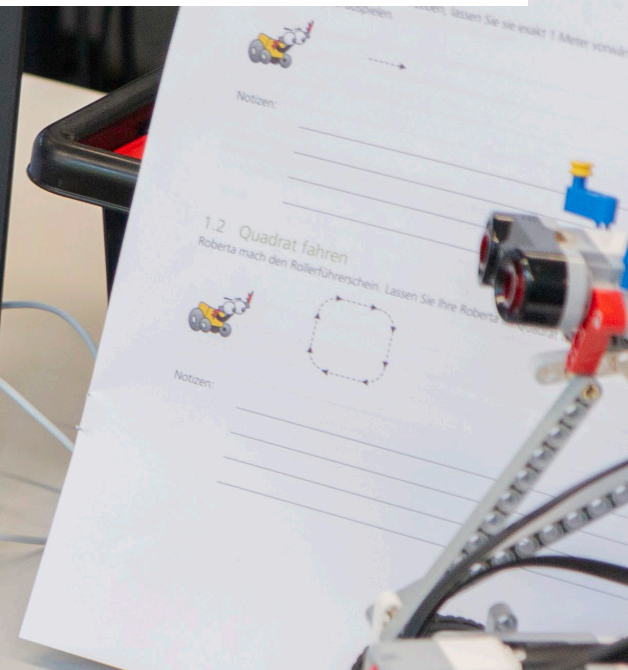
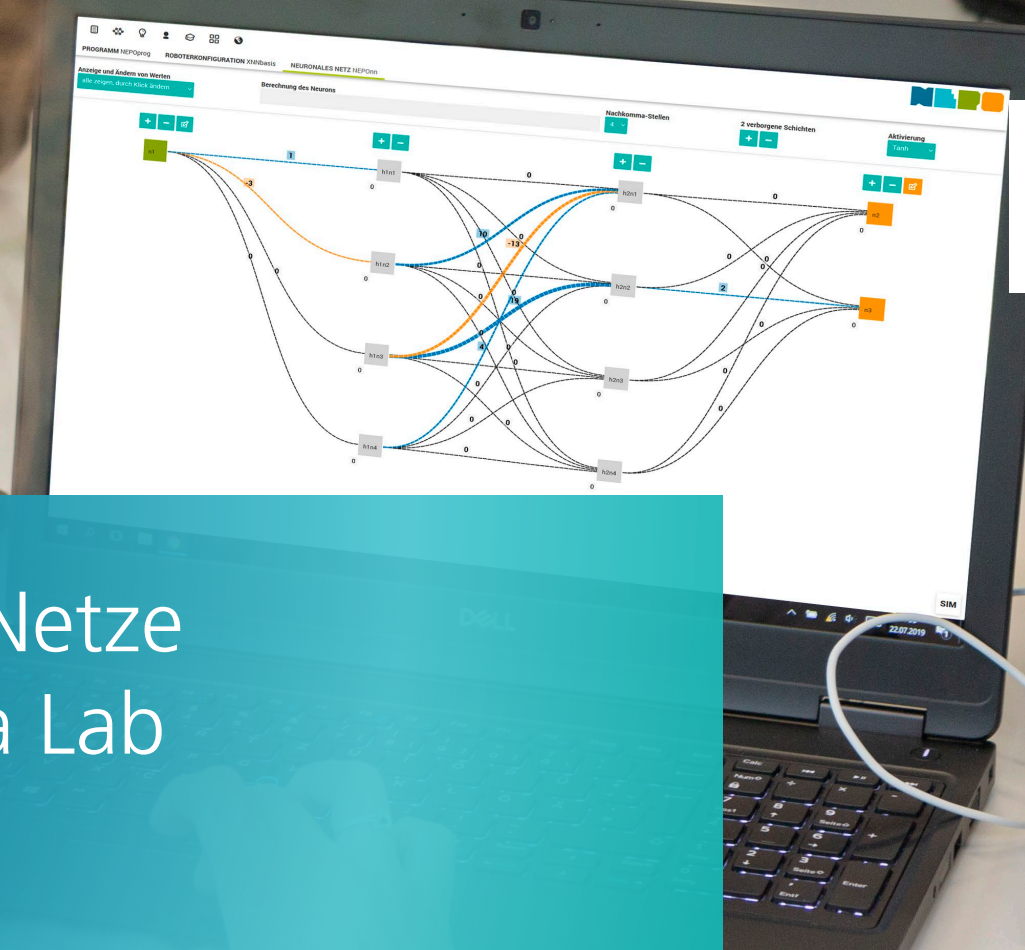


# Künstliche Neuronale Netze mit dem Open Roberta Lab

## Programmieren eines Roboters mit xNN

Thorsten Leimbach, Geschäftsfeld Smart Coding and Learning

Version: 1 (07-02-24)



# Open Roberta Lab

## xNN - Materialien

### Hinweise:

- Dieser Foliensatz baut auf den Foliensatz »[Open Roberta xNN Einführung.pdf](#)« auf.
- Wenn die Programmierung von Neuronalen Netzen im Open Roberta Lab für dich Neuland ist, empfehlen wir dir, zuerst die Einführung zu lesen.
- Die folgenden Folien zeigen, wie Roboter mit Open Roberta xNN programmiert werden können. Dies geschieht mit Hilfe der in Open Roberta Lab integrierten Robotersimulation.



# Zusammenfassung Foliensatz 101

- Im Open Roberta Lab werden Neuronale Netze mit den Blöcken der Kategorie »Neuronales Netz« programmiert
- Das Neuronale Netz wird im Tab »Neuronales Netz« erstellt.
- Das Neuronale Netz im Open Roberta Lab wird von links nach rechts durchlaufen.
- Das Neuronale Netz erhält immer nur Zahlenwerte als Eingabe-Daten und gibt auch ausschließlich Zahlenwerte aus.
- Diese Zahlenwerte können von Parameterblöcken, Variablen, Funktionsrückgaben oder auch Sensoren stammen.
- Die Berechnung des Werts eines Neurons erfolgt durch einfache Addition, Subtraktion und Multiplikation

The screenshot displays the Open Roberta Lab interface for configuring a neural network. The top navigation bar includes icons for home, settings, help, user, and simulation. The main workspace is divided into three tabs: 'PROGRAMM NEPOprog' (highlighted with a red circle), 'ROBOTERKONFIGURATION XNNbasis', and 'NEURONALES NETZ NEPOnn'. A vertical sidebar on the left lists various block categories: Aktion, Sensoren, Kontrolle, Logik, Mathematik, Neuronales Netz, Text, Farben, and Variablen. The central workspace shows a sequence of blocks: 'Mache einen NN Schritt', 'Schreibe Wert Eingabe-Neuron n1', and 'gib Wert Ausgabe-Neuron n2'. A red box highlights the 'zeige Sensordaten' block. The bottom control panel includes settings for 'Anzeige und Ändern von Werten' (set to 'alle zeigen, durch Klick ändern'), 'Berechnung des Neurons', 'Nachkomma-Stellen' (set to 2), '0 verborgene Schichten', and 'Aktivierung' (set to 'Linear'). A diagram at the bottom shows a simple neural network with two input nodes (n1 and n2) connected to a single output node (n1).



## Kapitel 01

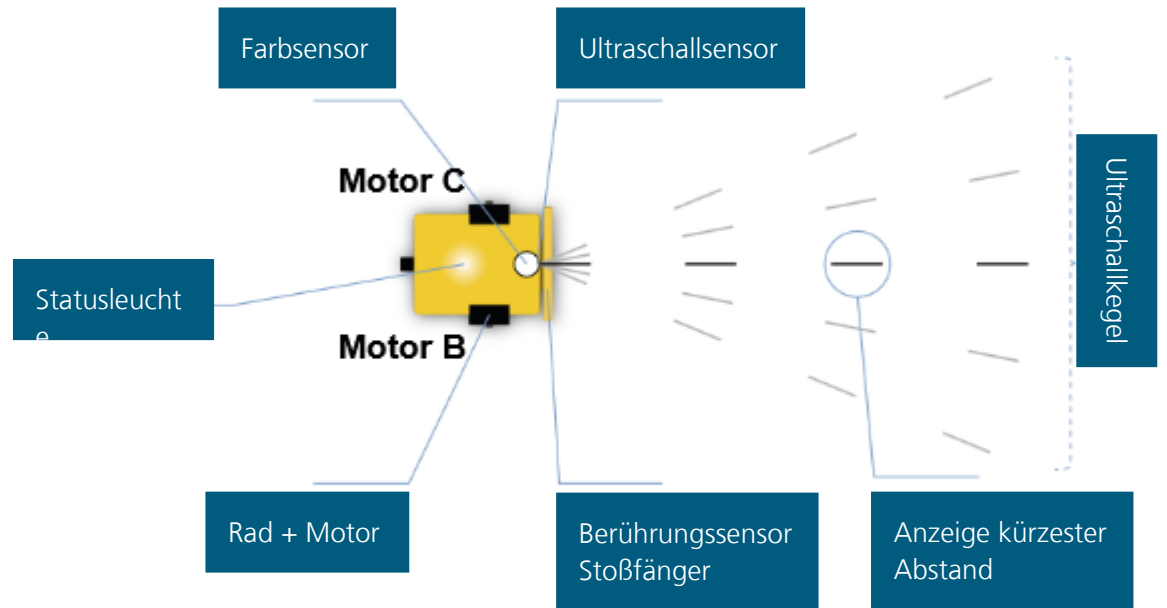
---

# Einführung xNN für Roboter

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

Anstelle eines festen Parameters können einem Eingabe-Neuron auch Sensorwerte übergeben werden. Ein Beispiel hierfür ist der Farbsensor. Der Farbsensor befindet sich in der Mitte des Roboters und erfasst die Licht- bzw. Farbwerte des Bodens, d.h. der Farbsensor ist nach unten gerichtet. Damit der Farbsensor für das neuronale Netz verwendet werden kann, muss »Licht« ausgewählt werden.

