Raspberry Pi processen regelen met tkinter.

In deze tkinter workshop experimenteren we met de Scale widget.

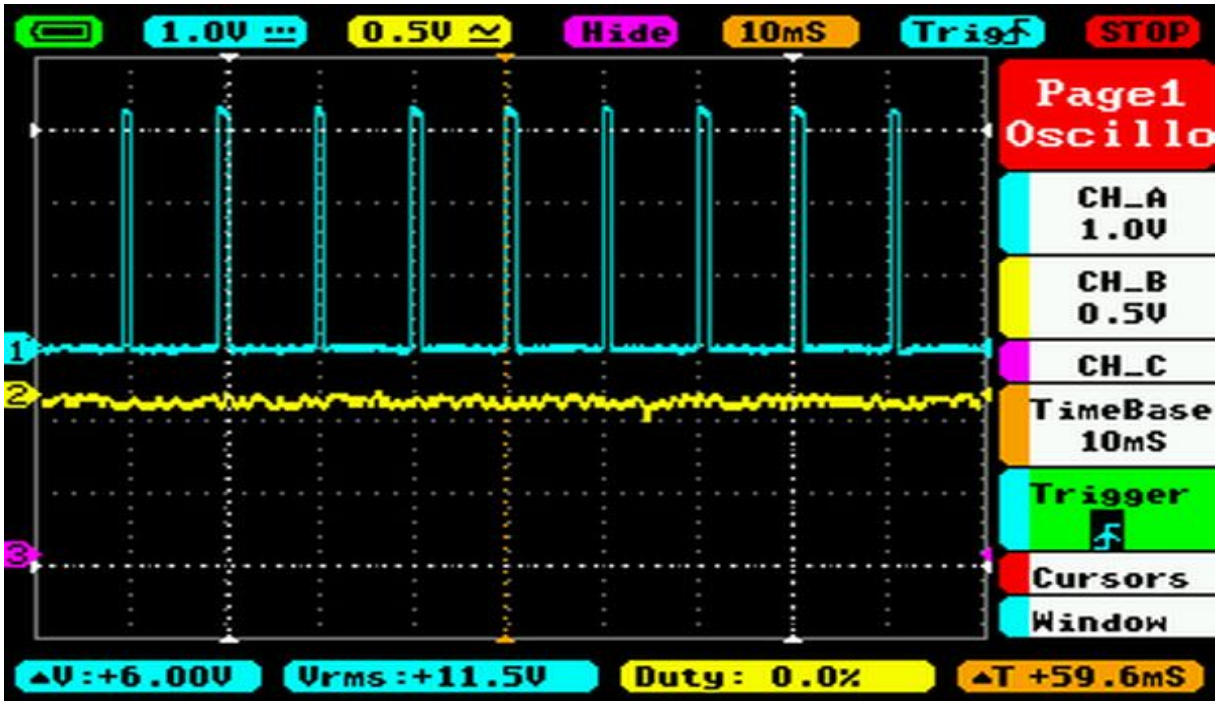
De scale widget geeft de mogelijkheid variabele instellingen te doen.

We zullen de scale widget gebruiken in combinatie met de PWM functie van de GPIO om de helderheid van een LED in te stellen en om het toerental van een ventilator te regelen.

PWM staat voor Pulse Width Modulation

Raspberry Pi processen regelen met tkinter.

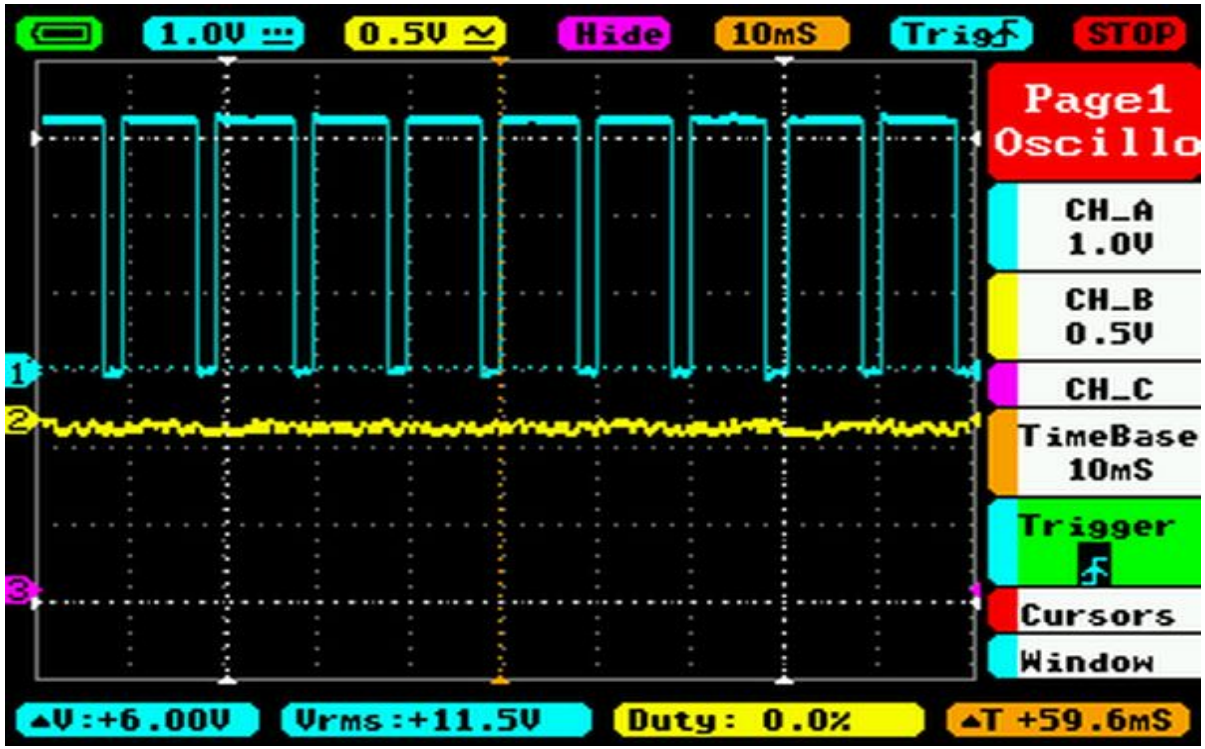
Wat is PWM ? Bij PWM regelen we de pulsbreedte van een periodiek pulssignaal. De gemiddelde waarde van de geleverde energie kunnen we zo regelen. De verhouding hoog tot de totale periodetijd is de duty cycle (dc).De (dc) is hier 15 % van de periodetijd.



