

Grundlagen des

# ***AUTONOMEN FAHRENS***

Jürgen Enders



## Konzept

Taschenrechner sind wahrscheinlich die bekanntesten Produkte von Texas Instruments (TI). Jedoch umfasst die Produktpalette des Unternehmens weit mehr. Im Zentrum stehen analoge und digitale Bausteine, u. a. für Industrie- und Automobilanwendungen.

Das Konzept der **TI MINT Workshops** verbindet die Erfahrungen von Texas Instruments aus dem Bereich Automotive Technology mit den Möglichkeiten der Programmierung mithilfe der TI-Nspire™ CX Technologie, dem TI-Innovator™ Hub und dem programmierbaren Roboterfahrzeug TI-Innovator™ Rover.

Der TI MINT Workshop „Grundlagen der Programmierung an Beispielen des Autonomen Fahrens“ wurde mit der Comenius EduMedia Medaille 2019 **ausgezeichnet**.



Die Gesellschaft für Pädagogik, Information und Medien e.V. (GPI) fördert seit 1995 mit Comenius-EduMedia-Auszeichnungen pädagogisch, inhaltlich und gestalterisch herausragende IKT-basierte Bildungsmedien.

## Zielgruppe

Zielgruppe dieses halbtägigen Workshops sind primär **Schülerinnen und Schüler** der Sekundarstufen I und II an weiterführenden Schulen. Vorkenntnisse im Bereich der Programmierung sowie im Umgang mit TI-Nspire™ CX sind hilfreich, aber keinesfalls Voraussetzung. Konzipiert ist der Kurs für bis zu 20 Schülerinnen und Schüler.

Interessierten **Lehrerinnen und Lehrern** bietet das TI Schulberater-Team das Workshopkonzept auch zur Fortbildung an, wobei die Schwerpunkte dieser Veranstaltungen im Vorfeld abgestimmt werden.



Zur Auftaktveranstaltung zum Girls' Day 2019 hatte Bundeskanzlerin Angela Merkel die Gelegenheit, sich das Workshopkonzept von Mädchen des Berliner Primo-Levy-Gymnasiums erklären zu lassen.

Mehr Informationen zum TI MINT Workshop und zur TI Technologie finden Sie auf den TI Webseiten **education.ti.com**.

# Einführung

Der TI-Innovator™ Hub mit TI Launchpad™ Board ist ein mit industriellen Komponenten aufgebautes Interface, das die Signale von Sensoren aufnehmen und Aktoren ansteuern kann. Dazu gibt es viele fertig aufgebaute Module, aber man kann auch eigene Schaltungen auf Steckplatinen (Breadboard) entwerfen und anschließen.

Der TI-Innovator™ Hub funktioniert nur im Zusammenspiel mit einem TI-Nspire™ CX / CAS, einem TI-Nspire™ CX II-T / CAS oder einem TI-84 Plus CE-T bzw. der entsprechenden Computersoftware, da er auf die Stromversorgung dieser Geräte angewiesen ist. Auf diesen Geräten werden auch die Programme geschrieben, die für den Betrieb des TI-Innovator™ Hub notwendig sind. Die möglichen Programmiersprachen sind TI Basic oder LUA.





# Kurze Einführung in das Programmieren mit TI Basic

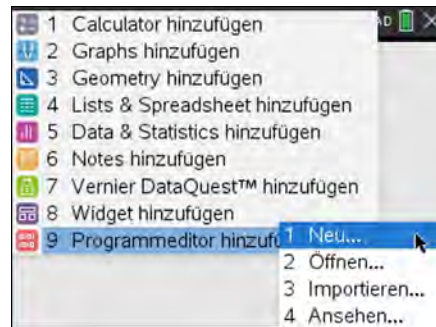
Dies ist keine vollständige Einführung in die Programmiersprache, sondern anhand eines Beispiels eine Zusammenfassung einiger Befehle, die zum Verständnis der Beispielprogramme sinnvoll sind.

## AUFGABE:

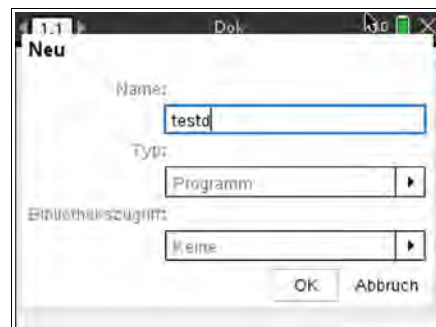
Es soll ein Programm geschrieben werden, das folgendes leistet:

Nähert man sich dem Ultraschall-Entfernungssensor (Ranger) auf weniger als 10 cm, so ertönt dreimal eine kurze Warntonfolge und die RGB-LED leuchtet rot auf. Der Ranger wird mit dem Eingang IN 1 des Hub verbunden, der Taschenrechner mit dem USB-Port (Steckertyp B des kurzen Kabels). Das Programm soll durch Drücken der Taste `[esc]` beendet werden.

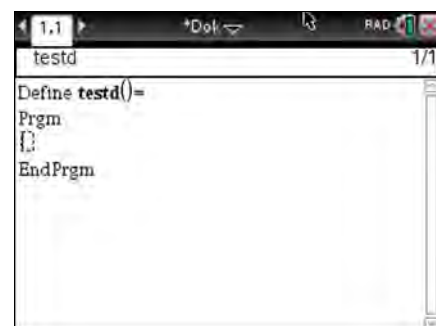
Soll ein neues Programm geschrieben werden, so fügt man eine neue Seite zu dem Dokument hinzu und wählt darin den Programmierer und die Auswahl `1:Neu ...`



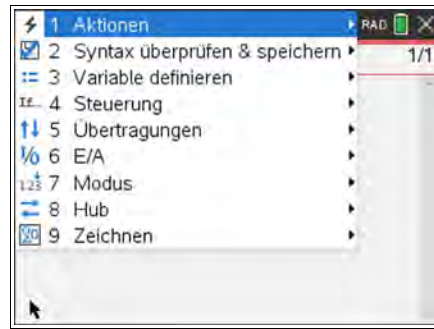
Es öffnet sich ein Fenster, in dem ein Name für das Programm eingegeben werden muss. Im Beispiel wurde der Name `testd` gewählt.



