

Jürgen Enders

Der Rover als Fernlenkauto



Anwendungen mit TI-Innovator™ Hub und Rover

Einleitung

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. Bei den folgenden Beispielen wird TI Basic verwendet.

Kurze Einführung in das Programmieren mit TI Basic

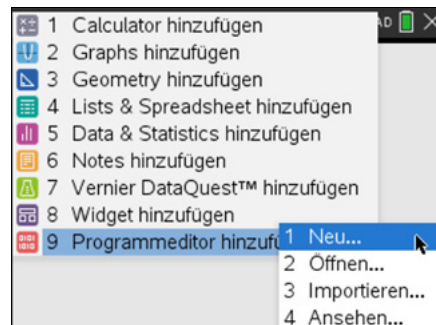
Dies ist keine vollständige Einführung in die Programmiersprache, sondern anhand eines Beispiels eine Zusammenfassung einiger Befehle, die zum Verständnis der Beispielprogramme sinnvoll sind.

AUFGABE:

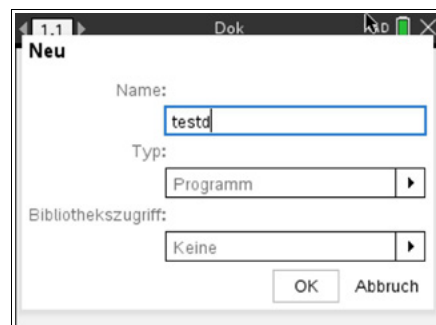
Es soll ein Programm geschrieben werden, das folgendes leistet:

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.

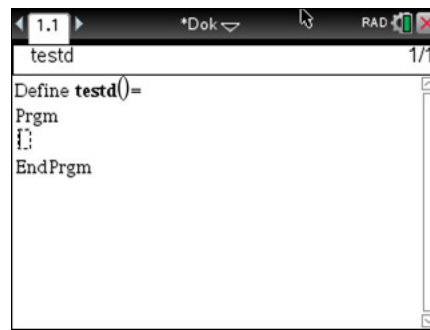
Soll ein neues Programm geschrieben werden, so fügt man eine neue Seite zu dem Dokument hinzu und wählt darin den Programmierer und die Auswahl `1:Neu ...`



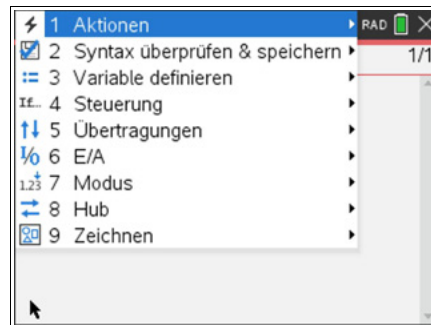
Es öffnet sich ein Fenster, in dem ein Name für das Programm eingegeben werden muss. Im Beispiel wurde der Name `testd` gewählt.



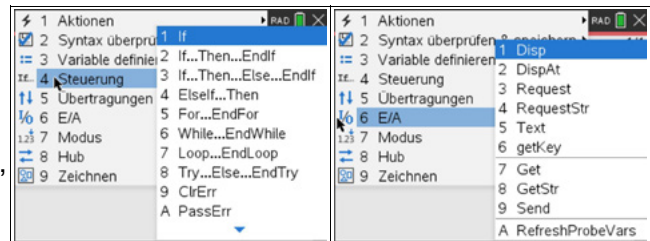
Schließt man das Fenster, so kommt man in den Editiermodus. Alle Programmierbefehle werden zeilenweise in den Bereich zwischen *Prgm* und *EndPrgm* eingefügt, wo sich jetzt das gestrichelte Rechteck befindet. Jeder Befehl kommt in eine neue Zeile, Leerzeilen werden später bei der Ausführung des Programmes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



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

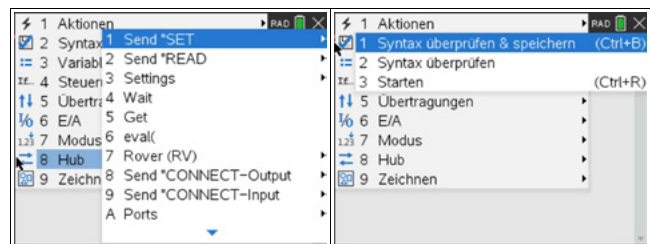


Das *Menü 4:Steuerung* enthält Befehle für Verzweigungen, Schleifen, usw.



