fischertechnik TXT community firmware

Die Community-Firmware ist ein freies und modernes Betriebssystem für den fischertechnik TXT. Stark erweiterte Internet-Fähigkeiten inklusive eigenem App-Store sowie die Steuerung und Programmierung von Modellen über PC, Tablet oder Smartphone machen deinen TXT fit für die Zukunft.

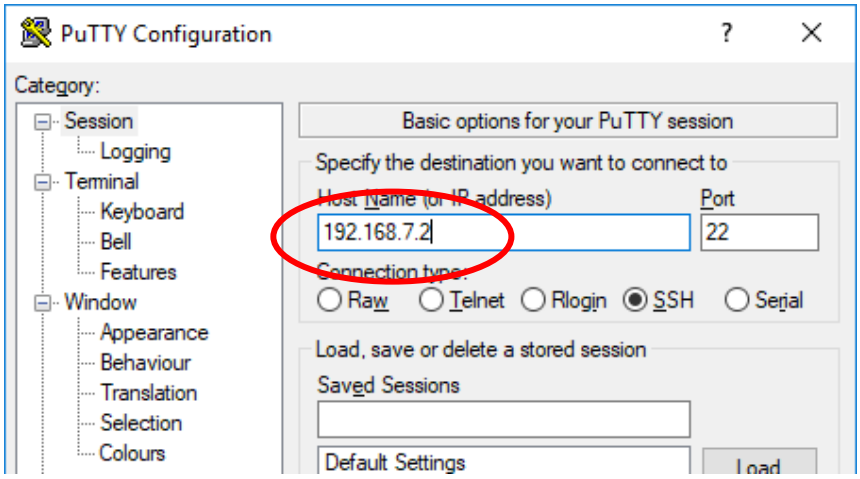
Dafür muss der TXT nicht geöffnet oder verändert werden, denn die Community-Firmware wird auf SD-Karte installiert und kann so jederzeit wieder entfernt werden.
Die App´s werden ebenfalls auf der SD-Karte abgelegt. Sie sind aber auf einem eigenen Linux Partition und daher unter Windows nicht sichtbar.

1 Vorraussetzung

- fischertechnik TXT
- eine MicroSD-Karte mit 2 bis 32GB Kapazität

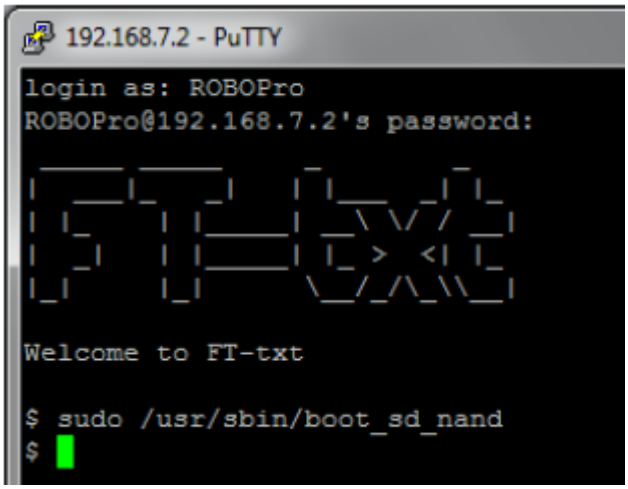
Die Installation erfolgt in vier einfachen Schritten:

1. Stelle sicher, dass du mindestens RoboPro Version 4.2.4 verwendest.
2. Schalte den Bootloader Deines TXT entsprechend der offiziellen fischertechnik-Anleitung frei.
   Unter http://www.putty.org wird das Konfigurationsprogramm heruntergeladen und gestartet.



Verbindung zum TXT herstellen und eine der folgenden Adressen eintragen:
USB: 192.168.7.2
WLAN(AP): 192.168.8.2 Bluetooth: 192.168.9.2

Klicke anschließend auf „Open“, um die putty-Konsole zu öffnen.
Login: ROBOPro
Password: ROBOPro
Um das Booten von SD-Karte zu aktivieren:
sudo /usr /sbin /boot_sd_nand



3. Entpacke die drei im Community-Firmware-ZIP-Archiv enthaltenen Dateien auf deine MicroSD-Karte.
   (https://github.com/ftCommunity/ftcommunity-TXT/releases/latest)

am335x-kno_txt.dtb	36.575	DTB-Datei
rootfs.img	109.010.944	Datenträgerimagedatei
ulmage	6.511.400	Datei

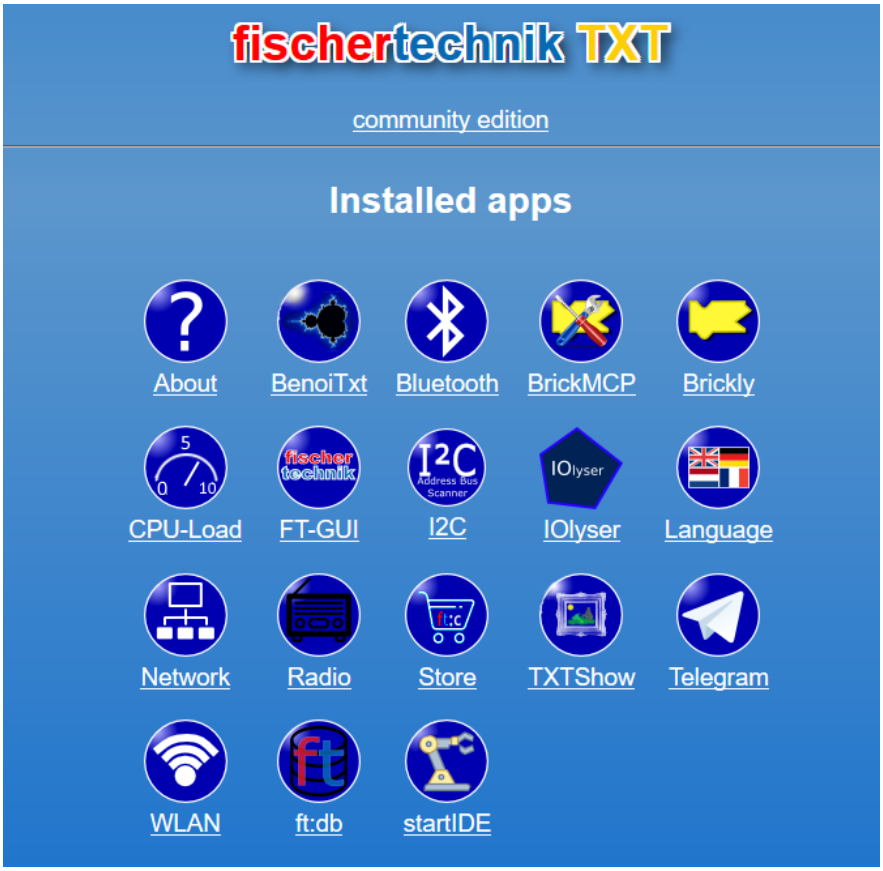
4. Stecke die MicroSD-Karte in deinen TXT und schalte ihn ein!

Jetzt kannst du verschiedene Apps nutzen und mit Brickly, Python und ROBOPro deinen TXT programmieren.
Wenn du die MicroSD-Karte wieder aus dem TXT entfernst, läuft nach dem nächsten Start wieder die eingebaute Original-Firmware.

2 Apps

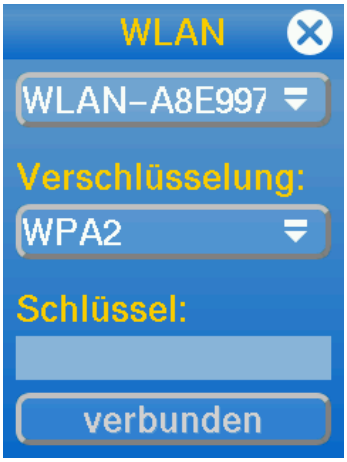
2.1 Netzwerk

Die Community-Firmware kann über den Webbrowser angesprochen werden.
Dazu muss in der App Netzwerk die Adresse für die Sevicedienste festgelegt werden. Im Beispiel wird die USB-Schnittstelle mit der Adresse 192.168.2.122 eingerichtet.



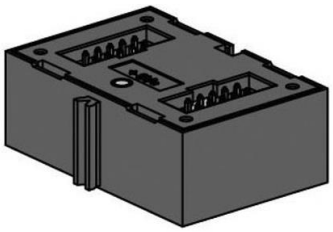
Von hier aus können dann die Apps ebenfalls geöffnet und bedient werden.

2.2 WLAN



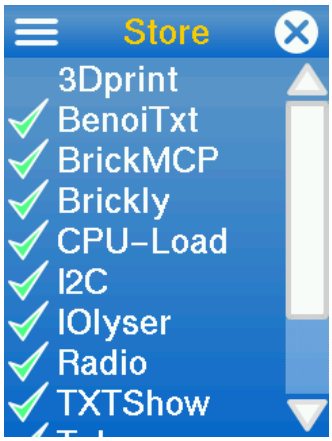
Hier wird die Verbindung zu Router per WLAN hergestellt. Dazu wird der Router ausgewählt, die Verschlüsselung übernommen und der Schüssel eingetragen.

2.3 I2C Scanner



Mit dem I2C-Scanner werden die Adressen der angeschlossenen Geräte ausgelesen. Hier am Beispiel des Kombisensors von Fischertechnik.

2.4 Store

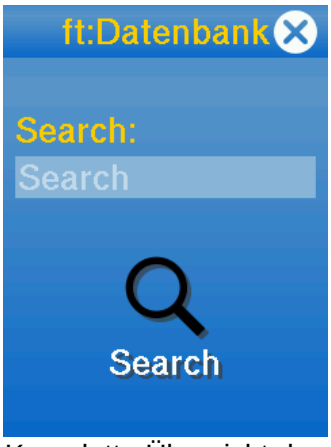


Im App-Store können diverse Programme heruntergeladen werden.

2.5 Sprache

Einfaches wechseln der Sprache

2.6 FT Datenbank

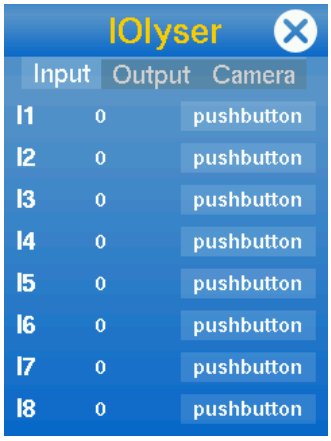


Komplette Übersicht der Bauteile von Fischertechnik

2.7 Radio

Web-Radio über den TXT zu empfangen.

2.8 IOlyser

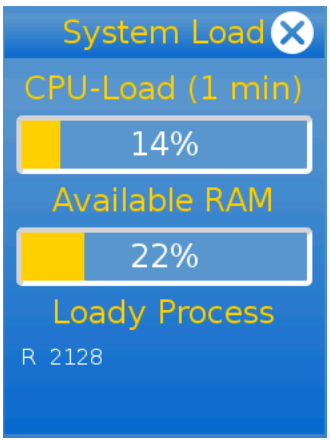


Mit dem IOlyser können die Ein- und Ausgänge des TXT Controllers angesprochen bzw. geprüft werden.

2.9 Über

Über den Button Links Oben lässt sich die GPL, LGPL und MIT-Lizenz auslesen.

2.10 CPU-Load



Eine kleine App für den TXT, um die durchschnittliche Prozessorauslastung und den verfügbaren Arbeitsspeicher zu überprüfen.

2.11 BenoiTxt

Mandelbrot-Bildgenerator

2.12 startIDE

Programmier-App fur die community firmware des TXT Controllers, mit der sich eine Vielzahl einfacher Modelle programmieren läst. Siehe Anleitung.

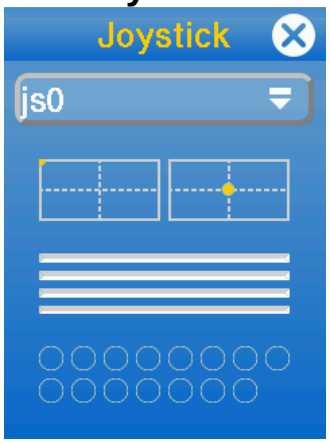
2.13 IDAS Ampel

Einfache steuerbare Ampelschal-tung für die Ausgänge O1 bis O5

2.14 Clock

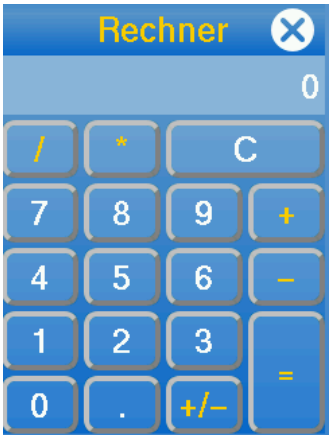
Anzeige der aktuellen Zeit und des Datums

2.15 Joystick



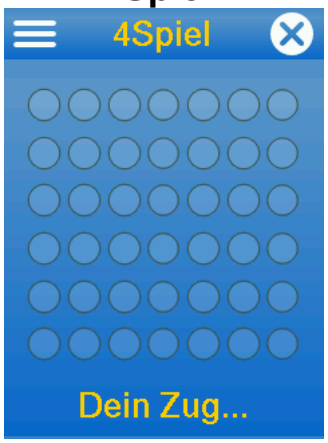
Programm zum Testen eines angeschlossenen Joysticks

2.16 Calc



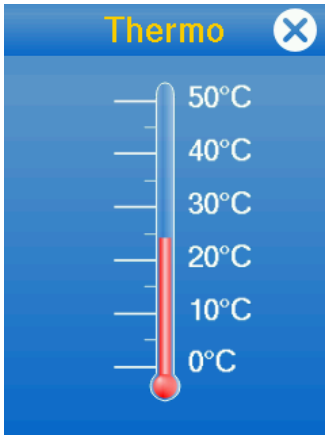
Einfacher Taschenrechner

2.17 4 Spiel



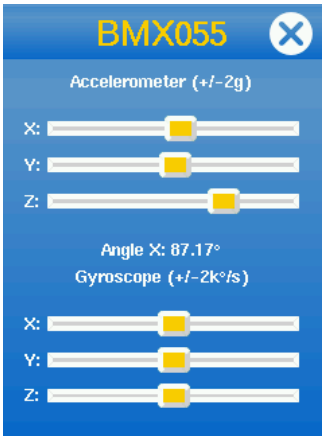
Bekannt als 4 Gewinnt

2.18 Thermo



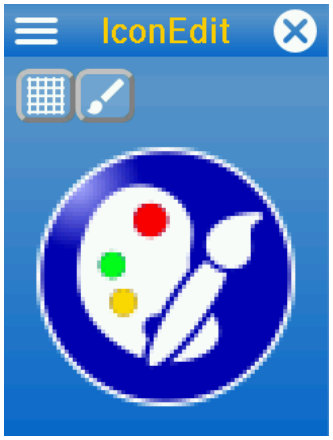
Thermometer. Liest den Wert von Eingang 1

2.19 BMX 055



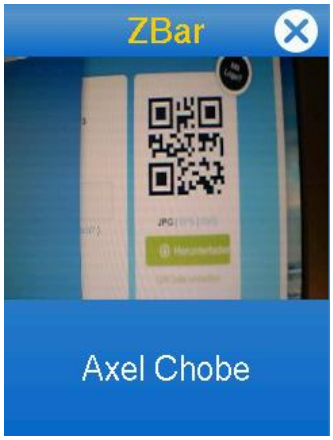
Anzeige der Werte vom Kombisensor (Gyroskop)

2.20 Icon Editor



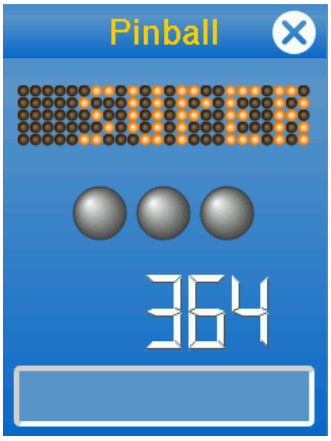
Es können alle vorhandenen Icons editiert werden.

