PROJECT PYTHON – RASPBERRY PI

INHOUDSOPGAVE

pre-inleiding 2

Betekenis GPIO – LED – Physical Computing 3

Python IDLE3 openen op de Raspberry Pi 3

Inleiding 4

Stap 1b. Een led aan- of uitzetten – ZONDER WEERSTAND 4

Opstelling maken: 4

Stap 1a. Een led aan- of uitzetten – met weerstand 5

Opstelling maken: 5

Code: LED aan en uit 6

Code: De gebruiker bepaalt of de lamp aan of uit moet 7

OEFENING 1: 7

OEFENING 2: 7

STAP 2: Werken met 1 knop 8

Knop aankoppelen 8

Code: “luisteren” naar de knop 9

OEFENING 3 9

Code: knop bedient LED 10

Oefening 4 10

STAP 3: Werken met 2 knoppen en 2 LEDS 11

Tweede knop aankoppelen 11

OEFENING 5 11

Code: Eerste knop haalt het 12

OEFENING 6 13

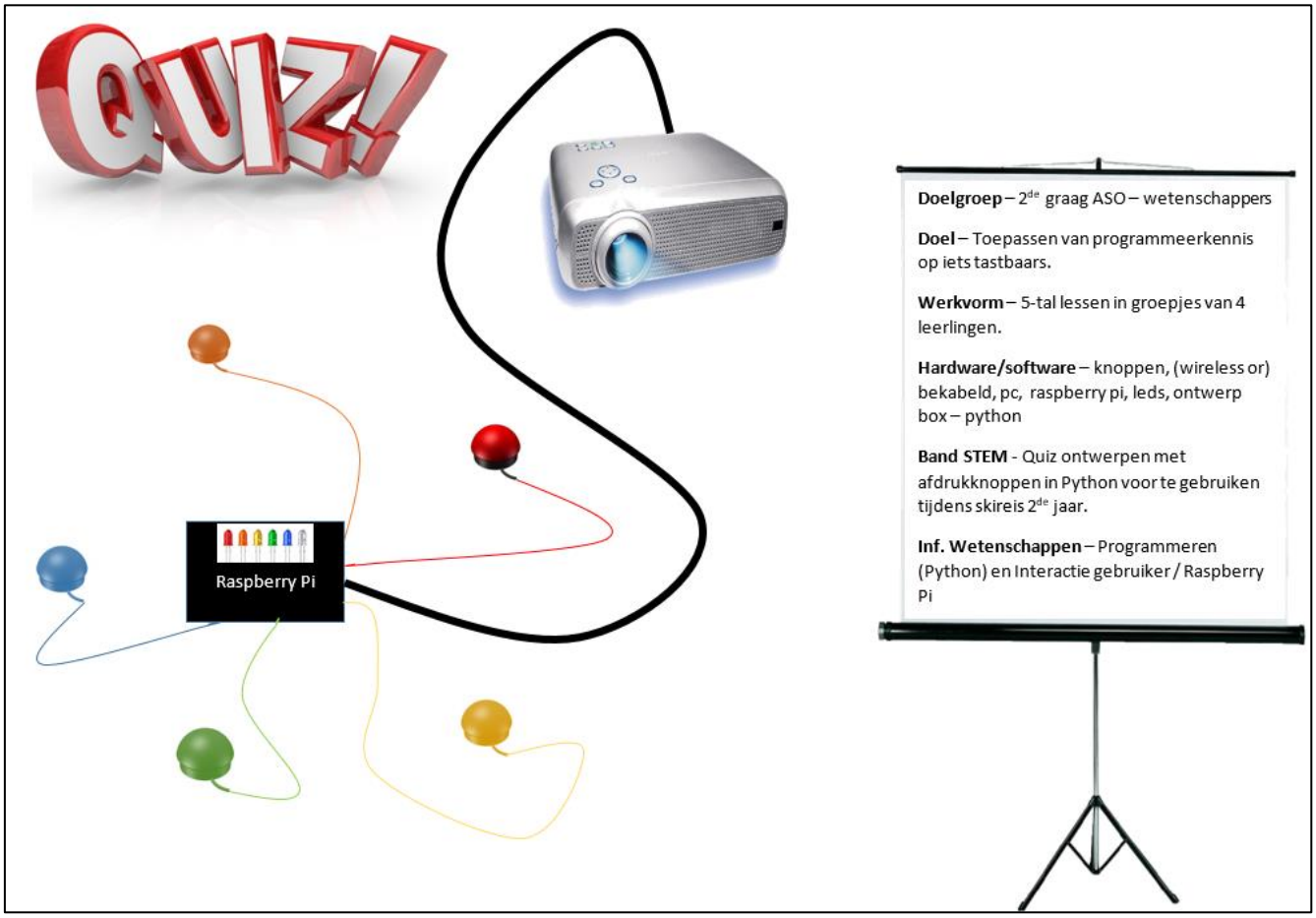
STAP 4: Reset functie toevoegen 13

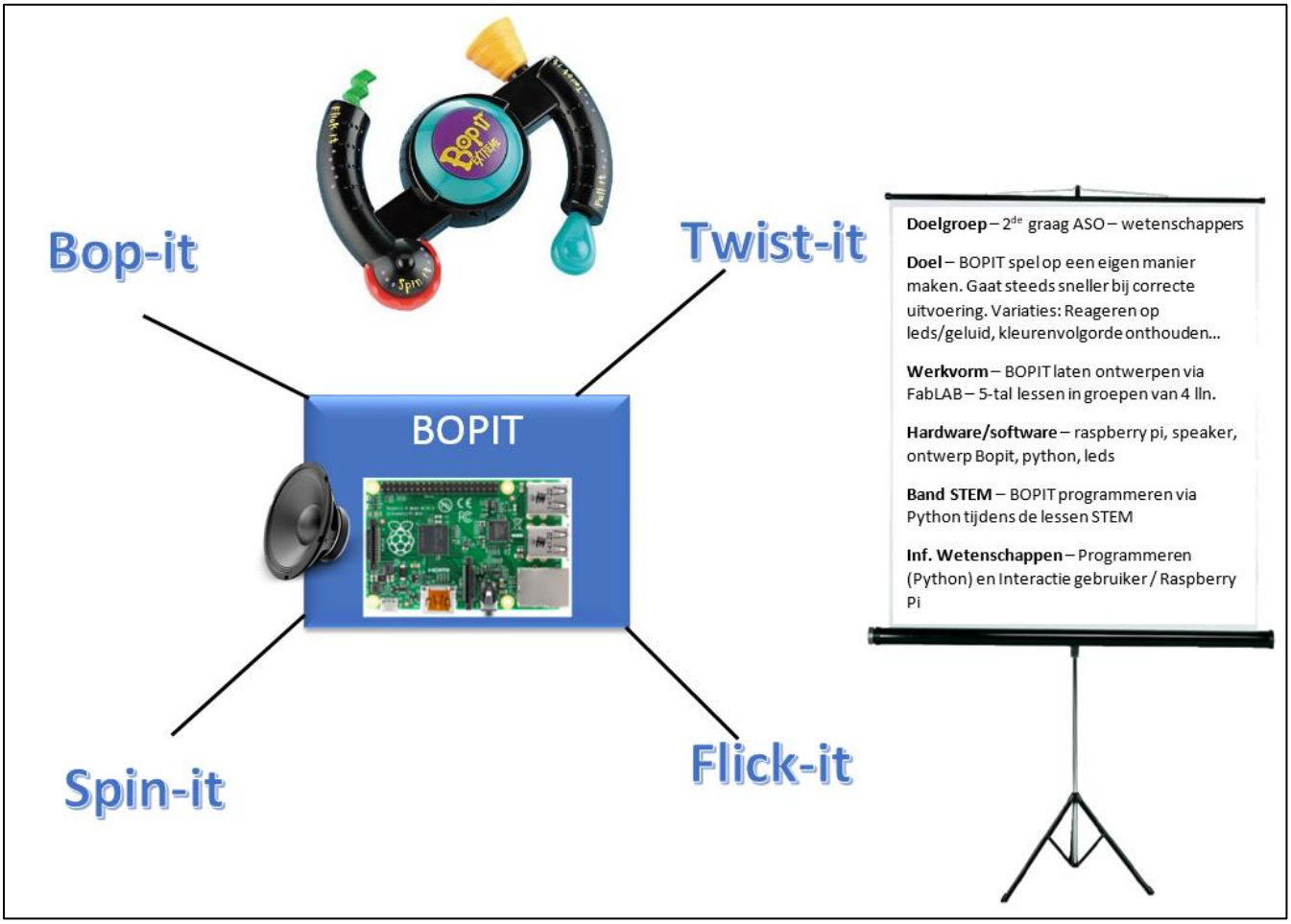
OEFENING 7 14

UITBREIDING: 7-segment display 15

Oplossing OEFENING 6 17

DOEL – QUIZ - BOPIT





PRE-INLEIDING

Wanneer je een aanpassing hebt gemaakt, laat je je opstelling steeds nakijken door een leerkracht.

Sluit de raspberry pi aan op het elektriciteitsnet.

