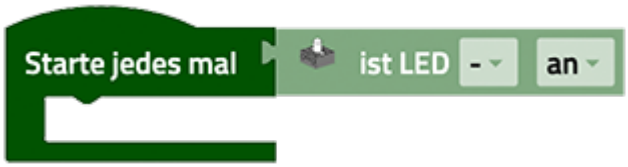
Aktoren

- Ausgang
- Motor
- Sound
- Anzeige

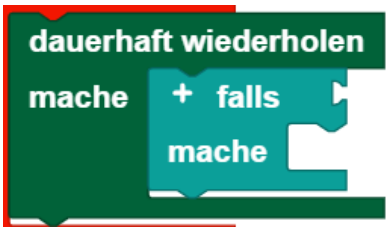
Ausgang

Der Starte jedes mal-Block

Der **Starte jedes mal-Block** bietet die Möglichkeit ein Programm ablaufen zulassen, wenn eine Bedingung erfüllt ist. Er funktioniert also ähnlich wie eine Fallunterscheidung wird aber nicht nur einmal durchlaufen, sondern jedes mal, wenn die Bedingung erfüllt ist, während des gesamten Ablauf des Programms. Der **Starte jedes mal-Block**:



Ist eine Abkürzung für folgendes Konstrukt:



Man kann in den **Starte jedes mal-Block** der Kategorie Ausgänge alle Bedingungen aus eben dieser Kategorie einsetzten.

Hinweis: Der Programmabschnitt innerhalb des Starte jedes mal-Block sollte kurz gehalten werden und keine blockierenden Aufrufe oder Endlosschleifen enthalten, so dass dieser Teil des Programms schnell abgearbeitet werden kann.

LEDs

Setzen

Mit den Blöcken **setze LED ...** und **setze LED Helligkeit ...** kann man die LED an-und ausstellen beziehungsweise ihre Helligkeit auf einen bestimmten Wert (von 0 bis 512) setzen.

Abrufen

Mit dem Block **hole LED Helligkeit** lässt sich die Helligkeit einer LED abrufen und als Wert weiterverarbeiten.

Abfragen

Mit den Blöcken **ist LED ...** und **ist LED Helligkeit ...** kann man die Aktivität beziehungsweise die Helligkeit einer LED als Bedingung nutzen. Im Beispiel wir die Helligkeit der LED auf 512 gesetzt, sofern sie nicht schon diese Helligkeit hat.

Motoren

Das Symbol auf den Motorblöcken steht stellvertretend für alle Motoren, die nicht Encoder- oder Servomotoren sind.

Setzen

Mit den Block **setze Motorgeschwindigkeit auf [] ...** kann man die Geschwindigkeit eines Motors auf einen bestimmten Wert (von 0 bis 512) setzen.

Abrufen

Mit dem Block **hole Motorgeschwindigkeit** lässt sich die Geschwindigkeit eines Motors abrufen und als Wert weiterverarbeiten.

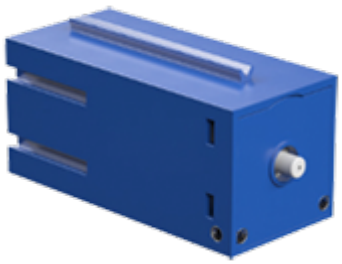
Abfragen

Mit den Blöcken **läuft Motor** und **ist Motorgeschwindigkeit ...** kann man die Aktivität beziehungsweise die Geschwindigkeit eines Motor als Bedingung nutzen.

Stoppen

Mit dem Block **stoppe Motor …** ist es möglich einen Motor zu stoppen.

Kompressor



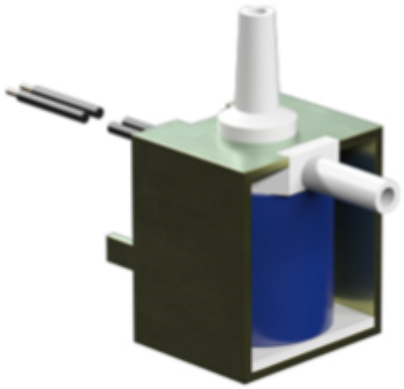
Setzen

Mit den Block **setze Kompressor []** kann man den Kompressor ein- oder ausschalten.