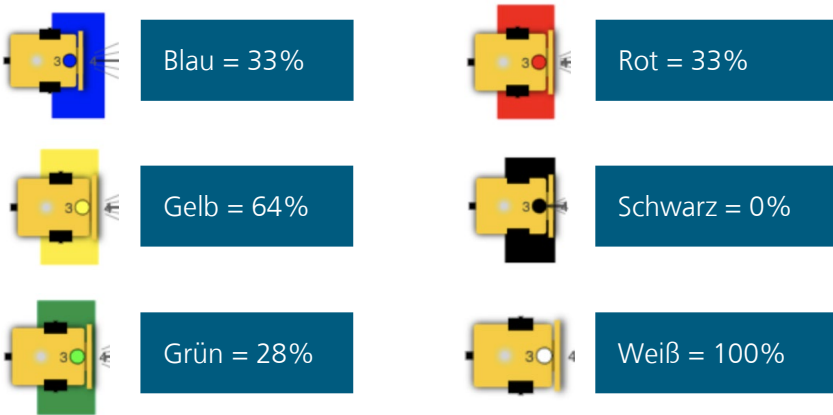


# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

Wurde »Licht« ausgewählt, gibt der Roboter die gemessenen, reflektierten Lichtwerte als Prozentwert von 0% (Schwarz) bis 100% (Weiß) an. Im Beispiel fährt der Roboter über ein blaues Feld, dabei wird ein Prozentwert von 33% gemessen.

Standardwerte:



Der aktuelle Messwert wird in der Ansicht Sensordaten angezeigt.

FPS	63
Time	0s
Robot X	218
Robot Y	221
Robot θ	0°
Motor left	0°
Motor right	0°
Touch Sensor 1	0
Light Sensor 3	33%
Ultra Sensor 4	167cm
Color Sensor 3	Blue

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

**Beachten: Die Prozentwerte der Farben Blau und Rot sind exakt gleich, nämlich 33%.** Dass diese Werte gleich sind, liegt daran, dass der Lichtsensor auf den Modus »Licht« gestellt wurde.



Im Open Roberta Lab gibt es die Möglichkeit, sich zu allen Programmierblöcken auch einen Hilfetext anzeigen zu lassen. Ziehe hierzu den Block »gib Farbe« auf die Programmieroberfläche und klicke auf das »Fragezeichen«.

Der Hilfetext beschreibt ausführlich die verschiedenen Modi.

gib Licht % Farbsensor Port 3



**»Gib Farbe«**

Mit dem Block »Gib Farbe« kannst du einem anderen Block »mitteilen«, welche Farbe der Farbsensor misst. Die Farbe wird als Datentyp »Farbe« übermittelt. Zudem kann dieser Block im Auswahlmenu auf »Licht«, »RGB« und »Umgebungslicht« gestellt werden. Diese drei zusätzlichen Typen übermitteln jeweils den Datentyp »Zahl«. Die Zahlenwerte liegen je nach gewählter Messmethode zwischen 0 und 255. Mit diesen verschiedenen Einstellung kann dieser Block je nach Anforderung eingestellt werden.

Im Modus »Farbe« sendet der Farbsensor Licht aus und erkennt, welche Grundfarbe unter dem Sensor erkannt wurde. Die Grundfarbwerte sind BLACK, WHITE, GREY, RED, GREEN, BLUE, YELLOW, BROWN.

Im Modus »Licht« sendet der Farbsensor mittels seiner roten LED Licht aus und misst die reflektierte Lichtstärke auf einer Skala von 0 bis 100.

Im Modus »RGB« sendet der Farbsensor rotes, grünes und blaues Licht aus und misst das reflektierte Licht entsprechend der Anteile Rot, Grün und Blau auf einer Skala von 0 bis 255 pro Farbe. Es wird eine Liste mit drei Werten für die jeweiligen Farben ausgegeben.

Im Modus »Umgebungslicht« wird dieselbe Skala (0-100) wie im Modus Licht verwendet. Dabei wird das ins Sensorfenster einfallende Umgebungslicht gemessen.

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

Bevor der Roboter programmiert werden kann, sollte in der Roboterkonfiguration geprüft werden, an welchem Sensor-Eingang der Farbsensor angeschlossen ist. (Standardeinstellung ist Eingang 3)

PROGRAMM NEPOprog    **ROBOTERKONFIGURATION**    NEURAL NETWORK NEPOnn

**Aktion**

**Sensoren**

**EV3**

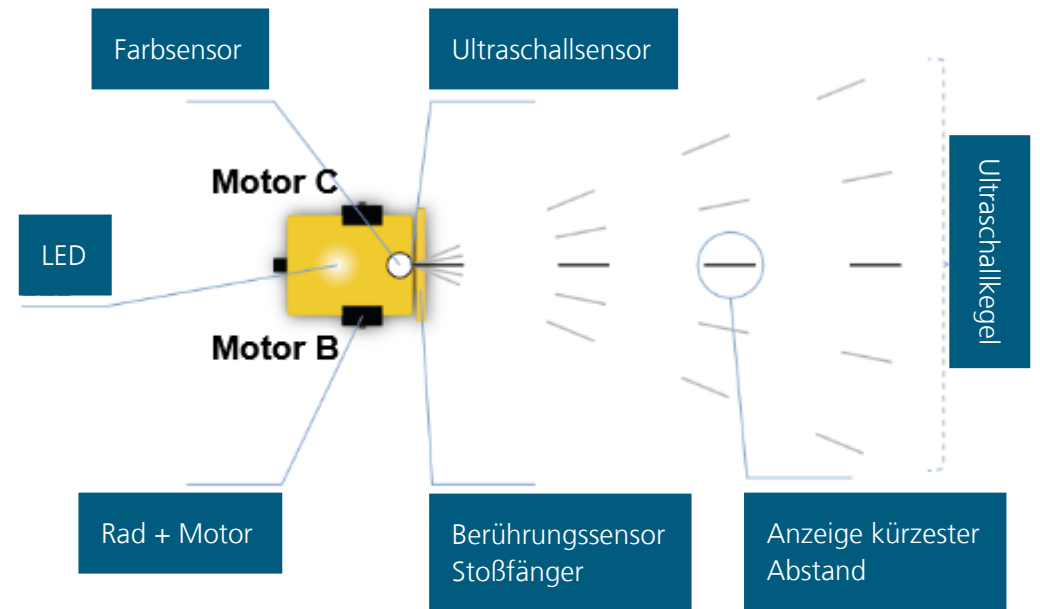
Raddurchmesser **5.6** cm  
Spurbreite **18** cm

Sensor 1 **Berührungssensor**  
Sensor 2 **Kreiselsensor**  
Sensor 3 **Farbsensor**  
Sensor 4 **Ultraschallsensor**

Motor A  
Motor B **Großer Motor**  
Regulierung **ja**  
Drehrichtung **vorwärts**  
Seite **rechts**

Motor C **Großer Motor**  
Regulierung **ja**  
Drehrichtung **vorwärts**  
Seite **links**

Motor D





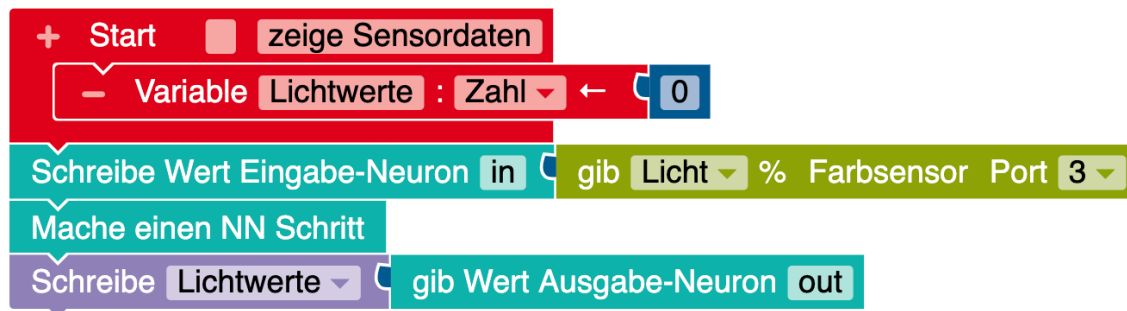
# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

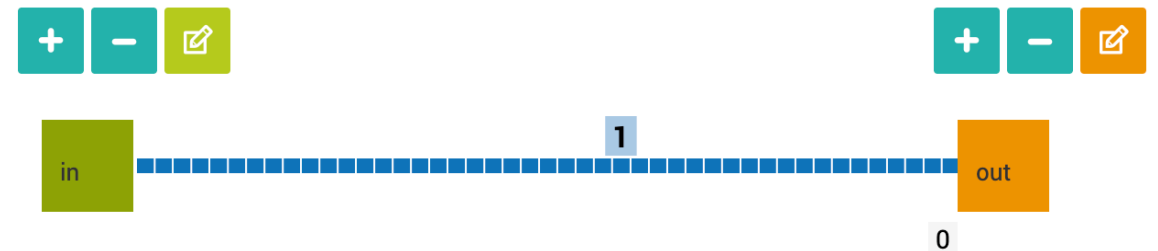
### Aufgabe

Positioniere deinen Roboter auf einer farbigen Fläche und starte das Programm. Lasse dir in der Simulation die Sensordaten anzeigen.

Das Programm sieht nun wie folgt aus.



Das dazugehörige Neuronale Netz so.



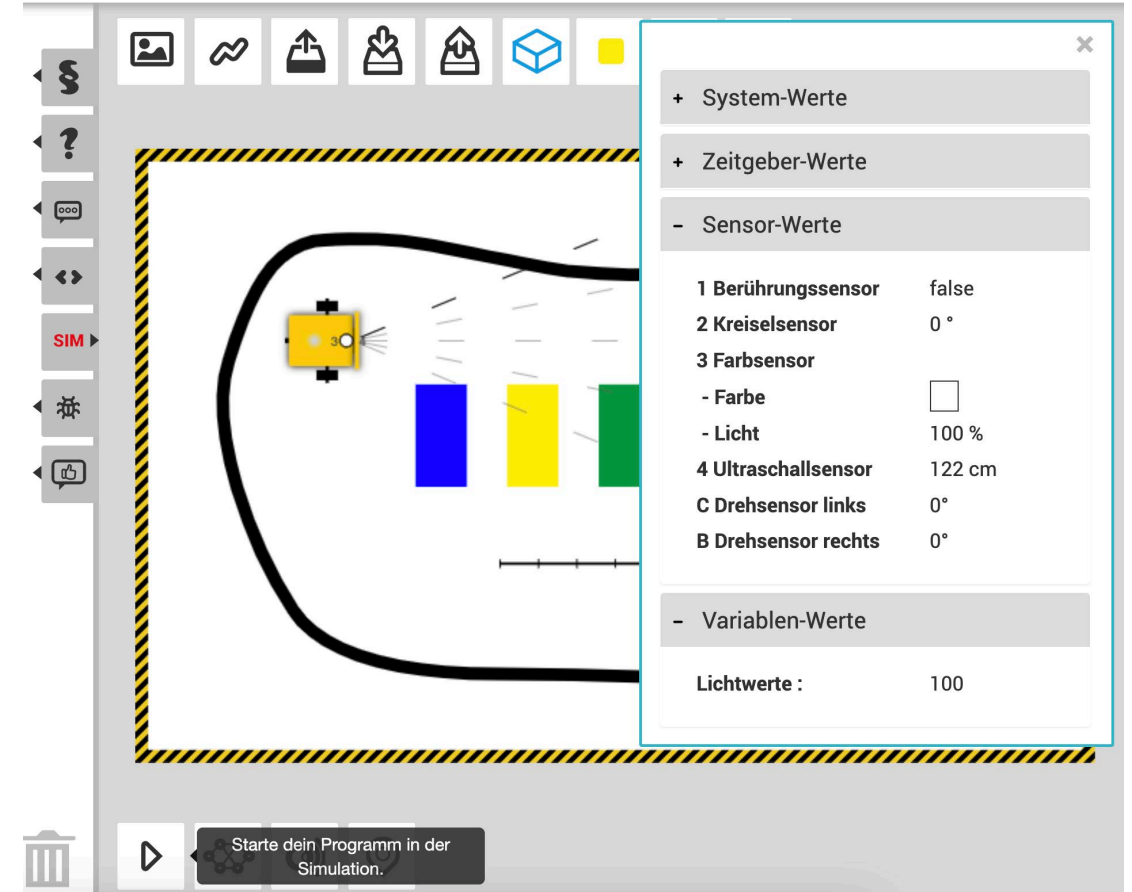
# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