Indien gebruikersnaam en wachtwoord worden gevraagd: (gebruikersnaam: pi //wachtwoord : raspberry)

Indien je grafische interface niet opstart typ je : startx en druk je op enter.

BETEKENIS GPIO – LED – PHYSICAL COMPUTING

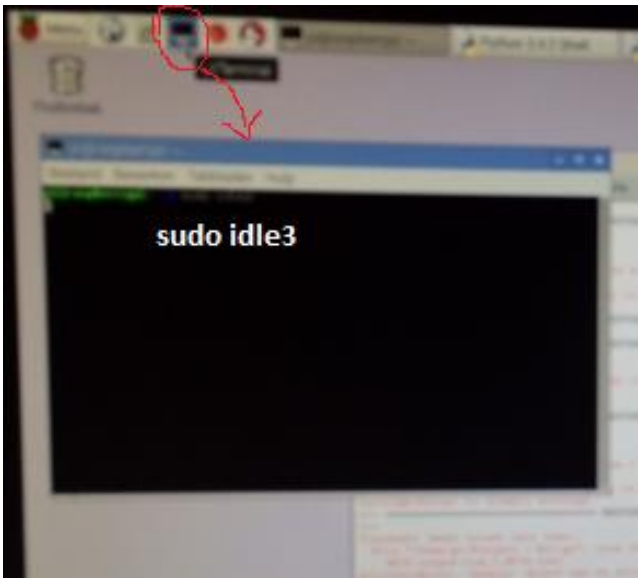
GPIO
General purpose input / output, in dit specifieke geval de pinnen op de Raspberry Pi en wat je kunt doen met deze pinnen. Je kan ze gebruiken voor allerlei doeleinden; de meeste kunnen gebruikt worden als in- of uitgangen, afhankelijk van het programma.

LED
Lichtgevende diode- een kleine , laag energie lichtbron op grote schaal gebruikt in de elektronica. Ideaal als inleiding voor physical computing op de Pi.

physical computing
Computing dat andere tastbare dingen aansluit op een computer, naast de standaard input en output apparaten zoals toetsenborden en monitors. Denk knoppen , verlichting , robots , alarmen , sensoren , domotica , en ga zo maar door. Wij houden van physical computing , want evenals het plezier, is het zo'n krachtige onderwijs- en leermiddel en stimuleert het de creativiteit, problemen oplossen, en samenwerking tussen leerlingen en je kunt heel leuke dingen te maken!