Abfragen

Mit den Block **ist Kompressor []** kann man die Aktivität eines Kompressors als Bedingung nutzen.

Magnetventil



Setzen

Mit den Block **setze Magnetventil []** kann man das Magnetventil ein- oder ausschalten. Hier bedeutet "ein", dass das Ventil offen ist und "aus", dass das Ventil geschlossen ist.

Abfragen

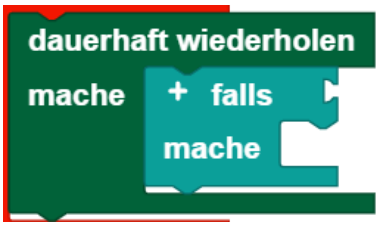
Mit den Block **ist Magnetventil []** kann man die Aktivität eines Magnetventils als Bedingung nutzen.

Motor

Der Starte jedes mal-Block

Der **Starte jedes mal-Block** bietet die Möglichkeit ein Programm ablaufen zulassen, wenn eine Bedingung erfüllt ist. Er funktioniert also ähnlich wie eine Fallunterscheidung wird aber nicht nur einmal durchlaufen, sondern jedes mal, wenn die Bedingung erfüllt ist, während des gesamten Ablauf des Programms. Der **Starte jedes mal-Block**:

Ist eine Abkürzung für folgendes Konstrukt:



Man kann in den **Starte jedes mal-Block** der Kategorie Motor alle Bedingungen aus eben dieser Kategorie einsetzten.

Hinweis: Der Programmabschnitt innerhalb des Starte jedes mal-Block sollte kurz gehalten werden und keine blockierenden Aufrufe oder Endlosschleifen enthalten, so dass dieser Teil des Programms schnell abgearbeitet werden kann.

Motor

Das Symbol auf den Motorblöcken steht stellvertretend für alle Motoren, die nicht Encoder- oder Servomotoren sind.

Setzen

Mit den Block **setze Motorgeschwindigkeit auf [] ...** kann man die Geschwindigkeit eines Motors auf einen bestimmten Wert (von 0 bis 512) setzen. Über das Dropdown-Menü (kleines Dreieck) kann die Drehrichtung gewählt werden.

Abrufen

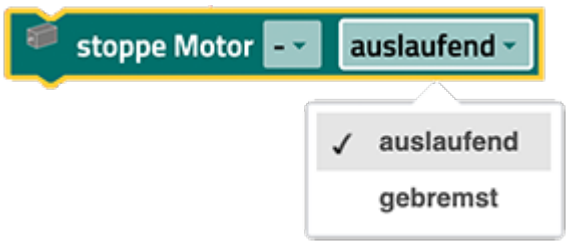
Mit dem Block **hole Motorgeschwindigkeit** lässt sich die Geschwindigkeit eines Motors abrufen und als Wert weiterverarbeiten.

Abfragen

Mit den Blöcken **läuft Motor** und **ist Motorgeschwindigkeit ...** kann man die Aktivität beziehungsweise die Geschwindigkeit eines Motor als Bedingung nutzen.

Stoppen

Mit dem Block **stoppe Motor []** ist es möglich einen Motor zu stoppen. Dabei bietet der Block **stoppe Motor []** die Optionen, einen Motor direkt oder auslaufend zu stoppen. Die gewünschte Option kann über das Dropdown-Menü (kleines Dreieck) ausgewählt werden:



Servomotor

Setzen

Mit den Block **setze Position auf ...** kann man die Position eines Servomotors auf einen bestimmten Wert (von 0-512) setzen. 0 und 512 sind die Werte für die maximale Auslenkung rechts und links. Bei dem Wert 256 steht der Servomotor dementsprechend in der Mitte.

Abrufen

Mit dem Block **rufe Position ab** lässt sich die Position eines Servomotors abrufen und als Wert weiterverarbeiten.