### Aufgabe

Positioniere deinen Roboter über eine Farbfläche und starte das Programm. Öffne die "Ansicht der Sensordaten", um den gemessenen Wert des Sensors und den aktuellen Wert der Variablen zu sehen.

Wiederhole dies mit unterschiedlichen Farben. Ziehe dafür den Roboter auf eine von dir ausgewählte Farbe.



The screenshot displays the Open Roberta Sim interface. A yellow robot is positioned on a black track within a simulation environment. To the right of the robot are three colored blocks: blue, yellow, and green. A panel on the right side of the interface shows sensor data:

+ System-Werte	
+ Zeitgeber-Werte	
- Sensor-Werte	
1 Berührungssensor	false
2 Kreisel sensor	0°
3 Farbsensor	
- Farbe	<input type="checkbox"/>
- Licht	100 %
4 Ultraschallsensor	122 cm
C Drehsensor links	0°
B Drehsensor rechts	0°
- Variablen-Werte	
Lichtwerte :	100

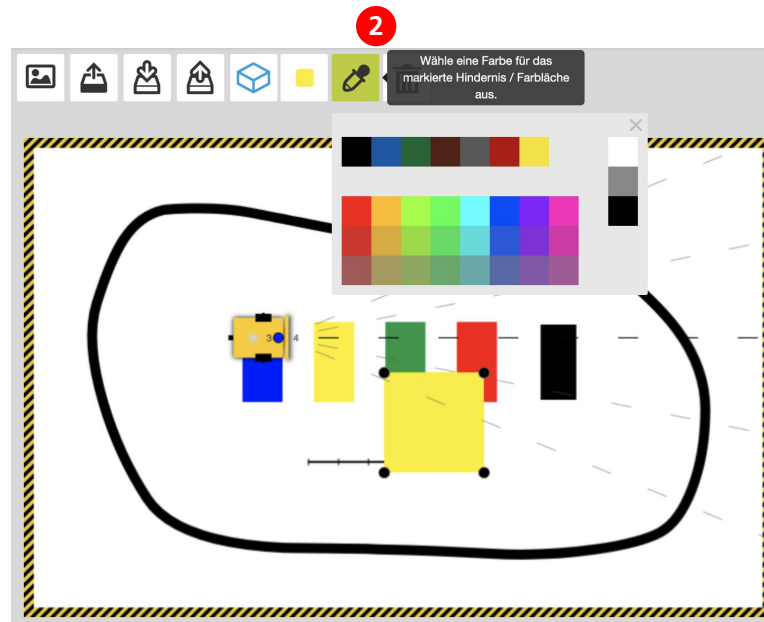
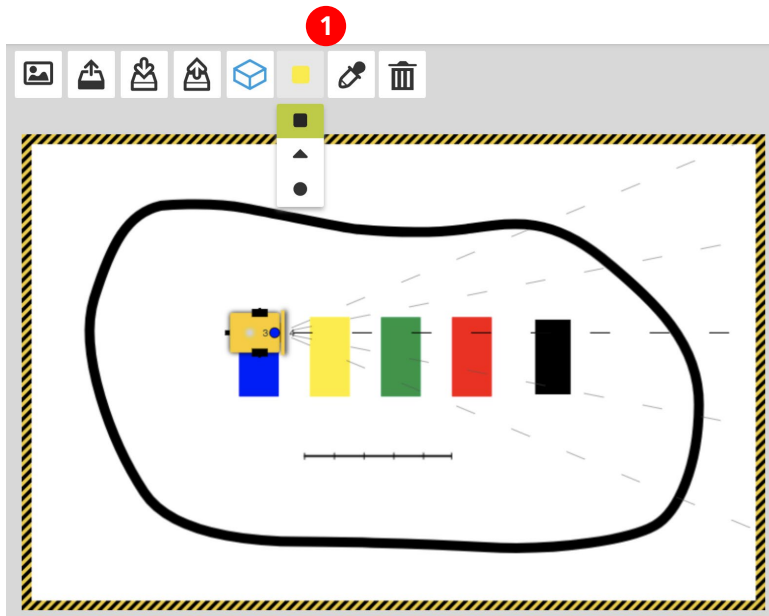
At the bottom of the interface, there is a play button and a message box that says "Starte dein Programm in der Simulation."

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

### Zusatzaufgabe

Du kannst dir auch eigene Farbflächen gestalten. Klicke dazu auf das gelbe Quadrat **1**, wähle erst eine Form aus und anschließend eine Farbe **2**.



# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

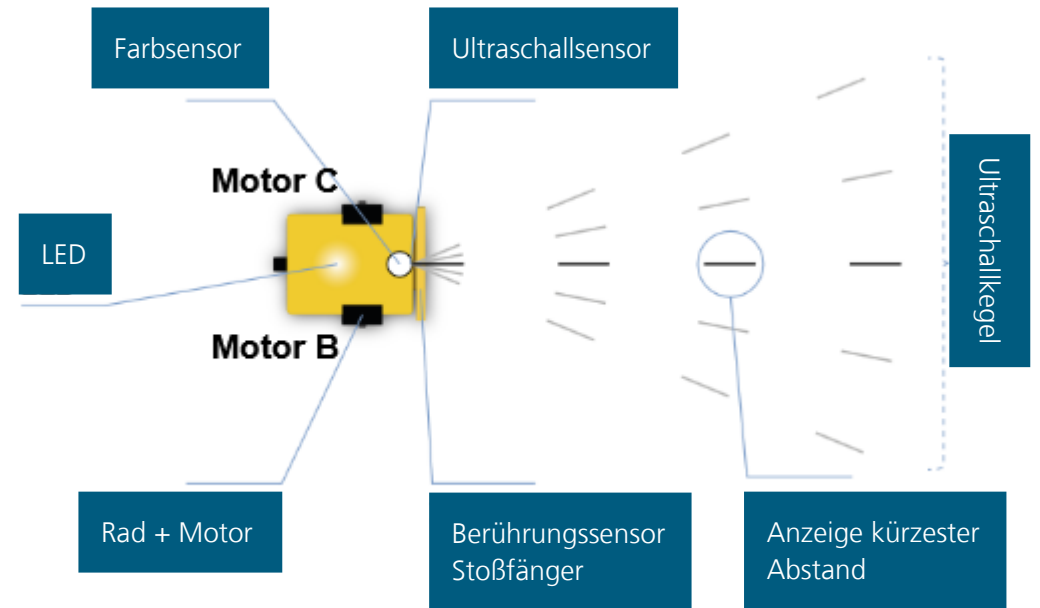
### Idee

Was passiert, wenn wir die gemessenen Lichtwerte verwenden, um das Tempo unseres Roboters zu steuern?

Wir erinnern uns: Unser Roboter hat Sensoren und zwei Motoren. Diese befinden sich links und rechts am Roboter. Damit unser Roboter z.B. vorwärts fährt, müssen sich beide Räder gleich schnell vorwärts drehen. Dafür gibt es einen einfachen Block:



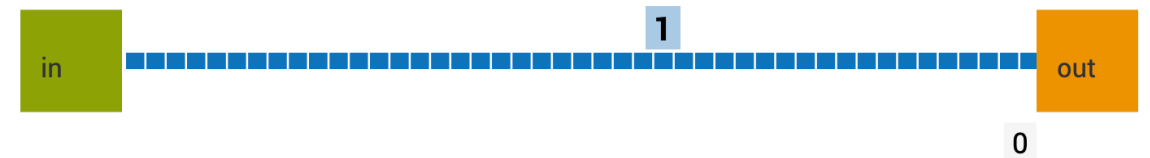
Dieser Block lässt unseren Roboter mit einer Geschwindigkeit von 30% der maximalen Geschwindigkeit vorwärts fahren. Anstelle des blauen Parameter-Blocks können wir die Variable *Ausgabe* anbringen, die die gemessenen Farbwerte beinhaltet:



# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

Das Programm sollte nun so aussehen:



### Frage:

Wie verhält sich unser Roboter? Probiere es aus und starte das Programm.



