OMNIS 2

Otwartość. Modernizacja. Nowoczesność. Integracja. Społeczność.

# Podstawy Elektroniki
# Ćwiczenia komputerowe

Generatory. Kryterium stabilności

Autor: Dariusz Tefelski

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
Politechnika Warszawska

Plac Politechniki 1
00-661 Warszawa

www.pw.edu.pl

# Spis treści



Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
Politechnika Warszawska

Plac Politechniki 1
00-661 Warszawa
www.pw.edu.pl

# 1 Wstęp i Cele Ćwiczenia

Celem niniejszego ćwiczenia laboratoryjnego jest zapoznanie z teoretycznymi i praktycznymi aspektami działania układów generacyjnych. Istotne jest zrozumienie warunków, przy których układ elektroniczny staje się generatorem, poznanie stabilności układów, a także zapoznanie się z architekturą podstawowych generatorów przebiegów sinusoidalnych oraz prostokątnych.

Ćwiczenie to realizowane jest z wykorzystaniem metod numerycznych i symulacyjnych:

- **Jupyter Lab (Python)** wraz z bibliotekami:
  - `numpy` i `scipy` - do symulacji sygnałowych i analizy częstotliwościowej (FFT).
  - `sympy` i `lcapy` - do analizy obwodów metodami symbolicznymi oraz analizy biegunów na płaszczyźnie zespolonej.
  - `matplotlib` - do wizualizacji przebiegów oraz widma sygnału.
- **NI Multisim** - dedykowane narzędzie wspomagające do walidacji schematycznej poszczególnych układów (Hartley, Colpitts itp.).

# 2 Podstawy Teoretyczne

## 2.1 Zagadnienie generacji sygnałów

Generator sygnałów to obwód elektroniczny prądu stałego z elementami nieliniowymi oraz pętlą dodatniego sprzężenia zwrotnego, służący do samoistnego wytwarzania powtarzających się sygnałów czasowych (np. sinusoidalnych, prostokątnych, trójkątnych) bez doprowadzania sygnału zmiennego z zewnątrz.

## 2.2 Warunki generacji (Kryteria Barkhausena)

Aby układ ze sprzężeniem zwrotnym mógł generować niegasnące drgania (o stałej amplitudzie), muszą być spełnione dwa podstawowe warunki zwane kryteriami Barkhausena:

Niech $\beta$ będzie transmitancją toru sprzężenia zwrotnego, a $K_u$ wzmocnieniem wzmacniacza napięciowego. Warunek startu układu zapisujemy jako pętlę $\beta \cdot K_u$.

1. **Warunek amplitudy:** Zjawisko generacji jest możliwe, gdy:

   $$|K_u \cdot \beta| \geqslant 1$$

   W praktyce podczas startu $|K_u \cdot \beta| > 1$, aby drgania narastały, a samoregulacja zmniejsza tę wartość do 1 w stanie ustalonym.

2. **Warunek fazy:** Całkowite przesunięcie w pętli sprzężenia zwrotnego musi wynosić wielokrotność $2\pi$ (lub zero):

   $$\arg(K_u \cdot \beta) = 2 \cdot k \cdot \pi, \quad \text{dla } k \in \mathbb{Z}$$

## 2.3 Kryterium stabilności układów i bieguny

W ujęciu automatyki i torów przesyłowych układ staje się generatorem oscylacji harmonicznych, gdy na płaszczyźnie zespolonej (w domenie Laplace'a) posiada dokładnie na osi urojonej $j\omega$ parę sprzężonych biegunów. Z fizycznego punktu widzenia generatory na początku oscylacji wymagają, aby para sprzężonych biegunów układu znajdowała się w **prawej półpłaszczyźnie zespolonej**. Zapewnia to narastanie amplitudy i wejście w obszar nieliniowości, który ogranicza amplitudę, spychając następnie bieguny z powrotem na oś urojoną $j\omega$.

## 2.4 Oscylatory budowane na elementach rezonansowych typu LC

W tego typu generatorach funkcję czwórnika determinującego odpowiedź częstotliwościową (spełnienie warunku fazy w jednej konkretnej częstotliwości) pełni obwód rezonansowy LC (cewka-kondensator). W zależności od tego, na jakich elementach oparte jest sprzężenie, wyróżniamy cztery klasy generatorów LC (np. bazujących na podziale napięcia na dzielniku pojemnościowym bądź indukcyjnym):

1. **Generator Meissnera (Armstronga):** Sprzężenie zwrotne jest zrealizowane na transformatorze (sprzężenie indukcyjne).
2. **Generator Hartleya:** Sprzężenie opiera się na tzw. dzielniku indukcyjnym. Cewka posiada odczep w środku nawinięcia, by sygnał trafiał z powrotem na wejście poprzez stosunek sprzężenia indukcyjnego.
3. **Generator Colpittsa:** Sprzężenie odbywa się przy pomocy pojemnościowego dzielnika napięcia na złączu tranzystora (bądź op-amp). Dwa uziemione szeregowe kondensatory tworzą ten dzielnik w rezonansie z równoległą cewką.
4. **Generator Clappa:** Modyfikacja układu Colpittsa - wprowadzony został dodatkowy, mały szeregowy kondensator do układu rezonansowego, co uniezależnia w dużej mierze częstotliwość rezonansową od małych pojemności sprzęgających czy błądzących. Zwiększa to stabilność układu.

## 2.5 Generator kwarcowy Pierce'a

Gdy wymagana jest nadzwyczajna stabilność częstotliwości oscylacji, konwencjonalne cewki zastępuje się rezonatorami kwarcowymi (działającymi w oparciu o zjawisko piezoelektryczne). Najpopularniejszym układem jest topologia układu Pierce'a. Rezonator kwarcowy w swoim obwodzie zastępczym składa się m.in. z gigantycznej pojemności zastępczej na gałęzi równoległej, zapewniając bardzo dużą dobroć układu $Q$.

## 2.6 Generatory przebiegów prostokątnych i relaksacyjne

W przeciwieństwie do generatorów fali sinusoidalnej generatory przebiegów prostokątnych opierają się typowo o ładowanie i rozładowywanie kondensatorów (układy całkujące) za-

mkniętych w pętli sprzężenia zawierającego histerezę (często za pomocą przerzutników Schmitta). Wymuszają one gwałtowne zmiany stanu wyjścia układu (oscylator relaksacyjny lub multiwibrator astabilny).

## 2.7 Ocena sygnałów: FFT, THD

Oceny generatora można dokonać za pomocą **Współczynnika Zawartości Harmonicznych** (*Total Harmonic Distortion - THD*). Mierzy on obecność obcych składowych harmonicznych w sygnale względem pierwszej częstotliwości podstawowej.

$$THD = \frac{\sqrt{V_2^2 + V_3^2 + \cdots + V_n^2}}{V_1} \cdot 100\%$$

Dla idealnego generatora wykrzykującego czystą funkcję sinus $THD$ wynosi blisko 0%.

