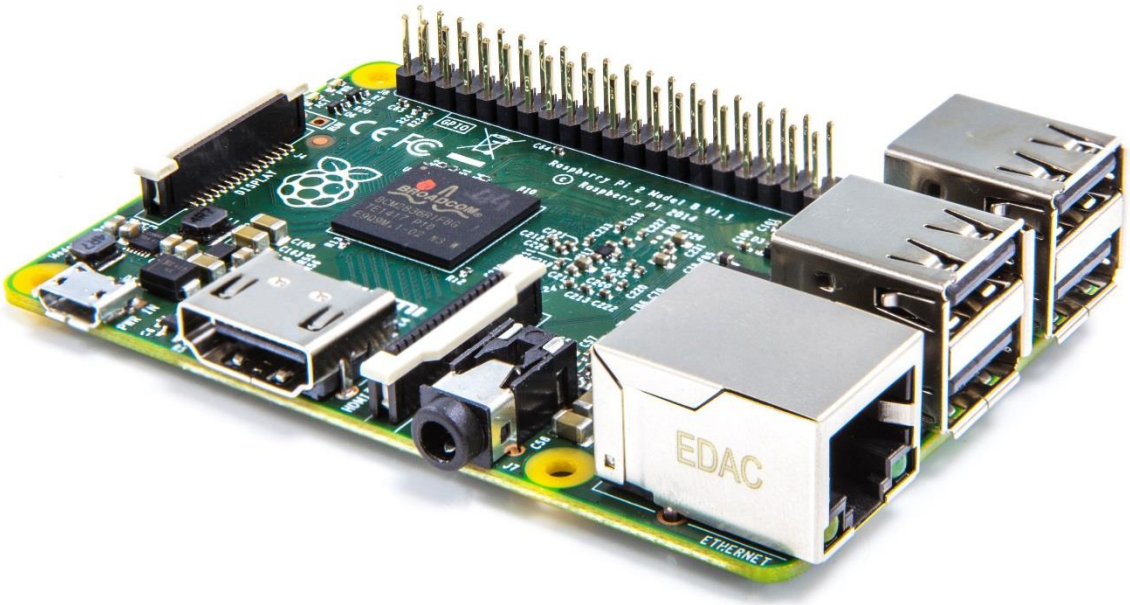
RaspberryPi

Werken met microprocessors



6 BI | 2015-2016

VAN WOENSEL K.

Inhoud

1. Inleiding ... 3
2. Basis elektronica ... 4
   2.1 Enkele belangrijke begrippen ... 4
      2.1.a De spanningsbron ... 4
      2.1.b Spanning (U) ... 4
      2.1.c Stroom (I - Intensity) ... 4
      2.1.d Weerstand (R – Resistance) ... 5
      2.1.e Oefeningen ... 5
      2.1.f Elektrisch vermogen (P – Power) ... 6
      2.1.g Oefeningen: vermogen ... 6
      2.1.h Open en gesloten circuits ... 7
      2.1.i Kortsluiting ... 7
      2.1.j Serie en parallelschakeling ... 7
      2.1.a Het elektrisch circuit ... 8
3. De Raspberry Pi ... 10
   3.1 De GPIO pinnen ... 10
   3.2 GPIO componenten ... 11
      3.2.a Breadboard ... 11
      3.2.b LED ... 11
      3.2.c OPDRACHT 1 ... 12
      3.2.d Drukknop ... 13
      3.2.e OPDRACHT 2 ... 13
      3.2.f The Shell ... 14
      3.2.g OPDRACHT 3 ... 14
      3.2.h Shell Commando's ... 15
4. Programmeren met Python ... 16
   4.1 Syntaxis ... 16
   4.2 Idle3 ... 16
   4.3 De GPIO module ... 17
      4.3.a Pin nummering ... 17
      4.3.b Warnings ... 18
      4.3.c Pinnen configureren ... 18

4.3.d Pinnen gebruiken ... 18
4.3.e Cleanup ... 19
4.3.f OPDRACHT 4 ... 19
4.3.g Blokstatements ... 19
4.3.h OPDRACHT 5 ... 21
5. Domoticaproject ... 22
5.1 HTML templates ... 23
5.1.a Uitbreiding ... 23
5.2 Bottle library ... 24
5.2.a Routes maken ... 25
5.2.b Stylesheets opladen ... 26
5.2.c Uitbreiding ... 27
5.2.a OPDRACHT 6 ... 28
5.3 Servomotoren ... 30
5.3.a Werking van een servomotor ... 30
5.3.b Aansturen van een servomotor ... 31
5.3.c OPDRACHT 7 ... 32
5.3.d OPDRACHT 8 ... 32
6. Oplossingen ... 33
6.1 Oefening 1 ... 33
6.2 Oefening 2 ... 33
6.3 Oefening 3 ... 33
6.4 Opdracht 1 ... 34
6.5 Opdracht 2 ... 34
6.6 Opdracht 4 ... 35
6.7 Opdracht 5.A ... 35
6.8 Opdracht 5.B ... 36
6.9 Opdracht 6.A ... 36
6.10 Opdracht 6.B ... 37
6.11 Opdracht 6.C ... 37
6.12 Opdracht 6.D ... 38
6.13 Opdracht 6 - Uitbreiding ... 39
7. Nuttige links ... 40

1. Inleiding

In deze cursus leer je werken met de Raspberry Pi. De Raspberry Pi is een eenvoudige 'computer' die door de University of Cambridge is ontworpen en vooral is bedoeld voor het onderwijs. De computer werkt op verschillende open-source Linuxdistributies (NOOBS/Raspbian) die op alle mogelijk manieren aan te passen zijn. In de klas wordt gewerkt met de Raspberry Pi 2.

De cursus focust vooral op het werken met de GPIO (General Purpose Input/Output) module. Deze module maakt het mogelijk om allerhande elektronische componenten met de microprocessor te verbinden. Signalen van deze componenten kunnen door de processor gelezen en/of aangestuurd worden.

Deze cursus bestaat uit vier grote onderdelen:

- Basis elektronica — Dit hoofdstuk geeft een overzicht van de belangrijkste bouwstenen van elektronica. Deze basis is vereist om op een verantwoorde manier met het materiaal te werken.
- De Raspberry Pi — Dit hoofdstuk richt zich vooral op de werking van de GPIO module van de Raspberry Pi, en geeft een overzicht van mogelijke elektronische componenten die aangesloten kunnen worden.
- Programmeren met Python — Als programmeertaal wordt gekozen voor Python. Deze taal is heel overzichtelijk en makkelijk leesbaar. In dit hoofdstuk worden de belangrijkste principes en structuren toegelicht.
- Domoticaproject — In het laatste hoofdstuk wordt de theoretische kennis in de praktijk gebracht. In dit project is het de bedoeling om elektronische componenten aan te sturen met behulp van een HTML pagina.

Een basiskennis van programmeren en van HTML is vereist.
Bekijk hiervoor eventueel http://www.w3schools.com/.

2. Basis elektronica

2.1 Enkele belangrijke begrippen

2.1.a De spanningsbron

Alles begint bij de bron. Om elektrische apparaten te doen werken gebruiken we **wisselspanning** die we, in de meeste gevallen, uit een stopcontact halen. Het stopcontact is een **wisselspanningsbron**.

In een elektrisch apparaat zit hoe langer hoe meer elektronica. De spanningsbron die nodig is om elektronica te laten werken is meestal een **gelijkspanning**. Intern wordt de wisselspanning omgevormd naar een gelijkspanning. Op het moment dat we op een spanningsbron een verbruiker (weerstand, lamp, motor) aansluiten zal er een **stroom (I)** beginnen vloeien. Afhankelijk van de gebruikte spanningsbron spreken we van een **wisselstroom (AC = Alternate Current)** of een gelijkstroom (**DC = Direct Current)**.

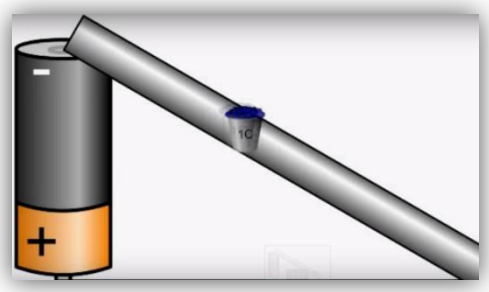
Een veel gebruikte spanningsbron voor kleine elektrische apparaten is een batterij. Een batterij is altijd een gelijkspanningsbron. Steeds aanwezig is de aanduiding van de positieve pool, aangeduid door een '+'-teken.

2.1.b Spanning (U)

Elektrische spanning is een potentiaal verschil tussen twee punten in een elektrisch circuit en wordt uitgedrukt in **Volt** (V). Specifiek worden gelijkspanningen aangeduid met VDC en wisselspanningen VAC. Een spanning wordt altijd gemeten ten opzichte van het massapotentiaal ground (**GND**). De spanningsbronnen die we in systemen met microcontrollers tegenkomen zijn veelal +3,3VDC, +5VDC en +12VDC.

Volgens de **Wet van Ohm** geldt: **U = I x R** (Volt)

Je zou spanning min of meer kunnen vergelijken met de werking van een rivier. Water in een stilstaand meer stroomt niet, omdat de bodem 'gelijk' ligt. Moest je de ene kant van de bodem kunnen opheffen, ontstaat spanning. Door de zwaartekracht, stroomt het water van hoog naar laag. Bij elektriciteit gaat het niet om energie uit zwaartekracht, maar om elektrische energie. We creëren een verschil in potentiële elektrische energie [1]. Elektronen vloeien van het punt met 'hoger' potentieel elektrische energie, naar het punt met 'kleiner' potentieel elektrische energie. Hoe groter het verschil, hoe meer de stroom tussen de polen vloeit.



Figuur 1: Elektrische deeltjes (uitgedrukt in Coulomb) vloeien van het punt met het hogere potentiële energie naar het punt met lagere potentiële energie.

