NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

Designnotat

Tittel: Turtallsindikator	
Forfattere: Øyvind Skaaden	
Versjon: 2.0	Dato: 12. september 2019

Innhold

---

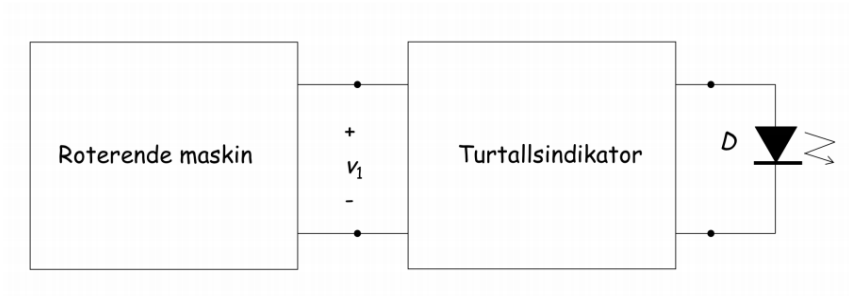
1 Problembeskrivelse

Overvåking og styring av fabrikker og prosessanlegg er et viktig anvendelsesområde for elektronisk-systemdesign. I slike installasjoner finnes ofte motorer og andre roterende innretninger, og det kan være viktig å sørge for at disse operere med riktig turtall. I dette notatet skal det beskrives en løsning på en turtallsindikator som gir et varsel når turtallet for en innretning er for lavt.

Vi skal ta for oss design av systemet som vist i Figur 1.

Kretsen skal

- Ta inn et pulstog med driftssyklus 50% på v_1
- Ha en lysdiode på utgangen, som begynner å lyse dersom omdreiningshastigheten er lavere enn en oppgitt ω



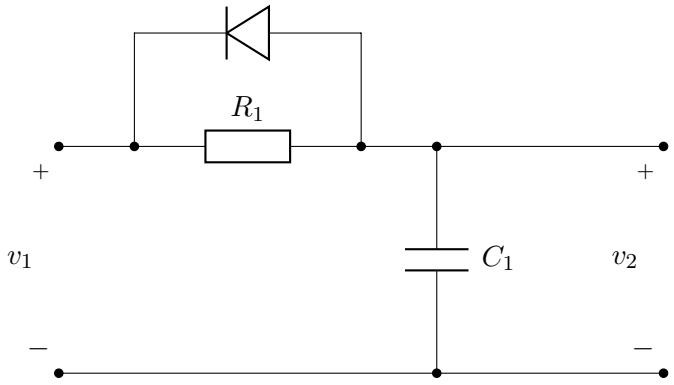
Figur 1: Blokkskjema for kretsen.

2 Prinsipiell løsning

Det er to hovedmåter å designe en krets som skal måle turtallet. Den ene baserer seg på en analog krets med elementer som bruker en viss tid på å lade seg opp. Den andre baserer seg på en digital krets, f.eks. en mikrokontroller, som måler omdreiningshastigheten. Dette notatet vil ta for seg den analoge måten å gjøre det på, men det blir lagt ved en smakebit på den digitale måten i vedlegg A.

Vi ønsker en krets som bruker litt tid på å lade opp et element. Vi kan da ta utgangspunkt i en kondensator. Med den kan vi styre hvor lang oppladningstid og utladningstid vi ønsker. Dersom vi ønsker at kondensatoren skal lade seg raskt opp, kan vi ha en diode, pekende inn mot kondensatoren, i parallell med motstanden i en RC-krets. Dersom vi ønsker rask utladning, kan vi snu dioden.