Z kolei **Transformata Fouriera** służy rozłożeniu sygnału wejściowego (dziedziny czasu) na składowe harmoniczne, prezentowane w dziedzinie częstotliwości. Skrypt z wyliczeniem FFT został zaimplementowany w zadaniach w Pythonie z wykorzystaniem algorytmu Cooley'a-Tukey'a jako funkcja w `numpy/scipy`.

**Definicja**

**DFT** - Dyskretna transformacja Fouriera, pozwala przekształcić skończony ciąg próbek sygnału z dziedziny czasu do dziedziny częstotliwości.

**Wzór (Czas $\longrightarrow$ Częstotliwość):**

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

Gdzie:

- $N$ – całkowita liczba próbek.
- $x[n]$ – wartość sygnału w czasie dla próbki $n$ ($n = 0, 1, \ldots, N-1$).
- $X[k]$ – wynikowy prążek widma dla częstotliwości $k$ ($k = 0, 1, \ldots, N-1$).
- $e^{-j\frac{2\pi}{N}kn}$ – **jądro transformacji** (wykorzystuje wzór Eulera: $e^{-j\theta} = \cos\theta - j\sin\theta$).

**Co to oznacza w praktyce?**
DFT sprawdza „podobieństwo" sygnału $x[n]$ do wirujących wektorów zespolonych o różnych częstotliwościach. Wynik $X[k]$ jest liczbą zespoloną, która mówi nam o:

- `Amplitudzie` danej częstotliwości (moduł $|X[k]|$).
- `Fazie` danej częstotliwości (argument $\arg(X[k])$).

**Problem wydajnościowy:**
Obliczenie DFT wymaga wykonania $N$ mnożeń dla każdego z $N$ prążków. Oznacza to złożoność obliczeniową $O(N^2)$. Dla sygnału o długości 1 miliona próbek, komputer musiałby wykonać bilion operacji.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
Politechnika Warszawska

Plac Politechniki 1
00-661 Warszawa
www.pw.edu.pl

**Definicja**

**FFT** (Fast Fourier Transform) – Szybka Transformacja Fouriera
FFT nie jest nową transformatą – to rodzina algorytmów (najpopularniejszy to **Cooley-Tukey**), które obliczają dokładnie to samo co DFT, ale w ułamku sekundy. Jest algorytmem zoptymalizowanym wydajnościowo.

**Idea "Dziel i zwyciężaj":**
Algorytm FFT wykorzystuje symetrię i okresowość funkcji sinus i cosinus (zawartych w jądrze $e^{-j\theta}$). Rozbija on jedną dużą sumę DFT na dwie mniejsze: jedną dla próbek parzystych $(2n)$ i jedną dla nieparzystych $(2n+1)$.

$$X[k] = \sum_{n=0}^{N/2-1} x[2n] e^{-j\frac{2\pi}{N/2}kn} + e^{-j\frac{2\pi}{N}k} \sum_{n=0}^{N/2-1} x[2n+1] e^{-j\frac{2\pi}{N/2}kn}$$

W skrócie:

$$X[k] = E[k] + W_N^k \cdot O[k]$$

Gdzie $E[k]$ to wynik dla próbek parzystych (Even), a $O[k]$ dla nieparzystych (Odd), a $W_N^k$ to tzw. **czynnik obrotu** (twiddle factor).

**Złożoność obliczeniowa:**
Dzięki temu rekurencyjnemu podziałowi, złożoność spada do $O(N \log_2 N)$.

Porównanie prędkości (dla $N = 1024$):
- DFT $(N^2)$: $\approx 1\,048\,576$ operacji.
- FFT $(N \log_2 N)$:** $\approx 10\,240$ operacji.

**Wynik:** FFT jest w tym przypadku ponad **100 razy szybsza**.

**Definicja**

**IDFT** – Transformata Odwrotna (Częstotliwość $\longrightarrow$ Czas)
Aby wrócić z widma do sygnału czasowego, używamy wzoru syntezy:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{j\frac{2\pi}{N}kn}$$

Zwróć uwagę na dwie różnice względem DFT:
1. Brak minusa w wykładniku potęgi $e$.
2. Mnożnik $1/N$ przed sumą (normalizacja)

**Warto wiedzieć**

1. Dlaczego ograniczamy wynik do połowy (‘N // 2‘)?
Gdy poddajesz analizie sygnał **rzeczywisty** (czyli taki, który składa się z liczb rzeczywistych, a nie zespolonych – a takie są prawie wszystkie sygnały fizyczne), widmo po operacji FFT jest **symetryczne**.
   - **Symetria hermitowska:** Druga połowa tablicy zwróconej przez FFT (‘N//2‘ do ‘N‘) zawiera tzw. częstotliwości ujemne. Dla sygnałów rzeczywistych są one lustrzanym odbiciem (ściślej: sprzężeniem zespolonym) pierwszej połowy.
   - **Twierdzenie Nyquista:** Maksymalna częstotliwość, jaką możemy poprawnie odczytać, to połowa częstotliwości próbkowania ($f_s/2$). To właśnie ten punkt znajduje się w połowie tablicy FFT.
   - **Wniosek:** Druga połowa nie niesie nowej informacji o amplitudzie – jest redundantna. Dlatego bierzemy tylko ‘[:N // 2]‘.

2. Skąd bierze się normalizacja przez ‘1/N‘?
Definicja matematyczna DFT, której używa ‘scipy‘ i ‘numpy‘, wygląda tak:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{kn}{N}}$$

Zauważ, że jest to **suma**. Jeśli Twój sygnał ma $N$ próbek, to wynik FFT dla danej częstotliwości będzie proporcjonalny do liczby tych próbek.
   - Jeśli masz falę o amplitudzie 1.0 i długości 1000 próbek, FFT ”zsumuje”tę energię do wartości 1000 (w uproszczeniu).
   - Aby wrócić do jednostek fizycznych (oryginalnej amplitudy), musisz podzielić wynik przez całkowitą liczbę punktów $N$.

3. Skąd bierze się mnożnik ‘2.0‘?
Gdy dzielimy wynik przez $N$, otrzymujemy średnią energię na dany prążek. Jednak musimy pamiętać o tym, co zrobiliśmy w punkcie pierwszym (odrzucenie połowy widma).
   - **Rozszczepienie energii:** Transformata Fouriera rozdziela energię fali rzeczywistej (np. sinusa) na dwie części: częstotliwość dodatnią ($+f$) i ujemną ($-f$).
   - Połowa amplitudy trafia do prążka dodatniego, a połowa do ujemnego.
   - Skoro odrzucamy częstotliwości ujemne, to aby zachować całkowitą energię sygnału (i odczytać poprawną amplitudę fali), musimy **pomnożyć wynik przez 2**.
   - **Wyjątek:** Składowa stała (DC, częstotliwość 0) oraz częstotliwość Nyquista nie są ”rozbite”na dwie części, więc ich teoretycznie nie powinno się mnożyć przez 2, ale w praktycznych wizualizacjach często stosuje się uproszczone ‘2/N‘ dla całości.