Raspberry Pi processen regelen met tkinter.

Bij Pulse Width Modulatie zijn twee parameters van belang:

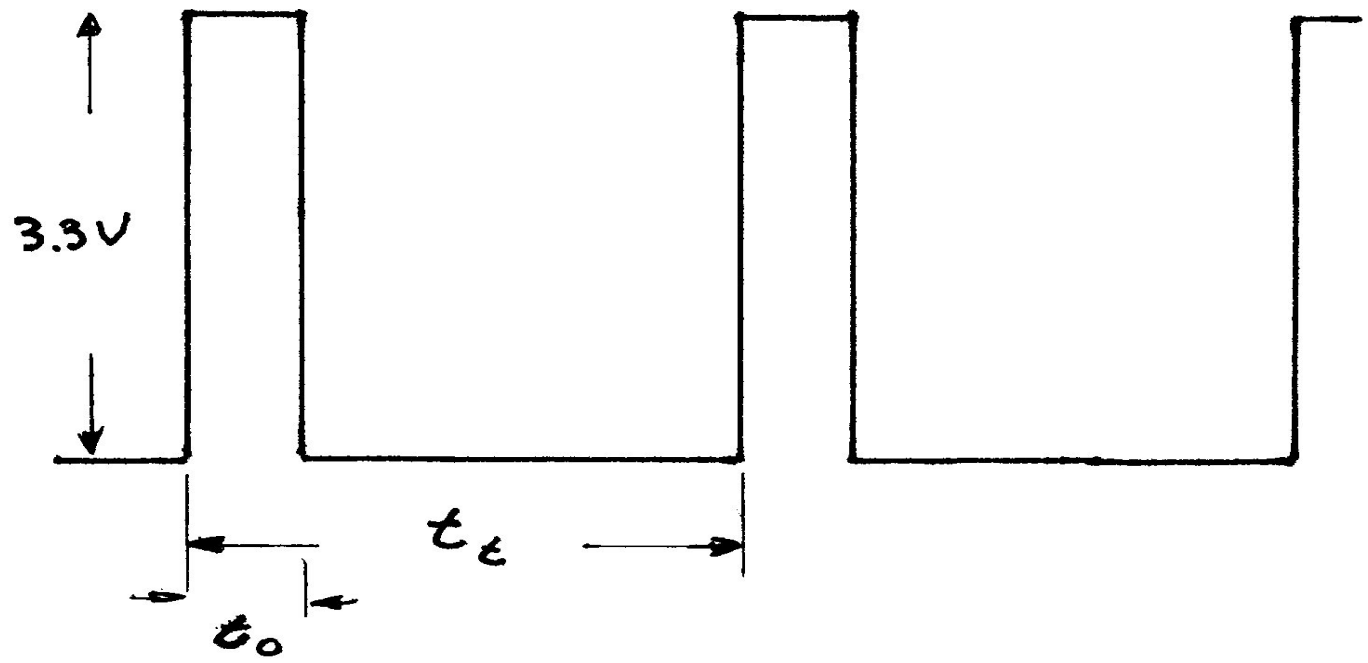
- Frequentie, (f) het aantal malen dat een cyclus herhaald wordt per seconde.
- Duty Cycle (dc) de verhouding tussen de tijd dat pulse hoog is en de tijdsduur van de hele cyclus.



dc is 80 %

Raspberry Pi processen regelen met tkinter.

Voorbeeld van de grootheden in een PWM output.



t_t is de totale periode lengte uitgedrukt in seconden.
t_o is de tijd dat de puls on, (hoog) is.
t_o / t_t keer 100 is de duty cycle uitgedrukt in procenten.
In het voorbeeld is t_t = 10 millisec. en t_o = 2 millisec.
De duty cycle (dc) is 0,2 X 100 = 20 %.

Raspberry Pi processen regelen met tkinter.

De Python PWM implementatie voor de RPi kent de volgende functies:

```
p = GPIO.PWM (channel, frequency)
p.start ( dc ) # duty cycle van 0.0 tot 100.0
p.ChangeFrequency ( freq ) # frequency in Hz
p.ChangeDutyCycle ( dc) # dc van 0.0 tot 100.0
p.stop ( ) # stopt de PWM mode
```

Nadat PWM gestopt is blijft het channel nog wel geactiveerd. Voor een goede afsluiting moet het channel gereset worden. Dat gebeurt met de functie:

```
GPIO.cleanup( ) # De GPIO is daarna niet meer actief.
```

Raspberry Pi processen regelen met tkinter.

