Jürgen Enders

Beleuchtungsschaltungen

Anwendung mit dem TI-Innovator™ Hub

Texas Instruments

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.

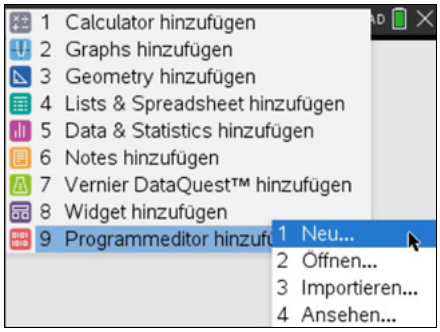
Kurze Einführung in das Programmieren mit TI Basic

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

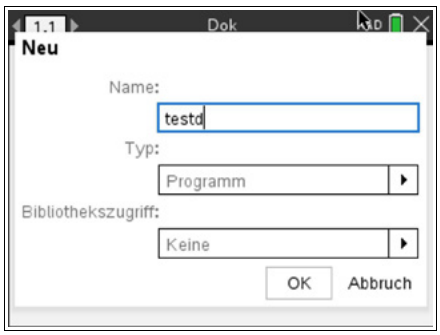
AUFGABE:

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.

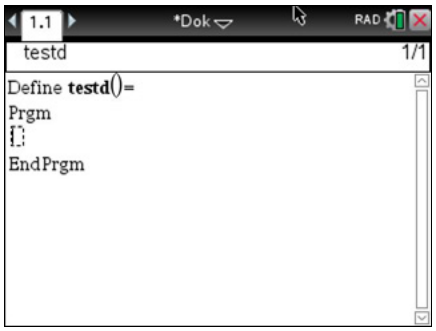
Soll ein neues Programm geschrieben werden, so fügt man eine neue Seite zu dem Dokument hinzu und wählt darin den Programmeditor und die Auswahl *1:Neu …*



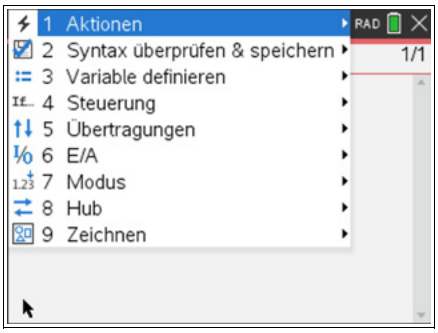
Es öffnet sich ein Fenster, in dem ein Name für das Programm eingegeben werden muss. Im Beispiel wurde der Name **testd** gewählt.



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.

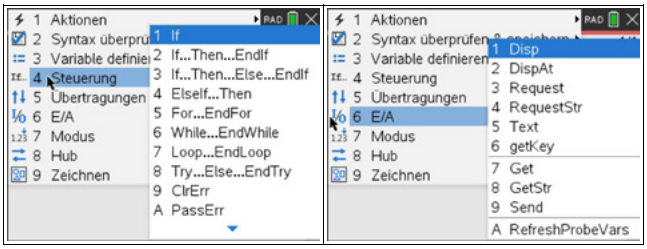


Im *Menü* befinden sich in Gruppen zusammengefasst alle Programmierbefehle.



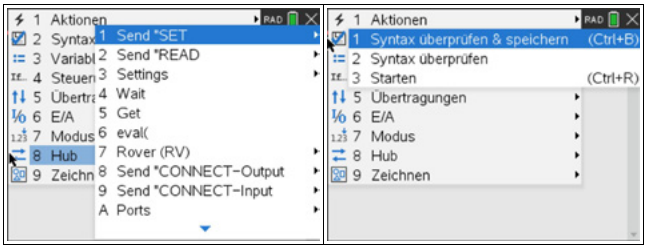
Das *Menü 4:Steuerung* enthält Befehle für Verzweigungen, Schleifen, usw.

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

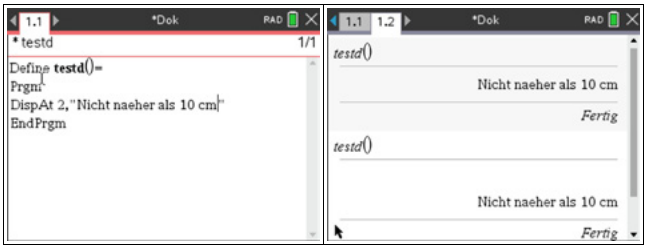


Das *Menü 8:Hub* enthält alle Befehle zur Kommunikation mit dem TI-Innovator™ Hub.