Przykład w Pythonie:

**Listing 2.1: Uproszczony kod do fft w Python**

```
import numpy as np
from scipy.fft import fft

# N - liczba próbek
yf = fft(signal)

# 1. abs(yf)  bo wynik FFT jest zespolony (chcemy amplitudę)
# 2. [:N//2]  bo widmo jest symetryczne (wyrzucamy lustrzane odbicie)
# 3. / N      bo normalizacja sumy do średniej
# 4. * 2.0    bo energia była rozdzielona na +f i -f
amplituda = 2.0 / N * np.abs(yf[:N//2])
```

Dzięki temu, jeśli Twoim sygnałem był ‘5 * sin(2*pi*t)‘, po takiej operacji najwyższy słupek na wykresie będzie miał wartość dokładnie **5**.

# 3 Zadania Praktyczne uwzględniające język Python

W tej części należy uruchomić środowisko Jupyter Lab i przetestować wszystkie dołączone kody w kolejnych oknach.

## 3.1 Zadanie 1: Generowanie sygnału i transformata Fouriera

W zadaniu symulujemy sygnał idealny pochodzący z doskonałego generatora Colpittsa, następnie sztucznie ucinamy go za pomocą zniekształcenia wprowadzającego nieliniowość.

**Zadanie**

Przedstaw idealny i zniekształcony sygnał jako funkcję czasu na jednym wykresie.

## 3.2 Zadanie 2: Wyznaczanie THD (Total Harmonic Distortion)

**Zadanie**

Wyznacz parametr THD (Total Harmonic Distortion) dla nieidealnego - zniekształconego sygnału z naszego generatora z wykorzystaniem biblioteki `numpy`.

**Listing 3.1: Skrypt przedstawiający na wykresie sygnał idealny oraz zniekształcony oraz wyznaczający parametr THD dla zniekształconego sygnału**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq


def calculate_thd_from_spectrum(amplitudes, fundamental_index):
    # Wybieranie częstotliwości podstawowej
    fund_amp = amplitudes[fundamental_index]

    # Wyłuskujemy wyłącznie niezerowe i wyraźne piki wyższych
    ↪  harmonicznych (THD)
    # Dla uproszczenia bierzemy kolejne wielokrotnosci fundamental index:
    # 2*f0, 3*f0, 4*f0 itd.
    sum_harmonics_pow2 = 0

    for i in range(
        2, 20
    ):  # sumujemy np powiedzmy az mniemalne 20 harmonicznych
        idx = fundamental_index * i
        if idx < len(amplitudes):
            sum_harmonics_pow2 += amplitudes[idx] ** 2
```

**Listing 3.1: Skrypt przedstawiający na wykresie sygnał idealny oraz zniekształcony oraz wyznaczający parametr THD dla zniekształconego sygnału (c.d.)**

```
    thd = (np.sqrt(sum_harmonics_pow2) / fund_amp) * 100
    return thd


# Częstotliwość poboru próbek i ich całkowita liczba
fs = 10000
N = 4096
t = np.linspace(0, N / fs, N, endpoint=False)

f0 = 50.0  # czestotliwość podstawowa generatora w Hz
v1 = 2.0 * np.sin(2 * np.pi * f0 * t)  # idealny generetowany sygnal sin

# Przesterowanie generatora (efekt odcięcia ze wzgledu na zasilanie)
v1_clipped = np.clip(v1, -1.8, 1.8)

plt.figure(figsize=(10, 4))
plt.plot(t[:400], v1[:400], label="Idealny sygnal z generatora",
↪  color="blue")
plt.plot(
    t[:400],
    v1_clipped[:400],
    label="Sygnal dystorsyjny i nieliniowy",
    linestyle="--",
    color="red",
)
plt.title("Sygnały ze sztucznego wzmacniacza")
plt.xlabel("Czas [s]")
plt.ylabel("Amplituda [V]")
plt.legend()
plt.grid(True)
plt.show()

# ---- Wykonywanie Transformat Fouriera ----
yf = fft(v1_clipped)
xf = fftfreq(N, 1 / fs)[: N // 2]
ampl_spectrum = 2.0 / N * np.abs(yf[: N // 2])

plt.figure(figsize=(10, 4))
plt.plot(xf[:200], ampl_spectrum[:200], color="green")
plt.title("Widmo amplitudowe FFT przesterowanego sygnału")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Amplituda z harmonicznymi [V]")
plt.grid(True)
plt.show()
```

**Listing 3.1: Skrypt przedstawiający na wykresie sygnał idealny oraz zniekształcony oraz wyznaczający parametr THD dla zniekształconego sygnału (c.d.)**

```
# Znajdujemy indeks naszej pierwszej harmonicznej (ok. 50Hz) na spectrum:
f0_idx = np.searchsorted(xf, f0)

thd_value = calculate_thd_from_spectrum(ampl_spectrum, f0_idx)
print(
    f"Obliczone przybliżone THD zniekształconego sygnału sinus to:
    ↪ {thd_value:.2f}%"
)
```

## 3.3 Zadanie 3: Analiza biegunów generatora Collpits'a

**Listing 3.2: Analiza położenia biegunów na płaszczyźnie zespolonej dla generatora Collpits'a w zależności od transkonduktancji $g_m$ wzmacniacza**

```
from lcapy import s
import numpy as np
import matplotlib
import matplotlib.pyplot as plt


def analyze_colpitts_transfer():
    # Definiujemy stałe
    C1 = 100e-12
    C2 = 100e-12
    L = 10e-6
    R = 500  # Straty

    gm_values = [0, 0.002, 0.005]
    fig, axes = plt.subplots(1, 3, figsize=(15, 5))

    for i, gm in enumerate(gm_values):
        # Wielomian charakterystyczny dla Colpittsa (mianownik)
        # s^3 * (L*C1*C2) + s^2 * (R*C1*C2) + s * (C1 + C2) + gm
        denom = (
            s**3 * (L * C1 * C2) + s**2 * (R * C1 * C2) + s * (C1 + C2) +
            ↪ gm
        )

        # Tworzymy funkcję transmitancji (bieguny to pierwiastki denom)
        H = (1 / denom).simplify()

        print(f"\n--- gm = {gm} S ---")
        poles = H.poles()
```

**Listing 3.2: Analiza położenia biegunów na płaszczyźnie zespolonej dla generatora Collpits'a w zależności od transkonduktancji $g_m$ wzmacniacza (c.d.)**