Encodermotor

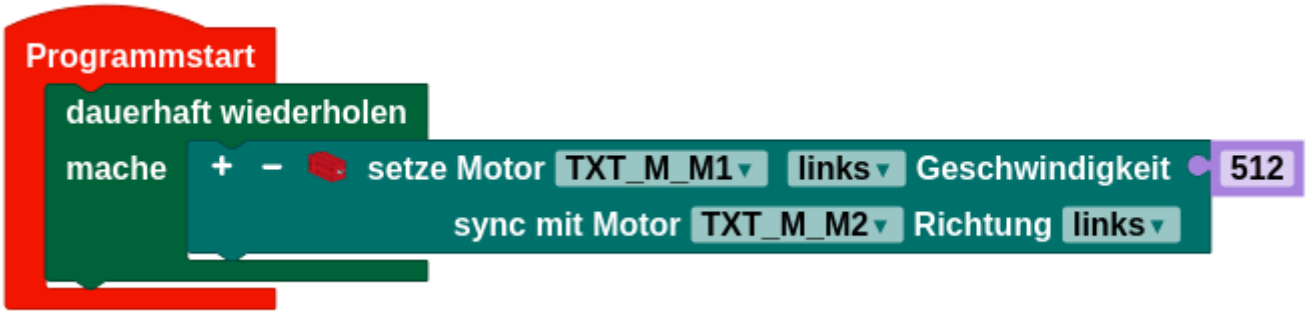
Der Encodermotor hat die gleichen Funktionen wie ein normaler Motor, bietet aber zusätzlich die Möglichkeit, die Umdrehungen zu zählen und mehrere Motoren synchron anzusteuern. Eine Umdrehung wird dabei in ~64 Schritte unterteilt.

Setzen

Mit dem Block

kann man die Geschwindigkeit eines Motors auf einen bestimmten Wert (von 0-512) setzen. Über das Dropdown-Menü (kleines Dreieck) kann die Drehrichtung gewählt werden. Zusätzlich kann man die Anzahl an Schritten eingeben, die der Motor zurücklegen soll. In diesem Beispiel dreht sich der Motor 100 Schritte, also eine und eine drittel Umdrehungen. Wie am Bespiel zu sehen hat dieser Block ein Pluszeichen, mit Hilfe dessen sich mehrere Motoren synchron ansteuern lassen. Es ist möglich Motoren am Master oder an einer Extension zu synchronisieren, eine übergreifenden Synchronisierung bspw. zwischen Motoren des Master und einer Extension ist nicht möglich.

Hinweis: Schnell aufeinanderfolgende Synchronisierungsaufrufe, wie sie z.B. durch eine Schleife möglich sind (siehe Beispiel), können die Synchronität beeinträchtigen oder sogar komplett verhindern.



Stoppen

Mit dem Block **stoppe Motor …** stoppt man einen Motor. Möchte man mehrere Motoren gleichzeitig stoppen, kann man über das Plus links am Block bis zu drei weitere Motoren hinzufügen.

Abfragen

Der Block **hat Position erreicht** wird genutzt, um das Erreichen der Position als Bedingung zu nutzen. Mit Position ist hier die Endposition eines Encodermotors nach vollendeter Schrittweite gemeint.

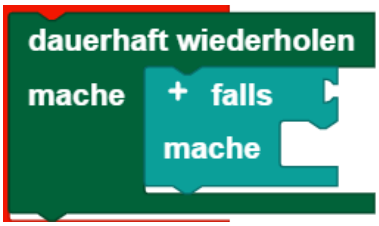
Sound

Der TXT 4.0 Controller hat einen eingebauten Lautsprecher und bietet somit die Möglichkeit Sounds abzuspielen.

Der Starte jedes mal-Block

Der **Starte jedes mal-Block** bietet die Möglichkeit ein Programm ablaufen zulassen, wenn eine Bedingung erfüllt ist. Er funktioniert also ähnlich wie eine Fallunterscheidung wird aber nicht nur einmal durchlaufen, sondern jedes mal, wenn die Bedingung erfüllt ist, während des gesamten Ablauf des Programms. Der **Starte jedes mal-Block**:

Ist eine Abkürzung für folgendes Konstrukt:



Man kann in den **Starte jedes mal-Block** der Kategorie Sound alle Bedingungen aus eben dieser Kategorie einsetzten.