# Open Roberta Lab

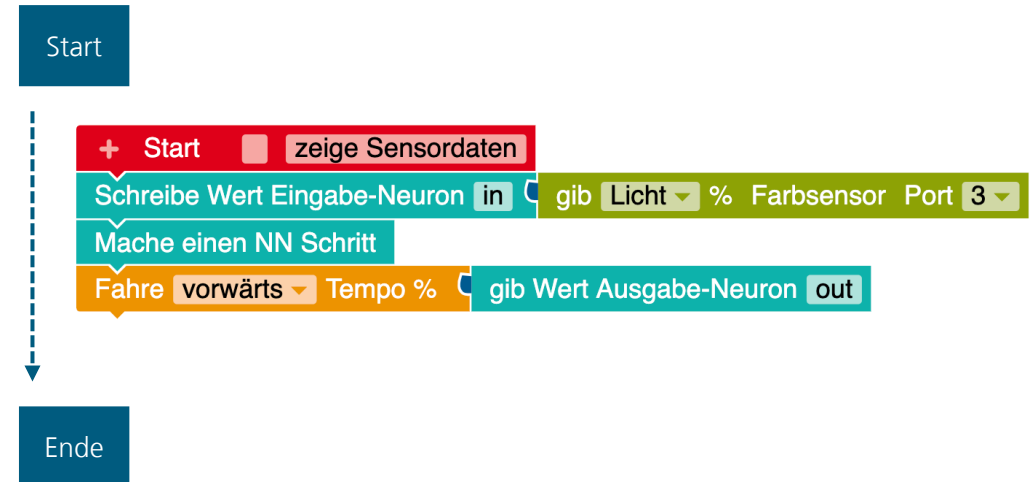
## Neuronales Netz – Roboter / Open Roberta Sim

### Warum hat sich der Roboter nur minimal bewegt?

Gehen wir das Programm Schritt für Schritt durch:

1. Das Programm startet mit dem Anlegen einer Variablen.
2. Anschließend wird bereits erzeugte das Neuronale Netz genutzt:
  1. Das Eingabe-Neuron erhält den zu diesem Zeitpunkt gemessenen Wert des Lichtsensors.
  2. Der Wert des Ausgabe-Neuron wird in der Variablen *Ausgabe* gespeichert.
3. Der Fahre-Block erhält als Tempo den Wert der *Ausgabe* Variablen.
4. Das Programm endet.

**Fazit: Das Neuronale Netz wird nur ein einziges Mal aufgerufen. Danach endet das Programm!**



# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

---

### Frage:

Was müssen wir also tun, damit unser Roboter dauerhaft fährt und dabei seine Geschwindigkeit entsprechend der gemessenen Lichtwerte anpasst?

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

---

### Antwort:

Das Programm müsste sich wiederholen, also nach dem Fahre-Block wieder von oben beginnen und den mit dem Farbsensor gemessenen Wert erneut an das Eingabe-Neuron leiten.

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

---

### Frage:

Wie oft sollen die Programmblöcke (das Programm) wiederholt werden?

### Antworten:

- 10-mal
- 100-mal
- 500-mal
- unendlich oft

# Open Roberta Lab

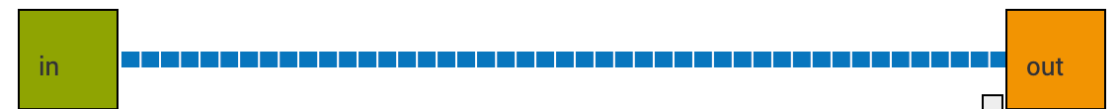
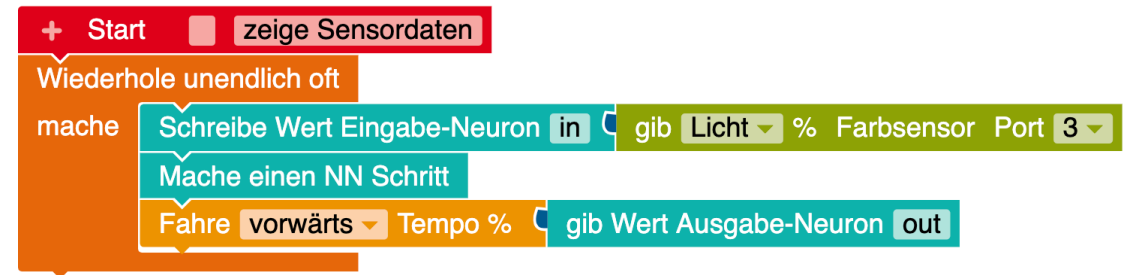
## Neuronales Netz – Roboter / Open Roberta Sim

### Frage:

Wie oft soll sich das Programm wiederholen?

### Antworten:

- 10-mal
- 100-mal
- 500-mal
- unendlich oft



Das Neuronale Netz bleibt gleich.



# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

Testet das Programm und beobachtet, wie sich der Roboter verhält.

The screenshot displays the Open Roberta Lab interface. On the left, a Scratch-style script is visible, starting with a 'Start' block followed by a 'zeige Sensordaten' block. A 'Wiederhole unendlich oft' loop contains three blocks: 'Schreibe Wert Eingabe-Neuron in' with 'Licht' selected and 'Farbsensor Port 3' as the source; 'Mache einen NN Schritt'; and 'Fahre vorwärts' with 'Tempo %' selected and 'gib Wert Ausgabe-Neuron out' as the destination. A faint 3D model of a yellow robot is visible behind the script. On the right, the simulation environment shows a yellow robot on a track with four colored blocks (blue, yellow, green, red) and a cyan square. A toolbar with various icons is located above the simulation area.

# Open Roberta Lab

## Neuronales Netz – Roboter / Open Roberta Sim

Der Roboter sollte je nach Farbe des Untergrundes schneller oder langsamer fahren und bei Schwarz anhalten.

The screenshot displays the Open Roberta Lab interface. On the left, a Scratch-style script is visible, starting with a 'Start' block and a 'zeige Sensordaten' block. A 'Wiederhole unendlich oft' loop contains three blocks: 'Schreibe Wert Eingabe-Neuron in' with 'Licht' selected and 'Farbsensor Port 3' as the source; 'Mache einen NN Schritt'; and 'Fahre vorwärts' with 'Tempo %' selected and 'gib Wert Ausgabe-Neuron out' as the destination. A faint image of a yellow robot is overlaid on the script. On the right, the simulation environment shows a yellow robot on a track. The track has a black boundary and a yellow and black striped border. In the center of the track, there are four vertical bars of different colors: blue, yellow, green, and red. A light blue square is positioned on the right side of the track. The interface includes a toolbar at the top with icons for image, line, upload, download, and other functions, and a vertical toolbar on the left with icons for help, search, and simulation controls.

# Open Roberta Lab

## Neuronales Netz – Experiment: Das Binär-Land

Der Roboter sollte je nach Farbe des Untergrunds schneller oder langsamer fahren und bei Schwarz stoppen.

The screenshot displays the Open Roberta Lab interface. On the left, a sidebar lists various programming blocks: Aktion, Sensoren, Kontrolle, Logik, Mathematik, Neuronales Netz, Text, Farben, and Variablen. The main workspace shows a program titled 'PROGRAMM NEPOprog' with the following logic:

- Start block: 'zeige Sensordaten'
- Variable block: 'output : Zahl ← 0'
- Loop block: 'Wiederhole unendlich oft' containing:
  - 'mache' block with sub-blocks:
    - 'Schreibe Wert Eingabe-Neuron in' (input: 'gib Licht % Farbsensor Port 3')
    - 'Mache einen NN Schritt'
    - 'Schreibe output' (input: 'gib Wert Ausgabe-Neuron out')
    - 'Fahre vorwärts' (Tempo: 'output')

The right side of the interface shows a simulation environment with a yellow robot on a white field, emitting sensor beams towards a blue square obstacle. The top right corner features a logo with four colored squares (blue, green, yellow, orange).

# Open Roberta Lab

## Neuronales Netz – Schwarz-Weiß-Land

---

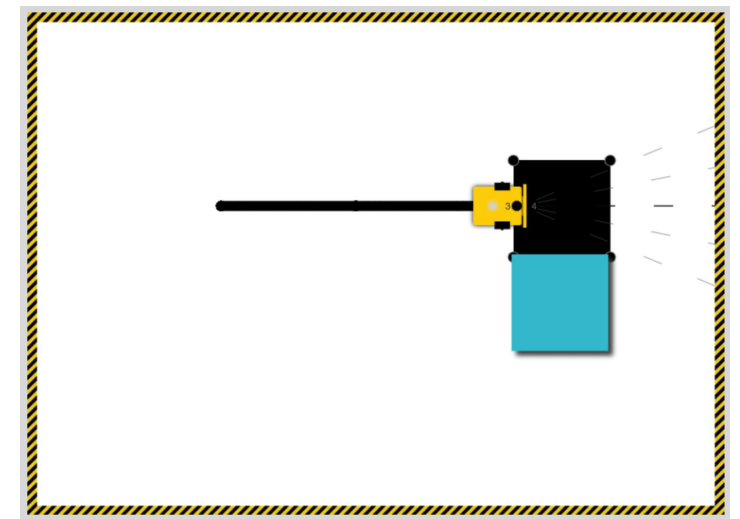
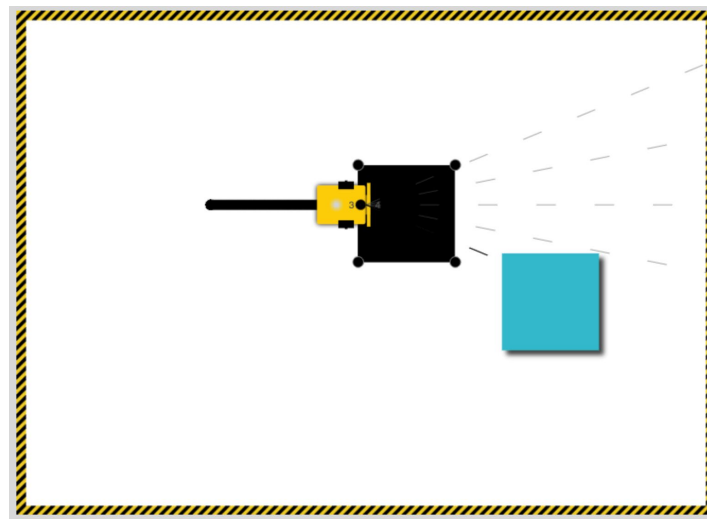
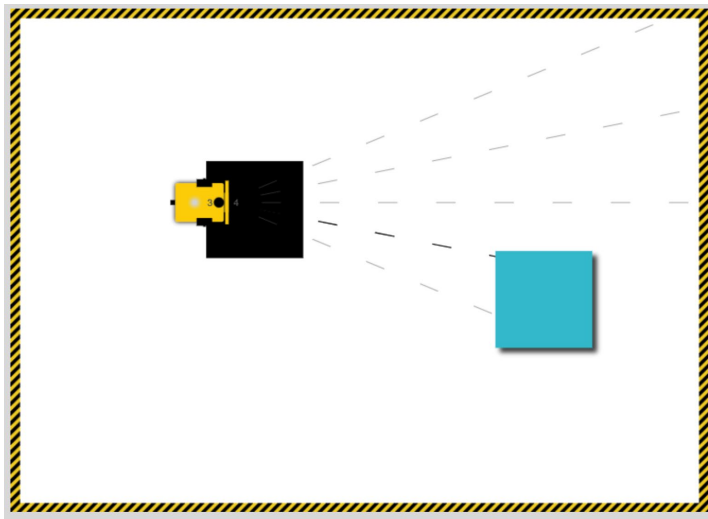
### Aufgabe:

Male ein schwarzes Feld. Positioniere den Roboter darauf und starte das Programm. Anschließend bewege das schwarze Feld nach rechts.