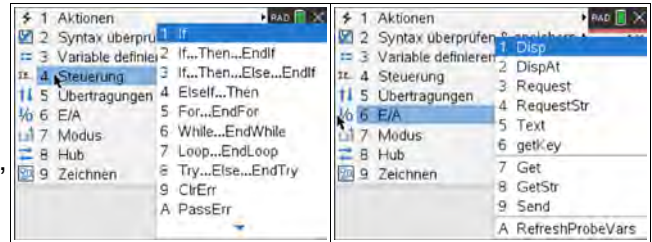
Schließt man das Fenster, so kommt man in den Editiermodus. Alle Programmierbefehle werden zeilenweise in den Bereich zwischen `Prgm` und `EndPrgm` eingefügt, wo sich jetzt das gestrichelte Rechteck befindet. Jeder Befehl kommt in eine neue Zeile, Leerzeilen werden später bei der Ausführung des Programmes ignoriert.



Im *Menü* befinden sich in Gruppen zusammengefasst alle Programmierbefehle.

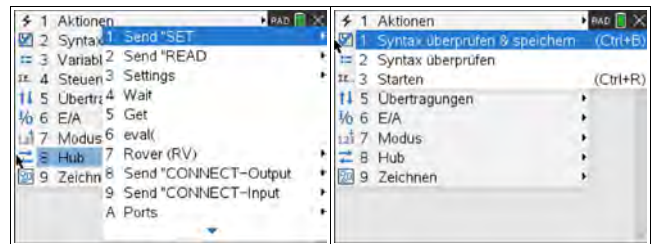


Das *Menü 4: Steuerung* enthält Befehle für Verzweigungen, Schleifen, usw.



Das *Menü 6: E/A* enthält alle Befehle zur Kommunikation mit dem Nutzer (Anzeigebefehle, Eingabebefehle)

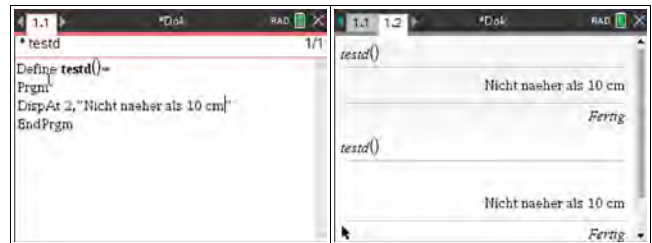
Das *Menü 8: Hub* enthält alle Befehle zur Kommunikation mit dem TI-Innovator™ Hub.



Das *Menü 2: Syntax überprüfen* enthält Prüfbefehle, den Speicherbefehl sowie den Startbefehl für das Programm. Das Speichern bezieht sich allerdings nur auf den Arbeitsspeicher!

a. Die Überschrift - der Befehl *DispAt* aus dem *Menü E/A* bewirkt, dass der in Anführungszeichen stehende Text *immer* in der 2. Zeile unter dem Trennstrich auf dem Home-Display dargestellt wird. *Disp* allein schreibt den Text immer in eine neue Zeile.

b. Zur Verdeutlichung:  
 DispAt 1,... oben  
 DispAt 2,... unten



a

b

Der Befehl `Send „CONNECT RANGER 1 TO IN 1“` bewirkt die Zuordnung des RANGER 1 zum Eingang IN 1. Die Nummer bei RANGER gehört zwingend dazu und muss per Hand eingefügt werden.

Die Befehle finden sich im *Hub-Menü* unter `Send „CONNECT - Input, Settings und Ports.` Man kann den ganzen Text in den Anführungszeichen auch von der Tastatur eingeben, muss allerdings die Syntax genau (Großschreibung!) beachten. Der `Send`-Befehl bewirkt, dass die in den Anführungszeichen stehende Zeichenkette an den Hub gesendet wird.

Einfügen der zentralen *While*-Schleife:

`getKey()` liest den Tastaturcode einer gedrückten Taste. Solange man nicht die Taste `[esc]` gedrückt hat, werden die Befehle zwischen `While` und `EndWhile` ohne Ende wiederholt.

`While` findet man in *Steuerung*, `getKey` in *E/A*.

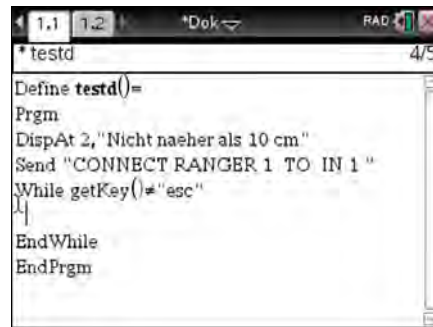
Innerhalb der *While*-Schleife wird durch den Befehl `Send „READ RANGER 1“` aus dem Menü `Send „READ` fortlaufend die Entfernung gemessen (in der Maßeinheit m) und in einem Zwischenspeicher auf dem Hub abgelegt. Bei der nächsten Messung würde der Wert sofort überschrieben werden; deshalb muss er vorher durch den Befehl `Get` ausgelesen und einer Variablen zugewiesen werden, hier der Variablen `d` (von *D*istance; die Wahl des Variablennamens ist aber beliebig).

Einfügen der Verzweigung *If ... Then ... EndIf*:

Ist  $d < 0,1$  m, so sollen Aktionen erfolgen.

$d < 0.1$  ist (ebenso wie `getKey() ≠ "esc"` in der *While*-Schleife) eine Bedingung, die entweder wahr (1) oder falsch (0) ist.