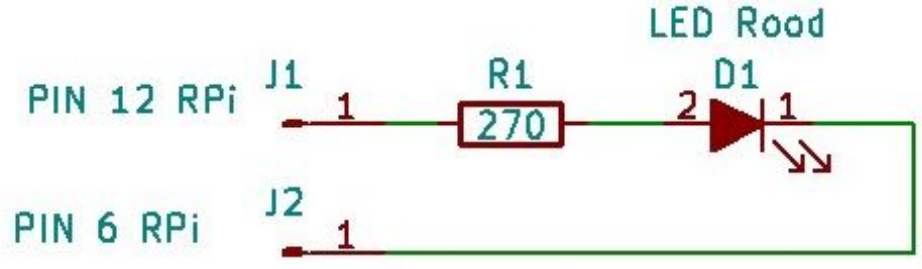
Demo programma Pulse Width modulatie

Sluit een LED met een serieweerstand van 270 – 330 ohm
aan op pin 12 van de GPIO bus.

De serieweerstand komt aan de plus zijde van de LED en
wordt aangesloten op pin 12 van de GPIO.
De min kant van de LED (korte draad) komt aan pin 6 (GND).

Voer het script in de Thonny IDE in dat staat op de volgende dia.
Sla het script op onder de naam pwm_test.py
Voer het script uit met F5.

Experimenteer met de waarden voor frequentie en dc.



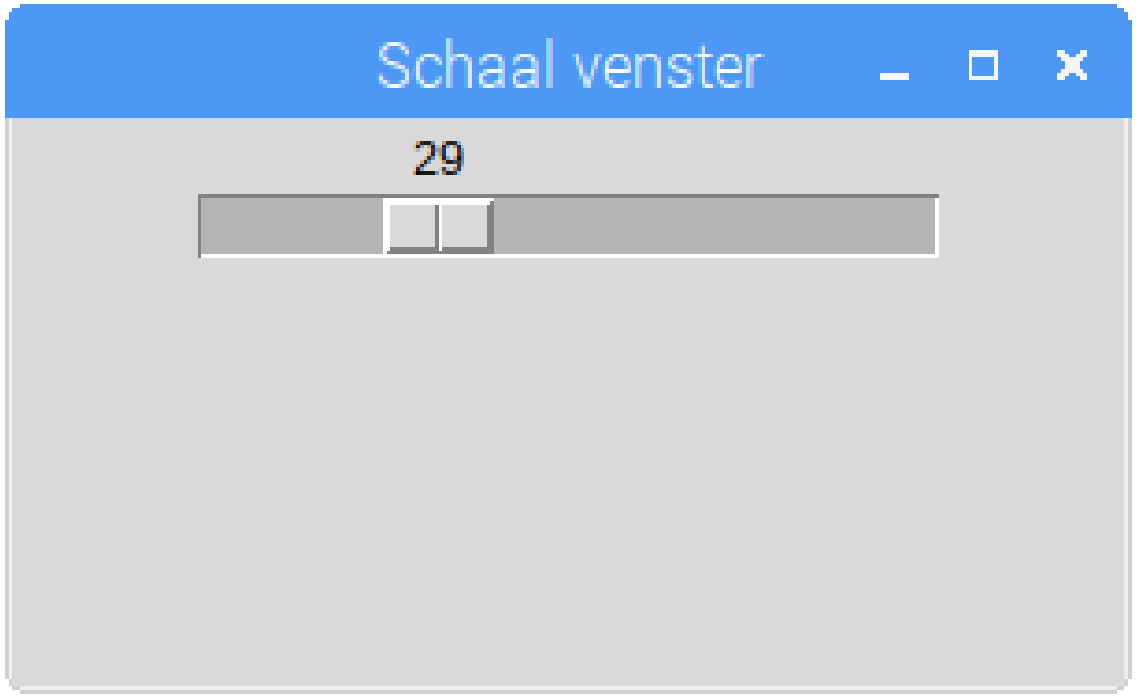
Raspberry Pi processen regelen met tkinter.

Demo programma Pulse Width modulatie

```
# 01_pwm_test_01.py

import RPi.GPIO as GPIO # Importeer de GPIO module met als naam GPIO.
GPIO.setmode (GPIO.BOARD) # het programma gebruikt boardpin nummering.
GPIO.setup (12,GPIO.OUT) # Initialisatie van pin 12 als output.
p = GPIO.PWM (12, 1) # De PWM mode wordt gestart met pin 12 als output.
# en een frequentie van 1 Hz.
p.start (10) # De PWM functie start met een duty cycle van 10%.
Input ("Druk op Enter om te stoppen:") # Aanwijzing hoe je moet stoppen.
p.stop () # stop functie geactiveerd via de Enter toets.
GPIO.cleanup () # Zet de pinnen van de GPIO terug naar de basis toestand.
```

Raspberry Pi processen regelen met tkinter.

Tkinter Scale widget



Raspberry Pi processen regelen met tkinter.

De Scale Widget kent de volgende atributen:

- lenght =, is de x of y dimensie van de schaal voor respectievelijk de horizontale en verticale schaal.
- orient =, de orientatie van de schaal, default is verticaal. Horizontaal moet je specificeren.
- from_=, minimum waarde, to= maximum waarde van de schaal.
- width=, de breedte van het schuifje.