2.21 ZBar



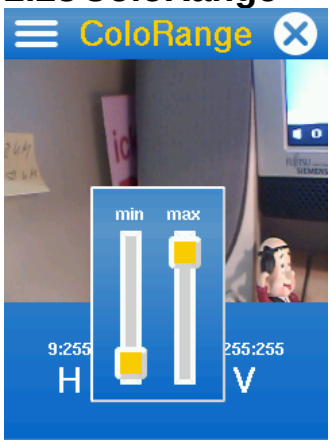
Liest den QR Code und gibt den entsprechenden Text aus

2.22 Pinnball



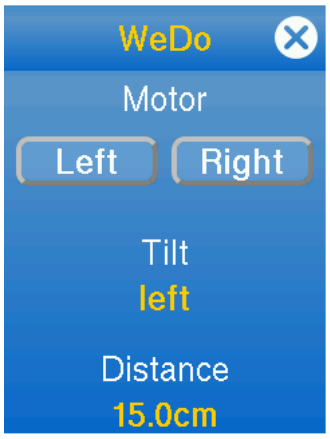
I1 = für M2 Schuß links
I2 = für M3 Schuß rechts
I3 = Lichschranke für Zähler
I4 = offen fürBallverlust
M4 = Licht für Lichtschranke

2.23 ColoRange



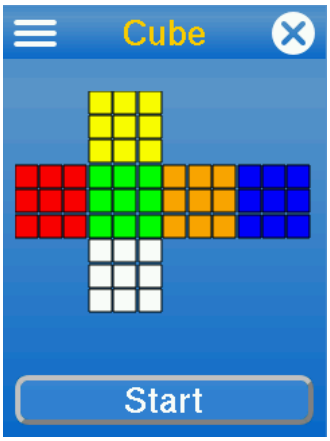
App, mit der man Farbbereiche zur Objekterkennung auswählen kann

2.24 WeDo



Zum steuern des Lego-Controllers WeDo 2.0

2.25 Cube

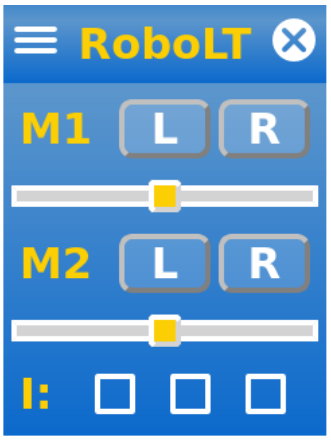


Liest den Zustand ein, errechnet die Lösung und steuert die Motoren

2.26 3 D Print

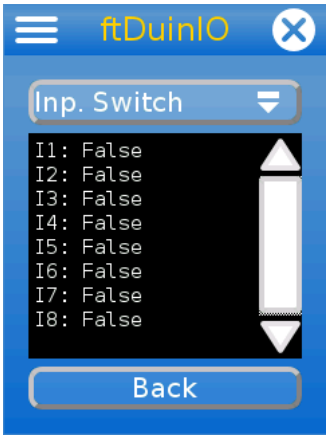
Steuert den 3 D Printer von Fischertechnik

2.27 RoboLT



Steuert das Interface Robo LT Fischertechnik Beginners

2.28 ftDuiIO

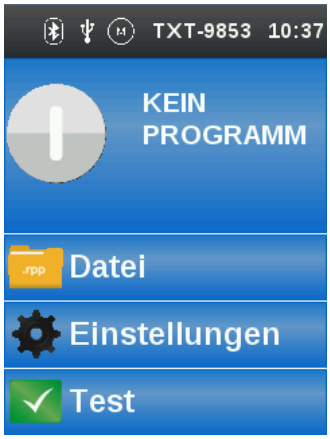


IO-Testprogramm für den ftduino

2.29 Widgets

Beispieldarstellungen von Dialog, Zustandsanzeige und Text.

2.30 FT-GUI



Hier lässt sich das System auf die Originalsoftware umschalten. Mit Hilfe des Knopfes ON/OFF geht es zurück.

3 TXT Show

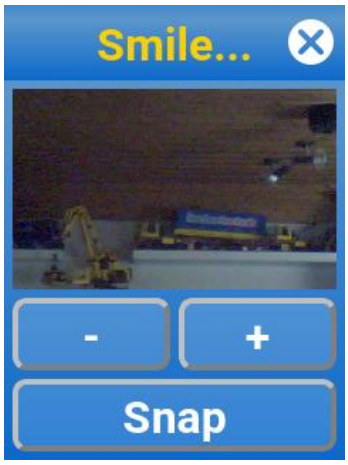
Sie dient dem Anzeigen und Verwalten von Fotos auf dem TXT. Es können mit der USB-Kamera Fotos gemacht werden, die übers Webinterface der App auch heruntergeladen werden können.

Die Funktionen:
- oben links: Menu aufrufen
- oben rechts: beenden
- Pfeile links und rechts: vorheriges bzw. nächstes Bild anzeigen
- unten links: Zoom an/aus
- unten rechts: Play/Pause

Menü1:
Schnellauswahl des aktuellen Bilds - über den Drehsteller und die grünen Buttons kann man schnell durch das aktuelle Fotoalbum springen - praktisch, wenn viele Bilder darin sind.
Mit der Uhr kann man die Bildwechselverzögerung für die automatische Diashow einstellen, die Return-Taste beendet den Menuscreen, mit der Kamera kommt man zum Fotos machen, mit dem blauen Pfeil auf Menuscreen II.

Menü2:
Bild in ein anderes Album kopieren (weiße Blätter), Bild in ein anderes Album verschieben (Schere), Bild umbenennen (Eingabefeld) und Bild löschen (Mülleimer)

Menü 3:
Aktuelles Album auswählen (Album-button), neues Album anlegen (Ordner +), Album löschen (Ordner -) und Album umbenennen (Eingabefeld)



Das Foto-Fenster:
Mit + bzw. - kann die Kameraanzeige gezoomt werden, um besser fokussieren zu können.
Der Zoom hat keine Auswirkung auf das gemachte Foto.
Mit "Snap" wird die Aufnahme im aktuellen Album gespeichert.

Das Webinterface:

Hier kann man ebenfalls Alben anlegen und löschen.
Wenn man ein Album auswählt, gelangt man in die Bilderübersicht des jeweiligen Albums.

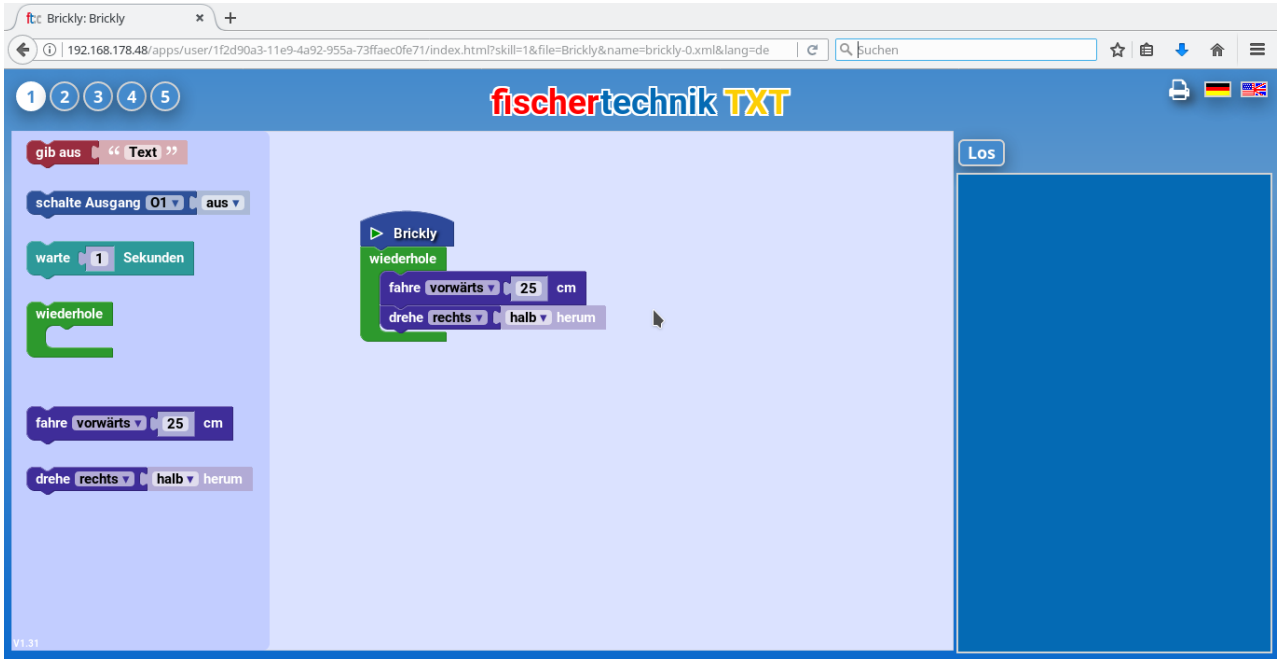
Hier können die Bilder gelöscht oder Bilder auf den TXT hochgeladen werden.
Außerdem kann man hier das Album umbenennen.

Wenn man das Vorschaubild anklickt, wird das Original angezeigt.
Mit Rechtsklick-"Ziel speichern unter..." kann man das Bild vom TXT herunterladen.

4 Brickly

4.1 Wozu braucht man Brickly?

Mit Brickly kannst du im Browser, z.B. auf einem Tablet oder einem PC, graphisch Programme erstellen, die auf dem TXT laufen. Der TXT dient dabei als Webserver, auf den du über WLAN zugreifen kannst. Die Programme können die Eingänge des TXT auslesen und seine Ausgänge, z.B. für Motoren ansteuern.



Du programmierst mit Brickly, indem du im Browser mit der Maus Bausteine zu einem Programm zusammenfügst. Um dir den Einstieg zu erleichtern, stehen fünf verschiedene Erfahrungsgrade zur Verfügung. Die Programme werden auf dem TXT gespeichert, können aber auch (mit der App BrickMCP) auf andere Rechner übertragen werden.

4.2 Installation und Start

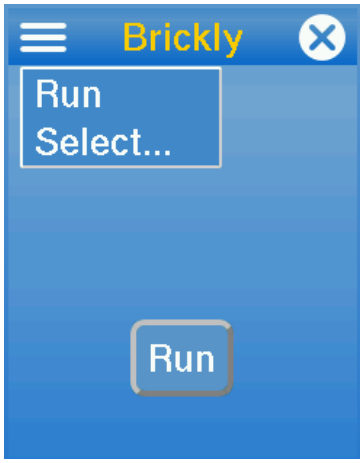
Zur Installation musst du den TXT mit der Community Firmware starten. Der einfachste Weg zur Installation von Brickly ist der App Store. Du startest die App “Store” auf dem Tochscreen deines TXT. Hier wählst du jetzt “Brickly” aus. Im Menü mit den drei Strichen oben rechst findest Du den Befehl “Install”. Hier drückst du drauf, und Brickly wird installiert.
Du kannst Brickly auch als Zip-Datei vom GitHub-Repository der fischertechnik community herunterladen. Auf dem Web-Interface siehst du unten einen Bereich “App Upload”. Hier wählst du die heruntergeladene Brickly-Zip-Datei aus und drückst dann auf den “Upload”-Knopf. Brickly erscheint nach der Installation in der Liste der installierten Apps.

4.3 Start im Web-Interface

Im Web-Interface kannst du Brickly jetzt mit diesem Symbol auswählen.

Auf der nächsten Seite kommst du mit “Open local application pages” auf die Seite von Brickly. Damit du mit Brickly programmieren kannst, muss im Browser die Ausführung von JavaScript möglich sein. Auf der Brickly-Seite kannst Du auch die Sprache auswählen, entweder Deutsch oder Englisch (mit den Flaggen-Symbolen rechts oben in der Ecke). Außerdem kannst du dein Brickly-Programm ausdrucken: wenn du auf das Drucker-Symbol rechts oben drückst, erhälst du ein Bild von deinem Programm, das du mit der Druck-Funktion deines Browsers drucken kannst.

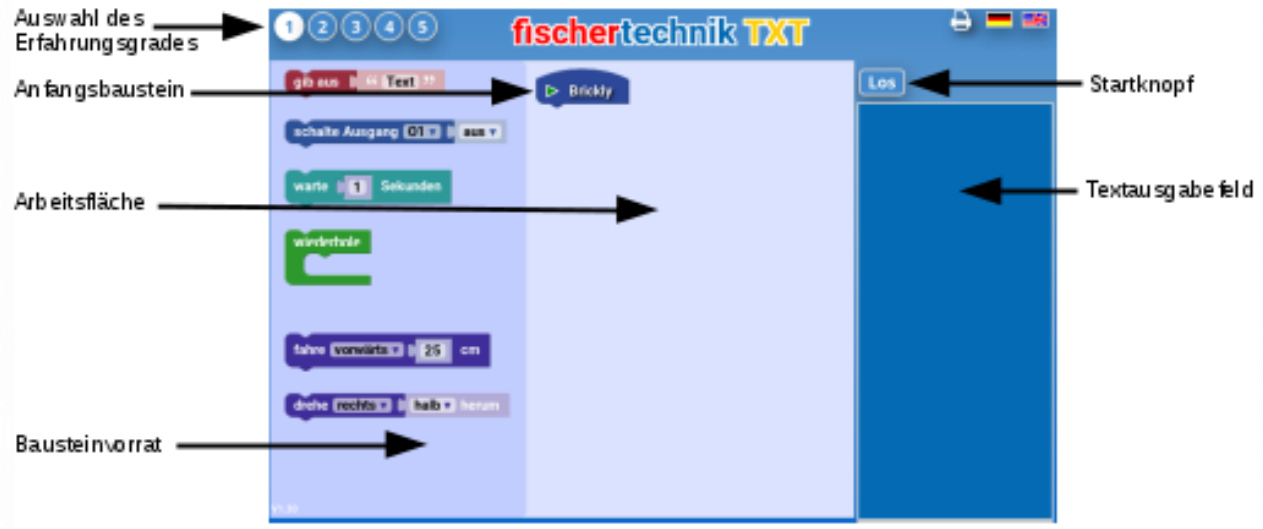
4.4 Benutzung auf dem TXT-Touch-Screen



Hier siehst du, wie Brickly auf dem TXT-Touch-Screen aussieht. Programmieren kannst du nur in dem Browser auf dem deinem PC oder Tablet. Auf dem TXT kannst du ein Programm auswählen (mit “Select…”). Wenn du auf “Run” drückst, startet das Programm. Wenn du Brickly auf dem TXT beenden willst, drückst du einfach auf das X rechts oben. Wenn du Brickly auf dem TXT beendet hast und das Browserfenster noch offen ist, kannst du durch Drücken auf “Verbinde” rechts im Browser Brickly wieder starten.

