hochschule mannheim

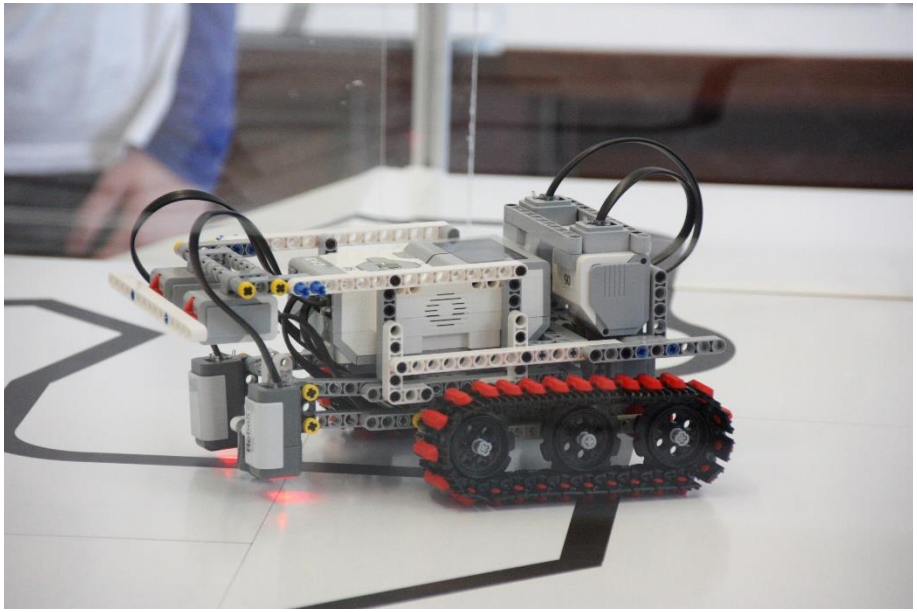
Fakultät für Informatik, Institut für Robotik

Laborpraktikum
Legorobotik in C – EV3

Ute Ihme

DAS LEGO® MINDSTORMS® System

Das EV3 System



Prinzip von LEGO® MINDSTORMS®

- Roboter wird gebaut mit
  - programmierbarem LEGO® Stein
  - bis zu 4 Motoren oder Lampen
  - bis zu 4 Sensoren
  - LEGO® TECHNIC Teile
- Erstellung eines Steuerprogramms am Computer
- Übertragen des Programms auf den Roboter
- Testen des Programms

DAS LEGO® MINDSTORMS® System

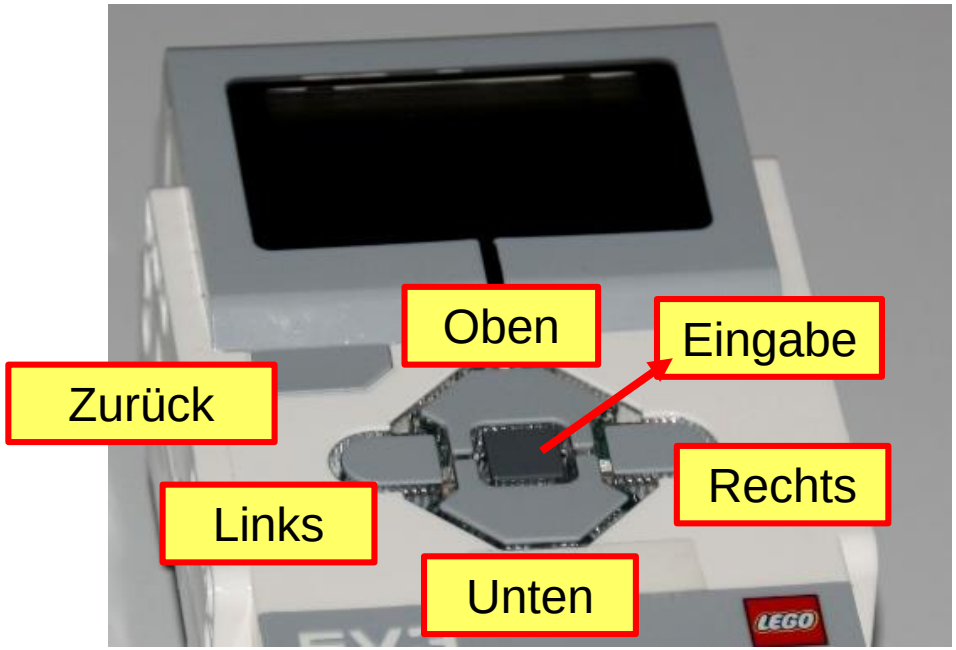
Der EV3 Stein

Motoren werden an die **Anschlüsse A, B, C und D** angeschlossen.

Sensoren werden an die **Anschlüsse 1, 2, 3 und 4** angeschlossen.

DAS LEGO® MINDSTORMS® System

Der EV3 Stein – Bezeichnung der Buttons



DAS LEGO® MINDSTORMS® System

Motoren

Quelle: Lego

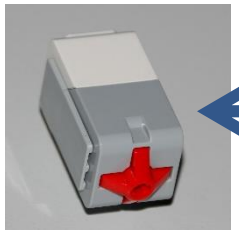
Motoren werden an die **Anschlüsse A, B, C und D** angeschlossen.

Servomotor

- Verfügt über integrierten **Rotationssensor**
  - misst Geschwindigkeit und Abstand
  - Leitet Ergebnisse an NXT Stein weiter
- Motor kann auf einen Grad genau gesteuert werden
- Kombinationen mehrerer Motoren möglich
  - arbeiten ggf. mit gleicher Geschwindigkeit

DAS LEGO® MINDSTORMS® System

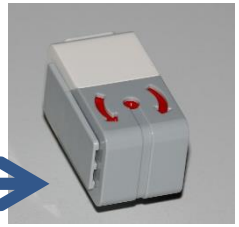
Standardsensoren



Berührungssensor
Tastsenor

Ultraschallsensor

Farbsensor

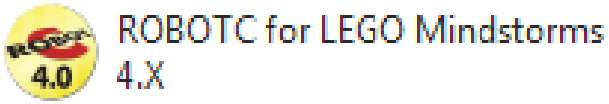


Gyrosensor

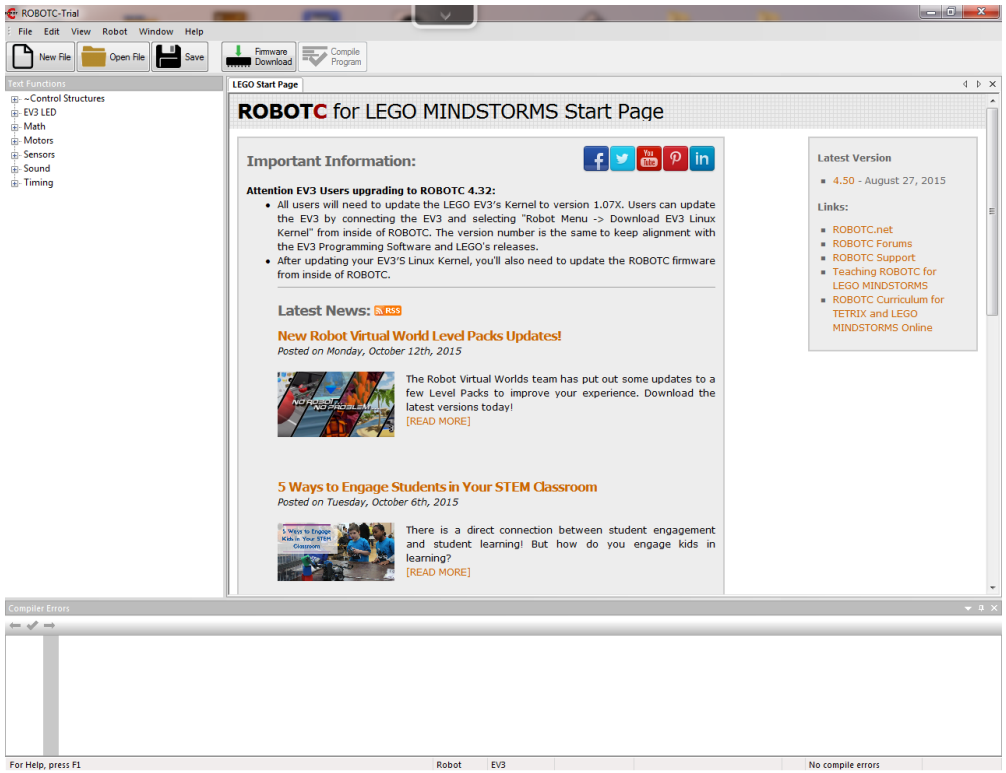
Start der Entwicklungsumgebung

Starten von RobotC

Startsymbol:



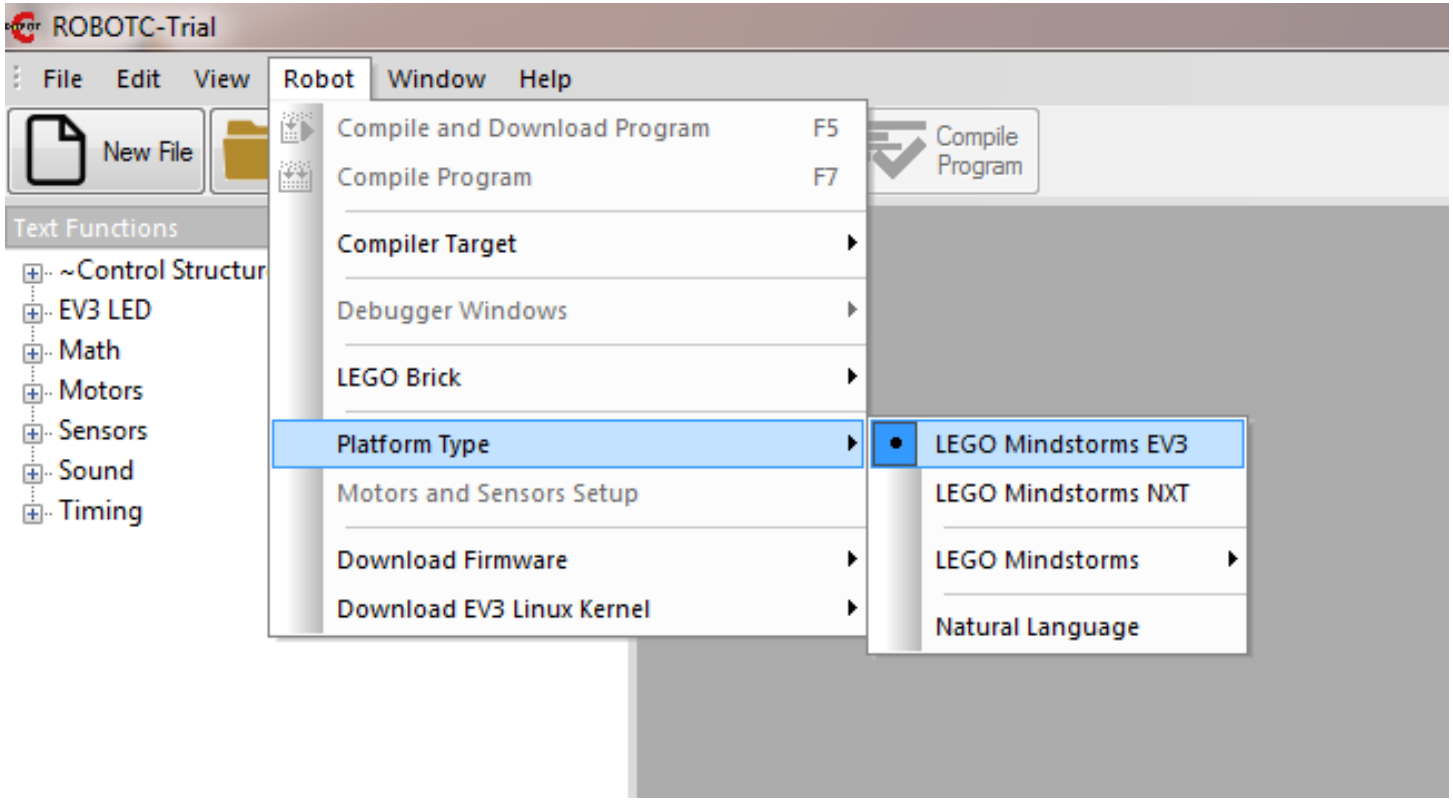
Startbildschirm:



Start der Entwicklungsumgebung

Starten von RobotC

Einstellen des Platformtypes – hier EV3 wählen

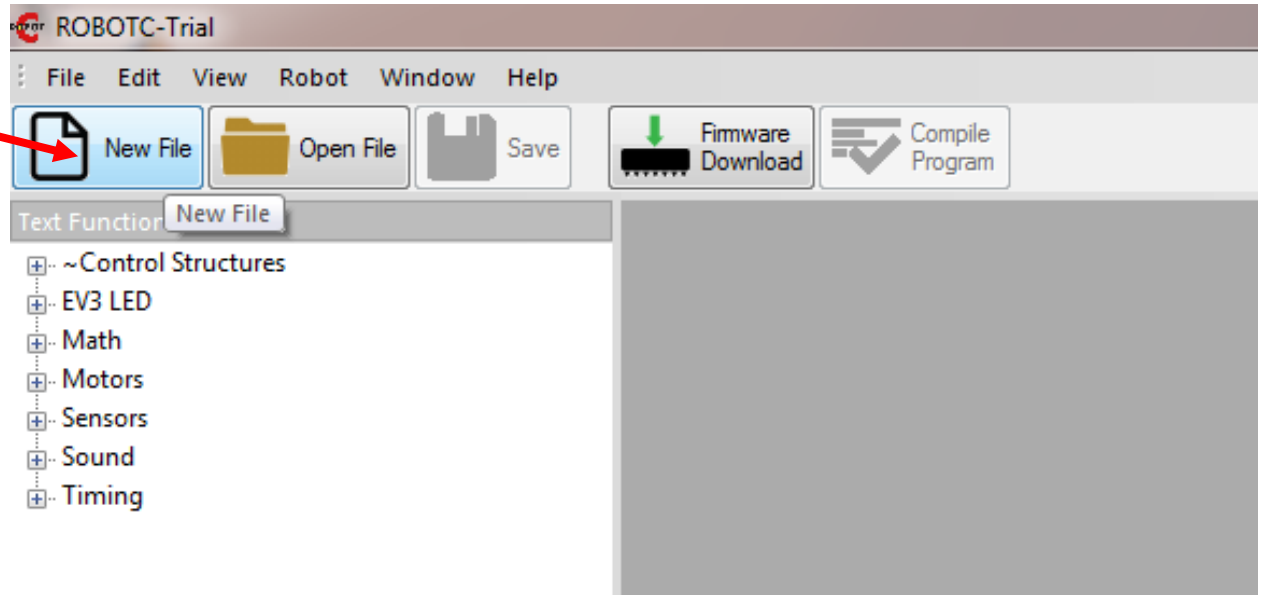


Start der Entwicklungsumgebung

Starten von RobotC

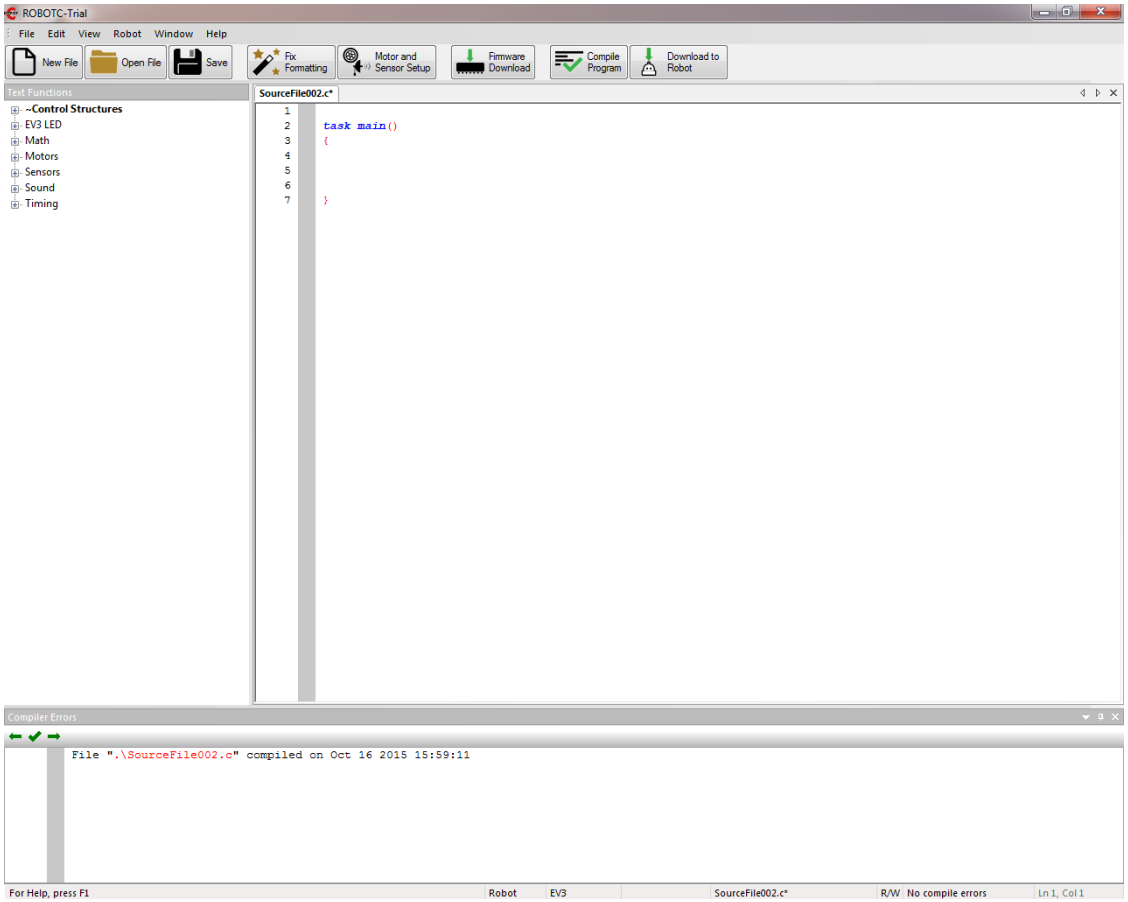
Erstellen einer neuen Datei

New File wählen



Start der Entwicklungsumgebung

Startbildschirm



Displayanzeige

displayTextLine(zeilennummer,text**)**

Parameter	Erklärung
zeilennummer	Zeile, in der der Text angezeigt wird
text	Anzuzeigender Text

Beispiel

```
displayTextLine(3,"Hello");
```

Displayanzeige löschen

eraseDisplay()

Beispiel

```
eraseDisplay()
```

Pausenbefehle

1. Sleep - Befehl

sleep(Zeit**)**	Beispiel
Zeit in ms	sleep(3000);

2. Warten auf Knopfdruck I

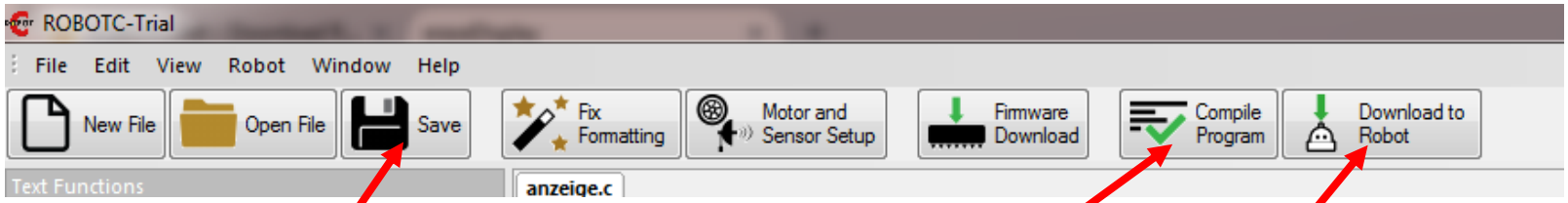
waitForButtonPress()

Dieser Befehl behält seinen Parameter, so dass dieser nicht mehrfach in einem Programm verwendet werden kann.