  En verder alle opmaak atributen.

Raspberry Pi processen regelen met tkinter.

De waarden gegenereerd door de Scale en Entry widgets zijn string variabelen.

Veel toepassingen hebben een integer of float variabele nodig.

Een getal in string vorm kunnen we converteren naar een integer of float waarde.

a = int(str) of a = float(str)

Raspberry Pi processen regelen met tkinter.

Print de actuele waarde van de schuif1 Scale widget.

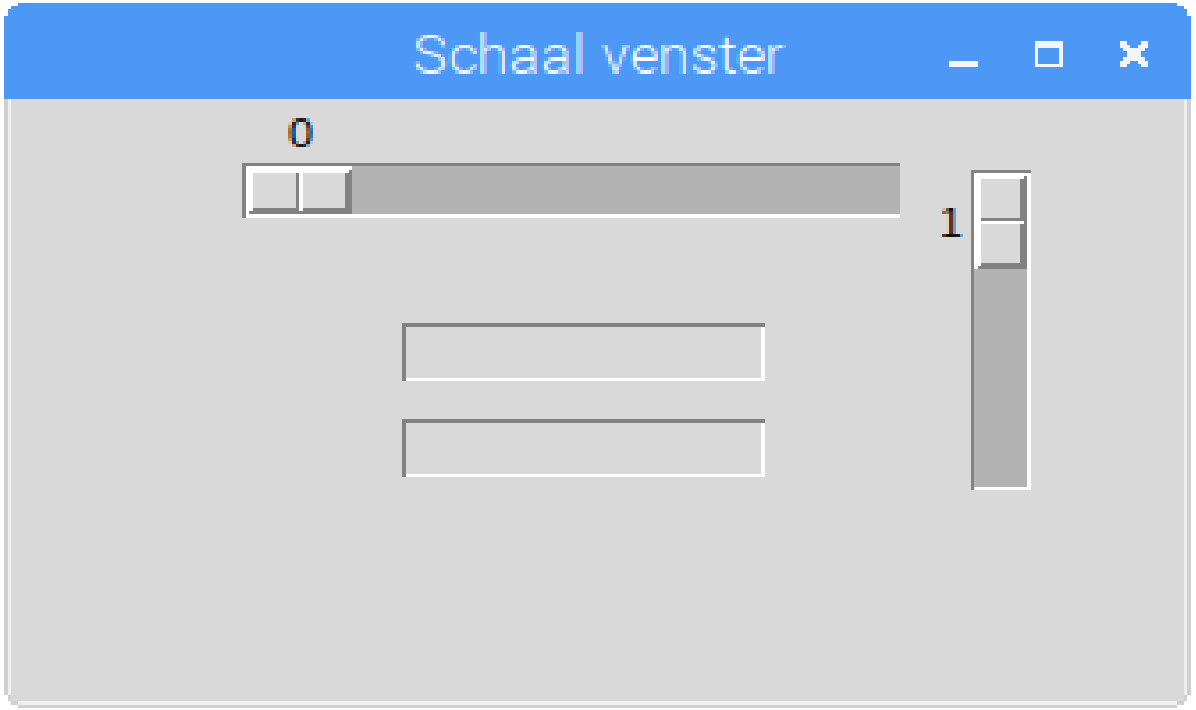
```
# 02_slider_output_01.py
# het programma print de output van de scale widget in de Shell

from tkinter import *

def print_waarde(str): # print functie waarnaar het commando verwijst
    waarde = int(str) # de string output van de widget geconverteerd naar een integer
    print (waarde)

venster = Tk()
venster.title( "Schaal venster")
venster.geometry( "300x150+400+400")
schuif1 = Scale(venster, orient= HORIZONTAL, length = 200, from_= 0, to= 100,
command = print_waarde).pack()
venster.mainloop()
```

Wijzig de regel in de functie print_waarde in: waarde = int(str) in waarde = float(str) en kijk water gebeurt als je het programma runt.

Raspberry Pi processen regelen met tkinter.

```
#03_slider_output_3.py
from tkinter import *

def print_waarde_1(str):
    mom_waarde = "DutyCycle: " + str + " %"
    label1.config(text = mom_waarde)

def print_waarde_2(str):
    waarde = "Frequency: " + str + "Hz"
    label2.config(text = waarde)

venster = Tk()
venster.title( "Schaal venster")
venster.geometry( "300x150+400+400")
schuif1 = Scale(venster, orient= HORIZONTAL, length = 200, from_0,to=100,
command = print_waarde_1).pack()
schuif2 = Scale(venster, length = 100, from_ = 1, to = 25,
command = print_waarde_2).place(x = 250, y = 20)
label1 = Label(venster, relief = SUNKEN, width = 14)
label1.place(x = 110, y = 70)
label2 = Label(venster, relief = SUNKEN, width = 14)
label2.place( x= 110, y = 100)

venster.mainloop()
```

Raspberry Pi processen regelen met tkinter.



Het programma op de vorige dia levert dit resultaat op.

Lamp dimmer.

We gaan nu de momentele waarde van de schuif gebruiken om een proces te regelen.

We combineren de PWM functie van de RPi met de de waarden gegenereerd door de tkinter Scale widget.

De *p.ChangeDutyCycle(dc)* functie die we gebruiken om te regelen, heeft floating waarden nodig.
De Scale widget levert string waarden, we moeten dus een conversie van string naar floating toepassen.