4.5 Erster Erfahrungsgrad “Anfänger”

Oben links auf der Startseite von Brickly siehst du fünf kleine Kreise mit Zahlen. Hier wählst du die 1 aus. Du kommst damit zu dem Erfahrungsgrad “Anfänger”.



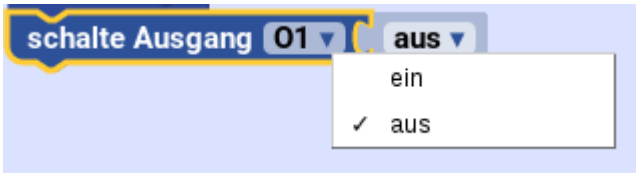
Auf der linken Seite siehst Du einige Bausteine, aus denen du dein Programm zusammenbauen kannst. In der Mitte ist deine Arbeitsfläche. Ganz rechts siehst du einen blauen Kasten. Hier wird der Text erscheinen, der von Brickly ausgegeben wird, aber auch Fehlermeldungen, wenn etwas ganz schief laufen sollte.

In der Arbeitsfläche siehst du schon einen ersten Baustein. Mit diesem Anfangsbaustein beginnt dein Programm. Ziehe mit der Maus einen Baustein von der linken Seite hierher. Du siehst, dass der Nippel auf der Unterseite des Anfangsbausteins gelb aufleuchtet. Jetzt kannst du deinen Baustein hier ablegen.

Bei manchen Bausteinen kannst du etwas auswählen. Dazu drückst du auf das kleine blaue Dreieck in dem Auswahlfeld. Wenn du eine Reihe von Bausteinen untereinander zusammengefügt hast, kannst du auf “Los” (über dem blauen Kasten auf der rechten Seite) drücken. Damit wird dein Programm auf den TXT hochgeladen und dort gleich gestartet. Die Befehle auf den Bausteinen werden jetzt von oben nach unten nacheinander abgearbeitet. Der Baustein, an dem der TXT gerade arbeitet, leuchtet auf. Wenn das Programm fertig ist, werden alle Ausgänge wieder zurückgesetzt, also z.B. alle Motoren gestoppt.

4.6 Die Bausteine im Erfahrungsgrad “Anfänger”

- gib aus <Text>Statt *Text* kannst du hier über die Tastatur an deinem Rechner einen anderen Text eingeben. Drücke dazu in das Feld und schreibe deinen Text, z.B. “Hallo TXT”. Beende die Eingabe mit der Eingabetaste auf deiner Tastatur oder indem du (mit der Maus) irgendwohin drückst. Wenn das Programm läuft, erscheint der Text in dem blauen Kasten auf der rechten Seite und auf dem Touchscreen deines TXT.



- schalte Ausgang <O1> <aus>: Mit diesem Baustein kannst du direkt einen Ausgang am TXT ansteuern, um z.B. eine Lampe leuchten zu lassen. Mit *ein* schaltest du den Ausgang ein, mit *aus* wieder aus. Achte darauf, den richtigen Ausgang auszuwählen, also den, an dem du auf dem TXT deine Lampe oder deinen Motor angeschlossen hast. Denk daran, dass am Programmende alle Ausgänge wieder ausgeschaltet werden und verwende einen *warte*-Baustein, wenn es nötig ist.
- warte <1> Sekunden: der TXT macht 1 Sekunde lang nicht mit dem Programm weiter. Du kannst hier auch eine andere Zahl eingeben. Wähle aber keine zu lange Zeit, sonst wird dir langweilig.

- wiederhole: Wenn du einen Befehl oder eine Gruppe von Befehlen nicht nur einmal, sondern immer wieder ausführen möchtest, verwendest du den *wiederhole*-Baustein. Die Befehlsbausteine schiebst du in den Bauch in dem Wiederhole-Baustein. Jetzt hast du ein Programm, das immer wieder durchläuft. Zum Stoppen des Programms musst du auf den “Stopp”-Knopf drücken. Du findest den “Stopp”-Knopf rechts oben über dem blauen Kasten, da, wo vorher der “Los”-Knopf war.
- fahre <vorwärts> <25> cm: Damit kannst du den Fahrroboter aus dem Fischertechnik TXT Discovery Set 25 cm geradeaus vorwärts fahren lassen. Statt *vorwärts* kannst du auch *rückwärts* auswählen, und du kannst eine längere Strecke eingeben. Mehr als 47 cm sind aber nicht möglich.
- drehe <rechts> <halb> herum: Damit kannst du den Fahrroboter sich auf der Stelle drehen lassen. Du kannst *rechts* oder *links* auswählen, je nachdem mit welchem Rad sich der Roboter drehen soll. *Rechts* dreht mit dem rechten Rad, so dass sich der Roboter nach links dreht. Außerdem kannst du auswählen, wie weit er sich drehen soll: *etwas* dreht um einen Achtelkreis, *halb* dreht genau zur Seite (einen Viertelkreis), *weit* noch ein Stück weiter, und bei *ganz* guckt der Roboter nach der Drehung genau in die entgegengesetzte Richtung, weil er sich um einen Halbkreis gedreht hat.

4.6.1 Kontextmenü

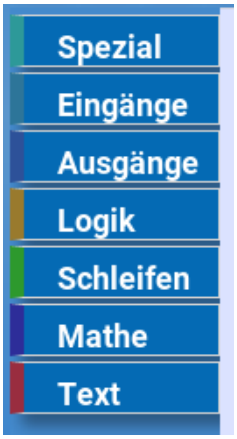
Wenn du auf die rechte Maustaste drückst, öffnet sich ein Kontextmenü. Wenn du auf einen Baustein drückst, hast du folgende drei Punkte zur Auswahl:

- “Kopieren”: damit kannst du den ausgewählten Baustein kopieren und die Kopie woanders einbauen.
- “Bausteine löschen”: Wenn du den Baustein nicht mehr brauchst, kannst du ihn von deiner Arbeitsfläche entfernen. Keine Angst: auf der linken Seite gibt es ihn noch, so dass du ihn dir wiederholen kannst.
- “Hilfe”: bringt dich zu einer Webseite, die dir weitere Informationen zu dem Baustein liefert.

Wenn du auf die leere Arbeitsfläche drückst, enthält das Kontextmenü folgende Punkte:

- “Rückgängig”: macht die letzte Aktion rückgängig.
- “Wiederholen”: wiederholt eine rückgängig gemachte Aktion, macht also rückgängig rückgängig.

4.7 Zweiter Erfahrungsgrad “Junior”



Als “Junior” hast du mehr Bausteine zur Verfügung. Damit die Auswahl übersichtlich bleibt, sind die Bausteine jetzt in Gruppen aufgeteilt. Es gibt jetzt einen Mülleimer, in den du Bausteine, die du nicht mehr haben willst, schieben kannst. Außerdem kannst du rechts unten die Geschwindigkeit auswählen, mit der der TXT das Programm abarbeiten soll: von Schildkrötengang (ganz langsam) bis zu Hasengang (ganz schnell).
Neue Bausteine in “Junior” beschäftigen sich mit Bedingungen und Logik. Eine Bedingung ist entweder *wahr* oder *unwahr*. Es gibt für diese Bedingungen beim Programmieren kein “vielleicht” oder “ich weiß nicht”. Eine Bedingung in Brickly ist z.B. Eingang <I1> ist an. Der Eingang ist an, dann ist die Bedingung wahr, oder aus, dann ist die Bedingung unwahr. Bedingungsbausteine kannst du an verschiedenen Stellen einsetzen, z.B bei wiederhole <solange>.

4.7.1 Bausteine “Spezial”

- warte <1> Sekunden
- spiele Geräusch und Flugzeug: der TXT macht das ausgewählte Geräusch, z.B. *Flugzeug*. Du kannst aus einer ganzen Reihe von Geräuschen auswählen. Den Geräusch-Baustein musst du rechts an den Befehlsbaustein anbauen.

4.7.2 Bausteine “Eingänge”

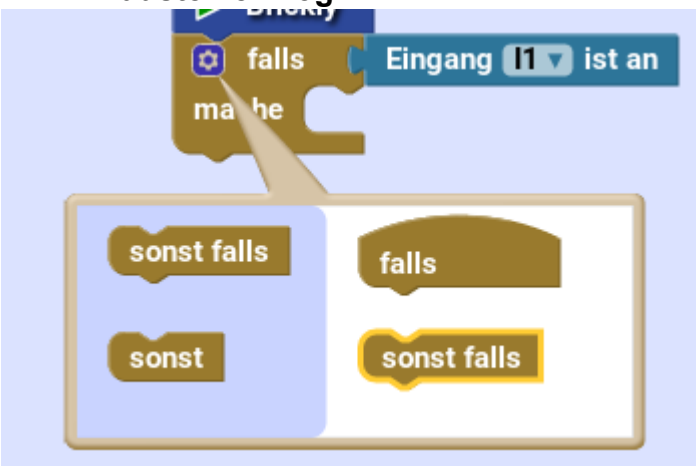
- Eingang <I1> ist an: Dies ist ein Bedingungs-Baustein, der den Zustand des ausgewählten Eingangs, z.B. *I1* abfragt.

4.7.3 Bausteine “Ausgänge”

- schalte Ausgang <O1> und ein
- fahre <vorwärts> <25> cm
- drehe <rechts> <halb> herum

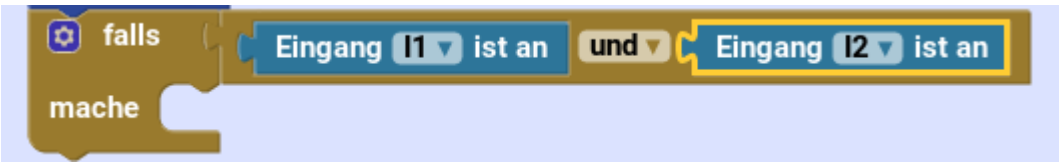
- fahre <vorwärts> <solange>: Mit diesem Baustein kannst du den Fahrroboter solange *vorwärts* oder *rückwärts* fahren lassen, wie ein Zustand dauert, also die angegebene Bedingung wahr ist. Als Zustand kannst du den Eingang <I1> ist an-Baustein nehmen, den du rechts an den fahre <vorwärts> <solange>-Baustein anbaust. Du kannst an den Eingang I1 einen Taster anschließen, und wenn du den gedrückt hälst, fährt mit diesem Befehl dein Roboter vorwärts. Statt *solange* kannst du auch *bis* auswählen. Damit kannst du einen Hinderniserkennungsroboter bauen, der solange vorwärts fährt, bis ein Hindernis den Taster an dem ausgewählten Eingang drückt.

4.7.4 Bausteine “Logik”



falls <> mache : Nach *falls* kommt eine Bedingung, und in den Bauch des Bausteins nach *mache* kommt ein Befehl oder eine Befehlsblock. Dieser Befehl wird einmal ausgeführt, falls die Bedingung wahr ist.
Wenn du auf das Zahnrad links oben in diesem Baustein drückst, kannst du den Baustein noch erweitern. Es öffnet sich ein Kasten, der auf der linken Seite noch zwei Bausteine enthält. Wenn du einen der Bausteine an den ‘falls’-Baustein auf der rechten Seite anbaust, erhälst du einen erweiterten ‘falls’-Baustein. Wenn du mit dem Zusammenbau fertig bist, drückst du wieder auf das Zahnrad und der Kasten verschwindet. Nach *sonst falls* kannst du noch eine Bedingung und dann wieder einen Befehl einsetzen. Nach *sonst* kommt dann der Befehl, den der TXT ausführen soll, wenn alle anderen Bedingungen in diesem Baustein unwahr sind.

- <> <und> <>: Hier kannst du zwei Bedingungs-Bausteine miteinander zu einer neuen



  Bedingung

  verknüpfen. Mit *und* müssen beide Bedingungen wahr sein, damit der Baustein den Wert *wahr* hat. Mit *oder* muss mindestens eine Bedingung wahr sein. Wenn beide Bedingungen unwahr sind, hat der Baustein auch den Wert *unwahr*. Diesen Baustein kannst du immer dort einsetzen, wo eine Bedingung gebraucht wird.

- nicht: diesen Baustein kannst du vor einen Bedingungs-Baustein einbauen, und er dreht diese Bedingung um: aus *wahr* wird *unwahr* und aus *unwahr* wird *wahr*.

4.7.5 Bausteine “Schleifen”

- wiederhole
- wiederhole <10> mal: mache: Hier gibst du die Anzahl der Wiederholungen an. Dein TXT führt dann die Befehle in dem Bauch des Bausteins 10 mal nacheinander aus. Du kannst auch eine andere Zahl eingeben.
- wiederhole <solange>: Hier gibst du eine Bedingung für die Wiederholung an. Wenn du *solange* auswählst, werden die Befehle nur dann ausgeführt, wie eine Bedingung wahr ist, dann aber immer wieder. Du kannst auch *bis* auswählen. Dann werden die Befehle so lange immer wieder ausgeführt, wie die angebene Bedingung wahr ist.

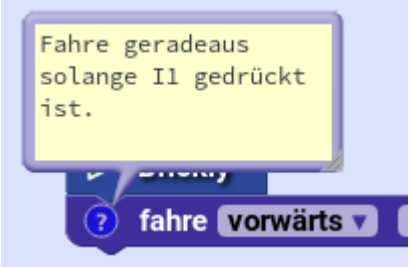
4.7.6 Bausteine “Mathe”

- <0>: ein Zahlenbaustein. Statt 0 kannst du eine andere Zahl eingeben. Diesen Baustein kannst du überall dort verwenden, wo eine Zahl gebraucht wird.
- <1> <+> <1>: ein Zahlenbaustein mit Rechenergebnis. Es gibt plus (+), minus (-), mal (x), geteilt (÷), hoch (^).

4.7.7 Bausteine “Text”

- "< >": ein Textbaustein, den du über die Tastatur mit Text füllen kannst.
- erstelle Text aus: setzt zwei Texte zu einem Text zusammen.
- gib aus "< >": wie hier. Du kannst hier einen Text eingeben, oder einen Textbaustein oder einen Zahlenbaustein einbauen. Wenn du einen Bedingungsbaustein anbaust, wird “true” für *wahr* oder “false” für unwahr ausgegeben.

4.7.8 Neu im Kontextmenü

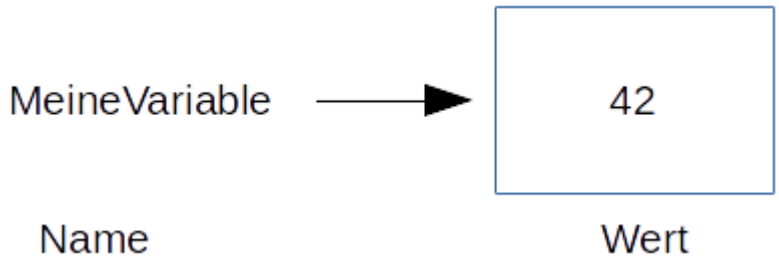


•“Kommentar hinzufügen”: Hier kannst du einen Kommentar zu deinem Baustein oder dem Programm schreiben oder auch wieder löschen. Wenn du den Kommentar fertig geschrieben hast oder zum Lesen eines Kommentares, drückst du auf das kleine Fragezeichen.

