Lösungsblatt

# Aufgabe 8: Spurfolger

Die Programmieraufgaben sind besonders gut für das Verständnis der Bedeutung und Konzeption Endlicher Automaten geeignet. Es sollte darauf geachtet werden, dass das Zustandsübergangsdiagramm vor Beginn der Programmierung optimiert und auf möglichst wenige Zustände reduziert wird.

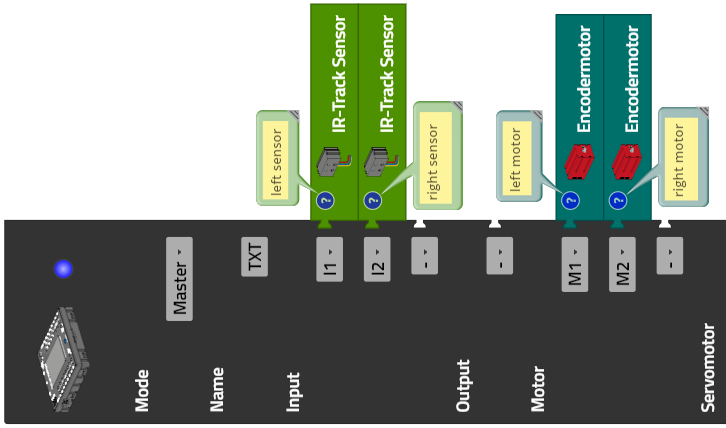
Die Experimentieraufgaben behandeln einen P- und einen PD-Regler.

## Konstruktionsaufgabe

Siehe Bauanleitung.

## Programmieraufgaben

Konfiguration der Sensoren und Aktoren:



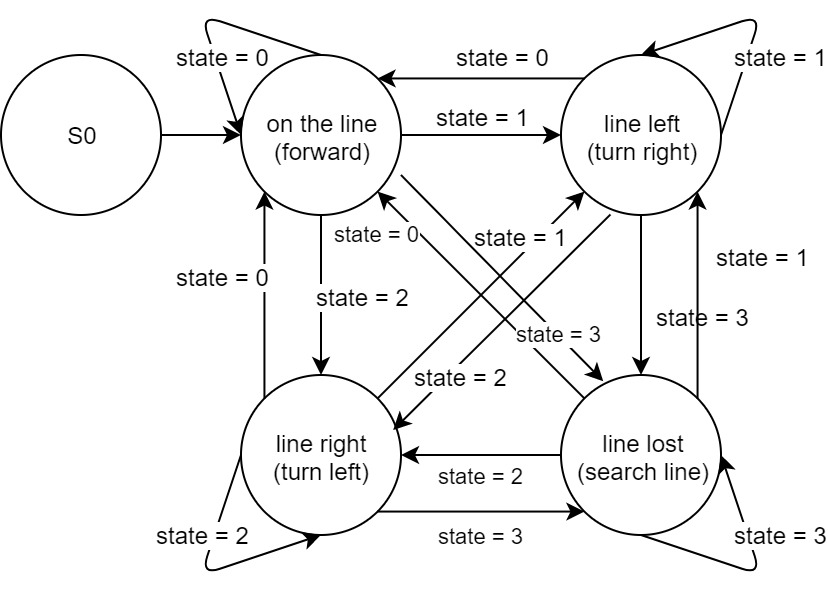
**1. Spurfolger mit Spursensor**

Bei der im Folgenden dargestellten Lösung wird der Buggy so gesteuert, dass die beiden IR-Sensoren mittig über der schwarzen Linie liegen. Der gewünschte Zustand für die Geradeausfahrt ist, dass beide Sensoren „0“ (schwarze Linie) anzeigen. Liefert der linke Sensor „1“, muss nach rechts, liefert der rechte Sensor „1“, muss nach links gesteuert werden. Wenn beide Spursensoren „1“ erkennen, hat er die Spur verloren und muss sie suchen. Mit den Werten der Spursensoren lassen sich also vier verschiedene Zustände unter­scheiden:

* auf der Spur (0)
* Spur links (1),
* Spur rechts (2) und
* Spur verloren (3).

Indem die Werte der beiden IR-Sensoren als Binärwerte interpretiert werden, kann aus den Sensorwerten die Nummer des Zustands („00“, „01“, „10“ und „11“) abgeleitet werden.

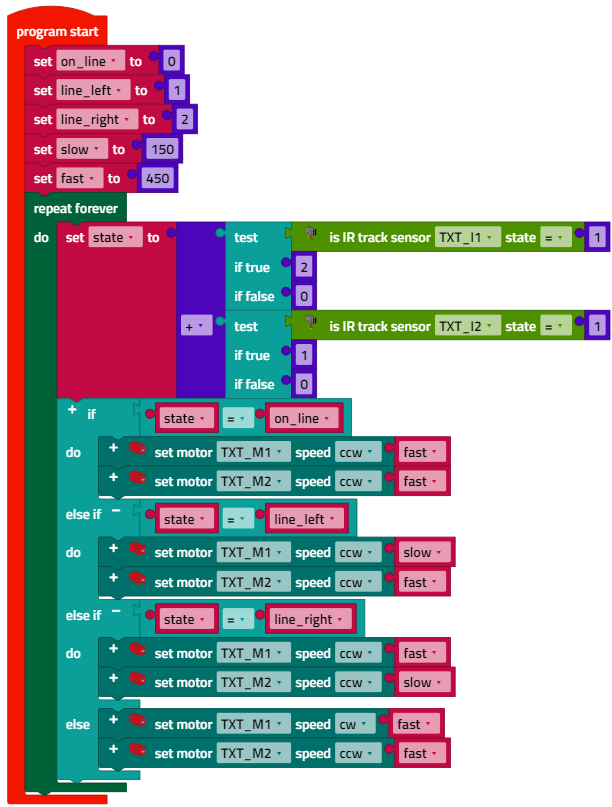
1a. Zustandsübergangsdiagramm:



*State-Transition\_Diagram\_Line\_Follower.drawio*

Das Suchen der Spur kann durch Drehen auf der Stelle oder durch „Zurücksetzen“ erfolgen. Beim Drehen kann es passieren, dass der Buggy die Fahrtrichtung wechselt.