2.1.c Stroom (I - Intensity)

Een elektrische stroom ontstaat wanneer we een verbruiker verbinden met een spanningsbron. De grootte van de stroom (de stroomsterkte) is afhankelijk van de belasting die er op wordt aangesloten. Elektrische stroom wordt uitgedrukt in **Ampère**, afgekort **A**. In microprocessorsystemen werken we

[1] *Potentiële energie is energie die in een gegeven situatie kan ontstaan en kan omgezet worden in een beweging.*

met micro-ampère (µA) en milli-ampère (mA). De stroomsterkte drukt uit hoeveel elektrische deeltjes per seconde tussen twee polen bewegen.

Met de wet van Ohm: **I = U/R** (Ampère)

2.1.d Weerstand (R – Resistance)

Weerstand is een elektrische grootheid die aangeeft hoe gemakkelijk er een elektrische stroom door een elektrisch circuit kan vloeien. Weerstand wordt uitgedrukt in **Ohm (Ω)**. Gebruikelijke grootheden in elektronische circuits zijn kilo-Ohm (kΩ) en mega-Ohm (MΩ).

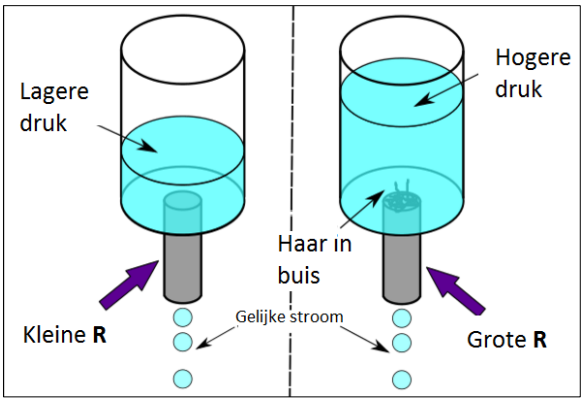
Met de wet van Ohm: **R = U/I** (Ohm)

Om weerstand eenvoudig uit te leggen, kunnen we de vergelijking maken met de afvoer van water. Een bepaalde hoeveelheid water oefent druk uit in een reservoir. Wanneer het water vanuit het reservoir afgevoerd wordt door een buis, ontstaat een stroom.

Wanneer de afvoer vervuild is (bijvoorbeeld met een haar), is een grotere druk nodig om het water met een gelijke stroom door de buis te laten stromen.

Bij elektrische stroom is de situatie vergelijkbaar.

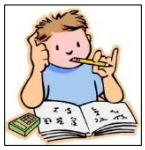
Wanneer we een gelijke stroom willen behouden nadat de stroom door een weerstand vloeit, hebben we een hogere spanning nodig.



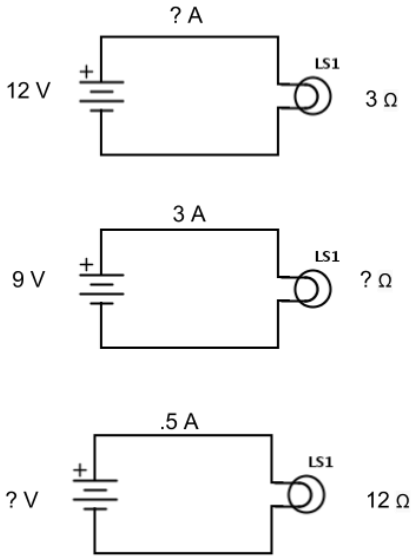
Figuur 2: Weerstand met water

2.1.e Oefeningen

<u>Oefeningen De Wet van Ohm 1</u>



Bereken '?'



<u>Opmerkingen</u>

- Voor de berekeningen hierboven gaan we steeds uit van een ideale spanningsbron. In de praktijk bestaat deze niet en meten we dus licht afwijkende waarden.

2.1.f Elektrisch vermogen (P – Power)

Een elektrisch vermogen geeft aan hoeveel energie (warmte) er opgewekt wordt op één bepaald moment. Men spreekt dan van een momentaan vermogen. Vermogen wordt uitgedrukt in **Watt (W)**. Wanneer we een vermogen verbruiken over een bepaalde tijdsspanne, dan spreken we over energie.

Bij gelijkstroom geldt: **P = U x I**

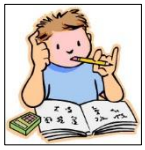
en volgens de wet van Ohm: U = I x R

P = (I x R) x I => **$P = I^2 \times R$**

$P = U^2 / R$

2.1.g Oefeningen: vermogen

Oefening 2:



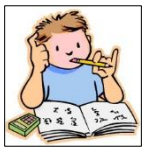
Gegeven:

Een lamp van 6V met een vermogen van 100W.

Gevraagd:

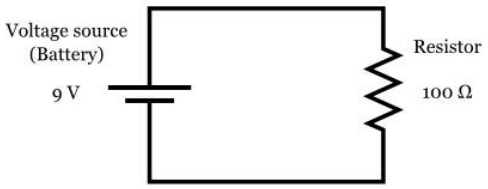
Bepaal de stroom die door de lamp vloeit.

Oefening 3:



Gegeven:

Een spanningsbron levert een spanning van 9V.
Om kortsluiting te vermijden, leggen we een weerstand van 100 Ohm in het circuit.



Gevraagd:

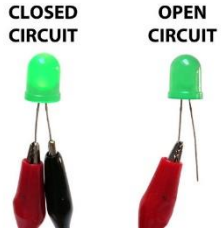
Mag hier een weerstand met een maximaal vermogen van 1/8W (0,125Watt) gebruikt worden?

1) Bereken de stroomsterkte

2) Bereken het vermogen

2.1.h Open en gesloten circuits

Een circuit is gesloten wanneer de stroom van de ene naar de andere pool kan vloeien. De stroom vloeit van de positieve naar de negatieve (GND) pool. De verbruikers in figuur 5 bijvoorbeeld, zijn een LED-lampje en een weerstand. De schakelaar is geen verbruiker. Om er voor te zorgen dat het LED-lampje niet teveel stroom krijgt, wordt er een weerstand in serie opgenomen.



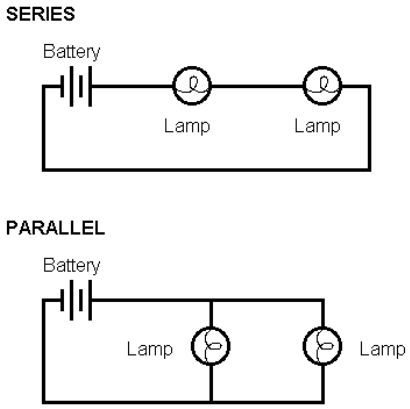
Figuur 3: Open en gesloten circuit

2.1.i Kortsluiting

Wanneer de positieve en de negatieve pool van een spanningsbron rechtstreeks met elkaar verbonden worden, ontstaat kortsluiting. Hierdoor kan de energiebron oververhit geraken, in brand schieten of ontploffen. Om dit te verhinderen worden zekeringen gebruikt om de elektronica te beschermen.

2.1.j Serie en parallelschakeling

Een circuit kan in serie of parallel geschakeld worden. Willen we bijvoorbeeld twee lampjes in het circuit schakelen, kunnen we ze in serie of parallel schakelen.



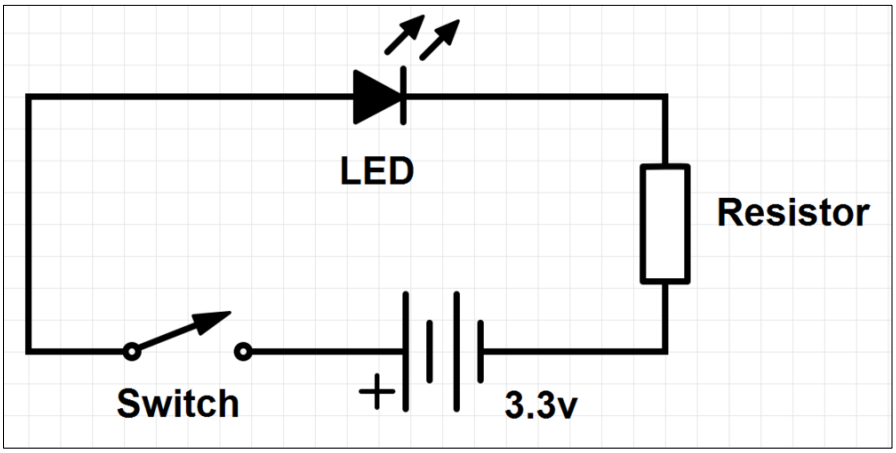
Figuur 4: Serie en parallelschakeling

We spreken van een serieschakeling wanneer de verbruikers of belastingen achter mekaar worden geschakeld. De stroom vloeit gelijktijdig door beide verbruikers en is even groot.

De (deel)spanning die over iedere verbruiker komt de staan, is afhankelijk van de weerstand van die verbruiker.

We spreken van een parallelschakeling wanneer de verbruikers of belastingen, ieder individueel rechtstreeks op de bron worden aangesloten. De spanning over iedere verbruiker is hetzelfde en is gelijk aan de bronspanning. De totale stroom door het circuit is gelijk aan de som van de stromen die door iedere verbruiker vloeit.

2.1.a Het elektrisch circuit



Figuur 5: Eenvoudig elektrisch circuit

In een elektrisch circuit komt het er op aan dat we iedere verbruiker (lamp, motor, LED) de juiste spanning aanbieden, zodat deze optimaal kan werken. Een lamp voor 12V kan je zomaar niet rechtstreeks op de netspanning van 230V schakelen. Op een verbruiker staat meestal aangegeven voor welke spanning hij geschikt is om te werken. Bij elektronische componenten moet men meestal een technische fiche (datasheet) raadplegen om te weten te komen welke spanning, stroom of vermogen een component mag hebben.

In het voorbeeld hierboven willen we een LED schakelen op een batterij. In de datasheet van de LED staat te lezen dat deze maar een maximale stroom mag hebben van 20mA. Om de stroom te beperken tot 20mA nemen we een weerstand (Resistor) op in serie met deze LED. In de datasheet staat ook dat, wanneer er door de LED een stroom vloeit van 20mA, er een spanning over de LED staat van 2,0V.