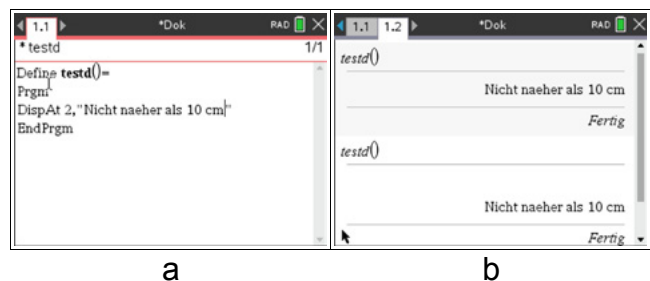
Das *Menü 6:E/A* enthält alle Befehle zur Kommunikation mit dem Nutzer (Anzeigebefehle, Eingabebefehle)

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



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*.



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

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).



```
* testd
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
```

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
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.

```

1.1 1.2 *testd 17/17
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. Menü 2: 1: Syntax überprüfen & speichern
2. Menü 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

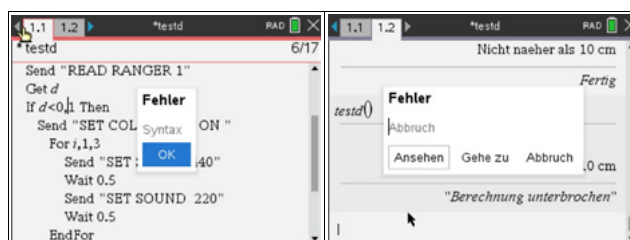
- auf dem Handheld durch Drücken von und mehrfach

- 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.

Beispiele für den ROVER

Der Rover als Fernlenkauto

Der Rover ist ein Fahrzeug, das von zwei getrennt ansteuerbaren Motoren angetrieben wird. Es ist so gebaut, dass es auf der Stelle wenden kann. Fest eingebaut sind ein Front - Entfernungssensor (Ranger), ein Farbsensor zur Spurverfolgung sowie eine RGB-LED auf der Oberseite und ein Stifthalter zur Aufnahme von Filzstiften. Der Rover benötigt zu seinem Betrieb einen TI-Innovator™, der in ein Fach an der Unterseite so eingeschoben wird, dass die Ports IN1 - IN3 sowie OUT1 - OUT3 zugänglich bleiben. Über einen Breadboard-Port-Stecker werden Rover und Innovator miteinander verbunden. Ferner benötigt der Rover noch einen Taschenrechner (TI-Nspire™CX (CAS) oder TI-84 Plus CE-T), der das Programm liefert und auf der Oberfläche des Rovers montiert werden kann. Ein Betrieb am PC ist nicht zu empfehlen, da er kabelgebunden erfolgen muss.

Der Rover enthält einen eigenen Akku, der etwa 6 1/2 h hält. In dieser Zeit legt der Rover ungefähr 3800 m mit der voreingestellten Standardgeschwindigkeit zurück.

Bei einem Fernsteuerauto werden die Steuerbefehle per Funk von einer portablen Konsole übertragen im Gegensatz zum Fernlenkauto, bei dem das Fahrzeug mit der ebenfalls portablen Konsole durch Kabel verbunden ist. In beiden Fällen gibt es einen Steuerhebel für den Motorlauf vorwärts/rückwärts und einen weiteren für ein Lenkservo links/rechts.

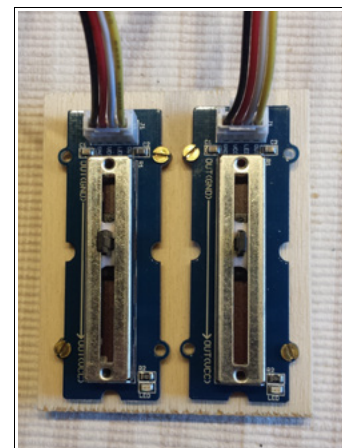
Der Räder des Rovers werden von zwei fest montierten Motoren angetrieben, so dass zum Fernlenken ein anderes Konzept gewählt werden muss. Gut geeignet sind dafür Potentiometer, mit denen jeder einzelne Motor angesteuert wird. Durch die unterschiedlichen Drehzahlen und Drehrichtungen der Motoren sind dabei alle Fahrmanöver möglich.

Aufgabe:

Es ist ein Programm zu schreiben, mit dem der Rover ferngelenkt werden kann.

Hinweise:

1. Im Beispiel werden Schiebepotentiometer 10 kΩ an den Eingängen IN 1 und IN 2 des Innovators verwendet. Ihr Wertebereich geht von 0 bis ca. 15500. Für die Motorsteuerung muss dieser Bereich dann in den Bereich von -256 bis 256 umgerechnet werden, wobei im Beispiel diese Grenzen nicht ganz erreicht werden, um ein fehlerfreies Fahren zu ermöglichen.
2. Es ist praktisch, wenn man die Potentiometer auf einer kleinen Platte montiert, da man die Steuerung dann leichter bedienen kann.
3. Soll der Rover auf dem Fußboden fahren, so muss man die Sensorkabel verlängern. Dabei reicht es aus, zwei Kabel zu je 50 cm miteinander zu verbinden, indem man jeweils einen Stecker entfernt und die abisolierten Kabelenden farbrichtig mit Lüsterklemmen zusammenführt oder verlötet und isoliert.



Programm:

Zuordnen der Potentiometer zu den Eingängen
IN 1 und IN 2

Anschluss des Rovers

zentrale *While*-Schleife, Abbruch mit 

Einlesen der Werte *p1* und *p2* der
Potentiometer

Umrechnen in den Motorwertebereich

Das Minuszeichen bei *vr* sorgt für die richtige
Drehrichtung des Motors.

Anzeige der Motorwertebereich

Übertragung von *v1* und *vr* an die Motoren

Anhalten des Rovers

Trennung des Rovers