- “Baustein zusammenfalten”: Damit wird ein Baustein zusammengefaltet und platzsparender dargestellt. Du kannst ihn auch wieder entfalten.
- “Baustein deaktivieren”: deaktiviert den Baustein, so dass er nicht mehr im Programm verwendet wird. Aktivieren macht das wieder rückgängig.

4.8 Dritter Erfahrungsgrad “Fortgeschritten”

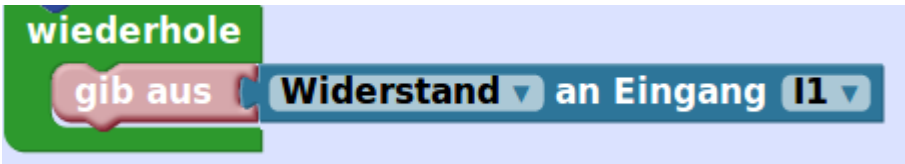
Für die Fortgeschrittenen gibt es neue Bausteine für die Nutzung der Eingänge, und du lernst Variablen kennen. “Variablen sind Behälter im Speicher, die einen Namen haben und deren Inhalt sich während der Ausführung eines Programms ändern kann. Der Begriff der Variablen ist eines der wichtigsten Konzepte in Mathematik und Informatik. Mit der Verwendung von Variablen fängt das “wahre Programmieren” an.” (Rüdeger Baumann) Mit dieser Erklärung im Hinterkopf kannst du dir überlegen, wie du Variablen in deinem Programm nutzen kannst, z.B. um den Wert eines Eingangs abzuspeichern. Variablen sind auch nützlich, um an verschiedenen Stellen im Programm immer den gleichen Wert zu verwenden.



Als erstes musst Du eine Variable erstellen. Danach hast du Bausteine zur Verfügung, mit denen du die Variable nutzen kannst.

Außerdem kannst du jetzt verschiedene Programme mit unterschiedlichen Namen speichern. Dazu drückst du das Symbol mit den drei Strichen ganz rechts im Browser. Dann gibst du einen Namen für dein Programm ein, drückst auf “Neu” und schon hast du einen neuen Startbaustein mit einer leeren Arbeitsfläche zur Verfügung. Deine auf dem TXT gespeicherten Programme kannst du mit dem gleichen Symbol auswählen.

4.8.1 Neue Bausteine bei “Eingänge”

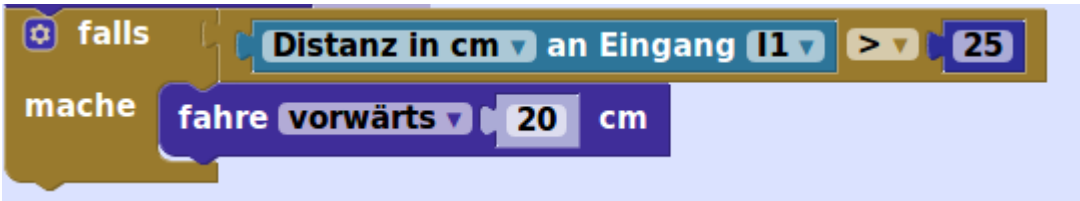


- <Spannung (mV)> an Eingang <E1>: mit dem Eingangsbaustein kannst du einen der Eingänge an dem TXT abfragen. Was dir der Eingang liefert und welche physikalische Größe (*Spannung*, *Schalterzustand*, *Widerstand* oder *Distanz*) du auswählen musst, hängt von dem verwendeten Sensor ab. Du kannst den Eingangsbaustein als Zahlenbaustein verwenden. Bei den Widerständen musst du einen Moment warten, bis sie ihren endgültigen Wert erreicht haben.
    - Farbsensor: liefert eine elektrische *Spannung*, je nach Farbe unterschiedliche Werte.
    - IR-Spursensor: wird an zwei Eingänge angeschlossen, liefert an beiden Eingängen einen *Schalterzustand*. Der IR-Spursensor befindet sich über einer einer schwarzen Spur, wenn beide Eingänge den Wert 1 für *ein* liefern.
    - Taster: liefert einen *Schalterzustand*, entweder 0 für *aus* oder 1 für *ein* (umgekehrt, wenn die Anschlüsse 1 und 2 am Taster verwendet werden).
    - Fototransistor: liefert einen *Schalterzustand*, 0 für kein Licht oder 1 für Licht, d.h. Lichtschranke offen.
    - Fotowiderstand: liefert einen *Widerstand*, der mit zunehmender Helligkeit sinkt.
    - Ultraschallsensor: misst die *Distanz*, also den Abstand zum nächsten Hindernis.
    - Wärmesensor: liefert einen *Widerstand*, der mit steigender Temperatur sinkt.
- Temperatur <°C>: diesen Baustein kannst Du verwenden, um aus dem Widerstandswert des Wärmesensor eine Temperatur zu berechnen. Dazu baust du ihn vor den Eingangsbaustein, bei dem du *Widerstand* auswählst. Du kannst die Einheit wählen: °C (Grad Celsius, die in Europa verwendete Einheit), °F (Fahrenheit, wird in den USA verwendet) und K (Kelvin, gibt die absolute Temperatur an).

4.8.2 Neue Bausteine bei “Ausgänge”

- <100%> (ein): mit diesem Baustein kann man wählen, ob der Ausgang mit voller Stärke (100%) oder nur mit weniger Stärke betrieben werden soll. Damit kannst du einen Motor langsamer laufen lassen oder ein Lämpchen weniger hell leuchten lassen.
- drehe <rechts> <90°>: wie hier. Jetzt kannst du aber den Winkel genau in Grad angeben. Damit du dir unter dem angegeben Winkel etwas vorstellen kannst, erscheint ein kleiner Kreis, auf dem du mit der Maus entlang fahren kannst, um den Winkel einzustellen.

4.8.3 Neue Bausteine bei “Logik”



- <> <=> <>: Hier kannst du zwei Werte, z.B. zwei Zahlenbausteine miteinander vergleichen. An die erste Stelle baust du den einen Baustein, dann kannst du den Vergleich auswählen: = (gleich), ≠ (ungleich), < (kleiner als), ≤ (kleiner oder gleich), > (größer als), ≥ (größer oder

gleich). An die letzte Stelle kommt der zweite Wert. Du kannst Variablen, Text, Zahlen (z.B. von Eingängen) und Wahrheitswerte vergleichen.

- wahr: ein Bedingungsbaustein, bei dem du *wahr* oder *unwahr* auswählen kannst. Damit kannst du z.B. einer Variablen einen Startwert geben.

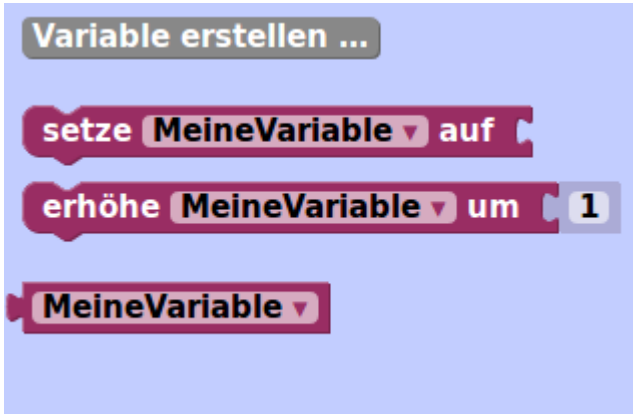
4.8.4 Neuer Baustein bei “Schleifen”

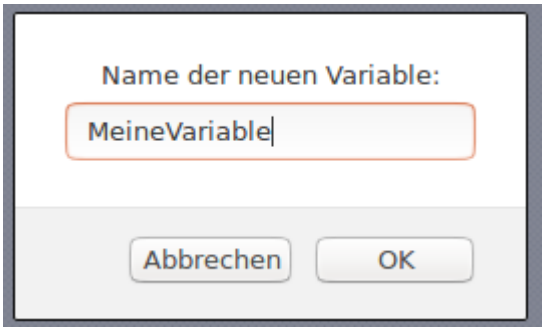
- zähle <i> von <1> bis <10> in Schritten von <1>: mache: Das ist eine Zählschleife. Die Anzahl der Durchläufe, also wie oft die Befehle im Bauch des Bausteins ausgeführt werden, steht fest. Beim ersten Durchlauf ist der Zähler *i* 1, wird dann um 1 erhöht, usw. bis *i* 10 ist. Mit diesen Zahlen wird diese Schleife also 10 mal ausgeführt. Der Zähler (hier *i*) ist im Bauch des Bausteins bekannt. Du findest einen Zahlenbaustein dafür bei den Variablen. Damit kannst du z.B. zuerst nur 1 Sekunde warten, dann 2 usw..

4.8.5 Neuer Baustein bei “Mathe”

- ganzzahlige Zufallszahl zwischen <1> und <100>: hier denkt sich der TXT eine ganze Zahl zwischen 1 und 100 aus. Wenn du als Grenzen 1 und 6 wählst, hast du eine Art Würfel.

4.8.6 “Variablen”





Alles, was du für die Verwendung von Variablen brauchst, findest du in dieser Gruppe.

- Variable erstellen: Wenn du auf diesen Baustein gedrückt hast, erscheint ein Fenster, in dem du den Namen deiner Variablen eingibst. Damit ist die Variable erstellt. Wähle einen möglichst aussagekräftigen Namen, unter dem sich auch ein anderer etwas vorstellen kann.
- setze <meineVariable> auf: Hier kannst einen Eingangsbaustein oder einen Zahlenbaustein anbauen und damit einen Wert für deine Variable speichern.
- erhöhe <meineVariable> um <1>: mit diesem Baustein kannst du zu dem Wert deiner Variablen 1 oder eine andere Zahl dazuzählen. Das ist praktisch, wenn du Ereignisse zählen willst, z.B. wie oft dein Roboter gegen ein Hindernis gefahren ist.
- <meineVariable>: mit diesem Baustein kannst du deine Variable als Zahlenbaustein verwenden und in anderen Bausteine, z.B. in Textbausteine einbauen.

4.8.7 Neu im Kontextmenü

- “Externe Eingänge”: ändert die Darstellung des Bausteins, so dass alle eingebauten Bausteine rechts angebaut sind.
- “Erzeuge “Schreibe u””: Mit dieser Funktion kannst du sehr schnell das Gegenstück zu dem Baustein erzeugen, z.B. setze Variable auf, wenn du auf einer Variablen bist.

4.9 Vierter Erfahrungsgrad “Senior”

In diesem Erfahrungsgrad gibt es viele neue Bausteine für die Nutzung von Encoder-Motoren. Alle Motorbausteine findest du jetzt in einer eigenen Untergruppe “Motoren” unterhalb von “Ausgänge”. Außerdem kannst du jetzt eine Gruppe von Befehlen zu einer Funktion zusammenbauen. Damit wird ein langes Programm viel übersichtlicher.

4.9.1 “Ausgänge” - “Motoren”

- setze Motor <M1> <Geschwindigkeit> auf: mit diesem Baustein kannst du den ausgewählten Motor (M1, M2, M3 oder M4) genau ansteuern. Je nach ausgewählter Eigenschaft (*Geschwindigkeit*, *Richtung*, *Entfernung*, *Typ*) musst du noch einen der folgenden drei Bausteine anbauen. Für die *Entfernung* baust du noch einen Zahlenbaustein an. Die *Geschwindigkeit* musst du für jeden Motor auf jeden Fall angeben, sonst bewegt er sich nicht. Der *Geschwindigkeit*-Baustein startet den Motor. **Wichtig**: Wenn der Motor gestartet ist, läuft das Programm weiter. Wenn es am Ende angekommen ist, werden alle Motoren ausgeschaltet. Es kann daher sein, dass der Motor anhält, bevor er die angegebene Entfernung zurückgelegt hat. Du musst also deinem Motor Zeit geben, wenn er eine bestimmte Entfernung zurücklegen soll, z.B. in dem du einen warte-Baustein einbaust.
- <links>: gibt die *Richtung* an, in die sich der Motor drehen soll (*rechts* oder *links*).
- <neuer Encodermotor (TXT)>: entweder *neuer Encodermotor* oder *alter Encodermotor*. Der Typ des Encodermotors ist wichtig, wenn du die Entfernung angibst. Die beiden Motoren unterscheiden sich nämlich in der Anzahl der Impulse pro Umdrehung der Antriebswelle. Wenn du den *Typ* deines Motors richtig setzt, wird dieser Unterschied berücksichtigt.
- <100% (ein): gibt die *Geschwindigkeit* an.
- stoppe Motor <M1>: stoppt den ausgewählten Motor.
- Motor <M1> hat gestoppt: dieser Bedingungsbaustein wird wahr, wenn der Motor angehalten hat, weil die angegebene Entfernung erreicht wurde. Das funktioniert nur bei einem Encodermotor, wenn die Entfernung angegeben ist.
- kopple Motoren <M1> und <M2>: zwei gekoppelte Motoren laufen gleichzeitig und mit gleicher Geschwindigkeit. Damit beide Motoren wirklich synchron laufen und auch zur gleichen Zeit ein Stopp-Signal geben, solltest Du bei beiden Motoren die gleiche Entfernung einstellen. Die Motoren können aber in unterschiedliche Richtung drehen: dann dreht sich der Fahrroboter.
- fahre <vorwärts> <25> cm
- fahre <vorwärts> <solange>
- drehe <rechts> <90>

4.9.2 Neu in “Schleifen”

- <die Schleife abbrechen>: mit diesem Baustein kannst du den Durchlauf durch eine Schleife abbrechen. Das Programm springt dann sofort zu dem ersten Baustein nach der Schleife. Du kannst diesen Baustein nach einer Bedingung in einer Schleife einbauen, um z.B. in einer Zählschleife im Notfall abzubrechen. Du kannst auch auswählen *sofort mit dem nächsten Schleifendurchlauf fortfahren*, wenn in einer Schleife einige Befehle unter bestimmten Bedingungen nicht ausgeführt werden sollen. **Wichtig**: beide Bausteine funktionieren nicht bei der “Wiederhole”-Schleife (die ohne “solange”, “bis” oder “10 mal”). Die ist eine Endlosschleife, die nur mit einem Drücken auf “Stopp” abgebrochen werden kann.