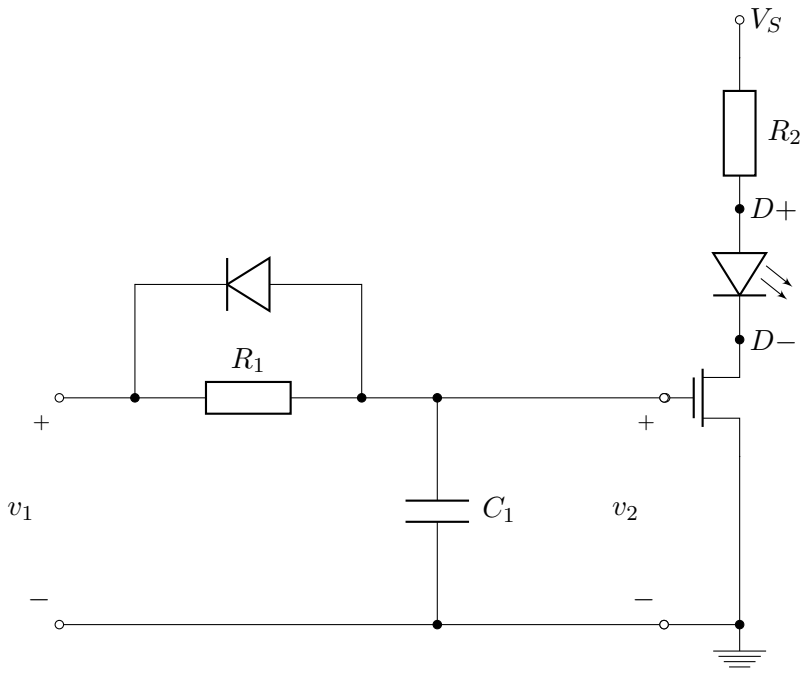
Se skjema for kretsen i Figur 2.



Figur 2: Foreslått krets for omdreiningsteller

Denne kretsen kan ta inn et pulstog med variabel driftssyklus inn på v_1, og det kommer ut et nærmest "sagtann"-signal ut på v_2, se Figur 4. Det er fordi kondensatoren lades normalt opp med tidskonstanten tau, $\tau = R \cdot C$, men lades spontant ut ned til ca 0.7V, som er terskelspenning for dioden. Kondensatoren vil deretter lades normalt ut igen.

Dersom vi deretter kombinerer dette med en transistor, slik at vi kan styre større strømmer vi kretsen se ut som i Figur 3.



Figur 3: Skjema for omdreinigsmåler med transistor

Dersom omdreiningshastigheten er større enn grenseverdien vil ikke spenningen over C_1, v_2, bli tilstrekkelig for å "åpne" transistoren. Når frekvensen er lavere enn grensen, vil kondensatoren kunne lade seg opp til terskelspenningen, V_T, til transistoren, og transistoren vil "åpne" seg.

Motstanden R_2 har som oppgave å begrense strømmen til led-en. Den trenger ikke å være der om det trengs en større strøm ut fra transistoren.

For å regne ut verdier til R_1 og C_1 må vi først finne ut hvilken tidskonstant vi trenger.

Vi vet at vi trenger en spenning på v_2 lik terskelspenningen til transistoren, $v_2 = V_T$, for at transistoren skal åpne seg.

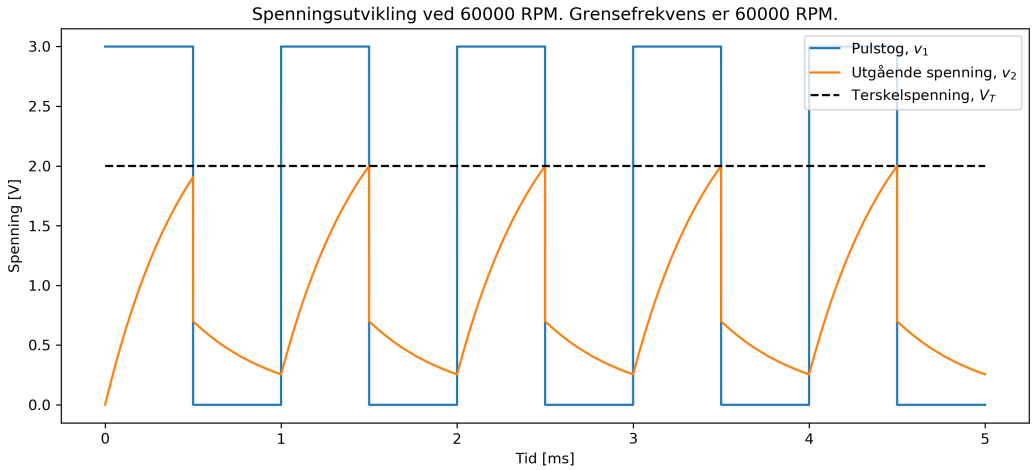
Vi kan bruke dette til å finne en tau, τ, gitt en frekvens, med periode T.

