Übersicht der Programmierung des TXT 4.0 mit ROBO Pro Coding (Kurzreferenz)

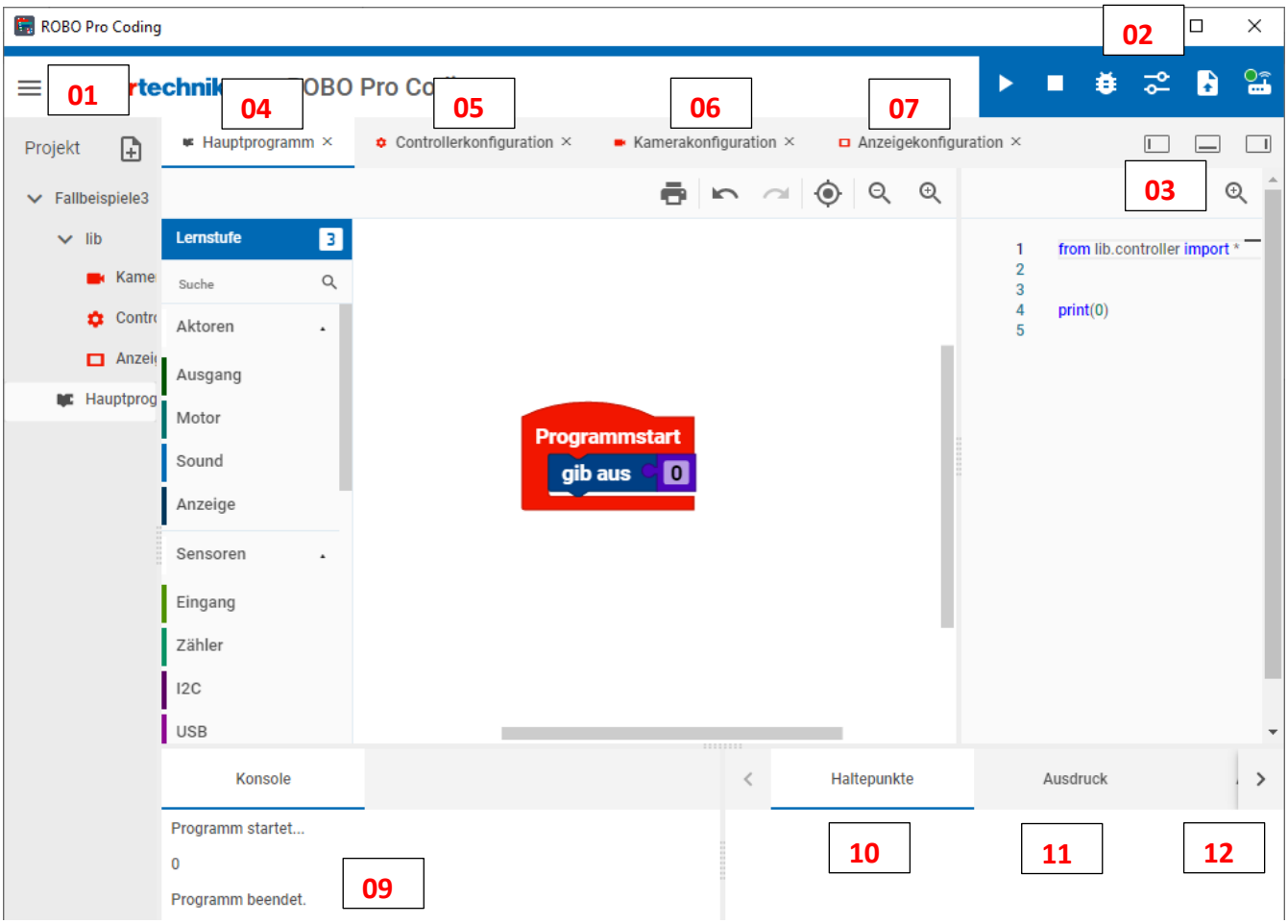
Zusammengestellt von A. Chobe
(axel.chobe@t-online.de / chobe.info)

Aktuelle Version vom 02.07.2024

Inhalt

Außer der Bedienungsanleitung (https://www.fischertechnik.de/de-de/service/elearning/lehren/txt-40-controller) des Controllers und einer Einleitung zu ROBO Pro Coding (https://www.fischertechnik.de/de-de/service/elearning/lehren/base-set-und-add-ons) gibt es noch keine Anleitung, um ROBO Pro Coding zu erlernen. Auch die vorgefertigten Programme zu den Modellen sind nicht immer hilfreich, um die Grundlagen zu verstehen.
Mit dem folgenden Heft habe ich versucht ein einfaches und umfangreiches Dokument zu erstellen, um ein besseres Verständnis für die Programmiersprache zu erhalten.

1 Oberfläche

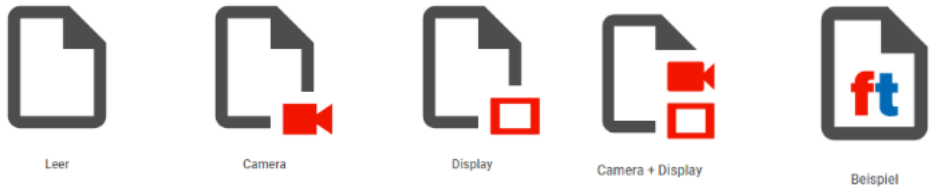


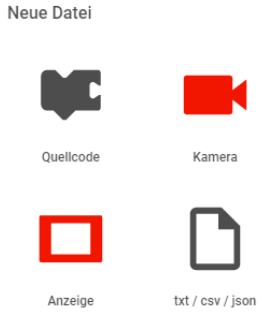
1.1 Menü (01)

1.1.1 Voreinstellungen/Einstellungen

1.1.2 Projekte

- **Neu** (Leer; Camera; Display; Camera + Display; Beispiel)





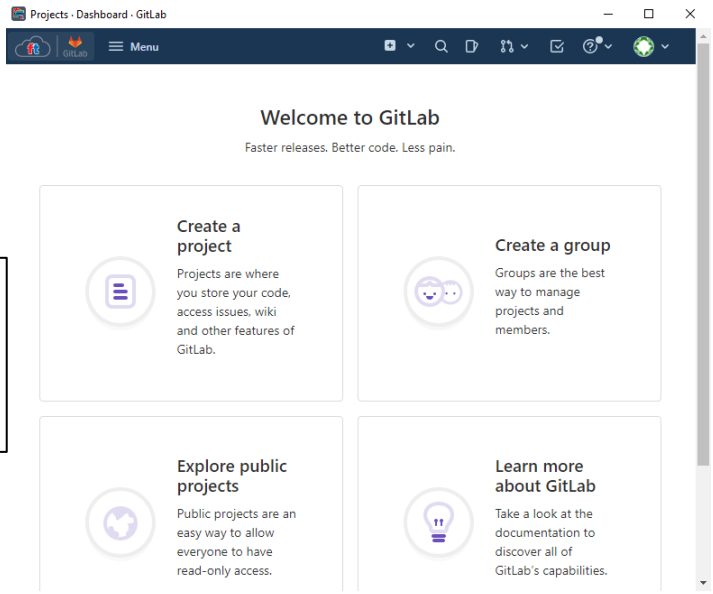
Die Konfigurationsfenster können später über das Symbol  erweitert werden.

Unter dem Punkt Beispiele können verschiedene Programme für die entsprechenden Baukästen aufgerufen werden. Zuert den Kasten wählen und dann das entsprechende Programm.

Um den Quellcode übersichtlicher zu gestalten, ist es möglich zusätzliche Module anzulegen.
Hier können Funktionen geschrieben werden, die dann im Hauptprogramm aufgerufen werden. (Siehe Punkt Import)
Alle Projetteile (Hauptprogramm, Controllerkonfiguration usw.) werden in einer Libery gespeichert.

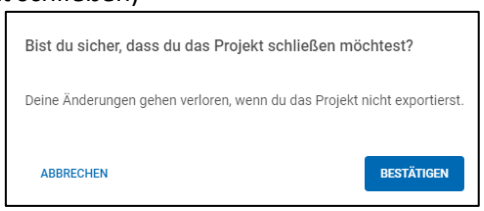
- **Exporte** (Lokal; fischertechnik GitLab)

Die erstellten Programme, können auf dem eigenen PC oder unter fischertechnik GitLab abgespeichert werden. Für diesen Fall muss man sich dort anmelden. Dazu kann man sein Fischertechnikzugang auswählen.



- **Importe** (Lokal; fischertechnik GitLab)
Auch das speichern dr Programme erfolgt entweder lokal oder auf GitLab.

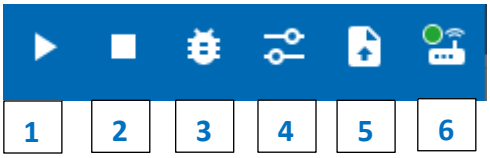
- **Schließen** (Projekt schließen)
Vorher erfolgt die Abfrage, ob das Projekt gespeichert werden soll.



1.1.3 Hilfe

- Dokumentation
- Datenschutzrichtlinie
- Impressum

1.2 Blaue Leiste (02)



1.2.1 Programm starten (1)

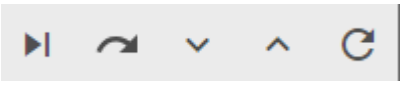
Der Code wird an den Controller übergeben und das Programm startet.

1.2.2 Programm beenden (2)

Das Programm auf dem Controller wird beendet.

1.2.3 Debugging (3)

Das Programm wird auf Fehler überprüft und diese werde im Konsolenfenster (09) angezeigt.

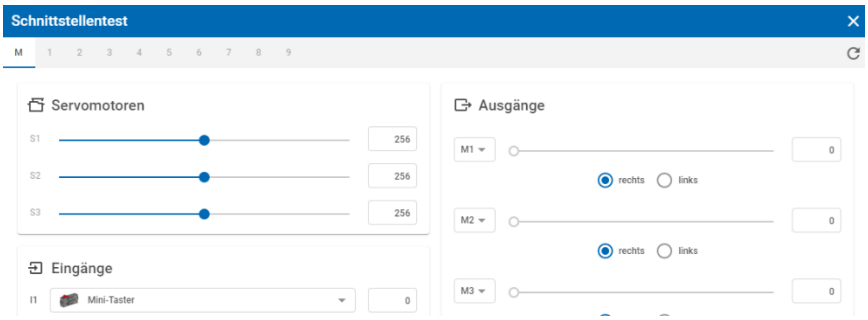


v.l.n.r.: Fortfahren, Überspringen, *Einzelschritt*, Rückschritt, Neustart

```
6  while True:
7      A = 1
8      print(A)
9      A = (A if isinstance(A, (int, float)) else 0) + 1
10     print(A)
11
```

Mit Einzelschritt, wird das Programm zeilenweise abgearbeitet und im Blockly sowie im Python farbig markiert. Entsprechende Textausgaben werden wie immer in der Konsole angezeigt.

1.2.4 Schnittstellentest (4)



Alle angeschlossenen Aktoren und Sensoren, können auf ihre Funktionsfähigkeit getestet werden.
Beim Controller RX können auch bei den 3 I²C-Bausteine die Werte ausgelesen werden.

1.2.5 Programm hochladen (5)

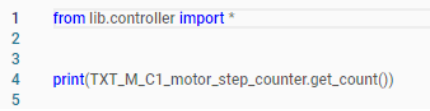
Das Programm (xxx.py) wird auf den TXT 4.0 geladen und kann dann ohne Entwicklungsumgebung gestartet werden.

1.2.6 Controller verbinden (6)

Verbindung kann über WLAN, USB oder Bluetooth hergestellt werden. Der Schlüssel kann in Controller ausgelesen werden.
Im Erfolgsfall wird das Symbol mit einem grünen Punkt angezeigt.

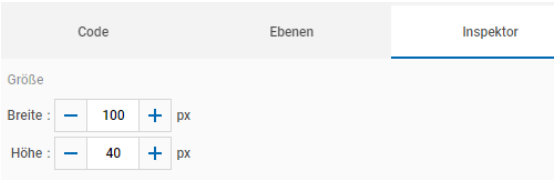
Weiterhin sind nun alle anderen Befehle aktiviert.

1.3 Rechte Seite (03)

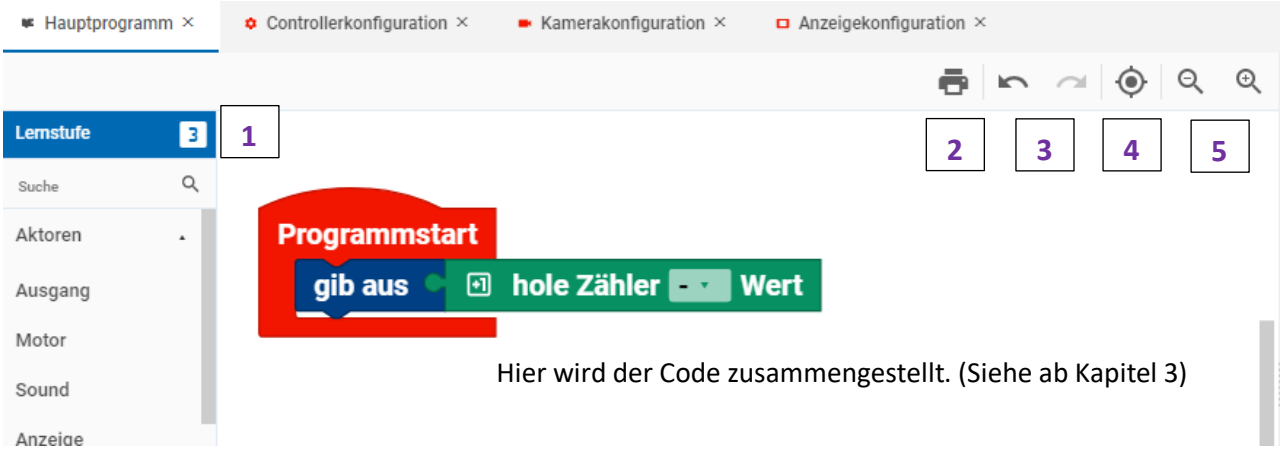


```
1  from lib.controller import *
2
3
4  print(TXT_M_C1_motor_step_counter.get_count())
5
```

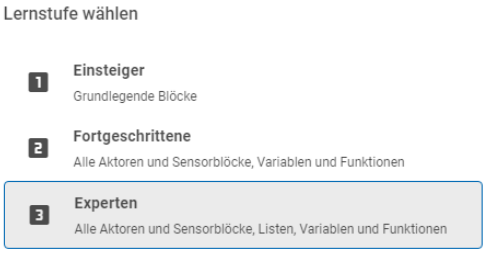
Diese Anzeige ändert sich entsprechend der Auswahl des Fensters. Im Hauptprogramm und der Controllerkonfiguration wird der Python-Code angezeigt. In der Kamera -und Anzeigekonfiguration, ist der dritte Punkt (Inspektor) wichtig. Hier werden die Eigenschaften des ausgewählten Elementes, wie z.B. des OK-Buttons eingestellt. Ligen bei der Kamerakonfiguration mehrere Sensorflächen übereinander, können unter dem Punkt Ebenen, die einzelnen Flächen ein- bzw. ausgeschaltet werden.



1.4 Hauptprogramm (04)



1.4.1 Lernstufen (1)



Es können unterschiedliche Programmier-Lernstufen eingestellt werden. Je nach benötigtem Schwierigkeitdgrad kann zwischen 3 Lerstufen ausgewählt werden.

1.4.2 Drucken (2)

Hierbei wird nur der Quellcode (das Programm) ausgedruckt.

1.4.3 Schritt zurück/vor (3)

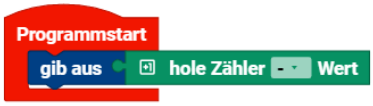
Der jeweils letzte Schritt kann zurückgenommen oder wieder benutzt werden.

1.4.4 Zentrieren (4)

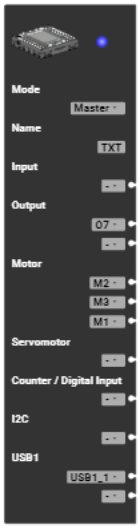
Der Code wird in die Mitte des Bildschirmes gerückt.

1.4.5 Verkleinern/ Vergößern (5)

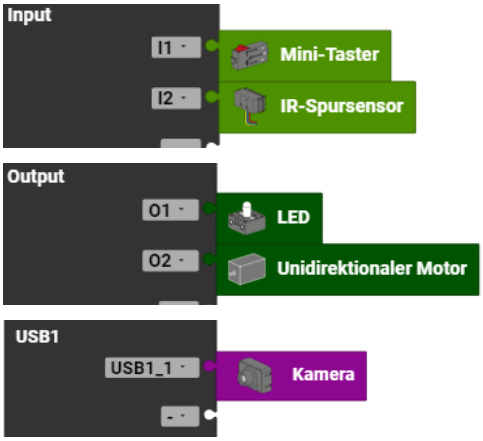
Der gesamte Quellcode wird verkleinert oder vergrößert dargestellt.
Tip: Einzelne Programmgruppen können auch zusammengefaltet werden, um die Übersichtlichkeit zu verbessern. Dazu rechte Maustaste auf Objektgruppe und die Option „Baustein zusammenfalten“ wählen.

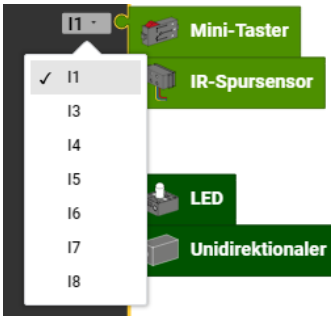


1.5 Controllerkonfiguration (05)



Es wird Symbolisch der Controller dargestellt. Hier müssen nun die entsprechenden Aktoren und Sensoren Positioniert werden. (siehe Beispiele)

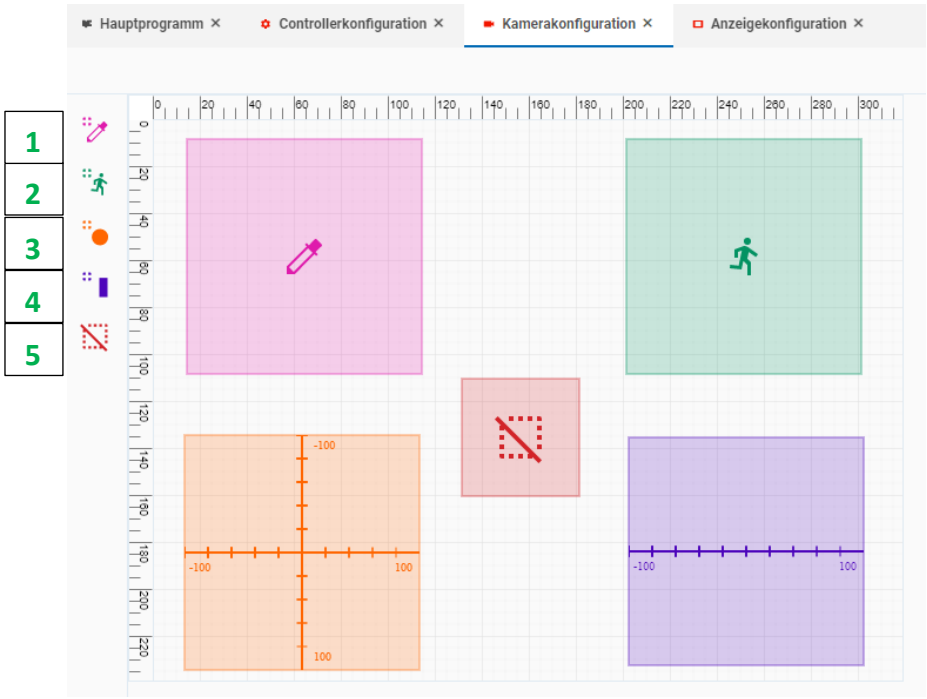




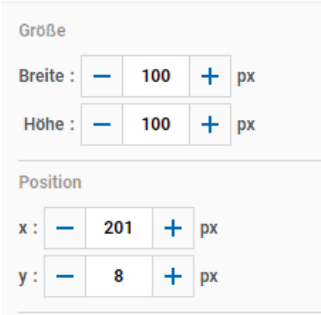
Die Position kann nachträglich über das kleine Dreieck geändert werden.

1.6 Kamerakonfiguration (06)

Anwendung unter 4.4



Diese



Elemente sind im Inspektor, für alle Elemente gleich. Jedes Element hat dazu noch individuelle Einstellmöglichkeiten.

1.6.1 Farberkennung (1)

FARBERKENNUNG

Name	Rot	Grün	Blau	Hex	Hue	Saturation	Wert
color_detector	118	115	107	76736b	43	9	46

Zur Kontrolle der Werte, gibt es eine Testumgebung unter der Konfigurationseinstellung im Punkt „camera“.

Der Inspektor wird um den Punkt Kontrast erweitert.

Der Abruf im Hauptprogramm erfolgt unter dem Punkt:

1.6.2 Bewegungserkennung (2)

BEWEGUNGSERKENNUNG

Name	Bewegung
motion_detector	false

Wird im aufgezogenen grünen Fenster eine Bewegung erkannt, ändert sich die Eigenschaft von motion_detector auf True.

Der Inspektor wird um den Punkt Toleranz erweitert:

1.6.3 Ballerkennung (3)

BALLERKENNUNG

Name	X	Y	Durchmesser
ball_detector			

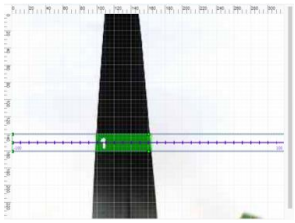
Der Inspektor wird um folgende Punkte erweitert:

In der Testumgebung wird die Position und der Durchmesser angezeigt.

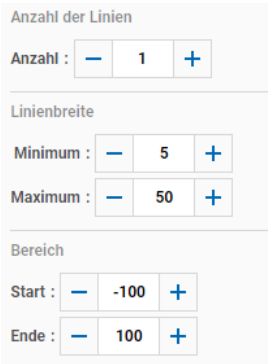
1.6.4 Linienerkennung (4)

LINIENERKENNUNG

Name	Linie	Position	Breite	Rot	Grün	Blau
line_detector	1	-30	10	0	0	0



Hier wird die Abweichung von der Mitte des Erkennungsfensters erkannt.