Beispielprogramm zur Display und Pause

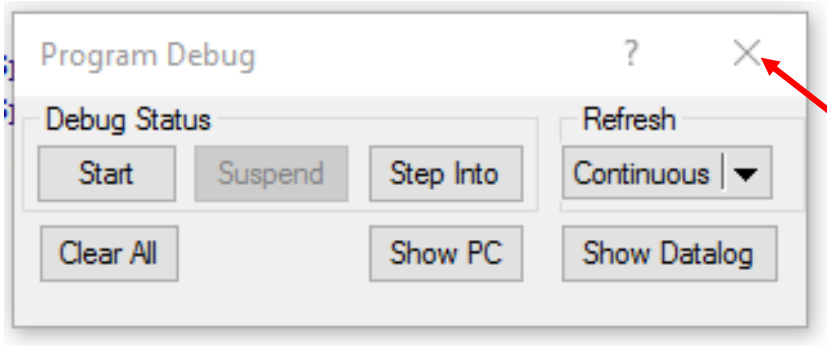
```
 5  task main()
 6  {
 7    displayTextLine(3,"Hello World");
 8    waitForButtonPress();
 9    eraseDisplay();
10    displayTextLine(4,"Robotic ind C");
11    sleep(3000);
12
13  }
```

Beispielprogramm zur Display und Pause



1. Programm Sichern

2. Programm kompilieren



3. Programm auf Roboter laden

4. Konsole schließen

5. Programm auf dem Roboter starten!

Pausenbefehle

3. Warten auf Knopfdruck II

waitUntil(getButtonPress(Knopf))

Dieser Befehl kann mehrfach verwendet werden. Allerdings muss eine Pause in Form einer Sleep-Funktion eingefügt werden. Siehe Beispiel

Beispiel – Mehrfaches Warten auf Knopfdruck

```
task main()
{
  displayTextLine(3,"Hello");
  sleep(1000);
  waitUntil(getButtonPress(buttonAny));
  displayTextLine(4,"Hello1");
  sleep(1000);
  waitUntil(getButtonPress(buttonAny));
  displayTextLine(5,"Hello2");
  sleep(1000);
  waitUntil(getButtonPress(buttonAny));
}
```

DAS SPIELFELD - Legostadt

Allgemeiner Aufbau

DAS SPIELFELD - Legostadt

Hinweise zur Bearbeitung der Praktikumsaufgaben

- Jede Aufgabe des Spielfeldes ist eine eigenständige Aufgabe. D. h. jede Aufgabe soll einzeln gelöst werden und muss nicht mit anderen Aufgaben kombiniert werden.
- Erstellen Sie ein Projekt mit einer Klasse, die eine Main Methode
- Schreiben Sie das entsprechende Programm
- Für eine neue Aufgaben, löschen Sie den nicht mehr benötigten Quelltext bzw. kommentieren diesen aus.
- Erstellen Sie für eine neue Aufgabe keine neue Klasse mit einer main-Methode in dem selben Projekt.
- Bei Bedarf erstellen Sie für eine neue Aufgabe ein neues Projekt mit einer neuen Klasse, die eine main-Methode enthält

DAS SPIELFELD - Legostadt

Übung 1: Bestimmung der Strecke, die der Roboter in 1 s vorwärts fährt

Start:
Einer der Übungsplätze Ü1, Ü2, Ü3 oder Ü4

Vorgehensweise:

- Schreiben eines Programm, dass den Roboter bei einer bestimmten Geschwindigkeit 1s vorwärts fahren lässt
- Platzieren des Roboters an der schwarzen Linie in einem der Übungsplätze
- Starten des Programms
- Messen der Strecke und Wert notieren
- Über Verhältsnisgleichungen kann man nun die Zeit bestimmen, die der Roboter braucht, um bestimmte Strecken zurückzulegen

DAS LEGO® MINDSTORMS® System

Motoren

Quelle: Lego

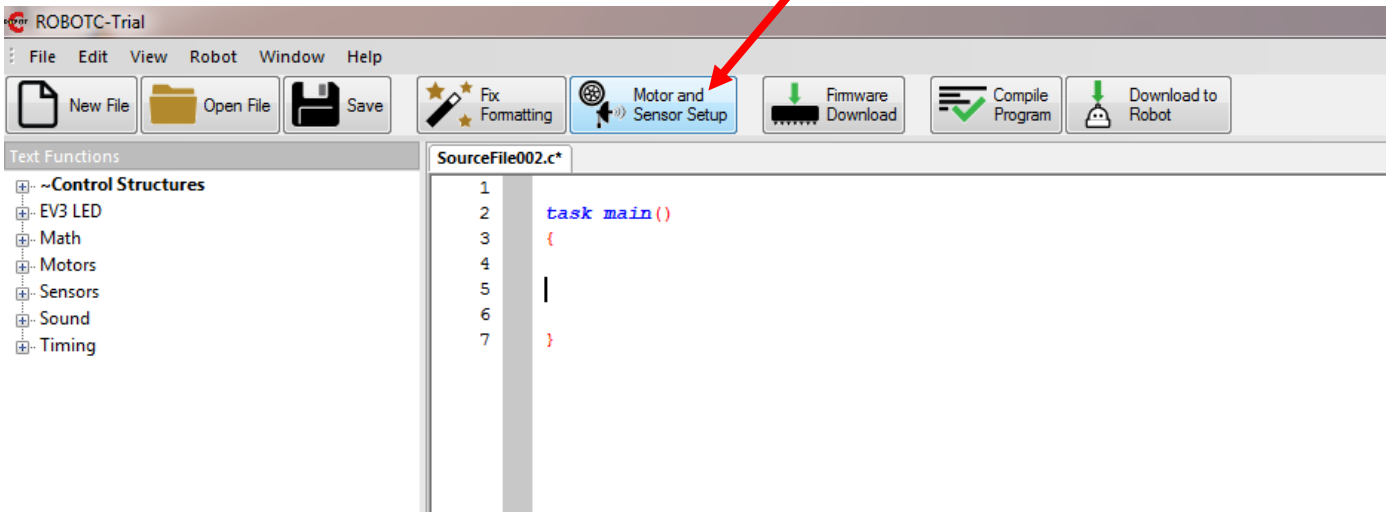
Servomotor

- Verfügt über integrierten **Rotationssensor**
  - misst Geschwindigkeit und Abstand
  - Leitet Ergebnisse an NXT Stein weiter
- Motor kann auf einen Grad genau gesteuert werden
- Kombinationen mehrerer Motoren möglich
  - arbeiten ggf. mit gleicher Geschwindigkeit