Vi kan anta at pulstoget har holdt på en stund, slik at systemet får balansert seg. Dersom vi da tar utgangspunkt i at vi har en diode med en diodespenning V_D. Startspenningen for stigningen av spenningen vil være sluttspenningen til utladningen over dioden, sett i Figur 4. Etter hver høye del av pulstoget vil spenningen v_2 raskt bevege seg mot V_D.

Spenningen v_2 når pulstoget er lavt, og endrer til høy, vil ha en likning som i (1).

$$v_D = V_D \cdot e^{-\frac{T}{2\tau}} \tag{1}$$

Siden vi vet at v_1 varierer fra v_D til en spenning V_0 i et pulstog med periode T, med driftssyklus



Figur 4: Eksempel på hvordan spenniger endrer seg i kretsen i Figur 3.

på 50%, er pulsen V_0 i $\frac{1}{2}T$. Vi vet da at spenningen v_2 må nå V_T etter $\frac{1}{2}T$ for at transistoren skal åpne seg.

Dersom vi setter dette sammen med hvordan spenningen utvikler seg gjennom perioden av pulstoget som er høyt vil vi få som i (2).

Vi bruker formelen for spenning over en kondensator og løser for tidskonstanten τ.

$$v_2 = V_T = V_0 \cdot (v_D - V_0)\, e^{-\frac{T}{2\tau}} \tag{2}$$

$$V_T = V_0 \cdot \left(V_D \cdot e^{-\frac{T}{2\tau}} - V_0\right) e^{-\frac{T}{2\tau}} \tag{3}$$

$$0 = V_D \cdot e^{-\frac{T}{\tau}} - V_0 \cdot e^{-\frac{T}{2\tau}} + (V_0 - V_T) \tag{4}$$

$$e^{-\frac{T}{2\tau}} = \frac{V_0 \pm \sqrt{(V_0)^2 - 4 \cdot V_D \cdot (V_0 - V_T)}}{2 \cdot V_D} \tag{5}$$

$$\tau = -\frac{T}{2 \cdot \ln\left(\frac{V_0 \pm \sqrt{(V_0)^2 - 4 \cdot V_D \cdot (V_0 - V_T)}}{2 \cdot V_D}\right)} \tag{6}$$

Merk: *Det må velges den verdien for τ som gir mening. Vi ser også at $V_0 > V_T$ for at likningen skal gi mening.*

Når vi har funnet en tidskonstant ved (6), velger vi bare en tilstrekkelig liten kondensator for C_1 og bruker likning (7) for tidskonstanten τ

$$\tau = R_1 \cdot C_1 \tag{7}$$

for å finne verdien for motstanden R_1.

3 Realisering og test

Grenseverdien for omdreiningshastigheten er gitt ved $f = 40000\text{rpm} \approx 666.67\text{Hz}$. Vi finner periodetiden ved (8).

$$T = \frac{1}{f} \Rightarrow T = 1.5 \text{ ms} \tag{8}$$

Pulstoget har en spenning $V_0 = 5\text{V}$. Det ble brukt en BS170 transistor. Den har en terskelspenning $V_T \approx 2\text{V}$, og dioden som ble brukt har en diodespenning på $V_D \approx 0.7\text{V}$.