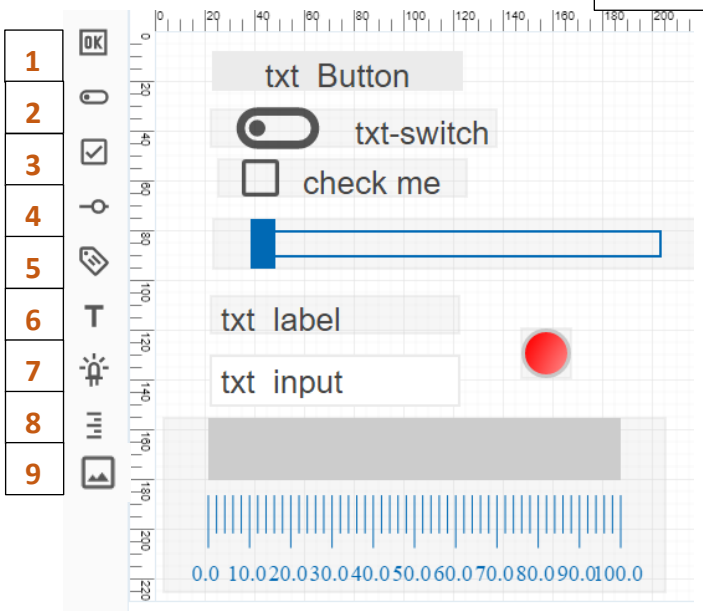


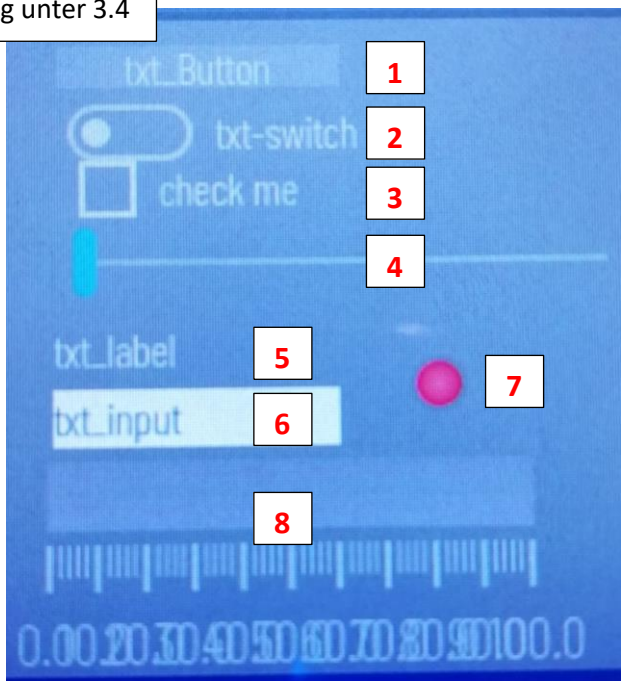
1.6.5 Aussparung (5)

Dieses Element dient dazu, Bereiche aus dem Erkennungsbereich auszublenden.

1.7 Anzeigekonfiguration (07)

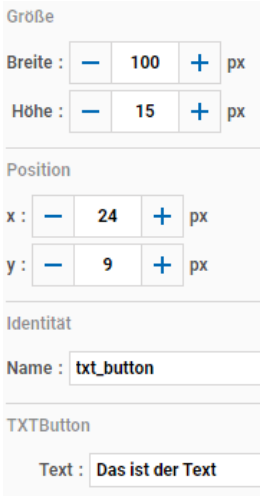
Anwendung unter 3.4





1.7.1 Inspektor

Hier können die Eigenschaften der Touchscreen Elemente verändert werden.
Gemeinsam haben alle die Möglichkeit der Veränderung von Größe, Position, Name und Text.
Daneben gibt es noch individuelle Einstellmöglichkeiten.



1.7.2 Button (1)

Mit Text belegbarer Knopf, dessen Status ausgewertet werden kann. Änderung der Textgröße siehe Punkt 5.4.9

1.7.3 Schiebeschalter (2)

Schiebeschalter, dessen Status ausgewertet werden kann.

1.7.4 Kontrollfeld (3)

Checkbutton zur Anzeige, aber auch zur Steuerung von Aktivitäten.

1.7.5 Schieberegler (4)

Für z.B. Servo, Motoren oder Lampenhelligkeit. Regelbereich kann im Inspektor verändert werden.

1.7.6 Label (5)

Einfache Textausgabe. Änderung der Textgröße siehe Punkt 5.4.9

1.7.7 Input (6)

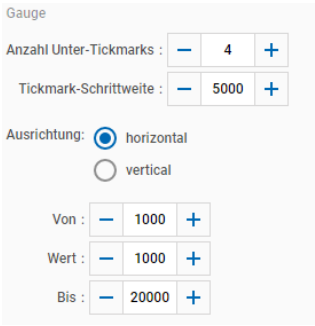
Feld zur Eingabe von Text, der im Programm ausgewertet werden kann.

1.7.8 Status Indikator (7)

Anzeige kann gesetzt bzw. ausgelesen werden

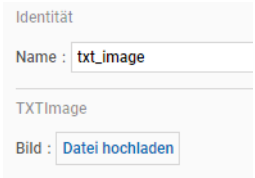
1.7.9 Messinstrument (Gauge) (8)

Zur Anzeige analoger Werte. Der Messbereich kann im Inspektor verändert werden.

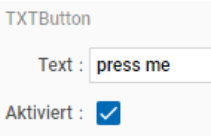


1.7.10 Bilddarstellung (9)

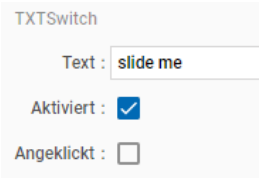
Es besteht die Möglichkeit, Bilder vom Rechner auf der Oberfläche darzustellen.



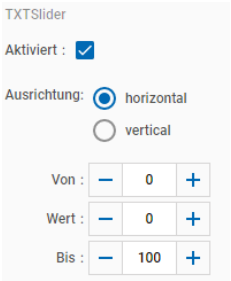
Button



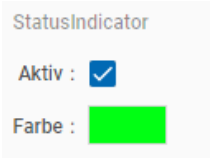
Schiebeschalter



Schieberegler



Status Indikator

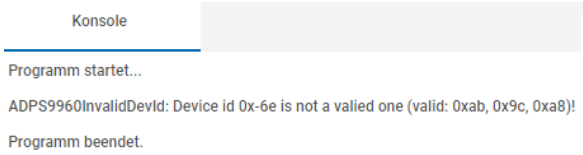


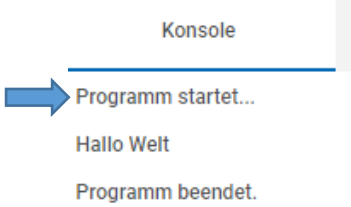
1.8 Bedienfeld

Das nachträglich ins Programm implementierte Bedienfeld wird unter Punkt 6.1 ausführlich beschrieben.

1.9 Konsole (09)

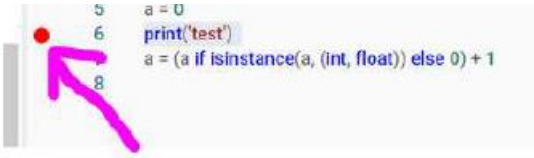
Im Gegensatz zu „Anzeige“, wo Werte auf dem Controller-Display erscheinen, werden hier Informationen auf dem Rechner im Fenster Konsole ausgegeben. Das können zum einen die Info über Start und Ende eines Programmes sein. Aber auch Werte zu Kontrolle oder Fehlermeldungen werden hier angezeigt.





1.10 Haltepunkte (10)

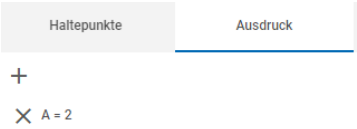
Haltepunkte werden für das Debuggen von Programmen genutzt. Sie definieren bestimmte Stellen im Code, bei denen das Programm angehalten wird.



Durch einen Doppelkick am Anfang des Pythoncodes, kann man einen Haltepunkt setzen. Bei einem Programm durchlauf, im Debugger, wird in dieser Zeile das Programm angehalten.

1.11 Ausdrucke (11)

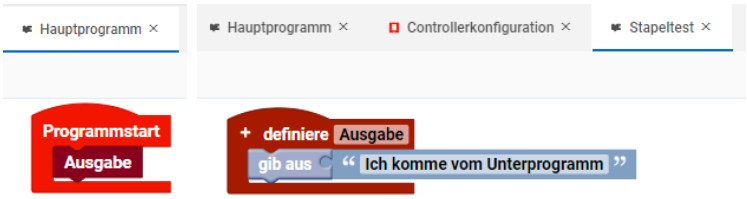
In Reiter „Ausdruck “, kann man über das Pluszeichen Variable hinzufügen, deren Wert angezeigt werden soll. Hier ist es die Variable „A “.

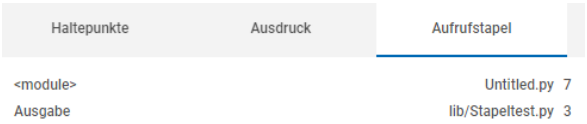


Nach dem Eingeben und drücken der [Enter] Taste wird jeweils der aktuelle Wert beim Debugging angezeigt.

1.12 Aufrufstapel (12)

Im Anrufstapel wird angezeigt, in welchem Programmteil man sich gerade befindet.



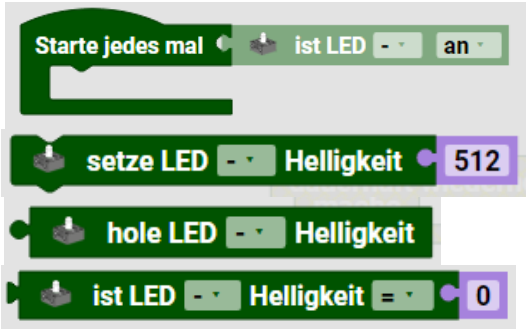


Im Beispiel wird aus dem Hauptprogramm (Untiled.py) das Unterprogramm Stapeltest (lib/Stapeltest.py) aufgerufen.

2 Informationen zum Umgang mit Blöcken

2.1 Arten der Blöcke

Zu einem Objekt (hier eine LED) gibt es verschiedene Blöcke zum Umgang mit dem Wert oder Ereignis. Erkennbar ist dies an der Art der Darstellung auf der linken Seite.



Startblock

Block zum Setzen von Werten

Block zum Lesen von Werten

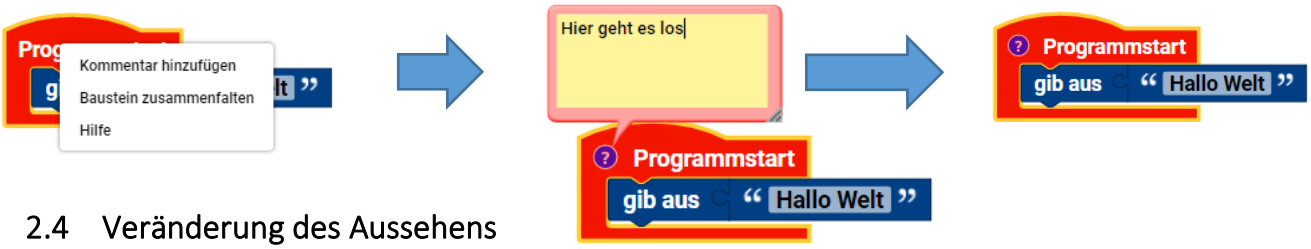
Block zum Vergleichen von Werten

2.2 Block vervielfältigen

Mit Klick der rechten Maustaste auf den Block eröffnet sich die Möglichkeit diesen Baustein zu kopieren oder zu duplizieren. Duplizieren geht schneller, da der neue Block sofort in der Nähe angelegt wird. Kopieren eignet sich besser, wenn der neue Block weit entfernt „Eingefügt“ wird.

2.3 Kommentar hinzufügen

Damit später ein Programm nachvollziehbar ist, gibt es die Möglichkeit, den Blöcken Kommentare zuzufügen. Mit Klick der rechten Maustaste auf den Block kann der Punkt „Kommentare hinzufügen“ benutzt werden.
Durch einen Klick auf das Fragezeichen wird die Kommentarblase geöffnet bzw. geschlossen.

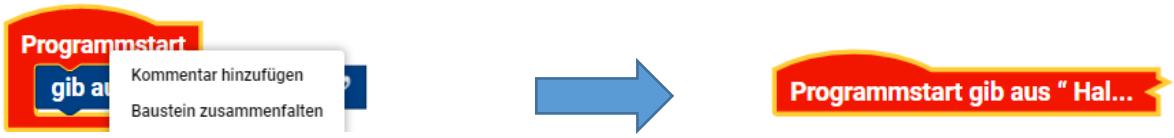


2.4 Veränderung des Aussehens

Es gibt einige Blöcke, bei denen man die Darstellung verändern kann. Mit Klick der rechten Maustaste auf den Block kann der Punkt „externe/interne Eingänge“ benutzt werden.

2.5 Verkleinerte Darstellung

Um die Übersicht der Programme zu verbessern, gibt es die Möglichkeit, die Blöcke verkleinert darzustellen.
Mit Klick der rechten Maustaste auf dem Block kann der Punkt „Baustein zusammenfalten“ benutzt werden.



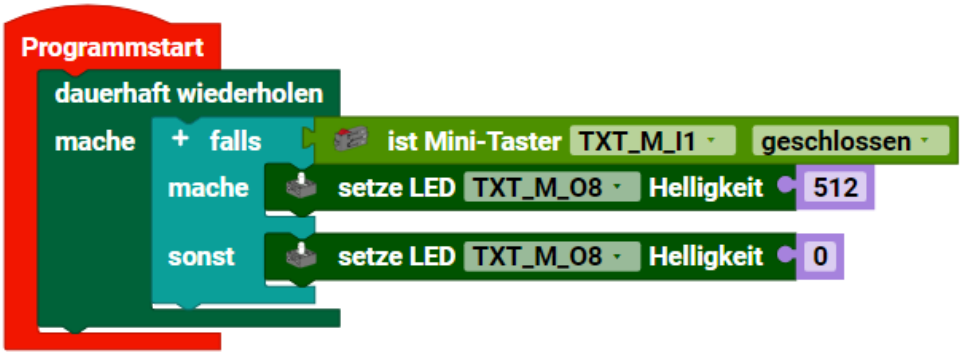
2.6 Baustein löschen

Das Löschen der Blöcke erfolgt ebenfalls im Kontextmenü des Blockes.
Mit Strg und der linken Maustaste kann ein einzelner Block aus dem Programm gelöscht werden.

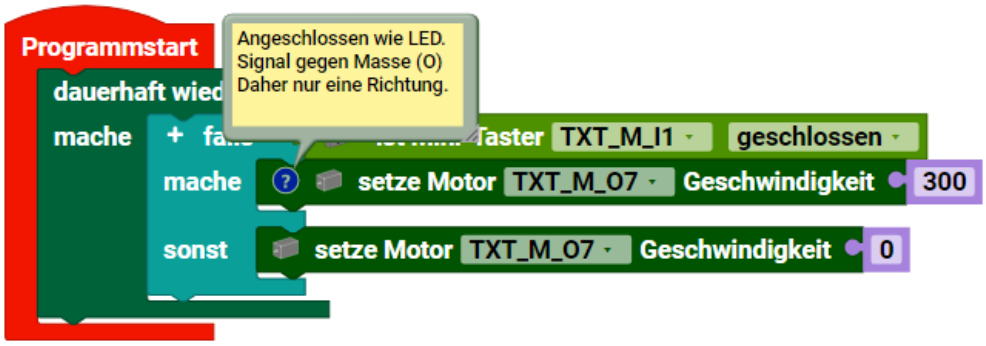
3 Aktoren

3.1 Ausgang

3.1.1 LED



3.1.2 Motor



3.1.3 Magnetventil

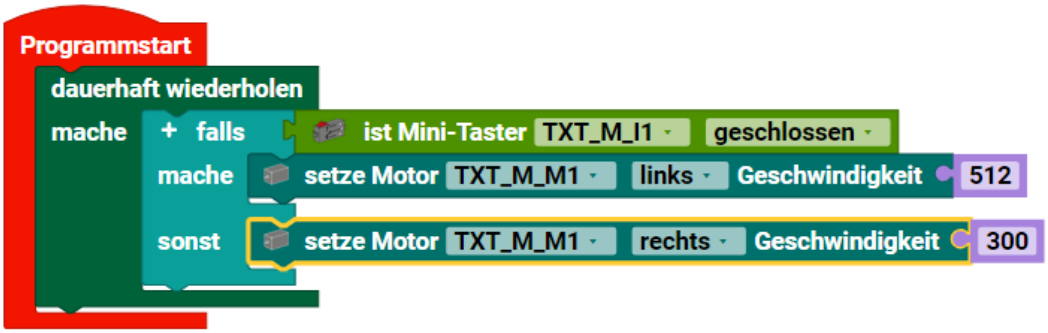
Anschluss wie unidirektionale Motoren.

3.1.4 Kompressor

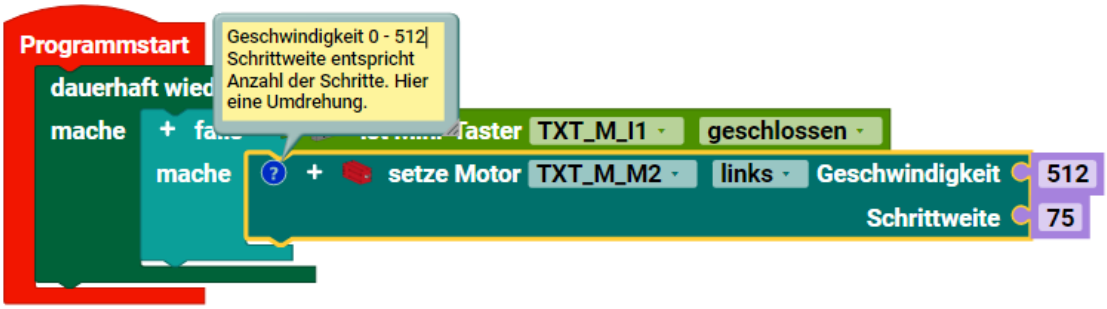
Anschluss wie unidirektionale Motoren.

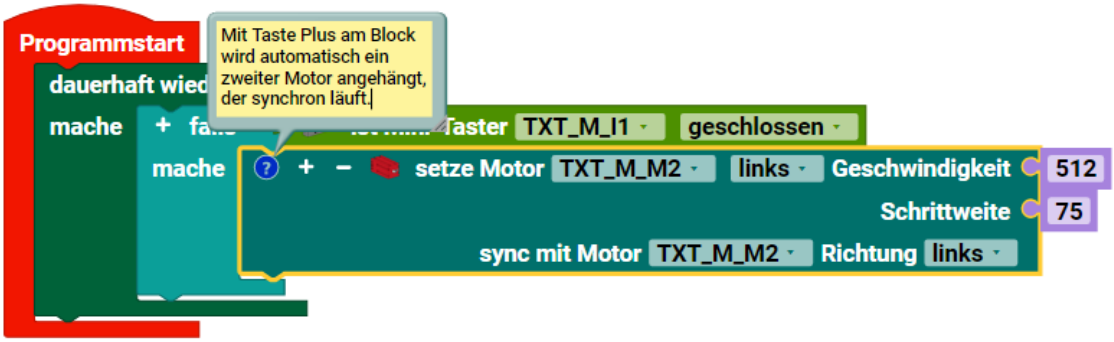
3.2 Motor

3.2.1 Motor (Bidirektional)

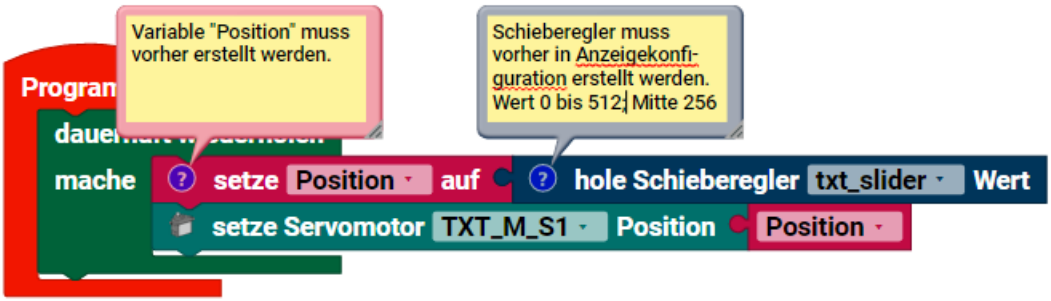


3.2.2 Encoder Motor

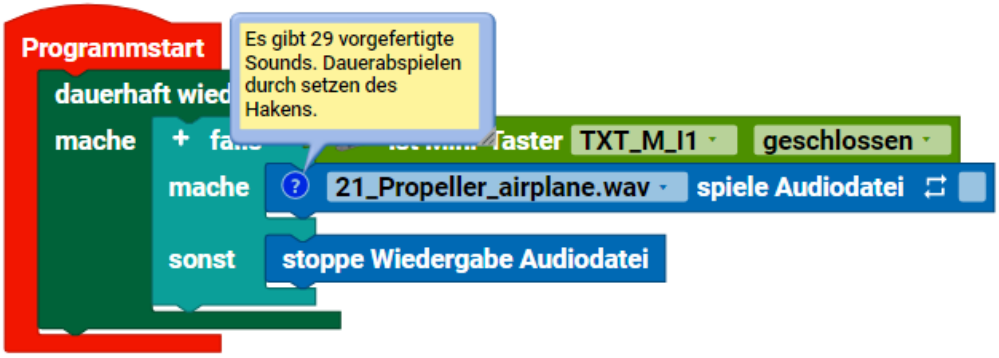


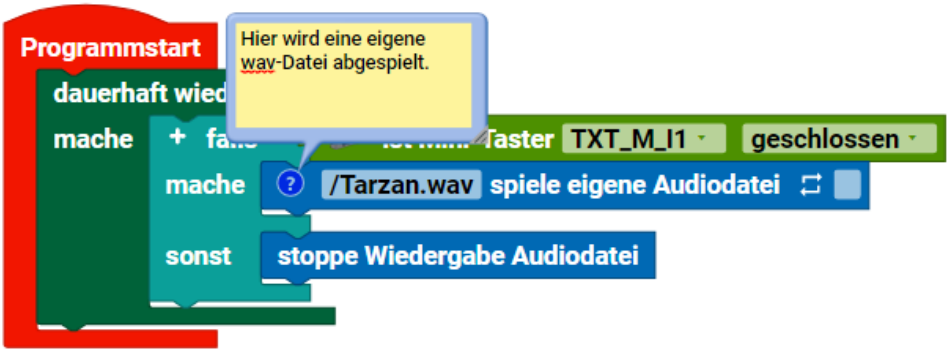


3.2.3 Servomotor

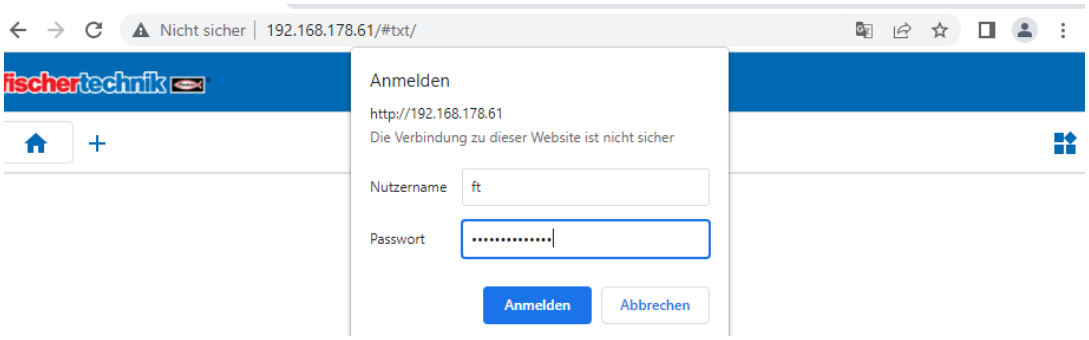


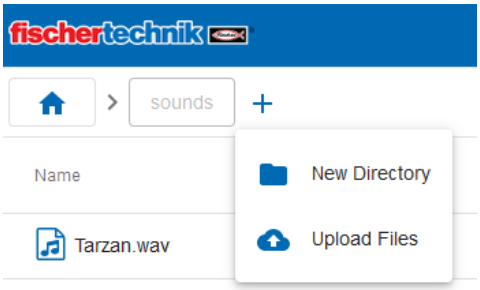
3.3 Sound





Für eigene Wav-Dateien muss man sich mit dem Controller verbinden. Aufruf der IP-Adresse des Controllers. Nutzername: ft und Passwort: fischertechnik eingeben.



