

Jürgen Enders

Der KFZ-Blinker



Anwendung mit dem TI-Innovator™ Hub

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

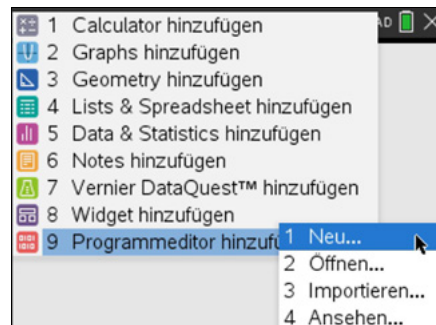
Dies ist keine vollständige Einführung in die Programmiersprache, sondern anhand eines Beispiels eine Zusammenfassung einiger Befehle, die zum Verständnis der Beispielprogramme sinnvoll sind.

AUFGABE:

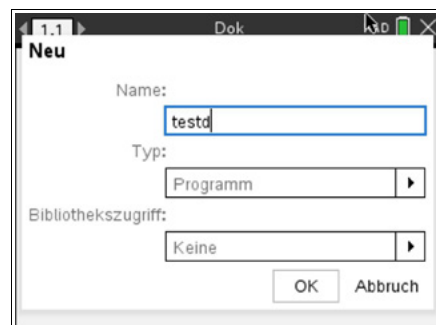
Es soll ein Programm geschrieben werden, das folgendes leistet:

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.

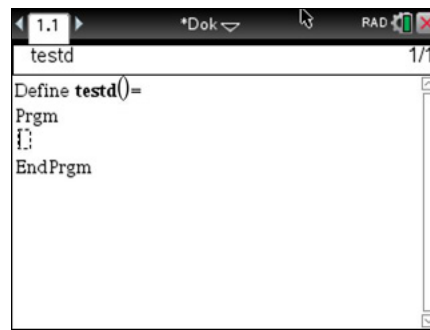
Soll ein neues Programm geschrieben werden, so fügt man eine neue Seite zu dem Dokument hinzu und wählt darin den Programmeditor und die Auswahl `1:Neu ...`



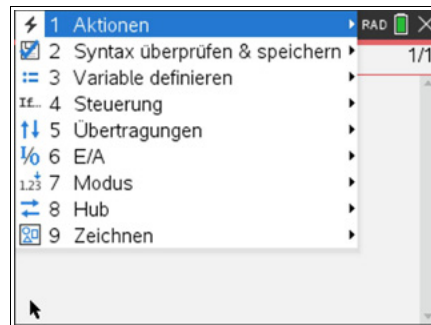
Es öffnet sich ein Fenster, in dem ein Name für das Programm eingegeben werden muss. Im Beispiel wurde der Name `testd` gewählt.



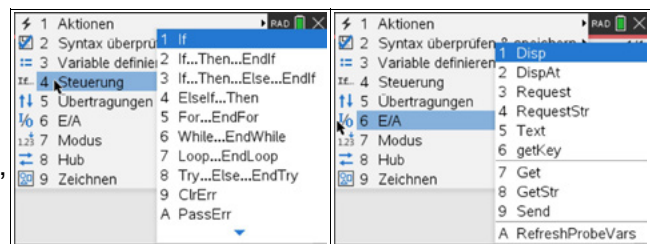
Schließt man das Fenster, so kommt man in den Editiermodus. Alle Programmierbefehle werden zeilenweise in den Bereich zwischen *Prgm* und *EndPrgm* eingefügt, wo sich jetzt das gestrichelte Rechteck befindet. Jeder Befehl kommt in eine neue Zeile, Leerzeilen werden später bei der Ausführung des Programmes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



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

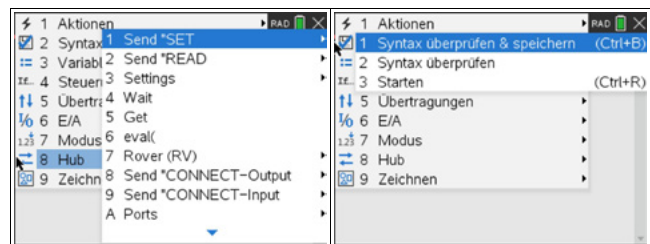


Das *Menü 4:Steuerung* enthält Befehle für Verzweigungen, Schleifen, usw.



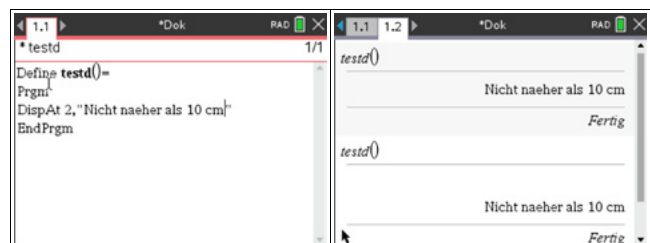
Das *Menü 6:E/A* enthält alle Befehle zur Kommunikation mit dem Nutzer (Anzeigebefehle, Eingabebefehle)

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*.

```

1.1 1.2 *Dok RAD
* 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

```

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).

```

1.1 1.2 *testd RAD
* 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

```

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!

```

1.1 1.2 *testd RAD
* 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.

```

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

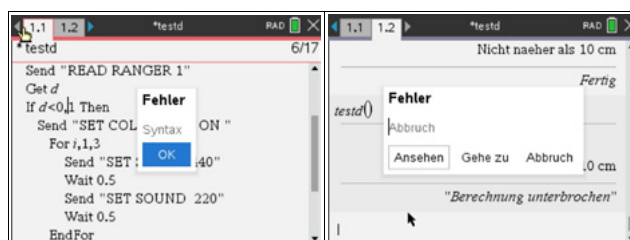
- auf dem Handheld durch Drücken von und mehrfach

- 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.

