

Fakultät für Informatik, Institut für Robotik

Laborpraktikum I

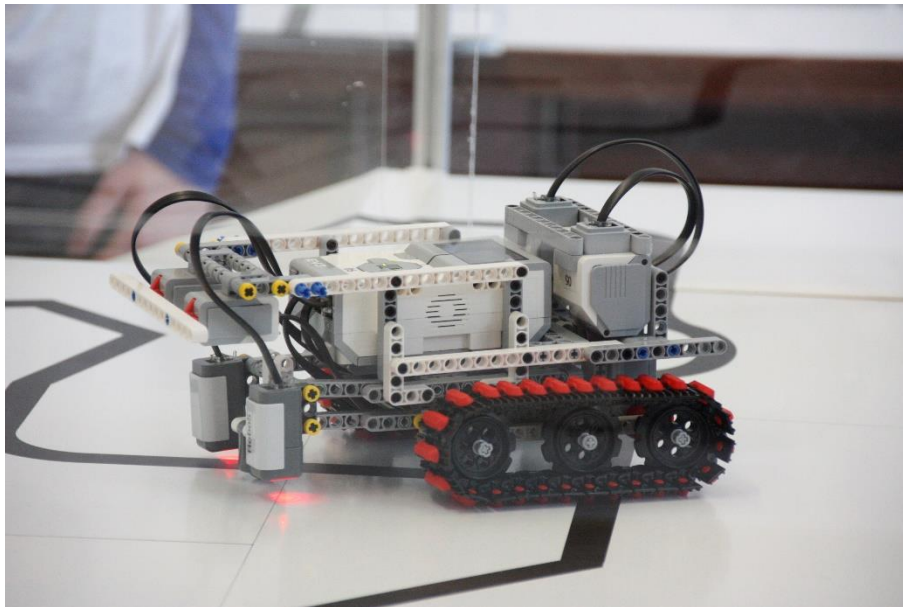
Legorobotik in C – EV3

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

Motoren



Quelle: Lego

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

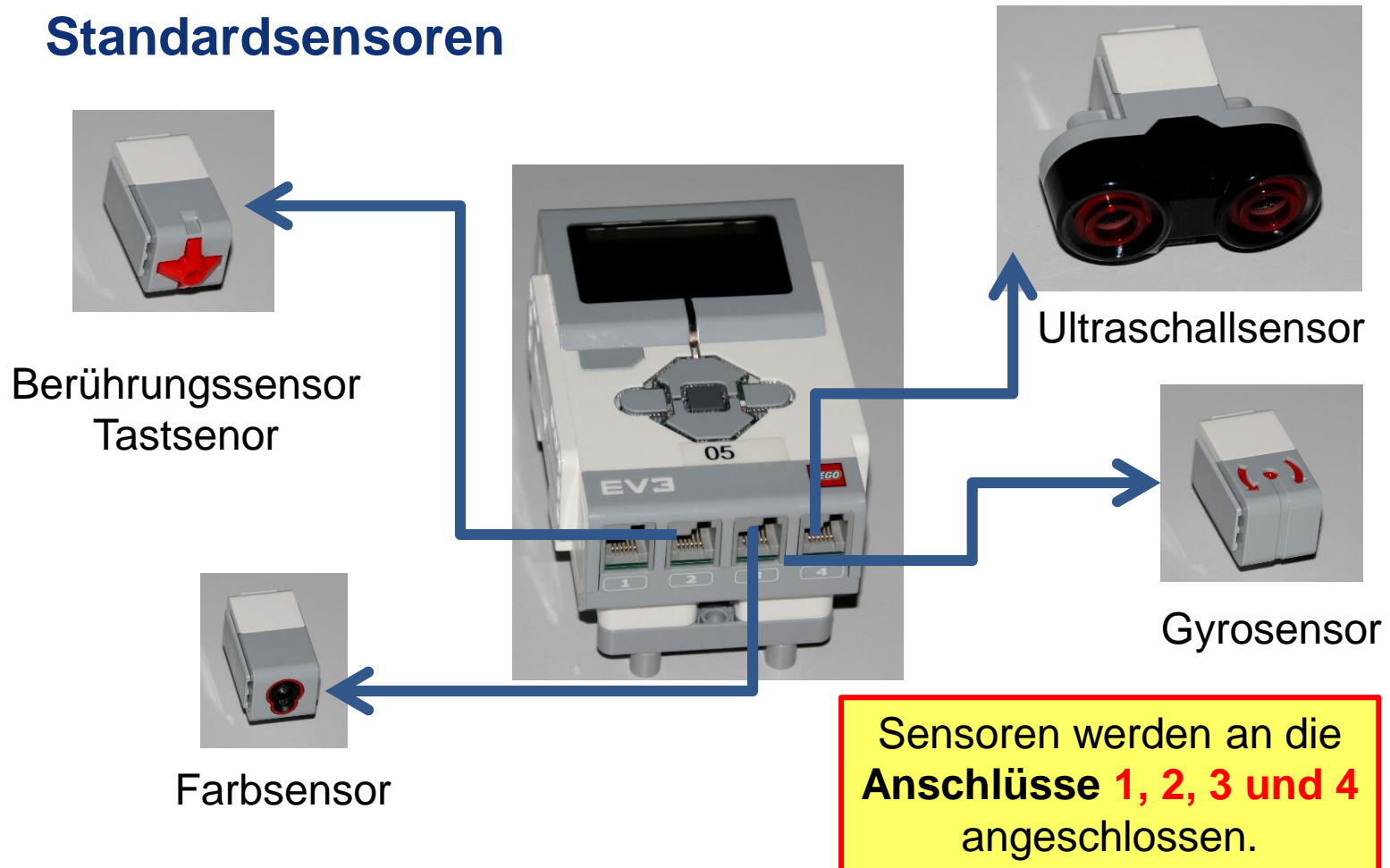
Servomotor

- 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

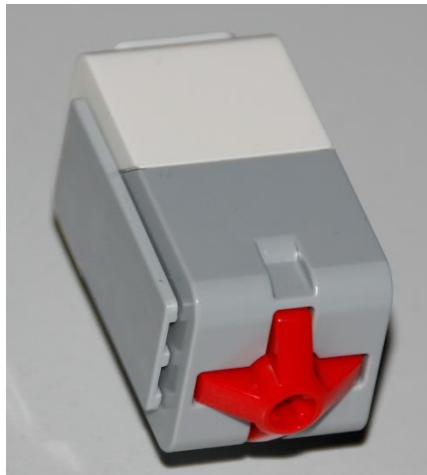
Standardsensoren





DAS LEGO® MINDSTORMS® System

Berührungssensor / Tastsensor



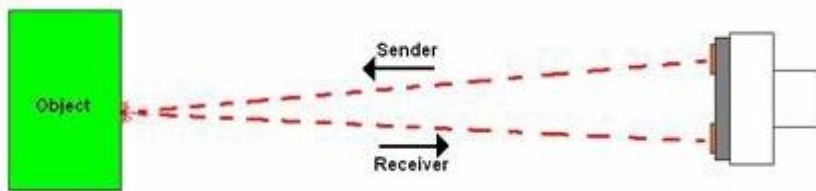
- Abfrage, ob Sensor gedrückt
- Werte des Sensors
 - 0: Sensor nicht gedrückt
 - 1: Sensor gedrückt

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
- Messbereich: 3 bis 250 cm
- Messgenauigkeit: +/- 1 cm



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

Wert	Farbe
-1	keine
0	Rot
1	Grün
2	Blau
3	Gelb
4	Magenta
5	Orange
6	Weiß
7	Schwarz
8	Pink
9	Grau
10	Hellgrau
11	Dunkelgrau
12	Zyan
13	Braun





DAS LEGO® MINDSTORMS® System

Colorsensor – ambient Light Mode



- Messung der Helligkeit mittels Fotodiode
- Helle Fläche reflektiert mehr Licht als dunkle
- Messbereich:
 - 0: dunkel
 - 100: hell
- Zur Ausleuchtung kann eine LED eingeschaltet werden



DAS LEGO® MINDSTORMS® System

Gyrosensor



- Messung der Drehbewegung und der Richtungsänderung
- Messbereich bis 440 °/s
- Messgenauigkeit; 1kHz
- Erfassungsrate: 1kHz