# Open Roberta Lab

## Neuronales Netz – Schwarz-Weiß-Land

Der Roboter sollte nun dem schwarzen Feld „gefolgt sein“.



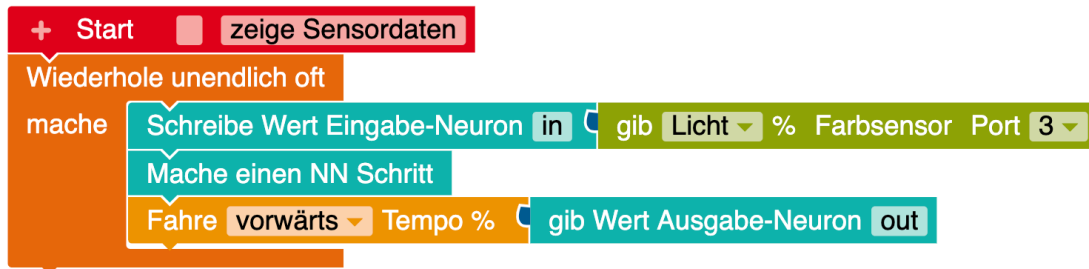


# Open Roberta Lab

## Neuronales Netz – Schwarz-Weiß-Land

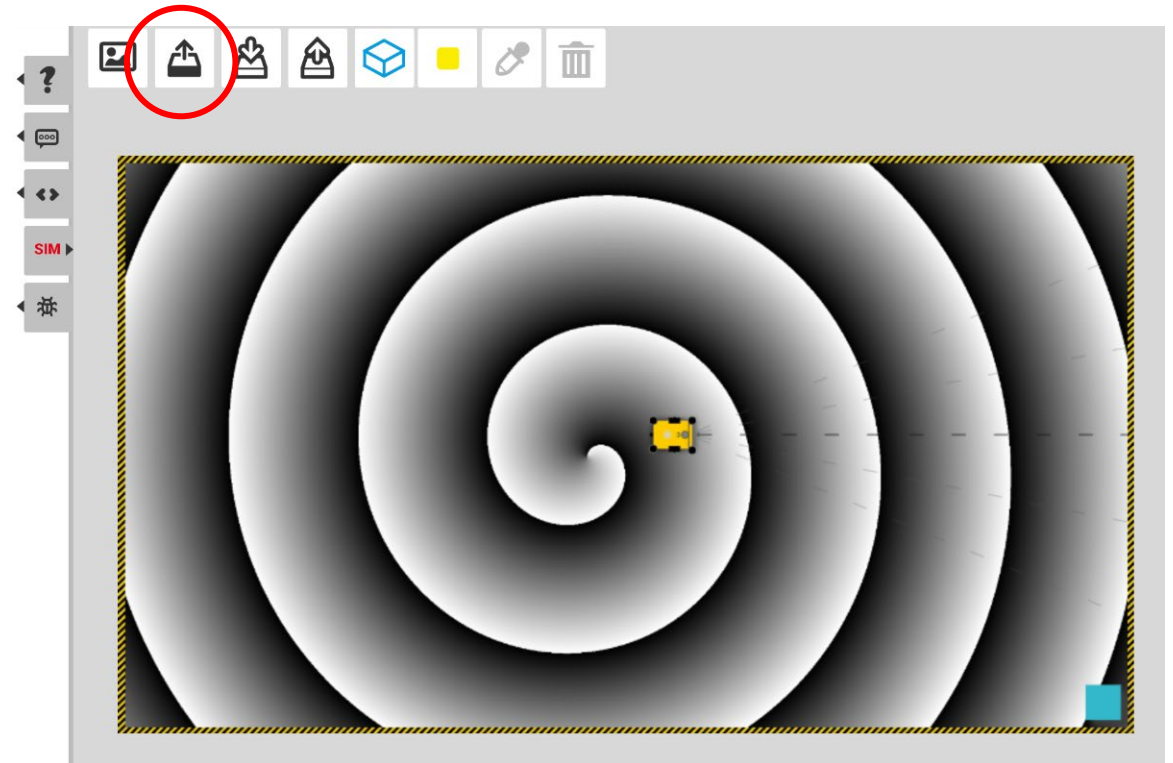
### Idee:

Eine weitere Aufgabe im Schwarz-Weiß-Land. Gestalte den Hintergrund etwas komplexer. Hier wurde eine Spirale gewählt. Je näher der Roboter dem Schwarzen Rand kommt, desto langsamer wird er.



### Hinweis:

Im in der Simulation kannst du auch eigenen Hintergrundbilder hochladen.



# Open Roberta Lab

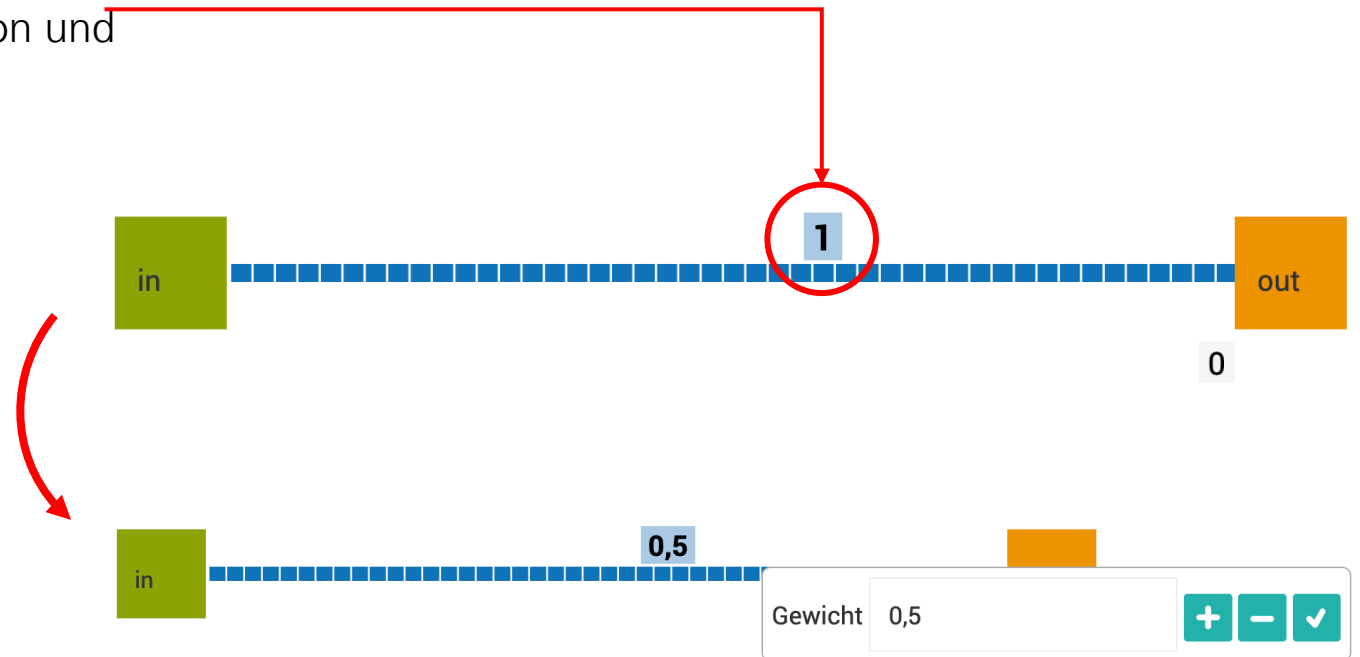
## Neuronales Netz – Schwarz-Weiß-Land

### Aufgabe:

Verringere das Kantengewicht zwischen Eingabe-Neuron und Ausgabe-Neuron z.B. auf 0,5.

### Frage:

Wie wird sich der Roboter verhalten?

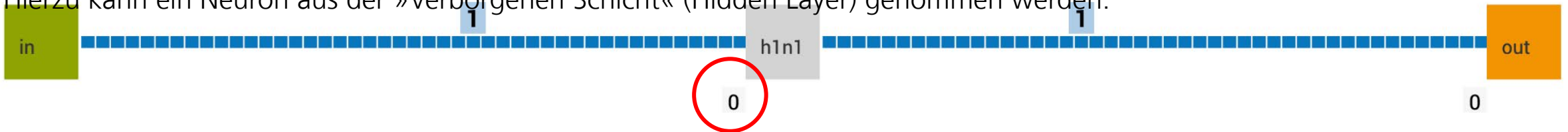


# Open Roberta Lab

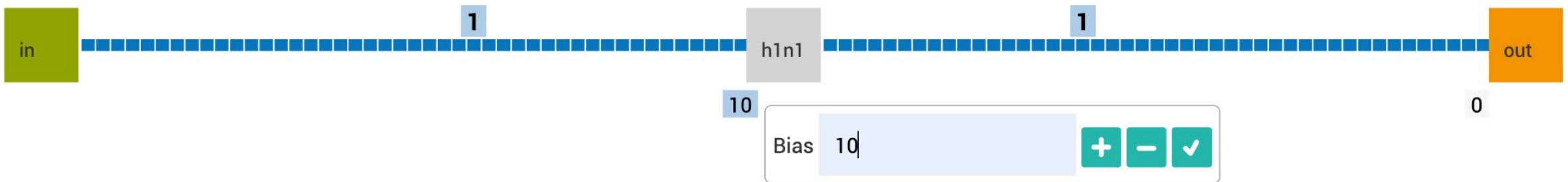
## Neuronales Netz – Schwarz-Weiß-Land

**Sollte der Roboter nicht bei Schwarz** anhalten, kann man ihm auch eine Mindestgeschwindigkeit über das Neuronale Netz mitgeben.

Hierzu kann ein Neuron aus der »Verborgenen Schicht« (Hidden Layer) genommen werden.



Diesem Neuron kann man einen „Grundwert“ mitgeben. Dieser Grundwert kann z.B. 10 sein.