PYTHON IDLE3 OPENEN OP DE RASPBERRY PI

De makkelijkste manier om Python op te starten is met de IDLE. Open IDLE via het startmenu / programmeren / **Python 3**



Lukt het niet om Python uit te voeren? (F5) Ga dan eerst je programma Idle3 als administrator uitvoeren via het command venster (LXTerminal) bovenaan je scherm en typ **sudo idle3**

-

INLEIDING

Een pin met een **nummer** kan gebruikt worden om signalen te ontvangen (= input) of om signalen te versturen (= output). In je programma maak je bij de start duidelijk of je de pin als input of output gaat gebruiken. Een pin kan dan een 0-signaal (laag signaal/geen stroom) of 1-signaal ontvangen/versturen (hoog signaal/wel stroom).

DIA → IT's ALIVE → led aansturen m.b.v. schuifschakelaar daarna met transistor, link → transistor verwerkt in led

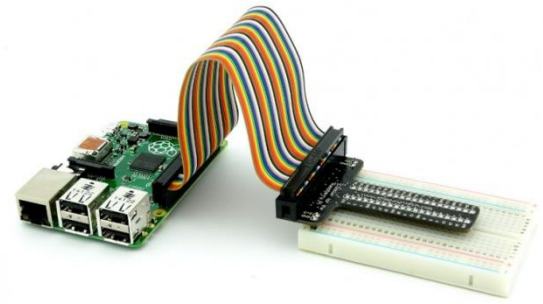
Sommige pinnen zorgen voor een stroomvoorziening, 5V of 3.3V → pluspool, GND → **minpool**

Led met ingebouwde weerstand → **rode snoer = pluspool (VCC,5V)**

zwarte snoer = minpool(GND)

STAP 1B. EEN LED AAN- OF UITZETTEN – ZONDER WEERSTAND

OPSTELLING MAKEN:

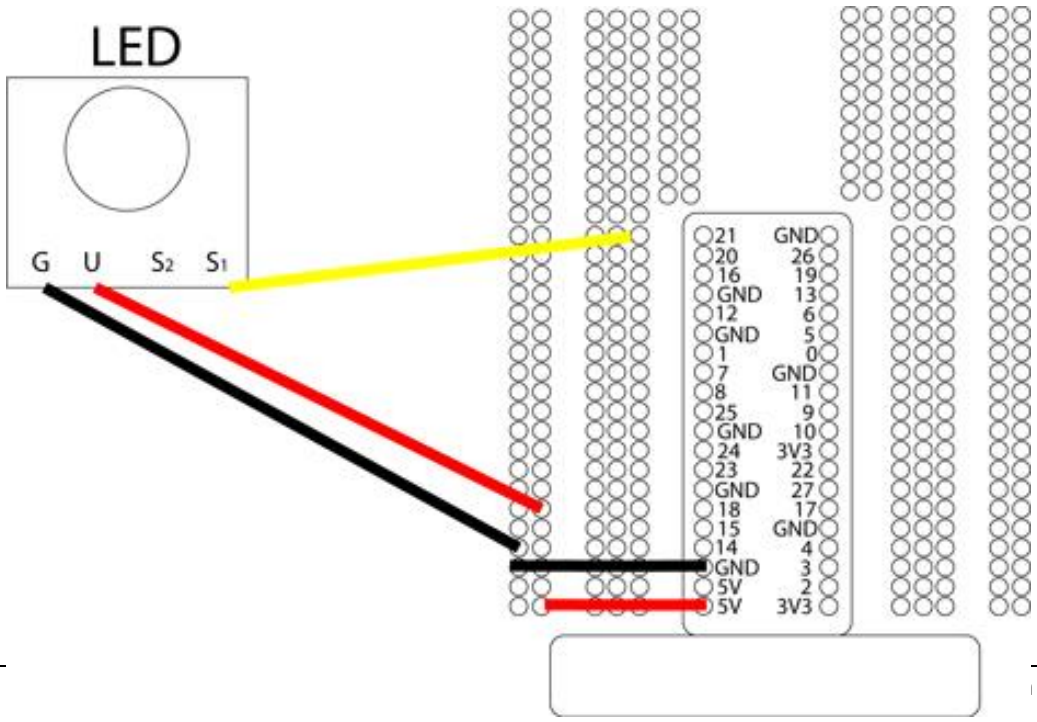


→ zorg dat de raspberry pi niet aangesloten is op het elektriciteitsnet

→ sluit de flatcable correct aan op de raspberry en op de cobbler
→ verbind een 5V pin met een rode/oranje draad en de pluspoolrail aan de rechterkant van je breadboard
→ verbind een GND-pin met een zwarte draad en de minpoolrail aan de linkerkant van je breadboard

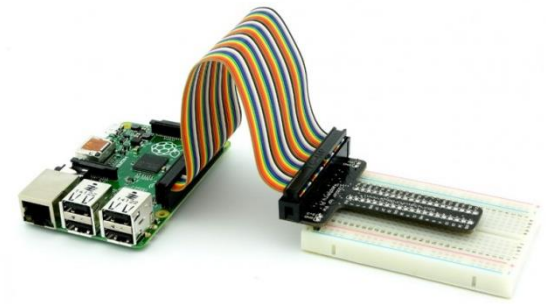
→ verbind een rode kabel van de 5V rail met de VCC pin van je ledlamp
→ verbind een zwarte kabel van de GND rail met de GND pin van je ledlamp

→ verbind met een gele/gekleurde kabel pin 21 met de SIG-pin van je ledlamp, met deze pin ga je de ledlamp aansturen.



STAP 1A. EEN LED AAN- OF UITZETTEN – MET WEERSTAND

OPSTELLING MAKEN:



→ Zorg dat de raspberry pi niet aangesloten is op het elektriciteitsnet

→ sluit de flatcable correct aan op de raspberry en op de cobbler
→ verbind een 5V pin met een rode/oranje draad en de pluspoolrail aan de rechterkant van je breadboard
→ verbind een GND-pin met een zwarte draad en de minpoolrail aan de linkerkant van je breadboard

→ Plaats de led in het breadboard, zorg dat 2 verbindingen zich niet in dezelfde horizontale verbindingen bevinden.