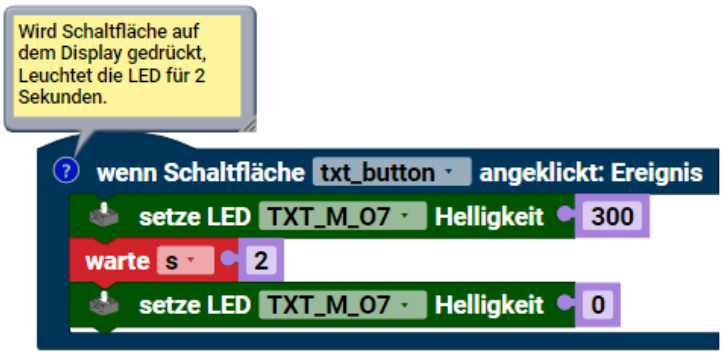


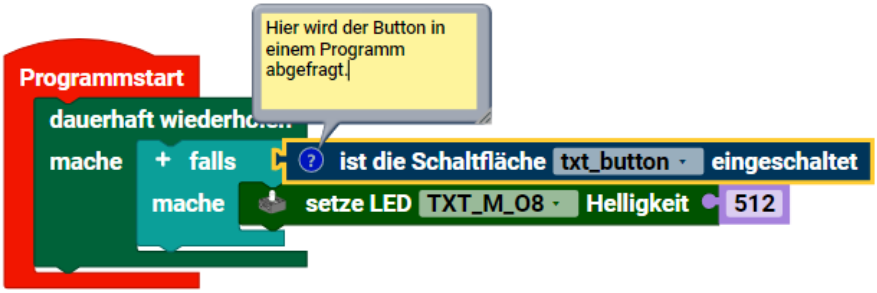
- Den Ordner Sounds bzw. Geräusche öffnen.
- Plustaste drücken.
- Upload Files.
- Add Files und die entsprechende Datei auswählen.
- Upload-Knopf drücken.
- Finish-Knopf drücken
- Abruf siehe Beispiel: /Tarzan.wav

3.4 Anzeige

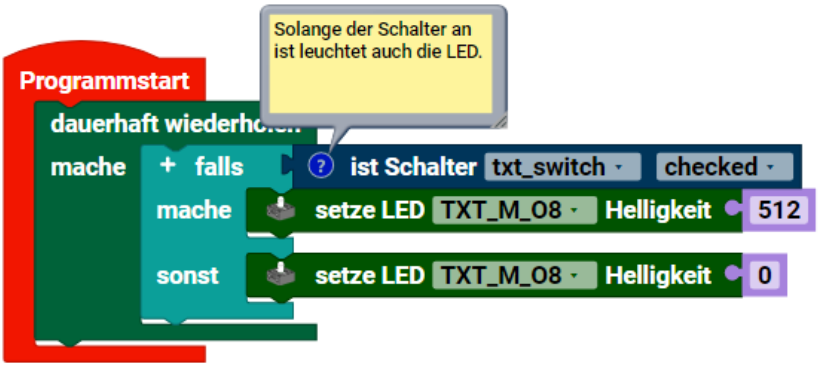
Für das jeweilige Element muss vorher das Objekt im Anzeige Konfigurator erstellt werden. Im Inspektor werden dann die Eigenschaften festgelegt.

3.4.1 Button (txt_button)

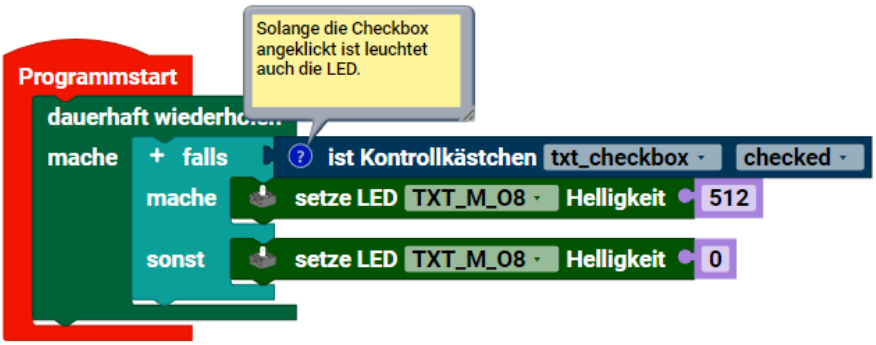




3.4.2 Schalter (txt_switch)

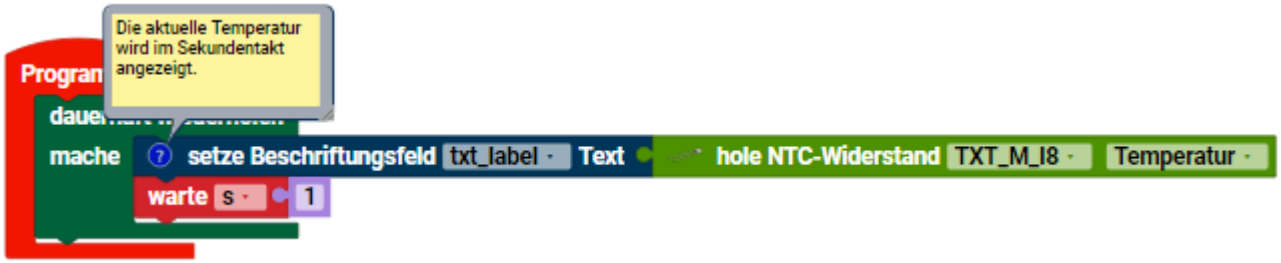


3.4.3 Kontrollfeld (txt_checkbox)

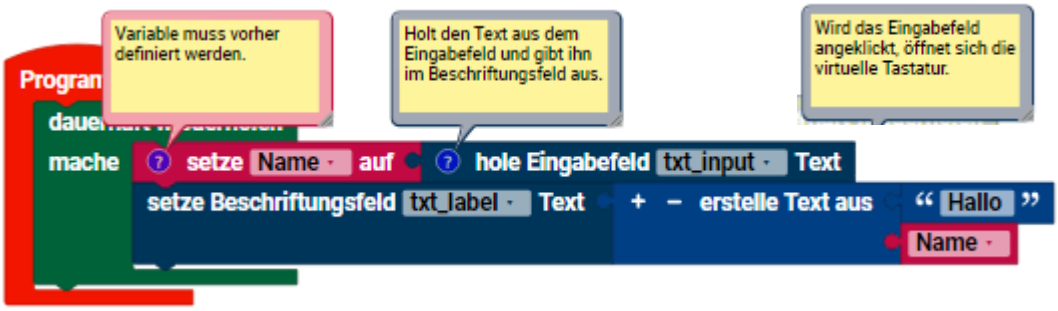


3.4.4 Schieberegler (txt_slider)

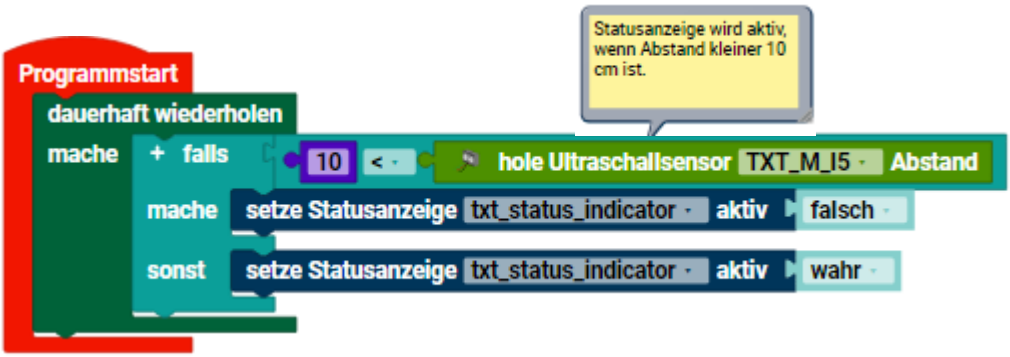
3.4.5 Beschriftungsfeld (txt_label)



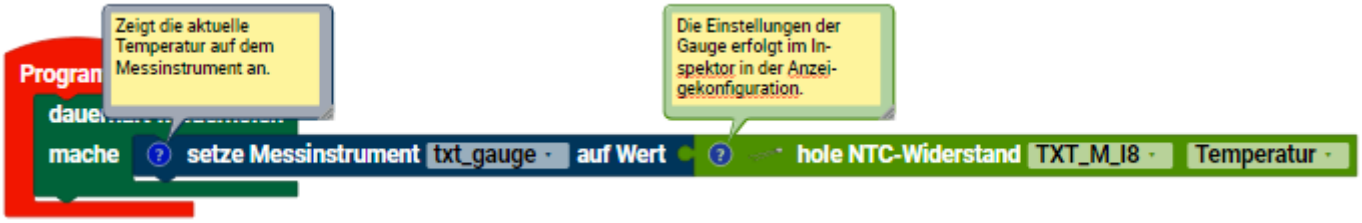
3.4.6 Eingabefeld (txt_input)



3.4.7 Statusanzeige (txt_status_indicator)



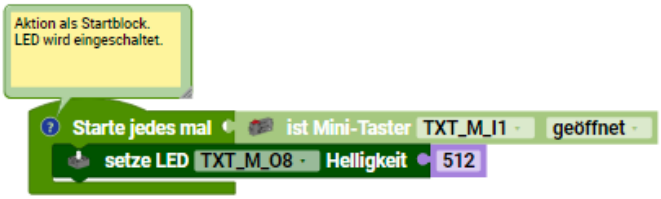
3.4.8 Messinstrument (txt_gauge)



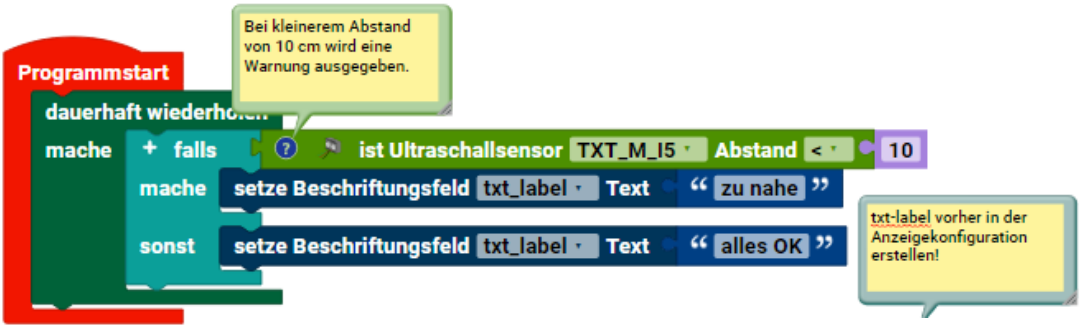
4 Sensoren

4.1 Eingang

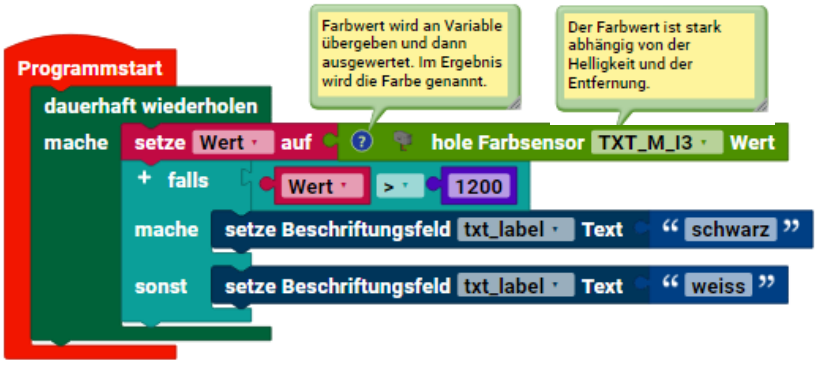
4.1.1 Taster



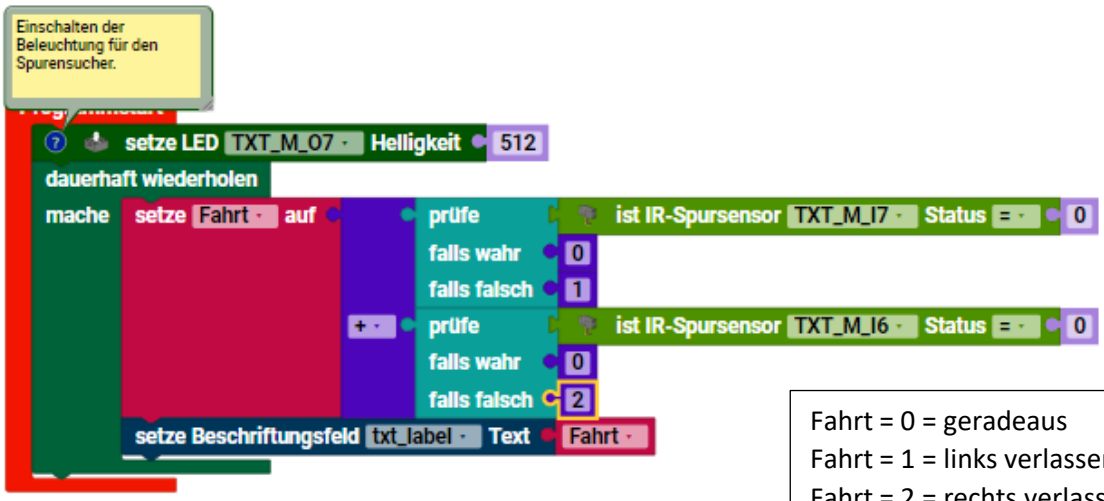
4.1.2 Ultraschallsensor



4.1.3 Farbsensor



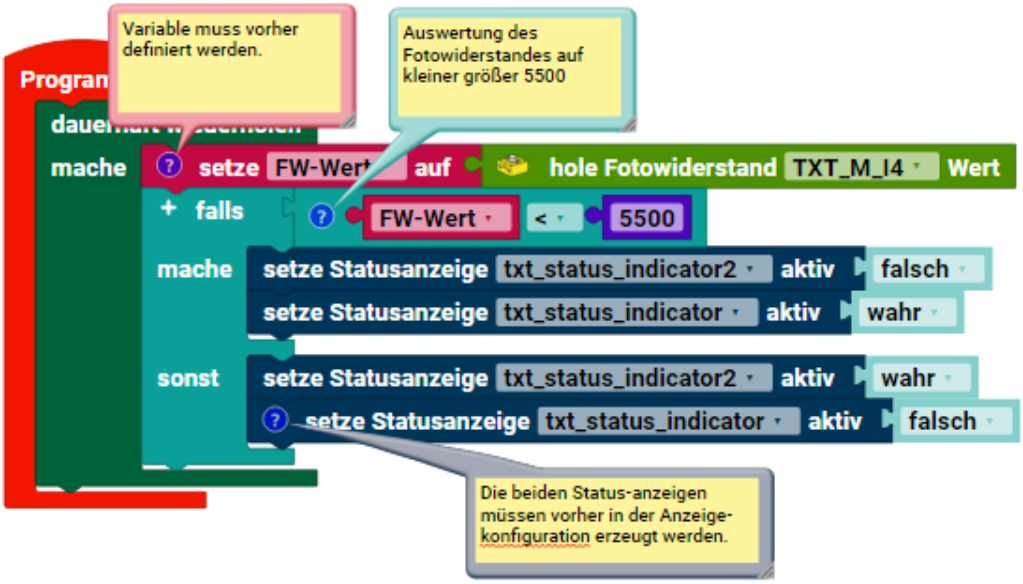
4.1.4 Spurensucher



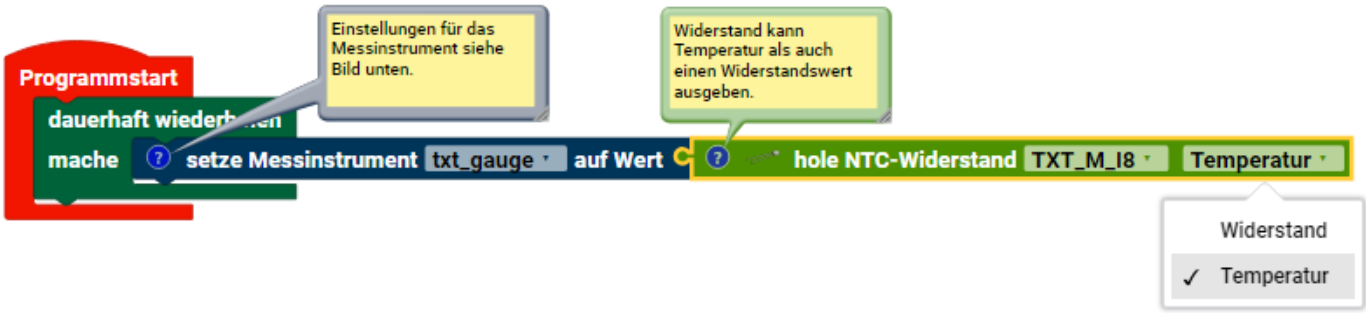
Fahrt = 0 = geradeaus
Fahrt = 1 = links verlassen
Fahrt = 2 = rechts verlassen
Fahrt = 3 = Spur verloren

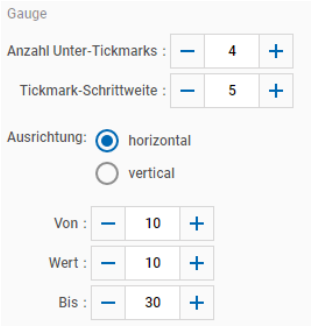
4.1.5 Fototransistor

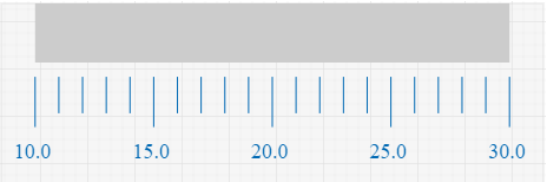
4.1.6 Fotowiderstand



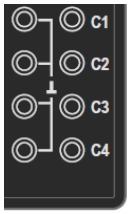
4.1.7 NTC Widerstand





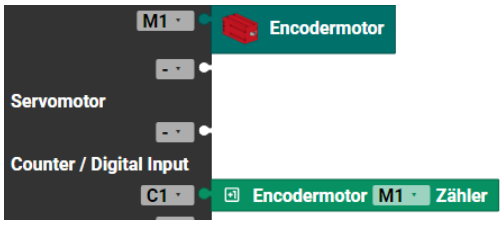


4.2 Zähler



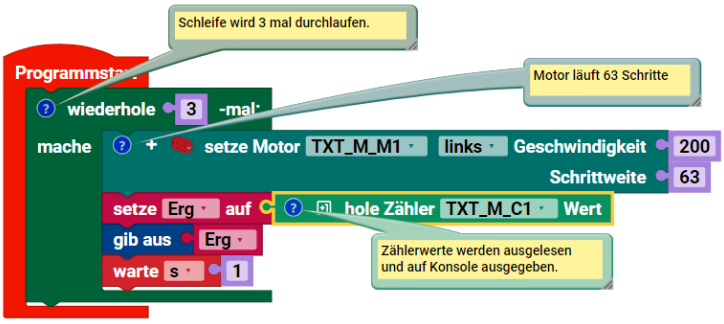
Impulse können auf zwei verschiedene Arten gezählt werden:
- Die Auswertung erfolgt über einen normalen I-Eingang
- Bei schnellen Impulsen würde ein C-Eingang (C1 – C4) besser funktionieren.
- Hauptsächlich gedacht sind die C-Eingänge für Encodermotoren, die über 3 zusätzliche Anschlüsse verfügen (Plus, Minus, Zählausgang)

Bei den aktuellen Motoren beträgt eine Umdrehung 63,3 Impulse.(Alt-63,9)

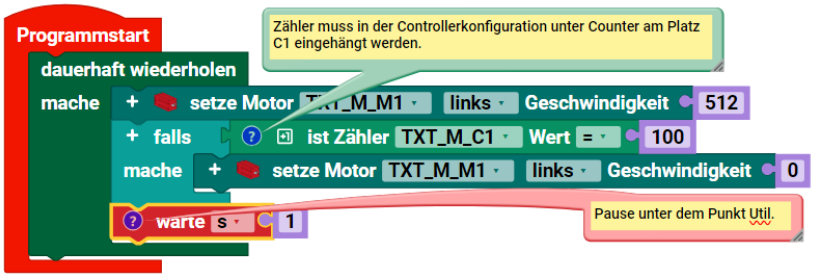


Um die Zählimpulse eines Motors auszuwerten, muss in der Controllerkonfiguration der Motor und der entsprechende Zähleingang am Controller angemeldet sein.

4.2.1 Hole Zählerwert

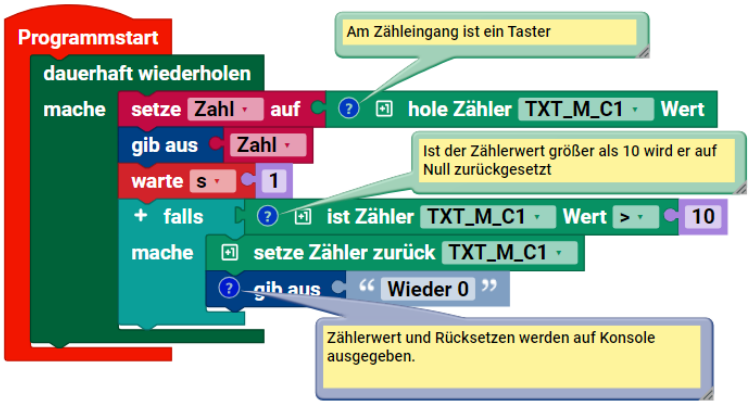


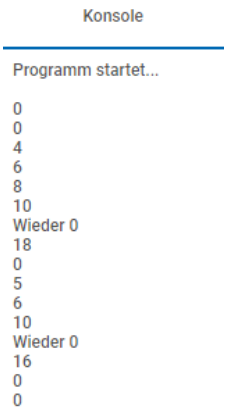
4.2.2 Ist Zähler … Wert



Ist die Zahl größer 100 wird der Motor Angehalten. Nach einer Pause startet der Motor erneut.