Zoals eerder aangegeven is de stroom in een serieschakeling gelijk voor alle verbruikers in de keten. De (bron)spanning verdeeld zich over deze verbruikers in verhouding tot hun weerstand.

We kennen de bronspanning: $U_{bron} = 3{,}3V$

We kennen de spanning over de $U_{LED} = 2{,}0V$

De spanning over de weerstand is dan $U_{bron} – U_{LED} = U_R$ ➔ 3,3V – 2,0V = 1,3V

De waarde van de weerstand moet dan zijn: $R = U_R / I_R$ ➔ 1,3V / 0,02A = 65 Ohm

Omdat er in de wereld van de elektronica er naar wordt gestreefd om alles zo klein mogelijk te maken, worden de componenten zo klein mogelijk gebouwd. Afhankelijk van het vermogen worden er grote en kleine weerstanden gebouwd. We moeten dus naast de weerstandswaarde ook nog het vermogen berekenen dat in deze weerstand in warmte wordt omgezet.

Met de formule P = U x I kunnen we dit berekenen: $P_R = U_R \times I_R$ => 1,3V x 0,02A = 0,026 Watt.

Iedere beschikbare weerstand kan dit vermogen omzetten in warmte (dissiperen).

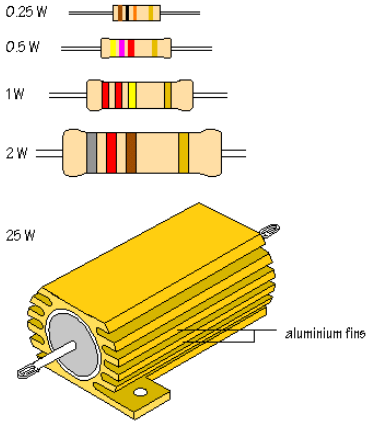
Door meer weerstanden te gebruiken kan men de warmteontwikkeling in een circuit verdelen. Op die manier kan oververhitting dus worden voorkomen. Weerstanden worden geleverd volgens een bepaald maximum vermogen. Wanneer de spanning of stroomsterkte te hoog is, en het vermogen van de weerstand boven het maximum komt, wordt de weerstand te warm en vergroot de kans op beschadiging.

In een circuit met een bronspanning van 12V en met een weerstand van 100 Ohm wordt een vermogen opgewekt van:

$$P = \frac{U^2}{R} = \frac{12^2}{100} = 1.44W$$

Een weerstand met een maximaal vermogen van 1W inschakelen zou dus fataal kunnen aflopen. De weerstand zou opbranden waardoor kortsluiting kan ontstaan.

Een weerstand heeft geen polariteit. Het maakt met andere woorden niet uit in welke richting de stroom door de weerstand stroomt.



Figuur 1: Soorten weerstanden

3. De Raspberry Pi

3.1 De GPIO pinnen

GPIO staat voor General-purpose Input/Output. De GPIO module bestaat uit een reeks van 40 pinnen waarmee externe apparaten kunnen aangestuurd worden. Er zijn 26 'vrije' pinnen die als input of output aangesloten kunnen worden. De overige pinnen dienen voor de stroomaanvoer (+) of als ground-pin (-).

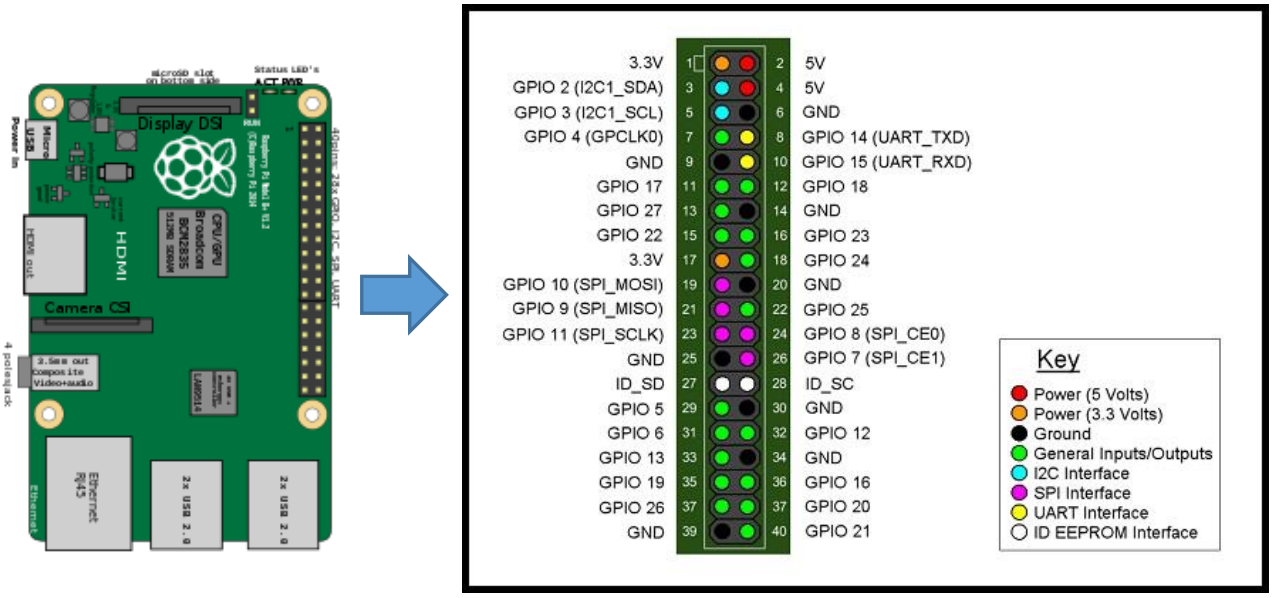
<u>Input:</u>

Een input komt voort van een sensor of als signaal van een ander apparaat.

<u>Output:</u>

De output kan allerhande acties uitvoeren, zoals het aanzetten van een LED of het versturen van een signaal naar een ander apparaat.

<u>Een overzicht van de GPIO:</u>

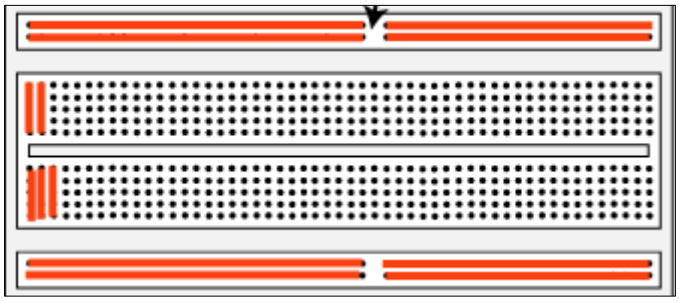


<u>Opgelet:</u>
Het willekeurig in- en uitpluggen van draden en voedingsbronnen kan de Raspberry Pi beschadigen of stukmaken. Men mag geen enkele verbinding maken of onderbreken wanneer het circuit onder spanning staat! Ook het aansluiten van apparaten die veel energie verbruiken, zonder een externe voeding, kan de Raspberry Pi beschadigen.

3.2 GPIO componenten

3.2.a Breadboard

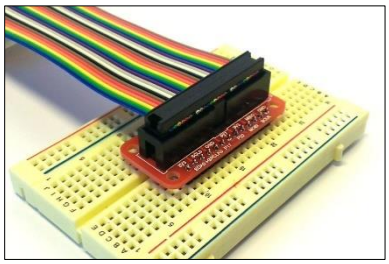
Een breadboard is een bordje, vaak van wit of geel plastic of pertinax, dat gebruikt wordt om elektrische schakelingen tijdelijk op te bouwen. In het bordje zitten per rij gaatjes metalen stripjes die die rijen gaatjes verbinden. In de figuur is te zien hoe de verschillende gaatjes met elkaar verbonden zijn. In het midden is te zien dat de verbinding verbroken is. Wil je de stroom toch over de ganse lengte laten lopen, dan moet je de twee helften met elkaar verbinden.



Figuur 2: Breadboard

<u>Raspberry Pi</u>

We kunnen de GPIO module van de Raspberry Pi verbinden met het breadboard door gebruik te maken van een breakout board en een lintkabel (ribbon cable). Hierdoor kunnen we het breadboard gebruiken om verbindingen te maken met de Raspberry Pi GPIO pinnen.

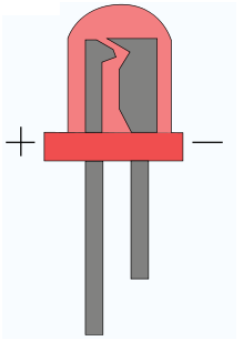


Figuur 3: Breakout board

Het is belangrijk om de lintkabel op de juiste manier te verbinden met het breakout board, zodat de nummers van de pinnen van de Raspberry Pi overeenkomen met die van het breakout board.

3.2.b LED

Een LED of light-emitting diode is een elektronische halfgeleidercomponent die licht uitzendt als er een elektrische stroom in de doorlaatrichting doorheen loopt. Een LED laat de stroom slechts in één richting door. Meestal heeft een LED een lange en een korte pin. De lange pin wordt met de positieve pool verbonden, de korte pin wordt met de negatieve pool verbonden.



Figuur 4: LED

Wanneer er teveel stroom door de lamp vloeit, brandt de lamp even heel fel, waarna ze onmiddellijk stuk brandt. Om de stroomsterkte te verminderen, maken we gebruik van een weerstand.

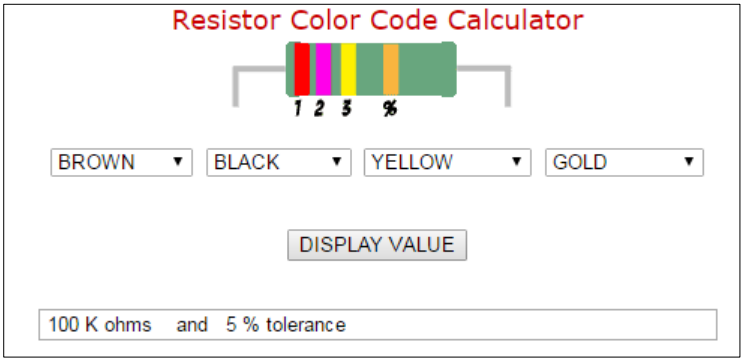
Een normale LED heeft een spanning nodig van 2V en een stroomsterkte van 20 mA. Werken we met een bronspanning van 3.3V, geldt:

R = (3.3V – 2V)/0.02A = 65 Ohm

Vermits deze weerstand niet verkrijgbaar is, nemen we een hogere weerstand die dicht bij deze waarde aanleunt (68 Ohm).

