Overzicht:

In deze STEM-les leer je hoe je een digitale invoerpin op de breadboard-connector van de TI-Innovator ™ -hub uitleest. De digitale invoerpin detecteert de aan- of uitstand van een tuimelschakelaar die wordt gebruikt om de stroom naar een LED-circuit in of uit te schakelen. Er wordt een programma geschreven om de status van de digitale invoerpin te lezen en vervolgens, op basis van de status van de pin, een passend bericht weer te geven.

Doelen:

1. Bouw een LED-circuit met een schakelaar (stroom) op een breadboard.
2. Sluit een digitale invoerpin op de TI-Innovator Hub aan op de circuitschakelaar om aan te geven of de LED aan of uit is.
3. Maak een TI Python-programma op een TI-84 Plus CE-T Python Edition dat een geschikt bericht weergeeft op basis van de stand van de aan/uit-schakelaar.

Achtergrond:

Als de BB1 digitale ingangspin is verbonden met de stroomschakelaar van een LED-circuit wordt er een digitaal ingangssignaal naar de TI-Innovator Hub gestuurd.

Met de schakelaar kan de gebruiker schakelen tussen een open circuit (uit) en een gesloten circuit (aan). Een schakelaar in de UIT-stand creëert een open circuit en de LED gaat niet aan omdat elektronen niet door het circuit kunnen bewegen. Een schakelaar in de AAN-status creëert een gesloten circuit en laat elektronen stromen, waardoor de LED wordt ingeschakeld.

Met behulp van TI Python kan de invoerpen net als een variabele worden gelezen. Wanneer 3,3 volt is aangesloten op de digitale ingangspen, zal de variabele waarde als 1 worden weergegeven. Als 0 Volt (aarde) op de pin is aangesloten, zal de waarde 0 zijn. Een "als-dan" –instructie (If …then …) kan worden gebruikt om het programma te laten reageren op basis van de pin-status.

Een echte toepassing van een digitale schakelaar is een knop op een game-controlepad die is geprogrammeerd om een geanimeerd personage in een game te laten springen wanneer op de knop wordt gedrukt.

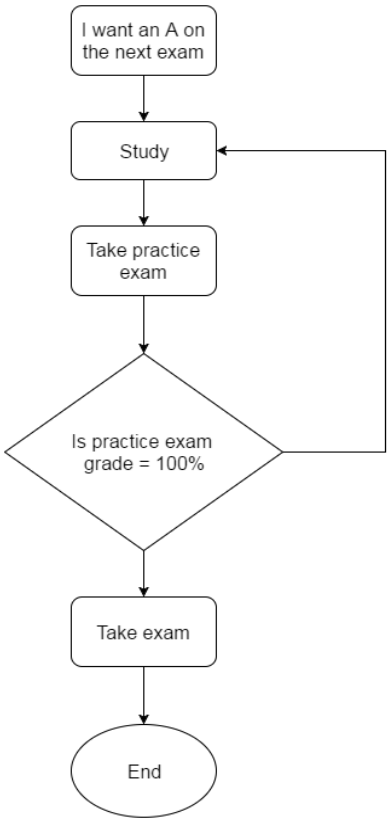
Stroomdiagrammen worden vaak gebruikt om computerprogramma's te maken of te ontwerpen.

Een algoritme is een reeks stappen die moet worden gevolgd om een taak te voltooien en wordt gebruikt om de volgorde van een computerprogramma te ordenen. Ze zijn niet in een specifieke computertaal geschreven, zodat iedereen ze kan begrijpen. Computeralgoritmen worden vaak geïllustreerd met stroomdiagrammen die rechthoeken, cirkels en ruiten bevatten. Rechthoeken zijn programma-instructies, terwijl diamanten "als-dan" beslissingen zijn en cirkels het einde van een programma vormen. Pijlen worden in het stroomdiagram gebruikt om te laten zien hoe het programma van begin tot eind stroomt.

Met behulp van deze stroomdiagramsymbolen kan een programmeur een computerprogramma beschrijven op een manier die vergelijkbaar is met hoe een voetbaltrainer een spel beschrijft met kruisjes en rondjes om spelers aan beide kanten te vertegenwoordigen en pijlen die individuele spelersbewegingen tijdens het spel tonen.

Oefenen:

Taak: Maak een stroomschema dat aangeeft hoe je voor een toets leert. Tip: waarschijnlijk controleer je tijdens het leren of je alles al kent. Zo niet, dan leer je nog een keer en controleer je dat na afloop weer. Zodra je het wel goed kent stopt je voorbereiding en ben je klaar voor de toets.