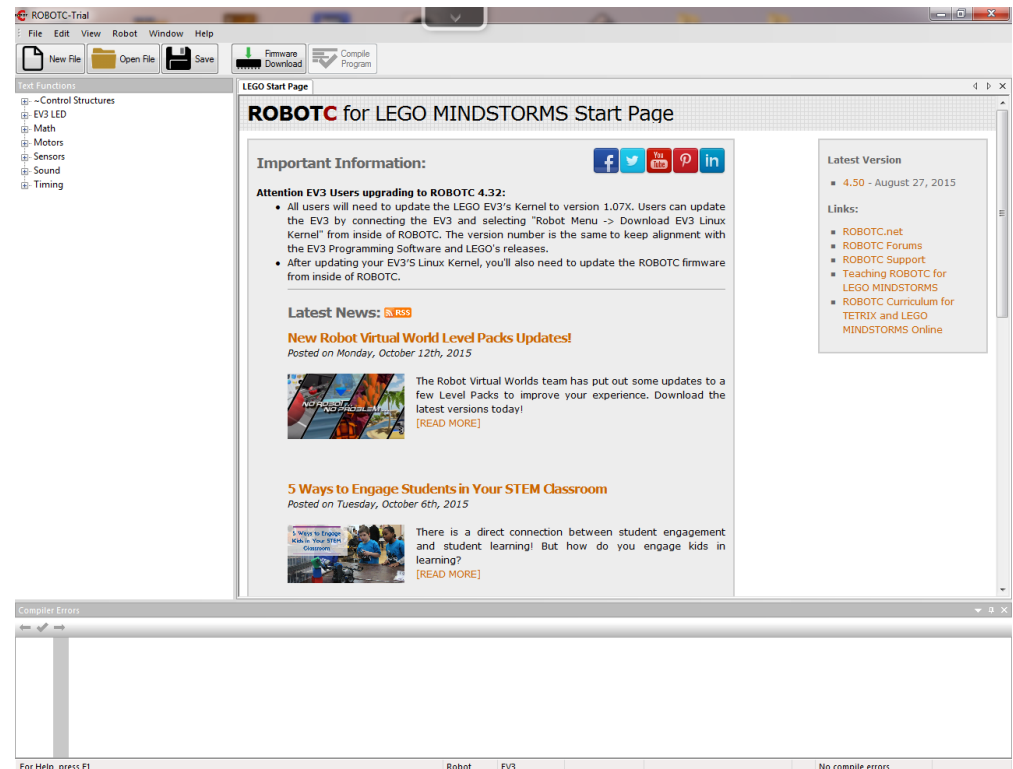
Start der Entwicklungsumgebung

Starten von RobotC

Startsymbol:



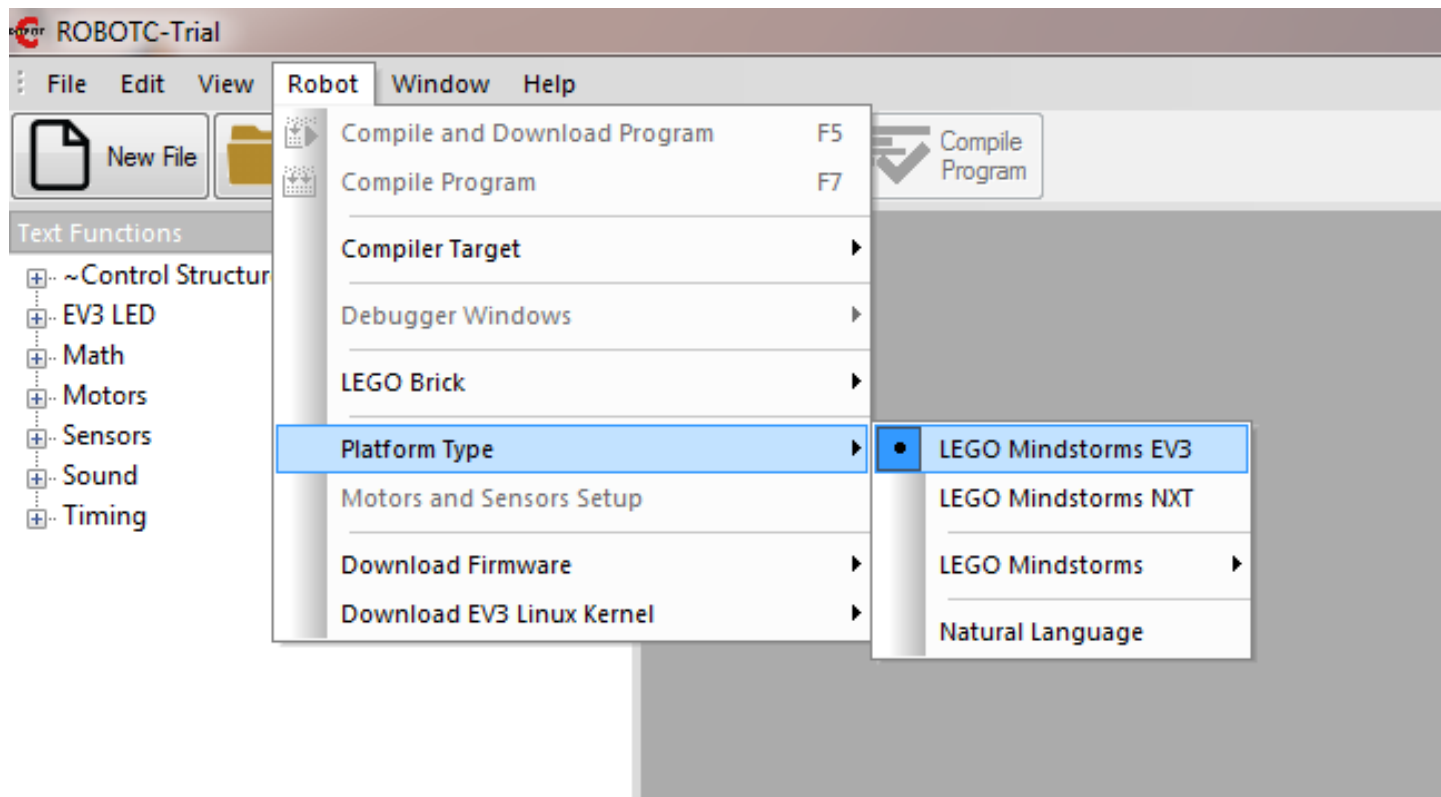
ROBOTC for LEGO Mindstorms
4.X





Start der Entwicklungsumgebung

Starten von RobotC

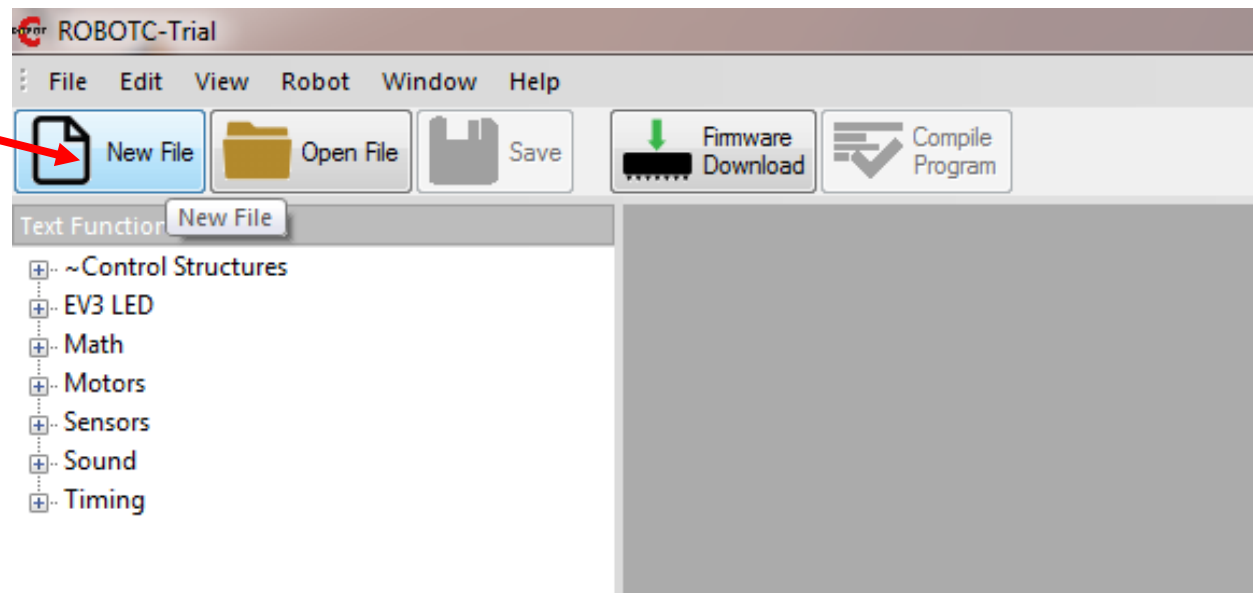




Start der Entwicklungsumgebung

Starten von RobotC

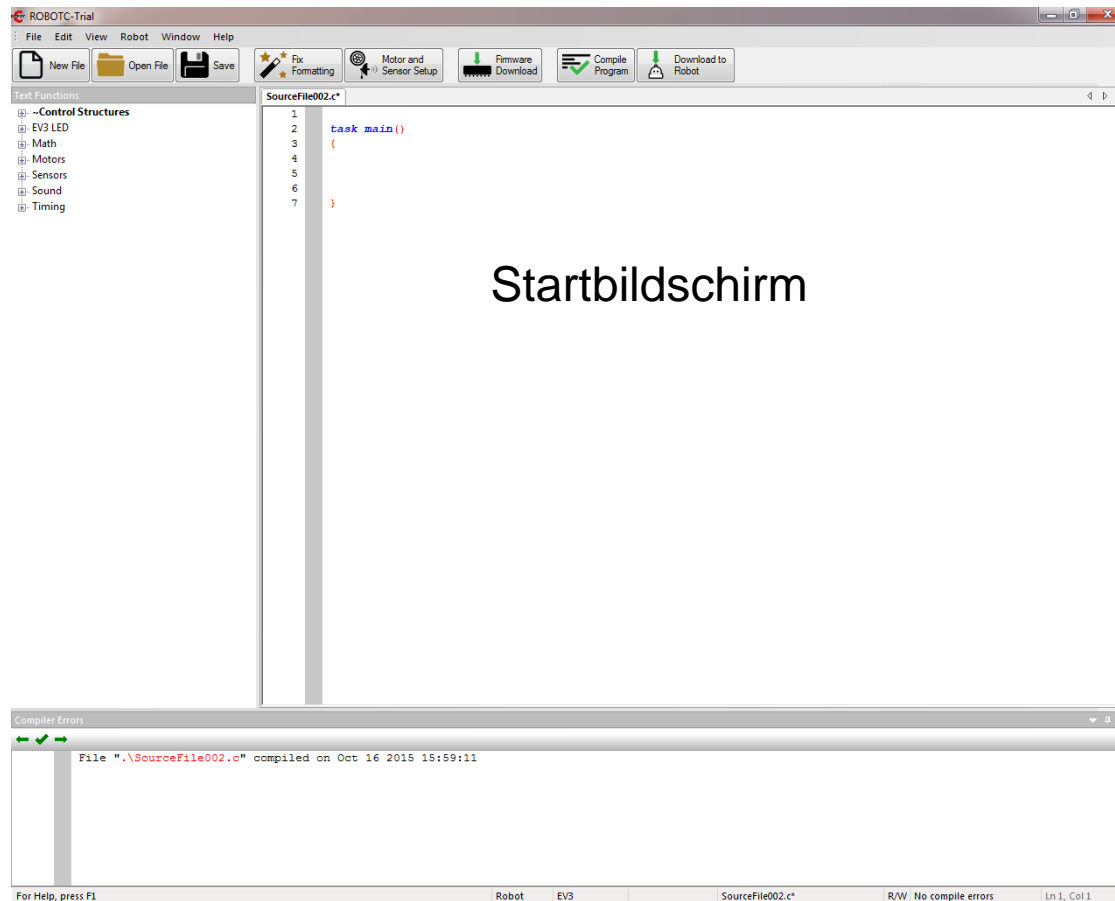
New File
wählen





Start der Entwicklungsumgebung

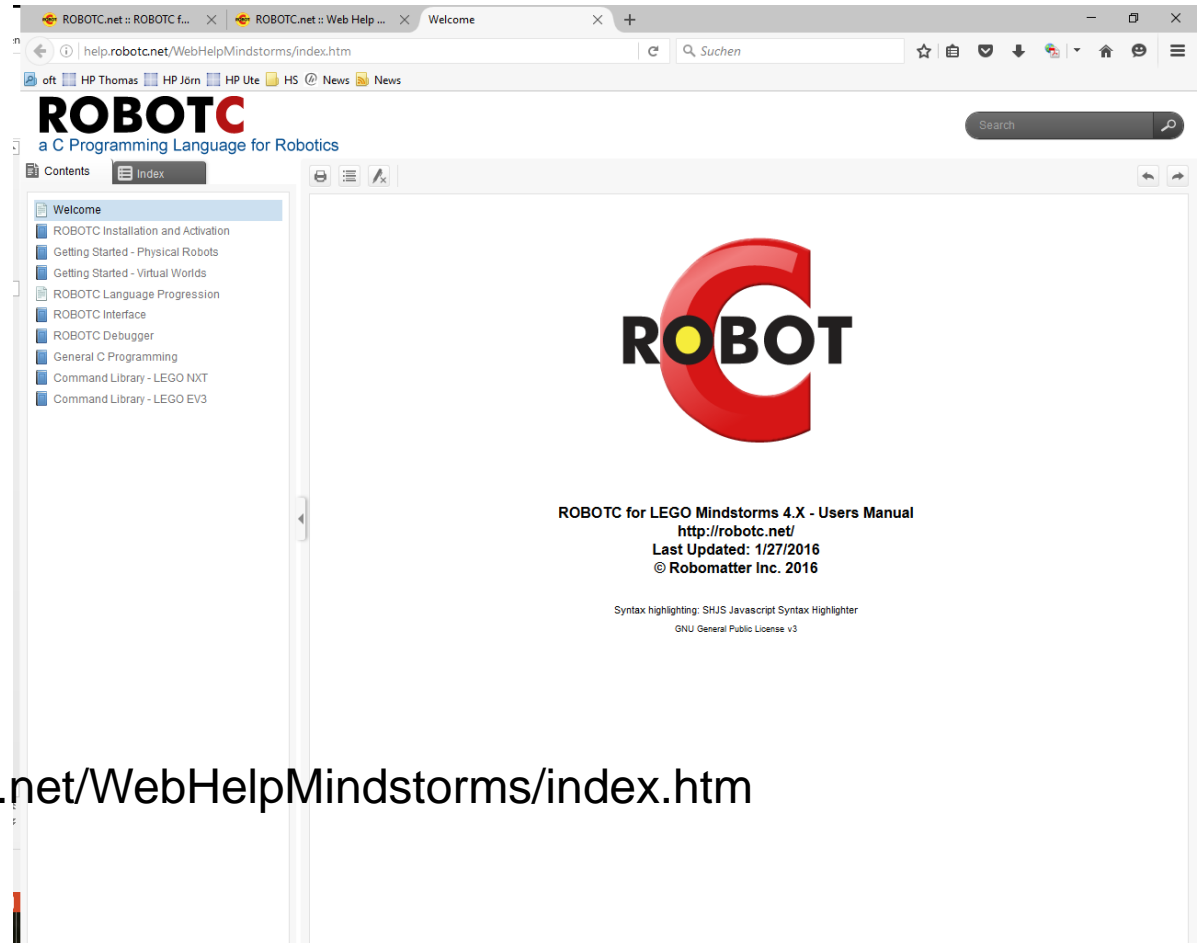
Starten von RobotC





Start der Entwicklungsumgebung

RobotC Tutorial



<http://help.robotc.net/WebHelpMindstorms/index.htm>



Arbeit mit RobotC

Bildschirmanzeige

1. Anzeige kleine Textzeile

`displayTextLine(zeile, text);`

Parameter	Erklärung	Datentyp
zeile	Zeile in der der Text stehen soll	Integer
text	Anzuzeigender Text	string

Beispiel:

`displayTextLine(3, "kleine Textzeile");`



Arbeit mit RobotC

Bildschirmanzeige

2. Anzeige große Textzeile

`displayBigTextLine`(zeile, text);

Parameter	Erklärung	Datentyp
zeile	Zeile in der der Text stehen soll	Integer
text	Anzuzeigender Text	string

Beispiel:

`displayBigTextLine`(3, "kleine Textzeile");

3. Display löschen

`eraseDisplay`();



RobotC EV3

Pausenbefehle

1. Sleep - Befehl

```
sleep(2000);
```

Parameter	Erklärung	Datentyp
zeit	Pausenzeit in ms	Integer

2. Warten auf Knopfdruck

```
waitForButtonPress();
```

Sobald ein beliebiger Knopf auf dem EV3 Stein gedrückt wird, wird das laufende Programm weiter ausgeführt.