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

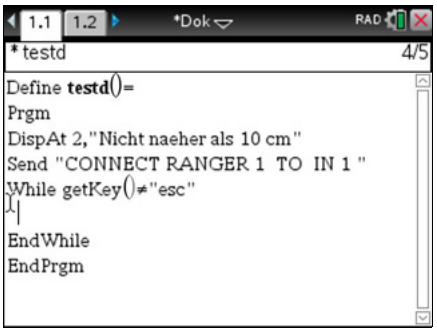


a. Die Überschrift - der Befehl *DispAt* aus dem *Menü E/A* bewirkt, dass der in Anführungszeichen stehende Text *immer* in der 2. Zeile unter dem Trennstrich auf dem Home-Display dargestellt wird. *Disp* allein schreibt den Text immer in eine neue Zeile.
b. Zur Verdeutlichung:
DispAt 1,... oben
DispAt 2,... unten

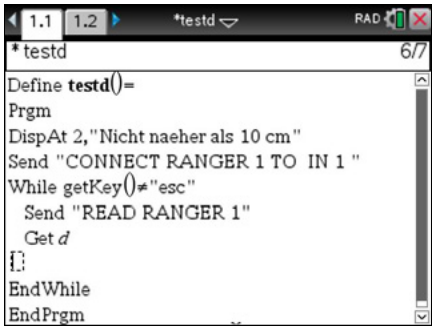


a b

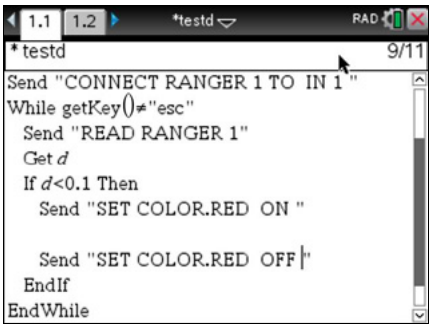
Der Befehl *Send „CONNECT RANGER 1 TO IN 1“* bewirkt die Zuordnung des RANGER 1 zum Eingang IN 1. Die Nummer bei RANGER gehört zwingend dazu und muss per Hand eingefügt werden.
Die Befehle finden sich im *Hub-Menü* unter *Send “CONNECT - Input, Settings und Ports.*
Man kann den ganzen Text in den Anführungszeichen 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*.

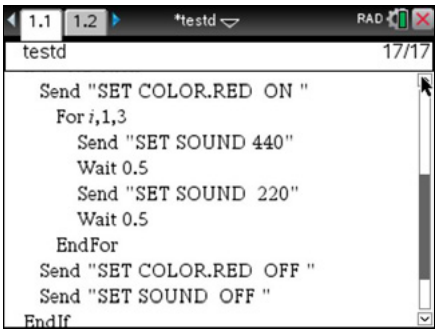


Innerhalb der *While*-Schleife wird durch den Befehl *Send “READ RANGER 1“* aus dem Menü *Send “READ* fortlaufend die Entfernung gemessen (in der Maßeinheit m) und in einem Zwischenspeicher auf dem Hub abgelegt. Bei der nächsten Messung würde der Wert sofort überschrieben werden; deshalb muss er vorher 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!





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.

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 enter wird das Programm gestartet.

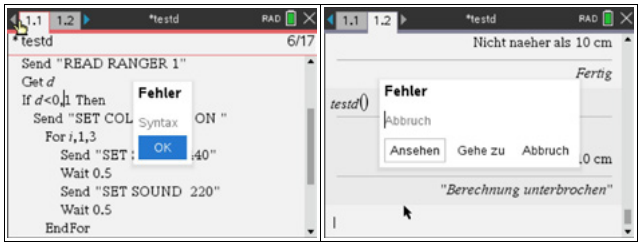
Abbrechen eines Programmes:

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

- auf dem Handheld durch Drücken von on und mehrfach enter

- 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: Beleuchtungsschaltungen

a. Der Näherungsschalter

Näherungsschalter arbeiten meist mit einem IR-Sensor, der bei Annäherung eines Objektes (das kann auch ein Tier oder ein größeres Blatt sein) eine Lampe einschaltet. Beim Beispielprogramm wird der IR-Sensor durch den Ultraschallsensor RANGER ersetzt; als Beleuchtung dient die eingebaute RGB-LED.

Anzeige von erläuterndem Text
Verbindung von RANGER 1 durch CONNECT mit dem Eingang IN 1
Zentrale *While*-Schleife: Abbruch mit esc
Einlesen der Entfernung
Abspeichern in der Variablen a
Einschalten der Beleuchtung für 3 s falls a < 1m
COLOR 255 255 255 erzeugt weißes Licht

Ausschalten der Beleuchtung mit COLOR 0 0 0

```
Define belaussen()=
Prgm
:Disp "Aussenlicht"
:Disp "Ende mit <esc>"
:Send "CONNECT RANGER 1 TO IN 1"
:While getKey()≠"esc"
:   Send "READ RANGER 1"
:   Get a
:   If a<1 Then
:      Send "SET COLOR 255 255 255"
:      Wait 3
:   EndIf
:   Send "SET COLOR 0 0 0"
:EndWhile
:EndPrgm
```