```
        re_vals = []
        im_vals = []
        for p in poles:
            val = complex(p.evaluate())
            print(f"Biegun: {val}")
            re_vals.append(val.real)
            im_vals.append(val.imag)

        ax = axes[i]
        ax.axvline(0, color="black", alpha=0.3)
        ax.axhline(0, color="black", alpha=0.3)
        ax.scatter(re_vals, im_vals, marker="x", color="red", s=100)
        ax.set_title(f"gm = {gm} S")
        ax.set_xlim(-1e8, 1e8)
        ax.set_ylim(-1e8, 1e8)
        ax.grid(True)

    plt.tight_layout()
    plt.show()


if __name__ == "__main__":
    analyze_colpitts_transfer()
```

### 3.3.1 Analiza wyników

Dla parametrów $C_1 = C_2 = 100$ pF, $L = 10$ $\mu$H, $R = 500$ $\Omega$:

1. Przy $g_m = 0$, mamy parę biegunów sprzężonych z częścią rzeczywistą ujemną (tłumienie przez $R$) oraz jeden biegun w zerze.
2. Przy $g_m = 0.005$, bieguny zespolone przesuwają się (część rzeczywista rośnie w stronę dodatnią). Gdyby wzmocnienie było jeszcze większe lub straty mniejsze, przeszłyby na prawą stronę (RHP), co oznaczałoby wzbudzenie drgań.

Wartości części urojonej ($\sim 3.7 \cdot 10^7$ rad/s) odpowiadają częstotliwości około **5.9 MHz**, co zgadza się z teoretycznym wzorem na rezonans Colpittsa dla tych wartości:

$$f = \frac{1}{2\pi\sqrt{L\frac{C_1 C_2}{C_1+C_2}}} = \frac{1}{2\pi\sqrt{10^{-5} \cdot 50 \cdot 10^{-12}}} \approx 7.1 \text{ MHz}$$

(Różnica wynika z obecności rezystancji strat w modelu).

**Zadanie**

Zmień wartość **R** z 500 na 50. Zauważ, że bieguny przy transkonduktancji $gm > 0$ wchodzą na prawą część płaszczyzny zespolonej, co oznacza, że amplituda sygnału będzie narastała. Sytuacja ta odpowiada stanowi nieustalonemu na początku symulacji, gdy sygnał narasta. Z powodu nieliniowości, bieguny następnie przechodzą do położenia na osi urojonej, co zapewnia stabilną generację sygnału.

## 3.4 Zadanie 4: Multiwibrator - przebieg prostokątny

Generujemy czysty sygnał prostokątny służący cyfrowym magistralom na podstawie klasycznego generatora (multiwibratora astabilnego 555) przy użyciu `scipy.signal.square`. Następnie ocenimy obecność harmonicznych parzystych w FFT.

**Listing 3.3: Skrypt generujący przebieg prostokątny i wykonujący szybką transformację Fouriera.**

```
import numpy as np
from scipy import signal
from scipy.fft import fft, fftfreq
import matplotlib.pyplot as plt

f0 = 100

# Częstotliwość poboru próbek i ich całkowita liczba
fs = 10000
N = 4096
t = np.linspace(0, N / fs, N, endpoint=False)

# Generowanie fali prostokątnej
# Przestaw opcję `duty=0.5` na inne jeśli chcesz testowas PWM
square_wv = signal.square(2 * np.pi * f0 * t, duty=0.5)

plt.figure(figsize=(8, 3))
plt.plot(t[:400], square_wv[:400], color="purple")
plt.title("Sygnał Prostokątny - Z Multiwibratora")
plt.grid(True)
plt.show()

# Wyznaczamy specyfikacje dla fali
sq_yf = fft(square_wv)
xf = fftfreq(N, 1 / fs)[: N // 2]
sq_ampl_spectrum = 2.0 / N * np.abs(sq_yf[: N // 2])

plt.figure(figsize=(10, 4))
plt.plot(xf[:300], sq_ampl_spectrum[:300], color="black")
```

**Listing 3.3: Skrypt generujący przebieg prostokątny i wykonujący szybką transformację Fouriera. (c.d.)**

```
plt.title("Widmo sygnału prostokątnego (zauważ brak harmonicznych
↪  parzystych)")
plt.grid(True)
plt.show()
```

Skrypt ten ilustruje jak idealny przebieg prostokątny składa się tylko na szereg sum z nieparzystych potęg sinusoid (brak słupków na FFT gdzie $\omega_n$ ma wskaźnik parzysty 200, 400 Hz).

## 3.5 Zadanie 5: Charakterystyka rezonatora kwarcowego

Często tam, gdzie potrzebne są stabilne w czasie drgania wykorzystuje się rezonatory kwarcowe.

**Listing 3.4: Charakterystyka rezonatora kwarcowego**

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from lcapy import Circuit, s  # Importujemy s bezposrednio z lcapy
import sympy as sp


def run_quartz_analysis():
    # 1. PARAMETRY (Rezonator 10 MHz)
    Lm, Cm, Rm, C0 = 25e-3, 10e-15, 50, 5e-12

    # 2. DEFINICJA OBWODU
    c = Circuit()
    c.add(f"Lm 1 2 {Lm}")
    c.add(f"Cm 2 3 {Cm}")
    c.add(f"Rm 3 0 {Rm}")
    c.add(f"C0 1 0 {C0}")

    # 3. OBLICZENIA ANALITYCZNE
    fs = 1 / (2 * np.pi * np.sqrt(Lm * Cm))
    fp = fs * np.sqrt(1 + Cm / C0)
    Q = (2 * np.pi * fs * Lm) / Rm

    print(f"--- Parametry Rezonatora ---")
    print(f"fs: {fs / 1e6:.6f} MHz")
    print(f"fp: {fp / 1e6:.6f} MHz")
    print(f"Dobroć Q: {Q:.0f}")

    # 4. KONWERSJA SYMBOLICZNA DO NUMERYCZNEJ
    Z_s = c.impedance(1, 0)
```

**Listing 3.4: Charakterystyka rezonatora kwarcowego (c.d.)**

```
# Wyciagamy czyste wyrazenie SymPy. Symbol 's' musi pochodzic z
↪  lcapy.
# Uzywamy sp.lambdify do stworzenia funkcji wektorowej
Z_func = sp.lambdify(s.expr, Z_s.expr, modules="numpy")

# 5. GENEROWANIE DANYCH
# Bardzo gesty zakres czestotliwosci (5000 punktow)
f_range = np.linspace(fs - 2000, fp + 2000, 5000)
s_vals = 1j * 2 * np.pi * f_range

# Obliczamy impedancje
Z_vals = Z_func(s_vals)

# 6. WYKRESY
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(11, 9))

# Amplituda
ax1.semilogy(f_range / 1e6, np.abs(Z_vals), lw=2, color="tab:blue")
ax1.axvline(
    fs / 1e6,
    color="green",
    ls="--",
    alpha=0.7,
    label=f"fs = {fs / 1e6:.4f} MHz",
)
ax1.axvline(
    fp / 1e6,
    color="red",
    ls="--",
    alpha=0.7,
    label=f"fp = {fp / 1e6:.4f} MHz",
)
ax1.set_title(
    f"Charakterystyka kwarcu BVD (Q = {Q / 1000:.0f}k)", fontsize=14
)
ax1.set_ylabel(r"|Z| [$\Omega$]", fontsize=12)
ax1.grid(True, which="both", alpha=0.4)
ax1.legend()

# Faza
phase = np.angle(Z_vals, deg=True)
ax2.plot(f_range / 1e6, phase, lw=2, color="tab:purple")
ax2.fill_between(
    f_range / 1e6,
    0,
    90,
    where=(phase > 0),
```