4.2.3 Setze Zähler zurück





4.3 I²C

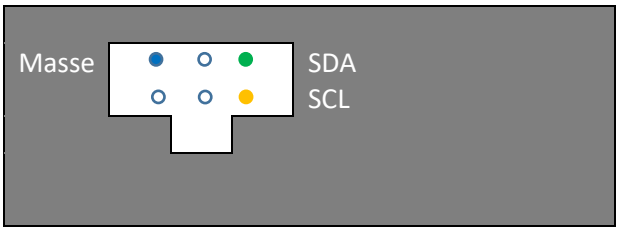
Hier können die 3 Sensoren von Fischertechnik angeschlossen werden. Das sind zurzeit:

2 mal 10 Polig für TXT	1 mal 6 polig für TXT 4.0	Beschreibung
 158402	 201257	**Kombisensor mit** Gyroskop Beschleunigungssensor Kompasssensor
 167358	 182974	**Umweltsensor mit** Temperatur Luftfeuchtigkeit Luftdruck Luftqualität
 186705	 183267	**Gestensensor mit** Dämmerungsschalter Farberkennung Licht und Nähe Gestensteuerung

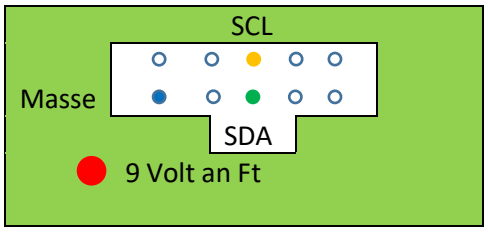
Es gibt die Sensoren teilweise für den alten TXT bzw. für den neuen TXT 4.0 Sie sind vom Anschluss nicht Kompatibel. Dafür bietet FT einen Adapter für über 20 Euro an.

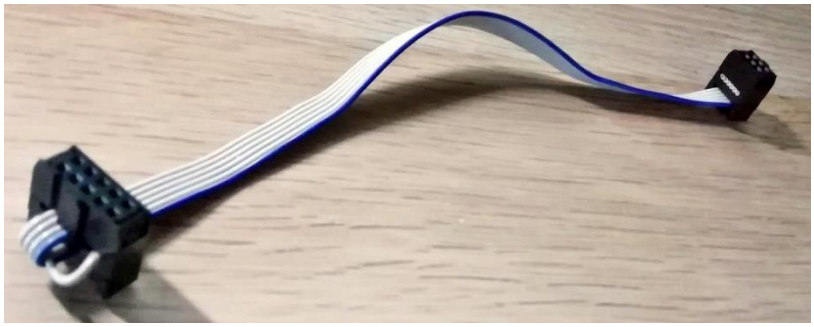
Alternativ braucht man nur ein Kabel im Eigenbau mit folgender Belegung:

Ft TXT 4.0 Ansicht auf Anschlussbuchse

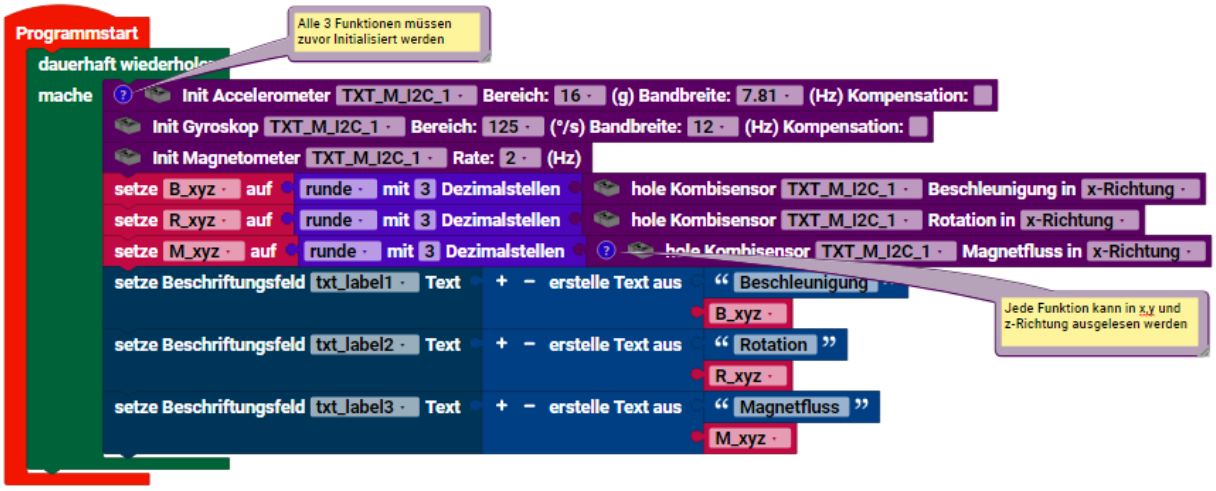


Ft Umweltsensor Ansicht auf Anschlussbuchse

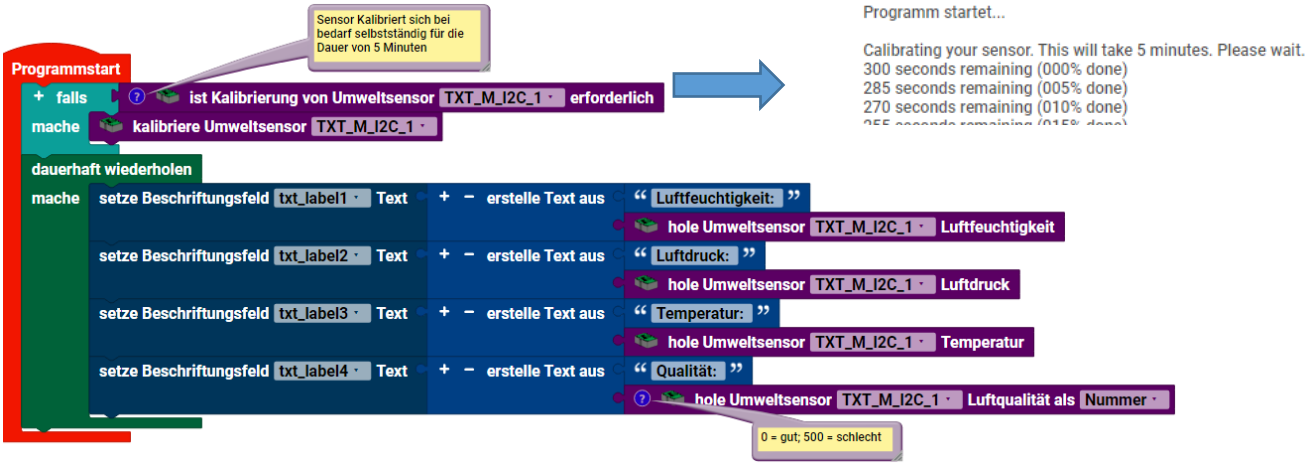




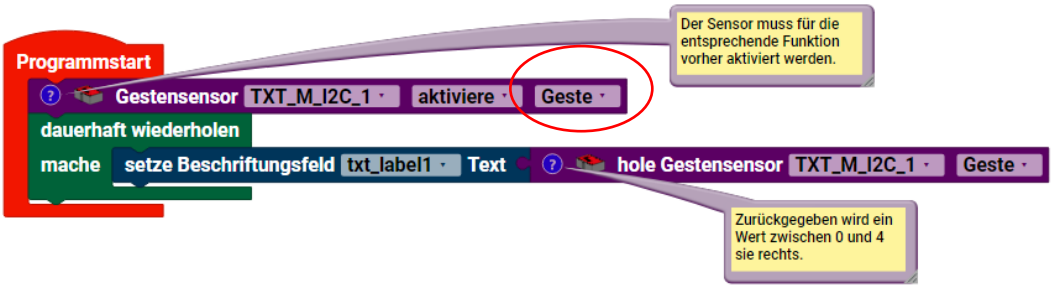
4.3.1 Kombisensor

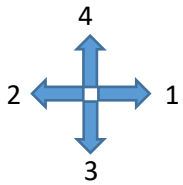


4.3.2 Umweltsensor



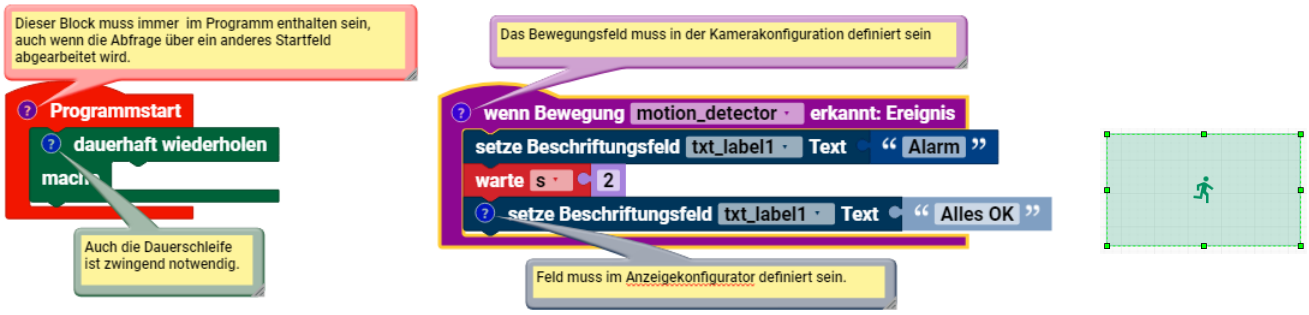
4.3.3 Gestensensor



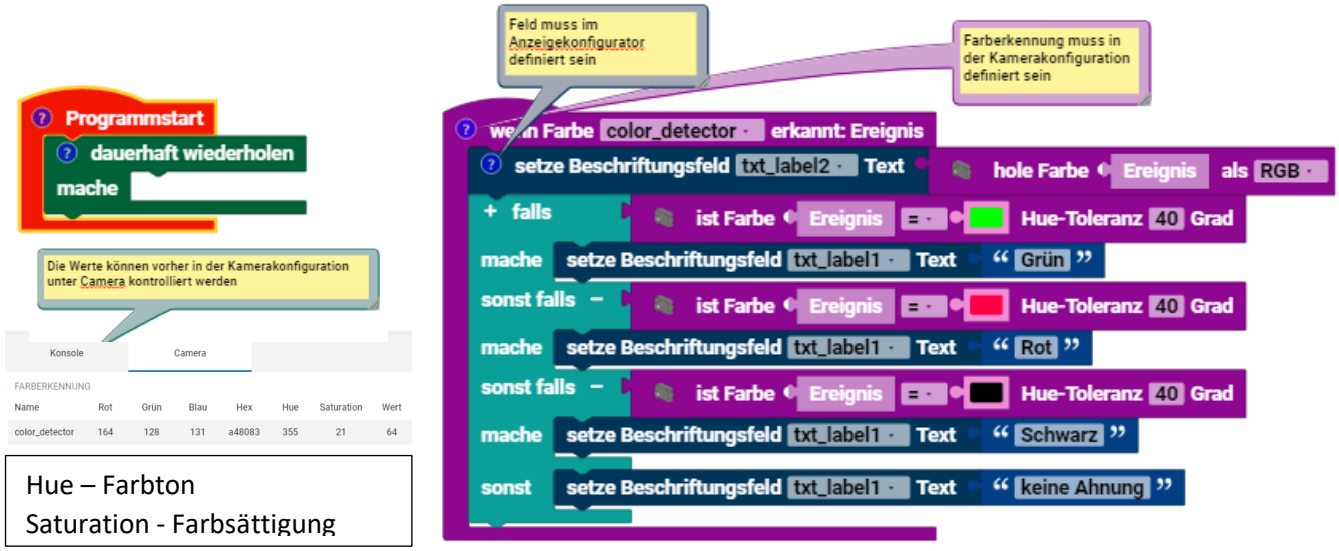


4.4 USB (Kamera)

4.4.1 Bewegungserkennung

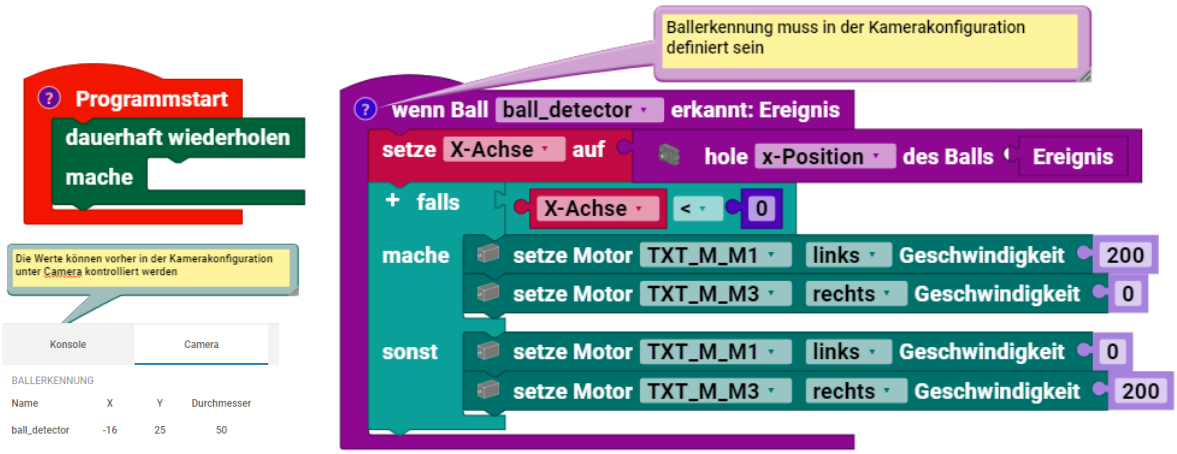


4.4.2 Farberkennung

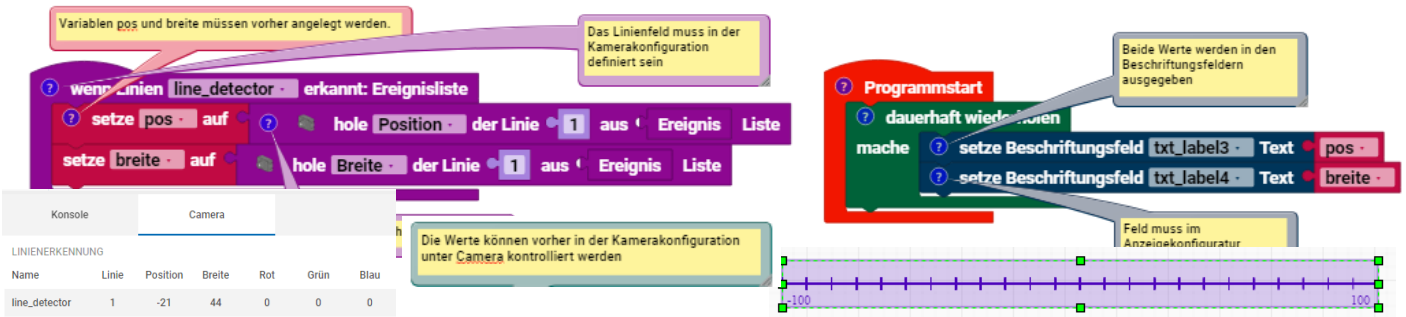


Hue – Farbton
Saturation - Farbsättigung

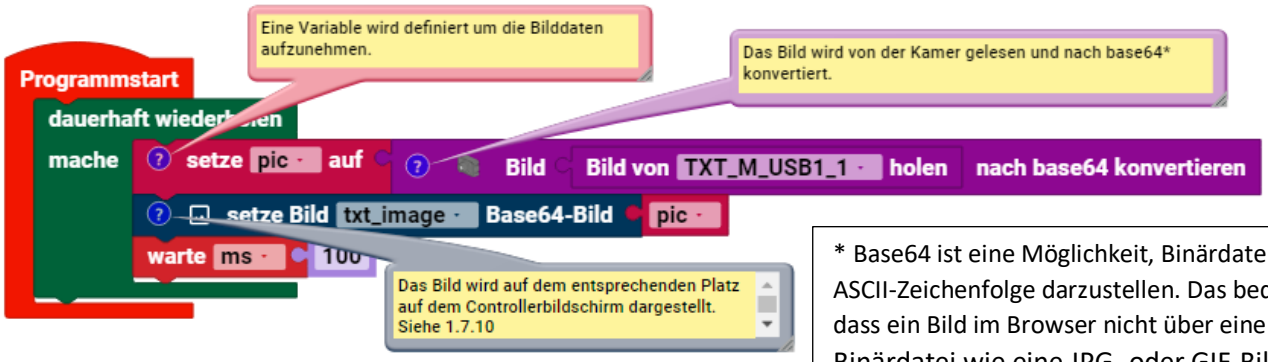
4.4.3 Ballerkennung



4.4.4 Linienerkennung

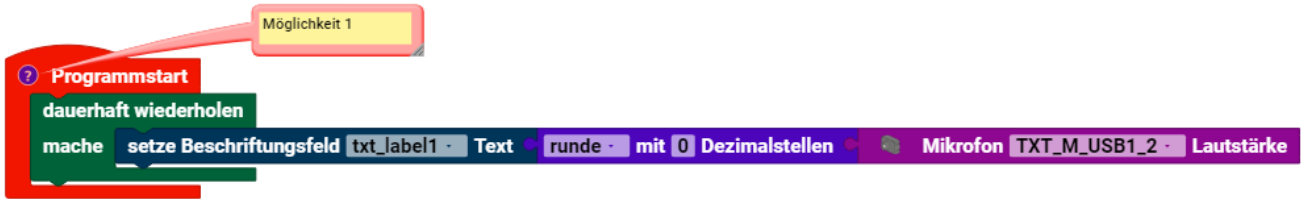


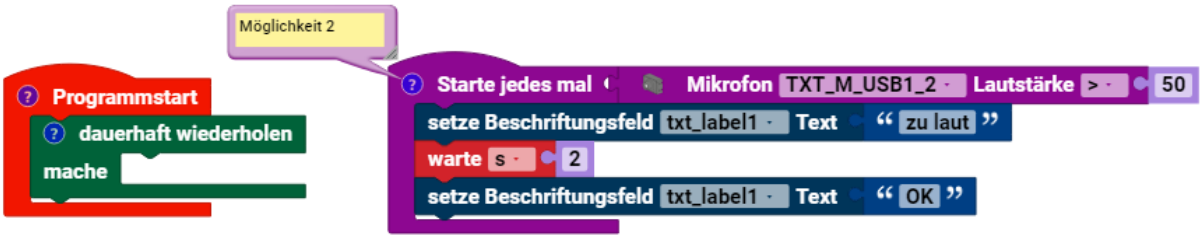
4.4.5 Bilddarstellung auf Controller



* Base64 ist eine Möglichkeit, Binärdaten in einer ASCII-Zeichenfolge darzustellen. Das bedeutet, dass ein Bild im Browser nicht über eine Binärdatei wie eine JPG- oder GIF-Bild eingebunden wird, sondern der Browser aus einer Zeichenfolge das Bild direkt berechnet.

4.4.6 Mikrofon





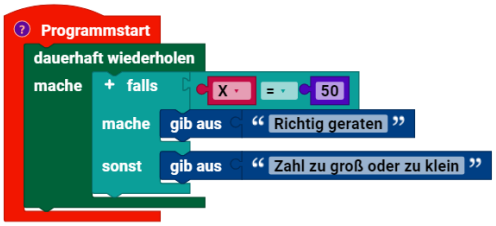
5 Verarbeitung

5.1 Logik

5.1.1 Boolesche Logik

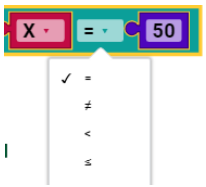
Einfaches mathematisches System, das zwei Werte hat, um Bedingungen und Schleifen zu kontrollieren. (wahr,falsch)

5.1.2 Wenn Funktion



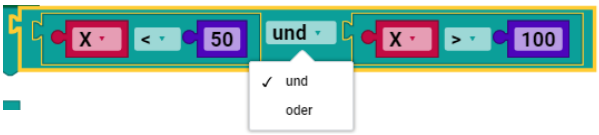
Es ist eine der häufigsten verwendeten Funktionen. Sie ermöglicht den logischen Vergleich zwischen einem aktuellen Wert und einem erwarteten Wert.

5.1.3 Vergleichsoperatoren



Jedem der 6 möglichen Vergleichsoperatoren werden 2 Eingänge übergeben und der Vergleich gibt wahr oder falsch zurück.

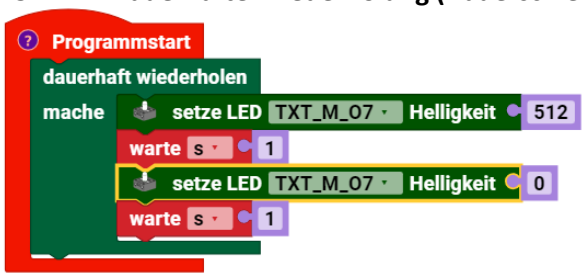
5.1.4 Logische Operatoren



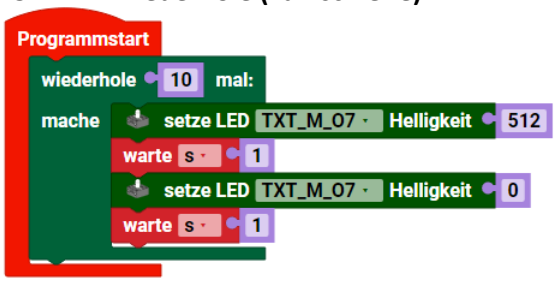
Der **und**-Block gibt dann und nur dann **wahr** zurück, wenn seine beiden Eingangswerte wahr sind.
Der **oder**-Block gibt **wahr** zurück, wenn mindestens einer seiner beiden Eingangswerte wahr ist.