De waarde van de weerstand is af te lezen aan de hand van de kleurenstrips.
Via onderstaande website kan de grootte van de weerstand berekend worden:

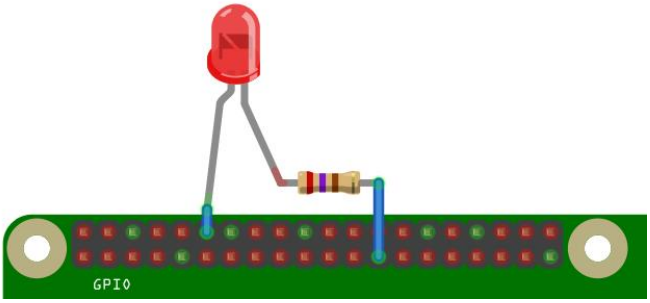
http://www.camradio.net/resistors.html



Figuur 5: Weerstand van 100 000 Ω of 100 kΩ

Raspberry Pi

Connecteren we de LED met de Raspberry Pi, dan moeten we een vrije GPIO pin met de positieve pin van de LED verbinden. De negatieve pin van de LED wordt verbonden met de ground-pin van de Raspberry Pi. Wanneer de GPIO pin op HIGH wordt gezet, gloeit de lamp. Wanneer we de output pin op LOW zetten, vloeit er geen stroom meer door het circuit en gaat de lamp uit.



Figuur 6: LED verbinding met RPi

Figuur 7: LK-LED

We kunnen ook gebruik maken van printplaten met een gemonteerde LED. Hierbij hoeven we ons geen zorgen meer te maken over de weerstand. Deze is immers reeds op de printplaat aangebracht. De positieve pool moet verbonden worden met S1-pin, de negatieve pool wordt verbonden met de ground-pin.

3.2.c OPDRACHT 1

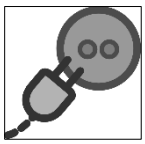
Ontwerp zelf een circuit door gebruik te maken van het breadboard, een LED en een spanningsbron. Wanneer alle componenten juist zijn ingeschakeld, zou het lampje moeten branden. Als spanningsbron gebruik je de batterijhouder, met twee AAA batterijen. Vergeet de juiste weerstand niet in te schakelen!

3.2.d Drukknop

Een drukknop kan gebruikt worden om het circuit te onderbreken, of te sluiten.

Figuur 8: Drukknop

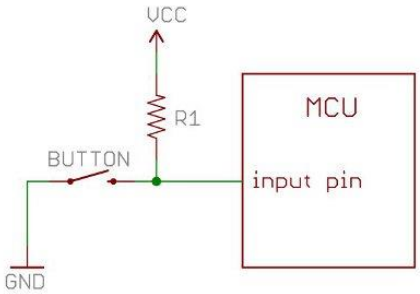
3.2.e OPDRACHT 2



Breid het circuit van opdracht 1 uit door een drukknop in te schakelen. Met een druk op de knop gaat de LED aan.

<u>Raspberry Pi</u>

Een drukknop als input monteren is iets minder vanzelfsprekend. We kunnen de knop niet zomaar verbinden met de input en ground pinnen van de Raspberry Pi, vermits de Raspberry Pi niet zeker weet in welke staat de input pin zich bevindt (HIGH of LOW). Om exact te weten in welke staat de pin zich bevindt, kunnen we de toestand van de pin forceren. We maken hiervoor gebruik van een 'pull up' (HIGH) of 'pull down' (LOW) systeem.



VCC: Voltage Common Connector
= Verbinding met bronspanning (5V of 3V3)

Figuur 9 Voorbeeld van een pull-up

Uit het bovenstaand circuit kunnen we het volgende afleiden:

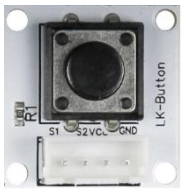
- Als de knop af staat (het circuit is doorbroken), wordt de Vcc met de input pin verbonden. De pin heeft dus een HIGH-toestand.

- Als de knop aan staat, wordt de Vcc met de ground verbonden. De input pin heeft dus een LOW-toestand.

Merk op: de weerstand wordt gebruikt om te vermijden dat kortsluiting ontstaat. Als de knop wordt ingedrukt, worden de positieve (Vcc) en negatieve (ground) pool rechtstreeks met elkaar verbonden!

Dit systeem geeft ons zekerheid over de status van de input pin. De pin staat nu op 'HIGH' als de knop <u>niet</u> is ingedrukt, en op 'LOW' wanneer de knop is ingedrukt.

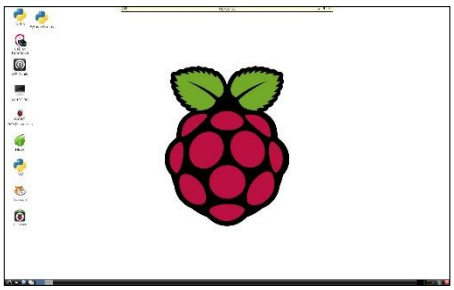
Ook voor de drukknop kunnen we gebruikmaken van een printplaat. Deze printplaat heeft reeds een ingebouwd 'Pull up' systeem. We verbinden de S1 pin met de input pin van de Raspberry Pi, de Vcc met de 5V of 3V3 pin en de ground pin van de drukknop met de ground pin van de Raspberry Pi.



Figuur 10: LK-Button

3.2.f The Shell

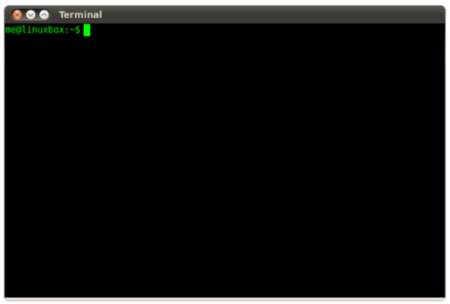
Het besturingssysteem van een Raspberry Pi is een variant van Linux (NOOBS/Raspbian). Werken met dit besturingssysteem is anders dan met het Windows besturingssysteem.



Figuur 11: Rpi GUI

Om opdrachten door te geven aan de computer, maken we meestal gebruik van een GUI (Grafical User Interface). Denk maar aan het bureaublad dat alle pictogrammen bevat van de programma's die we vaak gebruiken.

Ook de Raspberry Pi heeft een grafische interface. Wanneer je een klavier, een muis en een scherm aansluit op de Raspberry Pi, kan je met deze grafische interface werken.



Figuur 12: Linux terminal

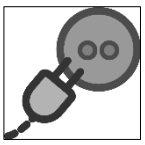
Omdat het vaak makkelijker is om op een computer of laptop te programmeren, kunnen we de Raspberry Pi ook aansluiten op een laptop of desktop. Via een ethernet of Wi-Fi verbinding kunnen we rechtstreeks op het apparaat programmeren. Er bestaan emulators waardoor je op je computer met de grafische interface van de Rpi kan werken. Het is echter eenvoudiger om niet met de grafische interface te werken, maar met de shell.

Een shell is een programma dat commando's van het keyboard ontvangt en deze doorgeeft aan het besturingssysteem. Via de terminal kunnen we interageren met de shell. Dit noemt men ook een command line interface (CLI).

Om de Rpi te verbinden met een Windows computer, maken we gebruik van:

- Een ethernet kabel
- Een programma dat de verbinding verzorgt (PuTTY)
- Een programma dat de grafische weergave van de shell verzorgt (XMing)

3.2.g OPDRACHT 3



- Start het programma XMing op. Rechts onderaan in de taakbalk zou het logo van Xming moeten verschijnen.
- Verbind de computer met de Raspberry Pi door gebruik te maken van een ethernet kabel.
- Zet de raspberry Pi op
- Start het programma PuTTY op. Je krijgt volgend scherm te zien:

  Typ bij 'Host Name' het juiste IP adres in van de Raspberry Pi. Maak gebruik van poort 22 en druk op 'Open'. Als alles goed gaat, krijg je nu de terminal te zien en kan je aanmelden bij de Rpi.

3.2.h Shell Commando's

In de les zullen we slechts een beperkt aantal commando's moeten gebruiken om met de Raspberry Pi te werken. In de lege rijen kan je zelf commando's aanvullen die je vaak gebruikt.

Commando	Betekenis
sudo Idle3	Start Idle3 op
sudo shutdown -h now	Sluit de Raspberry Pi af
sudo reboot	Herstart de Raspberry Pi
Lsusb	Geeft USB connecties met Rpi weer
Ifconfig	Geeft alle netwerkverbindingen weer
cat > filename	Creëer een nieuw bestand
cat filename	Toon de inhoud van een bestand
> filename	Verwijdert inhoud van een bestand
rm filename	Verwijdert bestand
sudo nano filename	Wijzig de inhoud van een bestand
mkdir foldernaam	Creëer een nieuwe map

4. Programmeren met Python

Python is een relatief recente programmeertaal en is begin jaren 90 ontworpen door Guido van Rossum (NL).

4.1 Syntaxis

Python is ontwikkeld met het oog op leesbare code. Structuur wordt aangebracht door indentatie, of regelinspringing in plaats van bijvoorbeeld accolades. Statements (vergelijkbaar met zinnen uit gewone taal) worden simpelweg beëindigd door het eind van de regel. Variabelen krijgen geen typedeclaratie.