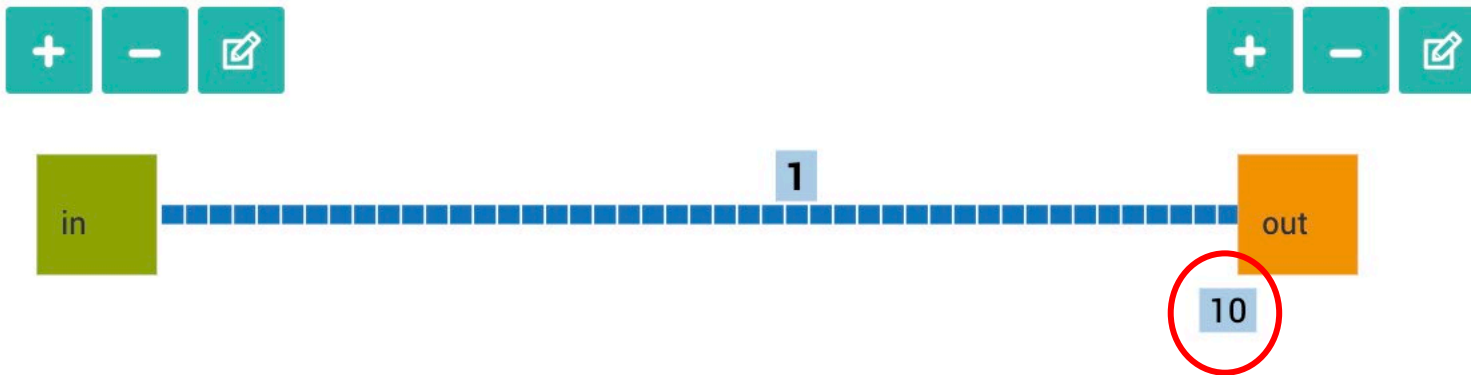
Dieser Grundwert (auch Bias genannt) führt allerdings auch dazu, dass der Roboter immer 10% schneller fährt als der vom Lichtsensor gemessene Wert beträgt. **Probiere es aus! Und überlege dir weitere Verhaltensweisen in deinem Schwarz-Weiß-Land.**

# Open Roberta Lab

## Neuronales Netz – Schwarz-Weiß-Land

Der gewünschte Effekt, dass der Roboter mit einer Grundgeschwindigkeit von 10% hat kann auch mittels folgender Alternativen erreicht werden:

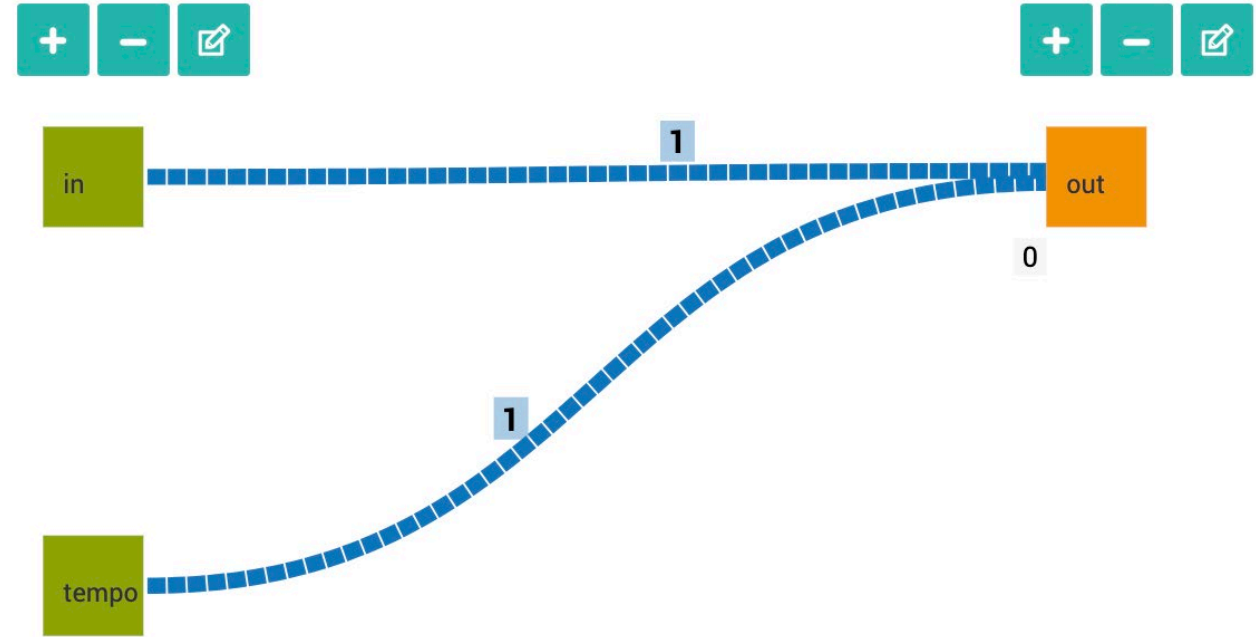
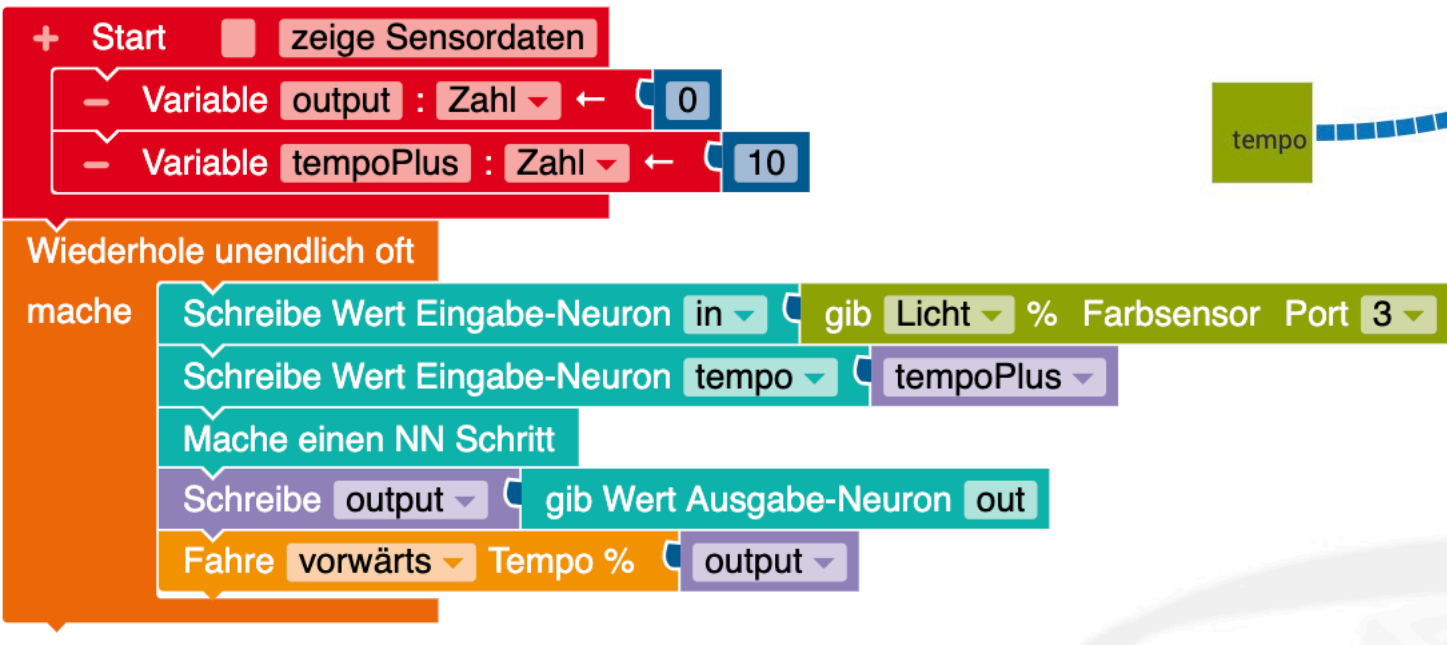
### Alternative 1:



# Open Roberta Lab

## Neuronales Netz – Schwarz-Weiß-Land

### Alternative 2:



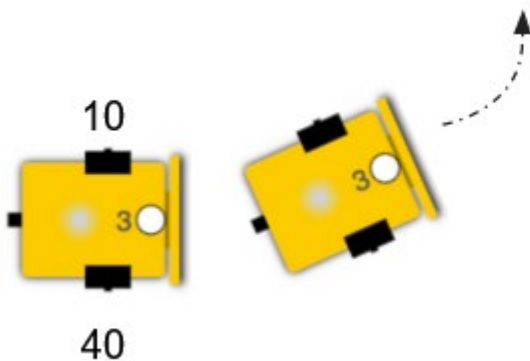
# Open Roberta Lab

## Neuronales Netz – Kurven Einführung

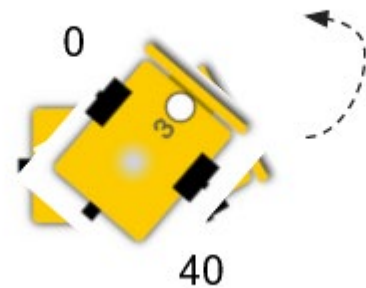
Bisher ist dein Roboter immer geradeaus gefahren. Wäre es nicht toll, wenn er auch Kurven fahren könnte? Damit dein Roboter im Binär-Land auch Kurven fahren kann, müssen wir die beiden Motoren (die links und rechts am Roboter angebracht sind) mit unterschiedlichen Geschwindigkeiten fahren lassen.

Die 3 Möglichkeiten, wie unser Roboter Kurven fahren kann:

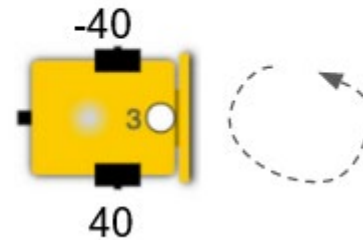
a) Großer Kurvenradius



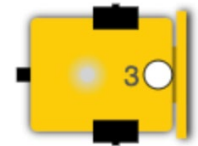
b) Kleiner Kurvenradius



c) Dreht auf der Stelle



Linker Motor / Port C



Rechter Motor / Port B

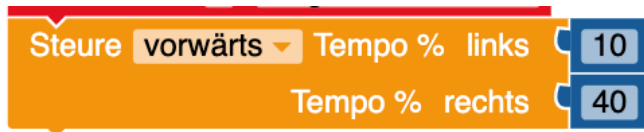
### Bemerkung zu a)

Je kleiner die Differenz zwischen den beiden Motorwerten, desto größer der Kurvenradius!