5.2 Schleifen

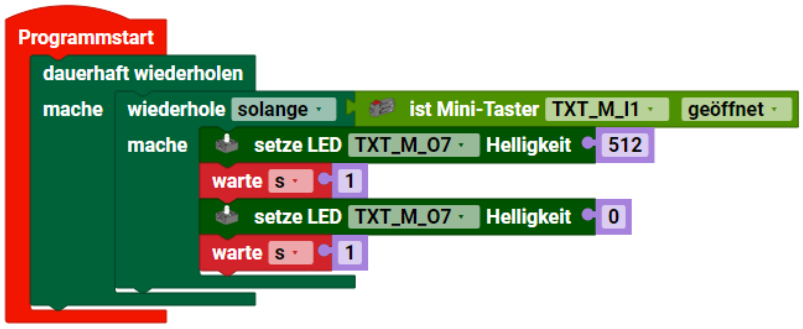
5.2.1 Dauerhafte Wiederholung (Dauerschleife)



5.2.2 Wiederhole (Zählschleife)

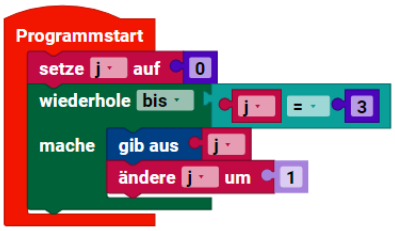


5.2.3 Wiederhole solange (Kopfschleife)

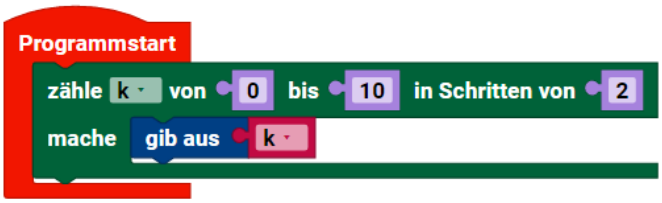


5.2.4 Wiederhole bis (Fußschleife)

Schleife wird mindestens 1 mal durchlaufen



5.2.5 Zählschleife von bis (For-Schleife)

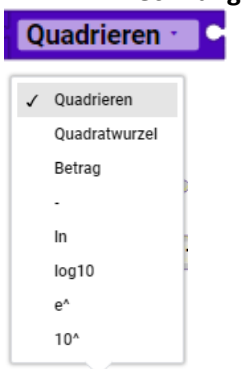


Die Variable „k“ wird wie folgt ausgegeben:
0, 2, 4, 6, 8, 10

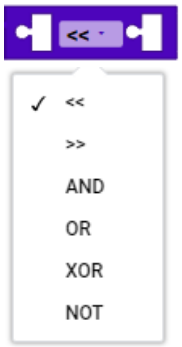
5.3 Mathe

5.3.1 Einfache Rechnungen

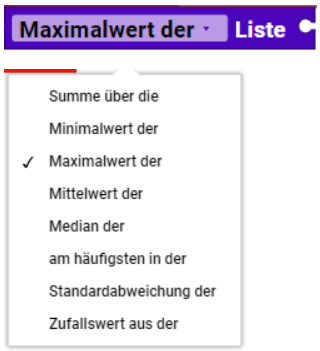
5.3.2 Spezielle Rechnungen



5.3.3 Operatoren *1

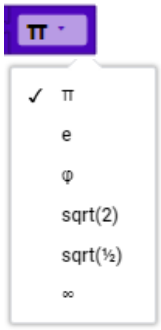


5.3.4 Auswertung von Listen *2

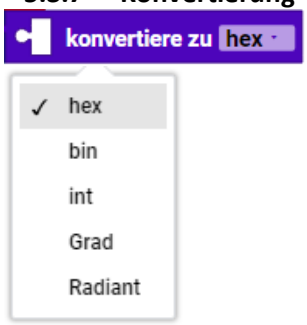


5.3.5 Funktionen

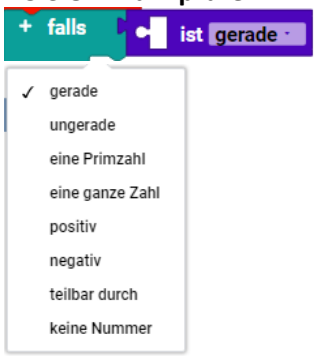
5.3.6 Konstanten



5.3.7 Konvertierung



5.3.8 Zahl prüfen



*1 Die „Bitweisen Operatoren“ sind für Ganzzahlen, die in das Binärformat umgewandelt werden und dann auf Bit Ebene je nach Operator „behandelt“ werden.

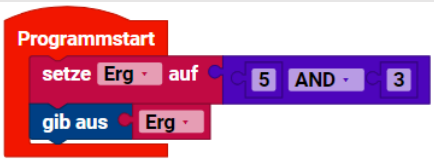
Bitzahl nach links schieben

Die Zahl 2 (0010) wird zweimal nach links verschoben
0010 = 2 (Ausgangswert)
0100 = 4 (erste Verschiebung)
1000 = ***8*** (zweite Verschiebung)

Bitzahl nach rechts schieben

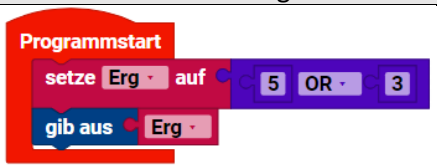
Die Zahl 9 (1001) wird zweimal nach rechts verschoben
1001 = 9 (Ausgangswert)
0100 = 4 (erste Verschiebung)
0010 = ***2*** (zweite Verschiebung)

Bitzahlen mit UND vergleichen



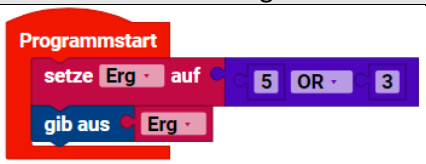
Potenz	2^2	2^1	2^0
Wert	4	2	1
5	1	0	1
3	0	1	1
1	0	0	1

Bitzahlen mit ODER vergleichen



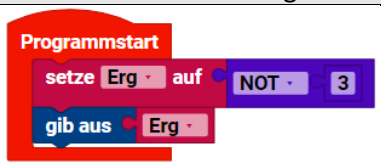
Potenz	2^2	2^1	2^0
Wert	4	2	1
5	1	0	1
3	0	1	1
7	1	1	1

Bitzahlen mit XOR vergleichen



Potenz	2^2	2^1	2^0
Wert	4	2	1
5	1	0	1
3	0	1	1
6	1	1	0

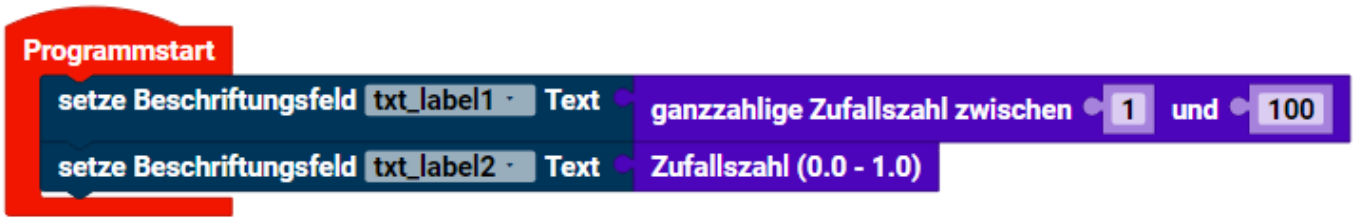
Bitzahlen mit NICHT vergleichen



Potenz	2^2	2^1	2^0
Wert	4	2	1
3	0	1	1
-4	1	0	0

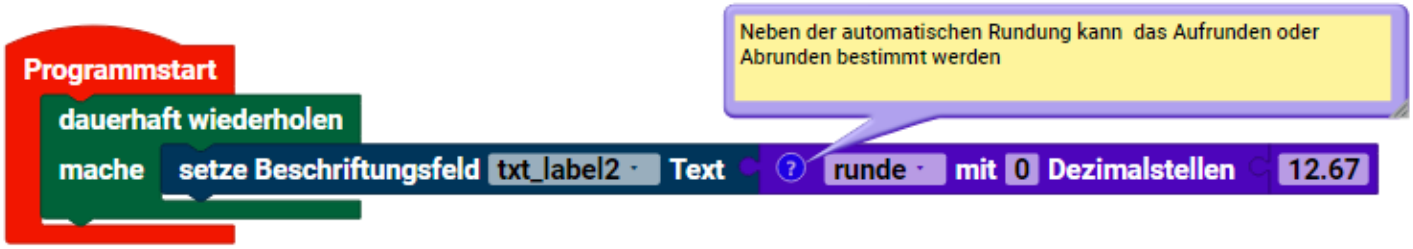
*2 Voraussetzung für 5.3.4 arbeiten mit Listen

5.3.9 Zufallszahlen

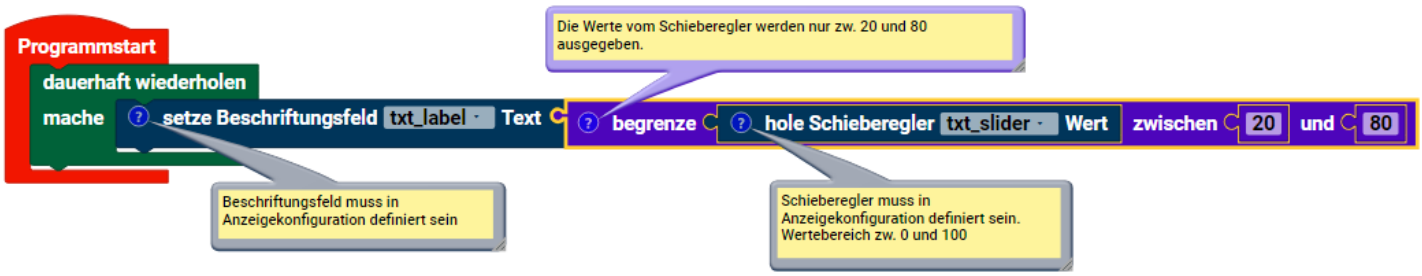


5.3.10 Rest einer Division

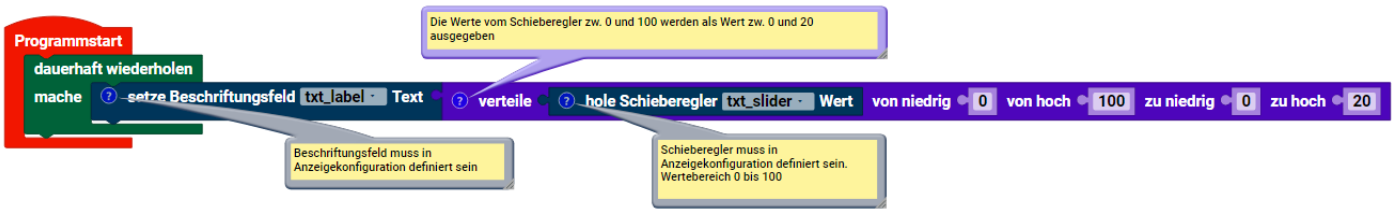
5.3.11 Zahl runden



5.3.12 Werte beschränken



5.3.13 Wertebereich wandeln



5.3.14 Atan2

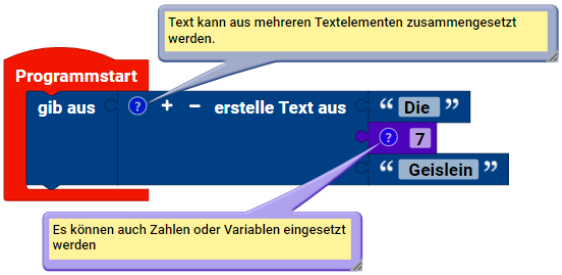
Mit der Funktion wird der Arkustangens (auf Basis von XY-Koordinaten) der Pixel in einem Raster berechnet.

5.4 Text

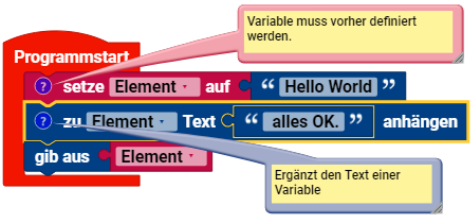
Die Ausgabe von Texten erfolgt immer in der Konsole unter dem Quelltext. Sie können aber auch auf dem Touchfeld des Kontrollers angezeigt werden. Dazu muss ein entsprechendes Feld (Beschriftungsfeld, Eingabefeld) in der Anzeigekonfiguration definiert werden.

5.4.1 Textausgabe

Konsole
Programm startet...
Hallo Welt
Programm beendet.



Konsole
Programm startet...
Die 7 Geislein
Programm beendet.



Kons
Programm startet...
Hello World alles OK.
Programm beendet.

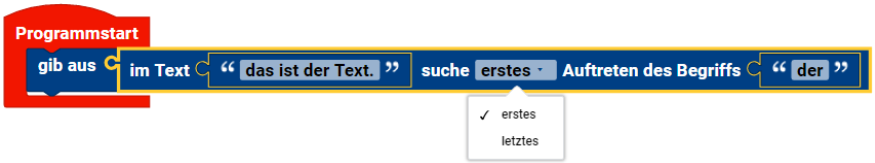
5.4.2 Textlänge ermitteln

Konsole
Programm startet...
24
Programm beendet.

5.4.3 Prüfen auf leeren Text

Konsole
Programm startet...
Kein Text vorhanden
Programm beendet.

5.4.4 Suche im Text

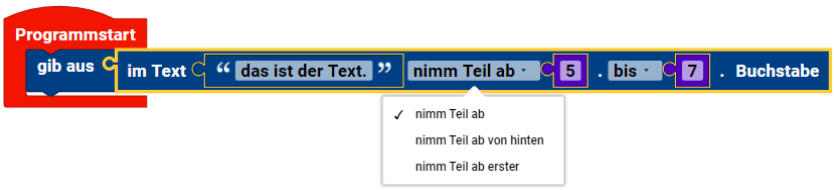


Konsole
Programm startet...
9
Programm beendet.

5.4.5 Extrahieren einzelner Zeichen

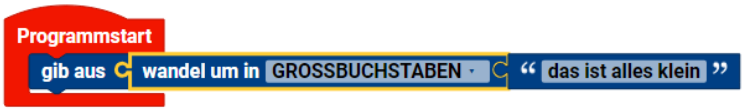
Konsole
Programm startet...
s
Programm beendet.

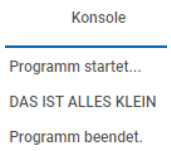
5.4.6 Extrahieren eines Textteils



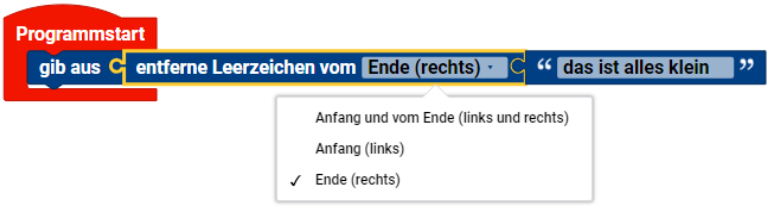
Konsole
Programm startet...
ist
Programm beendet.

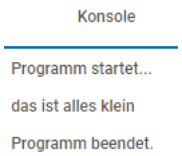
5.4.7 Groß und Kleinschreibung ändern





5.4.8 Leerzeichen entfernen (vorn bzw. hinten)

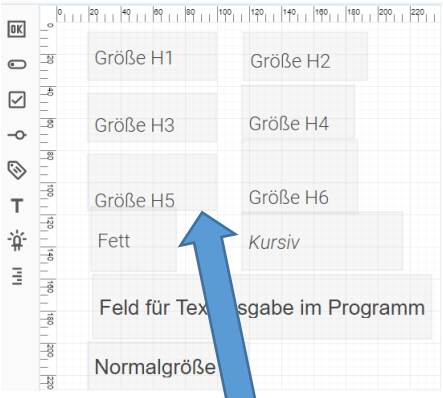


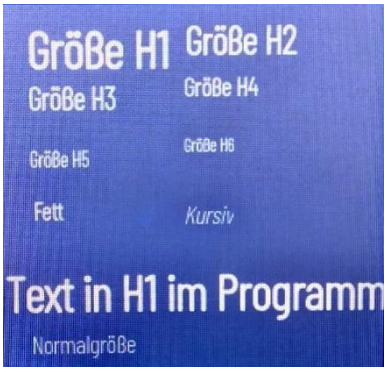


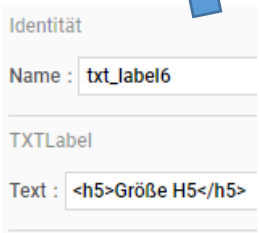
5.4.9 Schriftgöße ändern

Wie bei der Syntax in HTML, können Texte formatiert ausgegeben werden. Die Formatierung kann entsprechend der Anwendung in der Anzeigekonfiguration oder im Programm bei der Textausgabe durchgeführt werden. Die Größe wird mit <h1> bis <h6> eingeleitet und mit </h1> bis </h6> abgeschlossen.

Weitere Formatierungen wären **fett** mit <b>, *kursiv* mit <i>und <u>unterstrichen</u> mit <u>.

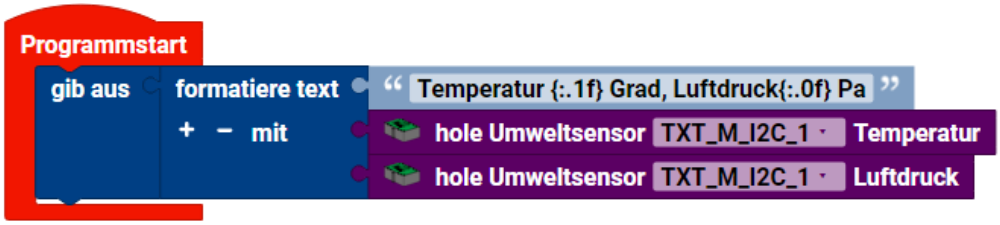


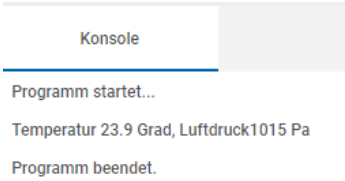




5.4.10 Texte formatieren

Gerade bei der Ausgabe von Daten, ist es zweckmäßig, eine Mischung aus Text und Daten formatiert auszugeben.

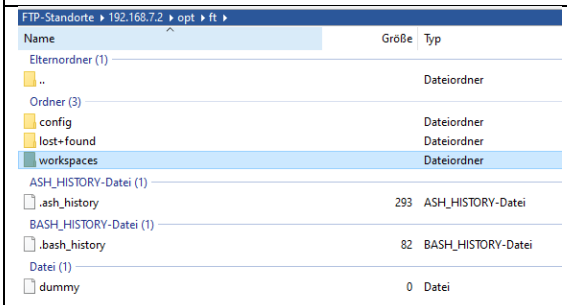
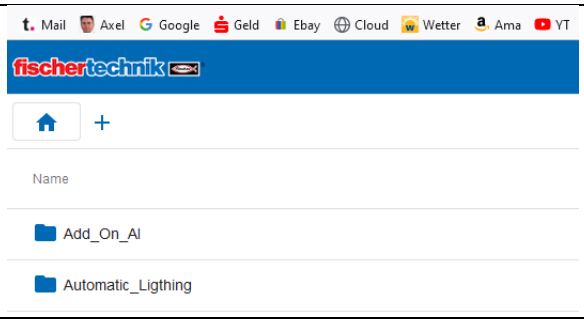




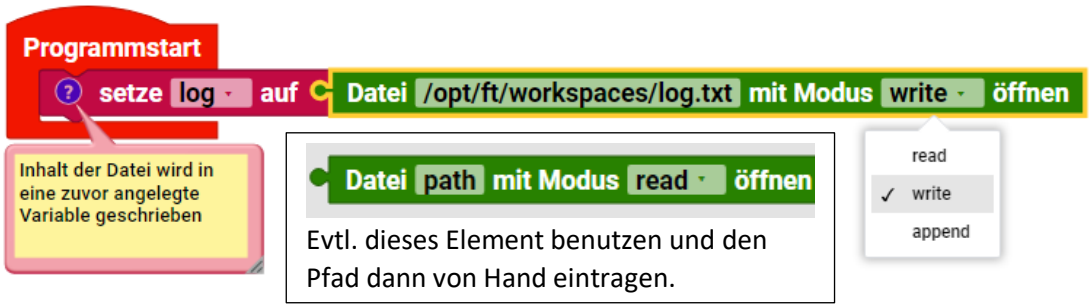
In jede geschweifte Klammer wird der folgende Wert aus dem Block“ formatiere text“ ausgegeben.
Hierbei können auch die Anzahl der Nachkommastellen mit „:.xf“ festgelegt werden. (x = Anzahl der Nachkommastellen)

5.5 Datei

Bei Arbeiten mit Dateien, werden diese nur auf dem TXT 4.0 gespeichert. Dabei ist der Pfad anfänglich vorgegeben: /opt/ft/workspaces/log.txt , um keine Systemdateien zu manipulieren.
Um die Dateien auslesen zu können, gibt es 2 Möglichkeiten:

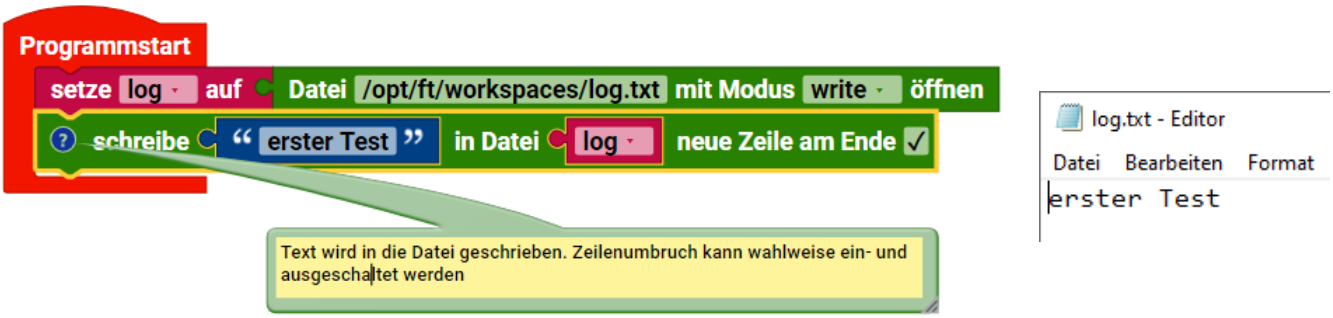
Mit einem FTP-Programm Server: 192.168.7.2 (Bei USB) Port: 22 Benutzername: ft Passwort: fischertechnik	Mit dem Browser URL: 192.168.7.2 (Bei USB-Verbindung) Benutzername: ft Passwort: fischertechnik
	

5.5.1 Datei … mit Modus… öffnen

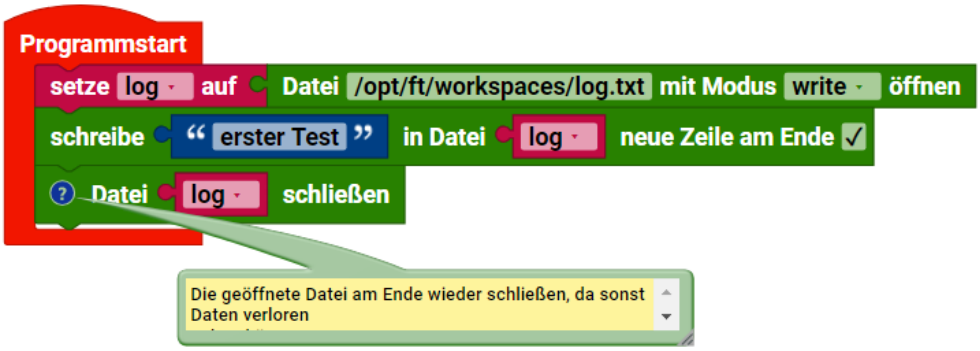


Sollte die Datei noch nicht existieren, wird sie hier automatisch angelegt.
Der Modus kann auf: lesen, schreiben oder anhängen gestellt werden.