Ist sie wahr, so soll die eingebaute RGB-LED rot leuchten. Der Befehl `Send „SET COLOR.RED` befindet sich im Menü `Send „SET`. Die Einstellungen `ON` und `OFF` befinden sich im Menü `SETTINGS`. Man darf nicht vergessen, die LED wieder auszuschalten, denn sonst leuchtet sie immer weiter, egal was passiert, selbst wenn das Programm beendet ist und man weiter editiert!



```
* testd 4/5
Define testd()=
Prgm
DispAt 2,"Nicht naeher als 10 cm"
Send "CONNECT RANGER 1 TO IN 1 "
While getKey()≠"esc"
}
EndWhile
EndPrgm
```



```
* testd 6/7
Define testd()=
Prgm
DispAt 2,"Nicht naeher als 10 cm"
Send "CONNECT RANGER 1 TO IN 1 "
While getKey()≠"esc"
Send "READ RANGER 1"
Get d
}
EndWhile
EndPrgm
```



```
* testd 9/11
Send "CONNECT RANGER 1 TO IN 1 "
While getKey()≠"esc"
Send "READ RANGER 1"
Get d
If d<0.1 Then
Send "SET COLOR.RED ON "
Send "SET COLOR.RED OFF "
EndIf
EndWhile
```

Es fehlt noch die Warntonfolge. Im Menü *Send* "SET" befindet sich unter der Bezeichnung *SOUND* der eingebaute kleine und recht leise Lautsprecher. Der Befehl muss noch vervollständigt werden durch die Frequenz des zu hörenden Tones in Hz.

Die in der Aufgabe geforderte Tonfolge besteht hier aus zwei Tönen mit den Frequenzen 440 Hz und 220 Hz, die beide 0,5 s lang ertönen sollen.

Dafür sorgt der Befehl *Wait*, der die weitere Ausführung des Programmes um die angegebene Zeit (hier 0,5 s) anhält.

Die Tonfolge wird dreimal innerhalb einer *For*-Schleife abgespielt:

*i* ist die Schleifenvariable (Name beliebig)

1 ist der Startwert, von dem aus in Schritten von 1 bis zum Endwert 3 hochgezählt wird.

Einmal eingeschaltet, würde auch der Lautsprecher unbegrenzt weiter laufen; deshalb wird er mit dem Befehl *Send* "SET SOUND OFF" ausgeschaltet.

```

testd
Send "SET COLOR.RED ON "
For i,1,3
  Send "SET SOUND 440"
  Wait 0.5
  Send "SET SOUND 220"
  Wait 0.5
EndFor
Send "SET COLOR.RED OFF "
Send "SET SOUND OFF "
EndIf
    
```

**Starten des Programmes:**

Das geschieht in zwei Schritten:

- 1. Menu 2: 1: Syntax überprüfen & speichern
- 2. Menu 2: 3: Starten

Jetzt befindet man sich im *Calculate*-Bereich des Taschenrechners. Mit einem Druck auf  wird das Programm gestartet.

**Abbrechen eines Programmes:**

Durch einen Fehler bei der Programmierung kann ein Programm endlos weiterlaufen. Man kann es jedoch unterbrechen

- auf dem Handheld durch Drücken von  und mehrfach

- auf dem PC durch *F12* und *Eingabe*.

**Fehlermeldungen:**

links: bei der Syntaxüberprüfung (0,1 statt 0.1)

rechts: bei der Programmausführung; mit *Gehe zu* kommt man in den Editiermodus und in die Nähe des Fehlers.



Fehlermeldungen, die Befehle für den Hub betreffen, werden durch einen kurzen Piepton und das Aufblitzen der roten Fehler-LED auf dem Hub angezeigt; das Programm läuft aber weiter.



## Der Rover

Der Rover ist ein zweirädriges Fahrzeug mit zwei Fahrmotoren und einer Kugel als dritter Stütze, die ab und zu gereinigt und ev. mit Graphit-Pulver eingestäubt werden muss, damit sie gut gleitet. Der Rover muss zwingend mit dem Innovator verbunden werden, der unter dem Chassis in ein Fach eingeschoben wird, so dass die Ports IN 1 .. IN 3 und OUT 1 .. OUT 3 zugänglich bleiben im Gegensatz zu den Breadboard-Ports und dem I<sup>2</sup>C-Port, die alle vom Rover selbst benötigt werden. Auf dem Rover ist dann der Platz für den Taschenrechner, der mit dem Innovator verbunden wird und das Programm für den Rover enthält. Der Rover verfügt über einen fest eingebauten Abstandssensor, einen RGB-Lichtsensoren mit weißer LED zur Spurverfolgung, eine RGB-LED und einen gesonderten Akku, mit dem über 6 Stunden Betrieb möglich sind. In dieser Zeit kann der Rover mit den Standardeinstellungen fast 4 km zurücklegen. Selbst wenn die Ladeanzeige nur noch 2 LEDs anzeigt, ist noch ein langer Betrieb ohne Einschränkungen möglich. Steigungen werden bis zu 12° überwunden, aber man benötigt zum Betrieb auf jeden Fall einen glatten Untergrund ohne Fugen. Ferner hat der Rover noch einen Stifthalter. Als Zeichenstifte eignen sich dünne Filzstifte vom Typ non-permanent, um eventuelle Spuren besser beseitigen zu können.

Gesteuert wird der Rover mit zahlreichen speziellen Basic-Befehlen vom Taschenrechner aus über den Innovator. In dessen Menü findet man einen Unterpunkt ROVER, in dem sich alle Befehle befinden. Die Befehle gliedern sich dabei in zwei Gruppen: Alle Befehle, die das Fahren betreffen, werden der Reihe nach abgearbeitet wie gewohnt. Da sie aber mehr Zeit benötigen als die übrigen Befehle wie z.B. das Anschalten der LED, werden diese anderen Befehle parallel dazu ausgeführt. Man muss beim Programmieren also stets die zeitliche Komponente mit berücksichtigen und den korrekten Programmablauf immer wieder überprüfen.