Bijvoorbeeld

```
# Importeren van library voor het tonen van willekeurige getallen
import random
win = random.randint(1,3)

print("Welkom bij Let's make a deal. Kies een deur.")
print("[1] Deur Een")
print("[2] Deur Twee")
print("[3] Deur Drie")
deur = int(input("Geef een keuze op:"))

# If-else-blok
if deur == win:
    print("Achter de deur zit een koffer met een miljoen euro. Jij wint!")
else:
    print("Achter de deur bevindt zich een pakezel waarmee je terug naar huis kan gaan. Jij verliest!")

# Ervan uitgaand dat de "win"-variabele hetzelfde is als de opgegeven invoer
...
>> Geef een keuze op:
<< 3
>> Achter de deur zit een koffer met een miljoen euro. Jij wint!
```

➔ Geen type declaratie van de variabelen!

➔ Python werkt met indentaties, blokstatements worden afgesloten met ':'

De basisstructuren van Python komen overeen met de structuren van andere programmeertalen. Gebruik het internet om te verifiëren hoe je een bepaalde structuur integreert in Python.

4.2 Idle3

De IDE (Integrated Development Environment) waarin we Python 3 zullen gebruiken, is IDLE. Dit is gratis te downloaden op www.python.org/. IDLE 3 is reeds geïnstalleerd op de Raspberry Pi. Via de terminal kan je Idle opstarten door het commando 'sudo idle3' te typen.

Om een nieuw script te schrijven, druk je op 'File', 'New File'. Om een bestaand script te openen druk je op 'Open Module'.

4.3 De GPIO module

Om de componenten die aangesloten zijn op het breadboard aan te spreken met Python, bestaat er een aparte GPIO bibliotheek. Deze is reeds gedownload op de Raspberry Pi. Met behulp van deze bibliotheek kunnen we signalen sturen naar de GPIO pinnen, en signalen lezen van de GPIO pinnen.

De volledige documentatie van de GPIO bibliotheek is te vinden op:

https://sourceforge.net/p/raspberry-gpio-python/wiki/

> Voor de Raspberry Pi heeft Python een speciale GPIO module ontwikkeld, de RPIO. Zie:
>
> https://pypi.python.org/pypi/RPIO

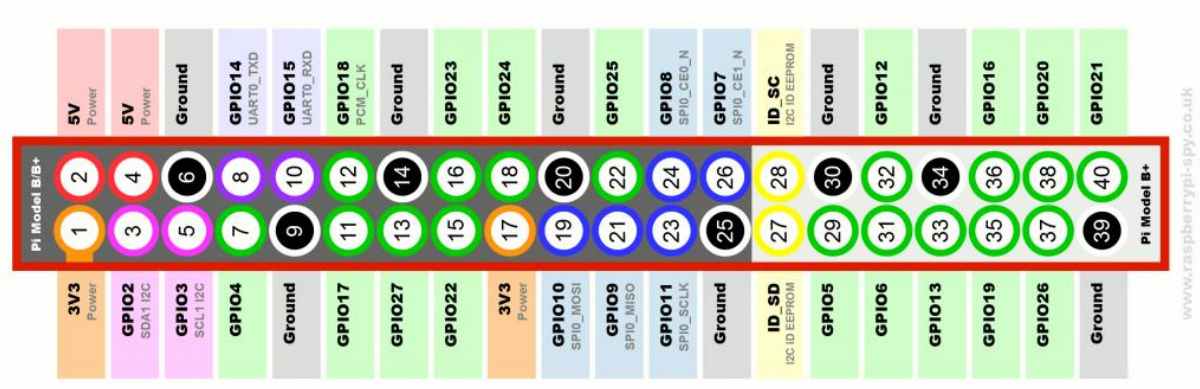
Vooraleer we deze bibliotheek kunnen gebruiken, moet deze geïmporteerd worden in het script:

```
import RPi.GPIO as GPIO
```

'as GPIO' ➔ we benoemen de bibliotheek met GPIO.
Elke keer als we een methode of eigenschap uit de bibliotheek willen aanspreken, gebruiken we deze naam.

4.3.a Pin nummering

Om te weten met welke I/O pinnen we aan het werken zijn, worden de pinnen genummerd. Er zijn twee types van nummering, BOARD nummering en BCM (Broadcom SOC.) nummering.



Figuur 13: BOARD en BCM nummering

Gebruik je BOARD nummering, dan spreken we de I/O pinnen aan met de nummers op het bord (1 t/m 25). Maken we gebruik van BCM nummering, dan refereer je naar de nummers die na het woord 'GPIO' komen (zie bovenstaande figuur). Zo heeft de vierde pin van links te tellen een BOARD nummer van 8 en een BCM nummer van 14. Wij maken gebruik van BCM nummering. Ook dit moet vermeld worden in het script:

```
GPIO.setmode(GPIO.BCM)
```

4.3.b Warnings

Omdat je meerdere scripts of circuits tegelijk kan gebruiken, kan het voorkomen dat de Raspberry Pi waarschuwingen toont. Dit gebeurt wanneer de Rpi.GPIO ontdekt dat een pin anders geconfigureerd wordt dan zijn standaardwaarde. We zetten deze waarschuwingen uit:

```
GPIO.setwarnings(False)
```

4.3.c Pinnen configureren

Vooraleer we een bepaalde I/O pin kunnen gebruiken, moeten we deze configureren in Python. Je moet met andere woorden bepalen welke pin je gebruikt en of deze pin als input of als output gebruikt wordt.

- Voor input:
  GPIO.setup(channel, GPIO.IN) met channel = BCM nummer

- Voor output:
  GPIO.setup(channel, GPIO.OUT)

Eventueel kan je zelfs de standaardtoestand van pin configureren:

```
GPIO.setup(channel, GPIO.OUT, initial=GPIO.HIGH)
```

Meerdere kanalen tegelijk configureren kan ook, door gebruik te maken van een array:

```
chan_list = [11,12]
GPIO.setup(chan_list, GPIO.OUT)
```

Array die alle te configureren kanalen bevat

4.3.d Pinnen gebruiken

- Om de toestand van een pin te lezen, gebruik je:

  ```
  GPIO.input(channel)
  ```

  De toestand is HIGH (1) of LOW (0).
  Deze code werkt ook om de toestand van een **output** pin te lezen!

- Om de toestand van een output pin te configureren, gebruik je:

  GPIO.output(channel, state) met state = HIGH (1) of LOW (0)

  Je kan ook de output van meerdere pinnen tegelijk configureren:

  chan_list = [11,12]
  GPIO.output(chan_list, GPIO.LOW) zet alle pinnen op LOW
  of
  GPIO.output(chan_list, (GPIO.HIGH, GPIO.LOW)) zet de eerste pin op HIGH, de tweede op LOW

4.3.e Cleanup

Wanneer je script is uitgevoerd, moeten de pinnen 'schoongemaakt' worden. Dit wil zeggen dat de toestand van de pinnen terug naar de standaardwaarde wordt gebracht. Dit is vooral om kortsluiting te vermijden, wanneer je niet zeker weet in welke toestand een bepaald pin zich bevindt. Gebruik hiervoor

```
GPIO.cleanup()
```

Wanneer je niet alle pinnen wil schoonmaken, kan je ook kiezen welke specifieke pin schoongemaakt moet worden:

```
GPIO.cleanup(channel)
GPIO.cleanup( (channel1, channel2) )
of GPIO.cleanup( [channel1, channel2] )
```

4.3.f OPDRACHT 4

Maak een circuit met de Raspberry Pi. Je verbindt een LK-Led lampje met pin 18 op het breakout board. Laat je opstelling controleren vooraleer je de Raspberry Pi aanzet! Schrijf vervolgens een script met Python. Je slaat je script op in de map 6BI als opdracht4.py.

Wanneer het script wordt uitgevoerd, wordt de lamp voor twee seconden aangezet. Na twee seconden zet je de lamp terug uit.

Om tussen twee codeerregels te pauzeren, maak je gebruik van de 'time' bibliotheek:

```
import time
```

Met de methode 'sleep' kan je de uitvoering van je script pauzeren:

```
time.sleep(t)     met t = tijd in seconden
```

4.3.g Blokstatements

Ook Python heeft blokstatements. Vooral de **if**, **def** en de **loop**-statements zijn voor ons de belangrijkste. Deze komen in grote mate overeen met wat we reeds gezien hebben in andere programmeertalen.

<u>If…elif…else…</u>

```
x = 5
if x < 0:
     print ('X is negative')
elif x == 0:
     print ('X is zero')
else:
     print ('X is positive')
```

Het programma print "X is positive"

Def

Een def-blok wordt gebruikt om een functie te programmeren.

```
def double( x ):
   Return 2 * x
```

Hierbij is double geprogrammeerd als functie. Roepen we bijvoorbeeld double(2) op, geeft de functie het getal 4 terug.

For loop

```
for x in range(0, 3):
    print "Hello”
```

Het programma print drie keer het woord “Hello” (De bovengrens is exclusief).

While loop

```
count = 0
while (count < 9):
   print ('The count is:', count)
   count = count + 1
```

Het programma print negen keer ‘The count is: …’.

Break/Continue

Met het codewoord ‘**continue**’ kan tijdens de loop een bepaalde actie uitgevoerd worden. Nadat deze actie is uitgevoerd, gaat de loop verder.

Met het codewoord ‘**break**’ wordt de loop vroegtijdig gestopt.

```
count = 0
while (count < 9):
   print ('The count is:', count)
   count = count + 1
   if count == 7:
       print(‘We are almost at 9!’)
       continue
   if count == 8:
       print(‘Let’s stop counting…’)
       break
```

OUTPUT

```
('The count is:', 0)
('The count is:', 1)
('The count is:', 2)
('The count is:', 3)
('The count is:', 4)
('The count is:', 5)
('The count is:', 6)
We are almost there!
('The count is:', 7)
Let's stop counting
```

Wanneer we een while loop gebruiken die nooit ‘onwaar’ wordt, spreken we van een **oneindige lus.**

```
while True:
   num = raw_input("Enter a number  :")
   print ("You entered: ", num)
```

Het programma vraagt een nummer en print vervolgens “You entered [ingevoerd nummer]”. De uitvoering van deze code wordt oneindig keer herhaald.