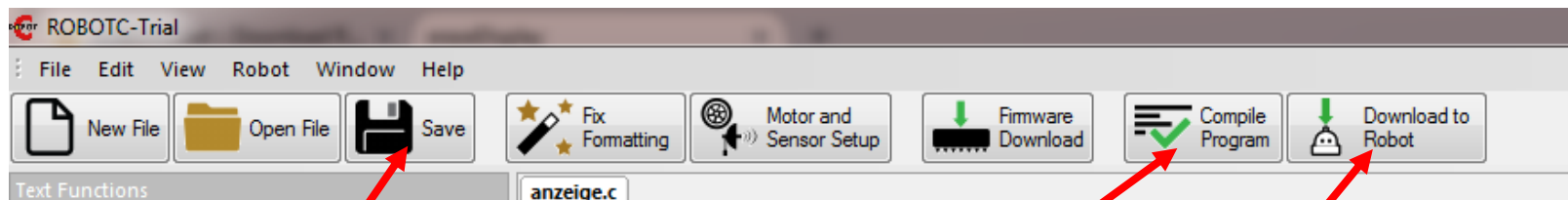
RobotC EV3

Beispielprogramm zur Bildschirmanzeige und Pausenbefehle

```
LEGO Start Page | Anzeige.c | Message Log
1  task main()
2  {
3      // Variablendeklaration
4      float zahl;
5      string szahl;
6
7      //Anzeige gross
8      displayBigTextLine(1,"Grosse Textzeile");
9      //Anzeige klein
10     displayTextLine(3, "kleine Textzeile");
11     //Pause;
12     sleep(2000);
13
14     //Bildschirmloeschen
15     eraseDisplay();
16
17     //Anzeige einer Zahl
18     zahl =3;
19     //Umrechnung zahl in String
20     szahl=sprintf(szahl, "%.1f", zahl);
21     displayBigTextLine(4,szahl);
22
23     //Warten auf Knofdruck
24     waitForButtonPress();
25 }
```

RobotC EV3

Programme speichern, kompilieren und übertragen



1. Programm Sichern

2. Programm kompilieren

3. Programm auf Roboter laden

Danach das Programm auf dem Roboter starten!



DAS SPIELFELD: Legostadt

Aufgabe 1: Fahrt zum Flughafen

Start: P1

Ende: Flughafenhalle

Der Roboter soll aus P1 zum
Parkfläche am Flughafen fahren.

Ziel:

Lernen der Steuerung des Roboters.

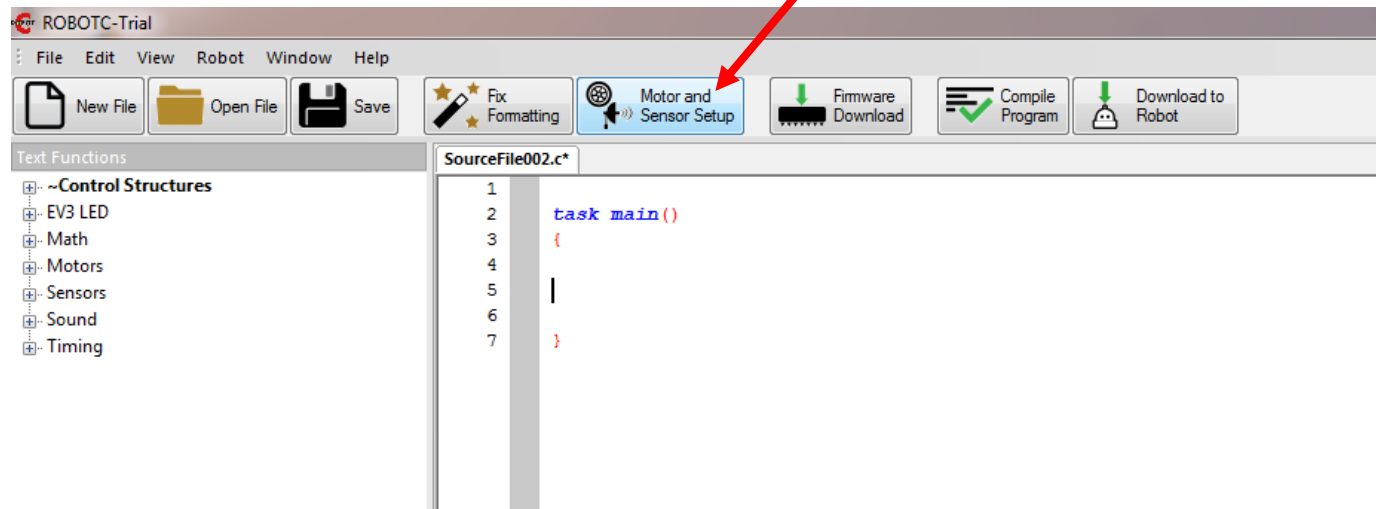
- Geradeausfahren
- Kurvenfahren



DAS SPIELFELD: Legostadt

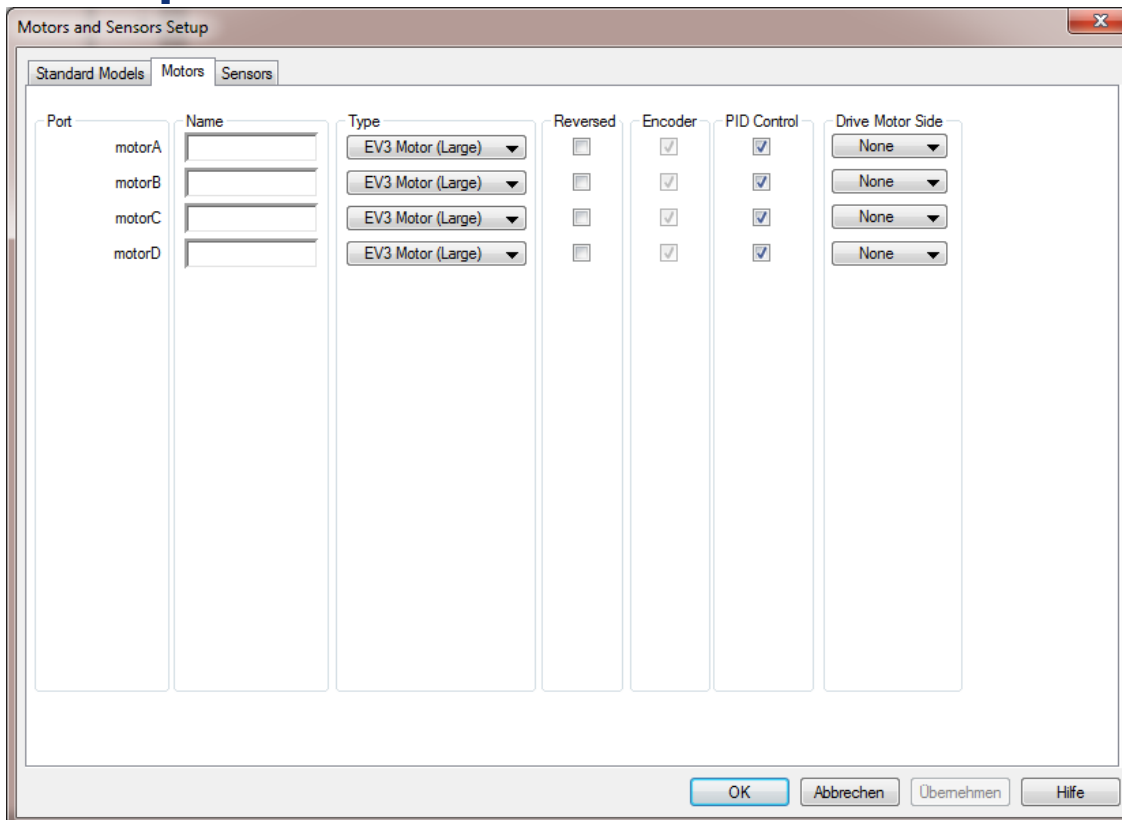
Setup Motoren

Motor und
Sensor Setup
wählen



DAS SPIELFELD: Legostadt

Setup Motoren



Port	Name	Type	Reversed	Encoder	PID Control	Drive Motor Side
motorA		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None
motorB		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None
motorC		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None
motorD		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None

Einstellungen
für die Motoren
entsprechend der
Roboterkonfiguration
vornehmen

Siehe nächste Folie



DAS SPIELFELD: Legostadt

Setup Motoren

Die großen Motoren dienen zum Fahren. Rechter und Linker Motor sollten korrekt angegeben werden.

Damit kann man einfache Steuerkommandos verwenden.