Ook moeten we het proces kunnen beeindigen, daarvoor is een stop toets nodig. Neem het script lamp_dimmer_01.py over van de USB stick.

Raspberry Pi processen regelen met tkinter.

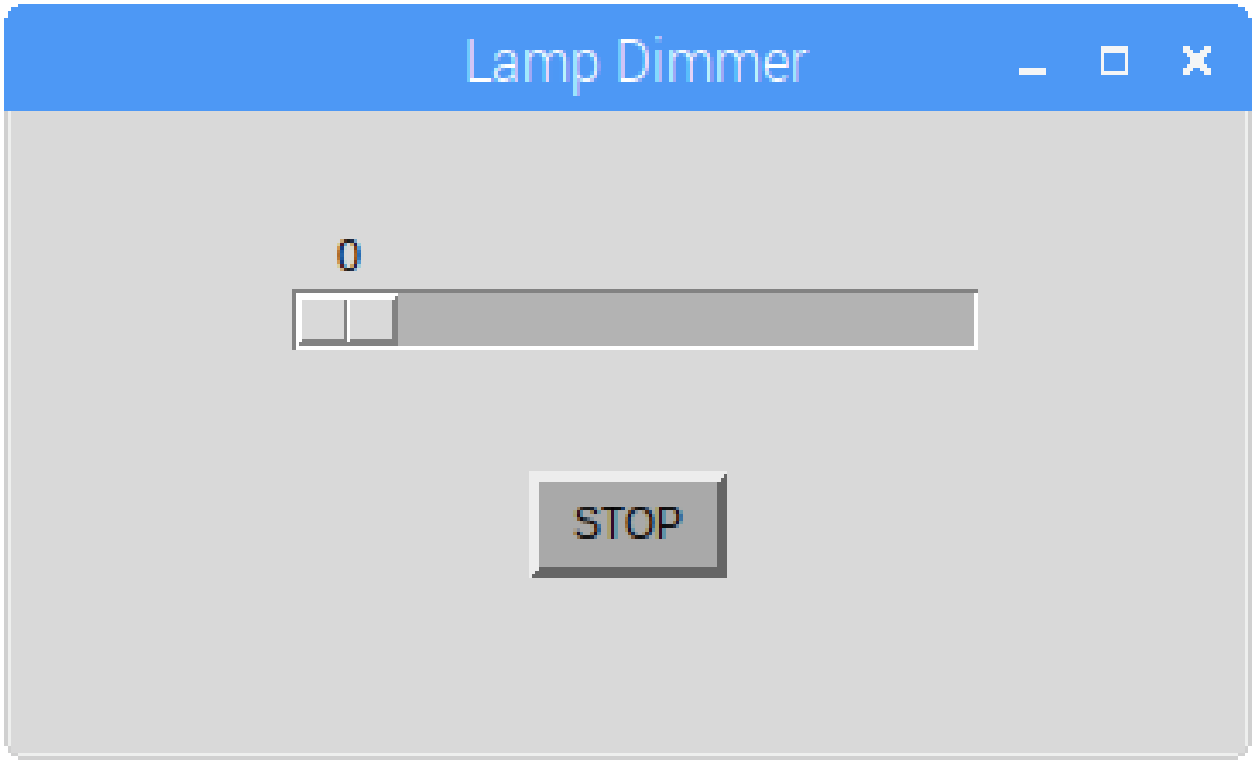
Als het programma is opgeslagen in het geheugen van je RPi, open het dan met de Thonny IDE.

Het programma bestaat uit een aantal onderdelen:

- **De GPIO en de PWM modules.**
- **De tkinter module waarmee het venster en de Scale en Button widget gemaakt zijn.**
- **Een tweetal functies die het interface vormen tussen de tkinter Scale functie en de PWM functie.**

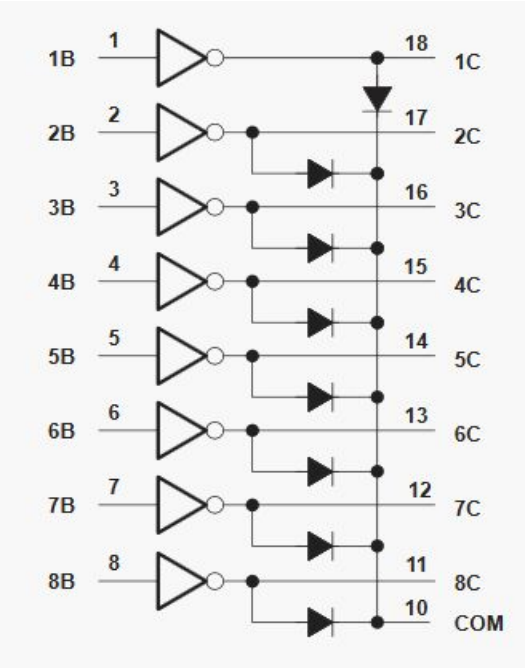
Raspberry Pi processen regelen met tkinter.