Vi finner τ ved hjelp av (6).

$$\tau = -\frac{1.5 \text{ ms}}{2 \cdot \ln\left(\frac{5\text{V} \pm \sqrt{(5\text{V})^2 - 4 \cdot 0.7\text{V} \cdot (3\text{V})}}{2 \cdot 0.7\text{V}}\right)} \tag{9}$$

$$\tau = \begin{cases} -0.40128946 \text{ ms} \\ 1.81296116 \text{ ms} \end{cases} \tag{10}$$

Velger den verdien som gir mening

$$\tau = 1.81296116 \text{ ms} \tag{11}$$

For å koble opp kretsen i Figur 3 trenger vi kun å regne ut verdiene for R_1 og C_1.

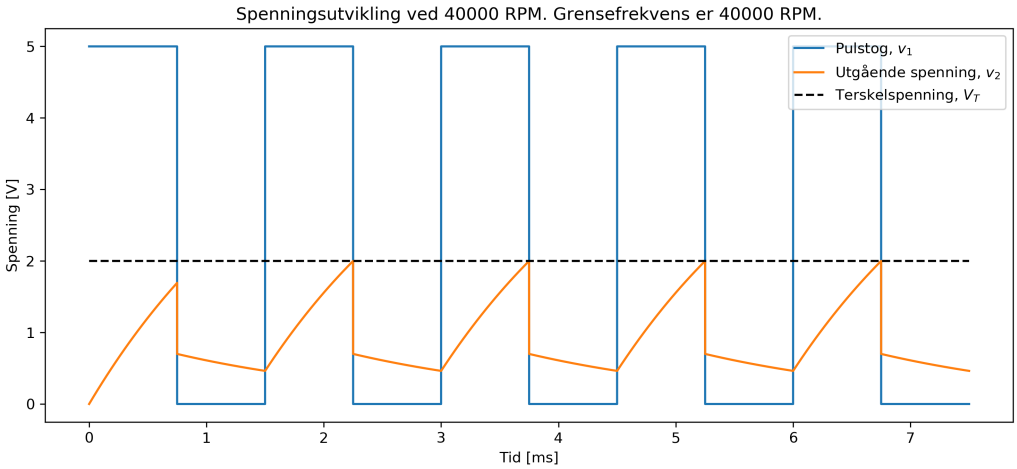
Velger motstand $R_1 = 1\text{M}\Omega$. Bruker (6) for å finne $C_1 \approx 1.8\text{nF}$.

Simulering av spenningene med disse forhåndsvalgte verdiene kan sees i Figur 5. Kode for simuleringen kan sees i Vedlegg B.

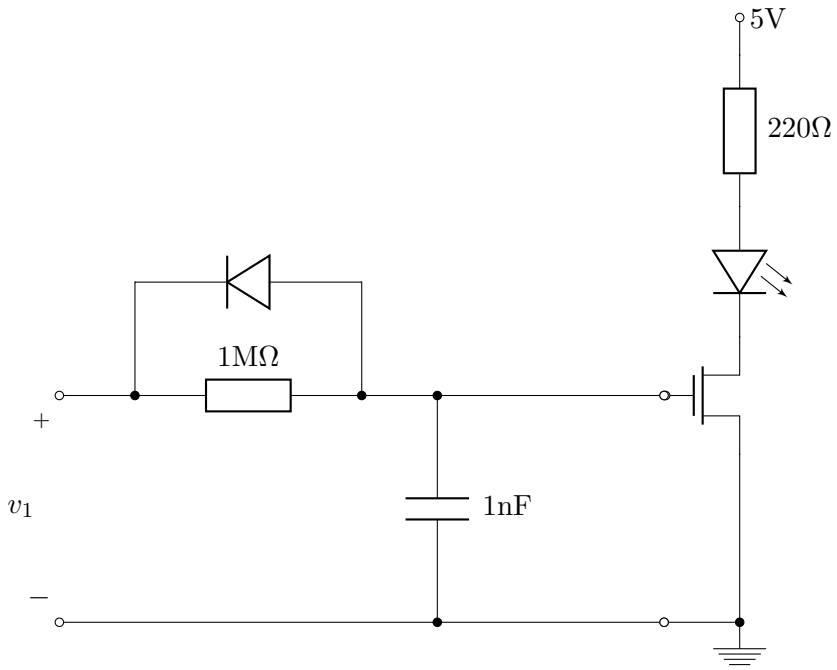
Etter testing og observasjoner ser vi at spenningen v_2 har en topp på rudnt 2V. Men det er ikke nok til å få dioden til å lyse. Dersom vi ønsker en mer lyssterk diode, kan vi bare endre terskelspenningen i (9) til noe høyere. Finner ut at for at lysdioden vi bruker skal "lyse", må vi ha en kondensator på 1nF.