# Open Roberta Lab

## Neuronales Netz – Kurven Einführung

Für alle drei Varianten können wir einen Block verwenden:



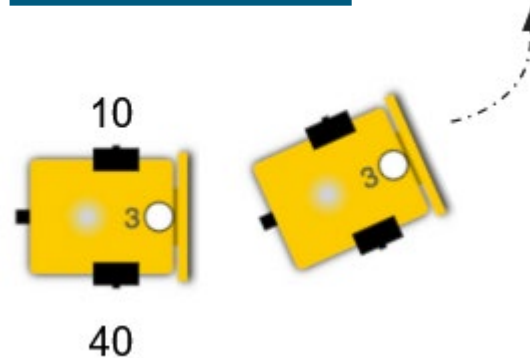
Mit diesem Block können das linke und das rechte Rad mit unterschiedlichen Geschwindigkeiten programmiert werden.

Damit wir diesen Block zusammen mit dem Neuronalen Netz programmieren können, werden zwei Ausgabe-Neuronen benötigt. Ein Ausgabe-Neuron für den linken Motor und ein weiteres für den rechten Motor.

Wie wir bereits gelernt haben, werden die Werte der Ausgabe-Neuronen in Variablen gespeichert. Deshalb muss auch eine weitere Variable angelegt werden.

Für die weiteren Beispiele genügt es, wenn wir **Variante a)** verwenden.

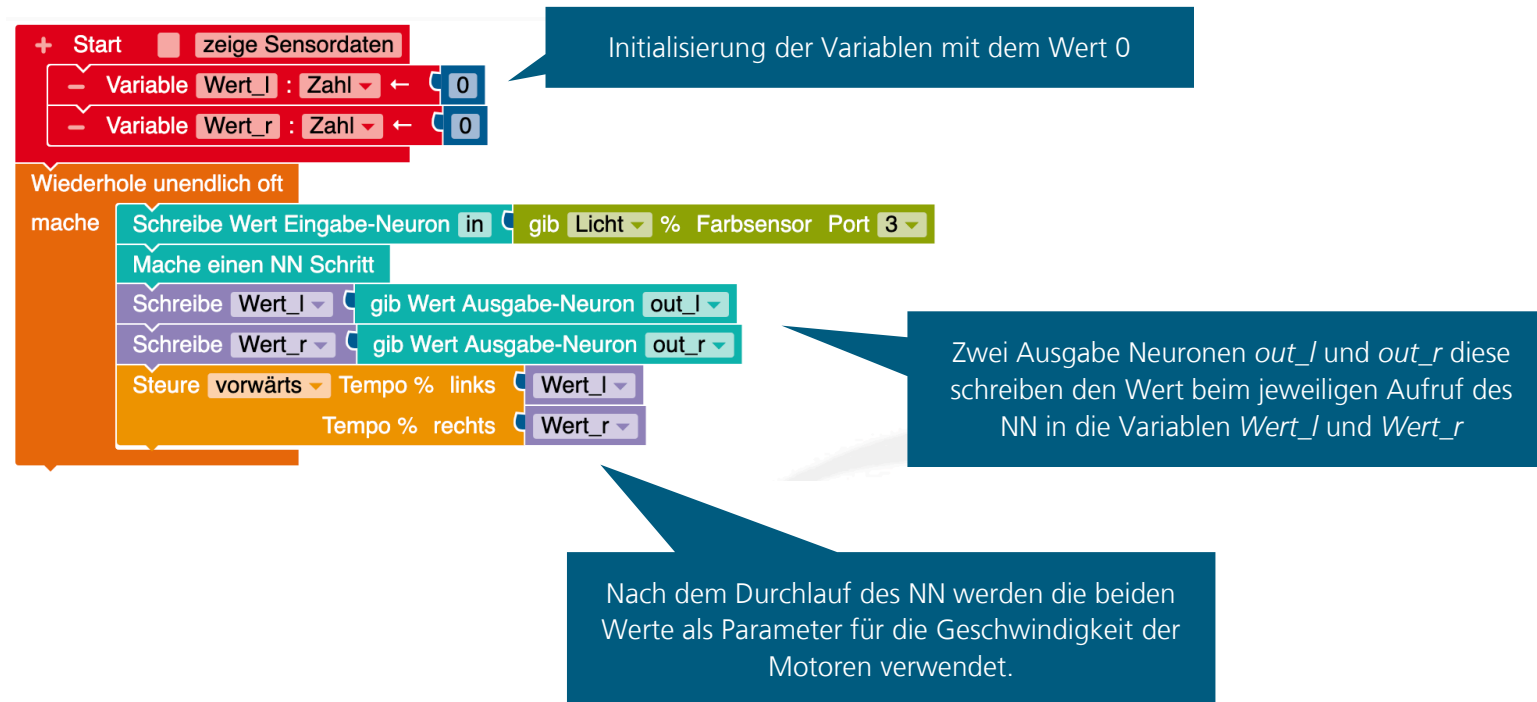
a) Großer Kurvenradius



# Open Roberta Lab

## Neuronales Netz – Kurven Programm

Das Programm mit den zwei Ausgabe Neuronen und zwei Variablen könnte wie folgt aussehen:

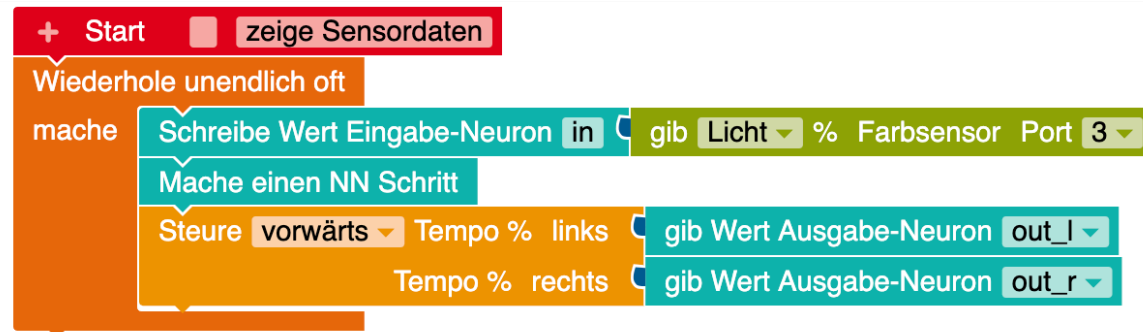




# Open Roberta Lab

## Neuronales Netz – Kurven Programm

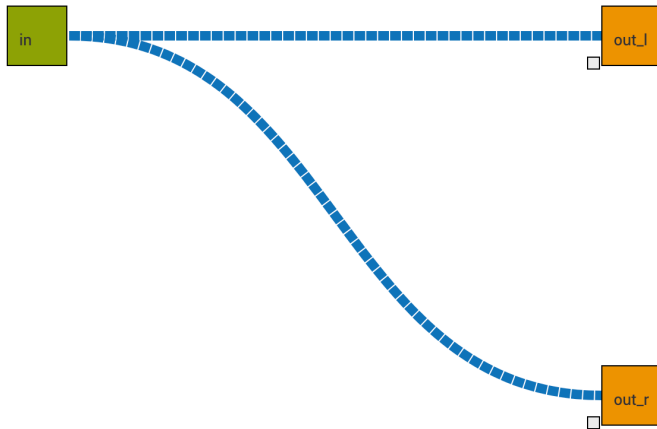
Das Programm übersichtlicher gestaltet werden, indem die Ausgabe-Neuronen *out\_l* und *out\_r* direkt mit dem Block »Steuere vorwärts« verbunden werden.



# Open Roberta Lab

## Neuronales Netz – Kurven Programm

Das Neuronale Netz sollte nun so aussehen:



### Aufgabe:

Gib den Kanten zwischen den Neuronen *in* und *out\_l* bzw. den Neuronen *in* und *out\_r* unterschiedliche Gewichte und lasse deinen Roboter in der Simulation wieder über verschiedene Farbfelder fahren. Beschreibe anschließend das Verhalten des Roboters (bzw. deine Beobachtung).

## Kapitel 02

---

# Aktivierungsfunktionen

# Open Roberta Lab

## Aktivierungsfunktionen

---

Aktivierungsfunktionen sind ein wichtiger Bestandteil jedes künstlichen neuronalen Netzes. Sie bestimmen, wie das Netz auf bestimmte Eingaben reagiert, und tragen wesentlich zur Leistungsfähigkeit des Netzes bei.

Aktivierungsfunktionen sind für Neuronale Netze wichtig, weil sie dem Netz die Fähigkeit verleihen, komplexere Beziehungen und Muster in den Daten zu modellieren. Ohne Aktivierungsfunktionen wäre ein neuronales Netz lediglich eine lineare Funktion, die nur einfache Beziehungen zwischen Eingaben und Ausgaben erkennt.

Aktivierungsfunktionen verleihen dem neuronalen Netz eine nichtlineare Eigenschaft. Das bedeutet, dass die Ausgabe des Netzes nicht einfach die Summe der Eingaben ist. Stattdessen kann die Ausgabe durch die Aktivierungsfunktion in eine Vielzahl von Möglichkeiten transformiert werden.

# Neuronales Netz

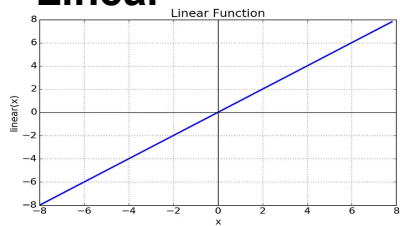
## Welche Aktivierungsfunktionen bietet das Open Roberta Lab

Es lassen sich diverse Aktivierungsfunktionen einstellen.



1

Linear



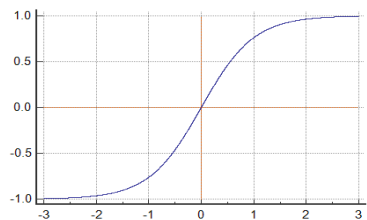
2

ReLU



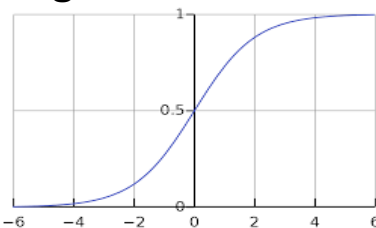
3

Tanh



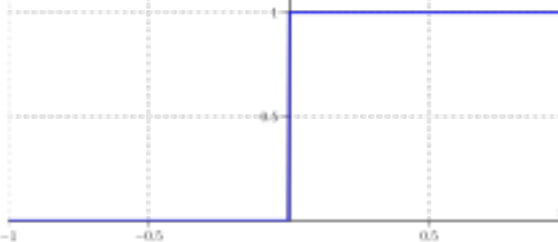
4

Sigmoid



5

Bool



Quelle: Wikipedia 12-10-23:  
[https://de.wikipedia.org/wiki/K%C3%BCnstliches\\_Neuron](https://de.wikipedia.org/wiki/K%C3%BCnstliches_Neuron)

# Neuronales Netz

## Aktivierungsfunktion – Ausgabe im Open Roberta Lab

Um die verschiedenen Aktivierungsfunktionen zu verstehen, empfiehlt es sich, die Ausgabewerte mit verschiedenen Eingabewerten zu betrachten.