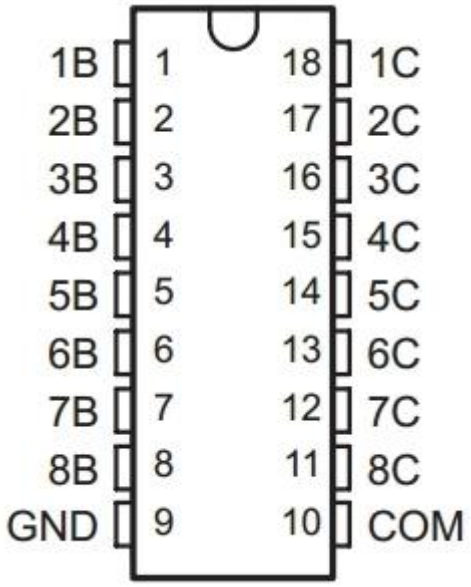
Als je het programma start moet dit het resultaat zijn. Controleer of je helderheid van de LED kunt regelen, en of je met de STOP toets de LED kunt uitschakelen.

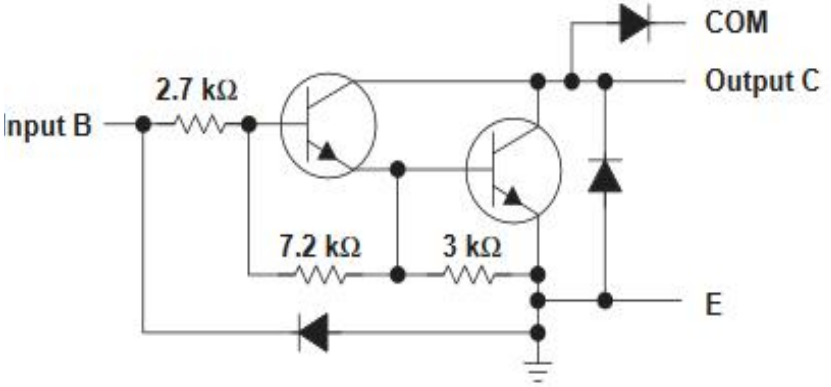


Raspberry Pi processen regelen met tkinter.

Grotere vermogens kunnen we leveren met externe energiebronnen en vermogens transistors als schakelaar. Een veel toegepaste chip is de ULN2803.



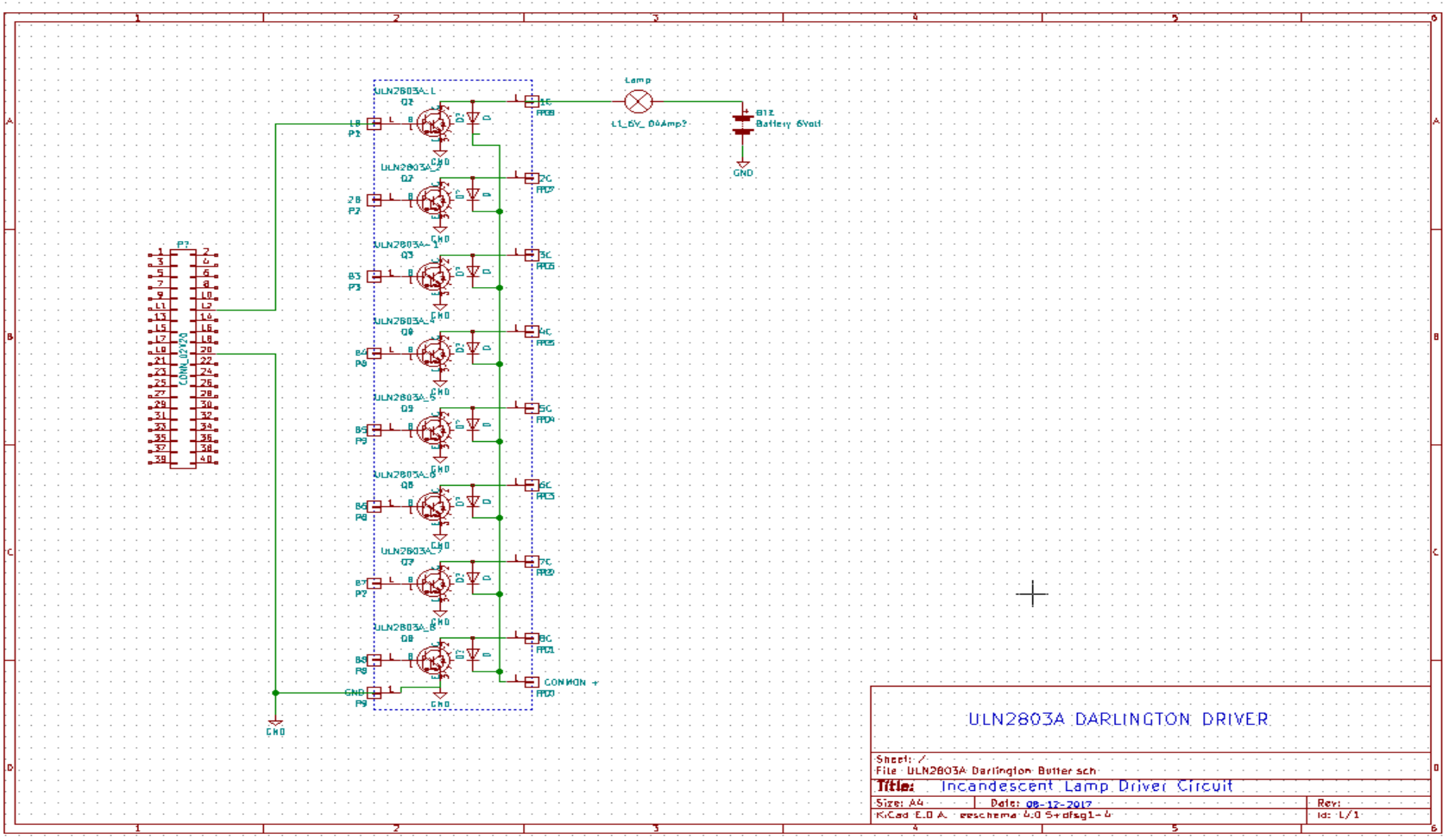




ULN2803 functioneel diagram, layout en principe schema.