### Beispiel:

**Unbedingt notwendig:** Rover einschalten und mit dem Handheld verbinden

- a. Rover 4 s *vorwärts* fahren lassen
- b. *währenddessen* blinkt die On-Board-LED 3 mal rot - aber nicht wie eingestellt im 1 s - Rhythmus, sondern viel schneller, da der nachfolgende Befehl den vorangehenden abbricht
- c. Rover 4 s *rückwärts* fahren lassen - das passiert unmittelbar im Anschluss an a.
- d. unmittelbar im Anschluss von b. blinkt nun die grüne LED korrekt, da die Programmausführung jeweils um 1 s angehalten wird. Der Rover fährt unbeeinflusst dabei weiter.
- e. der Rover steht nun still und es tut sich scheinbar nichts, da das Programm um 5 s nach dem letzten Blinken angehalten wurde
- f. jetzt blinkt die blaue LED wieder viel zu schnell, da der *Wait*-Befehl wieder fehlt

```
Define roverdemo()=
Prgm
:Send "CONNECT RV"
:Send "RV FORWARD TIME 4"
:For i,1,3
:  Send "SET RV.COLOR 255 0 0 TIME
1"
:  Send "SET RV.COLOR 0 0 0 TIME 1"
:EndFor
:Send "RV BACKWARD TIME 4"
:For i,1,3
:  Send "SET RV.COLOR 0 255 0 TIME
1"
:  Wait 1
:  Send "SET RV.COLOR 0 0 0 TIME 1"
:  Wait 1
:EndFor
:Wait 5
:For i,1,3
:  Send "SET RV.COLOR 0 0 255 TIME
1"
:  Send "SET RV.COLOR 0 0 0 TIME 1"
:EndFor
:EndPrgm
```

# Aufgaben

## 1. AUTONOMES FAHREN: STAUBSAUGER UND RASENMÄHER

Selbstfahrende Roboter, die unermüdlich den Teppich saugen oder den Rasen schneiden, sind mittlerweile weit verbreitet. Viele gehen dabei nicht systematisch vor, sondern wählen ihren Weg zufällig, so dass im Laufe der Zeit die gesamte Fläche bearbeitet wird. Sie benötigen dazu eine Begrenzung, die die zu bearbeitende Fläche definiert. Gerät der Roboter an die Begrenzung, so muss er diese erkennen und umkehren. In der einfachsten Form kann man dazu einen Aufprallschalter verwenden. In den ersten beiden Beispielen werden anstelle des Schalters Ultraschallsensoren verwendet.

Das einfache Programm 1 nutzt ausschließlich den im Rover fest eingebauten Ranger. Er reicht jedoch nicht aus, wenn der Rover schräg auf die Begrenzung auftrifft. Deshalb wird er noch mit zwei weiteren, vorne seitlich angebrachten Rangern ergänzt (Programm 2), die den Bereich schräg nach vorne umfassen. Sie lassen sich mit LEGO-Bauteilen am Rover befestigen. Zusätzlich wird über Licht- und Tonsignale vor dem fahrenden und wendenden Rover gewarnt.

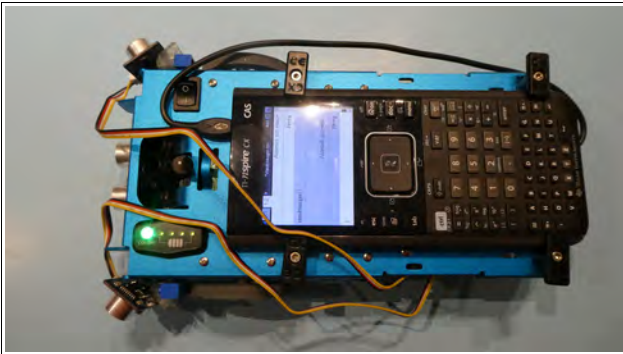
Eine andere Methode der „Einzäunung“ ist, dass ein Draht verlegt wird, dessen Signal vom Roboter erkannt wird. Dieses Verfahren ist besonders gut für Rasenmäher geeignet. Programm 3 simuliert diese Variante. Hier wird kein Kabel verlegt, sondern stattdessen ein breiter schwarzer Streifen Klebeband benutzt, der von dem eingebauten Lichtsensor erkannt wird.

### Programm 1:

Variablendeklaration  
 Verbindung mit dem Rover  
 Betriebsanweisung  
 zentrale *While*-Schleife: Abbruch mit `esc`  
 Rover fährt vorwärts  
 Entfernung *d* lesen  
  
*d* < 0,15 m : Ausweichmanöver:  
  
 anhalten  
 0,3 m rückwärts fahren  
 um 60° rechts drehen  
  
 Ausschalten des Rovers am Programmende

```
Define staubsauger()=
Prgm
:Local d
:Send "CONNECT RV"
:DispAt 2,"Abbruch mit <esc>"
:While getKey()!="esc"
:  Send "RV FORWARD"
:  Send "READ RV.RANGER"
:  Get d
:  If d<0.15 Then
:    Send "RV STOP"
:    Wait 1
:    Send "RV BACKWARD 0.3 M"
:    Wait 2
:    Send "RV RIGHT 60 DEGREES"
:    Wait 2
:  EndIf
:EndWhile
:Send "RV STOP"
:EndPrgm
```

**Programm 2:**



Anbau der Ultraschall-Sensoren. Für beide Sensoren werden etwas längere Kabel benötigt.

Die Sensoren werden mit LEGO-Technik-Elementen auf Dachsteinen angebracht, auf die sie mit kleinen Schrauben fixiert befestigt sind..

Unterschiede zu Programm 1:

Verbindung der zwei zusätzlichen Sensoren mit den Ports IN 1 und IN 2

Einschalten der bordeigenen RGB-LED: grün

Einlesen der Werte aller drei Sensoren in die Variablen d, e und f

Um e und f erweiterte Abfrage, wobei für e und f kleinere Schwellenwerte (0,1 m) als für d (0,15 m) verwendet werden. Die Werte wurden durch Versuch ermittelt.