4.9.3 Neu in “Mathe”

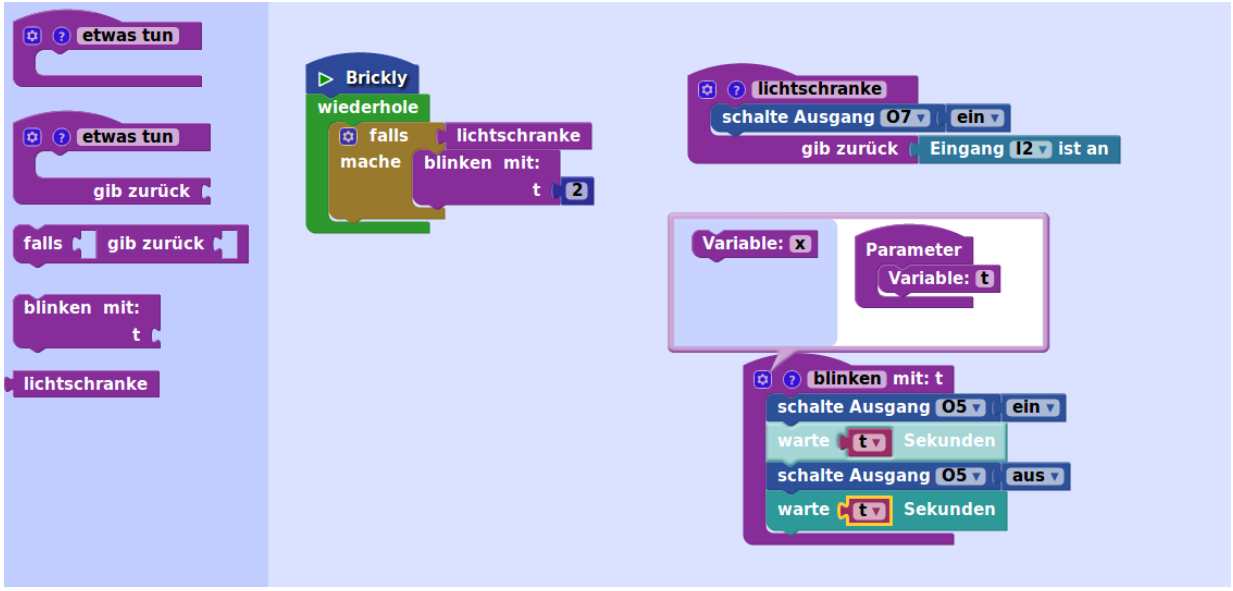
- <Quadratwurzel> berechnet die ausgewählte mathematische Funktion (*Quadratwurzel*, *Betrag*, -, *ln*, *log10*, *e^* oder *10^*) aus dem angebauten Zahlenbaustein.
- Rest von <> ÷ <>: berechnet den Rest der Division der 1. Zahl durch die 2. Zahl (Modulo-Funktion). Ein Beispiel: Bei einer geraden Zahl ist der Rest der Division durch 2 gleich 0. Damit kannst du in einer Schleife einen Befehl nur in jedem zweiten Durchlauf ausführen.

4.9.4 Neu in “Text”

- an <etwas> Text anhängen: hängt an die Variable *etwas* oder eine andere ausgewählte Variable den Inhalt des angehängten Textbausteines an.

4.9.5 “Funktionen”

Eine Funktion ist eine Gruppe von Befehlen mit einem eigenen Namen.



Du musste eine Funktion zuerst definieren und kannst sie dann verwenden.

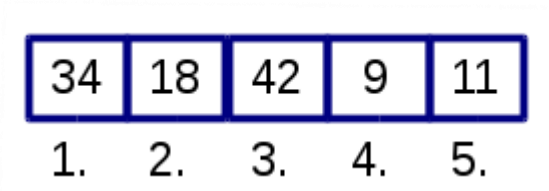
- etwas tun: Damit definierst du deine Funktion. Du legst den Baustein einfach auf der Arbeitsfläche ab. In den Bauch baust du die Befehlsbausteine, die deine Funktion ausführen soll. Statt *etwas tun* solltest du der Funktion einen sinnvollen Namen geben. Du kannst zusätzlich noch einen Text schreiben, der die Funktion beschreibt, wenn du auf das kleine Fragezeichen neben dem Namen drückst. Wenn du den Funktionsbaustein auf der Arbeitsfläche abgelegt hast, findest du in dieser Gruppe auch einen Baustein mit dem Namen deiner Funktion. Das ist der Funktionsaufruf. Diesen Baustein baust du in dein Programm ein. Damit wird deine Funktion aufgerufen, d.h. die Befehle im Bauch des Funktionsbausteins werden ausgeführt. Du kannst der Funktion auch einen oder mehrere Parameter hinzufügen. Das ist ein Wert, den das Programm der Funktion beim Aufruf

übergibt und den du innerhalb der Funktion nutzen kannst. Dazu drückst du auf das kleine Zahnrad links oben in dem Funktionsbaustein und ziehst dann den Baustein *Variable* in den Bauch des Funktionsbausteins auf der rechten Seite des Kastens. Gib auch den Parametern sinnvolle Namen! Beim Aufruf musst du dann einen passenden Wertbaustein an den Aufruf anbauen.

- etwas tun: gib zurück: Diese Funktion kann nicht nur einfach Befehle ausführen, sondern auch einen Wert zurückgeben (z.B. den Messwert eines Sensors). Den Funktionsbaustein verwendest du dann beim Aufruf wie einen Wertbaustein. Auch dieser Funktion kannst du Parameter hinzufügen.
- falls: gib zurück:: Dieser Block darf nur innerhalb eines Funktionsblocks genutzt werden. Nach *falls* kommt eine Bedingung. Wenn diese Bedingung *wahr* ist, beendet sich die Funktion sofort und gibt den angegebenen Wert zurück. Damit kannst du z.B. eine Fehlermeldung zurückgeben, wenn etwas schief gelaufen ist.

4.10 Fünfter Erfahrungsgrad “Experte”

Dies ist der höchste Erfahrungsgrad für die Experten. Neu sind hier die Listen-Bausteine. Eine Liste ist eine durchnummerierte Reihe von Elementen. Über die Nummer kannst du auf jedes Element zugreifen.



Die meisten Bausteine für Listen findest du in der neuen Gruppe “Listen”, es gibt aber auch einen neuen Baustein unter “Schleifen” und einen bei “Mathe”.

4.10.1 Neue Bausteine in “Eingänge”

Hier gibt es einige neue Bausteine für die Nutzung eines USB-Joysticks. Die Joysticks/Gamepads werden per USB am TXT angeschlossen - am Standard-USB-Port. Zur Zeit funktionieren leider nur einige Typen, z.B. ein Speedlink Competition Pro USB Joystick.

4.10.2 Neu in “Ausgänge”

- gleichzeitig. Mit diesem Baustein kannst du Ausgänge gleichzeitig setzen, z.B. zwei Motoren gleichzeitig starten, ohne gekoppelte Encoder-Motoren zu verwenden. Der Baustein funktioniert nicht mit den Bausteinen aus der Untergruppe “Mobil” (da drehen zwei Motoren ohnehin gleichzeitig).

4.10.3 Neue Untergruppe in “Ausgänge”: “Mobil”

- fahre <vorwärts> <25> cm
- fahre <vorwärts> <solange>
- drehe <rechts> <90>
- <90>° und <etwas> herum: diese beiden Bausteine können an den drehe-Baustein angebaut werden, um den Drehwinkel einzustellen.
- Mobilroboter-Konfiguration: Wenn du einen anderen Fahrroboter als den aus dem TXT-Discovery-Set gebaut hast, kannst du mit diesem Baustein am Anfang deines Programms einstellen, an welchen Anschlüssen die Motoren angeschlossen sind, von welchem Typ sie sind, wie die Antriebsübersetzung (Motorachse zu Radachse) ist, wie groß der Raddurchmesser ist und wie der Abstand zwischen den Antriebsrädern ist. Mit der richtigen Konfiguration sollen die Befehle zum Fahren und Drehen auch für deinen selbstkonstruierten Fahrroboter funktionieren. Für das Discovery-Set wird dieser Baustein nicht benötigt.

4.10.4 Neu in “Logik”

- null: der Wert, den eine Variable hat, die zwar einen Namen hat, aber für die kein Wert gesetzt wurde.
- prüfe: falls wahr: falls unwahr:: ein Wertbaustein, mit dem man den Wert auswählen kann. Nach *prüfe* kommt eine Bedingung. Wenn diese *wahr* ist, wird der Wert zurückgegeben, der nach *falls wahr* steht, sonst der Wert nach *falls unwahr*. (In der Informatik heißt das auch “Ternärer Operator” oder “Auswahloperator”).

4.10.5 Neu in “Schleifen”

- für jeden Wert <i> aus der Liste: mache: diese Schleife arbeitet sich von vorne bis hinten durch die Liste durch. Im Bauch der Schleife kannst du mit dem aktuellen Element (es heißt *i*) Befehle ausführen. Diese Befehle ändern die Liste selbst aber nicht.

4.10.6 Neu in “Mathe”

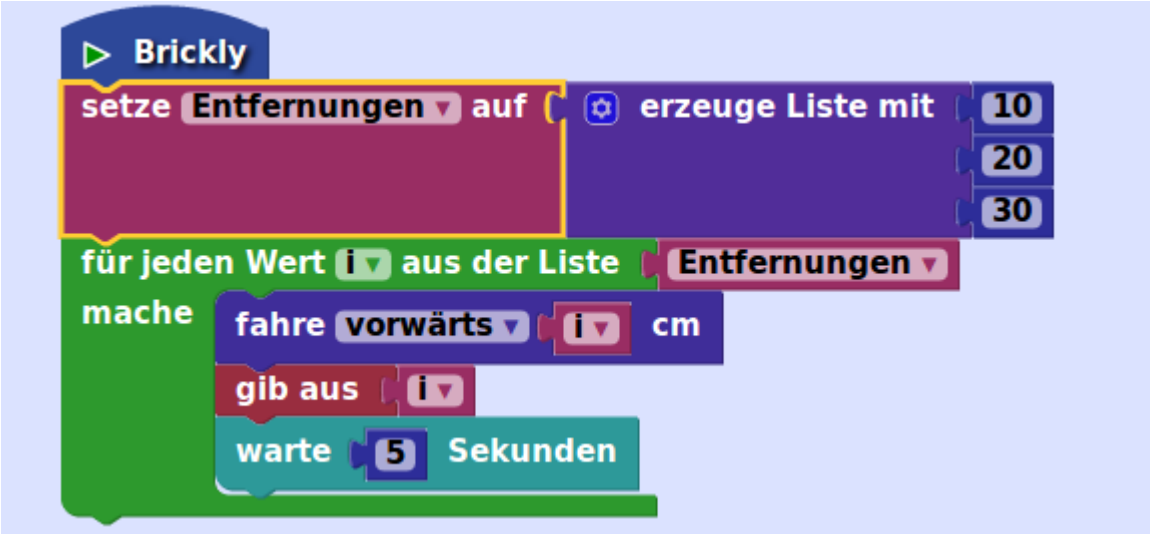
- <sin> <45> stellt die Winkelfunktionen *sin*, *cos* und *tan* und die Arkusfunktionen *asin*, *acos* und *atan* zur Verfügung.
- <π>: liefert die ausgewählte mathematische Konstante: die Kreiszahl π, die Eulersche Zahl e, den Goldenen Schnitt φ, die Quadratwurzel von 2 sqrt(2), die Quadratwurzel von 1/2 sqrt(1/2) oder unendlich ∞.
- <0> ist <gerade>: ein Bedingungsbaustein, der prüft, ob die Zahl (hier 0) die ausgewählte Eigenschaft hat: *gerade*, *ungerade*, *eine Primzahl*, *eine ganze Zahl*, *positiv*, *negativ* oder *teilbar durch* (braucht noch eine 2. Zahl).
- <runde> <3.1> rundet die angegebene Zahl. Du kannst auch *runde auf* (aufrunden zur nächstgrößeren ganzen Zahl) oder *runde ab* (abrunden zur nächst kleineren ganzen Zahl) wählen.
- <Summe über die> Liste: gibt die Summe aller Elemente der ausgewählten Liste aus. Du kannst eine andere Funktionen auswählen, die dann eine andere Berechnung über alle Elemente durchführt (*Minimalwert*, *Maximalwert*, *Mittelwert*, *Median*, *am häufigsten*, *Standardabweichung*, *Zufallswert*).
- begrenze <50> zwischen <1> und <100>: verwendet die angegebene Zahl (hier 50), wenn sie zwischen den beiden anderen Zahlen liegt. Wenn sie zu groß ist, wird die obere Grenze genommen (hier 100), und wenn wenn sie zu klein ist, die untere Grenze (hier 1). Sinnvollerweise nimmt man als Zahl hier eine Zahlenvariable.
- Zufallszahl (0.0 - 1.0): liefert eine zufällige Gleitkommazahl zwischen 0.0 und 1.0.

4.10.7 Neu in “Text”

Hier findest du jetzt viele neue Bausteine, um Texte zu bearbeiten, Teile daraus zu finden usw.. Als “Experte” wirst du in dieser Gruppe finden, was du brauchst. Ein Baustein ist besonders nützlich:

- gib aus in <> <"abc">: gibt den angegebenen Text in der ausgewählten Farbe aus.

4.10.8 “Listen”

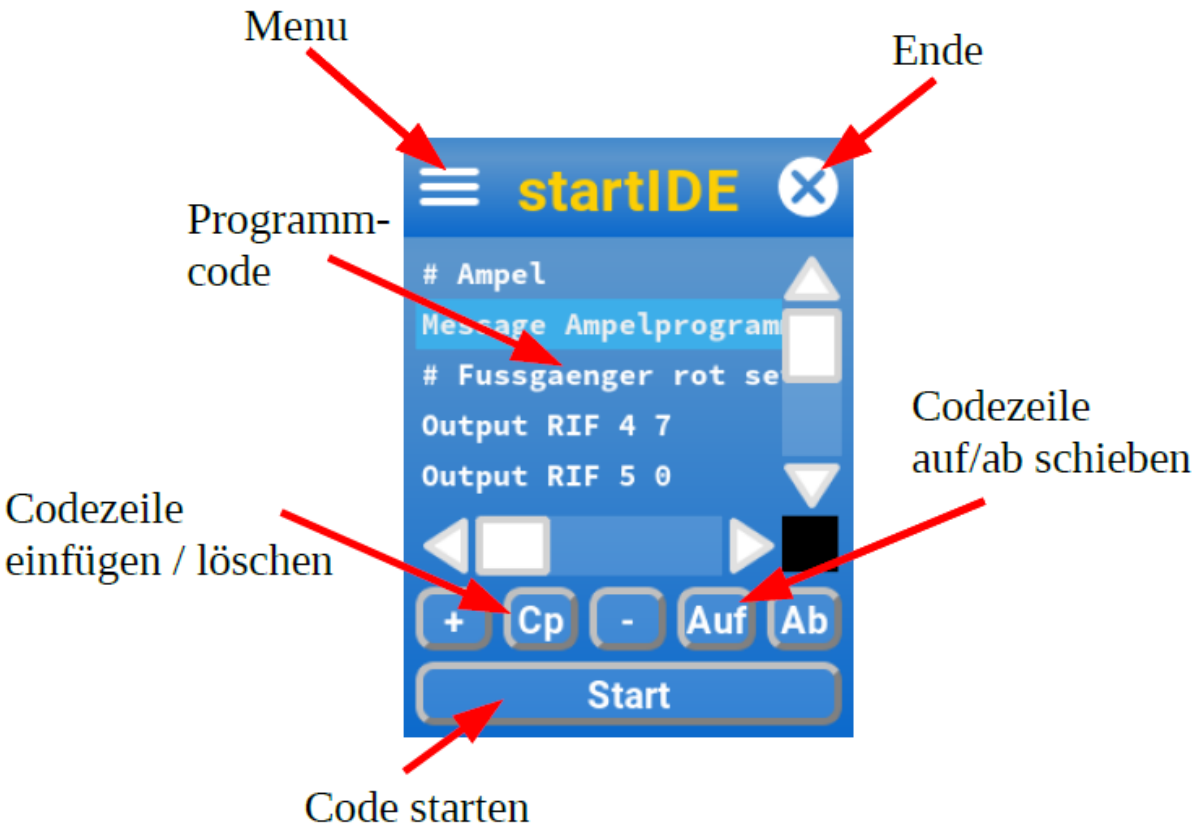


Um eine Liste zu erzeugen, erstellst du am einfachsten zuerst eine Variable für die Liste (z.B. “MeineListe”). Dann erzeugst du eine Liste mit einem der folgenden Bausteine und weist ihn der Variable zu. Die Variable mit deiner Liste findest du unter “Variablen”.