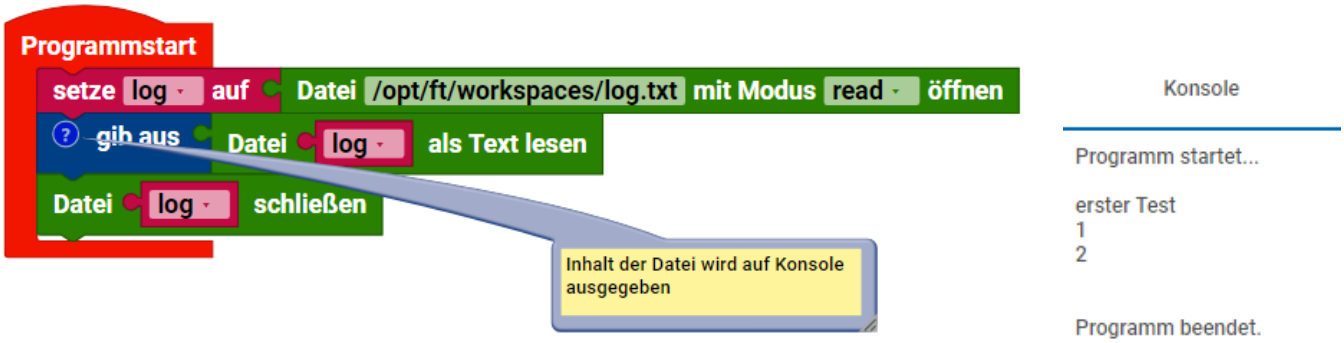
5.5.2 schreibe… in Datei… neue Zeile am Ende



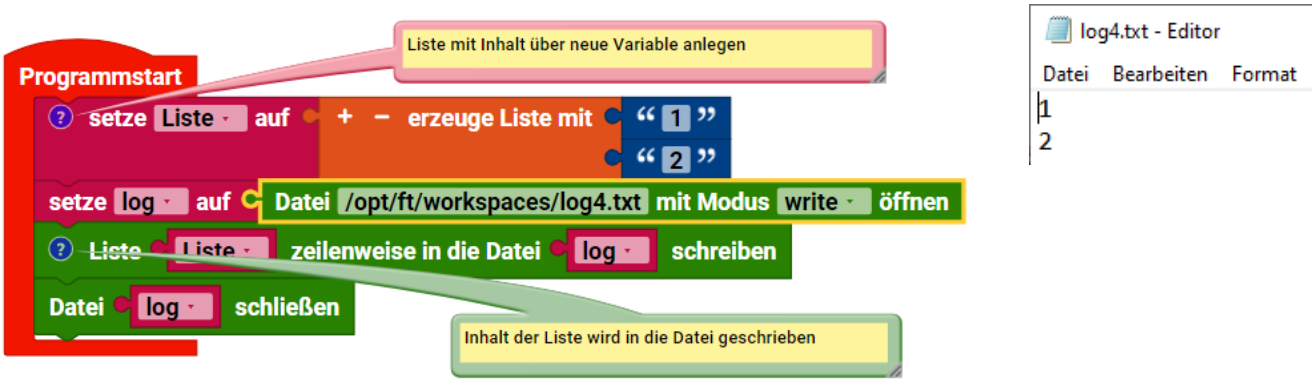
5.5.3 Datei … schließen



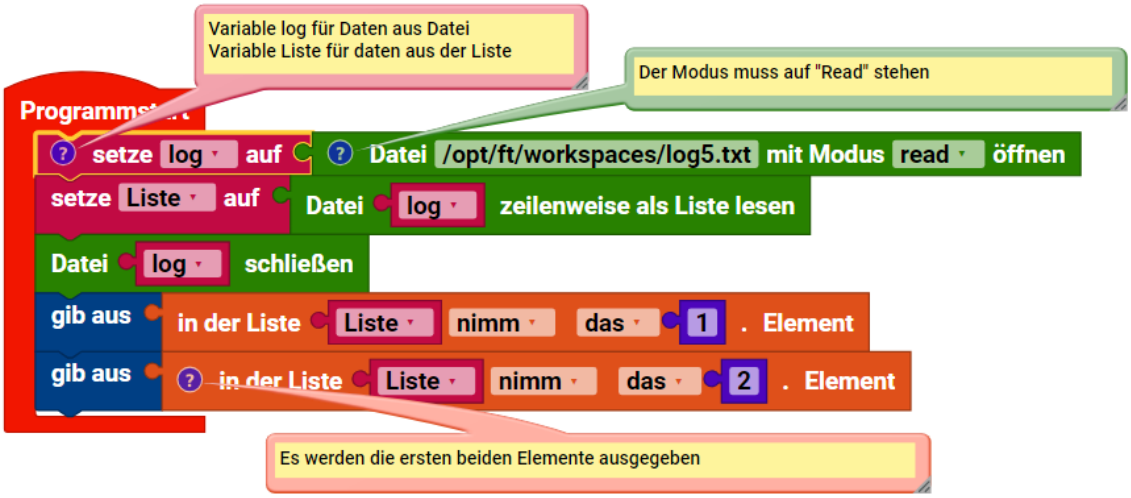
5.5.4 Datei… als Text lesen

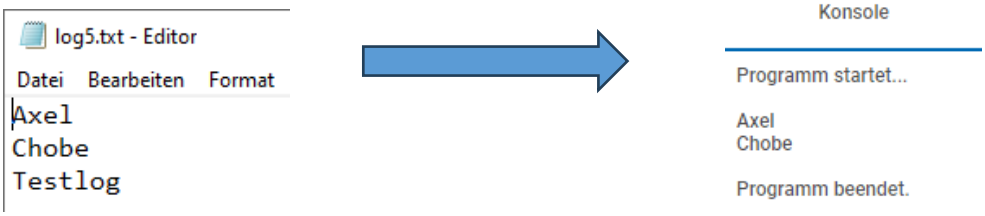


5.5.5 Liste… zeilenweise in die Datei … schreiben

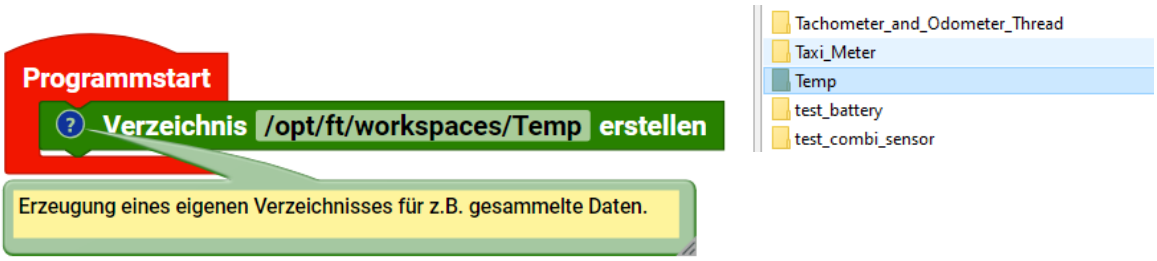


5.5.6 Datei … zeilenweise als Liste lesen

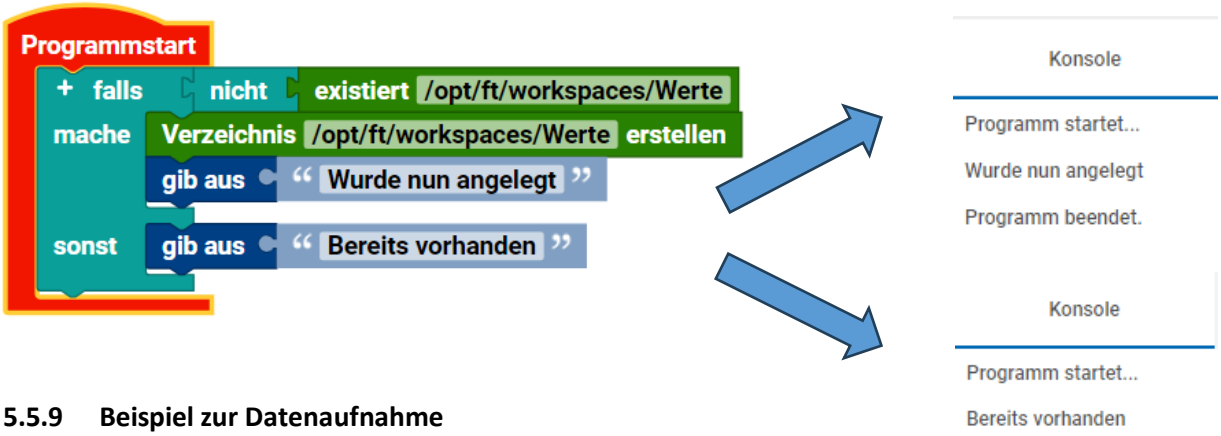




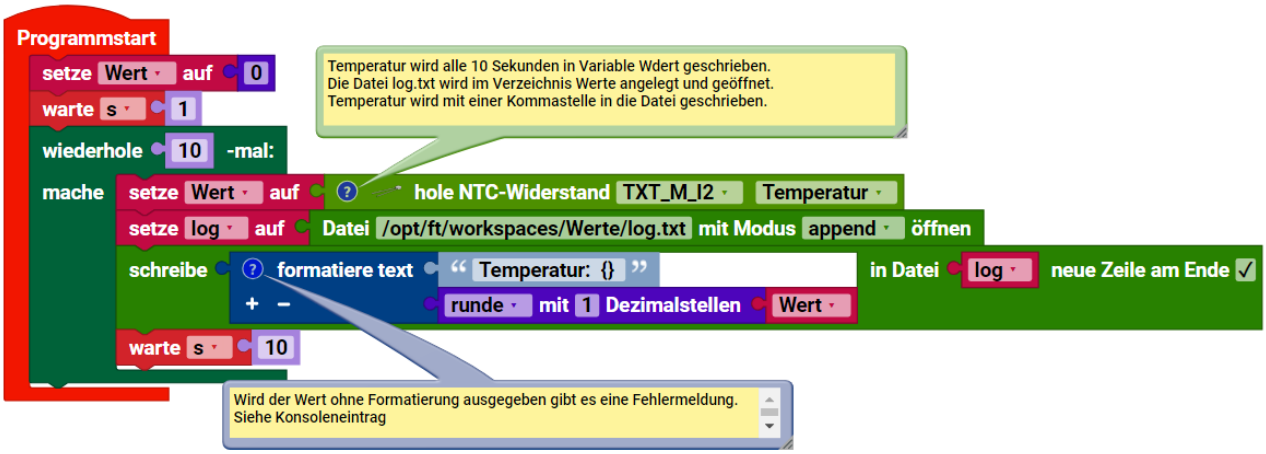
5.5.7 Verzeichnis path erstellen

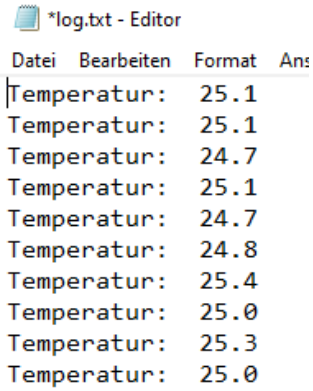


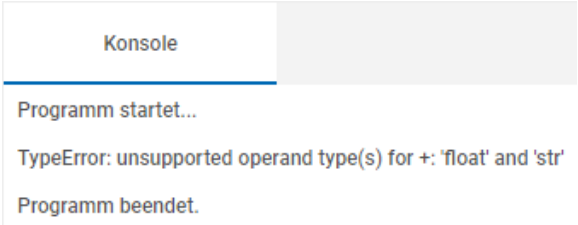
5.5.8 existiert …



5.5.9 Beispiel zur Datenaufnahme

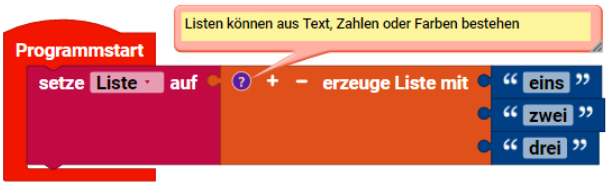


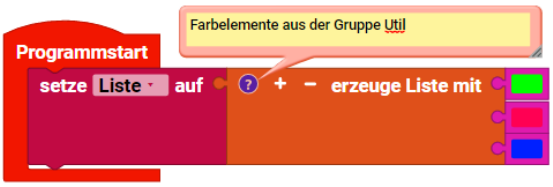




5.6 Datenstrukturen (Listen)

5.6.1 Liste erzeugen



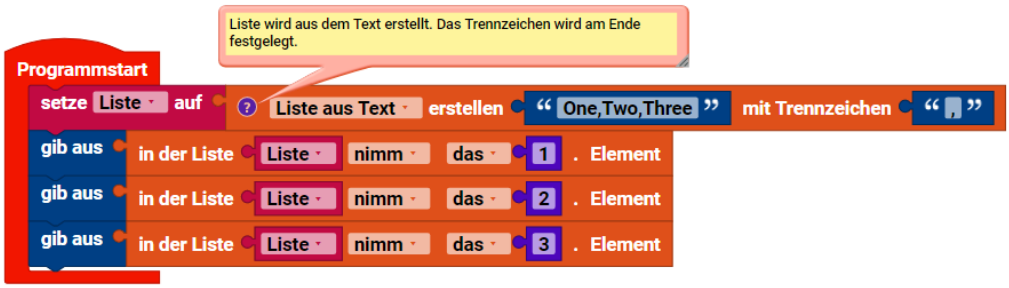


Konsole

Programm startet...

Test
Test
Test

Programm beendet.



Konsole

Programm startet...

One
Two
Three

Programm beendet.

5.6.2 Länge einer Liste ermitteln

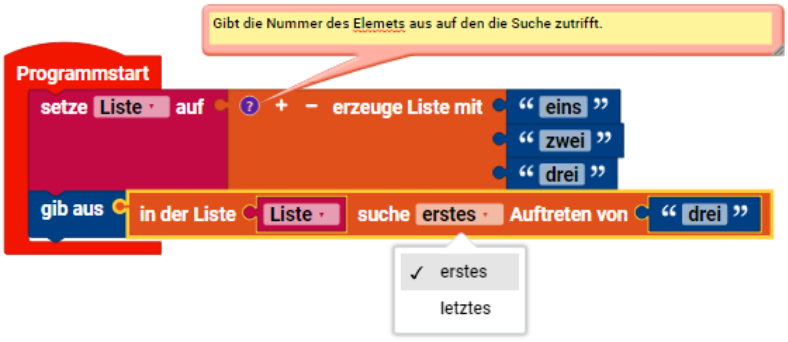
Konsole

Programm startet...

1

Programm beendet.

5.6.3 Suche und manipulieren von Elementen in einer Liste

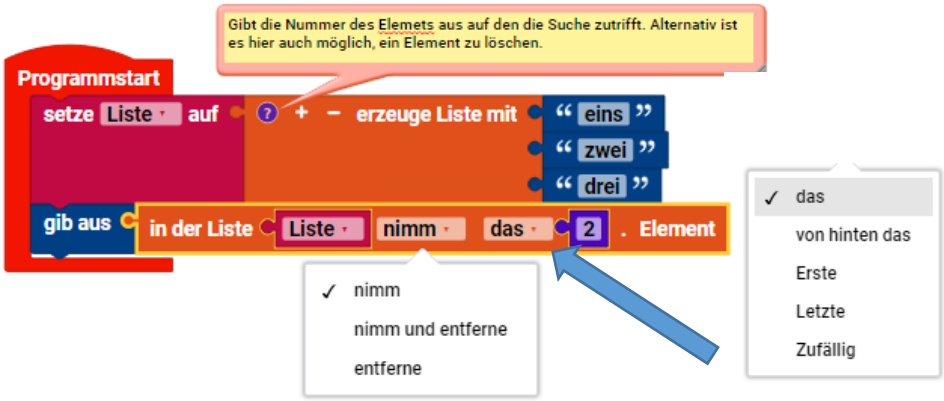


Konsole

Programm startet...

3

Programm beendet.



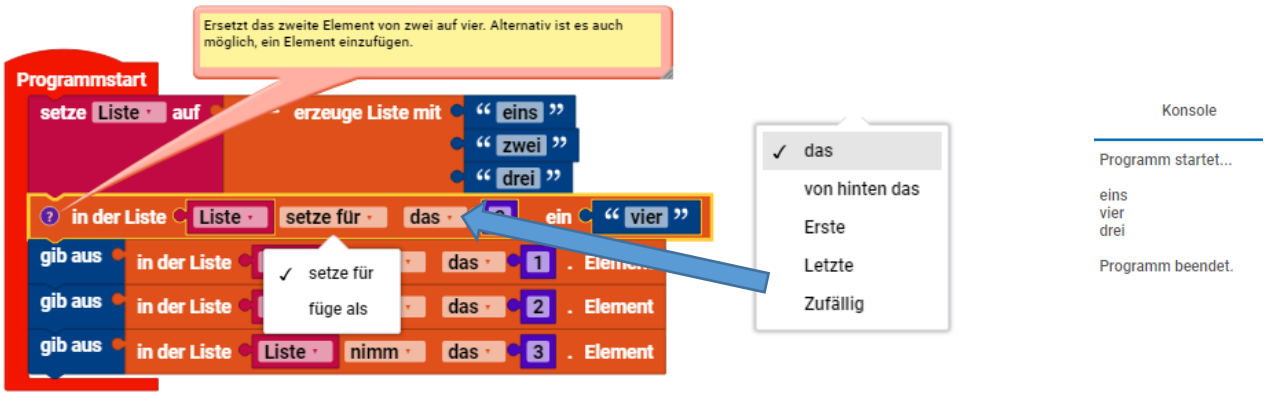
Konsole

Programm startet...

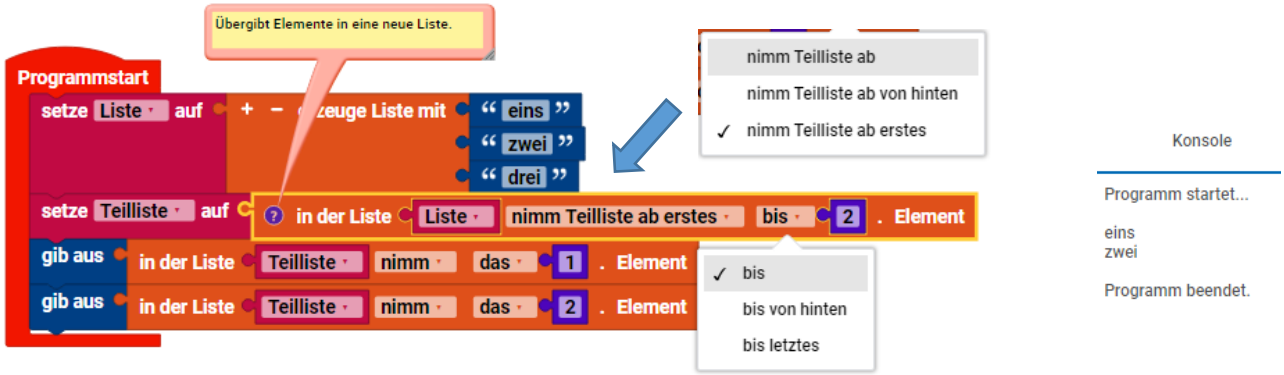
zwei

Programm beendet.

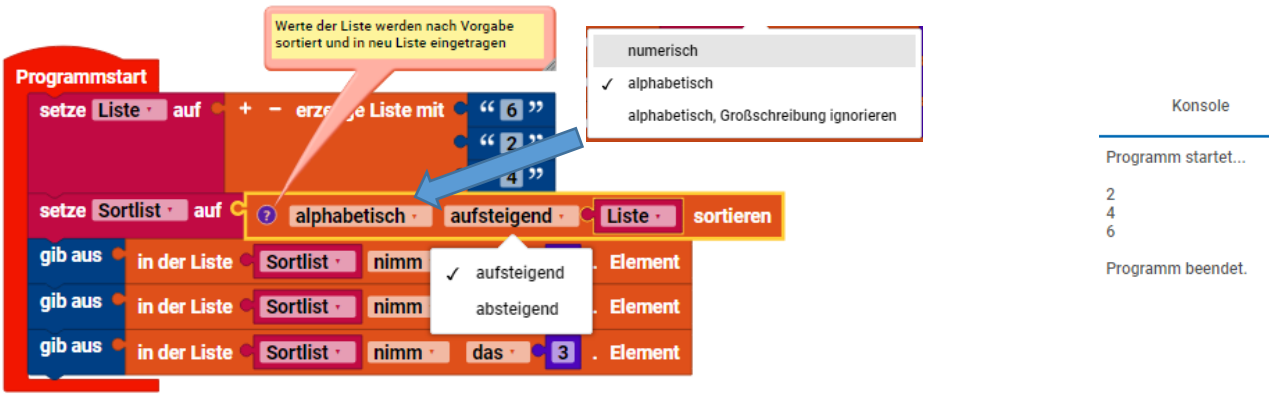
Bei Auswahl von „entferne“
Gibt es links keine Nase.