RGB-LED: Umschalten auf rot-blinkend bordeigener Lautsprecher für Dauer des Ausweichmanövers (5 s) mit 500 Hz einschalten

RGB-LED: Umschalten auf grün Lautsprecher ausschalten


RGB-LED ausschalten

```

Define staubsauger()=
Prgm
:Local d,e,f
:Send "CONNECT RV"
:Send "CONNECT RANGER 1 TO IN 1"
:Send "CONNECT RANGER 2 TO IN 2"
:DispAt 2,"Abbruch mit <esc>"
:While getKey()!="esc"
:  Send "RV FORWARD"
:  Send "SET RV.COLOR 0 255 0"
:  Send "READ RV.RANGER"
:  Get d
:  Send "READ RANGER 1"
:  Get e
:  Send "READ RANGER 2"
:  Get f
:  If d<0.15 or e<0.1 or f<0.1 Then
:    Send "RV STOP"
:    Send "SET RV.COLOR 255 0 0"
BLINK 1"
:    Send "SET SOUND 500 TIME 5"
:    Wait 1
:    Send "RV BACKWARD 0.3 M"
:    Wait 2
:    Send "RV RIGHT 60 DEGREES"
:    Wait 2
:    Send "SET RV.COLOR 0 255 0"
:    Send "SET SOUND OFF"
:  EndIf
:EndWhile
:Send "RV STOP"
:Send "SET RV.COLOR 0 0 0"
:EndPrgm
    
```

**Programm 3:**

Variablendeklaration  
Anschluss des Rovers

zentrale *While*-Schleife: Abbruch bei   
langsameres Fahren vorwärts, damit die Haltelinie nicht übersehen wird  
kurze Wartezeit zur Datenermittlung  
die Haltelinie besteht aus breitem schwarzen Klebeband, deshalb Einlesen als Graustufe und Anzeige des Grauwertes  
der Grenzwert  $g < 50$  wurde experimentell ermittelt  
die LED leuchtet rot auf, der Rover hält an, fährt rückwärts, wendet um  $60^\circ$

die LED erlischt

sicherheitshalber wird  $g=0$  gesetzt  
Ende der *While*-Schleife

Programmende

```

Define rasenmaeher()=
Prgm
:Local g
:Send "CONNECT RV"
:DispAt 2,"Abbruch mit <esc>"
:While getKey()!="esc"
:  Send "RV FORWARD SPEED 0.15 M/S"
:  Wait 0.1
:  Send "READ RV.COLORINPUT.GRAY"
:  Get g
:  DispAt 2,g
:  If g<50 Then
:    Send "SET RV.COLOR.RED 255"
:    Send "RV STOP"
:    Wait 1
:    Send "RV BACKWARD 0.3 M"
:    Wait 2
:    Send "RV RIGHT 60 DEGREES"
:    Wait 2
:    Send "SET RV.COLOR.RED 0"
:    Send "RV STOP"
:  EndIf
:  g:=0
:EndWhile
:Send "RV STOP"
:EndPrgm

```



## 2. AUTONOMES FAHREN: RÜCKWÄRTS EINPARKEN

Es ist die klassische Fahrschulaufgabe und vollzieht sich nach einem bestimmten Schema, so dass man diesen Vorgang auch gut automatisieren kann.

### a) Der einfache Fahrschulfall: Einparken hinter einem Fahrzeug

Der Vorgang vollzieht sich in folgenden Schritten:

1. links neben das parkende Fahrzeug fahren
2. rückwärts erst nach rechts lenken, dann nach links, bis man hinter dem Fahrzeug an der Bordsteinkante steht, ohne den (nicht vorhandenen) Hintermann anzustoßen
3. langsam wieder etwas vorfahren, so dass man etwa mittig in der (gedachten) Parklücke steht.

<b>Programm:</b>	Define einparken()=
	Prgm
Verbindung mit dem Rover	: Send "CONNECT RV"
langsam 4 s voraus fahren	: Send "SET RV.MOTORS LEFT -150 RIGHT 150 TIME 4"
warten auf den Abschluss der Fahrt	: Wait 5
Rückwärts nach rechts fahren für 1,2 s	: Send "SET RV.MOTORS LEFT 255 RIGHT -120 TIME 1.2"
Rückwärts nach links fahren für 1,2 s	: Send "SET RV.MOTORS LEFT 120 RIGHT -255 TIME 1.2"
Programm anhalten, bis diese Fahrten abgeschlossen sind	: Wait 3.5 : Send "RV STOP "
langsam 1,5 s voraus fahren	: Send "SET RV.MOTORS LEFT -100 RIGHT 100"
Anhalten	: Wait 1.5 : Send "RV STOP "
	:EndPrgm

### b) Der übliche Fall: Parkplatzsuche am Straßenrand und einparken

Der Rover fährt langsam dicht an der Reihe abgestellter „Autos“ vorbei und registriert über einen zusätzlichen seitlichen Abstandssensor die Lücken (LED leuchtet schwach gelb) zwischen den Autos (LED leuchtet schwach rot). Die Länge der Lücken wird über einen Zähler im Programm gemessen. Hat dieser Zähler einen gewissen Wert erreicht, so leuchtet erst die LED kräftig grün auf, dann das Bremslicht kräftig rot und der Rover stoppt. Der Zähler steht jetzt bei 30 und die Lücke ist ca. 58 cm lang. Der Rover steht etwas neben dem vorderen „Auto“.

Der Blinker schaltet sich ein (externe orange LED blinkt) ebenso wie der Rückfahrscheinwerfer (bordeigene LED schwach weiß) und der Rover fährt in der oben beschriebenen S-Kurve in die Parklücke und bremst wieder.

Anschließend fährt er unter Verwendung des eingebauten Abstandssensors so weit vor, bis er etwa 10 cm hinter dem vorderen „Auto“ steht. Der Rover bremst erneut ab und das Programm endet.