Een oneindige lus is heel interessant om het programma te laten wachten op een bepaald signaal, zoals bijvoorbeeld een input van een geschakelde component.

4.3.h OPDRACHT 5

Zet de Rpi terug uit met het commando ‘*sudo shutdown –h now*’.
Wijzig het circuit van opdracht 4. Voeg een LK-Button toe op pin 16. Bekijk ***3.2.d drukknop*** om de knop juist te schakelen.

Om de knop te testen, print je eerst “Gedrukt!” in de console wanneer er op de knop wordt gedrukt. Wanneer dit lukt, ga je verder met de oefening. Laat je opstelling controleren vooraleer je de Raspberry Pi aanzet!

Zolang de knop ingedrukt wordt, gaat de lamp aan. Als de knop wordt losgelaten, gaat de lamp uit. Je slaat je script op als opdracht5.py.

<u>Uitbreiding:</u>

Pas je code aan: als je binnen de 10 seconden (als de lamp aan is) nogmaals op de knop drukt, moet de lamp uitgaan. Wanneer binnen de 10 seconden niet op de knop wordt gedrukt, gaat de lamp tevens uit.

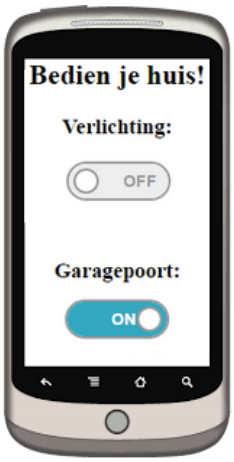
<u>Tip</u>: Gebruik de bibliotheek ‘time’. De methode time.time() geeft de huidige tijd in totaal aantal seconden weer!

5. Domoticaproject

Nu we de basis van de Raspberry Pi onder de knie hebben, zijn we in staat om een groter project uit te werken. Het project dat hier voorgesteld wordt, is een domoticasysteem waarbij verschillende elektrische componenten aangestuurd kunnen worden via een smartphone.

In de eerste plaats zorgen we voor de bediening van de verlichting:

Vervolgens kunnen we ook andere componenten aansturen, zoals bijvoorbeeld de garagepoort:



5.1 HTML templates

Jullie zijn ondertussen vertrouwd met het opmaken van websites met behulp van HTML en CSS. Om de componenten van het huis aan te sturen, kunnen we hiervan gebruik maken.

Ontwerp een formulier in een HTML document dat een submit knop bevat, waarmee we een ingeschakelde LK-LED kunnen bedienen.

Met een klik op de knop zal een actie uitgevoerd moeten worden door de Raspberry Pi (namelijk het aanzetten van de lamp). We zullen er met andere woorden voor zorgen dat de Raspberry Pi als server optreedt.

Om de waarde(n) van je formulier naar de server te sturen, gebruik je een POST method. De opening van je formulier zou er als volgt kunnen uitzien:

<form action="/domotica" ***method="post"*** *name="f1">*

De Python bibliotheek Bottle werkt met templates. Templates zijn HTML pagina's die worden opgeslagen onder de ".tpl" extensie. Sla je HTML pagina op als "domotica.tpl" in de map '/home/pi/6BI/domotica/views'. Een stylesheet kan wel onder de '.css' extensie worden opgeslagen.

<u>TIP:</u>

Ontwerp je HTML pagina op de laptop (met NetBeans of Notpad++).
Ga via de Rpi console naar '/home/pi/6BI/domotica/views' en maak een nieuw bestand aan met het commando '*cat > domotica.tpl*'. Plak de HTML code in dit bestand. Sla op en sluit af (Crl +C). Doe hetzelfde voor je stylesheet.

5.1.a Uitbreiding

Je zou in plaats van een submitknop ook een radiobutton of een checkbox kunnen gebruiken. Dit is nodig wanneer je meerdere knoppen in je formulier wil bedienen. Een formulier heeft immers maar één submit knop.

Vermits het formulier dan verzonden moet worden met een druk op een checkbox of radiobutton, voeg je onderstaande code toe in het element:

```
<input type="checkbox" onChange="this.form.submit()" >
```

5.2 Bottle library

Om het controlepaneel zichtbaar te maken, moet de Raspberry Pi het HTML document versturen naar de aanvrager (client). De Rpi fungeert met andere woorden als server. De Bottle bibliotheek van Python maakt dit heel eenvoudig. Deze bibliotheek is reeds geïnstalleerd op de Raspberry Pi.

De volledige documentatie van Bottle is te vinden op http://bottlepy.org/.

Open het bestand bottleServer.Py. Onderstaande code krijgen jullie gegeven. Hieronder wordt stap voor stap uitgelegd wat deze code precies betekent. Vervolgens wordt uitgelegd hoe we het script verder kunnen uitwerken.

2

3

1

```
def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)


    #Configureer hier je input en outputs

    #Configureer hier je routes



    try:
        run(host='192.168.0.220', port=8080, debug = True)
    except:
        print('error')
    finally:
        GPIO.cleanup()


from bottle import *
import RPi.GPIO as GPIO
main()
```

1

De eerste indentatie (de meest links uitgelijnde code) wordt steeds eerst uitgevoerd. In deze lijnen worden de opgeladen Vervolgens voeren we de functie main() uit. Deze functie bevat het hoofdscript.

2

In de main functie configureren we eerst en vooral de GPIO pinnen. Dit komt overeen met wat we reeds gezien hebben in de vorige oefeningen.

3

Try:
Hier wordt de Raspberry Pi opgestart als server.
Onder 'host' komt het host adres van de Raspberry Pi, onder 'port' de gebruikte poort. Dit is hetzelfde adres dat je gebruikt om via de laptop aan te melden op de Rpi (zie *0*

The Shell*).* Met *Debug = True* zorgen we ervoor dat foutmeldingen worden weergegeven in de browser. Wanneer je script klaar is voor gebruik, kan dit eventueel op 'False' worden gezet.

Except:
Wanneer er iets misgaat met het opstarten van de server, printen we de foutboodschap.

Finally:
De GPIO pinnen worden schoongemaakt. Deze code wordt altijd uitgevoerd nadat Try of Except worden uitgevoerd.

Voer het script uit. De server is opgestart:

```
Bottle v0.12.9 server starting up (using WSGIRefServer())...
Listening on http://192.168.0.220:8080/
Hit Ctrl-C to quit.
```

Op dit moment kunnen we de server reeds aanspreken via elk toestel dat geconnecteerd is met het lokale netwerk. Vermits we nog niet gecodeerd hebben wat er moet gebeuren wanneer de server wordt aangesproken, krijgen we een *'404 Not Found'* melding:

5.2.a Routes maken

Met een route kunnen een actie koppelen aan de URL die door de client in de webbrowser wordt ingetypt. Het is deze URL dat de server duidelijk zal maken welke code er moet uitgevoerd worden. Een route wordt als volgt ontworpen:

```
@('/routenaam')
```

Op dezelfde indentatie als de route, schrijf je de code die uitgevoerd moet worden als de klant surft naar http://host:poort/routenaam. Deze code schrijf je best in functies.

Bijvoorbeeld:

```
@route('/domotica')
def toon():
        print("Hallo!!!")
```

Surfen we nu naar http://192.168.0.220:8080/domotica[2], wordt het woord 'Hallo' geprint in het Idle output scherm.

We willen echter onze HTML pagina te zien krijgen in de browser van de client.
Bottle gebruikt hier de 'template()' functie voor.

Vervang de print functie in bovenstaande code met de volgende regel:

```
return template('domotica.tpl')
```
[3]

Je krijgt nu je webpagina te zien in de webbrowser!

Tot slot zorgen we ervoor dat de client de knop op de webpagina kan bedienen. Hiervoor maken we een nieuwe route aan. Deze route wordt enkel uitgevoerd in de 'POST' modus. Dit wil zeggen wanneer op de submit knop wordt gedrukt, en het formulier wordt ingediend. In je HTML code heb je reeds gecodeerd welke route er aangesproken moet worden bij het indienen van het formulier:

*<form action="/**domotica**" method="post" name="f1">*

Bijvoorbeeld:

```
@route('/domotica', method='POST')
def toon():
        print("Je hebt op de knop gedrukt!!!")
```

Druk je nu op de knop, zie je de zin 'Je hebt op de knop gedrukt!!!' verschijnen in het Idle output scherm. Vergeet niet om ook bij deze route je template naar de client te sturen!

5.2.b Stylesheets opladen

Wanneer je gebruik maakt van CSS bestanden, kan je een route ontwikkelen om deze op te laden.

```
@get('/<filename:re:.*\.css>')
def stylesheets(filename):
        return static_file(filename, root='…')
```

Onder root="…" komt het pad waar je stylesheet zich bevindt. In dit project is dit steeds '/home/pi/6BI/domotica/views'.

---

[2] Dit IP adres kan uiteraard verschillen.

[3] Om het absoluut pad of een pad naar een andere map dan 'views' te hanteren, gebruik je: TEMPLATE_PATH.insert(0, '*pad naar HTML documenten*')

Vergeet de stylesheets ook niet in het HTML document op te laden!

5.2.c Uitbreiding

<u>De waarde van een HTML element oproepen.</u>

Om de waarde (value) van een HTML element in je script te gebruiken, maak je gebruik van een GET request.

Bijvoorbeeld:

```
HTML: <input type="text" name="txtNaam">
```

Tom

```
Python:     naam = request.forms.get('txtNaam')
```

De waarde van de variabele 'naam' is in dit geval 'Tom'

<u>Een waarde naar een HTML element verzenden.</u>

Om een waarde naar een HTML element te versturen, gebruik je de volgende syntaxis.

```
template('template.tpl', {'variabele': 'Waarde'})
```

Willen we de knop bijvoorbeeld op 'disabled' zetten eenmaal er op gedrukt is, coderen we het volgende:

```
@route('/domotica', method='POST')
def toon():
        return template(domotica.tpl, {'status': 'disabled'})
```

Deze variabele moet uiteraard opgeroepen worden in het HTML document. Om een variabele in een HTML document op te roepen, maak je gebruik van dubbele accolades {{…}}.