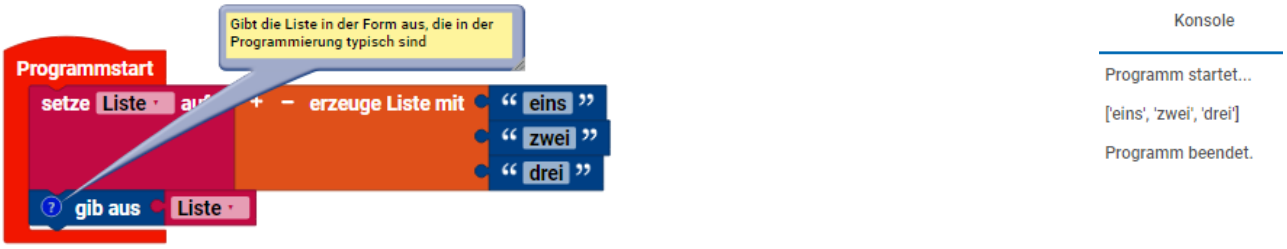
5.6.4 Sublist erstellen



5.6.5 Liste sortieren



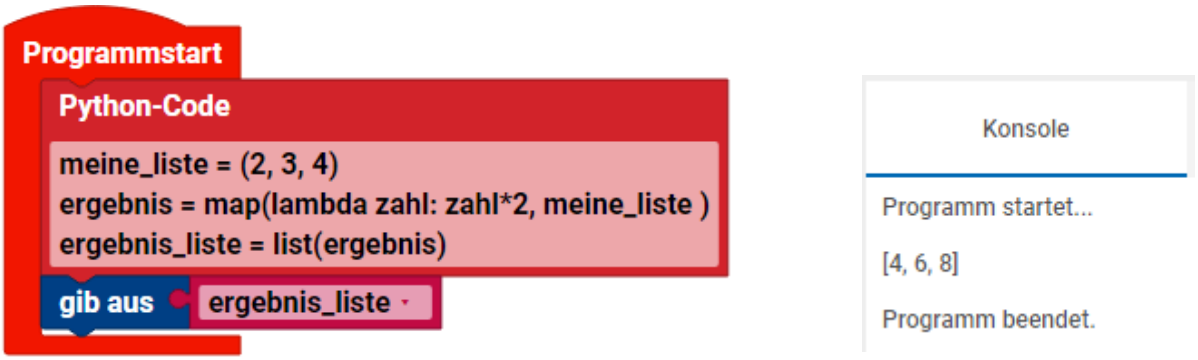
5.6.6 Liste drucken



5.7 Map

5.7.1 Map in Python

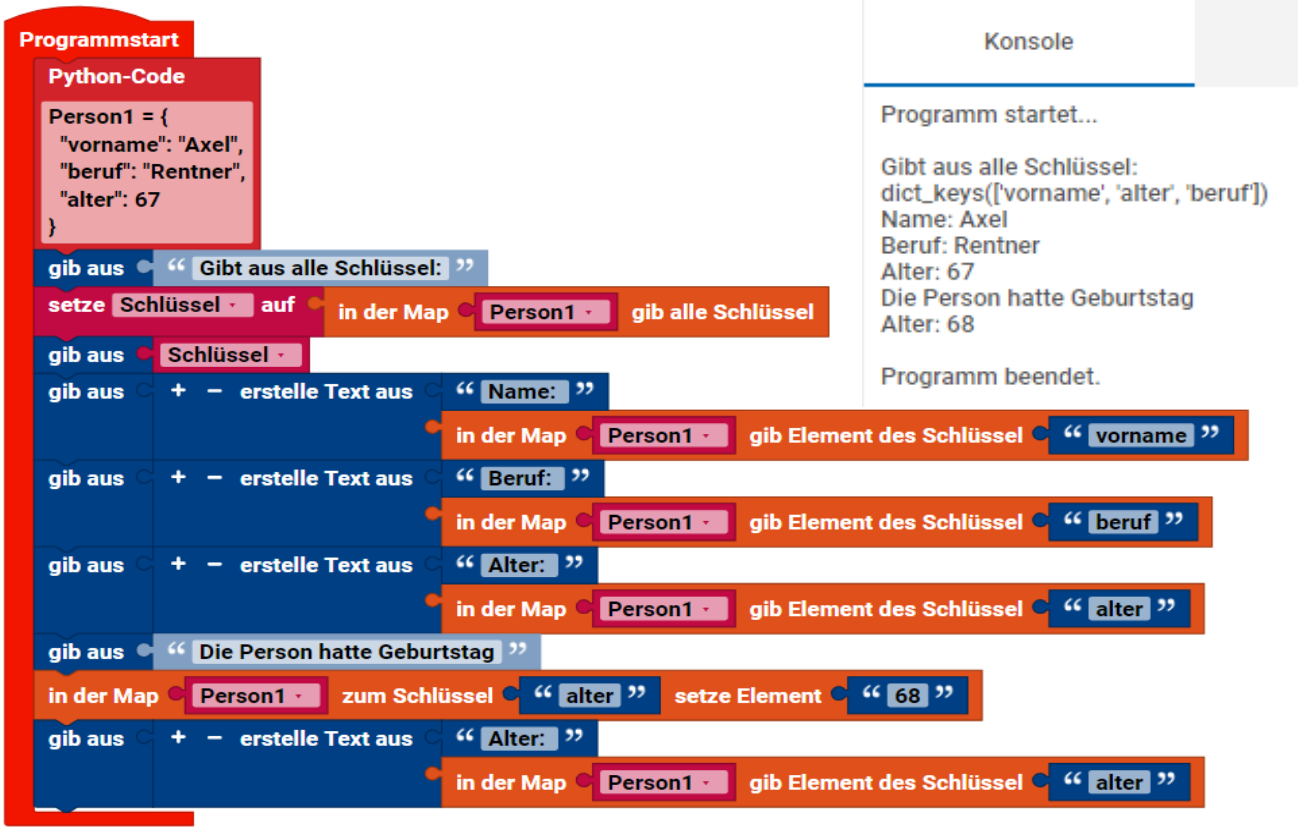
Die map()-Funktion in Python wird verwendet, um eine Funktion auf jedes Element einer Sequenz (wie einer Liste) anzuwenden und eine neue Sequenz mit den Ergebnissen zu erstellen. Im Beispiel werden die Werte der Liste quadriert und in einer weiteren Liste ausgegeben. Dabei ist „lambda“ eine Funktion ohne Namen, die es ermöglicht, die Transformation ohne Schleife zu erledigen.



Zeile 1: Erzeugung einer Liste mit den entsprechenden Ziffern.
Zeile 2: Die Funktion Map beginnt mit dem Lamda-Aufruf, der dann einen Paramater benötigt. Nach dem Doppelpunkt wird die eigentliche Funktion ausgeführt. Als letztes wird nach dem Komma der Name der zu bearbeitende Liste definiert.
Zeile 3: Das Map-Objekt wird in eine Liste zurückgewandelt und anschließend auf der Konsole ausgegeben.

5.7.2 Map in Fischertechnik

Eine Map in RPC, auch bekannt als Dictionary oder Assoziatives Array, ist eine Sammlung von Schlüssel-Wert-Paaren, bei der jeder eindeutige Schlüssel direkt auf einen Wert verweist. Die Integration von Dictionaries in RPC ist begrenzt. Die Erstellung ist nicht direkt möglich, sondern über den Block JSON (Siehe JSON zu Map) oder direkt mit Python.



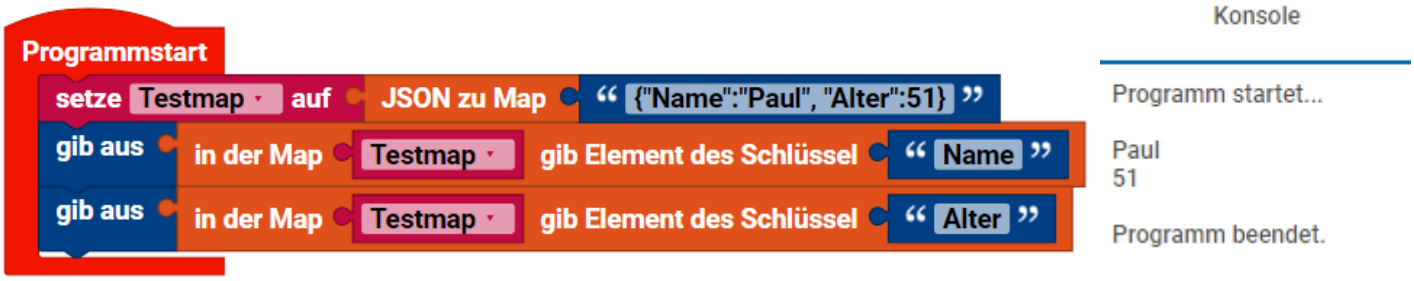
Zuerst wird die Map „Person1“ mit Python erstellt. Dann werden die Schlüssel angefordert um zu Testzwecken die Korrektheit zu dokumentieren.
Um nun mehr über die Person zu erfahren, muss man nicht die Position der Information kennen, sondern man fordert die Information direkt über den Schlüssel an.
Nach dem Geburtstag ist die Person ein Jahr älter. Dazu wird das Programm aktualisiert, indem das Alter verändert wird. Zum Abschluss wird nun auch das neue Alter ausgegeben.

5.8 JSON

JSON (JavaScript Object Notation) ist ein textbasiertes Datenformat, das für den Datenaustausch im Web genutzt wird. Es ist leicht lesbar und einfach zu handhaben. JSON strukturiert die Daten als Sammlung von Schlüssel-Wert-Paaren, ähnlich wie Dictionaries oder Maps.

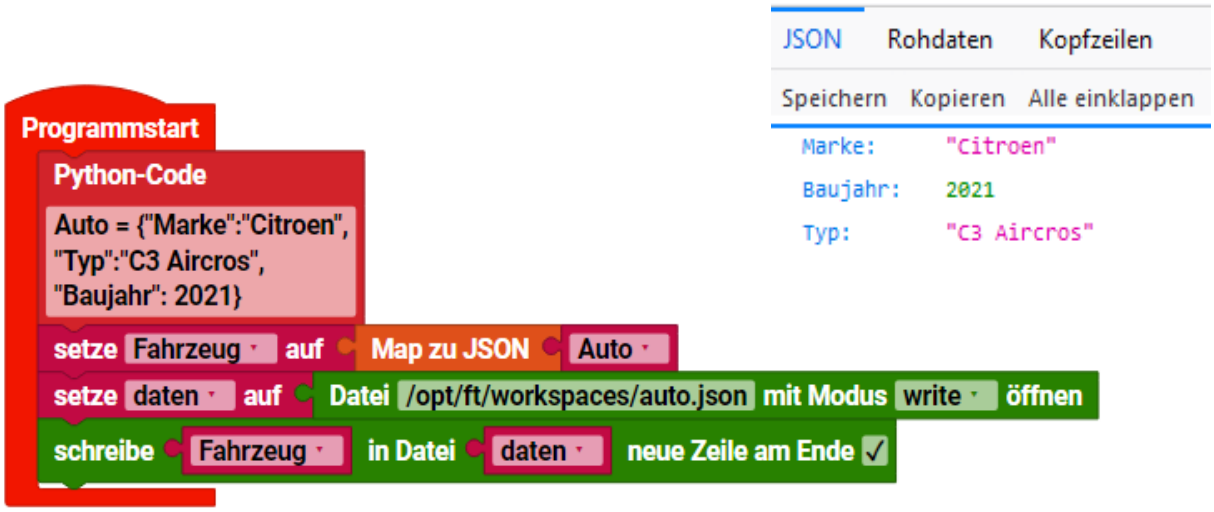
5.8.1 JSON zu Map

Dieser Block nimmt einen JSON-String als Eingabe und konvertiert diesen in eine Map. Jedes Schlüssel-Wert-Paar im JSON-String wird zu einem Eintrag in der Map und der Inhalt kann unter Verwendung des Schlüssels abgerufen werden.



5.8.2 Map zu JSON

Der Map zu JSON-Block ermöglicht die Umwandlung von Map-Datenstrukturen in einen JSON-String. Dieser Vorgang ist besonders nützlich, wenn die strukturierten Daten einer Map für eine Datenübertragung oder als Konfigurationsdatei in einem standardisierten Format benötigt werden.

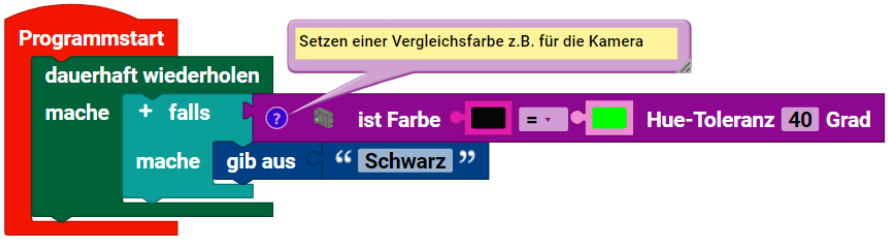


Im Beispiel wird eine Map „Auto“ über Python erzeugt. Diese wird nun zu einem JSON-String umgewandelt.
Im folgenden werden die Daten in die Datei „auto.json“ übertragen. Das Ergebnis nach dem öffnen der Datei ist oben zu sehen.

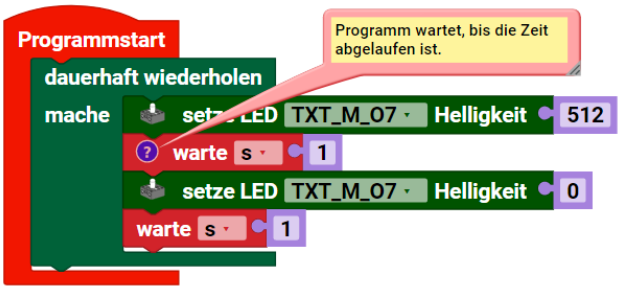
Hinweis: Alle in den Beispielen vorkommenden Variablen, müssen vorher angelegt werden!

5.9 Util

5.9.1 Farbauswahl

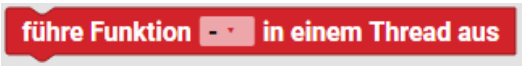


5.9.2 Warte (Zeit]



5.9.3 Warte (Bedingung)

5.9.4 Thread



Siehe 5.9.4

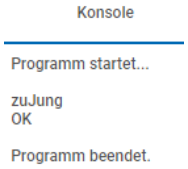
5.10 Variablen

5.10.1 Erstellen

5.10.2 Festlegen

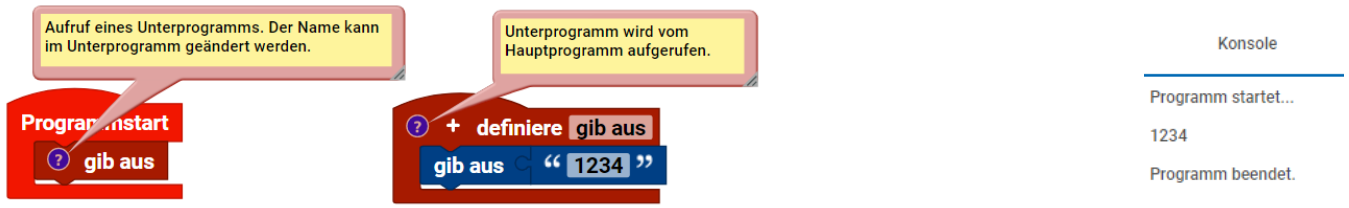
oder

5.10.3 Wert ändern

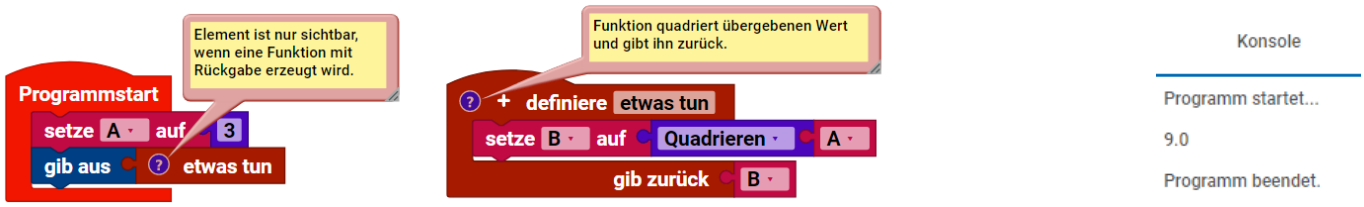


5.11 Funktionen (Unterprogramme)

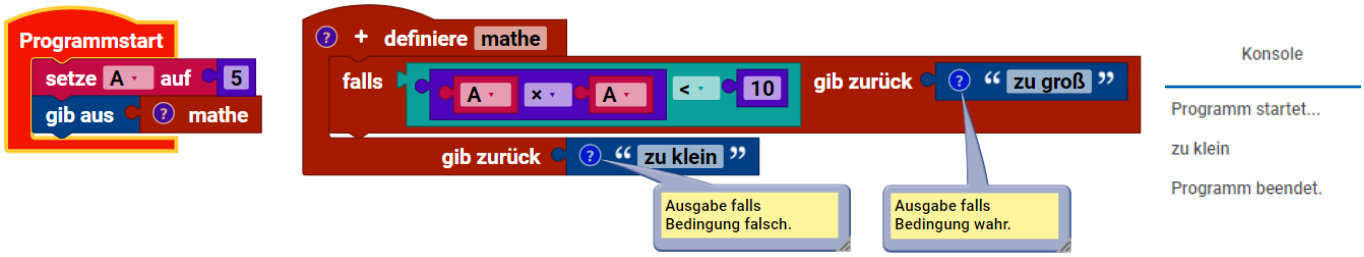
5.11.1 Einfacher Aufruf



5.11.2 Aufruf mit Rückgabe



5.11.3 Aufruf mit Rückgabe und Bedingung



5.11.4 Funktion als Thread ausführen

Im Normalfall werden Programme nacheinander ausgeführt. So wartet das Programm auch mit der Weiterführung, wenn ein Unterprogramm abgearbeitet wird. Da hier das Unterprogramm unendlich ist, läuft das Hauptprogramm nicht weiter.

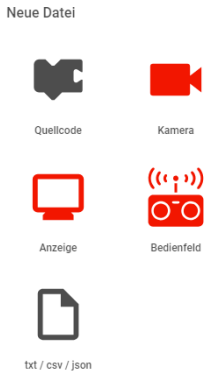
Deshalb wird hier das Unterprogramm 2 angewiesen, als Thread (also parallel) zu laufen.

5.12 Machine Learning

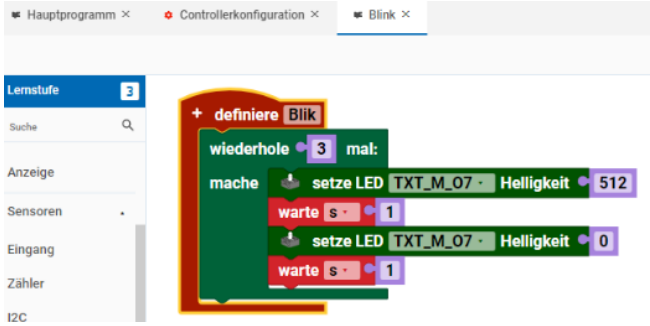
Noch nicht dokumentiert.

5.13 Importe

Um den Quellcode übersichtlicher zu gestalten, ist es möglich, zusätzliche Module(Dateien) anzulegen. Dort können z.B. Funktionen (Unterprogramme) abgelegt werden. Mit  wird das folgende Fenster geöffnet. Durch die Auswahl Quellcode wird eine neue Datei angelegt und als Reiter im Kopf angezeigt. 

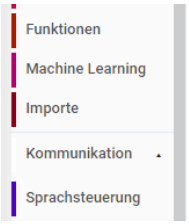


Hier kann das Unterprogramm geschrieben werden.



Im Hauptprogramm, wird die Funktion 

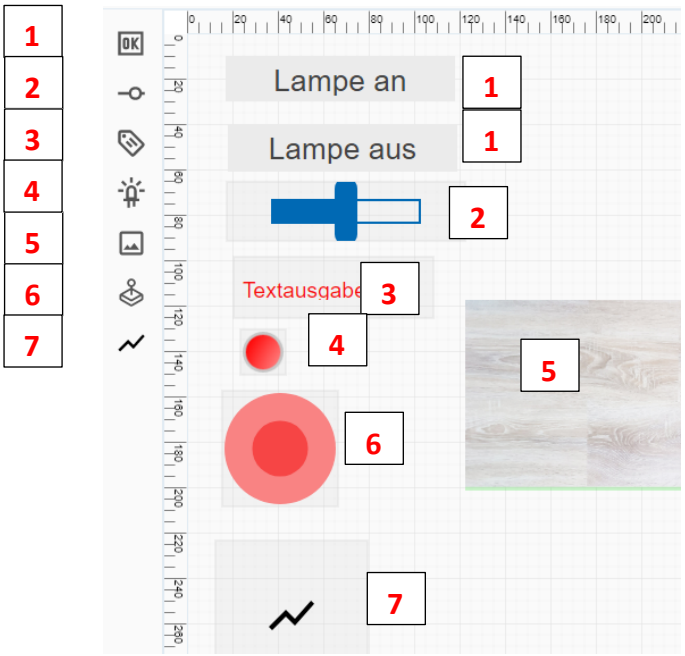
„Blink“ aus gewählt.

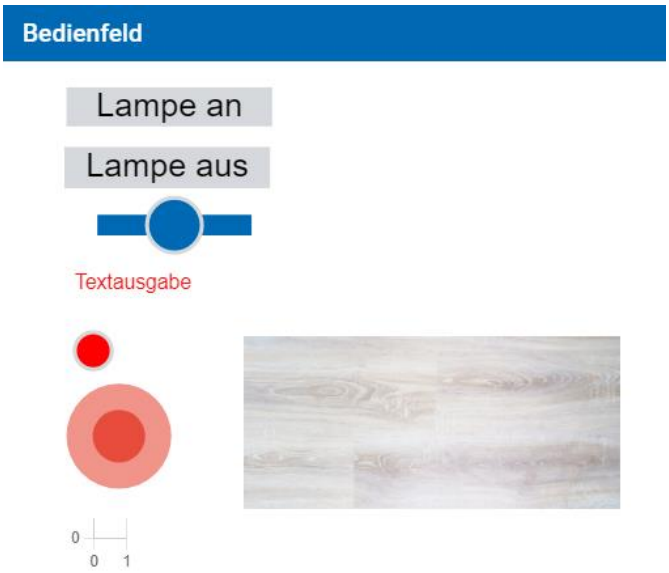


6 Kommunikation

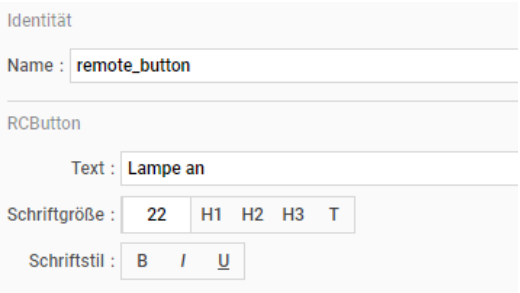
6.1 Fernbedienung

Über eine Benutzeroberfläche, können Fahrzeuge oder Modelle ferngesteuert werden.
Bevor Befehle ausgeführt werden können, dauert es derzeit bis zu 20 Sekunden bis Aktionen möglich sind.

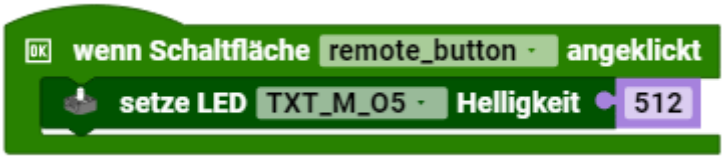




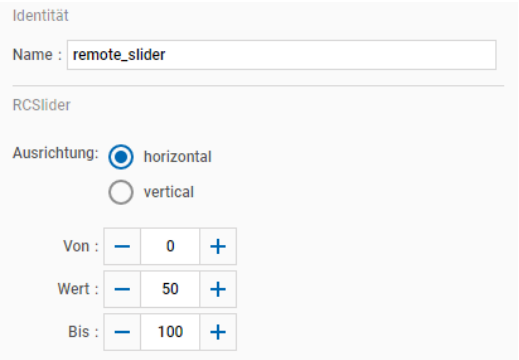
6.1.1 remote_button (1)



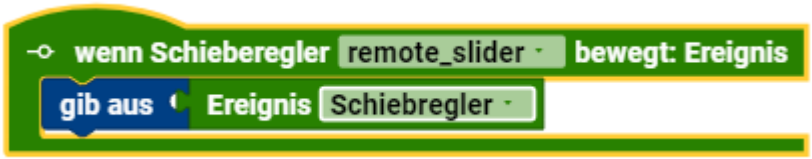
Durch Anklicken der Schaltfläche wird eine Aktion ausgeführt
Schriftgröße und Schriftstil können im Inspektor festgelegt werden.