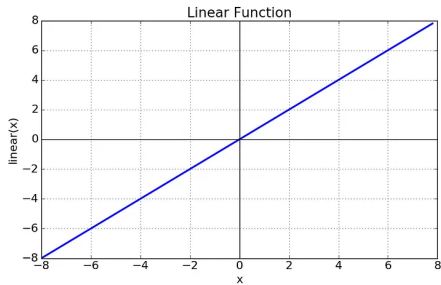
Die Variablenwerte können in der Simulation mit einem Klick auf die Schaltfläche Ansicht Sensordaten eingesehen werden.

The screenshot displays the Open Roberta Lab interface. The top navigation bar includes icons for home, settings, ideas, user, help, and a grid. The main workspace is divided into three tabs: 'PROGRAMM NEPOprog', 'ROBOTERKONFIGURATION XNNbasis', and 'NEURONALES NETZ NEPOnn'. The program editor shows a sequence of blocks: a red 'Start' block, a red 'zeige Sensordaten' block, a red 'Variable output : Zahl' block, two green 'Schreibe Wert Eingabe-Neuron' blocks, a blue 'Mache einen NN Schritt' block, a blue 'Schreibe output : gib Wert' block, and an orange 'Warte ms 1000' block. The simulation area shows a yellow robot on a track with four colored blocks (blue, yellow, green, red) and a blue square. A 'View Sensor Data' window is open, showing a list of categories: 'System-Werte', 'Zeitgeber-Werte', 'Sensor-Werte', and 'Variablen-Werte'. The 'Variablen-Werte' section is circled in red, showing 'output : 3.5'. The bottom of the interface has playback and simulation control icons.

# Aktivierungsfunktionen

Was hat es mit den Aktivierungsfunktionen auf sich?

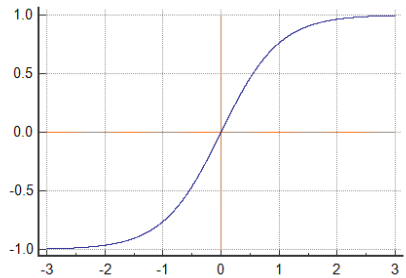
## Linear



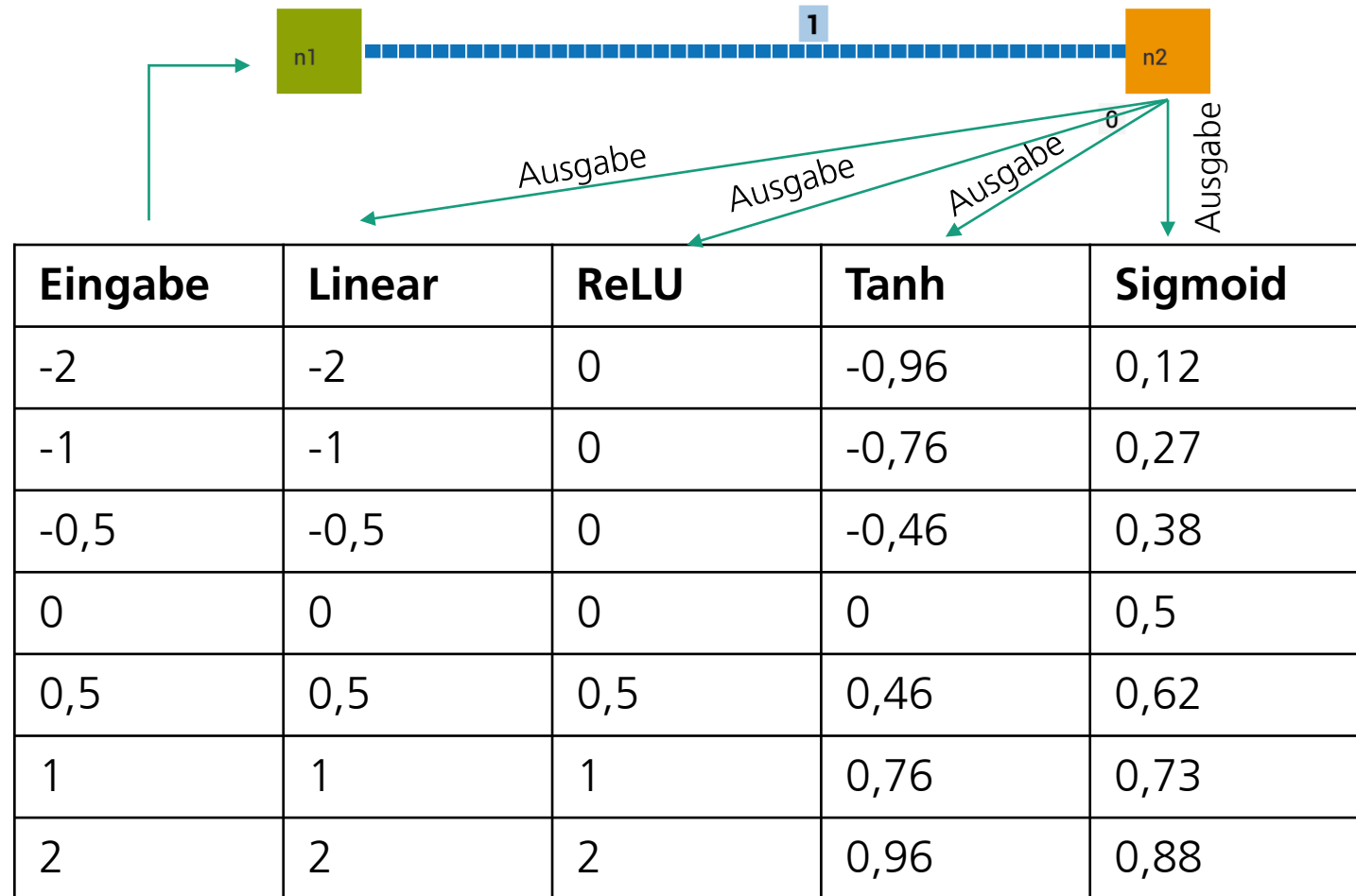
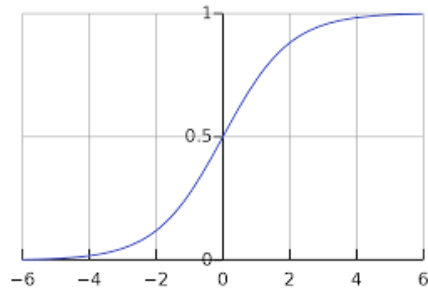
## ReLU



## Tanh



## Sigmoid



# Aktivierungsfunktionen ganz konkret

Beispiel welche Eingabe im Netz mit der jeweiligen Aktivierungsfunktion zu welcher Ausgabe führt:

- Lineare Aktivierungsfunktion: Eingabe -2 = Ausgabe -2
- ReLU Aktivierungsfunktion: Eingabe -2 = Ausgabe 0
- Tanh Aktivierungsfunktion: Eingabe -2 = Ausgabe -0,96
- Sigmoid Aktivierungsfunktion: Eingabe -2 = Ausgabe 0,12

Eingabe	Linear	ReLU	Tanh	Sigmoid
-2	-2	0	-0,96	0,12
-1	-1	0	-0,76	0,27
-0,5	-0,5	0	-0,46	0,38
0	0	0	0	0,5
0,5	0,5	0,5	0,46	0,62
1	1	1	0,76	0,73
2	2	2	0,96	0,88



## Kapitel 03

---

# Ausblick

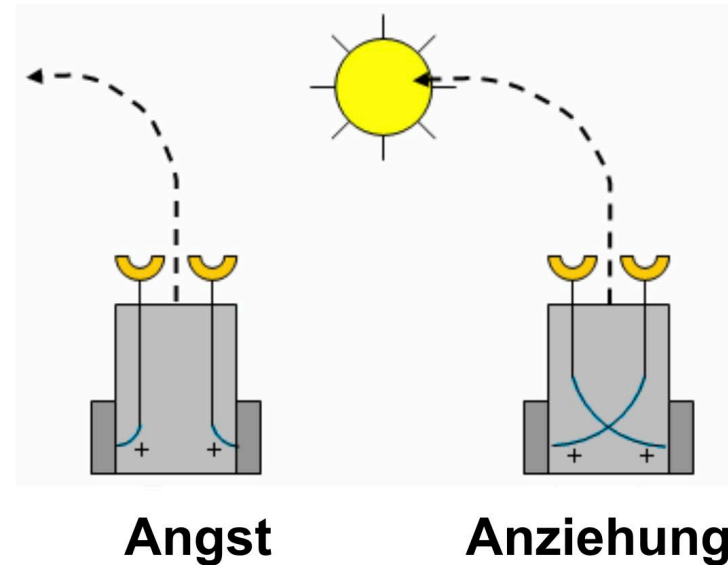
# Ausblick

## Braitenberg Vehikel

Der dritte Foliensatz dieser Reihe wird sich mit den sogenannten Braitenberg Vehikeln befassen.

In der deutschsprachigen Ausgabe des Online-Lexikons Wikipedia findet man nur indirekt eine Beschreibung der Braitenberg-Vehikel: (DE-März-2010)

»In der Roboterszene ist Braitenberg durch sein Buch “Vehicles: Experiments in Synthetic Psychology” bekannt geworden. Hierin beschreibt er in 14 Beispielen, wie mit Sensoren ausgestattete Fahrzeuge autark auf Umweltreize reagieren können. Spannend ist das vor allem durch die Einsicht, wie scheinbar sehr komplexes Verhalten schon durch verblüffend einfache Mechanismen bewirkt werden kann.«



# Kontakt

---

## **Thorsten Leimbach**

Geschäftsfeldleiter Smart Coding and Learning  
Roberta-zentrale@iais.fraunhofer.de

Fraunhofer-Institut für Intelligente Analyse-  
und Informationssysteme IAIS  
Schloss Birlinghoven 1  
53757 Sankt Augustin

**[www.iais.fraunhofer.de](http://www.iais.fraunhofer.de)**