Hinweis: Der Programmabschnitt innerhalb des Starte jedes mal-Block sollte kurz gehalten werden und keine blockierenden Aufrufe oder Endlosschleifen enthalten, so dass dieser Teil des Programms schnell abgearbeitet werden kann.

Abspielen

Vorinstallierte Audiodateien

Mit dem folgenden Block kann man einen von 29 vorinstallierten Sounds abspielen. Die gewünschte Audiodatei kann über das Dropdown Menü (kleines Dreieck) ausgewählt werden. Außerdem ist es möglich, den Ton in Dauerschleife abzuspielen. Dafür muss man das Kästchen hinter dem Dauerschleife-Symbol ankreuzen.

Eigene Audiodateien

Möchte man einen eigenen Sound abspielen, kann man den Block

nutzen. Um seinen eigenen Sound in den Block einzubetten muss man:

1. Mit dem Controller Verbunden sein
2. Die IP-Adresse des Controllers in den Browser eingeben (hierbei muss die IP gewählt werden, die auch zum Verbinden mit dem Controller genutzt wurde)
3. Auf der aufgerufenen Seite USER: ft, PASSWORD: fischertechnik eingeben
4. Ordner sounds öffnen und dort über das Plus die gewünschte Audiodatei auf den Controller laden (wichtig: die Audiodatei muss im wav-Format vorliegen)
5. Im ROBO Pro Coding-Block unter Pfad "./dateiname.wav" angeben

Auch hier gibt es die Option, den Sound in Dauerschleife abzuspielen.

Abfragen

Um abzufragen, ob eine Audiodatei abgespielt wird, nutzt man den Block **gibt Ton wieder**. Dieser kann als Bedingung im Programm genutzt werden.

Stoppen

Um einen Ton zu stoppen, wird einfach der Block **stoppe Tonwiedergabe** im Programm verwendet.

Anzeige

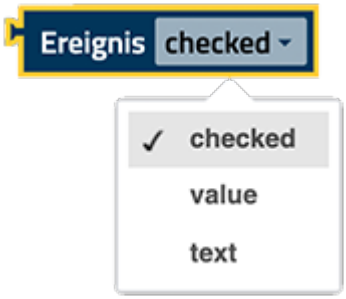
Mit den Blöcken der Kategorie Display lässt sich der Bildschirm des TXT 4.0 Controllers gestalten und nutzbar machen. Dies geschieht in zwei Schritten:

1. Konfigurieren, das heißt
   - Eine neue Datei der Kategorie Display öffnen, über das Seiten Symbol mit dem Plus oben links
   - die gewünschten Elemente auf den gerasterten Bereich ziehen (er stellt den konfigurierbaren Teil des Displays dar)
   - bei Bedarf Spezifikationen anpassen.
2. Programmieren, das heißt
   - Im Hauptprogramm mit den Blöcken der Kategorie Display die Wirkung von Interaktion mit dem Display programmieren.

Blöcke

Ereignisabfrage

Der Block **Ereignis []** ruft den Rückgabewert eines Elements ab. Dieser Block kann nur in den Ereignisprogrammen genutzt werden. In diesen Ereignisprogrammen bezieht sich der Block automatisch auf das Ereignis in dessen Programm er verwendet wird. Der geeignete Typ für den Rückgabewert, kann über das Dropdown-Menü (kleines Dreieck) gewählt werden:



Beschriftungsfeld

Mit dem Element Beschriftungsfeld kann man einen Text auf dem Bildschirm platzieren. Das Symbol im Anzeigekonfigurator ist das Etikett. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe des Beschriftungsfelds in Pixeln,
- die Position des Beschriftungsfelds in Pixeln (auf dem angegebenen Punkt liegt die obere linke Ecke des Textfeldes),
- der Name des Beschriftungsfelds und
- der Inhalt des Beschriftungsfelds (dieser Text wird bei Start des Displays abgebildet)

festgelegt werden.