1b. Programm (Beispiel):



*Buggy\_Line\_Follower\_digital.ft*

Eine verlorene Spur wird in dieser Lösung durch Drehen nach links gesucht.

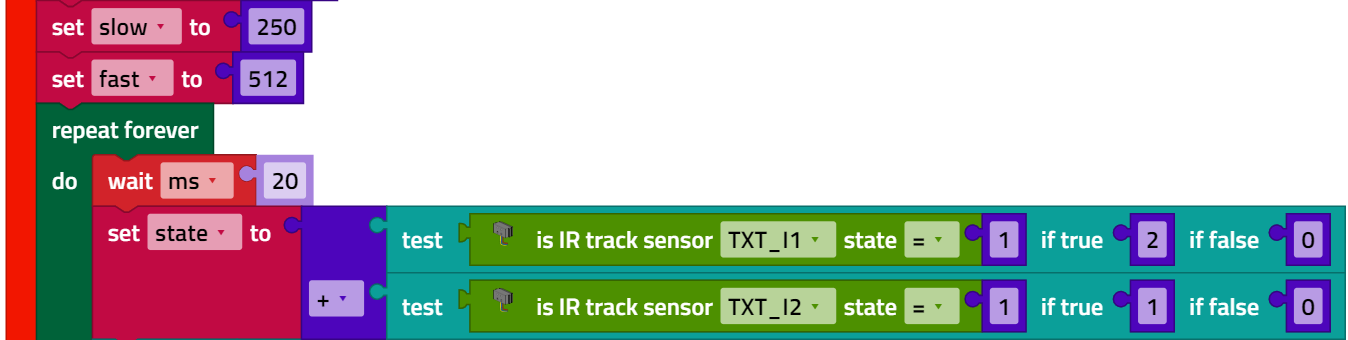
**2. Optimierter Spurfolger**

Die Geschwindigkeit des Spurfolgers lässt sich steigern, indem man

* die Fahrgeschwindigkeit erhöht
* den Geschwindigkeitsunterschied der beiden Motoren bei einer Richtungs­korrektur (im Falle einer Spurabweichung) kleiner wählt
* durch kurze Pausen in jedem Schleifendurchlauf die Anzahl der Richtungs­korrekturen insgesamt verringert.

Die Grenze ist erreicht, wenn der Buggy in einer Kurve die Spur verliert.

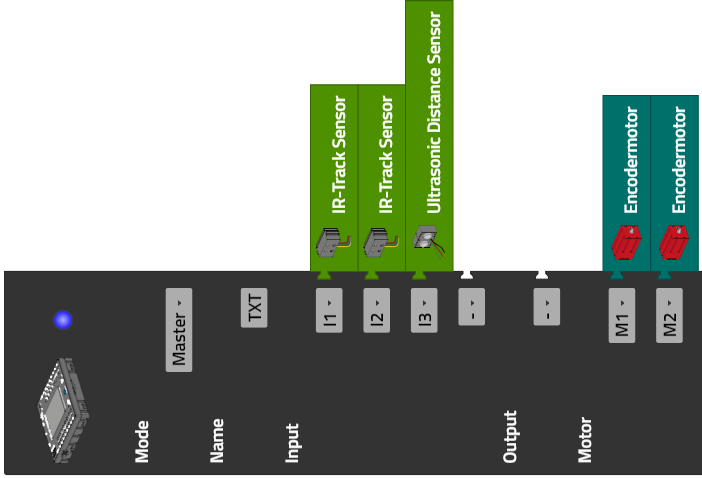
Programmauszug (Beispiel):



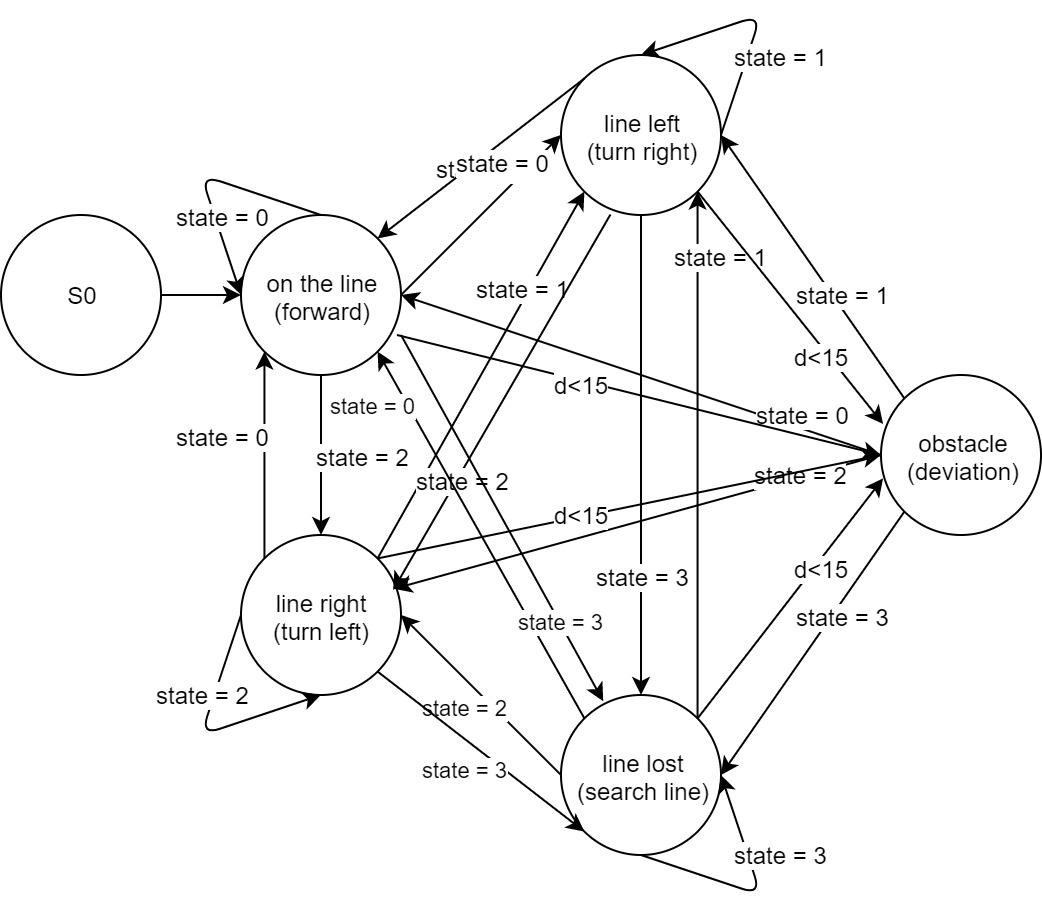
*Buggy\_Line\_Follower\_digital\_optimized.ft*

