4_hands-on2

April 23, 2025

1 Hands-on 2 : Working with Signals

Hands-on 2 IF3024_2024202502 Created by Toby
Reviewed and supervised by Martin Manullang

1.1 Revisit: Apa itu Sinyal

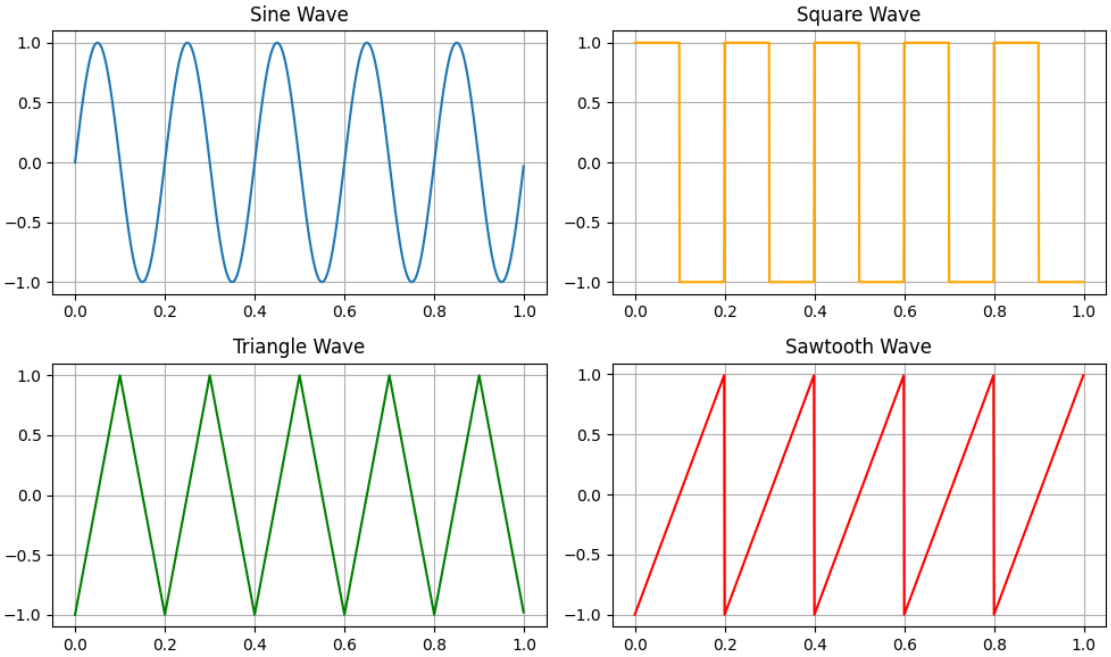
Sinyal adalah fungsi yang membawa informasi (atau bisa disebut $\frac{value}{time}$).

```
<img src="media/signals-101.png" alt="Signals 101" style="max-width: 100%; height: auto;">
<p><em>Figure 1: Tebak sinyal apa?</em></p>
```

Banyak representasi sinyal yang bisa kita temui, namun berikut merupakan beberapa sinyal yang bisa disebut sebagai sinyal dasar / fundamental dalam **Pengolahan Sinyal**

[22]:
```
from scipy.signal import square, sawtooth
import numpy as np
import matplotlib.pyplot as plt

# Time vector
t = np.linspace(0, 1, 1000, endpoint=False)

# Generate signals
sine_wave = np.sin(2 * np.pi * 5 * t)  # Sine wave with frequency 5 Hz
square_wave = square(2 * np.pi * 5 * t)  # Square wave with frequency 5 Hz
triangle_wave = sawtooth(2 * np.pi * 5 * t, 0.5)  # Triangle wave with␣
 ↪frequency 5 Hz
sawtooth_wave = sawtooth(2 * np.pi * 5 * t)  # Sawtooth wave with frequency 5 Hz

# Plot signals
fig, axs = plt.subplots(2, 2, figsize=(10, 6))

# Sine wave
axs[0, 0].plot(t, sine_wave, label='Sine Wave')
axs[0, 0].set_title("Sine Wave")
axs[0, 0].grid(True)

# Square wave
```

```
axs[0, 1].plot(t, square_wave, label='Square Wave', color='orange')
axs[0, 1].set_title("Square Wave")
axs[0, 1].grid(True)

# Triangle wave
axs[1, 0].plot(t, triangle_wave, label='Triangle Wave', color='green')
axs[1, 0].set_title("Triangle Wave")
axs[1, 0].grid(True)

# Sawtooth wave
axs[1, 1].plot(t, sawtooth_wave, label='Sawtooth Wave', color='red')
axs[1, 1].set_title("Sawtooth Wave")
axs[1, 1].grid(True)

plt.tight_layout()
plt.show()
```



1.2 Simulasikan Sinyal Lain

Kamu bisa mensimulasikan sinyal lain menggunakan library `neuroki2`, `pywt` atau berikut; - `neurokit2` adalah library yang berisi fungsi-fungsi untuk analisis sinyal EEG (electroencephalography) - `mne` adalah library yang berisi fungsi-fungsi untuk analisis sinyal EEG (electroencephalography) - `obspy` adalah library yang berisi fungsi-fungsi untuk analisis sinyal seismik - `scipy` adalah library yang berisi fungsi-fungsi untuk analisis sinyal - `PyWavelets` adalah library yang berisi fungsi-fungsi untuk analisis sinyal

Batasannya adalah imajinasi kamu untuk mensimulasikan sinyal yang sesuai kebutuhan kamu.