Zorg ervoor dat je de variabele op de juiste plaats zet! Een 'disable' attribuut komt altijd in het element:

```
<input type="submit" {{status}}>
```

De eerste keer dat de website geladen wordt (niet de POST methode) moet een lege status worden doorgestuurd:

```
return template(domotica.tpl, {'status': ''})
```

De collectie kan ook meerdere variabelen bevatten:

```
template(domotica.tpl, {'status': 'disabled', 'color': 'green'})
```

5.2.a OPDRACHT 6

DEEL A

Werk het script van het bestand bottleServer.py verder uit zodat je met een druk op de HTML submit-knop een LK-LED lampje kan laten branden.

DEEL B

Voeg een tweede knop toe waarmee je de lamp kan uitzetten. Hierbij kan je niet meer met een submit knop werken. Er is steeds één submit knop per formulier. Werk bijvoorbeeld met een checkbox. Zie *5.1.a Uitbreiding.*

DEEL C

Wanneer je een checkbox aanklikt, herlaadt de pagina. Merk op dat de checkbox 'Aan' is uitgevinkt, terwijl de lamp aanstaat. Dit is onlogisch. Je kan dit probleem oplossen door via het Python script de checkbox op 'checked' te zetten wanneer de pagina herladen wordt. Op deze manier is één checkbox voldoende. Aanvinken zorgt dat de lamp aan gaat, uitvinken zorgt dat de lamp uit gaat. Zie *0*
Onder root="…" komt het pad waar je stylesheet zich bevindt. In dit project is dit steeds '/home/pi/6BI/domotica/views'.

Vergeet de stylesheets ook niet in het HTML document op te laden!

Uitbreiding.

DEEL D

Ontwerp een stylesheet voor de webpagina (button.css) en sla dit bestand op in de 'view' folder. Dit bestand moet zowel in je HTML als in je Python bestand opgeladen worden. Maak in je CSS bestand het <h2> element rood. Zie *5.2.b Stylesheets opladen.*

Wanneer je merkt dat deze opmaak werkt, kan je je opmaak verder uitbouwen. Op het internet vind je tal van mogelijkheden om je checkbox op te maken. Kijk bijvoorbeeld op http://cssdeck.com/labs/css-checkbox-styles.
Enkele voorbeelden:

Uitbreiding

De oplossing van bovenstaande opdracht is niet de meest logische en efficiënte. Er zijn twee verbeteringen mogelijk:

- Wanneer we de lamp op 'AAN' zetten en het controlepaneel een tweede keer openen in een nieuw venster, merken we op dat de lamp toch nog op 'UIT' staat in het nieuwe venster. Dit probleem is op te lossen door je checkbox op 'checked' te zetten afhankelijk van de status van de output pin (HIGH of LOW). De status van je checkbox is met andere woorden niet meer afhankelijk van een klik op deze checkbox, maar

afhankelijk van de status van de output pin.

Ontwerp een functie statusControle() die 'checked' terug geeft als de output pin op HIGH staat, en die een lege waarde teruggeeft als de pin op LOW staat. Roep deze functie op bij het versturen van je template:

```
return template('domotica.tpl', {'status': statusControle()})
```

Bekijk *'4.3.d Pinnen gebruiken'* om de waarde van de output pin te achterhalen.

- Een verbetering is mogelijk in de IF-structuur. Wanneer we een IF-structuur gebruiken om de waarde van één variabele aan te passen afhankelijk van een bepaalde conditie, maken we het best gebruik van een ternaire operator.

  <u>Voorbeeld</u>

  ```
  X = 4 if b > 8 else 9
  ```

  X krijgt waarde 4 als b > 8, anders krijgt het de waarde 9

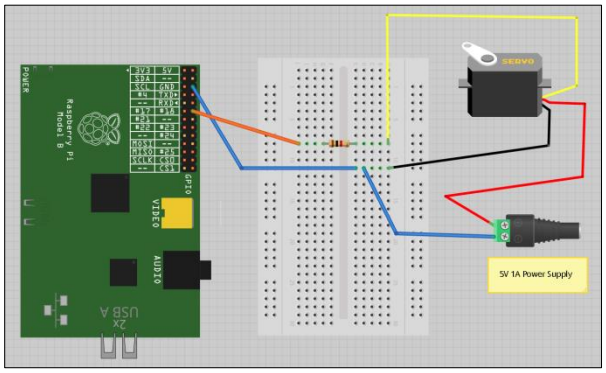
Pas bovenstaande verbeteringen toe in je oefening.

5.3 Servomotoren

Servomotoren zijn kleine elektrische motoren die een draaibeweging kunnen uitvoeren in twee richtingen. De motoren waarmee we werken kunnen 60 graden naar links en 60 graden naar rechts draaien.

De servomotoren vereisen een spanning van 4,8V tot 6V (zie technische fiche!). Het is mogelijk om een servomotor te voeden via de 5V pin van de Raspberry Pi. Dit zou echter voor problemen kunnen zorgen, en teveel stroom onttrekken uit het circuit. Hierdoor zou de Raspberry Pi plots kunnen uitvallen. Het is met andere woorden aangeraden om de motor met een externe voedingsbron te verbinden.

De servomotor heeft drie aansluitingen: één voor de positieve (rood), één voor de negatieve (zwart) en één voor het inputsignaal (wit). De juiste opstelling wordt weergeven in de volgende figuur:



Figuur 14: Servo motor inschakelen

1) Verbind de negatieve pool van de motor met de negatieve pool van de bron *en met de ground van de Raspberry Pi.*
2) Verbind de positieve pool van de motor met de positieve pool van de bron.
3) Verbind de witte draad (singaal) met een vrije pin op de Raspberry Pi.

De weerstand is niet noodzakelijk, maar zorgt ervoor dat de microprocessor beschermd wordt indien de servomotor stuk gaat.

5.3.a Werking van een servomotor

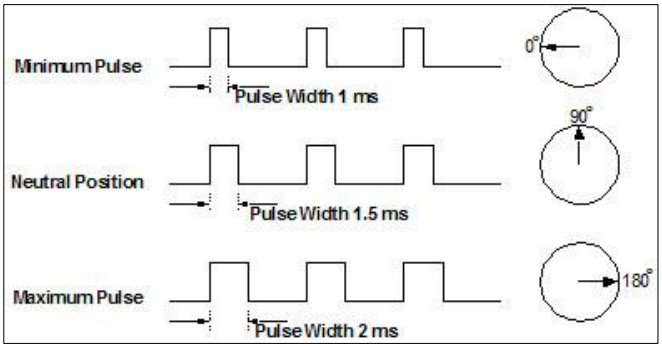
Een servomotor werkt op basis van puls-stromen van variabele lengte. De motor ontvangt deze puls-signalen in een vast tijdsinterval. Aan de hand van deze signalen, weet de motor wat hij moet doen.

De grootte van het puls-signaal wordt uitgedrukt in milliseconden(ms).
Een minimum puls-grootte is meestal 1ms, de maximum puls-grootte 2ms.

De 'midden' positie van de servomotor is die positie waarin de motor nog evenveel naar links als naar rechts kan draaien. Dit wordt bereikt bij een puls-grootte van 1.5ms.

Een puls die kleiner is dan 1.5ms zorgt voor een beweging in tegenwijzerzin, een puls groter dan 1.5ms zorgt voor een beweging in wijzerzin.

Zelfs indien de motor niet hoeft te bewegen, moet de puls binnen het tijdsinterval herhaald worden. Op die manier weet de servomotor dat hij in de huidige positie moet blijven.



Figuur 15: Puls-signalen voor een servomotor die 180° kan draaien.

5.3.b Aansturen van een servomotor

Om de servomotor aan te sturen in Python, maken we gebruik van de PWM (Pulse Width Modulation) module.

Om een PWM object te ontwerpen, geven we de GPIO pin door waarmee de motor is verbonden én de frequentie van de pulsering. Deze frequentie is af te lezen op de technische fiches van de motor. De frequentie geeft weer hoe snel de signalen naar de motor worden verstuurd, en wordt weergegeven in Hertz. Er geldt:

Herz = 1/s met s in seconden

Dus wanneer de puls om de 20 milliseconden verstuurd moet worden, geldt:

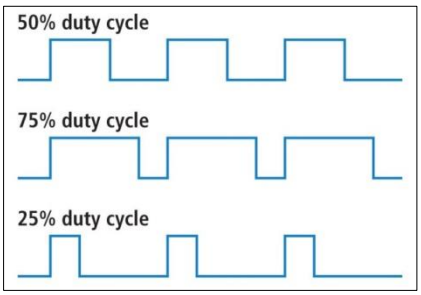
1 / 0.020s = 50Hz

Je kan dit ook makkelijk online berekenen:
https://www.unitjuggler.com/convert-frequency-from-ms(p)-to-Hz.html

In Python:

```
servo = GPIO.PWM(17,50)     GPIO pin = 17 en frequentie = 50Hz
```

Om de startpositie van de motor te bepalen, moeten we de grootte van de puls doorgeven. Dit gebeurt niet in milliseconden, maar door de 'duty cycle' te bepalen. De duty cycle is de procentuele grootte van de puls ten opzichte van de totale cyclus. De kleinste duty cycle is nul (geen puls), de grootste duty cycle bedraagt 100. De volgende figuur maakt dit duidelijk:



Figuur 16: Duty cycle

Dus in Python:

```
servo.start(50)      voor een duty cycle van 50%, of een puls-grootte van 1.5ms.
```

Om de positie vervolgens te veranderen, gebruiken we

```
servo.ChangeDutyCycle(100)

                     voor een duty cycle van 100%, of een puls-grootte van 2ms.
```

Wanneer het programma eindigt, moeten we ook de pulsering stoppen:

```
servo.stop()
```

Het versturen van pulsen kan ook voor andere componenten worden gebruikt. Op deze manier zou je bijvoorbeeld een lamp kunnen laten ‘dimmen’. Een duty cycle van 0 zou zorgen dat de lamp niet brandt, een duty cycle van 1 zou zorgen dat de lamp volledig brandt, en een duty cycle tussen 0 en 1 zou ervoor zorgen dat de lamp ‘minder fel’ brandt.