Mit dem Block **setze Beschriftungsfeld Text ...** lässt sich der abgebildete Text im Laufe des Programms ändern.

Eingaben

Das Element **Eingabe** erlaubt es, dass Nutzer*innen über den Controller Text eingeben. Das zugehörige Symbol im Anzeigekonfigurator ist das "T" Zeichen. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

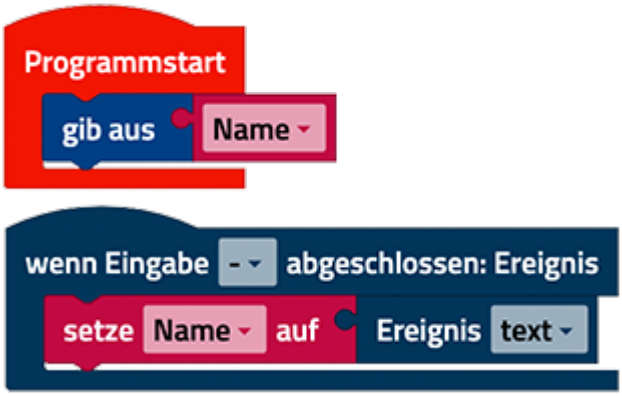
- die Größe des Eingabefeldes in Pixeln,
- die Position des Eingabefeldes in Pixeln (auf dem angegebenen Punkt liegt die obere linke Ecke des Eingabefeldes),
- der Name des Eingabefeldes und
- der Inhalt des Eingabefeldes (dieser Text wird bei Start des Displays abgebildet)

festgelegt werden.

Mit dem Block **setze Eingabefeld Text ...** lässt sich der abgebildete Text im Laufe des Programms ändern.

Eingabe-Programm

Das Eingabe-Programm läuft ab, wenn eine Eingabe abgeschlossen wurde. Es wird separat vom Hauptprogramm geschrieben. Variablen funktionieren global über beide Programme hinweg. Das Eingabe-Programm läuft im Block **wenn Eingabe abgeschlossen** ab. Der **Ereignis []**-Block wird im Eingabe-Programm auf "text" gesetzt. In diesem Beispiel wird die Variable **Name** auf den eingegebenen Text gesetzt, sie wird dann im Hauptprogramm genutzt, um den eingegebenen Text auszugeben:



Messinstrument

Die Messinstrument-Funktion kann Werte (keine Werte kleiner 1) darstellen. Das zugehörige Symbol im Displaykonfigurator ist die Skalierung. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe des Messinstruments in Pixeln,
- die Position des Messinstruments in Pixeln (auf dem angegebenen Punkt lieg die obere linke Ecke der Messlehre),
- der Name des Messinstruments,
- die Ausrichtung des Messinstruments
- der Wertebereich, den das Messinstrument darstellt, und
- der Wert des Messinstruments, der bei Start des Displays gezeigt wird

festgelegt werden.

Mit dem Block **setze Messinstrument auf Wert ...** lässt sich das Messinstrument auf den eingegebenen Wert setzen. Dieser Wert sollte im vorher definierten Wertebereich liegen. Liegt der Wert außerhalb des Wertebereich, wird, je nachdem ob der Wert zu groß oder zu klein ist, eine der Grenzen des Wertebereichs dargestellt.

Statusanzeige

Der Statusindikator zeigt die Aktivität von etwas an. Je nach Status leuchtet er ("aktiv") oder tut dies nicht ("inaktiv"). Das Symbol im Displaykonfigurator ist eine leuchtende Diode. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe der Statusanzeige in Pixeln,
- die Position der Statusanzeige in Pixeln (auf dem angegebenen Punkt liegt die obere linke Ecke des Statusindikators),
- der Name der Statusanzeige,
- die Farbe der Statusanzeige und
- ob die Statusanzeige zu Beginn aktiv oder inaktiv sein soll,

festgelegt werden.