Arbeit mit RobotC

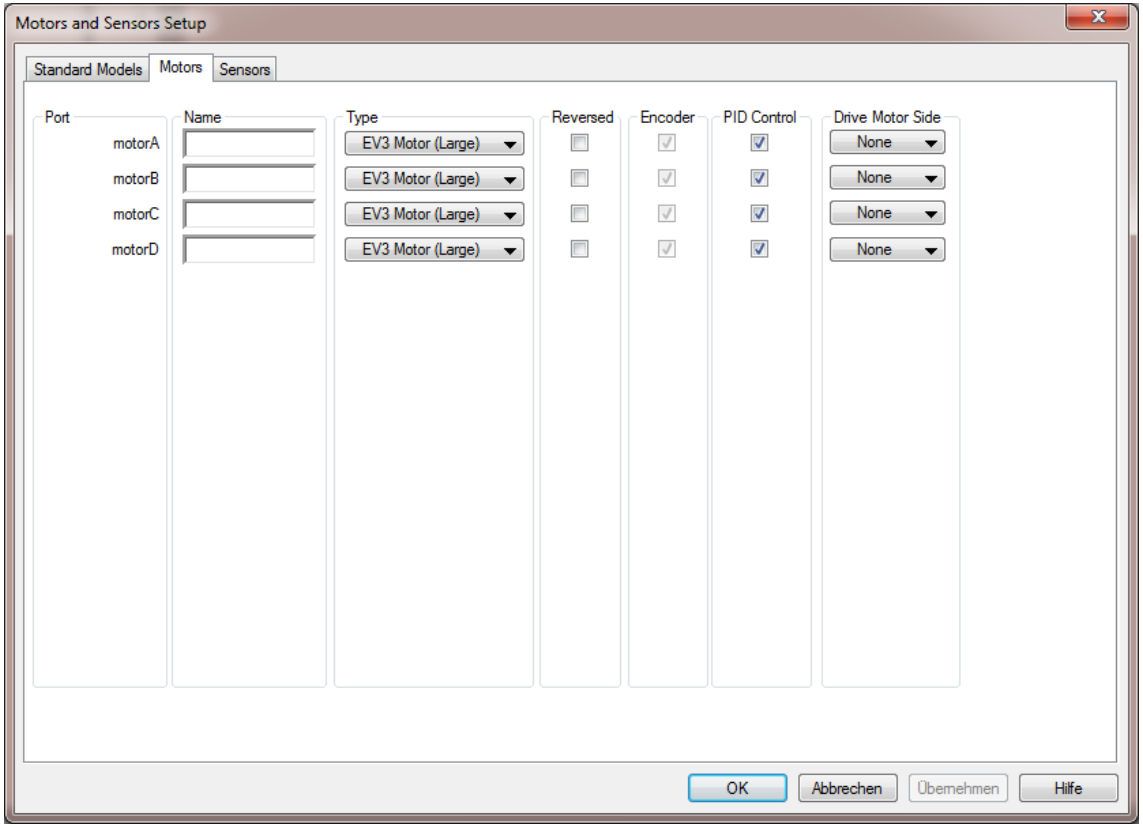
Setup der Motoren und Sensoren

Motor und Sensor Setup wählen



Arbeit mit RobotC

Setup der Motoren und Sensoren

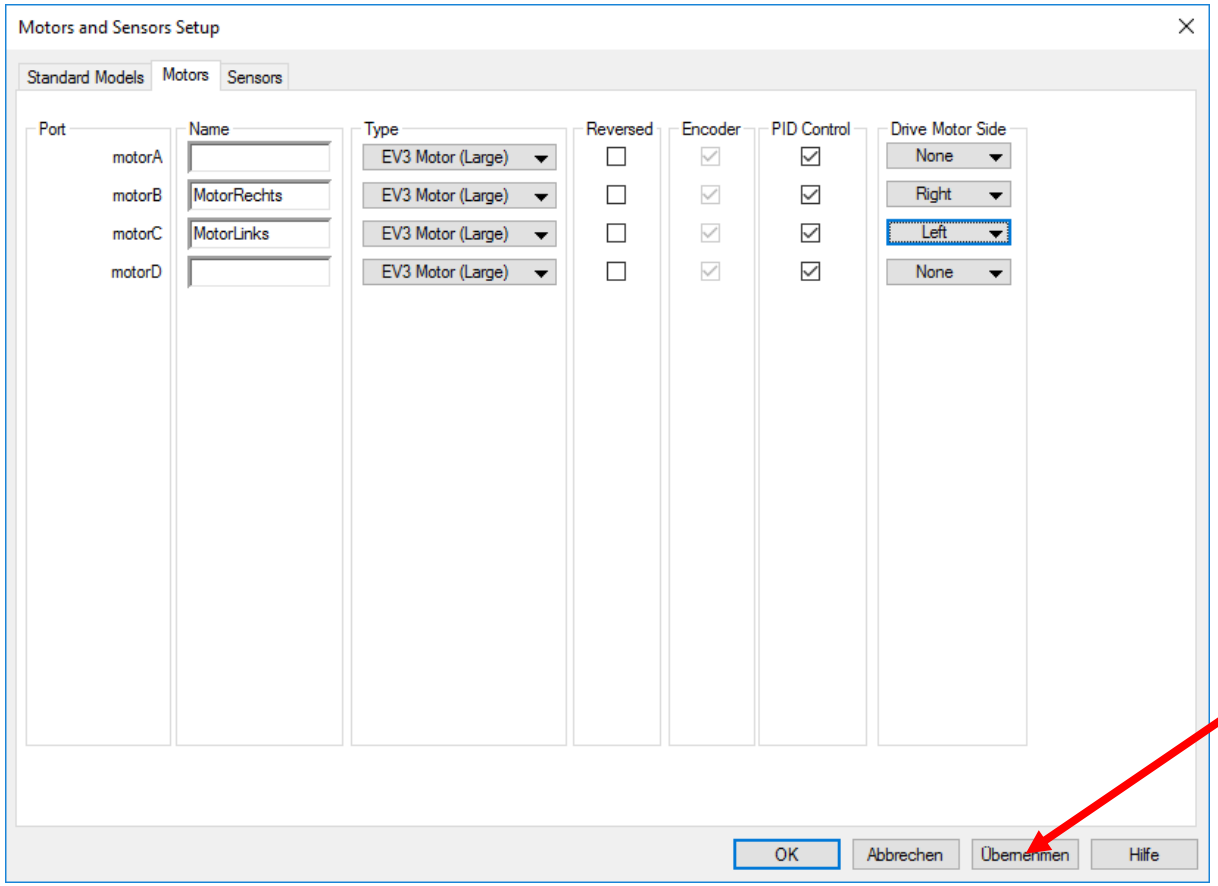


Einstellungen für die Motoren entsprechend der Roboterkonfiguration vornehmen

Siehe nächste Folie

Arbeit mit RobotC

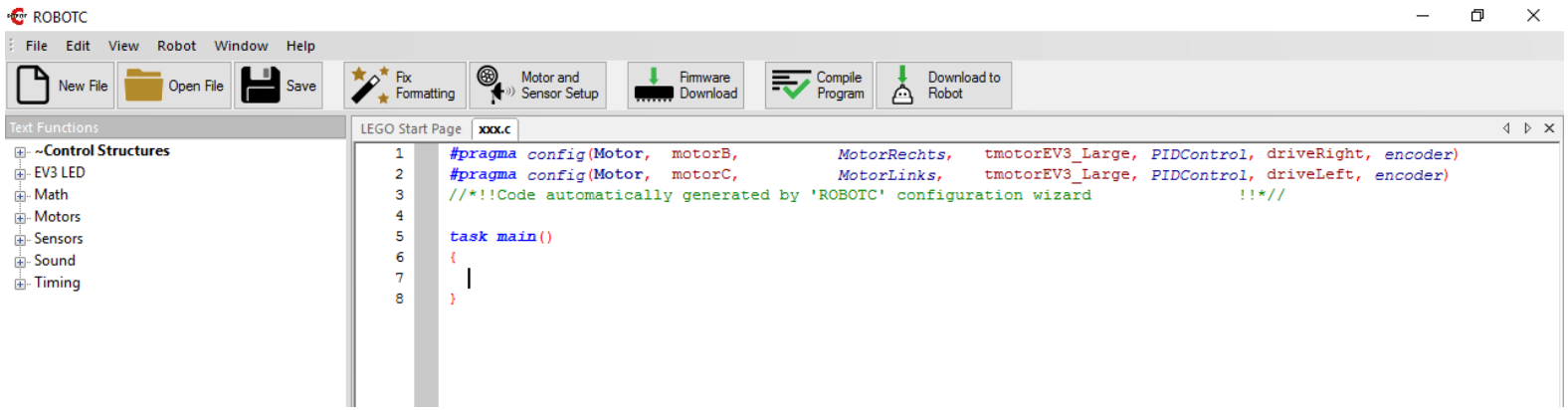
Setup der Motoren und Sensoren



auf Übernehmen klicken

Arbeit mit RobotC

Setup der Motoren und Sensoren



Befehle zur Motorsteuerung – Zeitgesteuert

setMotorSpeed(motorIndex,speed**)**

Parameter	Erklärung
motorIndex	Anzusteuernder Motor
speed	Geschwindigkeit Wertebereich: -100 bis 100 100: volle Geschwindigkeit vorwärts -100: volle Geschwindigkeit rückwärts 0: Stop

Befehle zur Motorsteuerung – Zeitgesteuert

```
// vorwaerts fahren
setMotorSpeed(motorB,70);
setMotorSpeed(motorC,70);

// rückwaerts fahren
setMotorSpeed(motorB,-70);
setMotorSpeed(motorC,-70);

// Anhalten
setMotorSpeed(motorB,0);
setMotorSpeed(motorC,0);
```

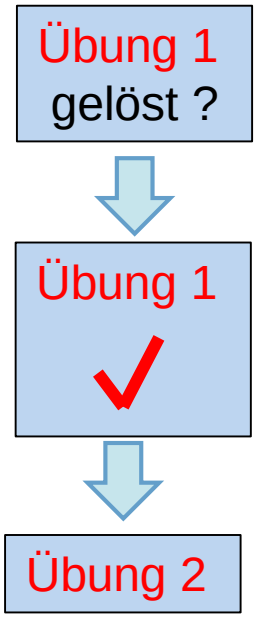
```
// Linkskurve
setMotorSpeed(motorB,-70);
setMotorSpeed(motorC,70);

// Rechtskurve
setMotorSpeed(motorB,70);
setMotorSpeed(motorC,-70);
```

Beispiel zur Motorsteuerung zeitgesteuert

Der Roboter fährt
- Geradeaus
- Dreht sich links herum
- Dreht sich rechts herum
- Fährt rückwärts
- Hält an.

```
task main()
{
  // vorwaerts fahren
  setMotorSpeed(motorB,70);
  setMotorSpeed(motorC,70);
  sleep(1000);

  // rückwaerts fahren
  setMotorSpeed(motorB,-70);
  setMotorSpeed(motorC,-70);
  sleep(1000);


  // Linkskurve
  setMotorSpeed(motorB,-70);
  setMotorSpeed(motorC,70);
  sleep(1000);

  // Rechtskurve
  setMotorSpeed(motorB,70);
  setMotorSpeed(motorC,-70);
  sleep(1000);

  // Anhalten
  setMotorSpeed(motorB,0);
  setMotorSpeed(motorC,0);
  sleep(1000);
}
```

DAS SPIELFELD: Legostadt



Übung 1: Bestimmung der Strecke, die der Roboter in 1 s vorwärts fährt

Start:
Einer der Übungsplätze Ü1, Ü2, Ü3 oder Ü4

Vorgehensweise:

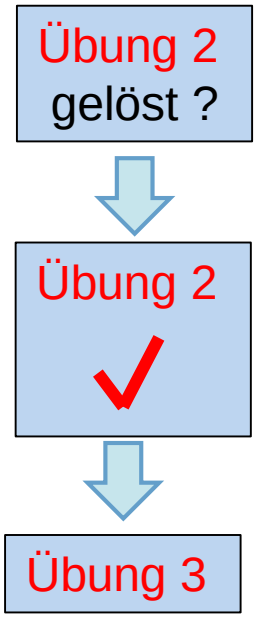
- Schreiben eines Programm, dass den Roboter bei einer bestimmten Geschwindigkeit 1s vorwärts fahren lässt
- Platzieren des Roboters an der schwarzen Linie in einem der Übungsplätze
- Starten des Programms
- Messen der Strecke und Wert notieren
- Über Verhältsnisgleichungen kann man nun die Zeit bestimmen, die der Roboter braucht, um bestimmte Strecken zurückzulegen

DAS SPIELFELD: Legostadt

Übung 2: Der Roboter soll von einem der Übungsplätze auf das Startfeld fahren

Start: Einer der Übungsplätze Ü1, Ü2, Ü3 oder Ü4

Ziel: Erreichen des Startfeldes



DAS SPIELFELD: Legostadt

Übung 3: Motorensteuerung - rotationsgesteuert

In dieser Übung sollen die Befehle zur rotationsgesteuert Motorensteuerung ausprobiert werden. Dazu machen Sie sich mit den Befehlen auf den nachfolgenden Folien vertraut und probieren Sie den Roboter vorwärts, rückwärts, nach links und rechts fahren zu lassen.

DAS SPIELFELD: Legostadt

Motorensteuerung - rotationsgesteuert

setMotorSyncEncoder(nmotorOne,nmotorTwo,nTurnRation, nEncoderCount,nSignedPower)

nTurnRation	
100	Positiver Antrieb auf MotorOne und negativer Antrieb auf MotorTwo
-100	Negativer Antrieb auf MotorOne und Positiver Antrieb auf MotorTwo
0	Gleicher Antrieb auf beide Motoren
50	Antrieb nur auf MotorOne; MotorTwo still
-50	MotorOne stiell, Antrieb nur auf MotorTwo

DAS SPIELFELD: Legostadt

Motorensteuerung - rotationsgesteuert

setMotorSyncEncoder(nmotorOne,nmotorTwo,nTurnRation, nEncoderCount,nSignedPower)

nmotorOne	Port des ersten Motors
nmotorTwo	Port des zweiten Motors
nTurnRation	Verhältnis vom ersten zum zweiten Motor
nEncoderCount	Rotationswinkel
nSignedPower	Powerlevel

DAS SPIELFELD: Legostadt

Motorensteuerung - rotationsgesteuert

waitUntilMotorStop(nmotor)

Befehl ersetzt einen sleep()-Befehl. Das Programm wartet mit der weiteren Ausführung darauf, das der Motor anhält.

DAS SPIELFELD: Legostadt

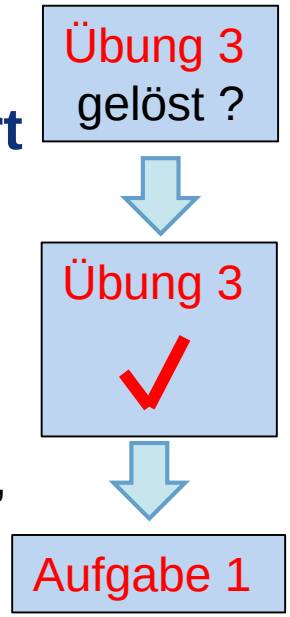
Motorensteuerung – rotationsgesteuert Programmbeispiel

```
task main()
{
  //vorwärts
  setMotorSyncEncoder(motorB,motorC,0,1000,70);
  waitUntilMotorStop(motorB);
  //rechts
  setMotorSyncEncoder(motorB,motorC,100,1000,70);
  waitUntilMotorStop(motorB);
  //links
  setMotorSyncEncoder(motorB,motorC,-100,1000,70);
  waitUntilMotorStop(motorB);
  //rückwärts
  setMotorSyncEncoder(motorB,motorC,0,1000,-70);
  waitUntilMotorStop(motorB);

}
```

DAS SPIELFELD: Legostadt

Übung 3: Motorensteuerung - rotationsgesteuert

In dieser Übung sollen die Befehle zur rotationsgesteuert Motorensteuerung ausprobiert werden. Dazu machen Sie sich mit den Befehlen auf den nachfolgenden Folien vertraut und probieren Sie den Roboter vorwärts, rückwärts, nach links und rechts fahren zu lassen.



DAS SPIELFELD: Legostadt

Die Übungsphase ist vorbei.
Alles verstanden?
Wenn ja, dann kann mit der Bearbeitung der nachfolgenden Aufgaben begonnen werden.

Die nachfolgenden Aufgaben müssen durch einen Praktikumsbetreuer abgenommen und unterschrieben bzw. bewertet werden!

DAS SPIELFELD: Legostadt

Aufgabe 1: Fahrt zum Haus

Start: Startfeld

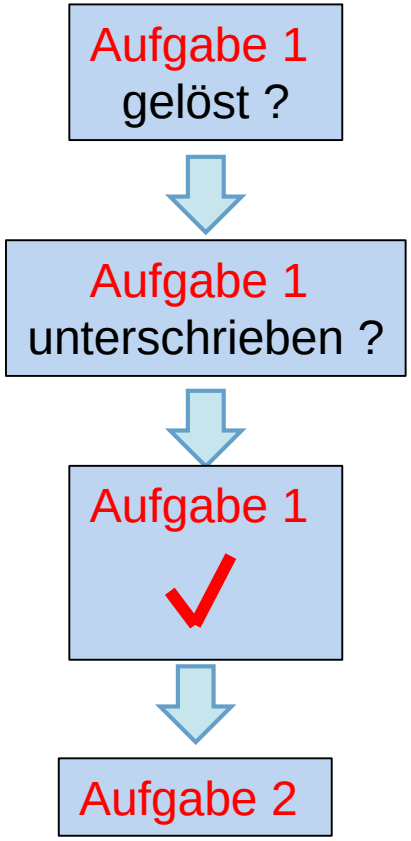
Ende: Parkplatz am Haus

Der Roboter soll vom Startplatz zum Parkfläche am Haus fahren. Dabei soll er der vorgegebenen Straße.

Ziel:

Festigung der Steuerung des Roboters.

- Geradeausfahren
- Kurvenfahren



C CODE

Artithmetische Operatoren

Operator	Beispiel	Wirkung
+	a + b	Addiert a und b
-	a – b	Subtrahiert b von a
*	a * b	Multipliziert a und b
/	a / b	Dividiert a durch b
%	a % b	Liefert den Rest bei der Division a durch b

C CODE

Vergleichsoperatoren

Operator	Beispiel	Wirkung
>	a > b	a größer als b
>=	a >= b	a größer oder gleich b
<	a < b	a kleiner als b
<=	a <= b	a kleiner oder gleich b
==	a == b	a ist gleich b
!=	a != b	a ist ungleich b

C CODE

Logische Operatoren (Auswahl)

Operator	Beispiel	Wirkung
&&	a && b	a und b müssen erfüllt sein
||	a || b	a oder b muss erfüllt sein

DAS SPIELFELD: Legostadt

Aufgabe 2: Beförderung von Fahrgästen zwischen Bahnhof und Airport (for Schleife)

Start und Ende: Parkfläche Bahnhof

Der Roboter soll als Shuttlebus Gäste zwischen Bahnhof und Airport hin und zurück befördern.

Der Roboter startet per Knopfdruck, wenn der Gast eingestiegen ist. Der Roboter fährt die Strecke vom Bahnhof zum Airport vorwärts. Lässt Gäste ein- und aussteigen und fährt nach Knopfdruck die gleiche Strecke rückwärts zurück.

Die Zahl soll angezeigt werden.

Insgesamt soll der Roboter die Strecke 3mal absolvieren!

Auf den Parkflächen darf der Roboter neu ausgerichtet werden!

C CODE

Die for Schleife

Eine Anweisung bzw. eine Folge von Anweisungen soll mehrfach wiederholt werden.

```
for(Startwert;Endwert;Erhöhung)
{
    Anweisung
    …
    Anweisung
}
```

Beispiel:

```
for(i=1;i<=7;i++)
{
    Anweisung
    …
    Anweisung
}
```

C CODE

Bespiel für for Schleife

Das Wort Hello soll in 5 Zeilen untereinander angezeigt werden.

```
1
2   task main()
3   {
4
5     for(int i = 1;i<=5;i++)
6     {
7       displayTextLine(i,"Hello");
8       sleep(2000);
9
10    }
11
12  }
```

DAS SPIELFELD: Legostadt

Aufgabe 2: Beförderung von Fahrgästen zwischen Bahnhof und Airport (for Schleife)

Start und Ende: Parkfläche Bahnhof

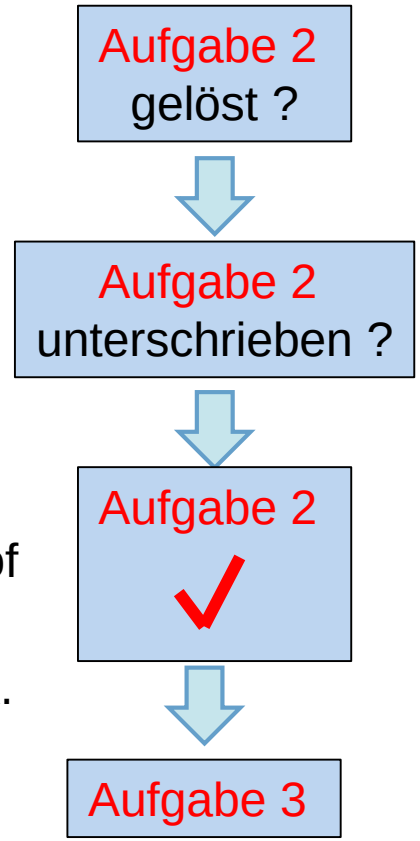
Der Roboter soll als Shuttlebus Gäste zwischen Bahnhof und Airport hin und zurück befördern.