**Listing 3.4: Charakterystyka rezonatora kwarcowego (c.d.)**

```
            color="orange",
            alpha=0.15,
            label="Obszar indukcyjny",
        )
        ax2.set_xlabel("Czestotliwosc [MHz]", fontsize=12)
        ax2.set_ylabel(r"Faza [$^\circ$]", fontsize=12)
        ax2.set_yticks([-90, -45, 0, 45, 90])
        ax2.grid(True, alpha=0.4)
        ax2.legend()

        plt.tight_layout()
        plt.show()


if __name__ == "__main__":
    run_quartz_analysis()
```

**Warto wiedzieć**

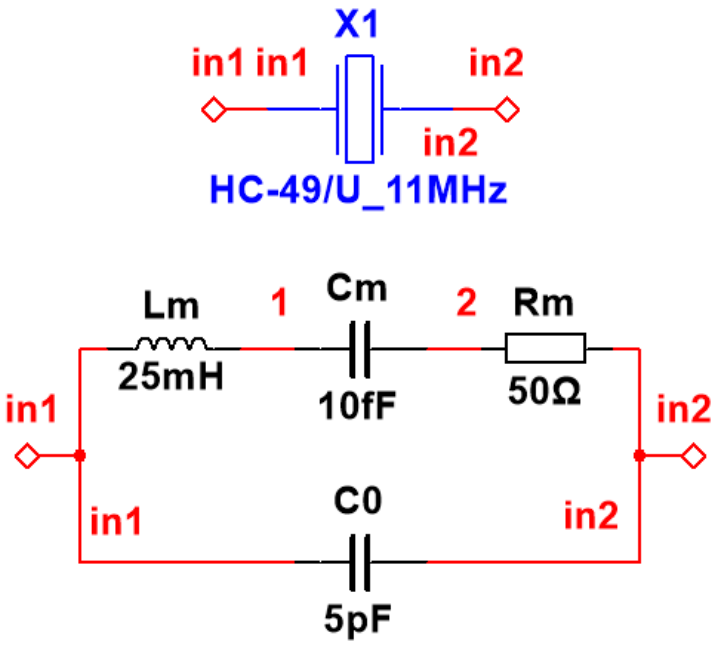
**Model BVD** (skrót od `Butterworth-Van Dyke`) to elektryczny schemat zastępczy rezonatora kwarcowego. Pozwala inżynierom analizować mechaniczne drgania kryształu kwarcu tak, jakby były zwykłym obwodem elektrycznym. Ponieważ kwarc wykorzystuje `zjawisko piezoelektryczne`, jego mechaniczne właściwości (masa, sprężystość, tarcie) mają swoje bezpośrednie odpowiedniki w świecie elektroniki.

`Składniki modelu BVD:`

**Model składa się z dwóch gałęzi połączonych równolegle:**

1. Gałąź dynamiczna (Motional Arm) – szeregowa. Reprezentuje ona właściwości fizyczne drgającego kryształu:
   - $L_m$ (Indukcyjność dynamiczna): Odpowiada `masie` (bezwładności) kryształu. W kwarcu przybiera ona ogromne wartości (milihenry), których nie dałoby się uzyskać za pomocą zwykłej cewki w tej skali.
   - $C_m$ (Pojemność dynamiczna): Odpowiada `sprężystości` kryształu. Jest ekstremalnie mała (femtofarady), co sprawia, że obwód jest bardzo "sztywny".
   - $R_m$ (Rezystancja dynamiczna): Odpowiada za `tarcie wewnętrzne` i straty energii. Im mniejsza, tym lepszej jakości jest kwarc.
2. Gałąź statyczna – równoległa.sftp
   - $C_0$ (Pojemność statyczna): To po prostu fizyczna pojemność mierzona między elektrodami kwarcu, gdy ten nie drga. Wynika z budowy uchwytu i obudowy.

**Model BVD wyjaśnia dwa rodzaje rezonansu, które widzieliśmy na wykresach:**




Rysunek 1: Multisim: Model BVD rezonatora kwarcowego - schemat zastępczy.

1. Rezonans szeregowy ($f_s$): Następuje, gdy gałąź dynamiczna ($L_m$, $C_m$) wpada w rezonans. Impedancja kwarcu jest wtedy minimalna (równa $R_m$). W tym punkcie kwarc zachowuje się jak czysty rezystor.
2. Rezonans równoległy ($f_p$): Następuje, gdy cała gałąź dynamiczna (która powyżej $f_s$ zachowuje się jak cewka) wpada w rezonans z pojemnością obudowy $C_0$. Impedancja kwarcu staje się wtedy ogromna (obwód zaporowy).

**Zastosowanie w praktyce:**

Bez modelu BVD nie dałoby się zaprojektować stabilnego zegara w komputerze czy telefonie. Pozwala on obliczyć:

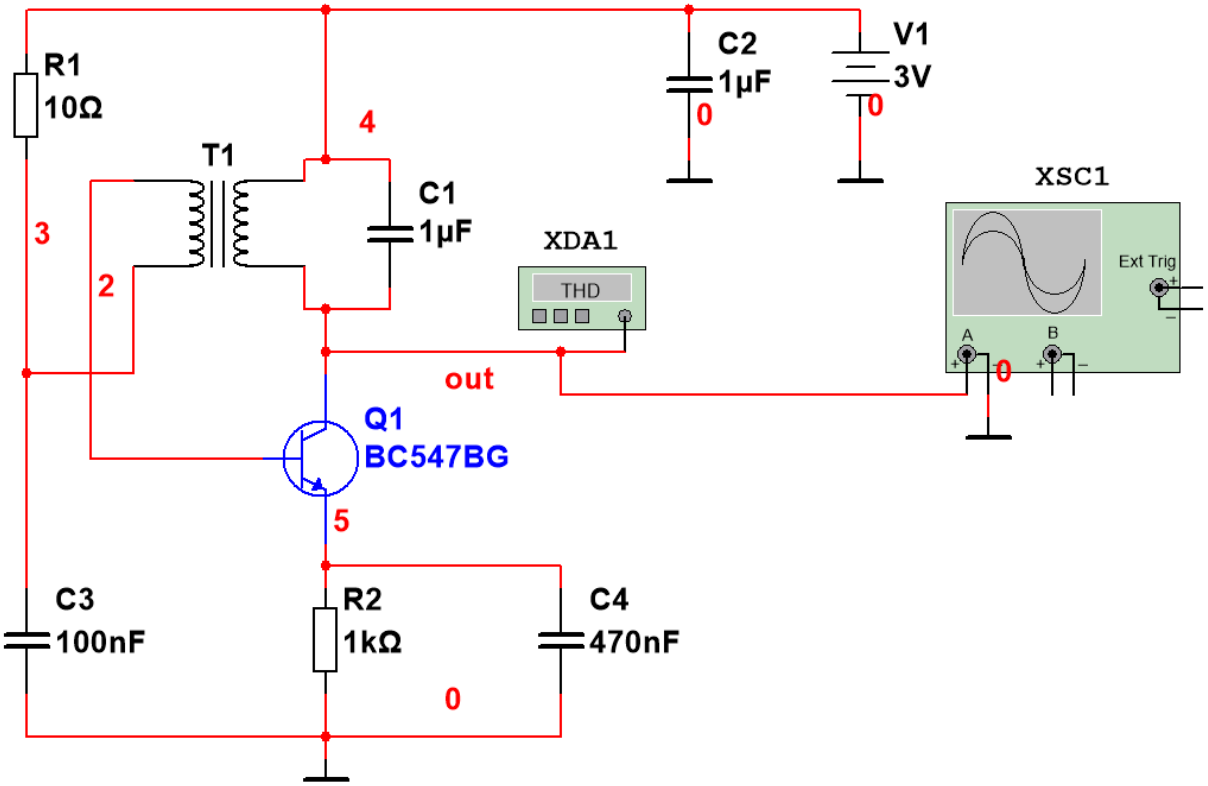
- Dobroć ($Q$): Stosunek energii zgromadzonej do rozproszonej. Dzięki ogromnemu $L_m$ i znikomemu $C_m$, kwarc ma $Q$ na poziomie 10 000 – 100 000, podczas gdy zwykłe obwody LC osiągają zaledwie 100.
- Stabilność: Dzięki modelowi wiemy, jak temperatura (zmieniająca parametry mechaniczne) wpłynie na częstotliwość elektryczną.

  W `generatorze Pierce'a`, układ pracuje w wąskim "oknie" między $f_s$ a $f_p$, gdzie kwarc udaje bardzo precyzyjną i stabilną cewkę.

# 4 Zadania do wykonania w NI Multisim

**Zadanie**

Przygotuj i przetestuj działanie generatorów Meissnera, Hartleya, Colpittsa i Clappa wykorzystując oprogramowanie NI Multisim. Do symulacji w programie użyj dowolnego tranzystora np. element 2N3904 (tranzystory bipolarne powszechnego użytku: BJT - Bipolar Junction Transistor). Do weryfikacji wytworzonych drgań w symulacji interaktywnej użyj wirtualnego oscyloskopu (`Oscilloscope`). Uruchom analizę symulacji Transient i zwróć uwagę, jak po czasie około $t = 1ms$ układ nieliniowo narasta aż do stabilnych w czasie drgań.

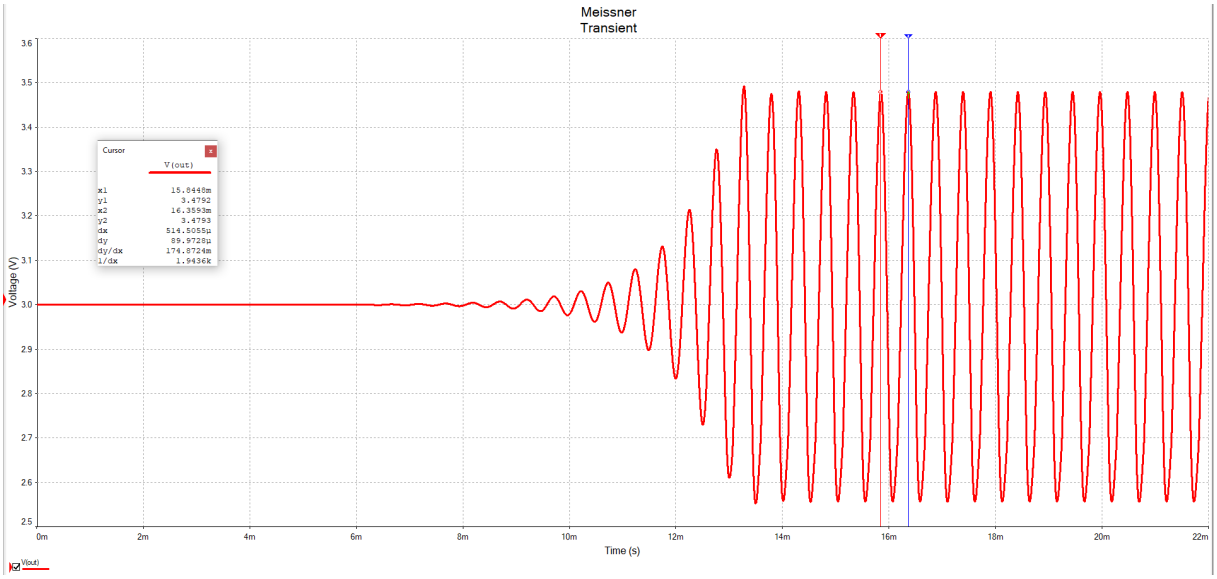



Rysunek 2: Multisim: Schemat układu generatora Meissner'a przygotowany do symulacji.