Mit dem Block **setze Statusanzeige aktiv []** lässt sich die Statusanzeige aktivieren bzw. deaktivieren. Im Dropdown-Menü (kleines Dreieck) lässt sich wählen, ob die Statusanzeige auf aktiv oder inaktiv gesetzt werden soll.

Schieberegler

Der Schieberegler gibt Werte abhängig von seiner Position zurück. Die Position kann dabei vom Nutzer*innen über den Touchscreen verändert werden. Der Wert kann über den **Ereignis []**-Block abgerufen werden, sobald der Schieberegler ruht. Der abgerufene Wert ist eine Dezimalzahl. Will man den Wert des Schiebereglers ganzzahlig haben, muss man den **runde**-Block einsetzen. Das zugehörige Symbol für den Schieberegler ist der Strich mit dem Kreis. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe des Schiebereglers in Pixeln,
- die Position des Schiebereglers in Pixeln (auf dem angegebenen Punkt liegt dann die obere linke Ecke des Schiebereglers)
- der Name des Schiebereglers,
- die Aktivität des Schiebereglers,
- die Ausrichtung des Schiebereglers,
- den Wertebereich der über den Schieberegler abgedeckt wird und
- der Wert, auf dem der Regler bei Start des Displays steht

festgelegt werden.

Mit dem Block **Setze Schieberegler Wert ...** kann man den Schieberegler auf einen anderen Wert verschieben.

Mit **setze Schieberegler aktiviert []** kann man die Aktivität über das Dropdown-Menü (kleines Dreieck) wechseln.

Schieberegler-Programm

Das Schieberegler-Programm läuft ab, nachdem der Schieberegler verschoben wurde. Es wird separat vom Hauptprogramm geschrieben. Variablen funktionieren global über beide Programme hinweg. Das Schieberegler-Programm läuft im Block **wenn Schieberegler bewegt** ab. Der **Ereignis []**-Block wird im Schieberegler-Programm auf "value" gesetzt. In diesem Beispiel wird die Geschwindigkeit des Motors über den Schieberegler gesteuert. Der Wert des Schiebereglers muss gerundet werden, da der Motor nur ganze Zahlen als Drehzahl akzeptiert:

Schaltfläche

Die Schaltfläche ist ein beschriftetes Feld, das gedrückt werden kann. Drückt man die Schaltfläche, läuft das Schaltflächen-Programm ab, sobald sie wieder losgelassen wird. Das zugehörige Symbol für die Schaltfläche ist das Quadrat mit der "OK" Beschriftung. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe der Schaltfläche in Pixeln,
- die Position der Schaltfläche in Pixeln (auf dem angegebenen Punkt liegt die obere linke Ecke des Knopfes),
- der Name der Schaltfläche,
- den Text, der auf der Schaltfläche steht und
- die Aktivität der Schaltfläche

festgelegt werden.

Mit dem Block **setze Schaltfläche aktiviert []** kann man die Aktivität über das Dropdown-Menü (kleines Dreieck) wechseln.

Schaltflächen-Programm

Das Schaltflächen-Programm läuft ab, sobald die Schaltfläche nicht mehr gedrückt ist. Es wird separat vom Hauptprogramm geschrieben. Variablen funktionieren global über beide Programme hinweg. Das Schaltflächen-Programm läuft im Block **wenn Schaltfläche angeklickt** ab. Der **Ereignis []**-Block kann im Schaltflächen-Programm nicht verwendet werden, da die Schaltfläche keinen Rückgabewert hat. In diesem Beispiel wird die LED aktiviert, wenn die Schaltfläche gedrückt wurde.