Port	Name	Type	Reversed	Encoder	PID Control	Drive Motor Side
motorA		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None
motorB	MotorLinks	EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Left
motorC	MotorRechts	EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Right
motorD		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None

1. Übernehmen klicken

2. auf OK klicken

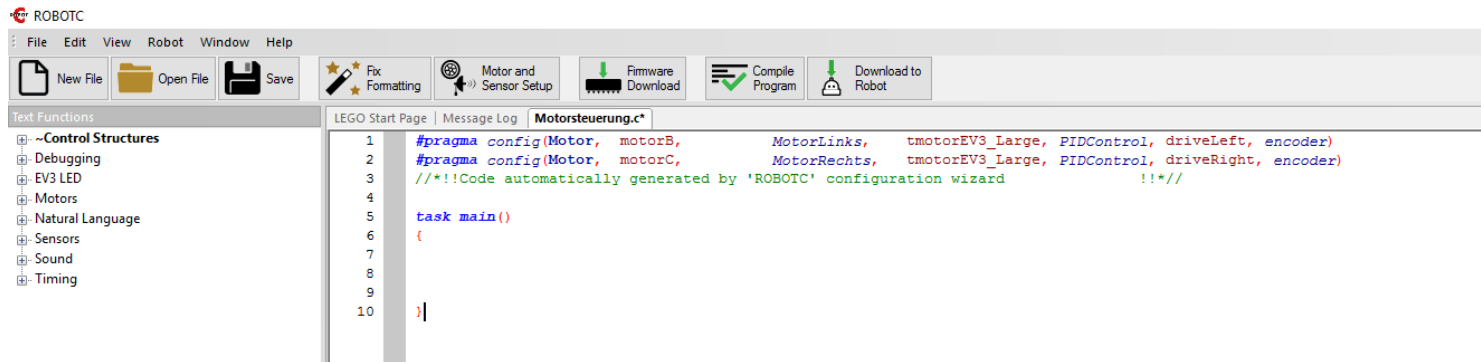
OK Abbrechen Übernehmen Hilfe



DAS SPIELFELD: Legostadt

Setup Motoren

Anzeige der Motoren im Programm





DAS SPIELFELD: Legostadt

Setup Motoren

Port	Name	Type	Reversed	Encoder	PID Control	Drive Motor Side
motorA		EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None
motorB	MotorLinks	EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Left
motorC	MotorRechts	EV3 Motor (Large)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Right
motorD	MotorGreifer	EV3 Motor (Medium)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	None

Buttons at the bottom: OK, Abbrechen, Übernehmen, Hilfe

Bei Bedarf können
weitere Motoren
meist mittlere Motoren
genutzt werden.

Motoren, die nicht zum Fahren
benutzt werden, sondern für
Greifarme usw. erhalten die
Drive Motor Side Einstellung:
None



DAS SPIELFELD: Legostadt

Motorensteuerung

setMotorSpeed(nMotorIndex, nSpeed)

Parameter	Erklärung	Datentyp
nMotorIndex	Bezeichnung Motor	Name
nSpeed	Leistung / Geschwindigkeit	integer



DAS SPIELFELD: Legostadt

Motorensteuerung (Zeit)

```
// vorwaerts fahren
```

```
setMotorSpeed(motorA, 70);
```

```
setMotorSpeed(motorB, 70);
```

```
sleep(1000);
```

```
// rueckwaerts fahren
```

```
setMotorSpeed(motorA, -70);
```

```
setMotorSpeed(motorB, -70);
```

```
sleep(1000);
```

```
//Linkskurve
```

```
setMotorSpeed(motorA, -70);
```

```
setMotorSpeed(motorB, 70);
```

```
sleep(1000);
```

```
//Rechtskurve
```

```
setMotorSpeed(motorA, 70);
```

```
setMotorSpeed(motorB, -70);
```

```
sleep(1000);
```

```
//Anhalten
```

```
setMotorSpeed(motorA, 0);
```

```
setMotorSpeed(motorB, 0);
```



DAS SPIELFELD: Legostadt

Beispiel: Motorensteuerung (Zeit)

Der Roboter fährt

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

```
4
5  task main()
6  {
7      // vorwaerts fahren
8      setMotorSpeed(motorA, 70);
9      setMotorSpeed(motorB, 70);
10     sleep(1000);
11
12     // rueckwaerts fahren
13     setMotorSpeed(motorA, -70);
14     setMotorSpeed(motorB, -70);
15     sleep(1000);
16
17     //Linkskurve
18     setMotorSpeed(motorA, -70);
19     setMotorSpeed(motorB, 70);
20     sleep(1000);
21
22     //Rechtskurve
23     setMotorSpeed(motorA, 70);
24     setMotorSpeed(motorB, -70);
25     sleep(1000);
26
27     //Anhalten
28     setMotorSpeed(motorA, 0);
29     setMotorSpeed(motorB, 0);
30
31 }
```



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Zeit)

setMotorSyncTime(nMotorOne, nMotorTwo, nTurnRatio, nTimeMsec, nSignedPower)

Parameter	Erklärung	Datentyp
nMotorOne	Bezeichnung erster Motor	Name
nMotorTwo	Bezeichnung zweiter Motor	Name
nTurnRatio	Drehverhältnis	long/integer
nTimeMsec	Zeit in ms	long/integer
nSigned Power	Power	long/integer



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Zeit)

nTurn Ratio	Erklärung	Reaktion
100	100% der Leistung an MotorOne und -100% der Leistung an MotorTwo	
-100	-100% der Leistung an MotorOne und 100% der Leistung an MotorTwo	
0	Gleiche Leistung auf beide Motoren	
50	100% Leistung auf MotorOne 0% Leistung auf MotorTwo	
-50	0% Leistung auf MotorOne 100% Leistung auf MotorTwo	



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Zeit)

```
4
5 task main()
6 {
7     // Geradeausfahren
8     setMotorSyncTime (motorA,motorB,0,1000,70);
9     sleep(1000);
10
11    //Rechtskurve beide Motren gegenlaeufig
12    setMotorSyncTime (motorA,motorB,100,1000,70);
13    sleep(1000);
14
15    //Linkskurve beide Motren gegenlaeufig
16    setMotorSyncTime (motorA,motorB,-100,1000,70);
17    sleep(1000);
18
19    //Rechtskurve mit einem Motor
20    setMotorSyncTime (motorA,motorB,50,1000,70);
21    sleep(1000);
22
23    // Linkskurve mit einem Motor
24    setMotorSyncTime (motorA,motorB,-50,1000,70);
25    sleep(1000);
26 }
```

Der Roboter fährt

- Geradeaus
- Dreht sich rechts mit beiden Motoren
- Dreht sich links mit beiden Motoren
- Dreht sich rechts mit einem Motor
- Dreht sich links mit einem Motor

Nach dem Befehl `setMotorSyncTime` muss ein `sleep`-Befehl folgen. Dieser muss mindestens so groß sein, wie die Zeitangabe im Befehl `setMotorSyncTime`



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Drehwinkel)

setMotorSyncEncoder(nMotorOne, nMotorTwo, nTurnRatio, nEncoderCount, nSignedPower)

Parameter	Erklärung	Datentyp
nMotorOne	Bezeichnung erster Motor	Name
nMotorTwo	Bezeichnung zweiter Motor	Name
nTurnRatio	Drehverhältnis	long/integer
nEncoderCount	Drehwinkel	long/integer
nSigned Power	Power	long/integer



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Drehwinkel)

nTurn Ratio	Erklärung	Reaktion
100	100% der Leistung an MotorOne und -100% der Leistung an MotorTwo	
-100	-100% der Leistung an MotorOne und 100% der Leistung an MotorTwo	
0	Gleiche Leistung auf beide Motoren	
50	100% Leistung auf MotorOne 0% Leistung auf MotorTwo	
-50	0% Leistung auf MotorOne 100% Leistung auf MotorTwo	



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Drehwinkel)

```
4
5 task main()
6 {
7     // Geradeausfahren
8     setMotorSyncEncoder(motorA,motorB,0,360,70);
9     waitUntilMotorStop(motorA);
10
11    //Rechtskurve beide Motren gegenlaeufig
12    setMotorSyncEncoder(motorA,motorB,100,360,70);
13    waitUntilMotorStop(motorA);
14
15    //Linkskurve beide Motren gegenlaeufig
16    setMotorSyncEncoder(motorA,motorB,-100,360,70);
17    waitUntilMotorStop(motorA);
18
19    //Rechtskurve mit einem Motor
20    setMotorSyncEncoder(motorA,motorB,50,360,70);
21    waitUntilMotorStop(motorA);
22
23    // Linkskurve mit einem Motor
24    setMotorSyncEncoder(motorA,motorB,-50,360,70);
25    waitUntilMotorStop(motorA);
26 }
```

Der Roboter fährt

- Geradeaus
- Dreht sich rechts mit beiden Motoren
- Dreht sich links mit beiden Motoren
- Dreht sich rechts mit einem Motor
- Dreht sich links mit einem Motor

Nach dem Befehl `setMotorSyncEncoder`, anschließend Befehl `waitUntilMotorStop` verwenden.



DAS SPIELFELD: Legostadt

Synchrone Steuerung zweier Motoren (Drehwinkel)

`waitUntilMotorStop()`

Parameter	Erklärung	Datentyp
nMotorIndex	Name / Bezeichnung des Motors	char

Dieser Befehl ist ein Wartebefehl.

Es wird so lange gewartet, bis der entsprechende Motor stoppt.



DAS SPIELFELD: Legostadt

Weitere Befehle zur Motorsteuerung (Simple Behaviors)

Hat man die Motoren für das Fahren festgelegt und den rechten und linken Motor richtig zugeordnet, so gibt es einfache Befehle zum Fahren des Roboters.