**3. Spurfolger mit Hinderniserkennung**

Konfiguration des Ultraschall-Sensors:



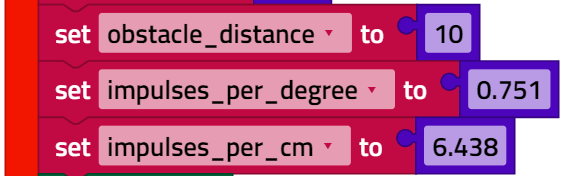
3a. Zustandsübergangsdiagramm:



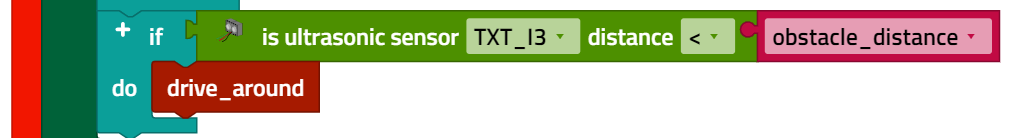
*State-Transition\_Diagram\_Line\_Follower\_with\_Obstacle\_Detection.drawio*

3b. Programmauszug (Beispiel):

Zunächst wird das Programm um drei Konstanten ergänzt: den Hindernisabstand (10 cm) und die beiden in den Aufgaben 6 und 7 bestimmten Umrechnungsfaktoren:



Falls ein Hindernis erkannt wird, soll der Buggy es umfahren:



*Buggy\_Line\_Follower\_with\_Obstacle\_Detection\_digital.ft*

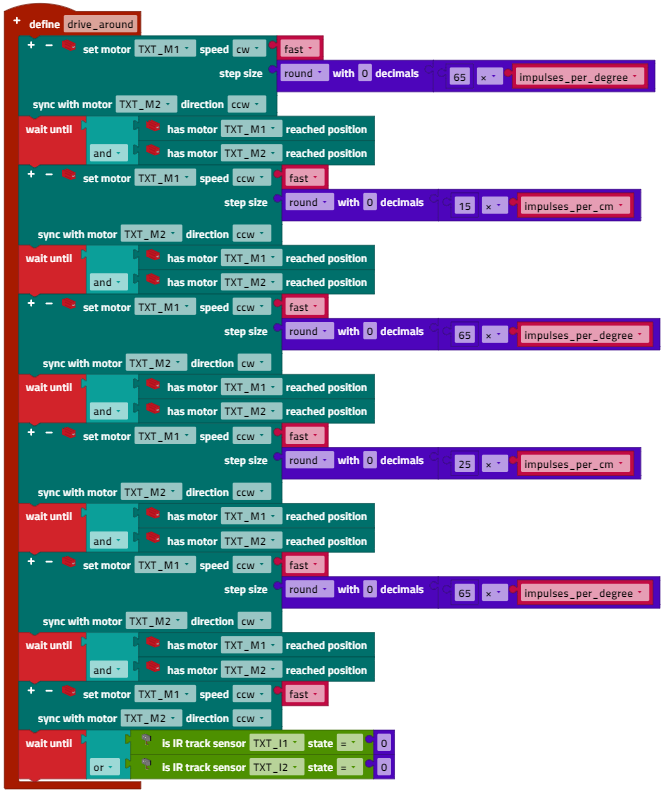
Die Umfahrung sollte in ein Unterprogramm „ausgelagert“ werden, um das Programm übersichtlich zu halten.

Eine Lösungsstrategie für die Umfahrung eines Hindernisses ist die folgende:

* den Buggy um 45-90° nach links drehen
* ein Stück (abhängig von der Größe des Hindernisses), z. B. 15 cm, geradeaus fahren
* um 45-90° zurück nach rechts drehen
* am Hindernis vorbei fahren (z. B. 30 cm)
* erneut um 45-90° nach rechts drehen
* fahren, bis der Buggy wieder auf die Spur trifft.

Die erste Drehung sollte in dieselbe Richtung erfolgen wie beim Suchen der Spur (hier also: nach links), damit der Buggy nicht die Fahrt­richtung wechselt, wenn er hinter dem Hindernis die Spur suchen muss. Hilfreich ist außerdem, den Buggy in einem spitzen Winkel (im folgenden Beispielprogramm: 65°) auf die Spur treffen zu lassen.

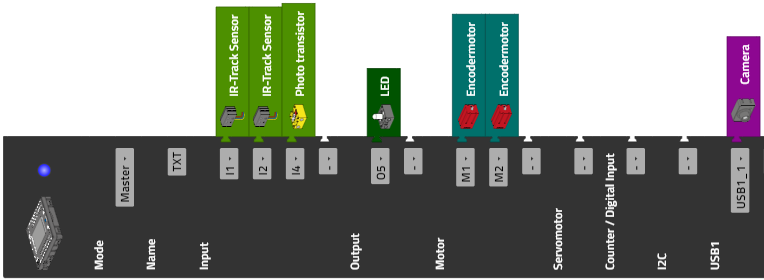
Programmauszug (Beispiel) „Hindernis umfahren“:



*Buggy\_Line\_Follower\_with\_Obstacle\_Detection\_digital.ft*

**4. Steuerung mit Farbflächen**

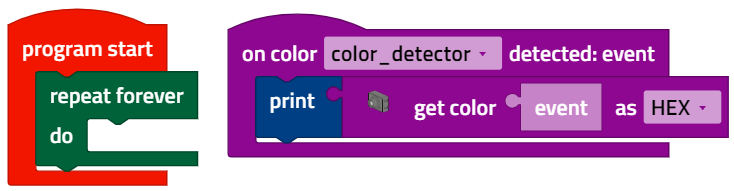
Konfiguration der Kamera und der Frontstrahler:



4a. Im folgenden Beispielprogramm reagiert der Buggy auf die folgenden Farbfelder:

* Rot: Hupen
* Grün: um 180° auf der Stelle drehen
* Blau: einmal kurz (0,1 s) blinken

4b. Der von der Kamerasoftware erkannte Farbton hängt stark von den Lichtver­hältnissen ab. Daher sollte die Farbe als RGB-Wert (HEX) gewählt werden. Diesen Wert kann man sich mit einer simplen Variante des Programms „*RGB\_Colorcode.ft*“ aus Aufgabe 4 (Experimentieraufgabe 1a) auf der Konsole ausgeben lassen:

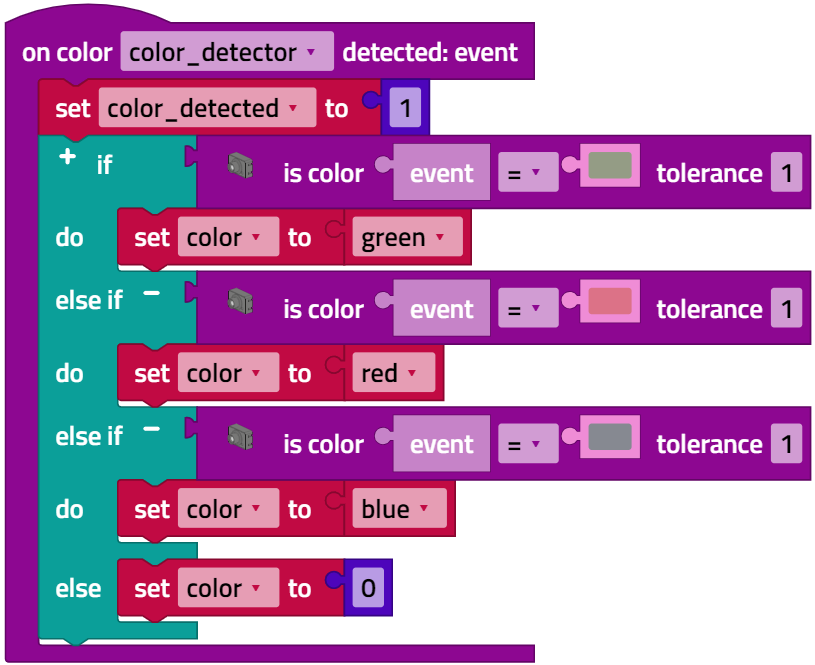


*HEX\_Colorcode.ft*

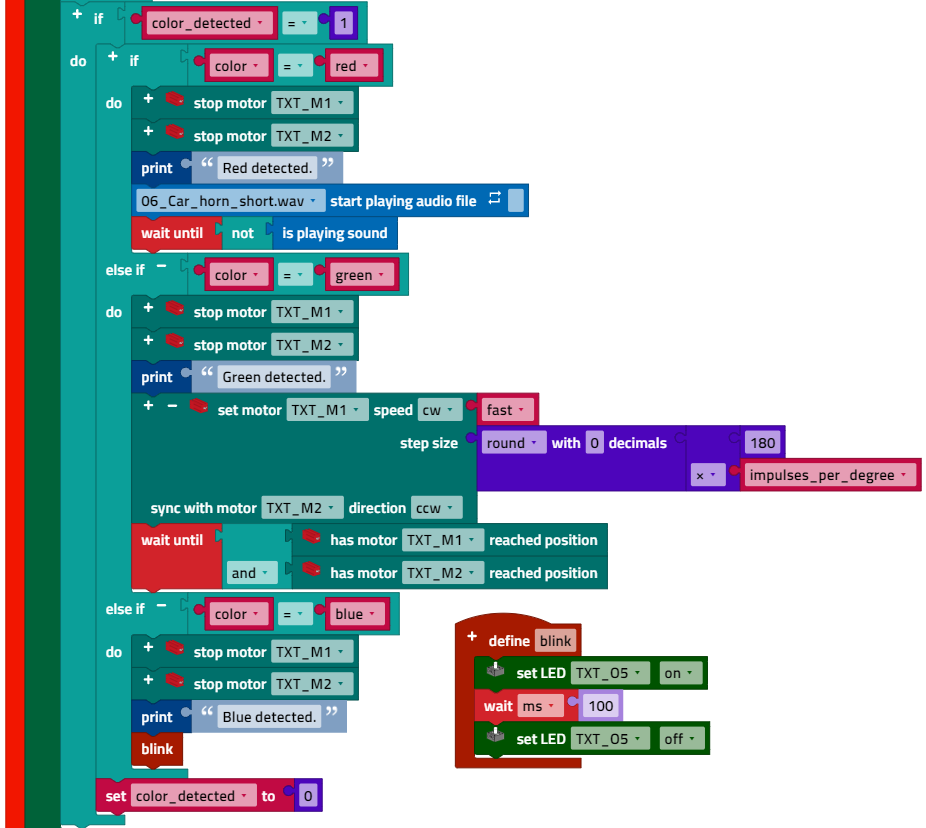
4c. In die Farberkennung werden die ermittelten Hex-Werte als Farbwerte eingetragen. Über die Toleranz kann die Empfindlichkeit der Erkennung eingestellt werden.

Die Farberkennung wird über eine Variable („color\_detected“) als Semaphor signali­siert.

Programmausschnitte (Beispiel):

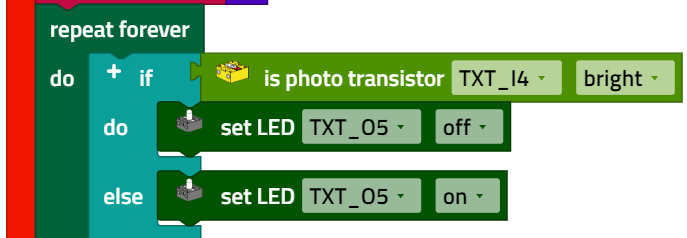


*Buggy\_Line\_Follower\_with\_Color\_Recognition\_digital.ft*



*Buggy\_Line\_Follower\_with\_Color\_Recognition\_digital.ft*

4d. Programmauszug (Beispiel) für die automatische (De-) Aktivierung der Front­strahler:

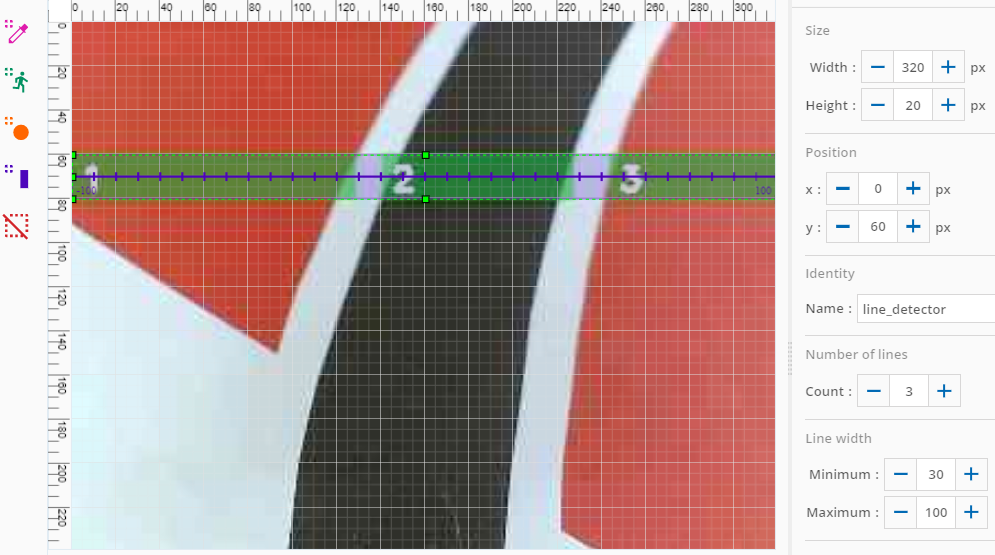


*Buggy\_Line\_Follower\_with\_Color\_Recognition\_digital.ft*

## Experimentieraufgaben

**1. Spurerkennung mit Kamera**

1a. Konfiguration der Linienerkennung:

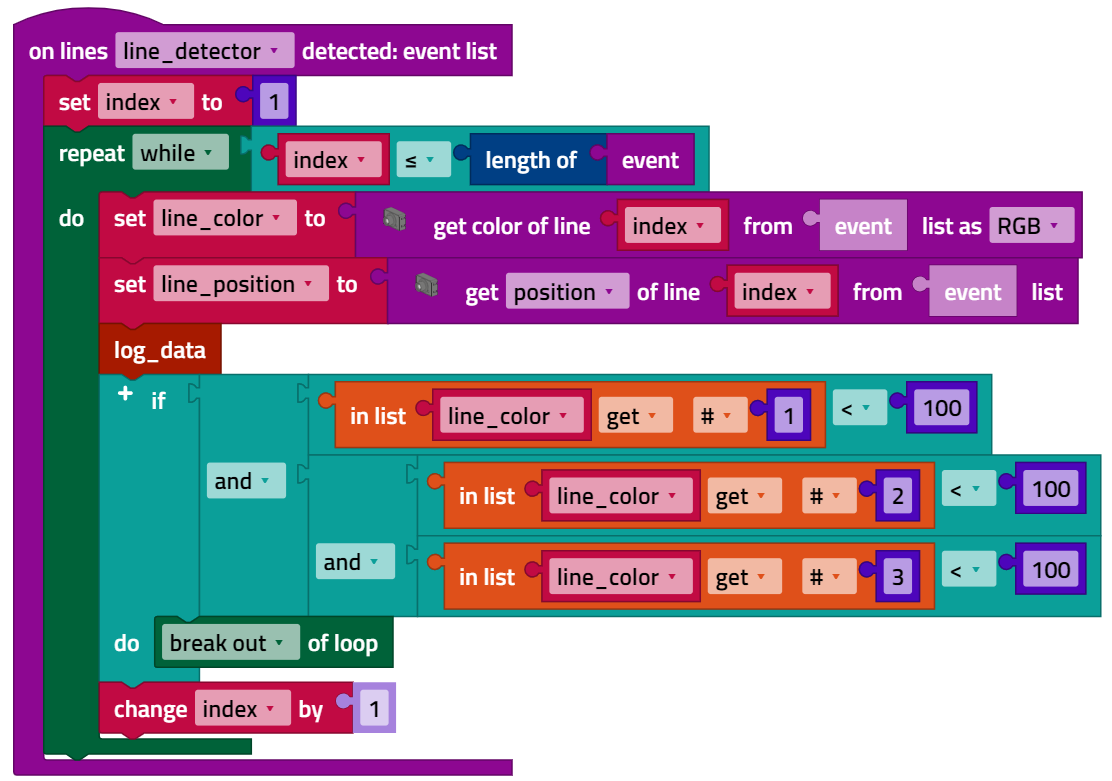


Die Linienerkennung sollte die gesamte Breite des Kamera-Sichtfelds umfassen. Sie sollte als schmaler Streifen gewählt werden, da die Erkennung ein Rechteck um den Bereich der Linie legt; dadurch wird die Linie in einer Kurve sonst sehr breit.

Da links und rechts der Linie auch Farbflächen auftreten können, muss die Liniener­kennung auf mindestens zwei Linien eingestellt werden.

Die schwarze Linie lässt sich am zuverlässigsten an der Farbe erkennen: Die RGB-Werte liegen alle deutlich unter 100, während bei den Farbflächen mindestens einer der drei RGB-Werte größer als 100 ist.