```

Define fernlenker()=
Prgm
:Send "CONNECT POTENTIOMETER 1 TO IN1"
:Send "CONNECT POTENTIOMETER 2 TO IN2"
:Send "CONNECT RV"
:DispAt 1,"Fernlenkauto"
:While getKey()!="esc"
:   Send "READ POTENTIOMETER 1"
:   Get p1
:   Send "READ POTENTIOMETER 2"
:   Get p2
:   v1:=(((p1)/(15500))-0.5)*450
:   vr:=-(((p2)/(15500))-0.5)*450
:   DispAt 2,v1
:   DispAt 3,vr
:   Send "SET RV.MOTORS  LEFT eval(v1)
RIGHT eval(vr) "
:EndWhile
:Send "RV STOP "
:Send "DISCONNECT RV"
:EndPrgm

```

Haben Sie Fragen zu Produkten von Texas Instruments? Oder sind Sie an weiteren Unterrichtsmaterialien oder einer Lehrerfortbildung interessiert? Gerne steht Ihnen auch unser Customer Service Center mit Rat und Tat zu Seite. Nehmen Sie mit uns Kontakt auf:



Customer Service Center
TEXAS INSTRUMENTS
education.ti.com/csc

education.ti.com/deutschland
education.ti.com/oesterreich
education.ti.com/schweiz

Weitere Materialien finden Sie unter:
www.ti-unterrichtsmaterialien.net

Dieses und weiteres Material steht Ihnen auf der TI Materialdatenbank zum Download bereit:
www.ti-unterrichtsmaterialien.net

© 2020 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.