- erzeuge eine leere Liste: erzeugt eine leere Liste.
- erzeuge eine Liste mit: : erzeugt eine Liste aus den angegeben Bausteinen. Durch Drücken auf das Zahnrad erhälst du wieder einen Kasten, mit dem du durch Hinzufügen oder Zurückschieben der Elementbausteine (*etwas*) mehr oder weniger Elementplätze in deiner Liste bereitstellen kannst. Wenn du die richtige Anzahl hast, drückst du wieder auf das Zahnrad. Um deine Liste zu füllen, baust du an die freien Plätze Wert-Bausteine an, z.B. Variablen, Zahlen oder Texte.
- erzeuge eine Liste mit <5> mal dem Element <>: erzeugt eine Liste mit der angegebenen Länge (hier 5), die an jeder Stelle das gleiche Element enthält.
- Länge von: : gibt die Länge der angebauten Liste (d.h. die Anzahl der Elemente) als Zahl aus.
- <> ist leer: ein Bedingungsbaustein, liefert wahr, wenn die eingebaute Liste leer ist, d.h. keine Elemente enthält.
- in der Liste <Liste> suche <erstes> Auftreten von <>: gibt die *erste* (oder *letzte*) Position zurück, an der sich das angegeben Element befindet. Der Wert ist 0, wenn kein passendes Element in der Liste ist.
- in der Liste <Liste> <nimm> <#tes> <>: liefert das Element an der angegebenen Position (Zahl nach *#tes*) in der Liste. Du kannst auch auswählen *nimm und entferne*, dann wird das zurückgelieferte Element aus der Liste gelöscht, die Liste wird kürzer. Mit *entferne* wird auch das angegebene Element gelöscht, aber du erhälst keinen Wert zurück. Du kannst statt von vorne auch von hinten zählen (*#tes von hinten*), oder *erstes*, *letztes* oder *zufälliges* Element wählen. Achte darauf, dass die Positionsangabe einen Wert zwischen 1 und der Länge deiner Liste ergibt, sonst gibt das Programm eine Fehlermeldung.
- in der Liste <Liste> <setze für> <#tes> ein <>: hier kannst du das Element an der angegebenen Position überschreiben, oder mit *füge ein* ein neues Element einbauen. Beim Einfügen wird die Liste länger.
- in der Liste <Liste> erhalte Unterliste <von #> <> bis <zu #> <>: gibt alle Elemente zwischen den ausgewählten Positionen aus. Wenn die zweite Position vor der ersten liegt, ist die Unterliste leer.
- <Liste aus Text> erstellen <> mit Trennzeichen <",">: erstellt eine Liste von Texten aus einem längeren Text, wobei die einzelnen Teile durch das angegebene Zeichen, z.B. ein Komma, getrennt sind.
- <nummerisch> <aufsteigend> <> sortieren: liefert eine sortierte (*nummerisch*, *alphabetisch* (mit Großbuchstaben zuerst) oder *alphabetisch, Großschreibung ignorieren*) Kopie der angegebenen Liste. Die Liste selbst bleibt unverändert. Wenn die sortierte Kopie behalten möchtest, musst du die Variablenzuweisung setze <meineVariable> auf und diesen Baustein verwenden.

5 startIDE

startIDE ist eine Programmier-App für die community firmware des TXTControllers (auch für den TX-Pi und RIF (Robo Interface Familie)), mit der sich eine Vielzahl einfacher Modelle programmieren last.



- Die wesentlichen Elemente des Hauptbildschirms sind in der Abbildung gekennzeichnet.
- Mit dem Kreuz-Knopf („Ende “) wird startIDE beendet. Der aktuelle Programmcode geht nicht verloren und steht beim nächsten Aufruf von startIDE wieder zur Verfügung.
- Mit dem Plus- und Minus-Knopf wird eine neue Programmzeile eingefugt bzw. die aktuelle Zeile gelöscht. Zum Loschen ist ein schnelles zweimaliges Antippen des Minus-Knopfs erforderlich.
- Der Cp- Knopf dupliziert die aktuelle Befehlszeile (Copy).
- Doppelklick auf eine Programmzeile erlaubt das edititieren
- Mit dem Up- bzw. Down-Knopf wird die aktuelle Zeile nach oben bzw. unten verschoben.
- Der „Start “-Knopf schließlich startet die Ausführung des Programmes.
-

5.1 Menü

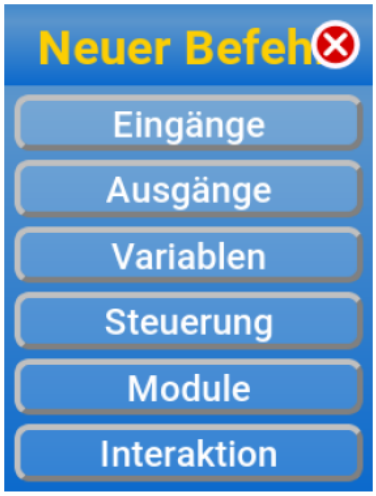
Projects New für neues Projekt
Load zum laden eines vorher gespeicherten Projekte
Save das aktuelle Projekt zu speichern
Delete ein Projekt dauerhaft zu löschen

Modules Import Programmodule von SD-Karte dazu zuladen
Export Programmodule auf die SD-Karte zu exportieren
Delete Module von der SD-Karte dauerhaft zu löschen

Interfases öffnet eine Benachrichtigung, die das Hardware Interface (TXT und/oder RoboInterface Familie und ftduino)anzeigt

About

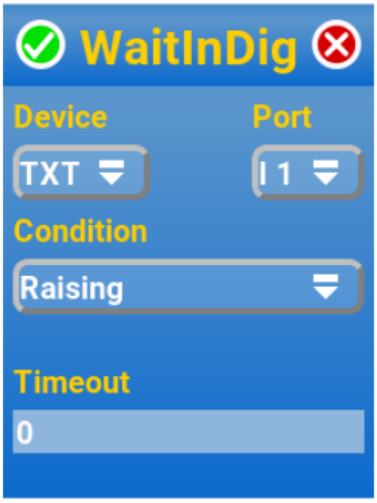
5.2 Funktionen



Über den „+ “-Knopf auf dem Hauptbildschirm Können Funktionen aus den folgenden Gruppen in den Programmcode eingefügt werden.
Mit dem Kreuz-Knopf oben rechts kann die Auswahl abgebrochen werden, ohne eine neue Code-Zeile einzufügen.
Soweit die Funktionen Parameter benötigen, werden diese jeweils in entsprechenden Bildschirmdialogen konfiguriert.
Dabei können in vielen Fällen, in denen Zahlen als Parameter erwartet werden, auch Variablen verwendet werden.

5.3 Eingänge

5.3.1 WaitForInputDig (Anfrage I1 bis I8)



WaitInDig TXT 1 Raising 500

TXT	Hardware TXT oder RIF für ->
FTD	Hardware ftduino
1	Nummer des Einganges
Raising	Raising - fallende Signalflanke (Öffner)
	Falling - steigende Signalflanke (Schließer)
500	Timeout in ms (0 = unendlich lange Wartezeit)

5.3.2 If InputDig (Zustandsabfrage I1 bis I8)

IfInDig TXT 2 False ende

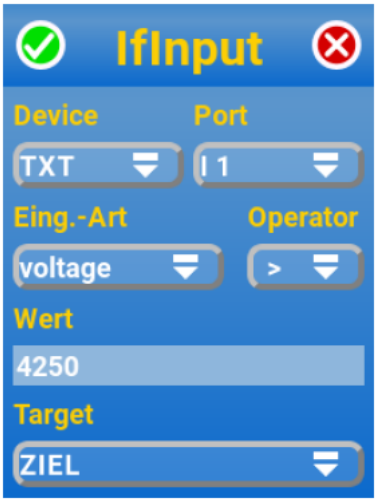
TXT/FTD	Hardware
2	Nummer des Einganges
False	False - Eingang offen
	True - Eingang geschlossen
ende	Sprungmarke, die angesprungen werden soll

5.3.3 WaitForInput (Wartet auf eintreten eines Zustandes bei Analog-Eingang)

WaitIn TXT 1 D < 50 2500

TXT/FTD	Hardware
1	Nummer des Einganges
D	V – Spannung; S – Digitaleingang
	D - Ultraschall-Abstandssenor; R – Widerstand
<	< kleiner als Vergleichswert
	== gleich Vergleichswert
	!= ungleich Vergleichswert
	> größer als Vergleichswert
50	Vergleichswert
2500	Timeout in ms (0 = unendlich lange Wartezeit)

5.3.4 IfInput (Abfrage analoger Werte mit Sprungmarke)



WaitIn TXT 1 V < 5 ende

TXT/FTD	Hardware
1	Nummer des Einganges
V	V – Spannung; S – Digitaleingang
	D - Ultraschall-Abstandssenor; R – Widerstand
<	< kleiner als Vergleichswert
	== gleich Vergleichswert
	!= ungleich Vergleichswert
	> größer als Vergleichswert
5	Vergleichswert
ende	Sprungmarke, die angesprungen werden soll

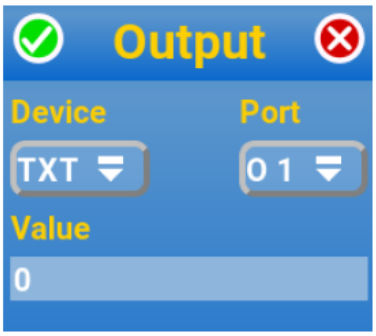
5.3.5 QueryInput (Zustandsabfrage mit Textausgabe)

QueryInput TXT 1 S T1 offen

TXT/FTD	Hardware
1	Nummer des Einganges
S	V – Spannung
	S – Digitaleingang
	D - Ultraschall-Abstandssenor
	R – Widerstand
T1 offen	auszugebener Text

5.4 Ausgänge

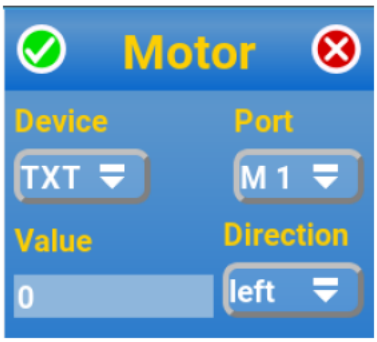
5.4.1 Output (Ausgange O1 bis O8)



Output TXT 1 512

TXT/FTD	Hardware
1	Nummer des Ausganges
512	Spannungswert (Zwischen 0 und 512)

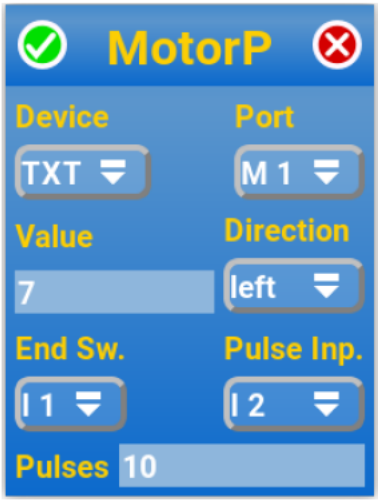
5.4.2 Motor (Ausgang M1 bis M4)



Motor 2 r 512

TXT/FTD	Hardware
2	Nummer des Motors
r	r - rechtslauf
	l - linkslauf
	s - stopp
512	Spannungswert (Zwischen 0 und 512)

5.4.3 MotorPulsewheel (Motorsteuerung über gekoppelten Impulsgeber und Endschalter)

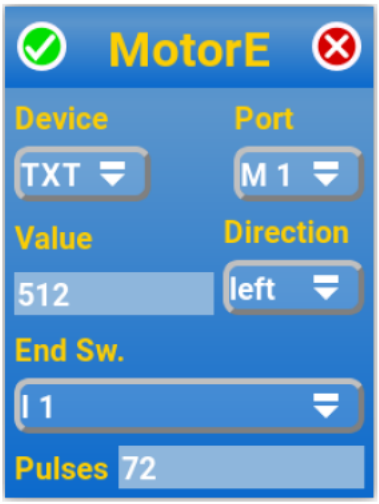


MotorP TXT 1 1 2 l 400 144

TXT/FTD	Hardware
1	Nummer des Motors
1	Nummer des Einganges für Endschalter
2	Nummer des Pulsschalters
l	r - rechtslauf
	l - linkslauf
	s – stopp
400	Spannungswert (Zwischen 0 und 512)
144	Anzahl der Impulse

Achtung! Der Endschalter wird nur bei Drehrichtung links überwacht.
Hauptsächlich für Industriemodelle ohne Encoder-Motor

5.4.4 MotorEncoder (Ansteuerung eines Encodermotors mit Endschalter)



MotorE TXT 2 2 l 400 150

TXT/FTD	Hardware
2	Nummer des Motors
2	Nummer des Endschalters
l	r - rechtslauf
	l - linkslauf
	s – stopp
400	Spannungswert (Zwischen 0 und 512)
150	Anzahl der Impulse

5.4.5 MotorEncoderSync (Synchrone Ansteuerung zweier Encoder Motoren)

MotorES TXT 3 4 l 400 150

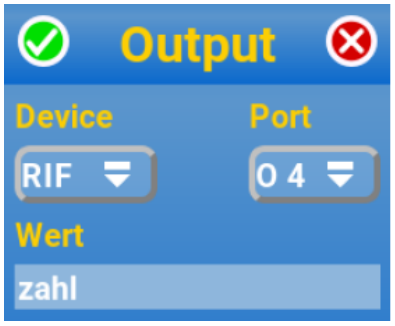
TXT/FTD	Hardware
3	Nummer des 1. Motors (Zählausgang ebenfalls 3)
4	Nummer des Motors der mit Motor 1 synchronisiert werden soll
l	r - rechtslauf
	l - linkslauf
	s – stopp
400	Spannungswert (Zwischen 0 und 512)
150	Anzahl der Impulse

Wird die Pulszahl 0 vorgegeben, laufen beide Motoren solange synchron, bis sie über einen neuen MotorEncoderSync-Befehl mit Drehrichtung „Stopp“ wieder angehalten werden.