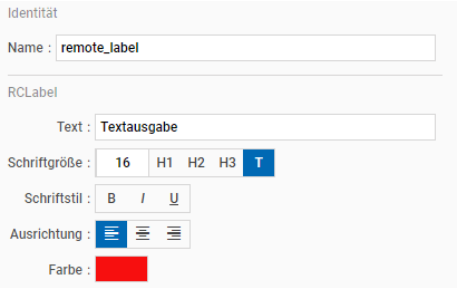
6.1.2 remote_slider (2)



Die Bewegung des Sliders kann zur Steuerung von Motoren genutzt werden. Vorher werden der Wertebereich und der Anfangswert festgelegt. (Hier Ausgabe auf Konsole)

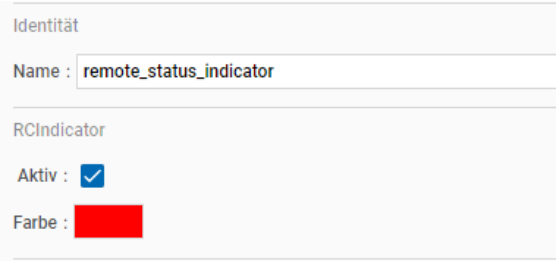


6.1.3 remote_label (3)

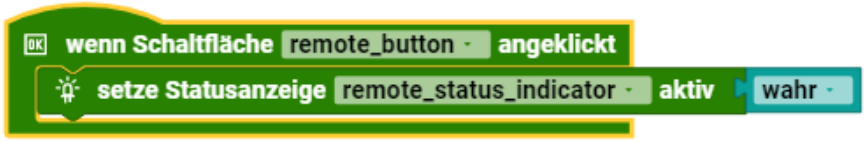


Einerseits kann hier eine Beschriftung der Fernbedienung erfolgen. Andererseits können auch Rückmeldungen als Text erfolgen. (Siehe Beispiel). Zusätzlich zur Textgröße und Schriftstil, kann auch die Ausrichtung und Farbe im Inspektor voreingestellt werden.

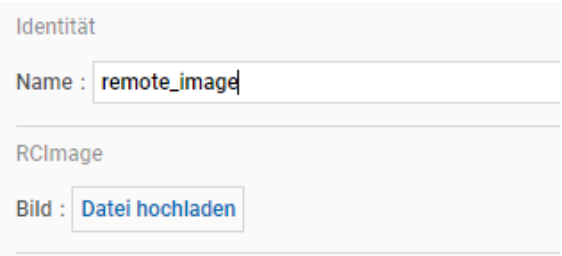
6.1.4 remote_status_indicator (4)



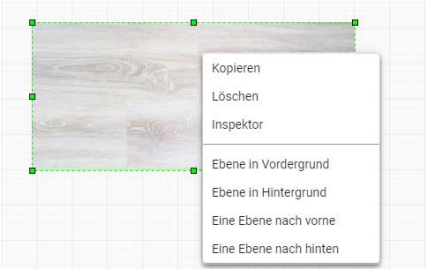
Wie bei dem remote_label kann hier die Rückmeldung einer Aktion angezeigt werden. Oder man nutzt die Anzeige für Fehlermeldungen oder das Ende einer Aktion.

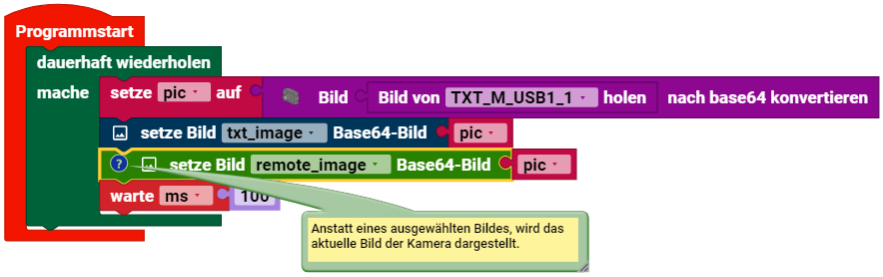


6.1.5 remote_image (5)



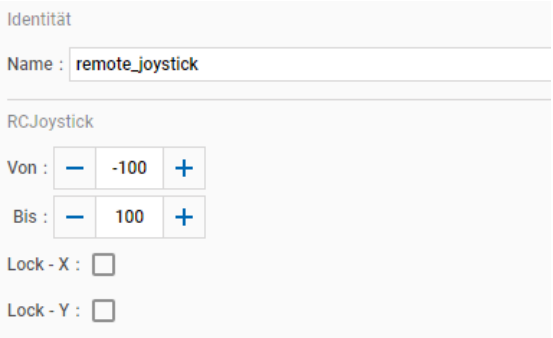
Hiermit lässt sich ein Bild auf die Bedienoberfläche legen. Entweder als Detailbild zur besseren Erkennung oder sogar als Hintergrundbild des ganzen Bedienfeldes. Über das Kontextmenü kann dann die Ebene Hintergrund ausgewählt werden.



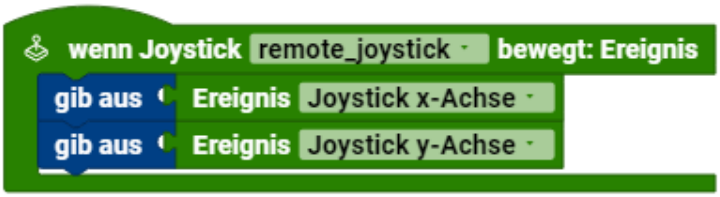


Eine zweite Möglichkeit, ist die Darstellung des Kamerabildes. Abhängig von der Größe des Bildes, kann es sehr lange dauern, bis das Bild dargestellt wird!!

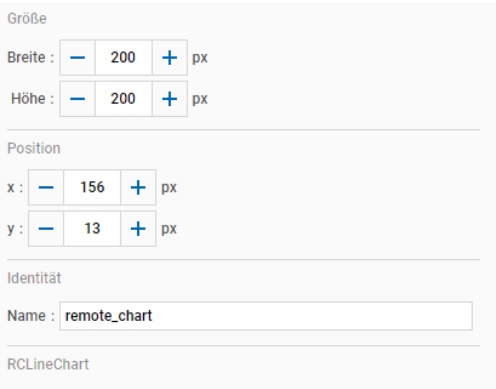
6.1.6 remote_joystick (6)

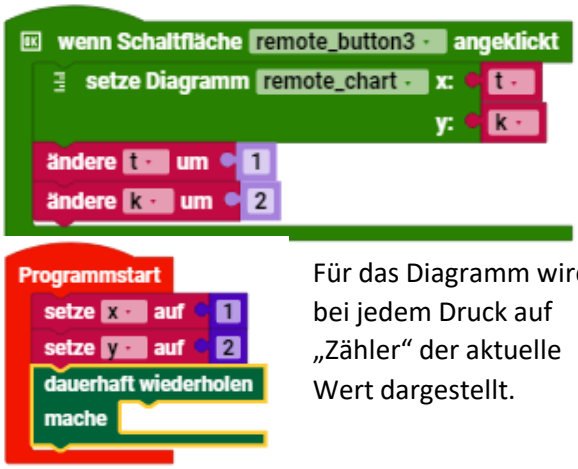


Die Bewegung des Joysticks kann in X und Y-Position ausgewertet werden. Im Inspektor können Anfangs und Endwert festgelegt werden. Im Beispiel erfolgt die Ausgabe auf der Konsole.



6.1.7 remote_chart (7)

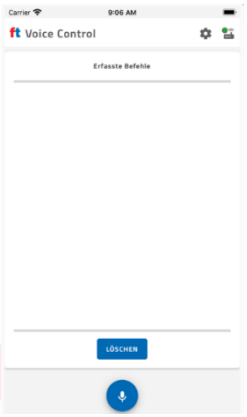


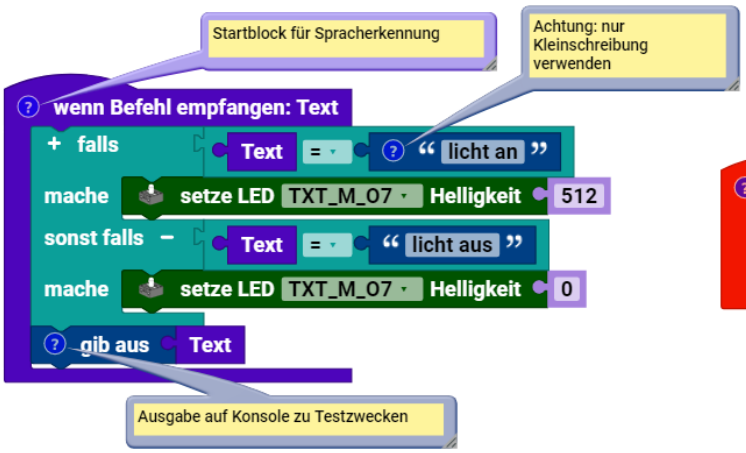


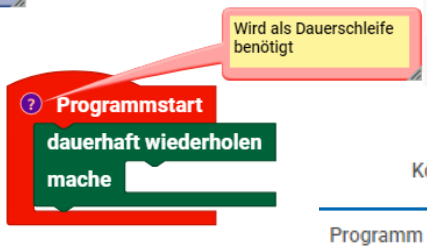
Für das Diagramm wird bei jedem Druck auf „Zähler“ der aktuelle Wert dargestellt.

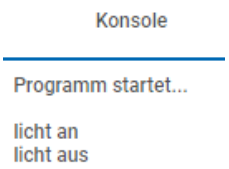
6.2 Sprachsteuerung

Um die Sprachsteuerung zu nutzen, ist die App „Voice Control Fischertechnik“ auf dem Handy erforderlich. Anschließend meldet man sich auf dem TXT-Controller über WLAN an.
Der gesprochene Befehl wird wie im Beispiel ausgewertet und es erfolgt die entsprechende Aktion.









6.3 Cloud / MQTT

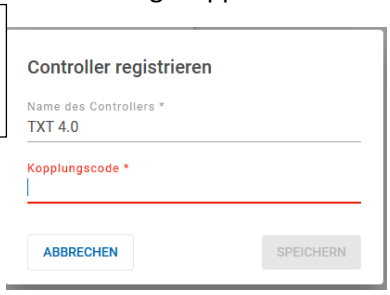
6.3.1 Cloud

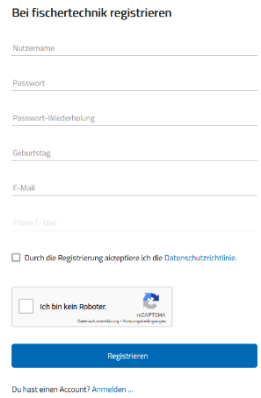
Die Cloud ist unter „https://www.fischertechnik-cloud.com“ zu erreichen. Bevor man mit diesem Dashboard arbeiten kann, muss eine Anmeldung erfolgen.

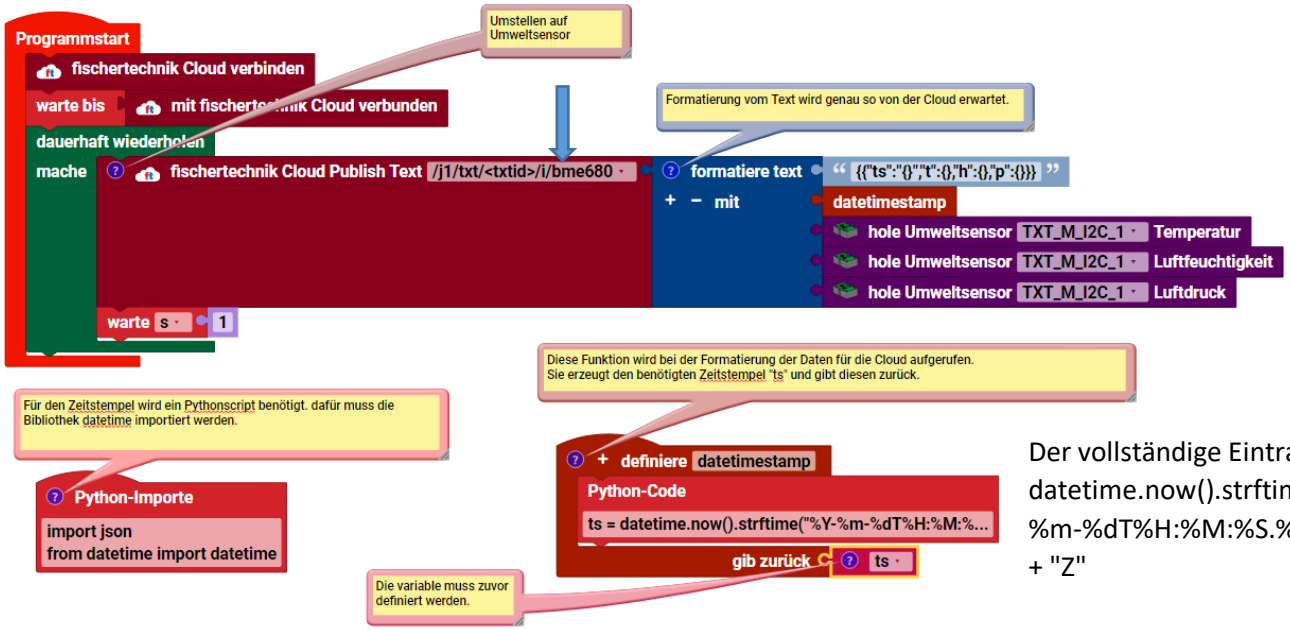
Anschließend wird der Controller mit der Cloud gekoppelt werden.

Einstellungen - Controller hinzufügen

Einstellungen - fischertechnik Cloud

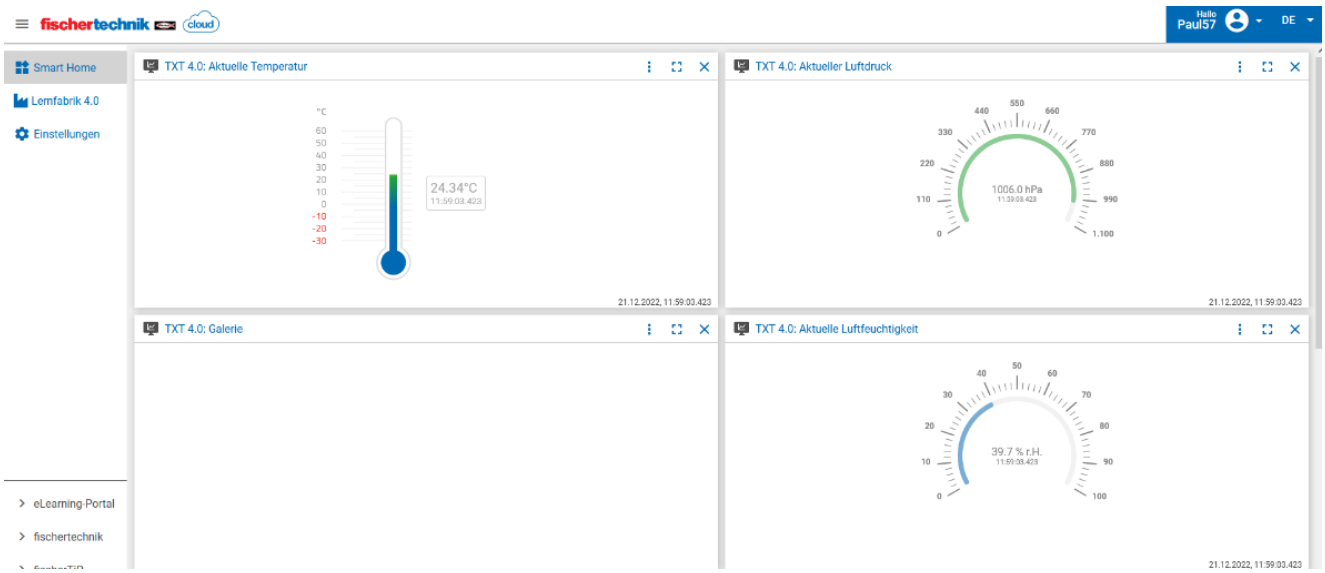






Der vollständige Eintrag: ts= datetime.now().strftime("%Y-%m-%dT%H:%M:%S.%f")[:-3] + "Z"

Wenn alles klappt, sollten die 3 Werte (Temperatur, Luftfeuchtigkeit und Luftdruck) in den jeweiligen Fenstern der Cloud im Sekundentakt angezeigt werden.



6.3.2 MQTT

Hinweis

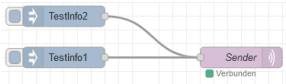
Für diesen Abschnitt, sind Grundkenntnisse von Node-RED notwendig.

Grundlagen

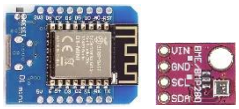
MQTT steht für „Message Queuing Telemetry Transport“. Es ist ein offenes Netzwerkprotokoll für Machine-to-Machine-Kommunikation, dass die Übertragung von Telemetriedaten in Form von Nachrichten zwischen Geräten ermöglicht. MQTT funktioniert nach dem Publisher- / Subscriber-Prinzip und wird über einen zentralen Broker betrieben. Das bedeutet, dass Sender und Empfänger keine direkte Verbindung haben.

Veröffentlichen

Datenausgabe über TXT 4.0



Datenausgabe über Node-RED
192.168.x.x:1880

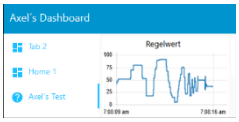


Datenausgabe über Microcontroller
z.B. ESP8266 mit MBE280

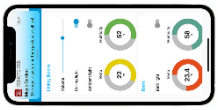
MQTT-Broker
127.0.0.1

Abonnieren

Datenanzeige über TXT 4.0

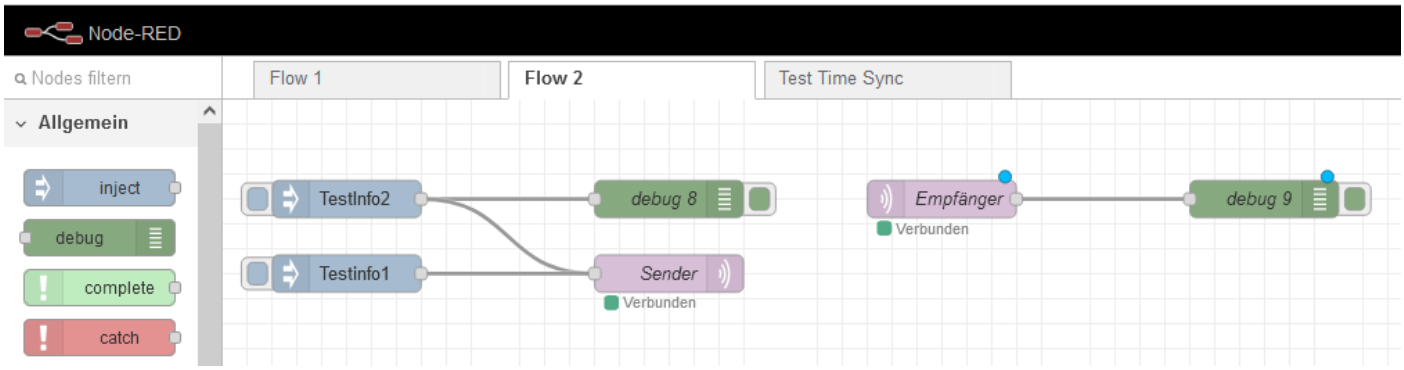


Datenanzeige über Dashboard
192.168.x.x:1880/ui



Datenanzeige über Dashboard
auf dem Smartphone

Node-RED ist ein von IBM entwickeltes grafisches Entwicklungswerkzeug. Die Software ermöglicht es, Anwendungsfälle im Bereich des Internets der Dinge mit einem einfachen Baukastenprinzip umzusetzen. Die einzelnen Funktionsbausteine werden durch Ziehen von Verbindungen verbunden.

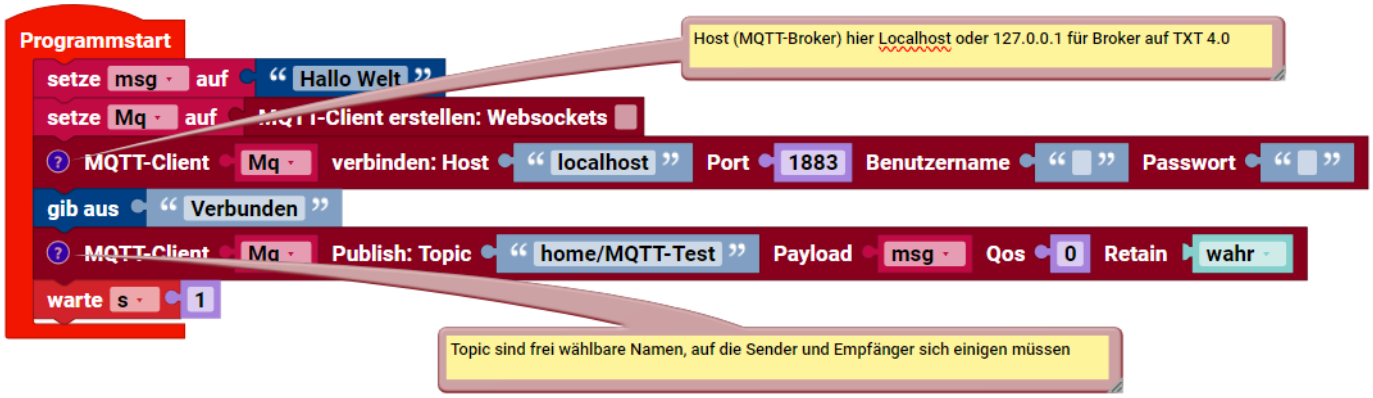


Für die Nutzung von MQTT und Node-RED gibt es von externen Anbietern entsprechende Software. Mit dem TXT 4.0 werden diese Softwareelemente bereits mitgeliefert. Diese werden über folgende Adressen aufgerufen:

MQTT-Broker: 127.0.0.1 oder Lokalhost
Node-RED: 192.168.x.x:1880
Dashboard: 192.168.x.x:1880/ui

Für .x.x muss die entsprechende Adresse des eigenen TXT 4.0 eingetragen werden.

Daten an MQTT senden (und über Node-RED auslesen)



Daten vom MQTT empfangen (die von Node-RED gesendet wurden)