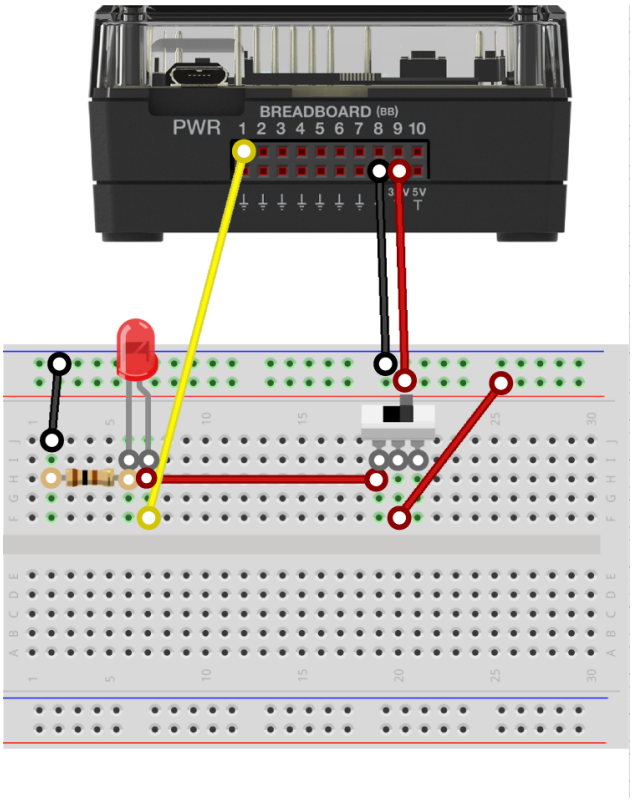
Materialen en gereedschappen:

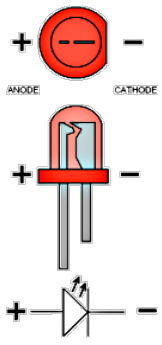
- TI-84 Plus CE-T Python Edition
- TI-Innovator™ Hub met USB-kabel
- Punttang (optioneel)
- Draadtang (optioneel)
- TI-Innovator Breadboard Pack:
  1. Breadboard
  2. Man-man verbindingskabels
  3. Weerstand 100 Ohm (bruin, zwart, bruin)
  4. Schuifschakelaar
  5. Rode LED

Bouw de hardware:

Monteer het circuit zoals in het diagram aan de rechterkant. Volg hiervoor deze stappen:

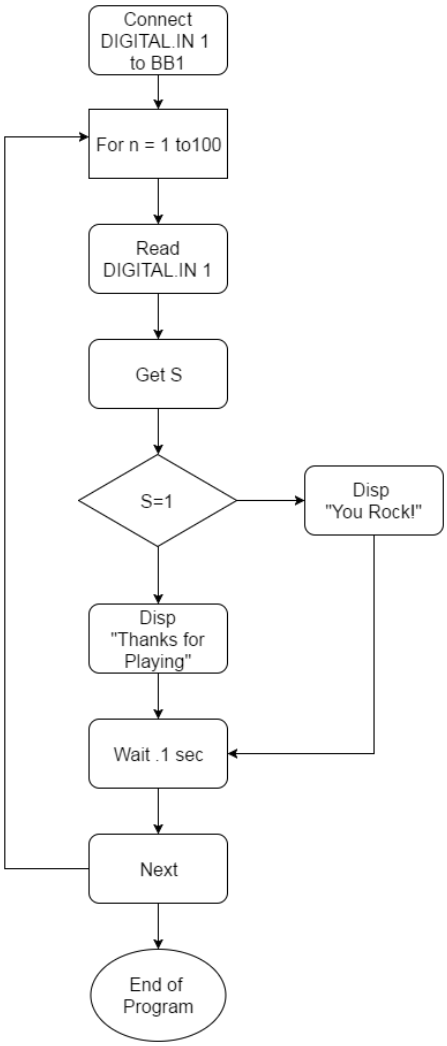
1. Plaats een schuifschakelaar en een rode LED op het breadboard op de locaties die zijn aangegeven in de afbeelding rechts.
2. Sluit een rode man-man verbindingskabel aan vanaf de 3.3V op de TI-Innovator Hub naar de rode 3.3V-bus op het breadboard.
3. Sluit een zwarte man-man verbindingskabel aan vanaf een willekeurige aardepin op de TI-Innovator Hub naar de blauwe aarde-bus op het breadboard.
4. Sluit een rode man-man verbindingskabel aan vanaf de middelste poot van de schuifschakelaar naar de rode 3.3V-bus op het breadboard.
5. Gebruik een rode man-man verbindingskabel om een buitenste poot van de schakelaar aan te sluiten op de anode (+) van de rode LED.
6. Sluit een 100 Ohm weerstand aan op de kathode (-) poot van de rode LED.
7. Gebruik een zwarte man-man verbindingskabel om het tegenoverliggende uiteinde van de weerstand van 100 ohm aan te sluiten op de blauwe aarde-bus op het breadboard.
8. Gebruik een gele man-man verbindingskabel om de anode (+) -poot van de rode LED te verbinden met BB1 op de TI-Innovator Hub. Dit punt wordt gehouden op 3,3 V wanneer de schuifschakelaar is ingeschakeld en wordt op 0V gehouden als hij is uitgeschakeld.
    - De digitale ingang op BB1 kan deze aan- of uit-status detecteren.
9. Steek het "B" -uiteinde van de mini-USB-kabel in de TI-Innovator Hub en het "A" -uiteinde in de TI-84 Plus CE-T Python Edition.