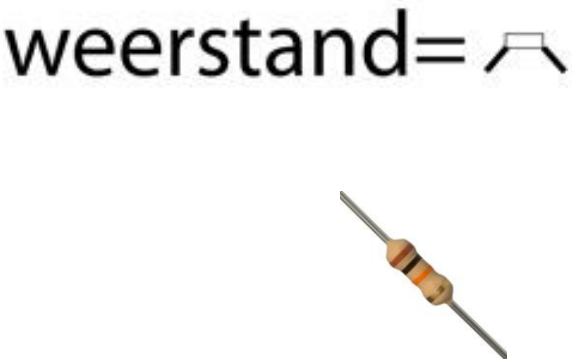
→ Een led verbruikt zo goed als geen stroom, om geen kortsluiting te veroorzaken, maak je gebruik van een weerstand

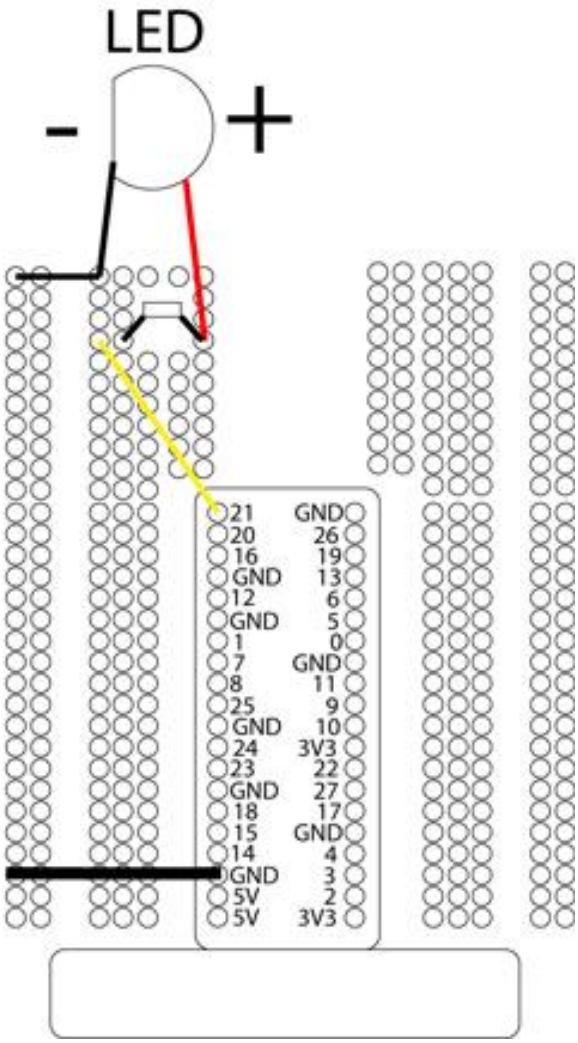
→ verbind een rode kabel van pin 21 naar de pluspool van de led.

→ verbind de minpool van de led met een weerstand.

→ verbind een zwarte kabel van de andere kant van de weerstand met de GND rail van je breadbord.

→ **Laat je opstelling door een leerkracht nakijken.**





CODE: LED AAN EN UIT

Betekenis GPIO = General Purpose Input / Output

Typ onderstaande code over (de commentaar hoe je niet te typen):

import RPi.GPIO as GPIO	# je importeert de module RPI.GPIO, als je deze module gebruikt dan moet je enkel de naam GPIO typen
led _1 = 21	# je gaat de eerste led aansluiten op pin 21,deze pin gaat een 0-sigaal versturen als de led uit moet zijn of een 1-signaal om de led # aan te zetten
GPIO.setmode(GPIO.BCM)	# dit geeft aan dat je op de cobbler je pinnummer kan aflezen en deze hiernaar kan verwijzen door het juiste nummer te gebruiken.
GPIO.setup (led_1,GPIO.OUT)	# je stelt de pin met als waarde led_1 (in dit geval 21) in als een uitgaand signaal, je gaat iets aansturen.
GPIO.output (led_1,GPIO.HIGH)	# laat de pin met als waarde led_1 (in dit geval 21) een 1-signaal versturen
GPIO.cleanup()	# zorg dat geen enkele pin stroom levert (zet ze allemaal op een 0-signaal anders kan de pin zijn laatste status blijven behouden (en blijvend een 1-signaal versturen)

Sla je bestand op, onder de naam **test_led**

Voer je programma uit.

Heb je de led zien aangaan?

De led is een fractie van een seconde aangegaan en dan heeft het programma al de pinnen “proper” gemaakt. Het programma heeft deze instructies zo snel uitgevoerd zodat je hiervan niets gemerkt hebt.

Welke module heb je reeds gezien om het programma even te laten pauzeren?

Importeer deze module en zorg dat je programma 2 seconden pauzeert nadat je pin 21 op hoog hebt gezet.

Kijk je programma na of de led 2 seconden brandt.

CODE: DE GEBRUIKER BEPAALT OF DE LAMP AAN OF UIT MOET

Stel dat je de gebruiker vraagt of hij/zij wil dat de lamp aan of uit gezet wordt. Indien de gebruiker stop antwoord, moet het programma stoppen en dan worden ook al de pinnen proper gemaakt

```
import RPi. GPIO as GPIO
led _1 = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup (led_1,GPIO.OUT)
antwoord = “uit”
while antwoord != “stop”:                    # zolang de variabele antwoord verschillend is van de waarde
                                             “stop” voer de code, die na het dubbel punt inspringt, uit.
    antwoord = input (“Wil je de lamp aan- of uitschakelen ? aan/uit/stop ”)
    if antwoord == “aan”:                    # als de waarde van de variabele antwoord gelijk is aan “aan”
        GPIO.output (led_1,GPIO.HIGH)        # de pin met waarde led_1 verstuurt een 1-signaal
    else:                                    # in al de andere gevallen (uit,stop,…)
        GPIO.output (led_1,GPIO.LOW)         # de pin met waarde led_1 verstuurt een 0-signaal
GPIO.cleanup()
```

Zoals je merkt, gebruik je hier geen TIME om je programma te pauzeren. Je programma wacht steeds totdat de gebruiker iets heeft ingetypt (of op ENTER heeft gedrukt)

OEFENING 1:

Pas de code aan zodat de ledlamp 5X knippert. (5X 1 seconde aan en 5X 1 seconde uit).