Raspberry Pi processen regelen met tkinter.

ULN2803A als lamp driver.



Raspberry Pi processen regelen met tkinter.

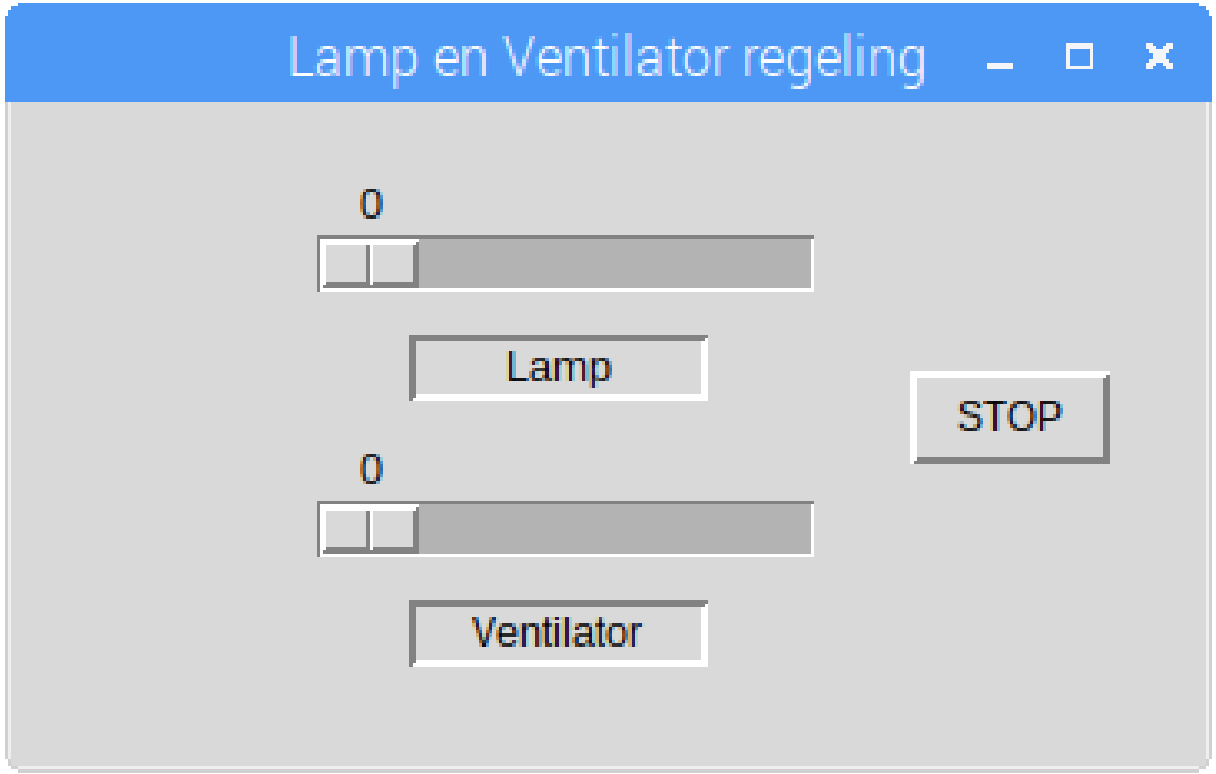
```
#lamp_vent_reg.py deel1

import RPi.GPIO as GPIO  # Zie GPIO cheatsheet voor GPIO instellingen.
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)

p = GPIO.PWM(12,100)     # PWM output op pin 12, frequency 100 Hz.
p.start(1)

v = GPIO.PWM(16, 20)
v.start (0)


def lamp_cntrl(str):
    dc = float(str)
    p.ChangeDutyCycle (dc)

def run_vent(str):
    tt = float(str)
    v.ChangeDutyCycle(tt)

def stop():
    p.stop()
    v.stop()
    GPIO.cleanup()
```

Raspberry Pi processen regelen met tkinter.

```
#lamp_ven_reg.py deel2

from tkinter import *
venster = Tk()
venster.title( "Lamp en Ventilator regeling")
venster.geometry( "360x200+400+400")

schuif1 = Scale(venster,orient= HORIZONTAL, length = 150, from_= 0, to= 100,  command = lamp_cntrl)
schuif1.place(x =90, y = 20)
schuif2 = Scale(venster, orient = HORIZONTAL,from_ = 0, to = 100, length = 150, command = run_vent)
schuif2.place(x = 90, y = 100)
stopbtn = Button(venster, text = "STOP", relief = RAISED, borderwidth = 2, command = stop)
stopbtn.place(x = 270, y = 80)
label1 = Label(venster, text = "Lamp",relief = SUNKEN,borderwidth = 2, width = 12)
label1.place(x = 120, y = 70)
label2 = Label(venster, text = "Ventilator", relief = SUNKEN, borderwidth = 2, width = 12)
label2.place( x = 120, y = 150)

venster.mainloop()
```

Raspberry Pi processen regelen met tkinter.