Den fungerende kretsen har følgende skjema, se Figur 6. Ferdig oppkoblet krets kan sees i Figur 7

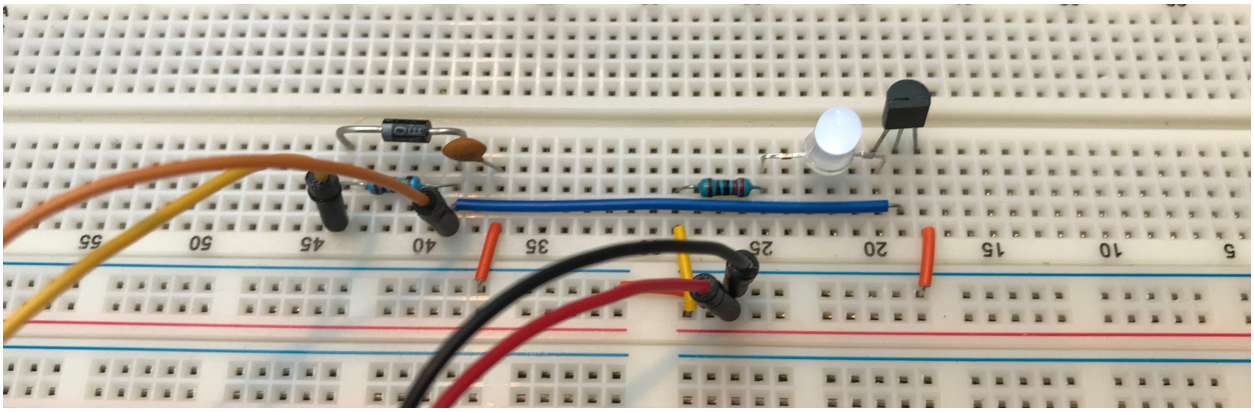
Dioden vil da begynne å lyse ved terskelfrekvensen og bli sterkere jo lavere frekvensen på pulstoget blir, og forsvinner ved frekvenser høyere enn terskelfrekvensen, se Figur 8 for se sammenhengen mellom frekvens og hvordan v_2 oppfører seg rundt terskelspenningen V_T.