Der Roboter startet per Knopfdruck, wenn der Gast eingestiegen ist. Der Roboter fährt die Strecke vom Bahnhof zum Airport vorwärts. Lässt Gäste ein- und aussteigen und fährt nach Knopfdruck die gleiche Strecke rückwärts zurück.

Die Zahl soll angezeigt werden.

Insgesamt soll der Roboter die Strecke 3mal absolvieren!

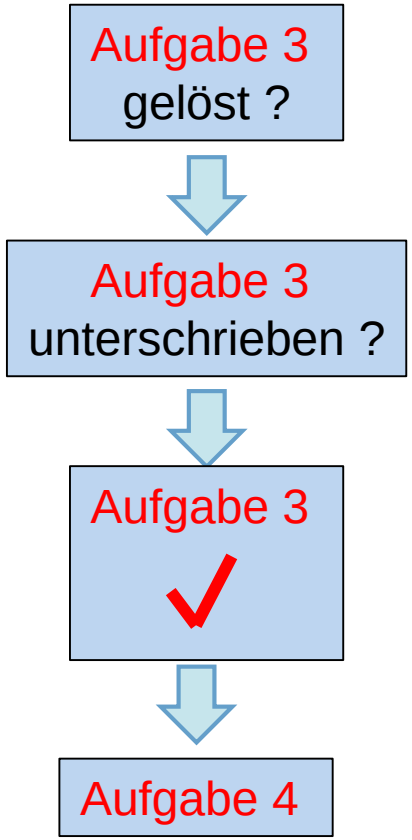
Auf den Parkflächen darf der Roboter neu ausgerichtet werden!



DAS SPIELFELD: Legostadt

Aufgabe 3: Anfahren und Anhalten

Schreiben Sie ein Programm, das den Roboter langsam anfahren lässt
(schrittweise Erhöhen der Power) danach eine gewisse Zeit mit gleichbleibender Geschwindigkeit fährt und danach langsam abbremst.



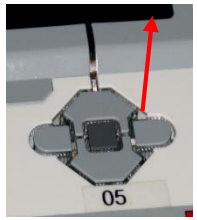
DAS SPIELFELD: Legostadt

Aufgabe 4: if … else Abfrage
Der Roboter soll entweder zum Krankenhaus oder zur Schule fahren

Start: Parkplatz am Haus

Ende: Parkplatz Krankenhaus bzw. Ein- und Ausstiegsfeld an der Schule

Oben



Der Roboter soll vom Parkplatz am Haus entweder zur Schule oder zum Krankenhaus fahren. Die Auswahl des Ziel erfolgt in Abhängigkeit vom Button, der am EV3 Stein gedruckt wird. Wird der obere Knopf gedrückt, soll der Roboter zum Krankenhaus, in allen anderen Fällen zur Schule fahren. Beide Wege sind gleichzeitig zum implementieren!
Das Ziel soll angezeigt werden.

Die if – else Anweisung

Wenn der Ausdruck erfüllt ist, so werden die Anweisungen im if-Block erfüllt, ansonsten die Anweisung im else-Block.

```
if(Ausdruck){
  Anweisung
  …
  Anweisung
}
else{
  Anweisung
  …
  Anweisung
}
```

Beispiel:

```
if(a==10){
  Anweisung
  …
  Anweisung
}
else{
  Anweisung
  …
  Anweisung
}
```

C- CODE

Bespiel für if – else Anweisung

Das Programm zeigt an, ob der Weg 1 oder der Weg 2 genommen werden soll. Der Weg 1, soll genommen werden, wenn der linke Button auf dem EV3 Stein gedrückt wurde. In allen anderen Fällen der Weg 2.

```
1  task main()
2  {
3    waitForButtonPress();
4    if(getButtonPress(LEFT_BUTTON)==1)
5    {
6      displayTextLine(1,"Weg 1");
7      sleep(2000);
8    }
9    else
10   {
11     displayTextLine(1,"Weg 2");
12     sleep(2000);
13   }
14
15 }
```

getButtonPressed Anweisung

Gibt zurück, ob ein bestimmter Button gedrückt wurde.

`getButtonPress`

	bool getButtonPress(tEV3Buttons button)	
Parameter	**Explanation**	**Data Type**
Return Type	The function returns a boolean value.	bool
button	The EV3 button to be checked.	tEV3Buttons
param2	Description of parameter 2.	long

- Returns whether the specified button is pressed or not
  - Pressed buttons will return a value of 1
  - Released buttons will return a value of 0
- The button mapping can be found below:

buttonNone:	No button (0)
buttonUp:	Up button (1)
buttonEnter:	Enter button (2)
buttonDown:	Down button (3)
buttonRight:	Right button (4)
buttonLeft:	Left button (5)
buttonBack:	Back button (6)
buttonAny:	Any button (7)

DAS SPIELFELD: Legostadt

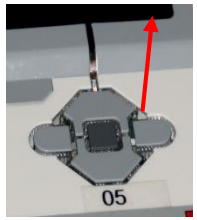
Aufgabe 4: if … else Abfrage
Der Roboter soll entweder zum Krankenhaus oder zur Schule fahren

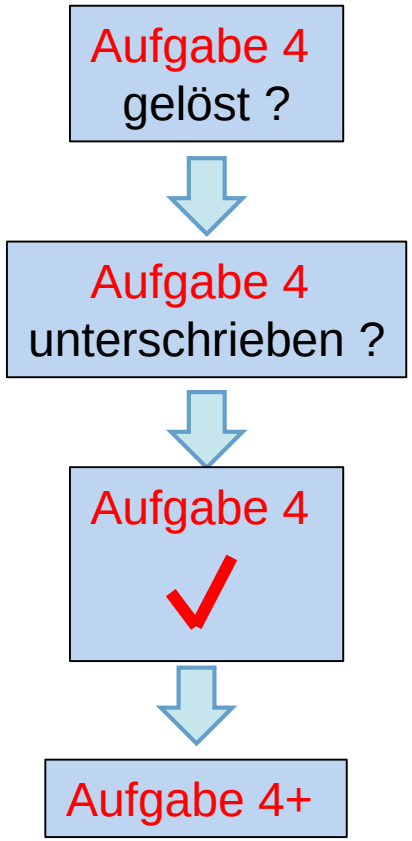
Start: Parkplatz am Haus

Ende: Parkplatz Krankenhaus bzw. Ein- und Ausstiegsfeld an der Schule

Der Roboter soll vom Parkplatz am Haus entweder zur Schule oder zum Krankenhaus fahren. Die Auswahl des Ziel erfolgt in Abhängigkeit vom Button, der am EV3 Stein gedruckt wird. Wird der obere Knopf gedrückt, soll der Roboter zum Krankenhaus, in allen anderen Fällen zur Schule fahren. Beide Wege sind gleichzeitig zum implementieren!
Das Ziel soll angezeigt werden.

Oben





DAS SPIELFELD: Legostadt

Aufgabe 4+: Erweiterung von Aufgabe4

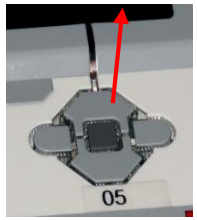
Start: Parkplatz am Haus

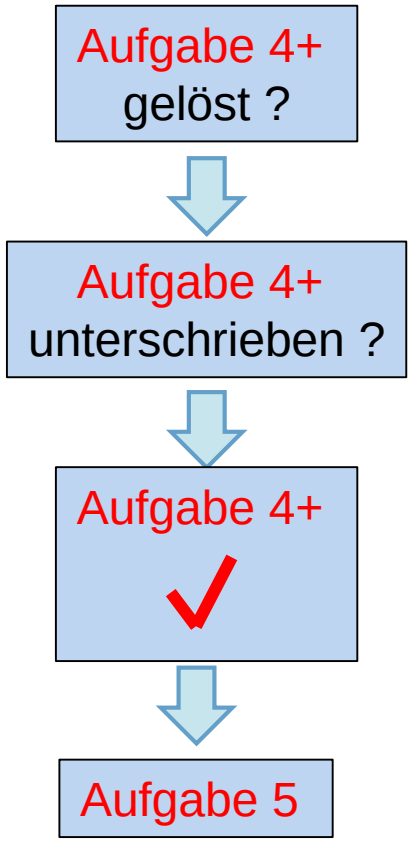
Ende: Parkplatz Krankenhaus bzw. Ein- und Ausstiegsfeld an der Schule

Erweitern Sie das Programm von Aufgabe 4, so dass

a) Die Auswahl zweifach ausgeführt werden kann

b) Die Auswahl beendet wird, wenn die mittlere Taste gedrückt wird. Realisieren Sie dies mit einer while-Schleife und berücksichtigen Sie die mittlere Taste in Ihrer if-Abfrage!





DAS SPIELFELD: Legostadt

Aufgabe 5: switch

Start: Parkplatz am Haus

Taste Oben: Parkplatz Krankenhaus

Taste Links: Ein- und Ausstiegsfeld an der Schule

Taste Rechts: Ladenstraße

Taste Unten: Berghütte

Schreiben Sie ein Programm, das den Roboter in Abhängigkeit der gedrückten Taste, zu einem bestimmten Ort fahren lässt. Verwenden Sie dazu einen switch. Die Fahrwege sind als einzelne Tasks zu deklarieren!