OEFENING 2:

Sluit je raspberry Pi af.

Verwijder de voeding uit het stopcontact want je gaat iets van hardware moeten veranderen.

Sluit een **2de ledlamp** aan, gebruik hiervoor het schema van ledlamp_1 . Je gebruikt **pin 18** om je lamp aan te sturen.

Laat je opstelling controleren door een leerkracht.

Steek de voeding in en start je RPi op.

Open idle3.

Pas je programma aan zodat je aan de gebruiker vraagt om lamp 1 of lamp 2 te laten branden. Het nummer van de pin waarmee je de ledlamp aanstuurt stop je in de variabele led_2 .

STAP 2: WERKEN MET 1 KNOP

Doel: Knop gebruiken om LED aan te steken en weer uit te zetten.

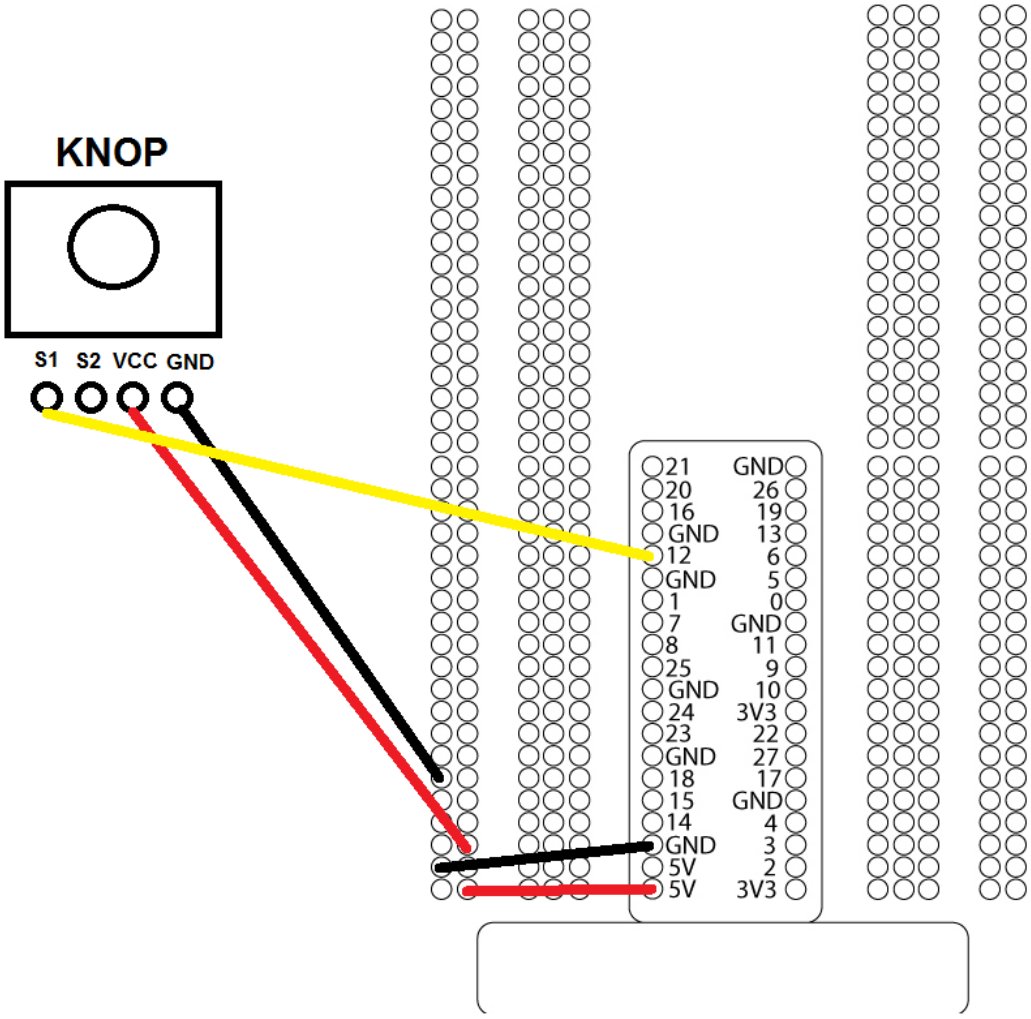
We kunnen al de LED lamp aan-en uit zetten via code. In deze stap willen we een knop toevoegen. De gebruiker zal de knop induwen om de lamp aan te steken en nog eens op de knop duwen om de lamp weer uit te zetten.

KNOP AANKOPPELEN

Opmerking: onze LEDs blijft zitten op pin 21 en 18.

Je verbindt de knop op dezelfde manier met de breadboard als je dat eerder met de LED deed. Maar we gebruiken wel een andere pin In het voorbeeld hieronder gebruiken we pin 12.

→ zorg dat de raspberry pi niet aangesloten is op het elektriciteitsnet !
→ verbind een rode kabel van de 5V rail met de VCC pin van je knop
→ verbind een zwarte kabel van de GND rail met de GND pin van je knop
→ verbind met een gele/gekleurde kabel pin 12 met de SIG-pin (of VCC pin) van je knop, via deze pin ga je de signalen van de knop ontvangen



CODE: "LUISTEREN" NAAR DE KNOP

De code hieronder wacht tot iemand op de knop duwt. Wanneer dit gebeurt, printen we een boodschap naar het scherm.