1b. Programmauszug (Beispiel) zur Anzeige von Farbe und Position von mehreren Spuren und der Erkennung der schwarzen Spur:



*Buggy\_Line\_Recognition.ft*

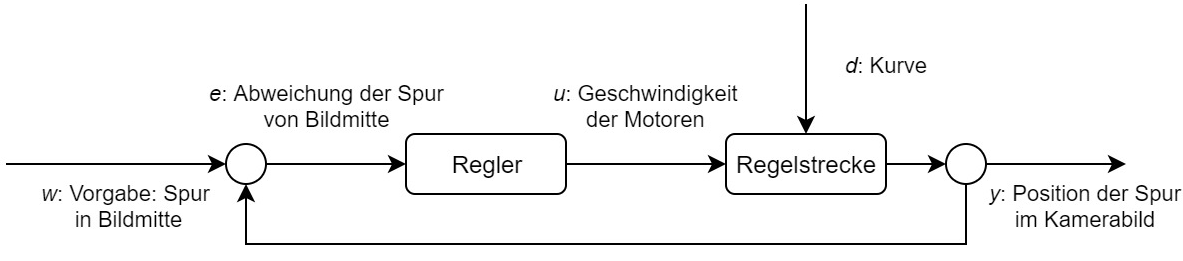
Es ist sinnvoll, für die Ausgabe der Daten auf der Konsole eine eigene Funktion („log\_data“) vorzusehen, um sie jederzeit schnell aus der zeitkritischen Event-Routine herauslöschen (oder hineinnehmen) zu können:



*Buggy\_Line\_Recognition.ft*

**2. Spurfolger mit Proportionalregler**

2a. Spurfolger mit der Kamera (Linienerkennung), dargestellt als Regelkreis:

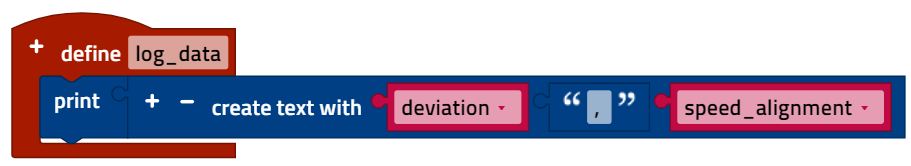


*Regelkreis\_Spurfolger.drawio*

2b. Einstellung des Proportionalreglers:

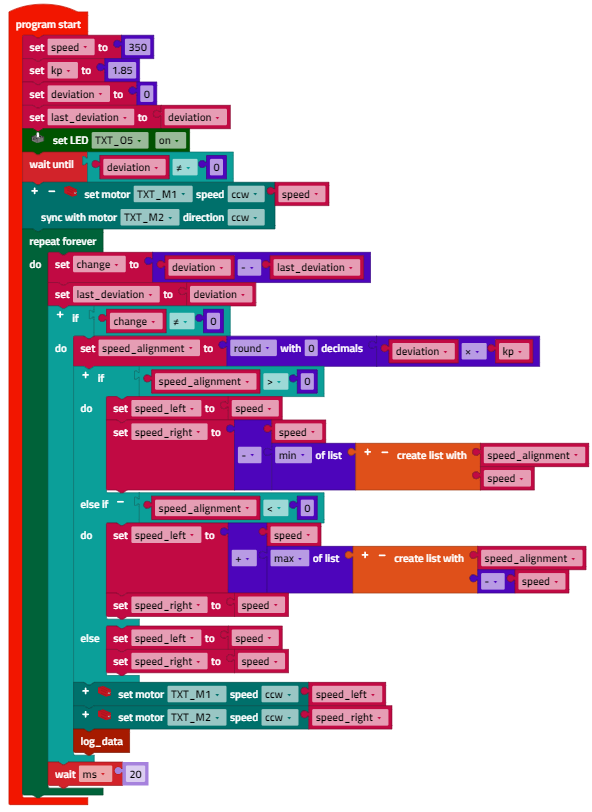
Durch die Lage des Linien-Erkennungsfensters der Kamera lässt sich die Vorausschau des Reglers verändern. Die Verstärkung (der Proportionalitätsfaktor) wird an einer gerade verlaufenden Spur experimentell bestimmt.

Bei einer Regelungs­frequenz von 50/s (Regelung alle 20 ms) liegt ein geeigneter Wert etwa im Bereich von 1,5-2. Damit nähert sich der Buggy langsam der Spur, fängt aber nicht an zu oszillieren. Die Regler-Werte können mit der Funktion „log\_data“ auf der Konsole ausgegeben werden:



*Buggy\_Line\_Follower\_with\_P\_Controller.ft*

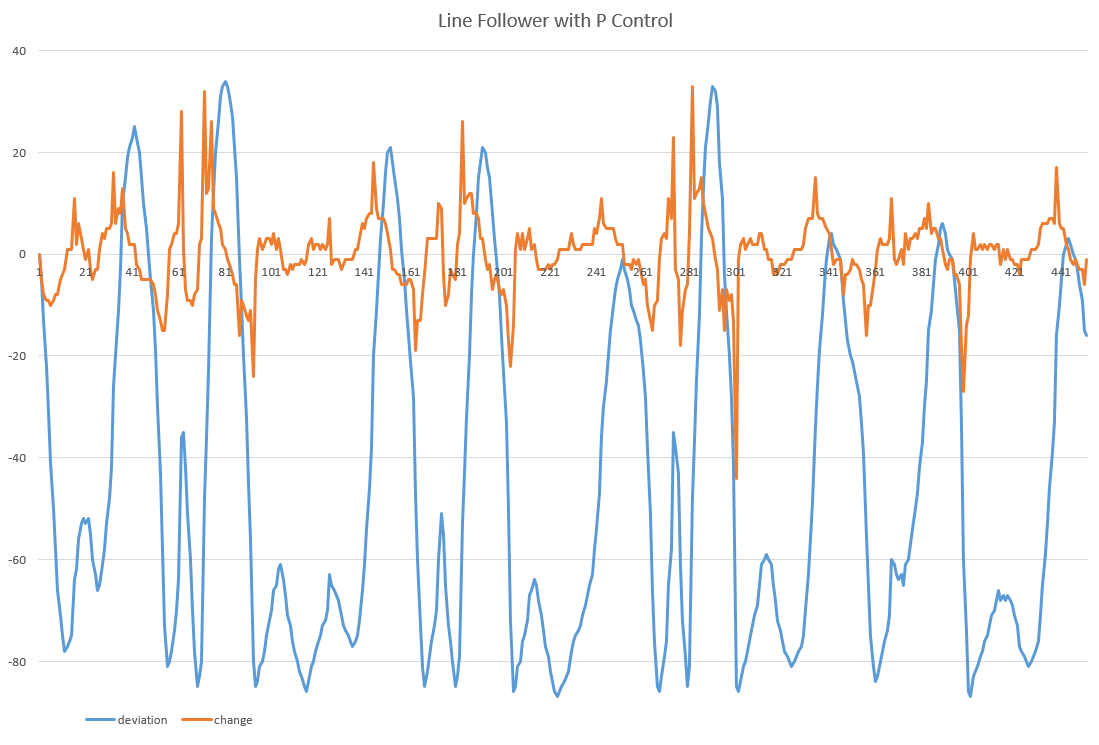
Programm (Beispiel) Spurfolger mit Proportional-Regler (mit ):