C Code

switch

```
switch(button)
{
case 1: startTask(anzeige1);
  break;
case 2: startTask(anzeige2);
  break;
case 3: startTask(anzeige3);;
  break;
default: displayTextLine(3,"nicht");
  break;
}
```

C Code

Deklaration von Tasks

```
task anzeige1()
{
  displayTextLine(3,"oben");
}
```

DAS SPIELFELD: Legostadt

Aufgabe 5: switch

Start: Parkplatz am Haus

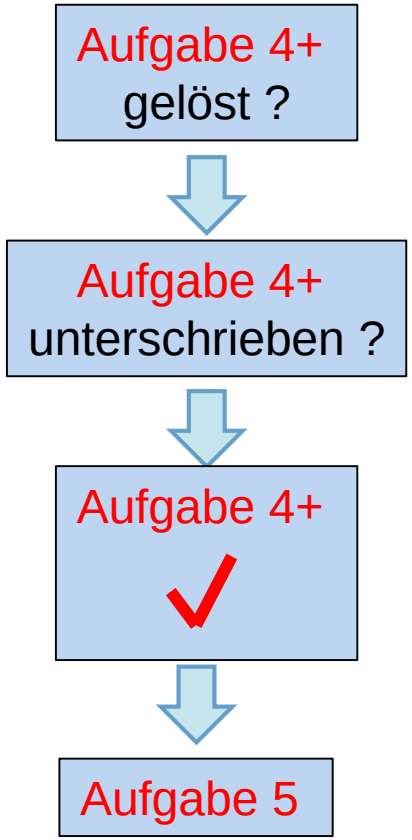
Taste Oben: Parkplatz Krankenhaus

Taste Links: Ein- und Ausstiegsfeld an der Schule

Taste Rechts: Ladenstraße

Taste Unten: Berghütte

Schreiben Sie ein Programm, das den Roboter in Abhängigkeit der gedrückten Taste, zu einem bestimmten Ort fahren lässt. Verwenden Sie dazu einen switch. Die Fahrwege sind als einzelne Tasks zu deklarieren!



Die Legostadt

Der Roboter braucht ein Update mit Sensoren!
- Bitte ans Laborpersonal wenden! -

DAS SPIELFELD: Legostadt

Aufgabe S1:
Fahrt zum Parkplatz Berghütte – Automatisches Anhalten mittels Tastsensor

Start: Parkplatz Bahnhof

Ende: Parkplatz Berghütte

Der Roboter soll vom Bahnhof zur Berghütte fahren. Mittels Tastsensor soll er selbständig anhalten, sobald der Tastsensor die Mauer vor dem Parkplatz berührt.

Die while Schleife
1. Unendliche while Schleife

Eine Anweisung bzw. eine Folge von Anweisungen soll unendlich oft wiederholt werden.

```
while(true)
{
    Anweisung
    …
    Anweisung
}
```

Beispiel für unendliche while Schleife

Das Wort Hello wird solange angezeigt, bis das Programm abgebrochen wird.

```
1
2   task main()
3   {
4
5     while(true)
6     {
7       displayTextLine(1,"Hello");
8     }
9
10  }
```

Die while Schleife
2. Endliche while Schleife

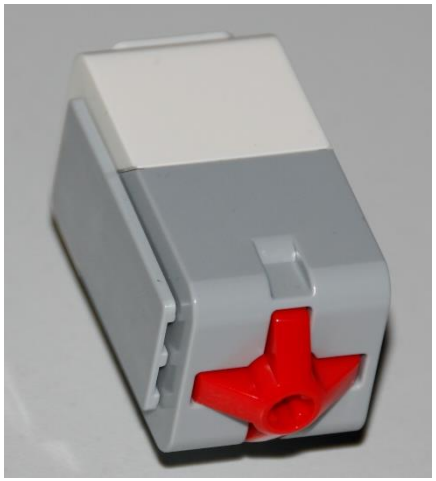
Eine Anweisung bzw. eine Folge von Anweisungen soll bis zu eine bestimmten Bedingung nicht mehr erfüllt is, wiederholt werden.

```
while(Bedingung)
{
    Anweisung
    …
    Anweisung
}
```

Beispiele für Bedingungen:
i==3;
pressed == true;
i<=5;
pressed!=false;

DAS LEGO® MINDSTORMS® System

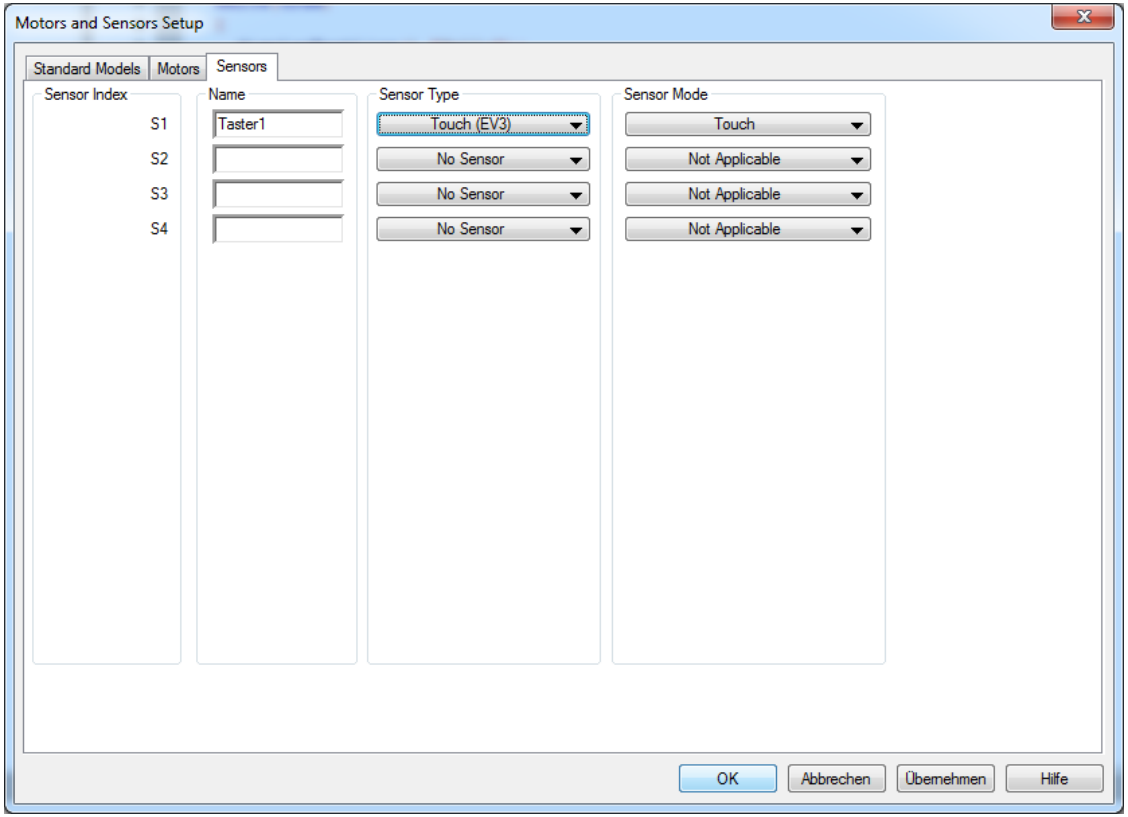
Berührungssensor / Tastsensor



- Abfrage, ob Sensor gedrückt
- Werte des Sensors
  - 0: Sensor nicht gedrückt
  - 1: Sensor gedrückt

Beispiel für Nutzung des Tastsensors mittels endlicher while Schleife

1. Sensor Setup durchführen



Beispiel für Nutzung des Tastsensors mittels endlicher while Schleife

getTouchValue

	bool getTouchValue(tSensors nDeviceIndex)	
Parameter	**Explanation**	**Data Type**
Return Type	The function returns a boolean value.	bool
nDeviceIndex	A sensor port or name.	tSensors

- Returns the value of the Touch Sensor plugged into nDeviceIndex
  - A logical 1 is returned if the Touch Sensor is pressed
  - A logical 0 is returned if the Touch Sensor is not pressed (released)

Show Code Sample

Beispiel für Nutzung des Tastsensors mittels endlicher while Schleife

```
1  #pragma config(Sensor, S1,     Taster1,        sensorEV3_Touch)
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard
3
4  task main()
5  {
6
7    while(getTouchValue(S1)==0)
8    {
9      displayTextLine(1,"Hello");
10   }
11
12 }
```

Das Wort Hello wird solange angezeigt, bis der Tastsensor am Port S1 gedrückt wird.

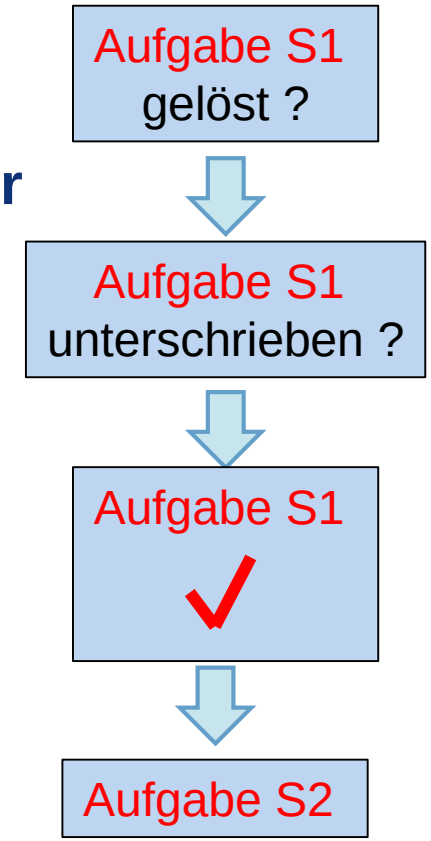
Legostadt

Aufgabe S1:
Fahrt zum Parkplatz Berghütte – Automatisches Anhalten mittels Tastsensor

Start: Parkplatz Bahnhof

Ende: Parkplatz Berghütte

Der Roboter soll vom Bahnhof zur Berghütte fahren. Mittels Tastsensor soll er selbständig anhalten, sobald der Tastsensor die Mauer vor dem Parkplatz berührt.