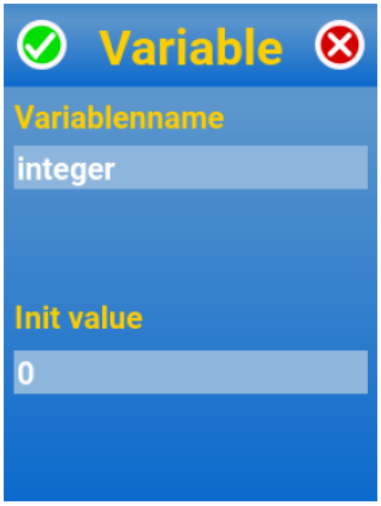
5.5 Variablen

startIDE kennt Integer (Ganzzahl)-Variablen und kann einige arithmetische Funktionen anwenden. Um eine zuvor festgelegte Variable zu benutzen muss das Feld „Wert“ länger als 0,5 Sek. Gedrückt werden. Dann erscheint die Auswahl der festgelegten Variablen. Bei kurzem Druck kann eine Zahlenkonstante eingegeben werden.
Eine Begrenzung des Wertebereiches erfolgt mit:

Calc variable variable min 512 bzw. *Calc variable variable 0*



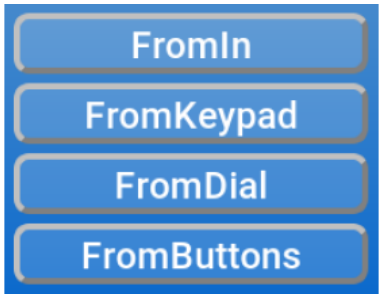
5.5.1 Init



Mit dem Init-Befehl wird der Name der Variable festgelegt und ihr ein Wert zugewiesen.

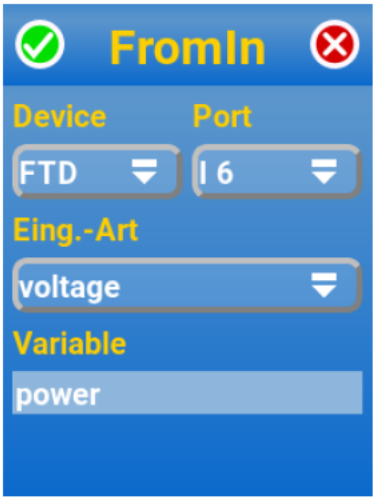
Init Zahl 5

5.5.2 From…



Hier sind mehrere Funktionen zum setzen eines Variablenwerdes zusammengefasst.

FromIn

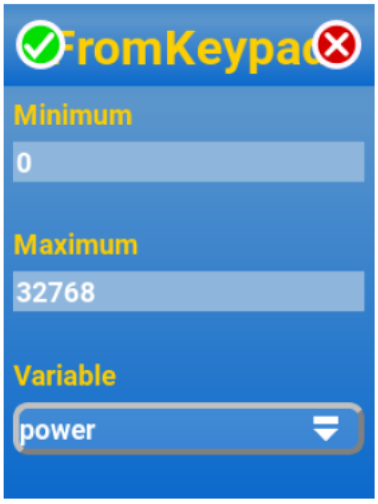


Setzt dem Wert der Variable auf den von einem angegebenen Eingang eingelesenen Wert.

FromIn TXT 6 V power

TXT/FTD	Hardware
6	Nummer Eingang
V	V – Spannung
	S – Digitaleingang
	D - Ultraschall-Abstandssenor
	R – Widerstand
power	Name der Variable

FromKeypad



Öffnet während des Programmablaufes die Bildschirmtastatur zur Eingabe eines Wertes.

FromKeypad power 0 512

power	Name der Variable
0	minimale Wert
512	maximale Wert

FromDial

Öffnet während des Programmablaufes ein Eingabefenster mit einem Drehknopf, an dem ein Wert zwischen „Minimum“ und „Maximum“ eingestellt werden kann.

FromDial power -10 10 Set level

power	Name der Variable
-10	minimaler Wert
10	maximaler Wert
Set level	Text im Eingabefenster

From Buttons

Öffnet während des Programmablaufes ein Eingabefenster mit bis zu 7 Buttons. Die Nummer des Knopfes wird in die Variable zurückgegeben.

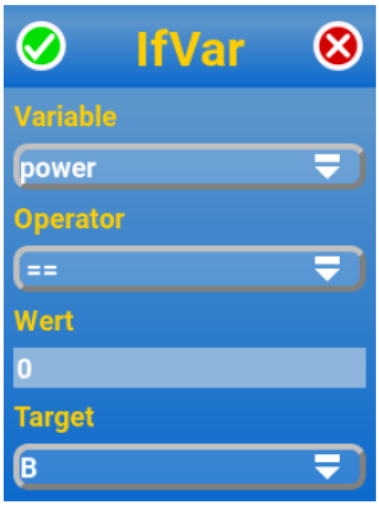
 # Hochregallager
Init Hochregal
Tag start
FromButtons Hochregal Einlagern Auslagern Inventur Ende
IfVar Hochregal == 1 einlagern
IfVar Hochregal == 2 auslagern
IfVar Hochregal == 3 inventur
Stop
 # Unterroutinen
Tag einlagern
...
Jump start
Tag auslagern
...
Jump start
Tag inventur
...
Jump start

5.5.3 QueryVar

Damit werden der Variablenname und der Wert, getrennt durch ein „:“ ausgegeben.

QueryVer integer
integer:5

5.5.4 IfVar

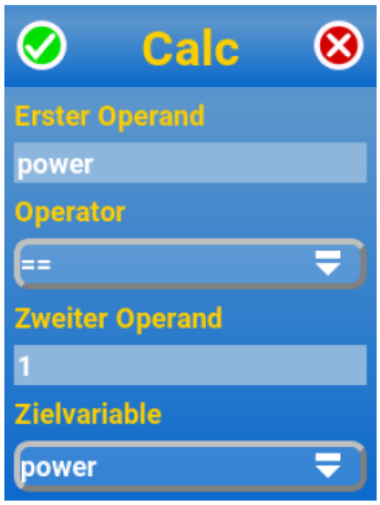


Führt einen Vergleich der angegebenen Variable mit dem gewählten Vergleichsoperator und dem vorgegebenen Wert durch. Fällt der Vergleich positiv aus, wird das Programm an der unter Target angegeben Sprungadresse (Tag) fortgesetzt.

IfVar power == 0 start

power	Name der Variable
==	Vergleichsoperator (<; <=; ==; !=; >=; >)
0	zu vergleichender Wert
B	Sprungmarke (Tag)

5.5.5 Calc



Calc ist eine komplexe Funktion. Sie führt die mathematische Verknüpfung zweier Operanden mittels eines Operators aus und übergibt das Ergebnis in die Zielvariable.
Die Operantden können dabei Konstanten oder Variablen sein.

Grundrechenarten: „+, -, *, /“
Calc var = var + 1

Modulo-Operatoren: mod (lefert den Rest einer Ganzzahldivision)
var = 9 mod 5 (lösung 4)

Exponentialoperatoren: exp
Calc var = var ^ 2

Wurzeloperatoren: root
Calc var 2 root 9 (Lösung 3) entspricht $\sqrt[2]{9}$

Min- und max-Operator: Liefern jeweils das Minimum oder Maximum der beiden Operanden zurück
Calc var 2 min 9 (Lösung 2)

Sin- und cos-Operatoren: Liefern einen skalierten ganzzahligen Sinus- oder Cosinus-Wert
Calc var 2 cos 9 (Lösung 1)

&& und || : Boolche und- bzw. oder-Verknüpfung. Rückgabe 0 für Falsch und 1 für wahr
Calc var 2 || 9 (Lösung 1)

<, <=, ==, !=, >=, >: Vergleichen Operand 1 mit Operand 2 und liefern 0(Falsch) oder 1(Wahr).
Calc var 2 < 9 (Lösung 1)

5.6 Steuerung

5.6.1 #comment

Hiermit wird ein Kommentartext in den Programmcode eingefügt,

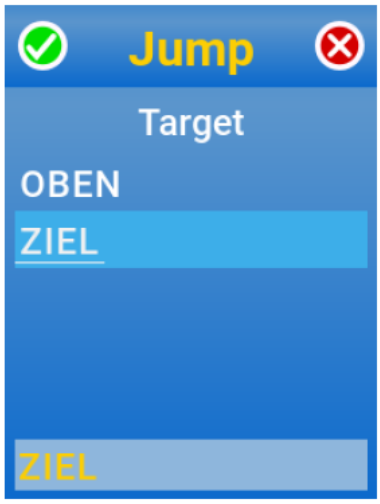
 # Hier startet das Hauptprogramm

5.6.2 Tag

Er definiert eine Sprungmarke innerhalb des Programmcodes, die mit „IfInputDigital“, „Jump“ oder „LoopTo“ angesprungen werden kann.

Tag START

5.6.3 Jump



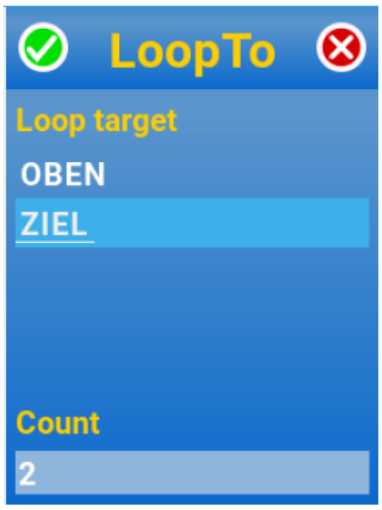
Jump ist ein Sprungbefehl. Bei Erreichen der Programmzeile „Jump <Jump Tag> “ wird die Programmausführung bei der Zeile „Tag <Jump Tag> “ fortgeführt. Damit ist die Programmierung von (Endlos-)Schleifen möglich. Es muss mindestens ein „Tag “ im Programmcode definiert sein, damit der Jump-Befehl eingefügt werden kann.

Jump START

Beispiel Endlosschleife:

Tag oben
Print Hallo
Jump oben

5.6.4 LoopTo



LoopTo ist eine Zählschleife. bei Erreichen des LoopTo Befehls wird die Programmausführung für eine bestimmte Anzahl von Durchläufen beim angegebenen Sprungziel fortgeführt. Ist die vorgegebene Anzahl von Durchlaufen erreicht, wird statt des Sprunges mit dem auf LoopTo folgenden Befehl fortgefahren.

LoopTo START 5

START	Sprungmarke, zu der gesprungen werden soll.
5	Anzahl der Wiederholungen (Anzahl min.1)

Beispiel Schleife:
Tag START
...
<weitere Befehle>
...
LoopTo START 5

Es ist zu beachten, dass LoopTo nicht prüft, ob die Sprungmarke vor oder nach dem LoopTo-Befehl liegt. Beides ist möglich und kann ggf. sinnvoll sein.

5.7 Time

Hier sind Funktionen zu finden, die eine zeitliche Steuerung des Programmablaufes ermöglichen.

5.7.1 Delay

Delay verzögert den Programmablauf für die angegebene Zeit in Millisekunden.
Wenn eine negative Zeit eingegeben wird, so wird für eine zufällige Zeitspanne zwischen 0 und der eingegebenen Zeit gewartet. Im Code wird dann der Betrag der eingegebenen Zahl mit einem nachgestellten „R “, dem random flag, angezeigt.

Delay 1500	*Pause 1,5 Sekunden*
Delay 5000 R	*Pause zwischen 0 und 5 Sekunden*

TimerQuery

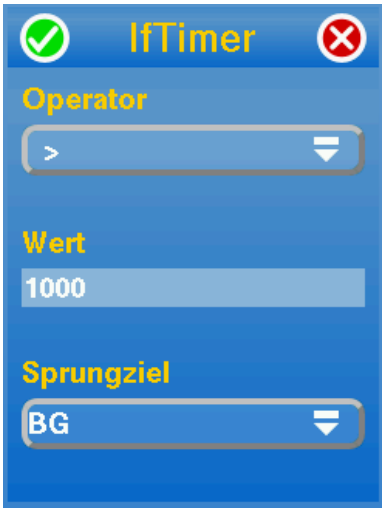
TimerQuery liefert die aktuelle Zeit seit Start des startIDE-Programmes in Millisekunden im Format „Timer: <Zeit> “ zurück.

Delay 1000	*Pause 1 sek*
TimeQuery	*aktuelle vergangene Zeit (Ausgabe im Display „Timer: 1009“)*

5.7.2 TimerClear

TimerClear setzt den Zeitzähler auf Null zurück, so dass mit TimerQuery die verstrichene Zeit seit Zurücksetzung abgefragt werden kann.

5.7.3 IfTimer



Mit IfTimer läst sich abfragen, ob seit Start des Programmes bzw. letztem TimerClear-Befehl eine bestimmte Zeit in Millisekunden (noch nicht) verstrichen ist.
Ist die Abfrage positiv, so wird die Programmausführung an der angegebenen Sprungmarke (Tag) fortgesetzt.

IfTimer < 1000 ziel

<	Vergleichsoperator (> oder <)
1000	Zeitfaktor
ziel	Sprungmarke

5.7.4 Interrupt

startIDE kann einen Interrupt, d.h. eine Unterbrechung der Programmausführung nach Ablauf einer bestimmten Zeit, ausführen. Dabei wird das angegebene Zielmodul aufgerufen und ausgeführt. Der Interrupt kann einmalig („After “) nach Ablauf der angegebenen Zeit (in Millisekunden) oder wiederholend („Every “) jeweils nach Ablauf der Zeitspanne ausgeführt werden.

5.7.5 QueryNow

QueryNow gibt das aktuelle Datum und die Uhrzeit in der Form: Now: YYYY-MM-DD_HH:MM:SS auf dem Log-Screen aus. Dies ist insbesondere für Langzeit-Logging in Verbindung mit der Log-Datei interessant, wenn man z.B. über mehrere Tage in größeren Zeitabstanden Messwerte sichern will.

5.7.6 Stop

Das Stop-Kommando beendet die Programmausführung. In seiner Wirkung entspricht es dem Erreichen des Programmendes. Alle Ausgänge werden abgeschaltet, das Ausgabelog bleibt geöffnet.
Stop

5.8 Module

Module sind in sich geschlossene Programmblocke.

Module LAMPEN_AN	*Modulbegin*
Output TXT 1 7	*Ausgang 1TXT auf Stufe 7*
Output TXT 2 7	*Ausgang 2 TXT auf Stufe 7*
MEnd	*Modulende*

Um ein Modul auszufuhren, muss es mit dem „Call <Modulname> “ Befehl aufgerufen werden. Nach Beendigung des Moduls wird die Programmausführung in der auf den Modulaufruf („Call “) folgenden Programmzeile fortgesetzt.

Stößt startIDE bei Programmausführung auf das „Module “- Schlüsselwort, endet die Programmausführung. Module werden nicht ausgeführt, wenn sie nicht explizit aufgerufen werden. Daher müssen Module immer am Ende des Programmcodes eingefügt werden.

5.8.1 Call / CallExt

Der Call-Befehl dient zum ggf. mehrmaligen Aufrufen eines im Programmcode definierten Moduls.

Call LAMPEN_AN 2

Lampen_An	Name des aufzurufenden Moduls
2	Anzahl der Aufrufe dieses Moduls

CallExt arbeitet analog, allerdings wird hierbei ein vorher exportiertes Modul, das nicht Bestandteil des Programmcodes ist, aufgerufen. Existiert ein gleichnamiges Modul im Programmcode, so hat dieses Vorrang.