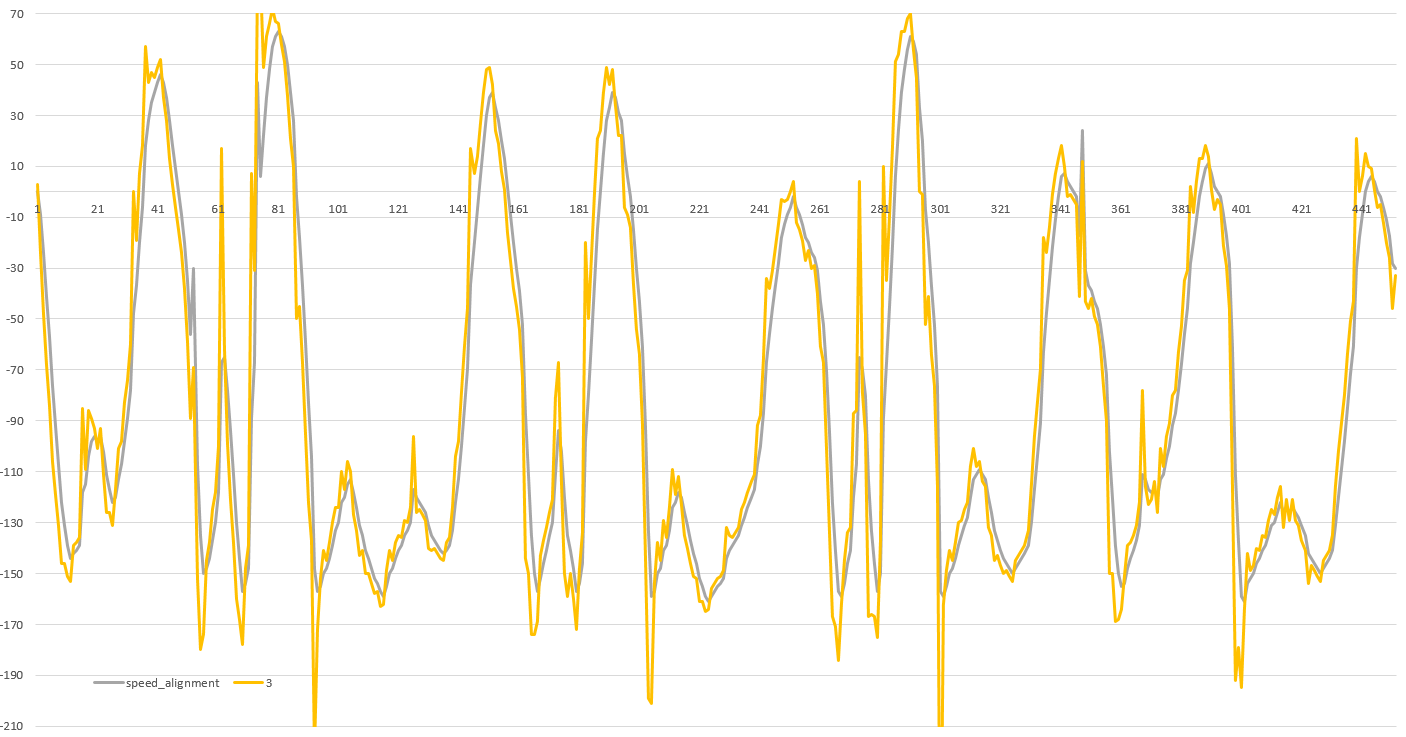
*Buggy\_Line\_Follower\_with\_P\_Controller.ft*

**3. Spurfolger mit PD-Regler**

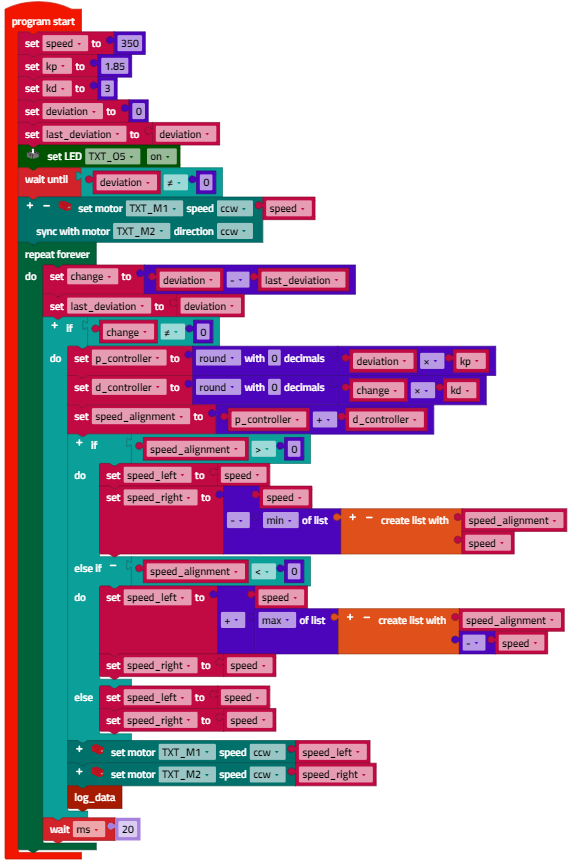
3a. Regler-Daten des P-Reglers: Abweichung von der Spur (blaue Linie), Größe der Änderung (orange-rote Linie):



Korrekturwerte P-Regler (graue Linie) und PD-Regler, hier mit (gelbe Linie):



3b. Programm (Beispiel) Spurfolger mit PD-Regler (mit und ):