Schrijf de software voor de TI-84 Plus CE-T Python Edition:

Taak: Schrijf een TI Python-programma dat een rode LED zal lezen die is verbonden met een schakelaar en pin 1 op de breadboard-connector van de TI-Innovator Hub. Als de schakelaar uit staat, 0, wordt "De LED staat uit." Weergegeven. Als de schakelaar aan staat, 1, geeft deze "De LED staat aan."



Code for the TI-84 Plus CE-T Python Edition:

```
from ti_hub import *
from digital import *
switch=digital("BB 1")
for n in range(50):
    s=switch.measurement()
    if s==1:
        print("De LED staat aan.")
    else:
        print("De LED staat uit.")
    sleep(0.1)
```

Extra voor Experts:

Achtergrond: Booleaanse algebra is de tak van de wiskunde die zich bezighoudt met logica. Variabelen zijn WAAR of NIET WAAR. Boolean-bewerkingen zijn woorden als "en", "of" en "niet". In digitale elektronica en informatica worden WAAR en AAN weergegeven met een 1, terwijl ONWAAR en UIT worden weergegeven met een 0. Omdat computercircuits transistoren gebruiken die AAN of UIT zijn, die digitaal zijn, en software énen en nullen gebruikt, dat is binair, is Booleaanse algebra handig om logische beslissingen te nemen, zoals de bewegingen van een stuk in een computerspel.

Taak: Voeg een extra schuifschakelaar toe aan BB2 als een tweede digitale ingang. Schrijf een programma dat de toestanden van beide switches leest en weergeeft en vervolgens op basis van die toestanden voor alle Booleaanse bewerkingen in TI BASIC (**and**, **or**, **xor**) aangeeft of die waar zijn of niet.

- **And** = beide voorwaarden zijn hetzelfde
- **Or** = minstens één van beide is waar, allebei mag dus ook
- **Xor** = precies één ven beide is waar, de andere dus niet

Hint: de Booleaanse operaties zijn te vinden onder TEST en dan LOGIC.

Waarheidstabel voor Digitale Input van BB1 en BB2				
BB 1	BB 2	BB1 **and** BB2	BB1 **or** BB2	BB1 **xor** BB2
AAN (1)	AAN (1)	WAAR	WAAR	ONWAAR
AAN (1)	UIT (0)	ONWAAR	WAAR	WAAR
UIT (0)	AAN (1)	ONWAAR	WAAR	WAAR
UIT (0)	UIT (0)	ONWAAR	ONWAAR	ONWAAR

Waarheidstabel voor Digitale Input BB1	
BB1	NIET(BB1)
AAN (1)	0
UIT(0)	1

Extra voor Experts programma:

```
from ti_hub import *
from digital import *
switch1=digital("BB 1")
switch2=digital("BB 2")
for n in range(50):
    a=switch1.measurement()
    b=switch2.measurement()
    if a==1:
        a=True
    else:
        a=False
    if b==1:
        b=True
    else:
        b=False
    print(" ")
    print("a is {}".format(a))
    print("b is {}".format(b))
    if a and b:
        print("a AND b is True")
    else:
        print("a AND b is False")
        print("b is {}".format(b))
    if a or b:
        print("a OR b is True")
    else:
        print("a OR b is False")
        print("b is {}".format(b))
    if a ^ b:
        print("a XOR b is True")
    else:
        print("a XOR b is False")
    print("NOT(a) is {}".format(not(a)))
    print("NOT(b) is {}".format(not(b)))
    sleep(5)
```

Python logische operatoren

- and – or – not

```
x=3
print(x>2 and x<5 )
print(x<2 or x>5 )
print(not(x>5))
```

```
>>>#Running test.py
>>>from test import *
True
False
True
```

- & – ^ – ~

```
x=3
# AND
print(x>2 & x<5 )
# OR
print(x<2 | x>5 )
# XOR
print(x<2 & x>5 )
```

```
>>>#Running test.py
>>>from test import *
True
False
False
```