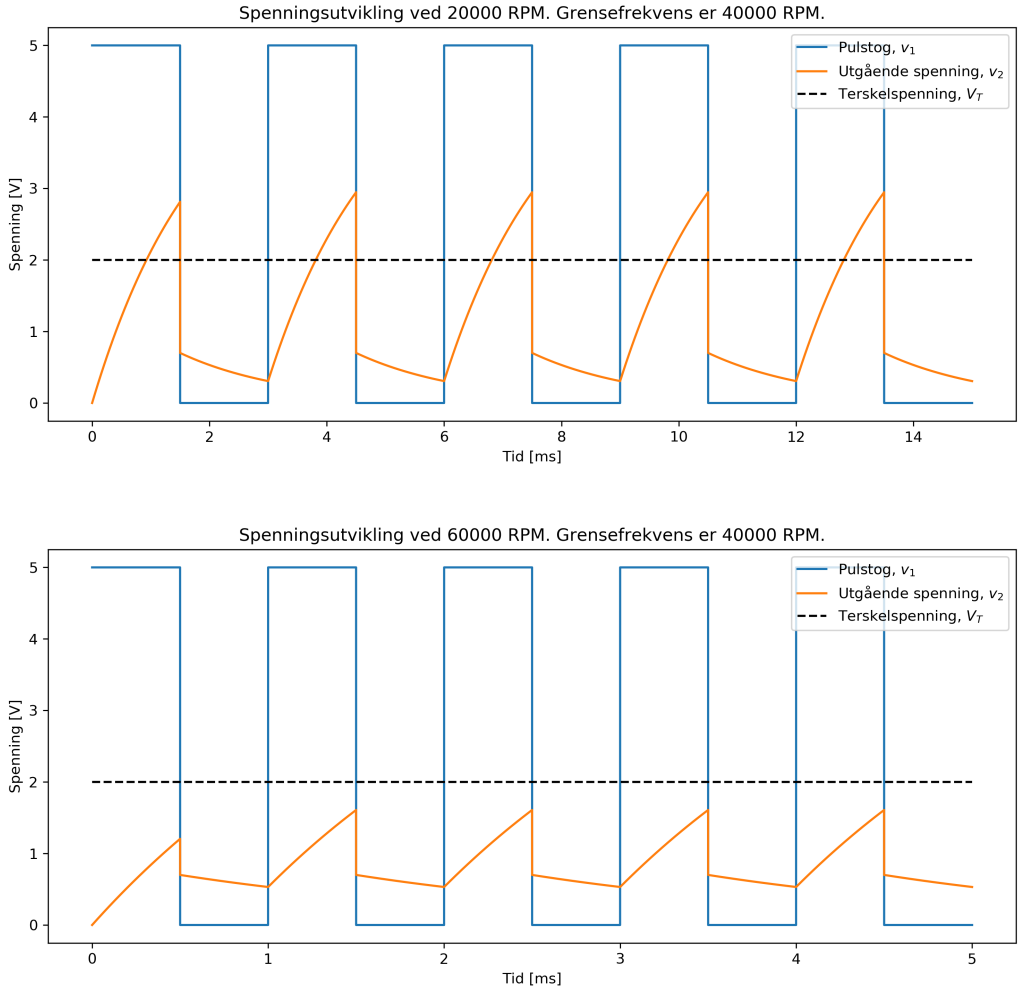
Figur 5: Simulering av kretsen.



Figur 6: Skjema for omdreinigsmåler med transistor, med verdier



Figur 7: Fysisk oppkobling av krets



Figur 8: Simulering av kretsen ved frekvens lavere (øverst) og høyere (nederst) enn terskelfrekvensen. Kode i vedlegg B.

4 Konklusjon

Kretsen fungerte innenfor kravene som ble gitt. Dioden begynner å lyse ved gitt omdreiningshastighet på 40000 rpm.

5 Takk

Takk til Ulrik Bredland og Magnus Oddstøl for bra samarbeid. Stor takk til lærere på elsys som har tatt seg tiden til å gi en tilbakemelding på dette designnotatet.

A Digital måte å måle omdreiningshastighet

Det er mulig å lage en krets med arduino for å måle frekvensen til et pulstog.

Ta for dere koden under. Pulstoget leses på pinne 8 på arduinoen, og en transistor kan styres på pinnen som heter "out" i koden. Frekvensen velges ved å endre variabelen "rpm".

```
/*  Program to measure the frequenncy of a input, on digital pin 8
 *  Made by Oyvind Skaaden
 */
#include <FreqMeasure.h>  // Library for measure
#define out 4             // The pin the transistor is controlled by

float rpm = 40000;        // The limit for when to open the transistor
bool ledUnderRPM = true;  // Change this to light the LED should light when the rpm
    are over rpm
float freq = rpm / 60;

void setup() {
  Serial.begin(57600);
  FreqMeasure.begin();
}

double sum=0;
int count=0;

void loop() {
  if (FreqMeasure.available()) {
    // average several reading together
    sum = sum + FreqMeasure.read();
    count = count + 1;
    if (count > 30) {
      float frequency = FreqMeasure.countToFrequency(sum / count);
      Serial.print(frequency);
      if (frequency < freq) { // If the measured frequency is below the RPM do this
        digitalWrite(out, ledUnderRPM);
        Serial.println(" ON");
      }
      else{
        digitalWrite(out, !ledUnderRPM);
        Serial.println(" OFF");
      }
      sum = 0;
      count = 0;
    }
  }
}
```