Beispiel: Der KFZ-Blinker

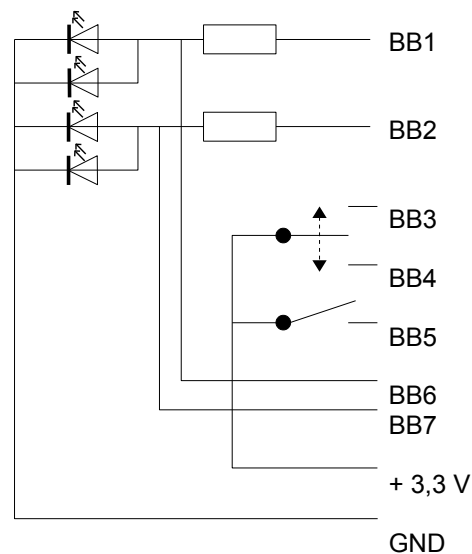
Richtungsänderungen bei KFZ oder Motorrädern werden üblicherweise durch orange blinkende Lampen angezeigt, die vorne und hinten angebracht sind. Ihre Blinkfrequenz ist festgelegt. Im Fahrzeug zeigen Kontrolllampen den eingeschalteten Blinker an; zusätzlich ertönt ein Klackgeräusch, das in der Anfangszeit durch das Blinkrelais erzeugt und bei modernen Autos als zusätzliches akustisches Signal beibehalten wurde. Geht eine Lampe kaputt, so wird dies üblicherweise angezeigt; gleichzeitig erhöht sich aber auch die Blinkfrequenz. Überdies dienen die Lampen auch als blinkende Warnleuchten im Pannenfall.

Aufgabe: Die im Eingangstext beschriebene Blinkanlage soll nachgebaut werden.

Material: 4 rote LED
2 Vorwiderstände 100Ω
1 einpoliger Schalter
1 einpoliger Umschalter mit mittiger Ruhestellung als Blinkhebel
Bis auf den Umschalter sind alle Teile im Breadboard-Pack enthalten. Der Umschalter findet sich vielleicht in einer Bastelkiste oder in der Physik-Sammlung.

Hilfen:

Schaltbild:



Programm:

Variablendeklaration

Überschrift und Abbruchbedingung

Portbelegung:

BB1, BB2 : Blinker links/rechts

BB3, BB4 : Blinkhebel

BB5 : Warnblinker

BB6, BB7 : Lampenausfall links/rechts

Variablenbelegung:

x : Abbruchkriterium

t,s : Lampenausfall

```

Define richtung()=
Prgm
:Local x,t,s,r,l,w
:Disp "Blinkeranlage"
:Disp "Ende mit <esc>"
:Send "CONNECT LED 1 TO BB 1"
:Send "CONNECT LED 2 TO BB 2"
:Send "CONNECT SWITCH 1 TO BB 3"
:Send "CONNECT SWITCH 2 TO BB 4"
:Send "CONNECT SWITCH 3 TO BB 5"
:Send "CONNECT ANALOG.IN 1 TO BB 6"
:Send "CONNECT ANALOG.IN 2 TO BB 7"
:1→x
:1→t
:1→s

```

Anfang der zentralen <i>While</i> -Schleife	:While x#"esc"
Variablenbelegung: 0 heißt keine Blinktätigkeit	: getKey()→x : 0→r : 0→l : 0→w
Abfrage der Schalterstellungen und entsprechende Variablenzuweisungen 1 heißt: Schalter betätigt	: Send "READ SWITCH 1" : Get r : Send "READ SWITCH 2" : Get l : Send "READ SWITCH 3" : Get w
Warnblinkmodus: alle 4 LED gehen für 0,5 s an und für 0,6 s aus, ein kurzer Ton von 5000 Hz ertönt.	: If w=1 Then : Send "SET LED 1 ON" : Send "SET LED 2 ON " : Send "SET SOUND 5000 TIME . 001"
verhindert bei eingeschaltetem Richtungsblinker ein einseitiges Blinken im Warnblinkmodus	: Wait 0.5 : Send "SET LED 1 OFF" : Send "SET LED 2 OFF" : Wait 0.6 : 0→r : 0→l
Rechtsblinker gesetzt, nur eine Seite blinkt, Zeiten und Geräusch bei heilen LEDs wie im Warnblinkmodus	: ElseIf r=1 Then : Send "SET LED 1 ON" : Send "SET SOUND 5000 TIME . 001"
Fehlermodus: Abfrage der Spannung an den Lampen und Speichern des Wertes in der Variablen a.	: Send "READ ANALOG.IN 1" : Get a : If a>8700 Then : 3→t : EndIf : If a>100 and a<8700 Then : 1→t : EndIf
Fällt eine der LEDs aus, so erhöht sich die Spannung an BB6 bzw. BB7. Der Teiler t wird von 1 auf 3 gesetzt und erhöht dadurch den Blinkrhythmus. Bei intakten LEDs wird t wieder auf 1 zurückgesetzt. Die „and“-Verknüpfung sorgt dafür, dass Fehlmessungen keine Rolle spielen.	: Wait ((0.5)/(t)) : Send "SET LED 1 OFF" : Wait ((0.6)/(t))
Linksblinker gesetzt, sonst wie oben	: ElseIf l=1 Then : Send "SET LED 2 ON" : Send "SET SOUND 5000 TIME . 001"
Ende der zentralen <i>While</i> -Schleife Programmende	: Send "READ ANALOG.IN 2" : Get a : If a>8700 Then : 3→s : EndIf : If a>100 and a<8700 Then : 1→s : EndIf : Wait ((0.5)/(s)) : Send "SET LED 2 OFF" : Wait ((0.6)/(s)) : EndIf :EndWhile :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.