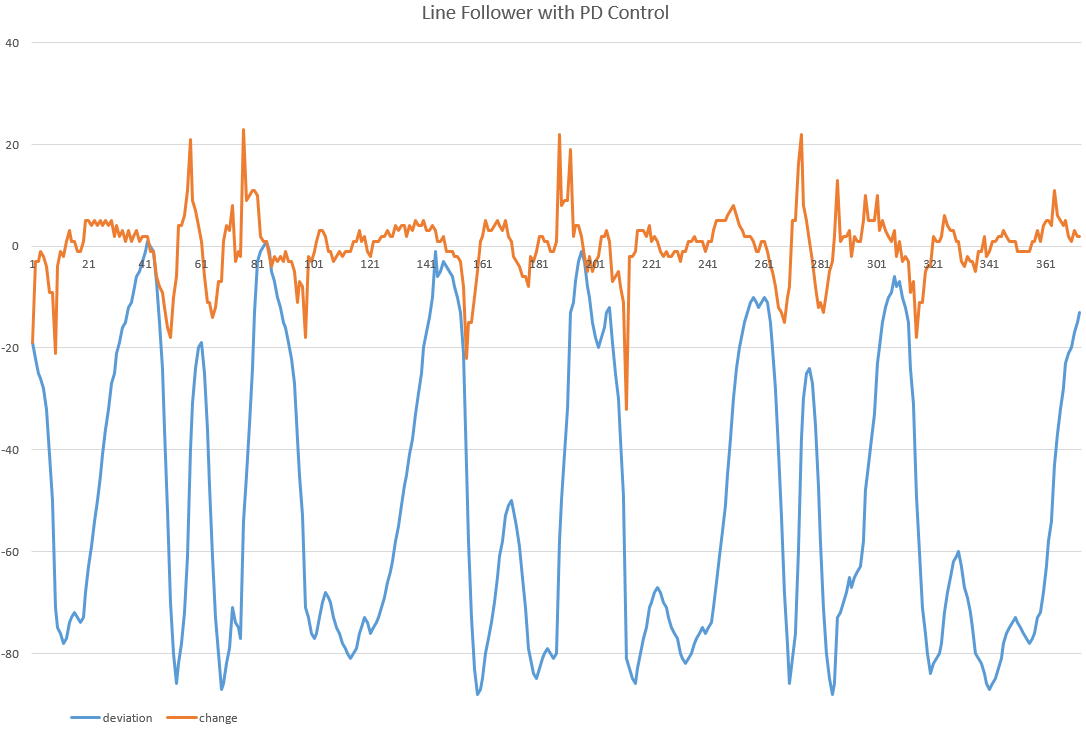


*Buggy\_Line\_Follower\_with\_PD\_Controller.ft*

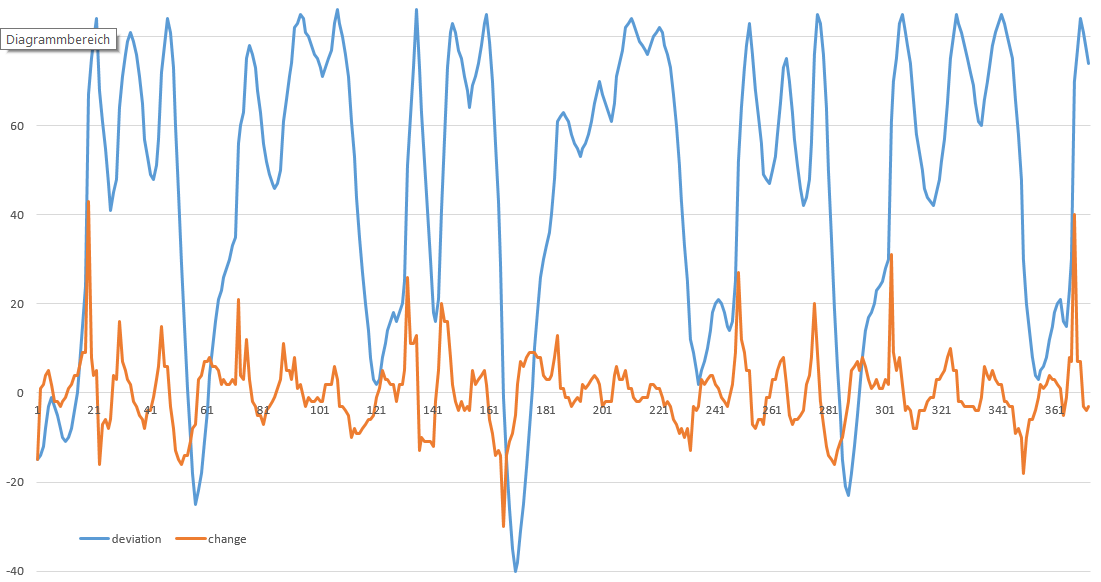
Funktion „log\_data“:



Regler-Daten des PD-Reglers: Abweichung von der Spur (blaue Linie), Größe der Änderung (orange-rote Linie):



*Regler-Verlauf bei Fahrt gegen den Uhrzeigersinn*



*Regler-Verlauf bei Fahrt im Uhrzeigersinn*

Anlagen

# Aufgabe 8: Spurfolger

## Erforderliches Material

* PC für Programmentwicklung, lokal oder über Web-Schnittstelle.
* USB-Kabel oder BLE- bzw. WLAN-Verbindung für die Übertragung des Programms auf den TXT4.0.
* Parcours-Bogen mit schwarzer, 2 cm breiter geschlossener Kreislinie und Farbflächen
* Hindernis (Karton, Dose, …)

## Weiterführende Informationen

[1] Wikipedia: [*Endlicher Automat (Zustandsautomat)*](https://de.wikipedia.org/wiki/Endlicher_Automat)

[2] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: [*Modeling Software with Finite State Machines. A Practical Approach*](http://is.ifmo.ru/download/modelingsoftwarewithfinitestatemachinesapracticalapproach.pdf). Auerbach Publications, 2006.

[3] Online-Diagrammeditor zur Erstellung von Zustandsübergangsdiagrammen (Format drawio): <https://www.diagrammeditor.de/>

[4] Wikipedia: [*Regelungstechnik*](https://de.wikipedia.org/wiki/Regelungstechnik).

[5] Wikipedia: [*Regler*](https://de.wikipedia.org/wiki/Regler).

[6] RN-Wissen: [*Regelungstechnik*](http://rn-wissen.de/wiki/index.php/Regelungstechnik).

[7] Tim Wescott: [*PID without a PhD*](https://www.magentacloud.de/lnk/oyB5kKLU). Embedded Systems Programming, 10/2000, S. 86-108.