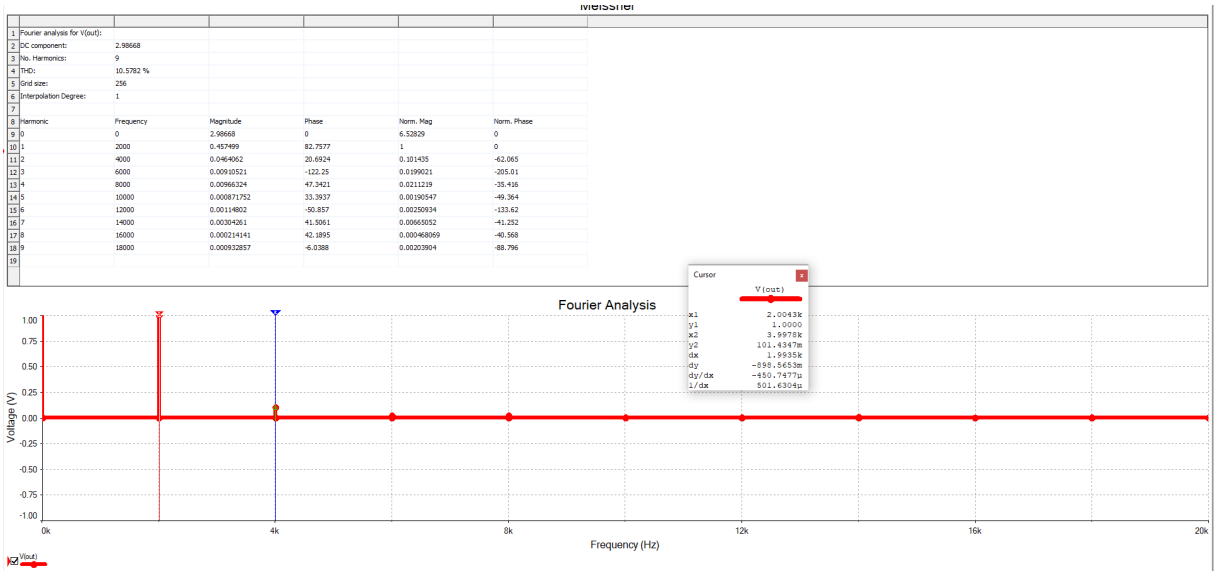
# 5 Pytania kontrolne i Odpowiedzi

1. **Jakie są warunki Barkhausena i jakie jest ich znaczenie przy pracy generatora?**
   *Odpowiedź:* Dwa warunki: amplitudy ($\beta \cdot K = 1$) oraz fazy ($\Delta\phi = 2 \cdot k \cdot \pi$). Ich spełnienie decyduje w pierwszym rzędzie o rozpoczęciu oraz o utrzymaniu stabilnych drgań wyjściowych jako układ samowzbudny.

2. **Czym się różni sprzężenie zwrotne w generatorze Colpittsa od generatora Hartleya?**



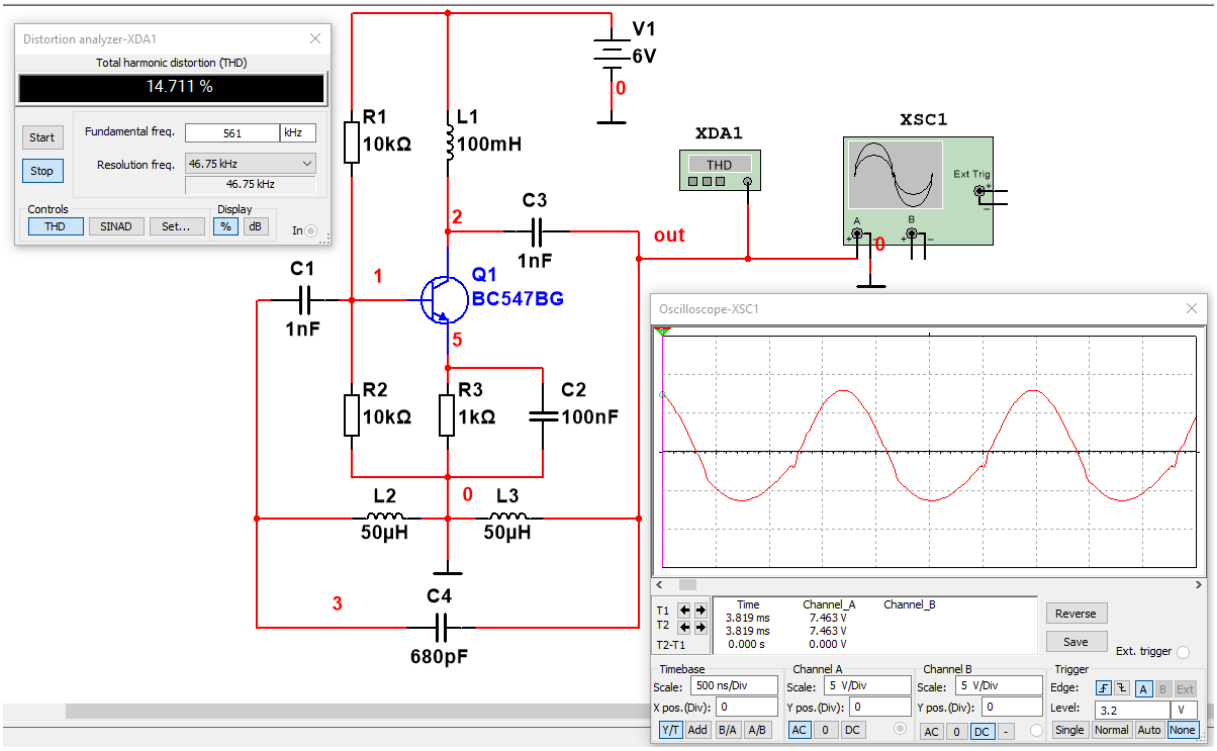

Rysunek 3: Multisim: Wynik symulacji czasowej (`Transient`) generatora Meissner'a.



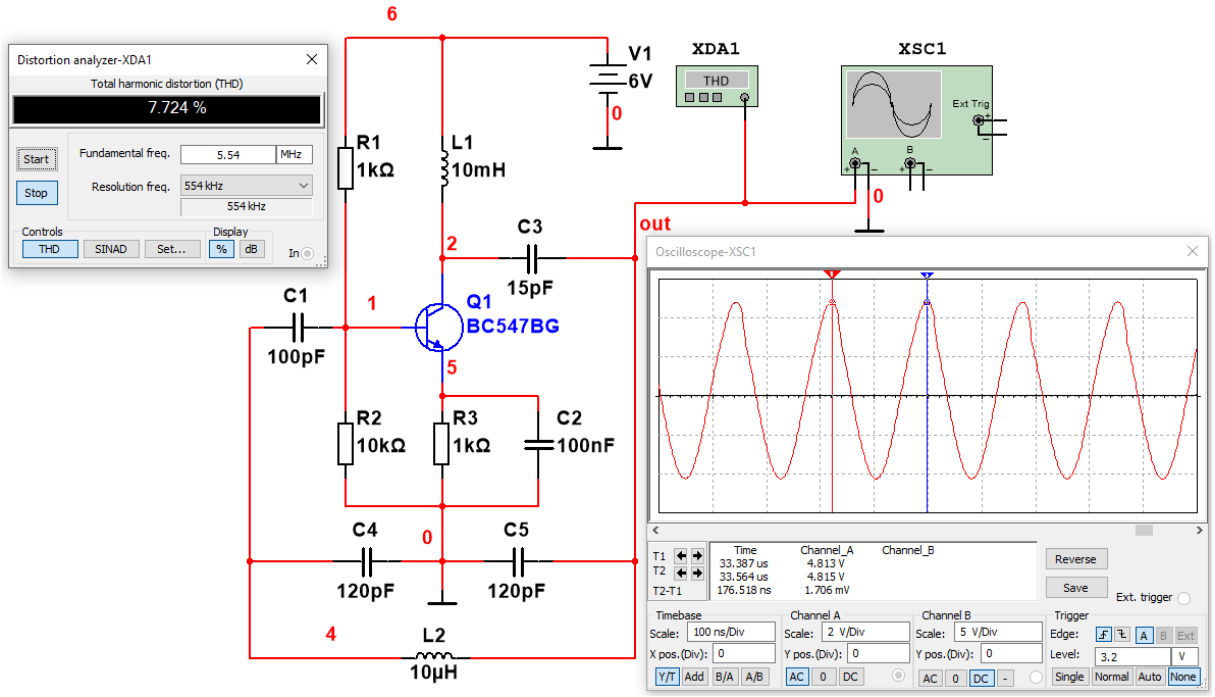

Rysunek 4: Multisim: Symulacja typu `Fourier Analysis` generatora Meissnera.