Maak een nieuw bestand en tik de code hieronder in (commentaar hoeft niet):

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

teller = 0

PIN_KNOP = 12                          # variabele waar we het nummer van de gebruikte pin bijhouden

GPIO.setup(PIN_KNOP, GPIO.IN)          # we definieren dat dit een binnenkomende pin is. De raspberry krijgt
                                       het signaal van de pin (0 of 1 afhankelijk van het feit of iemand op de
                                       knop duwt)

KNOP_uit = GPIO.input(PIN_KNOP)        # de waarde die we lezen voor het programma start is de waarde van
                                       deze knop wanneer hij niet is ingedrukt. We noemen deze variabele
                                       "KNOP_uit"

print ("Druk op knop om boodschap te geven.  Je kan 3 keer proberen.")

while teller < 3:

   while GPIO.input(PIN_KNOP) == KNOP_uit:  # knop niet ingedrukt

      time.sleep(0.01)                    # we wachten een honderdste van een seconde en kijken opnieuw of de
                                          knop al ingedrukt is

    print ("knop ingedrukt !!")           # aangezien we uit de while lus geraakt zijn, heeft iemand de knop
                                          ingedrukt

   teller += 1

   print ("teller: " + str(teller))

   time.sleep(1)                          # we wachten even vooraleer we terug gaan wachten (als de teller
                                          tenminste kleiner is dan 3)

print ("programma is klaar")              # als de teller 3 is, komen we hier en printen we dat het programma
                                          klaar is (zodat de gebruiker niet vruchteloos wacht op een telen van
                                          leven.)

GPIO.cleanup()
```

Bewaar je bestand als **Knop_eenvoudig.py** en test het uit.
Laat zien aan een leerkracht.

OEFENING 3

Verwijder de time.sleep(1) (de voorlaatste lijn code) en kijk wat er gebeurt.

Enig idee wat er aan de hand is?

CODE: KNOP BEDIENT LED

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

PIN_KNOP = 12

PIN_LED = 21

GPIO.setup(PIN_KNOP, GPIO.IN)          # knop geeft inkomend signaal (waar we op wachten)

GPIO.setup(PIN_LED, GPIO.OUT)          # LED is uitgaand signaal dat wij geven

KNOP_uit = GPIO.input(PIN_KNOP)


print ("Druk op knop om LED aan te steken.  LED zal blijven branden tot je nog eens op de knop duwt.")

while GPIO.input(PIN_KNOP) == KNOP_uit:    # knop niet ingedrukt

   time.sleep(0.01)                        # we wachten heel even en controleren of de knop is ingedrukt

GPIO.output(PIN_LED,1)                     # knop ingedrukt, lamp laten branden

while GPIO.input(PIN_KNOP) != KNOP_uit:    # extra lus om zeker te zijn dat de knop al is losgelaten.

   time.sleep(0.01)

while GPIO.input(PIN_KNOP) == KNOP_uit:    # Pas daarna beginnen we weer te wachten op het indrukken van de
                                           knop (om de lamp te kunnen uitdoen).

   time.sleep(0.01)

GPIO.output(PIN_LED,0)                     # Knop ingedrukt. Deze keer is dat het signaal om de LED uit te zetten


print ("Programma is klaar.")

GPIO.cleanup()
```

Bewaar je bestand als **Knop_LED.py** en test het uit.

OEFENING 4

Zet de middelste lus in commentaar (deze die dient om zeker te zijn dat de knop is losgelaten). Sla je programma op en test opnieuw. Wat stel je vast?

Wat zou een alternatieve methode zijn om dit probleem te omzeilen?

STAP 3: WERKEN MET 2 KNOPPEN EN 2 LEDS

Doel: We naderen het doel. We willen een quiz-systeem! Nu voegen we een tweede knop toe. Alleen de groep die het eerste op de knop duwt, zal zijn antwoord mogen geven. Het lampje van de eerste groep gaat branden.

TWEEDE KNOP AANKOPPELEN

Je verbindt de knop op dezelfde manier met de breadboard als je dat eerder met de LEDs en de eerste knop deed. Indien nodig, keer terug naar de beschrijving in Stap2.

Ondertussen hebben we al heel wat pinnen gebruikt. Hou je pin nummers bij in de tabel hieronder.

Item	**Pin**
Led 1 (kleur)	
Led 2 (kleur)	
Knop 1	
Knop 2	

OEFENING 5

Pas je code in het bestand **Knop_LED.py** (zie de stap hiervoor) aan om de nieuwe knop en de tweede LED uit te testen.

Laat zien aan een leerkracht.

CODE: EERSTE KNOP HAALT HET

Deze code werkt verder op het scenario uit stap2. Nu gebruiken we dus 2 knoppen en 2 LEDs. Alleen de groep die als eerste op de knop duwt, zal zijn LED zien branden en mag dus op de quizvraag antwoorden.

Vul de kleuren van je LEDS in in de code hieronder.

Enkele vraagjes:

- Wat gebeurt er in het groene deel van de code? .....
- Hoeveel seconden blijft de LED van het snelste team branden? .....

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

PIN_KNOP_1 = 12
PIN_KNOP_2 = 24
PIN_LED_1 = 21
KLEUR_LED_1="VUL IN"
PIN_LED_2 = 18
KLEUR_LED_2 ="VUL IN"

GPIO.setup(PIN_KNOP_1, GPIO.IN)
GPIO.setup(PIN_KNOP_2, GPIO.IN)
GPIO.setup(PIN_LED_1, GPIO.OUT)
GPIO.setup(PIN_LED_2, GPIO.OUT)
KNOP_Uit_1 = GPIO.input(PIN_KNOP_1)
KNOP_Uit_2 = GPIO.input(PIN_KNOP_2)
PIN_TO_USE = 0
KLEUR_TO_USE = ""

print ("Vraag 1: wie het antwoord weet, mag afdrukken!")
while (GPIO.input(PIN_KNOP_1)== KNOP_Uit_1) and (GPIO.input(PIN_KNOP_2)== KNOP_Uit_2):
   time.sleep(0.01)# we wachten tot één van de knoppen is ingedrukt
lees_knop_1 = GPIO.input(PIN_KNOP_1)
lees_knop_2 = GPIO.input(PIN_KNOP_2)
                                    # 1 van de knoppen is ingedrukt, actie nemen afhankelijk van de ingelezen waardes
if lees_knop_1!= KNOP_Uit_1:
   PIN_TO_USE = PIN_LED_1
   KLEUR_TO_USE = KLEUR_LED_1
if lees_knop_2!= KNOP_Uit_2:
   PIN_TO_USE = PIN_LED_2
   KLEUR_TO_USE = KLEUR_LED_2
GPIO.output(PIN_TO_USE,1)
print("Team "+ KLEUR_TO_USE + " aan de beurt!")
time.sleep(2)
GPIO.output(PIN_TO_USE,0)