B Kode for simulering av didoe-kondensator-krets

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 26 00:50:35 2019

@author: oyvind
"""
import math
import numpy as np
import matplotlib.pyplot as plt

# Name for the saved graph
filename = "SimuleringPy"

#RPM to simulate
RPM = 40000
#RPM to calculate tau
tauRPM = 40000

# Forward voltage of the diode
forwardVoltage = 0.7

# Threshold voltage of the transistor
thresholdVoltage = 2

# Upper voltage of the pulsetrain
upperVoltage = 5

# How many cycles you want it to run
cycles = 5
# The resolution of each cycle
resolution = 2000

# Converting from RPM to Hz and period in seconds and milliseconds
def RPMtoPeriod(rpm):
    Hz = rpm / 60.0
    period = 1 / Hz
    periodmS = period * 10**3
    return periodmS

periodmS = RPMtoPeriod(RPM)

print("Period in ms: " + str(periodmS))



# Calculates the tau based on the period, and the different voltages
def CalculateTau():
    # Calculate the roots of the tau-equation
    roots = np.roots([forwardVoltage, -upperVoltage, upperVoltage - thresholdVoltage
    ])
    # Calculate the possible taus
    taus = -(RPMtoPeriod(tauRPM)) / (2 * np.log(roots))
```

```
    print(taus)
    # Discard the non-real tau
    if taus[0] > taus[1] and taus[0] > 0:
        return taus[0]
    return taus[1]

# Just creates a pulsetrain with some voltage and duty-cycle
def GeneratePulsetrain(voltage = 5, dutyCycle = 0.5):
    pulseTrain = []
    for i in range(cycles):
        for i in range(round(resolution * 2 * dutyCycle)):
            pulseTrain.append(voltage)
        for i in range(round(resolution * 2 * (1 - dutyCycle))):
            pulseTrain.append(0)

    return pulseTrain

# Generate the time-signatures from the number of cycles, the resolution and
# the periodtime. result in milliseconds
def GenerateTime():
    times = []
    for t in range(cycles * resolution * 2):
        times.append(periodmS * t / (resolution * 2))
    return times


def OutVoltage(wave, times):
    cP = wave[0] # Start voltage-supply
    v0 = 0 # Start voltage
    cT = 0 #Start time
    volt = []
    for p in range(len(wave)):
        # If the voltage-supply changes, recalculate startvalues
        if wave[p] != cP:
            v0 = volt[-1] # New start voltage, uses the last voltage calculated
            if wave[p] == 0:
                v0 = forwardVoltage
            cP = wave[p] # Variable so the array is not accessed.
            cT = times[p] # Offset time for each period
        # Calculate the outgoing voltage over the CAPACITOR with the start values
        volt.append(cP + (v0 - cP) * math.exp(-(times[p] - cT) / tau ))
    return volt

def GenerateThreshLine(times):
    thresh = []
    for i in times:
        thresh.append(thresholdVoltage)
    return thresh

# Calculate Tau
tau = CalculateTau()
print("Tau in ms: " + str(tau))

# Generate all the lists of pulsetrain and time-signatures
time = GenerateTime()
pulse = GeneratePulsetrain(upperVoltage)
```

```
threshLine = GenerateThreshLine(time)

# Simulate the outgoing voltage
voltage = OutVoltage(pulse, time)

plt.figure(figsize=(12,5))
plt.plot(time, pulse)
plt.plot(time, voltage)
plt.plot(time, threshLine, 'k--')
plt.title("Spenningsutvikling ved " + str(RPM) + " RPM. Grensefrekvens er " +
          str(tauRPM) + " RPM.")
plt.xlabel("Tid [ms]")
plt.ylabel("Spenning [V]")
plt.legend(["Pulstog, " + r'$v_1$', "Utgående spenning, " + r'$v_2$', "
    Terskelspenning, " + r'$V_T$'], loc="upper right")
plt.savefig(filename + ".png", dpi = 300)
plt.show()
```