<b>Programm:</b>	Define einparken2()=
Überschrift	Prgm
Verbindungen (unbedingt diese Reihenfolge einhalten)	: DispAt 1,"automatisches Einparken"
Zähler $n$ auf Null stellen (Anfangswert)	: Send "CONNECT LED 1 TO OUT1"
<i>While</i> -Schleife: solange Zähler $n < 30$ langsam fahren	: Send "CONNECT RV"
	: Send "CONNECT RANGER 1 TO IN1"
	: n:=0
	: While n<30
	: Send "RV FORWARD SPEED 0.14 M/S"
	: Send "READ RANGER 1"
Lücke vorhanden (Entfernung $d > 0,1 m$ )	: Get d
ja: Rover-LED leuchtet schwach gelb	: If d>0.1 Then
Zähler wird um 1 erhöht	: Send "SET RV.COLOR 64 64 0"
	: n:=n+1
nein: Rover-LED leuchtet schwach rot	: Else
Zähler wird wieder auf 0 gesetzt	: Send "SET RV.COLOR 64 0 0"
	: n:=0
	: EndIf
	: EndWhile
Lücke gefunden: Rover-LED für 0,2 s grün	: Send "SET RV.COLOR 0 255 0"
Rover-LED für 1 s rot (Bremslicht)	: Wait 0.2
	: Send "SET RV.COLOR 255 0 0"
	: Send "RV STOP "
Rover stoppt	: Wait 1
externe LED blinkt	: Send "SET LED 1 TO ON BLINK 2"
„Rückfahrcheinwerfer“ geht an	: Send "SET RV.COLOR 125 125 125"
	: Send "SET RV.MOTORS LEFT 160 RIGHT 0"
	: Wait 1.8
	: Send "SET RV.MOTORS LEFT 160 RIGHT -160"
	: Wait 0.7
	: Send "SET RV.MOTORS LEFT 0 RIGHT -160"
	: Wait 1.8
Rover stoppt	: Send "RV STOP "
externe LED wird ausgeschaltet	: Send "SET LED 1 TO OFF"
Rover-LED für 0,3 s rot	: Send "SET RV.COLOR 255 0 0"
	: Wait 0.3
	: e:=1
Rover fährt vorwärts bis auf $e > 0,1 m$ an das vordere „Auto“ heran	: While e>0.1
	: Send "READ RV.RANGER"
	: Get e
	: Send "RV FORWARD SPEED 0.14 M/S"
	: EndWhile
Rover-LED für 0,3 s rot	: Send "SET RV.COLOR 255 0 0"
	: Wait 0.3
Ausschalten der Rover-LED	: Send "SET RV.COLOR 0 0 0"
Rover stoppt	: Send "RV STOP "
	:EndPrgm

### 3. AUTONOMES FAHREN: SPURFOLGER

In vielen Firmen sind Materialtransportwagen autonom unterwegs. Sie folgen dabei festgelegten Wegen, die auf dem Hallenboden als Gefahrenzone für Menschen gekennzeichnet sind. Eine Möglichkeit, einen solchen Weg zu kennzeichnen, besteht darin, eine farbige Markierung auf dem Boden anzubringen. Eine optische Leseeinheit im Wagen sorgt dafür, dass der Wagen genau der Spur folgen kann.

Eine solche Einheit, bestehend aus einer weißen LED und einem farbsensitiven Photoelement, befindet sich vorne unter dem Rover dicht über dem Boden.

#### Aufgabe:

Es ist ein Programm zu schreiben, so dass der Rover einer roten Spur auf hellem Untergrund folgt.

#### Hinweis:

Diese Spur muss relativ breit sein (>3 cm), da der Rover nur über ein Photoelement verfügt. Der Rover findet die Spur und fährt sie in einer Richtung ab, auch über Ecken und Kurven hinweg.

#### Programm:

<p>Rover verbinden</p> <p>zentrale <i>While</i>-Schleife: Abbruch mit <span style="border: 1px solid black; padding: 0 2px;">esc</span></p> <p>Einlesen des Farbwertes</p> <p>1 ist der Wert für ein rotes Signal/die rote Spur</p> <p>On-Board-LED: grün</p> <p>geradeaus mit 0,18 m/s</p> <p>Geradeausfahrt, solange c=1 ist</p> <p>Stopp, wenn die Spurfarbe nicht mehr rot ist</p> <p>ON-Board-LED: rot</p> <p>Drehung, um die Spur wiederzufinden</p> <p>drehen, solange c≠1</p> <p>Stopp, wenn die rote Spur wieder erreicht wird</p> <p>LED ausschalten und Rover anhalten</p>	<pre> Define spur()= Prgm :Local c :Send "CONNECT RV" :DispAt 1,"Abbruch mit &lt;esc&gt;" :While getKey()≠"esc" :  Send "READ RV.COLORINPUT" :  Get c :  If c=1 Then :    Send "SET RV.COLOR 0 255 0" :    Send "RV FORWARD SPEED 0.18 M/S" :    While c=1 :      Send "READ RV.COLORINPUT" :      Get c :    EndWhile :    Send "RV STOP " :  Else :    Send "SET RV.COLOR 255 0 0" :    Send "SET RV.MOTORS LEFT 150 RIGHT 50" :    While c≠1 :      Send "READ RV.COLORINPUT" :      Get c :    EndWhile :    Send "RV STOP " :  EndIf :EndWhile :Send "SET RV.COLOR 0 0 0" :Send "RV STOP " :EndPrgm </pre>
---	---

## 4. AUTONOMES FAHREN: ABSTANDSWARNER

Bei diesem Programm wird nur der Abstand zu einem vorausfahrenden oder stehenden Fahrzeug mit dem im Rover eingebauten Ranger gemessen. Ist die Entfernung  $d > 0,4$  m, so fährt der Rover weiter mit Maximalgeschwindigkeit und eine Kontrolllampe (RGB-LED) leuchtet grün. Im Bereich von  $0,4 \text{ m} < d < 0,2 \text{ m}$  wird der Rover abhängig von  $d$  immer langsamer, die Kontrolllampe leuchtet gelb und ein ansteigender Ton warnt vor dem Hindernis. Ist  $d < 0,2$  m, so bleibt der Rover stehen, die LED leuchtet rot und ein hoher Dauerton ist zu hören.

Der im Innovator eingebaute Lautsprecher ist sehr leise und wird von den Fahrgeräuschen des Rovers überdeckt. Deshalb ist ein externer Lautsprecher an OUT 1 vorgesehen.

