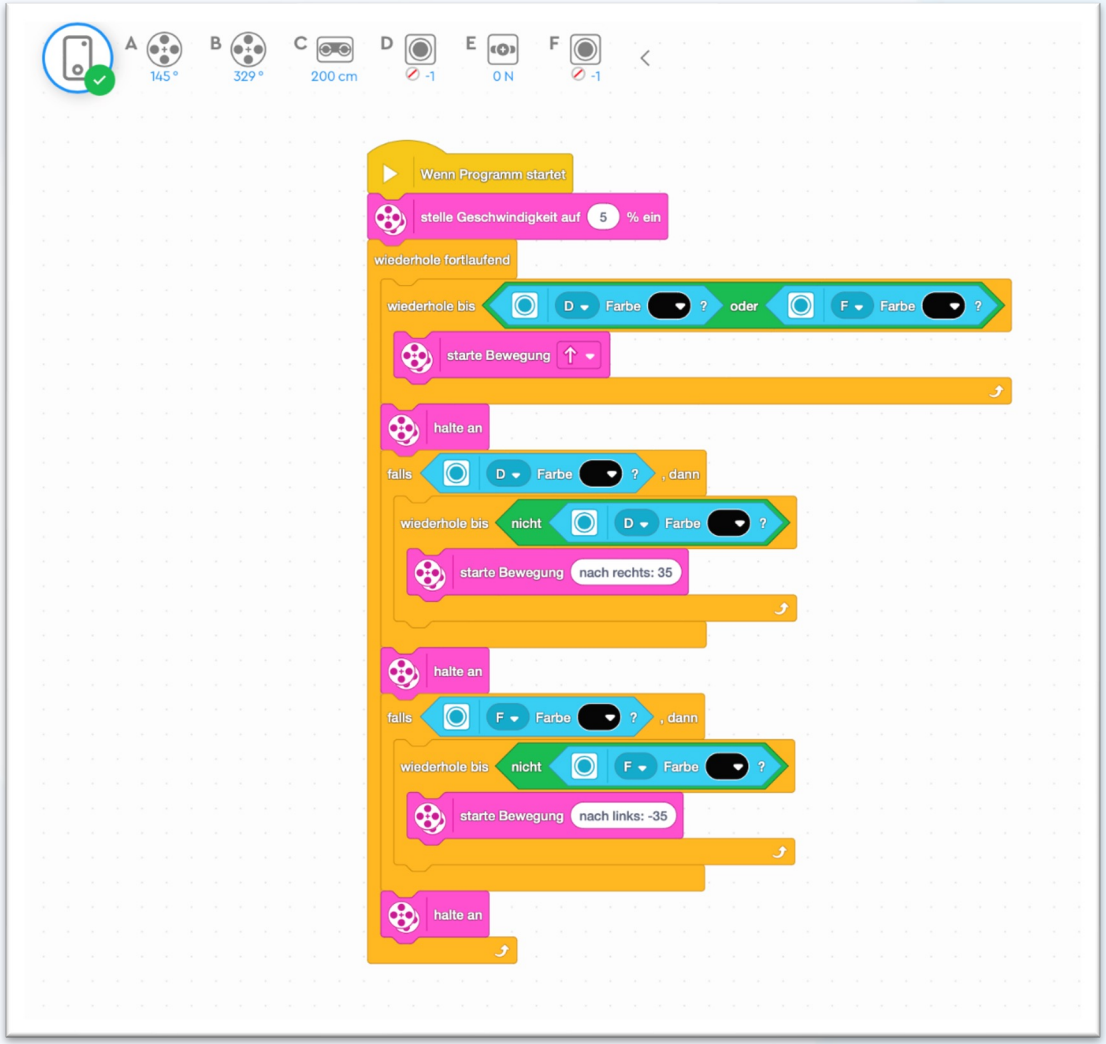




Zurück

Hilfekarte - Beispiel Lösung von „Der Linie folgen“





Zurück