5.8.2 Return

Der Return-Befehl beendet die Ausführung eines Moduls VOR Erreichendes durch MEnd definierten Modulendes.

5.8.3 Module

Hiermit wird unter Angabe des gewünschten Namens ein Modul-Anfang definiert.

Module LAMPEN_AN

5.8.4 MEnd

MEnd schliest einen Modulblock ab. Bei Erreichen von MEnd wird die Ausführung des Moduls beendet und zum aufrufenden Call-Befehl zurückgekehrt.
Ein Beispiel fur die Verwendung von Modulen:

```
# Start
Call LAMPEN_AN
Delay 1000
Call LAMPEN_AUS
Delay 1000
# Programmende
# Lampen an
Module LAMPEN_AN
Output RIF 1 7
Output RIF 2 7
MEnd
# Lampen aus
Module LAMPEN_AUS
Output RIF 1 0
Output RIF 2 0
MEnd
```

5.9 Interaction

Unter „Interaction “ finden sich einige Befehle, die eine grundlegende Interaktion mit dem Nutzer ermöglichen.

5.9.1 Print

Der Print-Befehl gibt eine Nachricht auf dem Log-Screen aus.

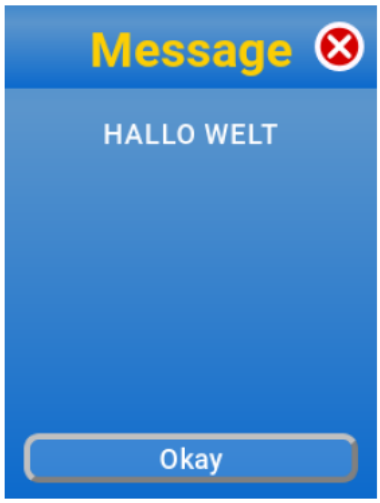
Print Hallo Welt

Alles, was auf Print folgt, wird als auszugebender Text interpretiert.

5.9.2 Clear

Der Clear-Befehl löscht den Inhalt des Log-Screens.

5.9.3 Message



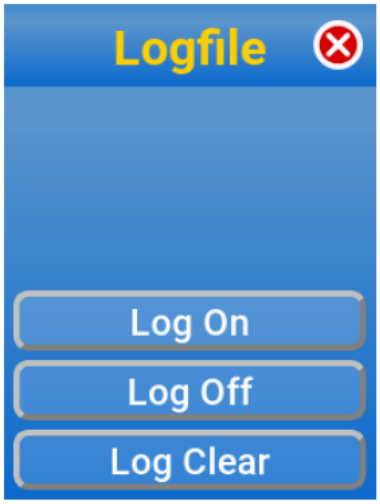
Der Message-Befehl öffnet ein Benachrichtigungsfenster, das mit einem Knopfdruck bestätigt werden muss. Damit kann man wichtige Meldungen ausgeben oder auf Freigabe durch den Benutzer warten.

Message HALLO WELT‘Okay

HALLO WELT	auszugebener Text
Okay	Text für Button

Nachricht und Knopf-Text sind durch ein Hochkomma (einfaches Anführungszeichen oben) getrennt.

5.9.4 Logfile



- Mit Log On wird das Protokollieren der Log-Screen-Ausgaben in eine Datei gestartet.
- Log Off beendet das Protokollieren in die Datei.
- Log Clear loscht ALLE auf dem TXTvorhandenen Logfiles.
- Die Logfiles werden nach dem Schema log <yyyymmtt-hhmmss>.txt z.B. log20171228-014045.txt benannt.

5.10 Beispielprogramme

Programmname	Gerät	Belegung
C_Ample	RIF	I1-Fußgängertaste; I2-Zusatztaste; O1 bis O3 Lampen für Fahrzeugampel; O4 bis O5 Fußgängerampel; O6-Signal kommt
C-Blink	RIF	O1 Lampe blinkt im 2 Hz Takt
C_Haendetrockner	LT Beginner	M2-Lampe für FT; I3-Fototransistor; M3-Motor
C-Lauflicht	RIF	Steuert nacheinander alle Lampen der Ampel an

5.11 Kopieren von Programmteilen

Mochte man einen gröseren Block von Programmzeilen kopieren, so ist es sinnvoll, diesen in ein Modul zu verpacken (Vorangestelltes „Module <name> “ und nachgestelltes „MEnd “)
Dann kann man dieses Modul exportieren und beliebig oft wieder importieren.
Die Zeilen „Module <name> “ und „MEnd “ sind danach, ebenso wie das temporär gespeicherte Modul, sinnvollerweise wieder zu loschen.

5.12 Debugging / Fehlersuche

5.12.1 Fehlermeldung

„DontKnowWhatToDo -	Wenn startIDE bei der Programmausführung auf einen unbekannten Befehl trifft.
„CompleteConfusionError“	Wenn während der Ausführung eines Befehls Komplikationen auftreten. Dies kann insbesondere der Fall sein, wenn die Parametrierung des gerade ausgeführten Befehls in irgendeiner Form fehlerhaft ist.

Die beiden vorgenannten Ausnahmefehler können auch dann auftreten, wenn der Programmcode außerhalb von startIDE modifiziert wurde und dabei syntaktische oder logische Fehler gemacht wurden.

5.12.2 Sonderfunktionen

# TRACEON	Es wird jede darauffolgende ausgeführte Codezeile auf dem Logscreen ausgegeben.
# TRACEOFF	Beendet die Funktion #TRACEON
# STEPON	Es wird der Programmcode in Einzelschritten ausgeführt. Nach jeder Codezeile muss durch antippen des Bildschirms die Ausführung bestätigt werden.
# STEPOFF	Beendet die Funktion #STEPON
# GETELAPSEDTIME	Beim Start eines startIDE-Projektes wird ein interner Zeitstempel gesetzt. Die seit Programmstart verstrichene Zeit in Sekunden wird als Flieskommazahl auf dem Log-Screen ausgegeben.
# TIMERCLEAR	Es wird während des Programmablaufs ein neuer Zeitstempel gesetzt, so dass die Zeitmessung wieder bei Null beginnt.

5.13 Dateien mit PC erstellen

Experten-Ecke

Lade ein Projekt oder Modul als Textdatei herunter.
Sende ein Projekt oder Modul aus einer Textdatei.

- Mit dem Browser die entsprechende IP öffnen. (192.168.2.122)
- Die App StartIDE öffnen
- Open Local application pages auswählen

Nun kann eine Programm-Datei als Text-Datei transferiert werden.

```
$ cd 0e500e10-33ee-11e7-9598-0800200c9a66/
$ pwd
/home/ftc/apps/0e500e10-33ee-11e7-9598-0800200c9a66
$ ls
TextToJson.py   index.py        modules         startide_de.ts
__pycache__     jsonToText.py   projects        translator.py
htmlhelper.py   logfiles        startide.py
icon.png        manifest        startide_de.qm
$
```

Eine weitere Möglichkeit ist der ist Secure Shell Fernzugriff:
puTTY öffnen und die entsprechende IP eingeben (192.168.2.122)
Das Passwort und der Login: ftc

Mit cd apps/0e500e10-33ee-11e7-9598-0800200c9a66/ wechselt man in das DIE-Verzeichnis.
Nun kann man mit *ls projects/* bzw. *ls modules/* die gespeicherten Projekte bzw. Module auflisten lassen.
Der ssh-Fernzugriff wird mit *exit* wieder beendet.

5.14 Befehlsreferenz startIDE

Eingänge	WaitForInDig Abfrage I1 bis I8	WaitInDig TXT 1 Raising 500	Raising = öffner; Falling = schließer Timout 0 = unendlich
	IfInputDig Zustandsabfrage	IfInDig TXT 2 False ende	False = Eingang offen; True = Eingang geschlossen; ende = Sprungmarke
	WaitForInput Warten auf Zustand I1 bis I8	WaitIn TXT 1 D < 50 2500	V=Spannung; D=Ultraschall/Abstand R=Widerstand; S=Digitaleingang <; ==; !=; >; Timout 0= unendlich
	IfInput Abfrage mit Sprungmarke	IfIn TXT 1 V < 5 ende	V=Spannung; D=Ultraschall/Abstand R=Widerstand; S=Digitaleingang <; ==; !=; >; ende=Sprungmarke
	QueryInput Zusandsabfrage mit Textausgabe	QueryIn TXT 1 S T1 offen	V=Spannung; D=Ultraschall/Abstand R=Widerstand; S=Digitaleingang T1 offen= auszugebener Text
Ausgänge	Output Ausgänge 01 - 08	Output TXT 1 512	512= Spannungswert zwischen 0 und 512
	Motor Ausgänge M1-M4	Motor 2 TXT r 512	r=rechtslauf; l=linkslauf; s=stopp; 512=Spannung zwischen 0 - 512
	MotorPulsewheel Motorsteuerung m. Impulsgeber u. Endschalter	MotorP TXT 1 1 2 l 400 144 *Endchalter wird nur bei Linksdrehung überwacht*	1=Nr. Motor; 1=Nr. Endschalter; 2=Nr. Impulseschalter; r=rechts (l=links;s=stopp); Spannung zw. 0 und 512; 144=Anzahl Impulse
	MotorEncoder Motorsteuerung TX mit Impuls u. Endschalter	MotorE TXT 2 2 l 400 150 *Endchalter wird nur bei Linksdrehung überwacht*	2=Nr. Motor; 2=Nr. Endschalter; l=links (r=rechts; s=stopp); Span- nung zwischen 0 und 512; 150= Anzahl der Impulse (66 1/3 ^ Drehung)
	MotorEncoderSync synchrone Ansteu- erung von zwei TXT - Motoren	MotorES TXT 3 4 l 512 400 *Bei Pulszahl 0 dauerhaft bis neuer MotorES-Befehl*	3=Nr. des 1.Motors; 4=Nr. des Motors der mit Motor 1 synchro- nisiert;r=rechts (l=links; s=stopp); Spannung = 0-512; 400 = Impulse
Variablen	Init Variablendefinition	Init Zahl 5	Zahl= Variablenname; 5 = Wert der Variable
From…	FromIn Variablenwert aus Eingangsgröße I1 - I8	FromIn TXT 6 V power	6=Nr. des Einganges; V=Spannung D=Ultraschall/Abstand;W=Widerstand S=Digitaleingang; power=Zielvariable
From…	FromKeypad Variablenwert aus Eingabe	FromKeypad power 0 512	power = Variablenname; 0 = minimaster Wert; 512 = maximalster Wert;
From…	FromDial Variablenwert aus Drehknopfeingabe	FromDial power -10 10 Set level	Power = Variablenname; -10 = mini- malster Wert; 10 = maximalster Wert; Set level = Text im Fenster
From…	From Buttons Variablenwert aus Max 7 Buttons	FromButtons wert aa bbr Ende	wert=Variable; aa = 1; bb = 2; Ende= 3 (Variable erhält Wert zw. 1 und 7 je nach gewähltem Button
	IfVar Bei einem positivem Vergleich wird an der	IfVar wert == 1 AA IfVar wert == 2 BB Sprungmarke fortgesetzt	Power=Name der Variable; == Vergleichsoperator (<;<=;!=;>=;>); 0=Vergleichswert; AA=Sprungmarke
	QueryVar Ausgabe d. Variable	QueryVer integer	Ausgabe: integer:5
	Calc Mathematische Verknüpfungen	Calc var = var + 1	Grundrechenarten = +, - *, / Mod = Rest einer Division;root = Wurzel;
Steuerung	#comment Kommentartexte	# Hier ist das Programm	Achtung: Leerzeichen zwischen Raute und Text
	Tag Sprungmarke	Tag START (Möglichst Großschreibung)	Wir mit Jump, LoopTo, IfVar oder IfInputDigital angesprungen

	Jump Sprungbefehl	Jump START	Es muss mindestens ein Tag definiert sein
	LoopTo Zählschleife	LoopTo START 5	START = Sprungmarke; 5 = Anzahl der Wiederholungen (mind. 1)
Time	Delay Verzögert den Programmablauf	Delay 1500 Delay 500 R	1500 = Zeit in Millisekunden; R = Randomflag; zufällige Pause zw. 0 und 500 (bei Eingabe negativer Wert)
Time	TimerQuery Aktuelle Laufzeit	TimeQuery	Ausgabe im Display = „Timer: 1009“
Time	TimerClear Zähler auf 0	TimeClear	Setzt die Zeit auf 0, so das die folgende Zeit mit TimeQuery angezeigt wird
Time	IfTimer Prüft vergangene Zeit mit Sprung	IfTimer < 1000 ZIEL	< = Vergleichsoperator (od. >); 1000 = Zeitfaktor ZIEL = Sprungmarke
Time	Interrupt Unterbrechung	Interrupt Every 100 stoptaster	Der Interrupt wird nur jeweils nach Ausführung einer Programmzeile überprüft
Time	QueryNow Datum und Uhrzeit	QueryNow	Ausgabe: Now: 2018-02-02_19:45:23
	Stop Beendet Programm	Stop	Wie Programmende, alle Ausgänge werden abgeschaltet
Module	Call / CallExt Mehrmaliges Aus-führen eines Modules	Call LAMPEN_AN 2 CallExt LAMPEN_AUS 2	LAMPEN_AN = Modulname 2 = Anzahl der Aufrufe CallExt = für externes Modul
	Return Vorzeitige Beendigung	Return	Beendet die Ausführung des Modules vor Ende mit MEnd
	Module Module Definition	Module LAMPEN_AN	LAMPE_AN = Name des Modules
	MEnd Abschluß Modulblock	MEnd	Die Ausführung wird beendet und zurück zum Call-Befehl gesprungen
Interaction	Print Textausgabe	Print Hallo Welt	Gibt Nachricht auf dem Log-Screen aus (Danach Delay für Anzeigezeit)
	Clear Bildschirm löschen	Clear	Löscht den Inhalt des Log-Screens
	Message Infofenster	Message HALLO ‘Okay	HALLO = Message Okay = Text für Button
	Logfile Protokoll Log-Screen	Log On Log Off	Log On= Umleitung der Ausgabe in eine Textdatei
Fehler-meldung	DontKnowWhatToDo CompleteConfusionError	unbekannten Befehl Komplikationen bei Ausführung des Befehls	
Sonder-funktionen	# TRACEON # TRACEOFF # STEPON # STEPOFF # GETELAPSEDTIME # TIMERCLEAR	Ausgabe jeder Codezeile auf den Log-Screen Beendet die Funktion # TRACEON Einzelschrittausführung (Nach jeder Zeile Bildschirm antippen) Beendet die Funktion # STEPON Ausgabe der Zeit seit Programmstart Neuer Zeitstempel.	

Sprung	**Module**	**Auswahl**	**Schalterabfrage**
# Test1 Tag TOP Jump U1 End Tag U1 Message HierU1‘OK	# Test2 Tag TOP Call Kurz Jump TOP # Module Module Kurz … MEnd	# Test 3 FromButtons wert a b IfVar wert == 1 AA IfVar wert == 2 BB Tag AA … Tag BB …	# Test4 Tag TOP WaitInDig TXT 1 Raising 0 Motor TXT 2 r 512 Delay 1000 Motor TXT 2 r 0 Jump TOP

Platz für Notizen