print ("Einde van de quiz !")
GPIO.cleanup()
```

Bewaar je bestand als **Twee_Knoppen.py** en test het uit.

OEFENING 6

Pas het programma aan zodat er 4 vragen kunnen gesteld worden vooraleer het programma ten einde is.

Zorg ervoor dat het nummer van de vraag verschijnt in de print boodschap “Vraag x: wie het antwoord weet, mag afdrukken!".

Laat zien aan een leerkracht vooraleer te testen.

STAP 4: RESET FUNCTIE TOEVOEGEN

In het vorige scenario, bleef de LED een paar seconden branden en gingen we dan over naar de volgende vraag. In realiteit kan het beantwoorden meer of minder tijd in beslag nemen.

Daarom willen we een “reset” functie toevoegen. Het scenario is als volgt:

- De eerste vraag wordt gesteld, de snelste ploeg drukt en zijn lamp brandt.
- Dan wacht het programma op een actie van de quizmaster. Wanneer de quizmaster op de reset knop drukt, gaat de lamp uit en gaan we naar de volgende vraag.
- Dit herhalen we 4 maal (zoals in de vorige versie van ons programma al het geval was).

De code hieronder werkt verder op de vorige stap maar mist nog de essentie van de reset-functie.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

teller = 0
PIN_KNOP_1 = 12
PIN_KNOP_2 = 24
PIN_KNOP_3 = 20
PIN_LED_1 = 21
KLEUR_LED_1="Geel"
PIN_LED_2 = 18
KLEUR_LED_2 ="Rood"

GPIO.setup(PIN_KNOP_1, GPIO.IN)
GPIO.setup(PIN_KNOP_2, GPIO.IN)
GPIO.setup(PIN_KNOP_3, GPIO.IN)
GPIO.setup(PIN_LED_1, GPIO.OUT)
GPIO.setup(PIN_LED_2, GPIO.OUT)

KNOP_Uit_1 = GPIO.input(PIN_KNOP_1)
KNOP_Uit_2 = GPIO.input(PIN_KNOP_2)
PIN_TO_USE = 0
KLEUR_TO_USE = ""
KNOP_Uit_3 = GPIO.input(PIN_KNOP_3)
```

```
while teller<4:
    print ("Vraag " + str(teller+1) +": wie het antwoord weet, mag afdrukken!")

    while (GPIO.input(PIN_KNOP_1)== KNOP_Uit_1) and (GPIO.input(PIN_KNOP_2)== KNOP_Uit_2):
        time.sleep(0.01)# we wachten tot één van de knoppen is ingedrukt
    lees_knop_1 = GPIO.input(PIN_KNOP_1)
    lees_knop_2 = GPIO.input(PIN_KNOP_2)
                                    # 1 van de knoppen is ingedrukt, actie nemen afhankelijk van de ingelezen waardes
    if lees_knop_1!= KNOP_Uit_1:
        PIN_TO_USE = PIN_LED_1
        KLEUR_TO_USE = KLEUR_LED_1
    if lees_knop_2!= KNOP_Uit_2:
        PIN_TO_USE = PIN_LED_2
        KLEUR_TO_USE = KLEUR_LED_2
    GPIO.output(PIN_TO_USE,1)
    print("Team "+ KLEUR_TO_USE + " aan de beurt!")

                                    # TODO: reset functie toevoegen

    teller +=1
print ("Einde van de quiz !")
GPIO.cleanup()
```

Sla dit bestand op als **Quiz_met_reset.py**.

Test het nog niet !! Werk eerst de volgende oefening af.

OEFENING 7

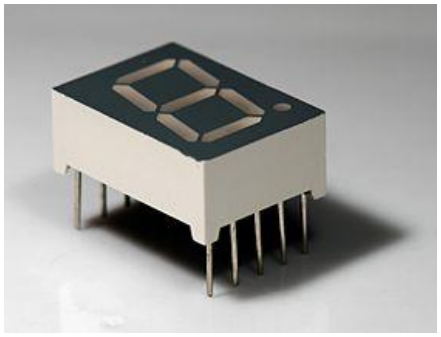
Voeg de nodige code toe op de plaats waar **#TODO** staat:

- Mogelijkheid 1: Installeer nog een derde knop die dienst zal doen als reset knop. Wanneer de quizmaster op knop 3 drukt, dan moet de lamp die brandt, afgezet worden. Enkele tips:
    - Herinner je: om te weten of een knop is ingedrukt, “wachten” we telkens 0,01 seconde zolang de knop niet ingedrukt is.
    - Welke variabele bewaart de pin van de lamp die brandt?

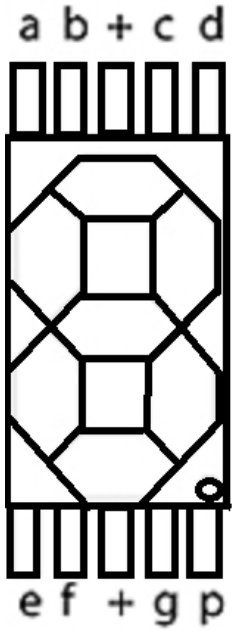
- Mogelijkheid 2: Als er geen tijd genoeg is om een extra knop te installeren, kan je hetzelfde effect ook bekomen door een print boodschap met een vraag: “Klaar voor de volgende vraag? Antwoord Y. ”. Als de quizmaster Y antwoordt, dan moet de lamp die brandt afgezet worden.