Schalter

Der Schalter kann zwei Positionen einnehmen und befindet sich immer in genau einer dieser beiden Positionen. Je nach Position gibt er **wahr** oder **falsch** zurück. Das zugehörige Symbol für den Schalter ist das Oval mit dem Punkt. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe des Schalters in Pixeln,
- die Position des Schalters in Pixeln (auf dem angegebenen Punkt liegt die obere linke Ecke des Schalters),
- der Name des Schalters,

- den Text, der neben dem Schalter steht,
- die Aktivität des Schalters und
- den Zustand in dem sich der Schalter bei Start des Programms befinden soll

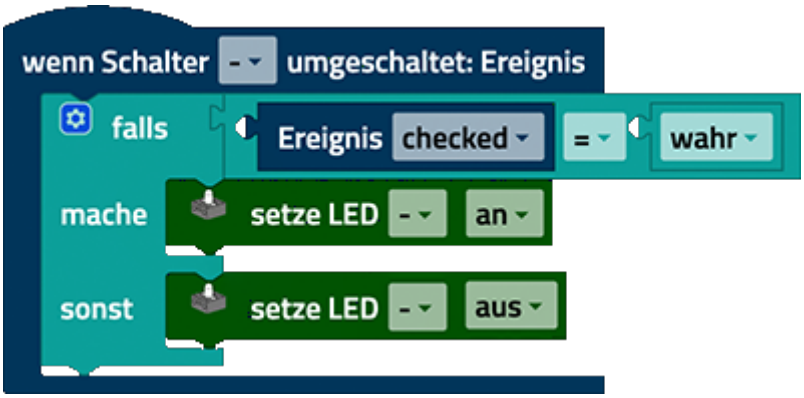
angepasst werden.

Der Block

übernimmt zwei Funktionen. Man kann entweder die Aktivität (enabled im Dropdown-Menü wählen) oder den Zustand (checked im Dropdown-Menü wählen) auf **wahr** oder **falsch** setzen.

Schalter-Programm

Das Schalter-Programm läuft jedes mal ab, wenn der Schalter umgelegt wird. Es wird separat vom Hauptprogramm geschrieben. Variablen funktionieren global über beide Programme hinweg. Das Schalter-Programm läuft im Block **wenn Schalter umgeschaltet** ab. Der **Ereignis []**-Block wird im Schalter-Programm auf "checked" gesetzt, er gibt **wahr** oder **falsch** zurück. Dieses Beispielprogramm schaltet die LED ein, wenn der Schalter umgelegt ist, andernfalls wird die LED ausgeschaltet:



Kontrollkästchen

Das Kontrollkästchen kann zwei Zustände annehmen und befindet sich immer in genau einem dieser beiden. Je nach Zustand gibt es **wahr** oder **falsch** zurück. Das Symbol für das Kontrollkästchen ist das Quadrat mit dem Haken. Zieht man dieses Symbol in den gerasterten Bereich, öffnet sich rechts ein Fenster. Hier kann unter Inspektor

- die Größe des Kontrollkästchens in Pixeln,
- die Position des Kontrollkästchens in Pixeln (auf dem angegebenen Punkt liegt die obere linke Ecke des Kontrollkästchens),
- der Name des Kontrollkästchens,
- den Text, der neben dem Kontrollkästchen steht,
- die Aktivität des Kontrollkästchens und
- den Zustand in dem sich das Kontrollkästchen bei Start des Programms befinden soll

festgelegt werden.

Der folgende Block übernimmt zwei Funktionen. Über das Dropdown-Menü (kleines Dreieck) kann gewählt werden, welche man nutzt. Man kann entweder die Aktivität (enabled im Dropdown-Menü wählen) oder den Zustand (checked im Dropdown-Menü wählen) auf **wahr** oder **falsch** setzen.

Kontrollkästchen-Programm

Das Kontrollkästchen-Programm läuft jedes mal ab, wenn das Kontrollkästchen gedrückt wird. Es wird separat vom Hauptprogramm geschrieben. Variablen funktionieren über beide Programme hinweg. Das Kontrollkästchen-Programm läuft im Block **wenn Kontrollkästchen umgeschaltet** ab. Der **Ereignis []**-Block wird im Schalter-Programm auf "checked" gesetzt, er gibt **wahr** oder **falsch** zurück. Dieses Beispielprogramm schaltet die LED ein, wenn das Kontrollkästchen angehakt ist, andernfalls wird die LED ausgeschaltet.