<b>Programm:</b>	Define wdistanz()=
Verbindungen herstellen	Prgm
Überschrift	:Send "CONNECT RV"
zentrale <i>While</i> -Schleife, Abbruch mit beliebiger Taste	:Send "CONNECT SPEAKER 1 TO OUT 1"
Abstand einlesen und in $d$ speichern	:DispAt 1,"Entfernungswarner"
	:While getKey()=""
	: Send "READ RV.RANGER"
	: Get d
	: If $d < 0.2$ Then
$d < 0,2$ m: RGB-LED rot	: Send "SET RV.COLOR 255 0 0"
Warnton 880 Hz	: Send "SET SPEAKER 1 880"
Rover anhalten	: Send "RV STOP"
$0,2 \text{ m} < d < 0,4 \text{ m}$	: ElseIf $d > 0.2$ and $d < 0.4$ Then
RGB-LED gelb	: Send "SET RV.COLOR 255 255 0"
Warnton abnehmender Frequenz $f$	: $f := 1500 * (0.6 - d)$
zunehmende Geschwindigkeit $v$	: Send "SET SPEAKER 1 eval(f) "
	: $v := ((d - 0.15) / (0.2)) * 255$
	: Send "SET RV.MOTORS LEFT
	eval(-v) RIGHT eval(v) "
$0,4 \text{ m} < d$ : RGB-LED grün	: ElseIf $d > 0.4$ Then
kein Warnton	: Send "SET RV.COLOR 0 255 0"
schnelle Fahrt	: Send "SET SPEAKER 1 OFF"
	: Send "SET RV.MOTORS LEFT -255
	RIGHT 255"
	: EndIf
	:EndWhile
Rover anhalten	:Send "RV STOP "
RGB-LED ausschalten	:Send "SET RV.COLOR 0 0 0"
Warnton ausschalten	:Send "SET SPEAKER 1 OFF"
	:EndPrgm



## 5. AUTONOMES FAHREN: RÜCKKEHR NACH HAUSE

Es erscheint derzeit nicht unmöglich, dass man in nicht allzu ferner Zukunft nach getaner Arbeit sich in sein Fahrzeug setzt und ihm den Befehl „Nach Hause“ erteilt. Sodann setzt sich das Fahrzeug in Bewegung und man verlässt es erst im heimischen Carport.

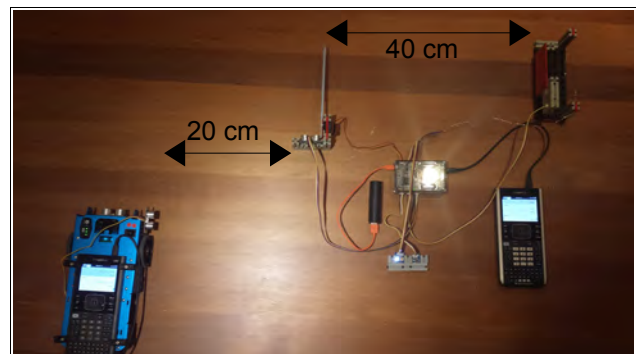
Der letzte Teil einer solchen Fahrt soll nun mit dem Rover modelliert werden. Da aber auch Funktionen des Hauses mit einbezogen werden, ist dafür noch ein zusätzlicher Innovator nötig.

### Aufgaben (2 Programme):

- Der Rover fährt mit Fahrlicht die Straße entlang.
- Er erkennt die Einfahrt zum Grundstück auf der rechten Seite in ca. 20 cm Entfernung und bremst (Bremslicht leuchtet auf).
- Er dreht sich nach rechts um 90° und blinkt dabei gelb.
- Dann fährt er langsam bis zum Tor und hält solange an, bis das Tor geöffnet wird.
- Er fährt weiter langsam durch das geöffnete Tor in den Carport in ca. 40 cm Entfernung (Fahrlicht) und hält dort endgültig an (Bremslicht; anschließend Licht aus).
- Am Tor wird erkannt, dass der Rover davor steht.
- Das Tor öffnet sich (Servo schwenkt die Schranke nach oben) und die RGB-LED auf dem Innovator leuchtet rot.
- Das Tor bleibt solange geöffnet, wie der Rover hindurchfährt und schließt sich hinter dem Rover wieder. Die rote LED erlischt.
- Der Bewegungsmelder (motion sensor) am Carport registriert die Bewegung beim „Aussteigen“ und schaltet die LED-Kette für den Weg zum „Haus“ an.
- Im Haus kann man mit den Taster (button) die Innenbeleuchtung an- und wieder ausschalten.

### Aufbau:

*Ausgangssituation.* Gut zu erkennen ist der vorne rechts angebrachte Ultraschallsensor am Rover, mit dem das „Tor“ erkannt wird. Am „Tor“ selbst befindet sich ein zweiter Ultraschallsensor, mit dem erkannt wird, ob der Rover vor dem „Tor“ steht. Er sollte hoch genug angebracht sein, damit der Rover auf ganzer Länge erkannt wird. Wegen des Servos muss der Innovator mit einem zusätzlichen Akku verbunden werden.



*Endsituation.* Der Rover parkt jetzt im „Carport“. Der Bewegungssensor und die LED-Lichterkette sind am Aufbau der „Carport“-Rückwand befestigt. Unter dem Innovator sieht man die Halterung für den Lichtschalter (Taster) und die leuchtende „Innenraum“-LED.



**Zusätzliches Material:**

- 2 Ultraschall-Entfernungssensoren (ranger)
- 1 Bewegungssensor (motion sensor)
- 1 Servo (nicht-kontinuierlich)
- 1 Taster (button)
- 1 weiße LED
- 1 kurze Lichterkette (selbst angefertigt) oder ersatzweise eine weitere weiße LED
- 1 Zusatzakku

Die Aufbauten (Tor, Rückwand) können aus LEGO o.ä. angefertigt werden.