UITBREIDING: 7-SEGMENT DISPLAY

Een 7-segment-display bevat 8 ledjes die apart aangestuurd kunnen worden.
Met behulp van 7 ledjes kan je al de digitale cijfers vormen.(en enkele letters)
Het 8ste ledje is om het decimale punt weer te geven.



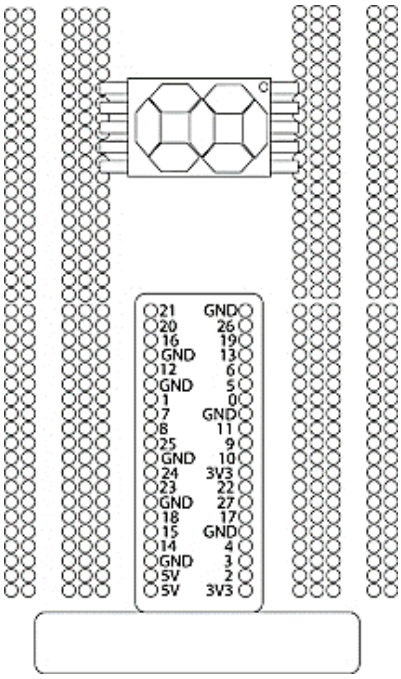
Als je het aantal connectorpinnen van een display telt dan merk je dat je er 10 hebt.
Elke middelste connectorpin is verbonden met elke andere pin. Bij onze display dient deze pin om een verbinding te maken met de pluspool (5V). Je hoeft slechts 1 van de 2 middelste connectorpinnen te gebruiken om je display aan te sturen.



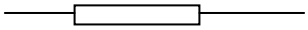
Zorg dat je raspberry pi afgesloten is en ontkoppel hem van het elektriciteitsnetwerk.

→ verbind de 5V pin van de GPIO met de pluspool rail.

→ plaats je display zoals op afbeelding 1(zie hieronder).(laat voldoende plaats tussen de cobbler en de display)



→elk ledje kan apart aangestuurd worden, dus elk ledje wordt door een andere pin aangestuurd. Een led biedt zo goed als geen weerstand tegen de stroom. Om kortsluiting te vermijden moeten we zorgen dat er in de elektrische kringloop bij elke led ten minste 1 weerstand aanwezig is.[1]Maak nu al de juiste verbindingen zoals in de afbeelding.

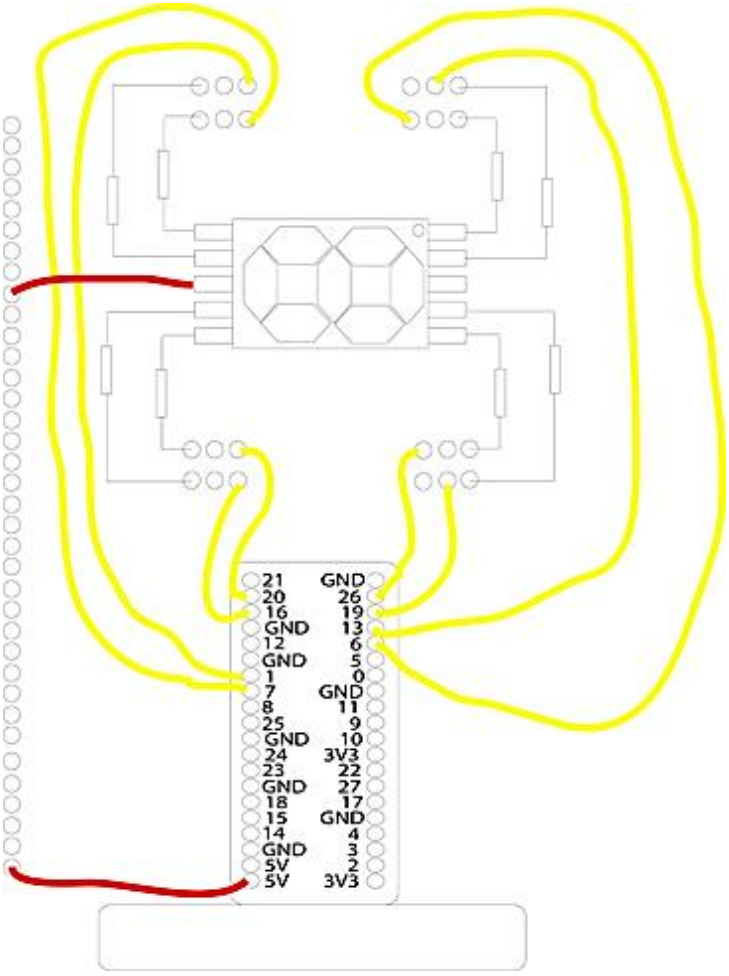


Dit is het symbool van een weerstand[2], zorg dat je deze op de juiste manier plaatst in je elektrische kringloop.

→ laat je opstelling controleren door een leerkracht
→ Schrijf nu eerst een programma om uit te pluizen door welke pin elke led wordt aangestuurd. Noteer je oplossing op een blad.

→ Schrijf nu een programma zodat een digitaal cijfer laat zien op het display gedurende 2 seconden.
→ Indien dit lukt, schrijf dan een programma waarbij aan de gebruiker een cijfer vraagt en vervolgens het juiste digitale cijfer laat zien gedurende 2 seconden

→ herschrijf je programma van de quiz, zodat je op de display laat verschijnen of speler 1 of 2 als eerste heeft afgedrukt. (LET OP: Je moet je drukknoppen terug op de juiste manier aansluiten, hiervoor moet je eerst je raspberry uitschakelen en verwijderen uit het stopcontact.)



[1] Om deze weerstand te berekenen, moet je de wet van ohm toepassen cfr WW

[2] Ook al bij techniek en wetenschappelijk werk gezien ☺

OPLOSSING OEFENING 6

```
                                                    # reset knop om volgende vraag te stellen
while (GPIO.input(PIN_KNOP_3)== KNOP_Uit_3):
    time.sleep(0.01)                                # we wachten op indrukken van de reset knop
GPIO.output(PIN_TO_USE,0)
```