DAS SPIELFELD: Legostadt

Weitere Befehle zur Motorsteuerung (Simple Behaviors)

1. Vorwärtsfahren

//Vorwärtsfahren fuer z Sekunden mit v Geschwindigkeit

`forward(z, seconds, v);`

//Vorwärtsfahren fuer z Umdrehungen mit v Geschwindigkeit

`forward(z, rotations, v);`

//Vorwärtsfahren fuer z Grad mit v Geschwindigkeit

`forward(z, degrees, v);`



DAS SPIELFELD: Legostadt

Weitere Befehle zur Motorsteuerung (Simple Behaviors)

2. Rückwärtsfahren

//Rueckwaertsfahren fuer z Sekunden mit v Geschwindigkeit

`backward(z, seconds, v);`

//Rueckwaertsfahren fuer z Umdrehungen mit v Geschwindigkeit

`backard(z, rotations, v);`

//Rueckwaertsfahren fuer z Grad mit v Geschwindigkeit

`backward(z, degrees, v);`



DAS SPIELFELD: Legostadt

Weitere Befehle zur Motorsteuerung (Simple Behaviors)

3. Linksfahren

//Linksfahren fuer z Sekunden mit v Geschwindigkeit
`turnLeft(z, seconds, v);`

//Linksfahren fuer z Umdrehungen mit v Geschwindigkeit
`turnLeft(z, rotations, v);`

//Linksfahren fuer z Grad mit v Geschwindigkeit
`turnLeft(z, degrees, v);`

Drehung in Grad bedeutet,
das sich der Motor um z Grad dreht,
nicht aber der Roboter selbst.
Gleiches gilt für Umdrehungen.



DAS SPIELFELD: Legostadt

Weitere Befehle zur Motorsteuerung (Simple Behaviors)

4. Rechtsfahren

//Rechtsfahren fuer z Sekunden mit v Geschwindigkeit
`turnRight(z, seconds, v);`

//Rechtsfahren fuer z Umdrehungen mit v Geschwindigkeit
`turnRight(z, rotations, v);`

//Rechtsfahren fuer z Grad mit v Geschwindigkeit
`turnRight(z, degrees, v);`

Drehung in Grad bedeutet,
das sich der Motor um z Grad dreht,
nicht aber der Roboter selbst.
Gleiches gilt für Umdrehungen.



DAS SPIELFELD: Legostadt

Weitere Befehle zur Motorsteuerung (Simple Behaviors)

5. Beispiel

```
GO Start Page | Message Log | Motorsteuerung.c
1  #pragma config(Motor,  motorB,                MotorLinks,   tmotorEV3_Large, PIDControl, driveLeft,  encoder)
2  #pragma config(Motor,  motorC,                MotorRechts,  tmotorEV3_Large, PIDControl, driveRight, encoder)
3  /**!!Code automatically generated by 'ROBOTC' configuration wizard      !!*/
4
5  task main()
6  {
7      //Vorwaertsfahren fuer 1s mit einer Geschwindigkeit von 50
8      forward(1,seconds,50);
9
10     // Rechtsfahren fuer 90 Grad und Geschwindigkeit von 75
11     turnRight(90,degrees,75);
12
13     //Rueckwaertsfahren fuer 1 Umdrehung mit Geschwindigkeit von 100
14     backward(1,rotations,100);
15
16     //Linksfahren fuer 1 Umdrehung mit Geschwindigkeit von 50
17     turnLeft(1,rotations,50);
18
19 }
```



DAS SPIELFELD: Legostadt

Aufgabe 1: Fahrt zum Flughafen

Start: P1

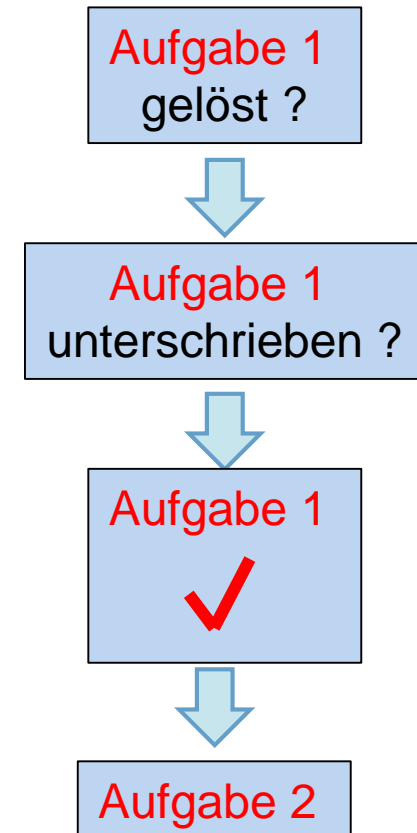
Ende: Flughafenhalle

Der Roboter soll aus P1 zum
Parkfläche am Flughafen fahren.

Ziel:

Lernen der Steuerung des Roboters.

- Geradeausfahren
- Kurvenfahren





DAS SPIELFELD: Legostadt

Aufgabe 2: Fahrt zum Krankenhaus auf verschiedenen Wegen

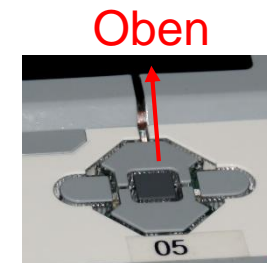
Start: P2

Ende: Parkfläche Krankenhaus

Der Roboter soll von P2 aus über 2 verschiedene Weg zum Krankenhaus fahren. Die Auswahl des Weges ist abhängig vom gedrückten Knopf des EV3 Steines.

Knopf Oben: über Cafe

alle anderen: über Hotel





DAS SPIELFELD: Legostadt

Die if-else Abfrage

```
if(<<Ausdruck>>)  
{  
    <<Anweisung>>  
    ...  
    << Anweisung>>  
}  
Else  
{  
    << Anweisung>>  
    ...  
    << Anweisung>>  
}
```

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

Beispiel: `if(<<a==10>>)`

```
{  
    <<Anweisung>>  
    ...  
    << Anweisung>>  
}  
Else  
{  
    << Anweisung>>  
    ...  
    << Anweisung>>  
}
```



DAS SPIELFELD: Legostadt

Arithmetische 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



DAS SPIELFELD: Legostadt

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



DAS SPIELFELD: Legostadt

Einige Logische Operatoren

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



DAS SPIELFELD: Legostadt

Abfrage von EV3 Buttons

Warten auf Knopfdruck:

```
waitForButtonPress() ;
```

Abfrage, ob Knopf oben
gedrückt ist:

```
getButtonPress(button)
```

Button	Name (button)	ID
Kein Button	buttonNone	0
Oben Button	buttonUp	1
Enter Button	buttonEnter	2
Unten Button	buttonDown	3
Rechter Button	buttonRight	4
Linker Button	buttonLeft	5
Zurück Button	buttonBack	6
Jeder Button	buttonAny	7



DAS SPIELFELD: Legostadt

Abfrage von EV3 Buttons

```
1
2  task main()
3  {
4      //Warten darauf, dass ein Knopf gedrueckt wird
5      eraseDisplay();
6      displayBigTextLine(1,"Druecke eine Taste");
7      waitForButtonPress();
8
9      //Abfrage, ob Oberer Knopf gedrueckt ist#
10     if(getButtonPress(buttonUp)==1)
11     {
12         displayBigTextLine(3,"Linker Weg");
13
14     }
15     else
16     {
17         displayBigTextLine(3,"Rechter Weg");
18     }
19
20     sleep(2000);
21
22 }
```



DAS SPIELFELD: Legostadt

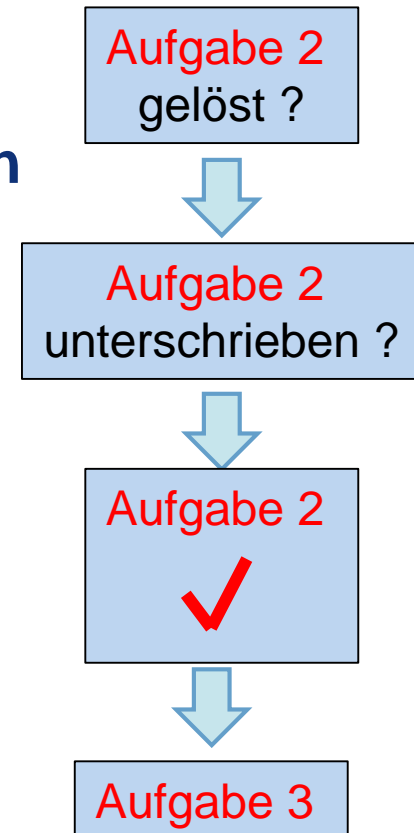
Aufgabe 2: Fahrt zum Krankenhaus auf verschiedenen Wegen

Start: P2

Ende: Parkfläche Krankenhaus

Der Roboter soll von P2 aus über 2
verschiedene Weg zum Krankenhaus fahren.
Die Auswahl des Weges ist abhängig vom
gedrückten Knopf des
EV3 Steines.