**Programm 1 (Rover):**

```

Define home1()=
Prgm
:   DispAt 1,"Coming Home - Rover"
:   Send "CONNECT RV"
:   Send "CONNECT RANGER 1 TO IN 1"
:   Send "SET COLOR 255 255 255"
:   Send "RV FORWARD SPEED 0.14 M/S"
:   d:=1
:   While d>0.25
:       Send "READ RANGER 1"
:       Get d
:   EndWhile
:   Send "RV FORWARD SPEED 0.14 M/S"
:   Wait 1.5
:   Send "RV STOP "
:   Send "SET RV.COLOR 255 0 0"
:   Wait 0.3
:   Send "SET RV.COLOR 255 255 0 BLINK 4"
:   Send "SET RV.MOTORS LEFT -150 RIGHT 0"
:   Wait 2
:   Send "RV STOP "
:   Send "SET RV.MOTORS LEFT -100 RIGHT
100"
:   Send "SET RV.COLOR 255 255 255"
:   d:=1
:   While d>0.05
:       Send "READ RV.RANGER"
:       Get d
:   EndWhile
:   Send "SET RV.COLOR 255 0 0"
:   Send "RV STOP "
:   Wait 2
:   While d<0.05
:       Send "READ RV.RANGER"
:       Get d
:   EndWhile
:   Send "SET RV.COLOR 255 255 255"
:   Send "SET RV.MOTORS LEFT -100 RIGHT
100"
:   While d>0.1
:       Send "READ RV.RANGER"
:       Get d
:   EndWhile
:   Send "RV STOP "
:   Send "SET COLOR 255 0 0"
:   Wait 0.5
:   Send "SET COLOR 0 0 0"
:EndPrgm

```

Anzeige auf dem Display

Anschluss des Rovers

Anschluss des zusätzlichen Rangers an IN1

Anschalten weißes Fahrlicht

langsame Fahrt geradeaus

solange, bis seitlicher Abstand  $d < 0,25$  m

kurze Weiterfahrt, damit der Rover vor dem „Tor“ anhält

anhaltend, Bremslicht (rot) für 0,3 s

gelbes Blinklicht

Drehung nach rechts für 2 s (Zeit hängt ab von der Art des Untergrundes und der Gleitfähigkeit der Kugel im Rover)

langsame Fahrt geradeaus bei eingeschaltetem Fahrlicht

solange, bis der Rover 0,05 m vor dem Tor steht

Bremslicht einschalten und anhalten

2 s auf jeden Fall warten

warten bis das Tor auf ist

Fahrlicht einschalten und langsam in den „Carport“ fahren

solange fahren, bis der Rover 0,1 m vor der Rückwand steht

anhaltend und für 0,5 s Bremslicht anschalten

Licht ausschalten

**Programm 2 (Haus, Innovator):**

Überschrift

Anschlussbelegungen

LED 1 ist die Lichterkette

SERVO 1 muss an OUT 3 angeschlossen werden

setzt SERVO 1 in die Nullstellung

zentrale *while*-Schleife, Abbruch mit `esc`

Einlesen aller Sensoren

*Torsteuerung:*

Servo um 90° auslenken (Tor auf), wenn ein Objekt im Bereich &lt; 20 cm vor dem Tor erkannt wird. RGB-LED (Innovator) leuchtet rot.

Tor auf, solange ein Objekt erfasst wird  
*Wait 0.5* bewirkt, dass der Rover sicher durch das geöffnete Tor fahren kann

Servo wieder in Ausgangsstellung

LED aus

*Lichterkettensteuerung:*bei  $m=1$  (Bewegung) wird die Lichterkette eingeschaltet, sonst aus.*Innenbeleuchtung:* $b=2$  heißt, dass der Taster gedrückt wurde. Variable  $a$  zeigt an, ob die LED leuchtet ( $a=1$ ) oder nicht ( $a=0$ ). Bei  $a=1$  bewirkt  $b=2$ , dass die LED ausgeschaltet wird, bei  $a=0$ , dass sie eingeschaltet wird.

Ausschalten aller LED.

```

Define home2()=
Prgm
:   DispAt 1,"Home2"
:   Send "CONNECT RANGER 1 TO IN 1"
:   Send "CONNECT MOTION 1 TO IN 2"
:   Send "CONNECT BUTTON 1 TO IN 3"
:   Send "CONNECT LED 1 TO OUT 1"
:   Send "CONNECT LED 2 TO OUT 2"
:   Send "CONNECT SERVO 1 TO OUT 3"
:   Send "SET SERVO 1 0"
:   a:=0
:
:   While getKey()!="esc"
:
:       Send "READ RANGER 1"
:       Get d
:       Send "READ MOTION 1"
:       Get m
:       Send "READ BUTTON 1"
:       Get b
:
:       If d<0.2 Then
:           Wait 0.5
:           Send "SET SERVO 1 90"
:           Send "SET COLOR.RED TO ON"
:           While d<0.2
:               Send "READ RANGER 1"
:               Get d
:               Wait 0.5
:           EndWhile
:       Else
:           Send "SET SERVO 1 0"
:           Send "SET COLOR.RED TO OFF"
:       EndIf
:
:       If m=1 Then
:           Send "SET LED 2 TO ON"
:       Else
:           Send "SET LED 2 TO OFF"
:       EndIf
:
:       If b=2 and a=0 Then
:           Send "SET LED 1 TO ON"
:           a:=1
:           b:=0
:       EndIf
:       If b=2 and a=1 Then
:           Send "SET LED 1 TO OFF"
:           a:=0
:           b:=0
:       EndIf
:
:       EndWhile
:
:       Send "SET LED 1 TO OFF"
:       Send "SET LED 2 TO OFF"
:       Send "SET COLOR 0 0 0"
:EndPrgm

```



Dieses und weiteres Material steht Ihnen auf der TI Materialdatenbank zum Download bereit:  
**[www.ti-unterrichtsmaterialien.net](http://www.ti-unterrichtsmaterialien.net)**

© 2019 Texas Instruments

Dieses Werk wurde in der Absicht erarbeitet, Lehrerinnen und Lehrern geeignete Materialien für den Unterricht an die Hand zu geben. Die Anfertigung einer notwendigen Anzahl von Fotokopien für den Einsatz in der Klasse, einer Lehrerfortbildung oder einem Seminar ist daher gestattet. Hierbei ist auf das Copyright von Texas Instruments hinzuweisen. Jede Verwertung in anderen als den genannten oder den gesetzlich zugelassenen Fällen ist ohne schriftliche Genehmigung von Texas Instruments nicht zulässig. Alle Warenzeichen sind Eigentum ihrer Inhaber.