DAS SPIELFELD: Legostadt

Aufgabe S2:
Fahrt zum Meer – Automatisches Anhalten mittels Ultraschallsensor

Start: Parkplatz Schule

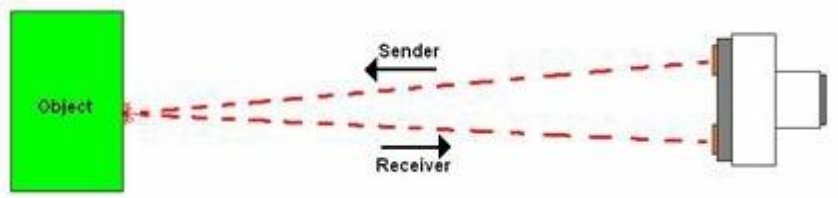
Ende: Parkplätze Leuchtturm / Strandhütte

Der Roboter soll von der Schule zum Meer fahren. Mittels Ultraschallsensor soll er selbständig anhalten, sobald ein geeigneter Abstand von den Mauern um die Parkplätze am Meer (Leuchtturm, Strandhütte) erreicht ist.

DAS LEGO® MINDSTORMS® System

Ultraschallsensor

- Sensor sendet Ultraschall aus
- Schall wird von Hindernis reflektiert
- Reflektierter Schall wird vom Empfänger registriert
- Aus Laufzeit des Schalls kann auf die Entfernung geschlussfolgert werden
- Messwerte werden in cm ausgegeben



Abfrage Ultraschallsensors

getUSDistance(Sensor)

Beispiel für Nutzung des Ultraschallsensors

```
#pragma config(Sensor, S2,     ,               sensorEV3_Ultrasonic)
#pragma config(Motor,  motorB,          MotorRechts,    tmotorEV3_Large, PIDControl, driveRight, encoder)
#pragma config(Motor,  motorC,          MotorLinks,     tmotorEV3_Large, PIDControl, driveLeft, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  // vorwaerts fahren
  setMotorSpeed(motorB,70);
  setMotorSpeed(motorC,70);

  while(getUSDistance(S2)>10)
  {
  }
  setMotorSpeed(motorB,0);
  setMotorSpeed(motorC,0);
}
```

Der Motor fährt vorwärts bis ein Hindernis in ca. 10 cm Entfernung detektiert wird.

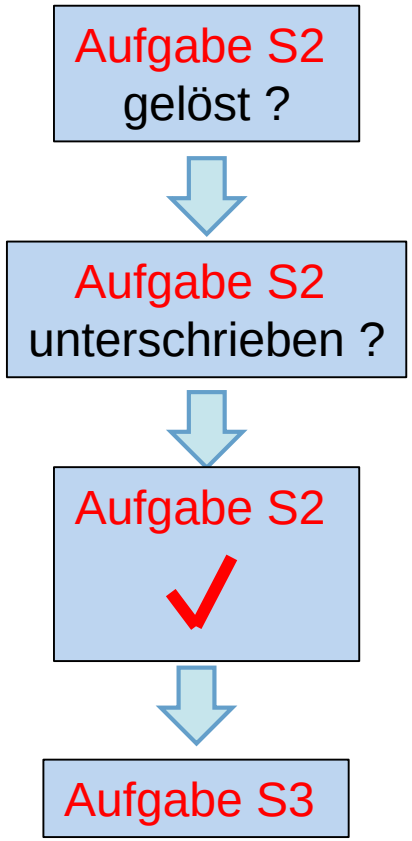
DAS SPIELFELD: Legostadt

Aufgabe S2:
Fahrt zum Meer – Automatisches Anhalten mittels Ultraschallsensor

Start: Parkplatz Schule

Ende: Parkplätze Leuchtturm / Strandhütte

Der Roboter soll von der Schule zum Meer fahren. Mittels Ultraschallsensor soll er selbständig anhalten, sobald ein geeigneter Abstand von den Mauern um die Parkplätze am Meer (Leuchtturm, Strandhütte) erreicht ist.



DAS SPIELFELD: Legostadt

Aufgabe S3:
Einkaufen – Halten am richtigen Geschäft mittels Farbsensor

Start: Parkplatz Bahnhof

Ende: Geschäft

Der Roboter soll vom Bahnhof in die Einkaufsstraße fahren und beim Cafe (Pfeiltasten oben), der Post (Pfeiltasten links) oder der Zoofachgeschäft (Pfeiltasten rechts) halten. Die Parkflächen sind verschieden farbig markiert. Ermöglichen Sie ein mehrfaches Auswählen!

DAS LEGO® MINDSTORMS® System

Colorsensor

- Verfügt über mehrere Moden, z. B.
  - Bestimmung des Farbwertes (ColorID)
  - Bestimmung der reflektierten Helligkeit
- Zur Ausleuchtung kann eine LED eingeschaltet werden

DAS LEGO® MINDSTORMS® System

Colorsensor – ColorID Mode

- Bestimmung der Farbe
- Jede Farbe hat einen Wert
- Werte für EV3 Colorsensor

	Color name	Enumerated Value
colorNone:	No object is detected by the color sensor	0
colorBlack:	A black object is detected by the color sensor	1
colorBlue:	A blue object is detected by the color sensor	2
colorGreen:	A green object is detected by the color sensor	3
colorYellow:	A yellow object is detected by the color sensor	4
colorRed:	A red object is detected by the color sensor	5
colorWhite:	A white object is detected by the color sensor	6
colorBrown:	A brown object is detected by the color sensor	7

C Code

Colorsensor – ColorID Mode

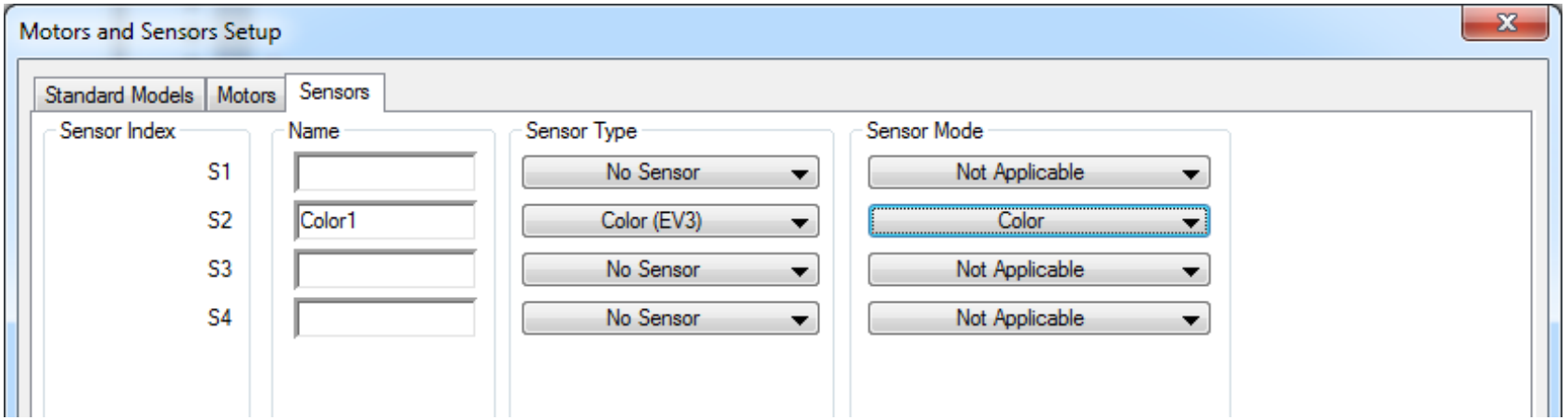
getColorName

	TLegoColors getColorName(tSensors nDeviceIndex)	
Parameter	**Explanation**	**Data Type**
Return Type	Returns the current color value of the color sensor attached to nDeviceIndex.	TLegoColors
nDeviceIndex	A sensor port or name.	tSensors

- One of 8 different color names will be returned, depending on the color of the object detected.
- If no color is detected, colorNone will be returned instead.

Beispiel: Anzeige der ColorID
Colorsensor – ColorID Mode

Sensor Setup



Beispiel: Anzeige der ColorID
Colorsensor – ColorID Mode

```
1  #pragma config(Sensor, S2,     Color1,         sensorEV3_Color, modeEV3Color_Color)
2  //*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
3
4  task main()
5  {
6    string farbe;
7    while(true)
8    {
9
10     if(getColorName(S2)==colorBlack)
11     {
12       farbe = "Black";
13     }
14     else
15     {
16       farbe = "Nicht hinterlegt";
17     }
18     displayTextLine(1,farbe);
19     displayTextLine(2,"Druecke eine Taste");
20     waitForButtonPress();
21   }
22
23
24
25 }
```

Die ColorID wird abgefragt und sofern schwarz erkannt wird, dies angezeigt.

DAS SPIELFELD: Legostadt

Aufgabe S3:
Einkaufen – Halten am richtigen Geschäft mittels Farbsensor

Start: Parkplatz Bahnhof

Ende: Geschäft

Der Roboter soll vom Bahnhof in die Einkaufsstraße fahren und beim Cafe (Pfeiltasten oben), der Post (Pfeiltasten links) oder der Zoofachgeschäft (Pfeiltasten rechts) halten. Die Parkflächen sind verschieden farbig markiert. Ermöglichen Sie ein mehrfaches Auswählen!