Knopf Oben: über Cafe
alle anderen: über Hotel





DAS SPIELFELD: Legostadt

Aufgabe 3: Beförderung von Fahrgästen zwischen Flughafen und Hotel

Start und Ende: Parkfläche Flughafen

Der Roboter soll als Shuttlebus Gäste zwischen Flughafen und Hotel hin und zurück befördern. An jedem Ort warten 3 Gäste. Es soll jeweils ein Gast transportiert werden.

Der Roboter wartet eine gewisse Zeit, bis der Gast eingestiegen ist. Der Roboter fährt die Strecke vom Flughafen zum Hotel vorwärts. Lässt den Gast ein- und aussteigen und fährt nach entsprechender Wartezeit die gleiche Strecke rückwärts zurück.

Auf den Parkflächen darf der Roboter neu ausgerichtet werden!



DAS SPIELFELD: Legostadt

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>>  
}
```

DAS SPIELFELD: Legostadt

Beispielprogramm: for Schleife

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

Das Programm gibt
das Wort Test
4mal aus.



DAS SPIELFELD: Legostadt

Aufgabe 3: Beförderung von Fahrgästen zwischen Flughafen und Hotel

Start und Ende: Parkfläche Flughafen

Der Roboter soll als Shuttlebus Gäste zwischen Flughafen und Hotel hin und zurück befördern. An jedem Ort warten 3 Gäste. Es soll jeweils ein Gast transportiert werden.

Der Roboter fährt die Strecke vom Flughafen zum Hotel vorwärts. Lässt den Gast ein- und aussteigen und fährt nach einer Wartezeit die gleiche Strecke rückwärts zurück.

Auf den Parkflächen darf der Roboter neu ausgerichtet werden!

Aufgabe 3
gelöst ?



Aufgabe 3
unterschrieben ?



Aufgabe 3



Aufgabe 4



DAS SPIELFELD: Legostadt

Aufgabe 4: Einparken mittels Tastsensor

Start: Parkfläche vor Hotel

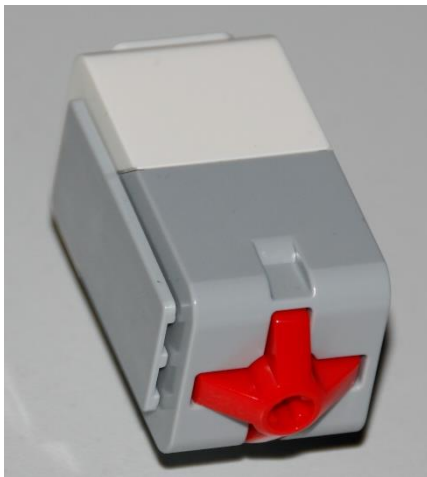
Ende: P3

Der Roboter soll rückwärts einparken. Er soll anhalten, wenn der Tastsensor die Bande berührt.



DAS SPIELFELD: Legostadt

Berührungssensor / Tastsensor



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



DAS SPIELFELD: Legostadt

Zur Arbeit mit Sensoren

Im Menü:

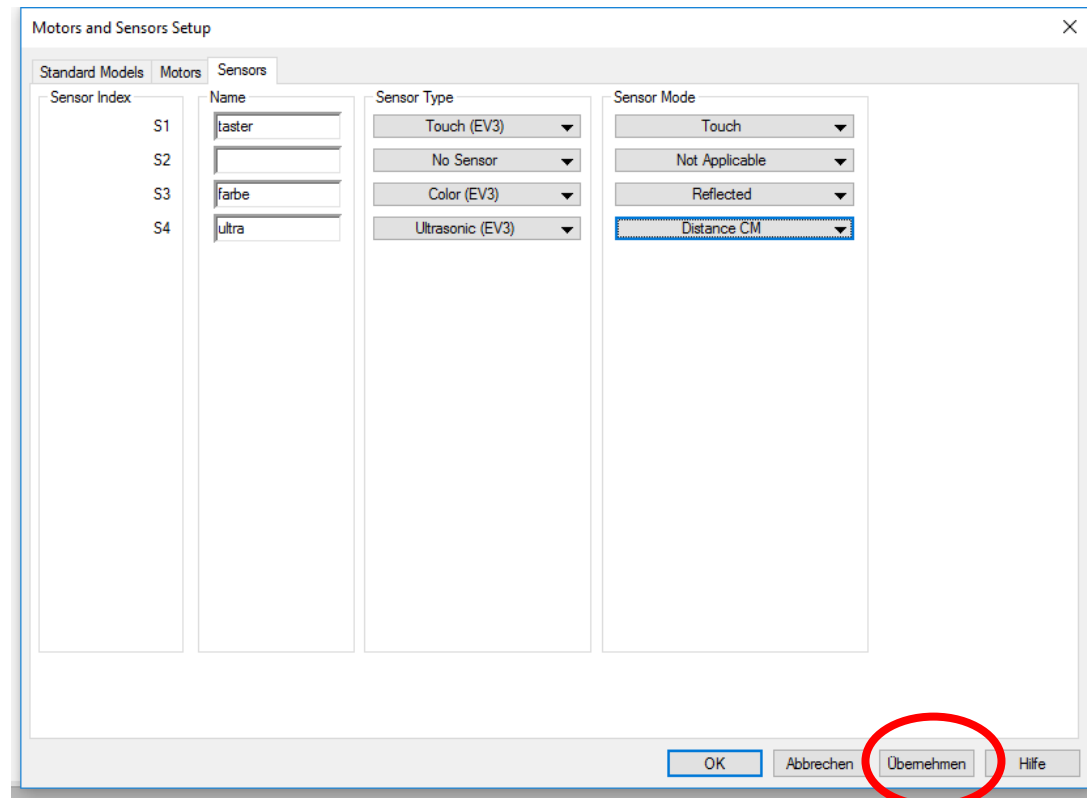
**Motor und
Sensor Setup**

Sensoren festlegen!

Anschließend auf

Übernehmen

klicken!

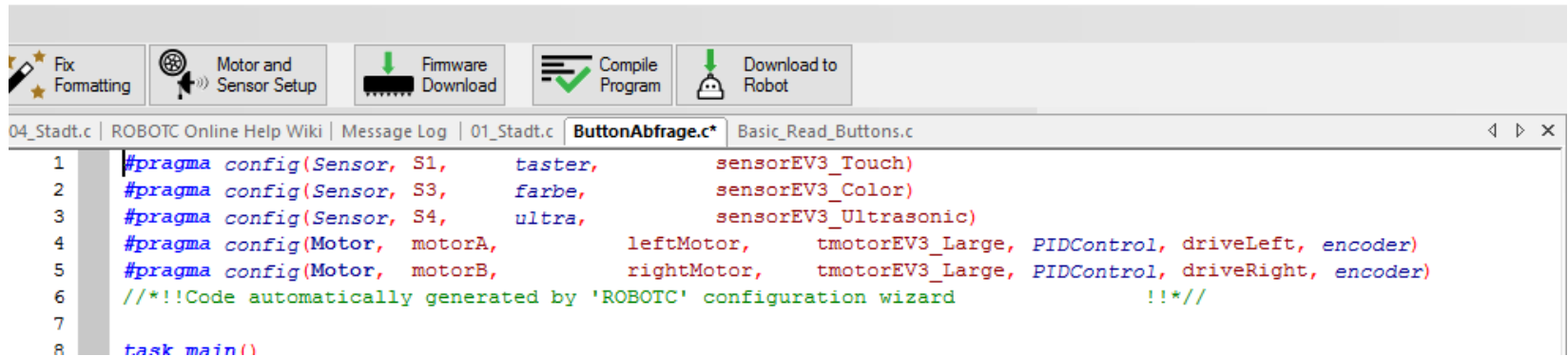




DAS SPIELFELD: Legostadt

Zur Arbeit mit Sensoren

Screenshot nach Festlegung der Sensoren



The screenshot shows the ROBOTC IDE interface. The top toolbar includes icons for Fix Formatting, Motor and Sensor Setup, Firmware Download, Compile Program, and Download to Robot. The file explorer shows the following files: 04_Stadt.c, ROBOTC Online Help Wiki, Message Log, 01_Stadt.c, ButtonAbfrage.c* (selected), and Basic_Read_Buttons.c. The main editor displays the following code:

```
1  #pragma config(Sensor, S1,    taster,    sensorEV3_Touch)
2  #pragma config(Sensor, S3,    farbe,    sensorEV3_Color)
3  #pragma config(Sensor, S4,    ultra,    sensorEV3_Ultrasonic)
4  #pragma config(Motor,  motorA,    leftMotor,    tmotorEV3_Large, PIDControl, driveLeft, encoder)
5  #pragma config(Motor,  motorB,    rightMotor,    tmotorEV3_Large, PIDControl, driveRight, encoder)
6  /**!!Code automatically generated by 'ROBOTC' configuration wizard    !**//
7
8  task main()
```



DAS SPIELFELD: Legostadt

Zur Arbeit mit dem Tastsensor

Abfrage:

getTouchValue(nDeviceIndex)

nDeviceIndex: Anschlussport (S1, S2, S3 oder S4)



DAS SPIELFELD: Legostadt

Die bedingte while-Schleife

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

Beispiel:

```
while(<<Bedingung>>){  
  <<Anweisung>>  
  ...  
  <<Anweisung>>  
}
```

```
while(<<i<=5>>){  
  <<Anweisung>>  
  ...  
  <<Anweisung>>  
}
```



DAS SPIELFELD: Legostadt

Beispiel: Tastsensor

```
EGO Start Page | Message Log | TastsensorBeispiel.c | Anzeige.c
1  #pragma config(Sensor, S1,      Taster,      sensorEV3_Touch)
2  /**!!Code automatically generated by 'ROBOTC' configuration wizard      !!*/
3
4  task main()
5  {
6      int zahl=0;
7      string szahl;
8      //Display loeschen
9      eraseDisplay();
10     displayTextLine(1,"Tastsenor druecken!");
11     while(getTouchValue(S1)==0)
12     {szahl=sprintf(szahl, "%1.0f", zahl);
13         displayBigTextLine(2,szahl);
14         zahl=zahl+1;
15     }
16
17     displayTextLine(4,"Druecke ein Button");
18     waitForButtonPress();
19
20 }
```



DAS SPIELFELD: Legostadt

Aufgabe 4: Einparken mittels Tastsensor

Start: Parkfläche vor Hotel

Ende: P3

Der Roboter soll rückwärts einparken. Er soll anhalten, wenn der Tastsensor die Bande berührt.

Aufgabe 4
gelöst ?



Aufgabe 4
unterschrieben ?



Aufgabe 4
✓



Aufgabe 5



DAS SPIELFELD: Legostadt

Aufgabe 5: Einparken mittels Ultraschallsensor

Start: Parkfläche Schule

Ende: P1 – Garage

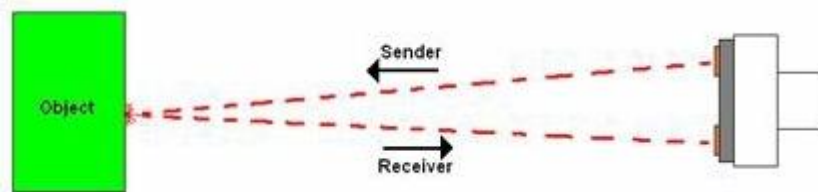
Der Roboter holt einen Schüler ab. Dabei parkt er selbstständig in die Garage ein. Er soll stehenbleiben, wenn der Abstand zur Wand kleiner als 5 cm ist.

DAS SPIELFELD: Legostadt

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
- Messbereich: 3 bis 250 cm
- Messgenauigkeit: +/- 1 cm
- **Messwerte werden in Meter ausgegeben**





DAS SPIELFELD: Legostadt

Zur Arbeit mit dem Tastsensor

Abfrage:

getUSDistance (distanceCM)

distanceCM – Entfernung in cm



DAS SPIELFELD: Legostadt

Beispielprogramm: Ultraschallsensor

Das Programm zeigt die Entfernung in Metern an, solange der Abstand größer ist als 10 cm.



DAS SPIELFELD: Legostadt

Aufgabe 5: Einparken mittels Ultraschall-sensor

Start: Parkfläche Schule

Ende: P1 – Garage

Der Roboter holt einen Schüler ab. Dabei parkt er selbstständig in die Garage ein. Er soll stehenbleiben, wenn der Abstand zur Wand kleiner als 5 cm ist.

Aufgabe 5
gelöst ?



Aufgabe 5
unterschrieben ?



Aufgabe 5
✓



Aufgabe 6



DAS SPIELFELD: Legostadt

Aufgabe 6: Ausflugsziel

Start: P4

Ende: entsprechendes Farbfeld

Der Roboter soll in Abhängigkeit von ermittelten Farbe am entsprechenden Ausflugsziel anhalten. Das Farbfeld wird über eine Zufallszahl ermittelt (siehe Folie73). Die Zufallszahl soll angezeigt werden.

0 – Gelb (Farb-ID: 3)

1 – Blau (Farb-ID: 2)

2 – Schwarz (Farb-ID: 7)

3 – Rot (Farb-ID: 0)



DAS SPIELFELD: Legostadt

Colorsensor – ColorID Mode



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



DAS SPIELFELD: Legostadt

Zur Verwendung des Farbsensors (ColorID Mode)

Initialisierung:

```
SensorModes colorSensor = new EV3ColorSensor(SensorPort.S3);  
SampleProvider col = colorSensor.getMode("ColorID");
```

Abfrage der Messwerte:

```
// Intialisierung der Messwerte  
int sampleSize = colorSensor.sampleSize();  
float[] sample = new float[sampleSize];  
int farbe;  
  
// Abfrage der Messwerte  
col.fetchSample(sample, 0);  
// Umrechnung float in integer  
farbe = (int)sample[0];
```

Jeweiligen Anschlußport
angeben (S1, S2, S3 oder S4)

Die Variable **farbe**
gibt den erkannten
Farbwert aus. Diese
ist abzufragen.



DAS SPIELFELD: Legostadt

Beispielprogramm: Farbsensor

```
import lejos.hardware.Button;
import lejos.hardware.lcd.LCD;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.*;
import lejos.robotics.*;
import lejos.utility.Delay;

public class FarbsensorBeispiel {

    public static void main(String[] args) {

        // Inhalt nächste Folie
    }
}
```

Das Programm zeigt 4 Messwerte an.



DAS SPIELFELD: Legostadt

Beispielprogramm: Farbsensor

```
public static void main(String[] args) {  
  
    // Initialisierung Farbsensor  
    SensorModes colorSensor1 = new EV3ColorSensor(SensorPort.S3);  
    SampleProvider col1 = colorSensor1.getMode("ColorID");  
  
    // Intialisierung der Messwerte  
    int SampleSize = colorSensor1.sampleSize();  
    float[] sample = new float[SampleSize];  
  
    // Variable für den Farbwert  
    int farbe;  
    LCD.clearDisplay();
```

Das Programm zeigt 4 Messwerte an.



DAS SPIELFELD: Legostadt

Beispielprogramm: Farbsensor

```
for(int i=1;i<=4;i++){  
  LCD.drawString("Messung starten", 0, 1);  
  LCD.drawString("Knopf druecken", 0, 2);  
  Button.waitForAnyPress();  
  // Messwert erfassen  
  coll.fetchSample(sample, 0);  
  // Umrechnung des Messwertes in eine Integervariable  
  farbe = (int)sample[0];  
  
  // Anzeige Messwert  
  LCD.drawString("Farbwert:", 0, 3);  
  LCD.drawInt(farbe, 0, 4);  
  Delay.msDelay(2000);  
  LCD.clearDisplay();  
}
```

Das Programm zeigt 4 Messwerte an.



DAS SPIELFELD: Legostadt

Abfrage einer Zufallszahl

Benötigt wird die Import-Funktion:

```
import java.util.*;
```

Festlegung des Wertebereiches:

```
Random wuerfel = new Random();
```

Erzeugung einer Zufallszahl (integer) im Wertebereich 0...3:

```
int zahl zahl = wuerfel.nextInt(3);
```



DAS SPIELFELD: Legostadt

Aufgabe 6: Ausflugsziel

Start: P4

Ende: entsprechendes Farbfeld

Der Roboter soll in Abhängigkeit von ermittelten Farbe am entsprechenden Ausflugsziel anhalten. Das Farbfeld wird über eine Zufallszahl ermittelt (siehe Folie 73). Die Zufallszahl soll angezeigt werden.

0 – Gelb (Farb-ID: 3)

1 – Blau (Farb-ID: 2)

2 – Schwarz (Farb-ID: 7)

3 – Rot (Farb-ID: 0)

Aufgabe 6
gelöst ?



Aufgabe 6
unterschrieben ?



Aufgabe 6
✓



Aufgabe 7



DAS SPIELFELD: Legostadt

Aufgabe 7: Folge dem Weg zum Leuchtturm

Start: P3

Ende: Gelbes Feld beim Leuchtturm

Der Roboter soll der schwarzen Linie zum Leuchtturm folgen. Der Roboter soll anhalten, sobald das Endfeld (gelb) erreicht ist.

Aufgabe 7
gelöst ?



Aufgabe 7
unterschrieben ?



Aufgabe 7
✓



Ende