Het toerental van een DC Motor regelen met PWM en tkinter.
Het regelbereik loopt van 10 tot 100 procent dutycycle.
De minimum waarde is noodzakelijk om de motor te starten.
Laad het script lamp_vent_reg.py om lamp en ventilator te regelen.



Raspberry Pi processen regelen met tkinter.

Script voor toerental regelen van een DC electromotor of ventilator.

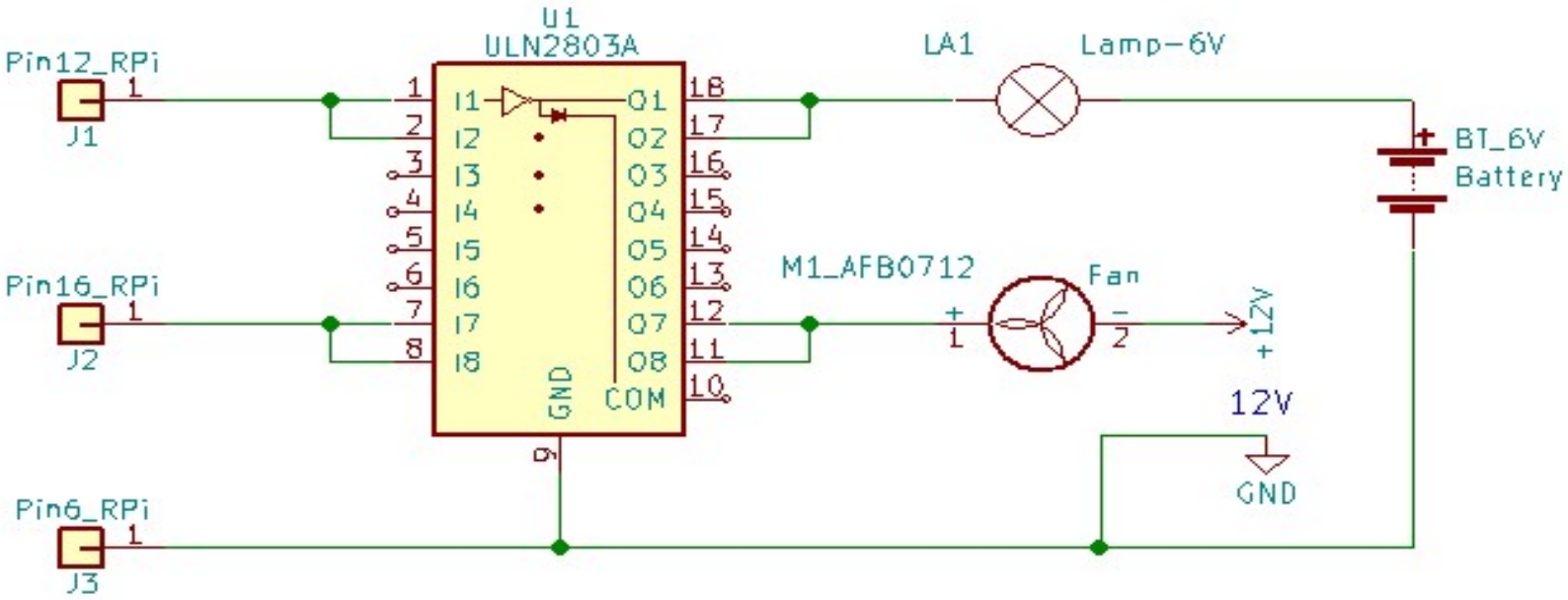
Gebruik is gemaakt van de ULN2803 octal darlington chip om het vereiste vermogen te kunnen leveren.

Bij stromen van meer dan 200 mA is het verstandig om twee darlingtontransistors parallel schakelen.

Zie het schema op de volgende dia.

Raspberry Pi processen regelen met tkinter.

ULN2803A als buffer/driver tussen Rpi en gebruikers.



Raspberry Pi processen regelen met tkinter.

```
# Python 3.5.3 (/usr/bin/python3)
# Demo programma tkinter Scale widget  in combinatie met PWM via de GPIO.
# Met de schuifregelaar is het toerental van de motor in te stellen
#speed_control.py
import RPi.GPIO as GPIO  # Zie GPIO cheatsheet voor GPIO instellingen.
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12,200)     # PWM output op pin 12, frequency 200 Hz.
p.start(0)               # Start PWM met een duty cycle van 0 %

def change_dc( str):
    dc = float(str) # Ingevoerde waarde wordt geconverteerd naar een float variabele
    p.ChangeDutyCycle(dc)

def motor_stop(): # Functie waarmee de Pulse Width Modulator wordt gestopt
    p.stop()          # De feitelijke stop functie
    GPIO.cleanup() # Het GPIO interface wordt weer in de maagdelijke toestand gebracht
    print("Motor is uitgeschakeld") # Bericht dat Pulse Width Modulator is uit geschakeld.
# Om weer te starten moet het script opnieuw uitgevoerd worden
```

Raspberry Pi processen regelen met tkinter.

Deel 2 van speed_control script

```
#
from tkinter import *
venster = Tk( )
venster.geometry( "360x180+400+400")
venster.title("DC Motor Speed Control.")
schaal = Scale(venster, orient = HORIZONTAL, from_ = 25, to = 100, length = 200,
command = change_dc)
schaal.place( x = 80, y = 30)

btn_stop =Button(venster, text = "STOP", bg = "Dark Grey", width = 4,
relief = RAISED, borderwidth = 3, command = motor_stop).place(x = 150, y = 100)

venster.mainloop()
```