*Odpowiedź:* Generator Colpittsa posiada w sprzężeniu zwrotnym układ typu Π, w którym, w ramionach są kondensatory, a w „poprzeczce" jest cewka. Natomiast układ Hartleya w ramionach ma cewki, które zazwyczaj są technicznie jedną cewką z odczepem po środku.

3. **Dlaczego w przypadku konieczności stabilnego generowania sygnału stosuje się układ oscylacyjny Pierce'a z rezonatorem kwarcowym, a nie popularny i wszechstronny układ RLC zbudowany z elementów dyskretnych?**
*Odpowiedź:* Rezonator kwarcowy odznacza się kilkadziesiąt do kilkuset razy więk-

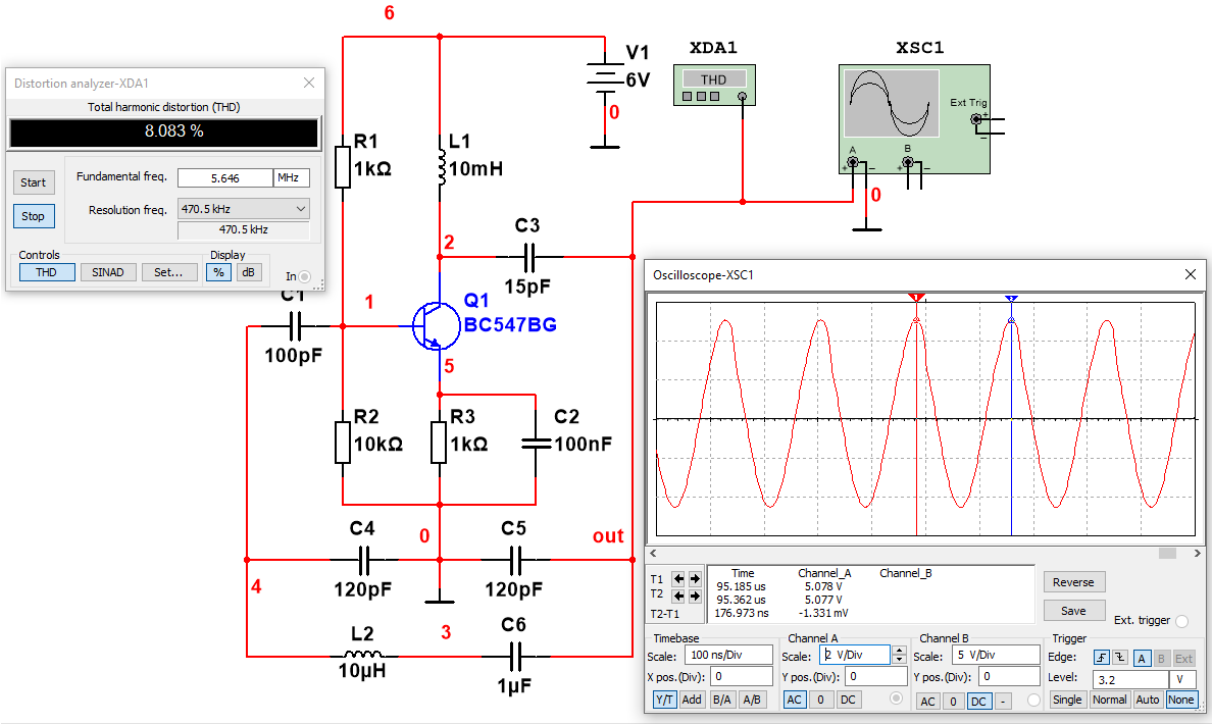



Rysunek 5: Multisim: Schemat układu generatora Hartley'a przygotowany do symulacji.




Rysunek 6: Multisim: Schemat układu generatora Collpits'a przygotowany do symulacji.

szym współczynnikiem dobroci $Q$, ze względu m.in. na fizykę zjawisk piezoelektrycznych w porównaniu z klasycznymi obwodami LC zawierającymi cewki nawi-




Rysunek 7: Multisim: Schemat układu generatora Clapp'a przygotowany do symulacji.

nięte z miedzianego drutu. Posiadając niewielki odchył parametrów rzędu ppm/°C gwarantuje bardzo dokładną pracę obwodów czasowych (np. zegarów taktujących mikrokontrolery).

4. **Co to jest współczynnik THD i co jego ogromne wartości znaczą o wygenerowanym sygnale?**
   *Odpowiedź:* THD (Total Harmonic Distortion) to wartość opisująca całkowity ułamek obcych harmonicznych prądu lub napięcia niebędących oryginalną sinusoidą bazową. Jeśli procent tego wskaźnika jest znaczny dla testowanego generatora, to obwód działa poza obszarem liniowym - np. obcina amplitudę lub nie tłumi skutecznie echa.

**Informacje dodatkowe**

Więcej informacji dostępne jest na stronie przedmiotu EwEF, w zakładce dotyczącej ćwiczeń: https://labe.engined.eu/index.php/%C4%86wiczenia_EwEF. W szczególności ta tematyka zawarta jest w materiałach z wykładu 2: https://labe.engined.eu/data/_uploaded/media/EwEF/EwEF_w2.pdf. Wykład 2 w postaci video zaczyna się w połowie, jednakże zawarte tam informacje uzupełniają skróconą zawartość wykładu w postaci pdf. https://labe.engined.eu/data/_uploaded/media/EwEF/EwEF_W2_cz2.mp4.

## 5.1 Wymagane oprogramowanie

Do wykonania ćwiczenia wymagany jest NI Multisim, Jupyter-lab i Python 3 z zainstalowanymi modułami:

- `Sympy` — do obliczeń symbolicznych i transformat,
- `Matplotlib` — do wizualizacji wyników,
- `Numpy` — do podstawowych struktur danych,
- `Scipy` — do obliczeń na sygnałach

# 6 Autorzy i historia opracowania

- dr inż. Dariusz Tefelski - wersja z 2026r.