5.3.c OPDRACHT 7

Schakel een lkLed lampje in op pin 16 en open het bestand ‘PWM_voorbeeld.py’

Wat gebeurt er als je het script uitvoert?

……

Welke code zorgt voor het verminderen van de lichtsterkte?

……

Druk ctrl-c in het consolescherm om het script te beëindigen.

5.3.d OPDRACHT 8

DEEL A

Schakel de Raspberry Pi uit en sluit de servo motor aan in het circuit op pin 16. De juiste opstelling vind je in “*5.3.b Aansturen van een servomotor”.*

Open het bestand “Servo_voorbeeld.py”. In dit script is een schuifbalk toegevoegd om de positie van de servomotor te bepalen.

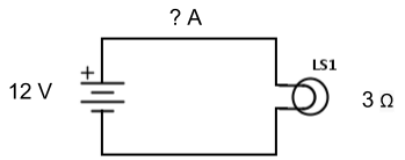
DEEL B

Sluit de servomotor van het speelgoedhuis aan in je circuit.

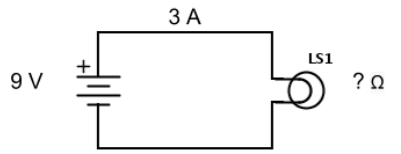
Werk het script van het bestand bottleServer.py verder uit zodat de linkerdeur van het huis open (toe) gaat met een druk op de knop in het HTML-controlepaneel.

6. Oplossingen

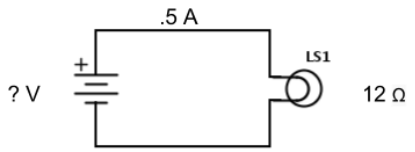
6.1 Oefening 1



$I = \frac{U}{R} = \frac{12}{3} = 4A$



$R = \frac{U}{I} = \frac{9}{3} = 3\ Ohm$



$U = I \times R = 0.5 \times 12 = 6V$

6.2 Oefening 2

$I = \frac{P}{U} = \frac{100}{6} = 16{,}67\ A$

6.3 Oefening 3

Bereken de stroomsterkte
I = V / R = 9 V / 100 Ω = 90 mA

Bereken het vermogen
P = U x I = 9 x 0.09 = 0.81 Watt

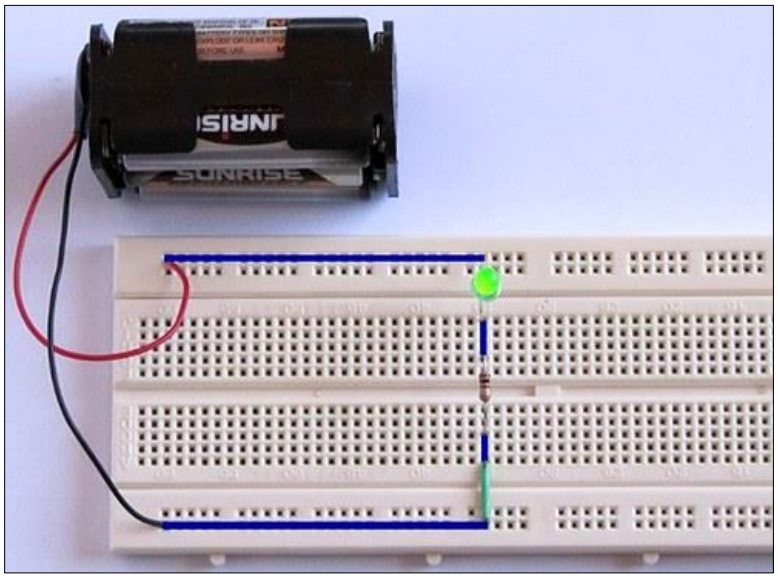
Alternatief:

$P = \frac{U^2}{R} = \frac{81}{100} = = 0.81\ Watt$

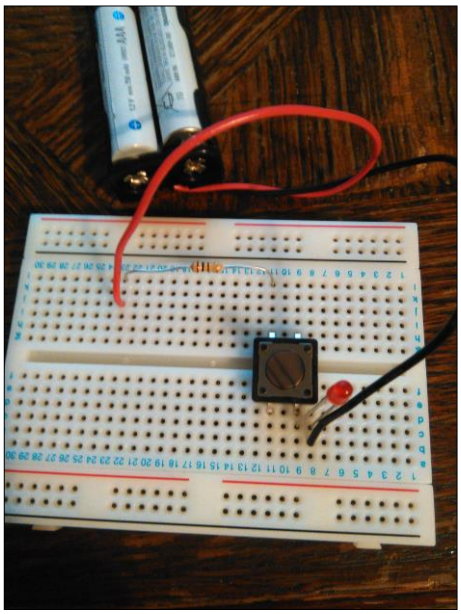
<u>Oplossing:</u>

De weerstand van 1/8W is onvoldoende, een weerstand met een maximaal vermogen van 1W is aangewezen.

6.4 Opdracht 1



6.5 Opdracht 2



6.6 Opdracht 4

```
import RPi.GPIO as GPIO
import time

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

PIN=18
GPIO.setup(PIN,GPIO.OUT)


GPIO.output(PIN,1)
time.sleep(2)
GPIO.output(PIN,0)


GPIO.cleanup()
```

6.7 Opdracht 5.A

```
import RPi.GPIO as GPIO
import time

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

IN = 16
OUT = 18
GPIO.setup(IN, GPIO.IN)
GPIO.setup(OUT, GPIO.OUT)

while True:
    if GPIO.input(IN):
        GPIO.output(OUT,1)
    else:
        GPIO.output(OUT,0)


GPIO.cleanup()
```

6.8 Opdracht 5.B

```
import RPi.GPIO as GPIO
import time

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


IN = 16
OUT = 18
GPIO.setup(IN, GPIO.IN)
GPIO.setup(OUT, GPIO.OUT)
GPIO.output(OUT,0)

while True:
    if GPIO.input(IN):
        start = time.time()
        GPIO.output(OUT,1)
        time.sleep(1)

        while True:
            if time.time() - start < 10 and GPIO.input(IN):
                GPIO.output(OUT, 0)
                time.sleep(1)
                break
            elif time.time() - start > 10:
                GPIO.output(OUT, 0)
                time.sleep(1)
                break

GPIO.cleanup()
```

6.9 Opdracht 6.A

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Domotica</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>

    <body>
        <h2>Bedien je huis!</h2>

        <form action="/domotica" method="post" name="f1">

        <h3>Verlichting: </h3>
            <label for="lampAan">AAN</label>
            <input type="submit" name="lampAan" id="lampAan">

        </form>


    </body>
</html>
```

Python

```
def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(18, GPIO.OUT, initial = GPIO.LOW)

    @route("/domotica")
    def toon():
        return template('domotica.tpl')

    @route("/domotica", method='POST')
    def toon():
        GPIO.output(18,1)
        return template('domotica.tpl')


    try:
        run(host='192.168.0.220', port=8080, debug = True)

    except:
        print('error')
    finally:
        GPIO.cleanup()


from bottle import *
import RPi.GPIO as GPIO
TEMPLATE_PATH.insert(0,'/home/pi/6BI/domotica/views')
main()
```

6.10 Opdracht 6.B

Wijziging in HTML

```
<label for="lampAan">AAN </label>
<input type="checkbox" name="lampAan"id="lampAan" onChange='this.form.submit()'>

<label for="lampUit" onChange='this.form.submit()'>UIT </label>
<input type="checkbox" name="lampUit"id="lampUit" onChange='this.form.submit()'>
```

Wijziging in Python

```
@route("/domotica", method='POST')
def toon():
    lampAan = request.forms.get('lampAan')
    lampUit = request.forms.get('lampUit')

    if lampAan:
        GPIO.output(18,1)
    elif lampUit:
        GPIO.output(18,0)
```

6.11 Opdracht 6.C

Wijziging in HTML

```
<form action="/domotica" method="post" name="f1">

    <h3>Verlichting: </h3>
    <label for="lampAan">Lamp </label>
    <input type="checkbox" name="lampAan"id="lampAan" {{status}} onChange='this.form.submit()'>

</form>
```

Wijziging in Python

```
@route("/domotica", method='POST')
    def toon():
        lampAan = request.forms.get('lampAan')

        if lampAan:
            GPIO.output(18,1)
            status = 'checked'
        else:
            GPIO.output(18,0)
            status = ''
        return template('domotica.tpl', {'status': status})
```

6.12 Opdracht 6.D

Wijziging in HTML

```
<head>
    <title>Fire My Light</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="button.css">
</head>
```

Wijziging in Python

```
static_file("button.css" , root='/home/pi/6BI/domotica/views')
```

of:

```
@get('/<filename:re:.*\.css>')
def stylesheets(filename):
    return static_file(filename, root='/home/pi/6BI/domotica/views')

stylesheets("button.css")
```

6.13 Opdracht 6 - Uitbreiding

```
def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(18, GPIO.OUT, initial = GPIO.LOW)
    static_file('button.css', root='/home/pi/6BI/domotica/views')

    def statusControle():
        return 'checked' if GPIO.input(18)else ''

    @route("/domotica")
    def toon():
        return template('domotica.tpl', {'status': statusControle()})

    @route("/domotica", method='POST')
    def toon():
        lampAan = request.forms.get('lampAan')

        value = 1 if lampAan else 0
        GPIO.output(18, value)

        return template('domotica.tpl', {'status': statusControle()})


    try:
        run(host='192.168.0.220', port=8080, debug = True)

    except:
        print('error')
    finally:
        GPIO.cleanup()


from bottle import *
import RPi.GPIO as GPIO
main()
```

7. Nuttige links

Basis elektronica

- http://www.instructables.com/id/Basic-Electronics/step15/Breadboards/

Raspberry Pi documentatie

- https://www.raspberrypi.org/documentation/

Python documentatie

- https://docs.python.org/3.3/