Verwendet man zusätzlich ein Relais, so kann man auch eine „richtige“ Beleuchtung damit ein- und ausschalten. Die drei zusätzlichen Programmzeilen sind fett gedruckt. Mit diesen Programmzeilen lassen sich auch alle anderen Beleuchtungsprogramme ergänzen.

```
Define belaussen()=
Prgm
:Disp "Aussenlicht"
:Disp "Ende mit <esc>"
:Send "CONNECT RANGER 1 TO IN 1"
:Send "CONNECT RELAY 1 TO OUT 1"
:While getKey()≠"esc"
:   Send "READ RANGER 1"
:   Get a
:   If a<1 Then
:      Send "SET COLOR 255 255 255"
:      Send "SET RELAY 1 ON"
:      Wait 3
:   EndIf
:   Send "SET COLOR 0 0 0"
:   Send "SET RELAY 1 OFF"
:EndWhile
:EndPrgm
```

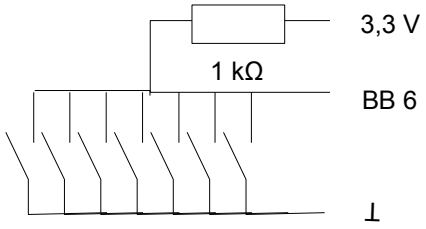
b. Dämmerungsschalter

Dämmerungsschalter werden überall da eingesetzt, wo eine Beleuchtung in Abhängigkeit von der Umgebungshelligkeit eingeschaltet werden soll. Im Beispiel wird der eingebaute Helligkeitssensor BRIGHTNESS verwendet. Er muss nicht durch einen CONNECT-Befehl mit einem Eingang verbunden werden. Dunkelt man ihn ab, so wird ab dem Wert a = 1 der Variablen die Beleuchtung eingeschaltet; wird es wieder heller, wird sie wieder ausgeschaltet.

```
Define beldunkel()=
Prgm
:Disp "Dämmerlicht"
:Disp "Ende mit <esc>"
:While getKey()≠"esc"
:   Send "READ BRIGHTNESS "
:   Get a
:   If a<1 Then
:      Send "SET COLOR 255 255 255"
:   EndIf
:EndWhile
:Send "SET COLOR 0 0 0"
:EndPrgm
```

c. Treppenhauslicht

Bei einem Treppenhauslicht kann man über viele Schalter das Licht einschalten, das dann für eine gewisse Zeit eingeschaltet bleibt. Dazu sind alle Schalter parallel geschaltet.
Da ein ANALOG-Eingang erforderlich ist, muss der BREADBOARD-Anschluss 6 verwendet werden:
CONNECT ANALOG.IN 1 TO BB 6
Über den 1 kΩ - Widerstand ist er mit +3,3 V verbunden.
Drückt man nun auf einen Schalter, so sinkt das Potential auf GND und die Beleuchtung wird für 3 s eingeschaltet. Die überdimensionierte Bedingung a < 100 sorgt dafür, dass die Schaltung sicher ausgeführt wird.




```
Define beltreppe()=
Prgm
:Disp "Treppenhaus"
:Disp "Ende mit <esc>"
:Send "CONNECT ANALOG.IN 1 TO BB 6"
:While getKey()≠"esc"
:   Send "READ ANALOG.IN 1"
:   Get a
:   If a<100 Then
:      Send "SET COLOR 255 255 255"
:      Wait 3
:   EndIf
:   Send "SET COLOR 0 0 0"
:EndWhile
:EndPrgm
```


d. Raumbeleuchtung

Bei großen Räumen oder langen Fluren ist es sinnvoll, die Raumbeleuchtung von verschiedenen Stellen ein- und auch wieder ausschalten zu können. Die Beschaltung von BB 6 ist dabei dieselbe wie im vorherigen Beispiel.
Variable b: b = 0 Licht aus (Anfangszustand)
b = 1 Licht ein

Einschalten, wenn b = 0 und a < 100
b = 1 , da Licht eingeschaltet
a = 1000 verhindert sofortiges Ausschalten ohne erneutes Einlesen von a
0,2 s Wartezeit, um die Zeit des Schalterprellens zu überbrücken
Ausschalten, wenn b = 1 und a < 100
b = 0 , da Licht ausgeschaltet
a = 1000 verhindert sofortiges Einschalten
Wartezeit 0,2 s

```
Define belraum()=
Prgm
:Disp "Raumlicht"
:Disp "Ende mit <esc>"
:Send "CONNECT ANALOG.IN 1 TO BB 6"
:b:=0
:While getKey()≠"esc"
:   Send "READ ANALOG.IN 1"
:   Get a
:   If a<100 and b=0 Then
:      Send "SET COLOR 255 255 255"
:      b:=1
:      a:=1000
:   EndIf
:   Wait 0.2
:   If a<100 and b=1 Then
:      Send "SET COLOR 0 0 0"
:      b:=0
:      a:=1000
:   EndIf
:   Wait 0.2
: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