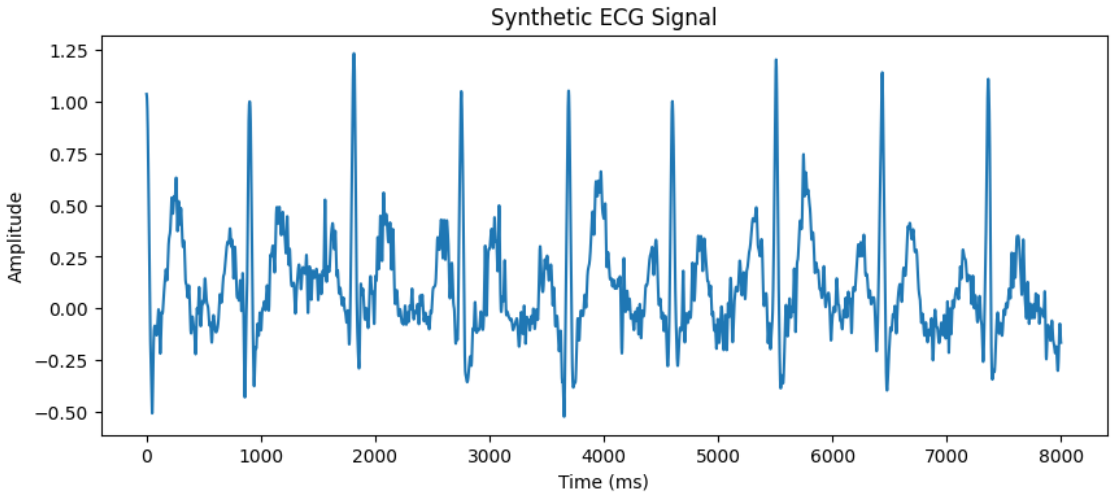
1.2.1 Sinyal-sinyal yang bisa kamu coba

1. Sinyal Fisiologis
   - Sinyal ECG (Electrocardiography)
   - Sinyal EEG (Electroencephalogram)
   - Sinyal Respirasi (Respiration)
   - Sinyal EMG (Electromyogram)
2. Sinyal Lingkungan (Seismic / Gempa Bumi)
   - Sinyal Seismik (Gempa Bumi)
   - Sinyal Perubahan Suhu
3. Sinyal Komunikasi
   - Sinyal Radio AM
   - Sinyal Radio FM
4. Sinyal Mekanik
   - Sinyal Getaran

1.2.2 Sinyal Fisiologis

Sinyal ECG (*Electrocardiography*). Sinyal kontraksi Jantung (Kardio) Sinyal EKG adalah sinyal listrik yang dihasilkan oleh aktivitas jantung. Sinyal ini dapat digunakan untuk mendiagnosis berbagai kondisi jantung, seperti aritmia, infark miokard, dan penyakit jantung koroner. Sinyal EKG biasanya direkam menggunakan elektroda yang ditempatkan di dada.

[23]:
```
## Simulate ECG signal using Neurokit2
import neurokit2 as nk

# Generate a synthetic ECG signal
ecg_signal = nk.ecg_simulate(duration=8, sampling_rate=1000, heart_rate=65,␣
 ↪noise=0.2, random_state=240925)

# Plot the ECG signal
plt.figure(figsize=(10, 4))
plt.plot(ecg_signal, label='Synthetic ECG Signal')
plt.title("Synthetic ECG Signal")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude")
plt.show()
```



Deduksi Heart Rate (detak jantung) Berdasarkan Sinyal ECG Kamu bisa mendeduksi detak jantung (***Beat Per Minute***) dari sinyal ini dengan cara. 1. Melihat jumlah peak (puncak) = 9 Beat 2. Melihat jumlah sampel waktu = 8000 ms = 8 detik

Kamu tinggal menggunakan formula sederhana ini

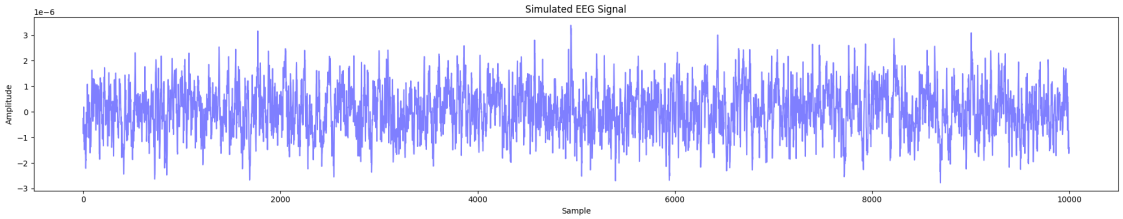
$$BPM = \frac{\text{Puncak}}{\text{Detik}} * 60 \text{ Detik} = \frac{9}{8} * 60 = 67.5 \text{ Beat Per Minute}$$

Sinyal EEG (*Electroencephalogram*). Sinyal aktivitas Otak Sinyal EEG adalah sinyal listrik yang dihasilkan oleh aktivitas otak. Sinyal ini dapat digunakan untuk mendiagnosis berbagai kondisi neurologis, seperti epilepsi, gangguan tidur, dan gangguan mental. Sinyal EEG biasanya direkam menggunakan elektroda yang ditempatkan di kulit kepala.

> Warning: Menjalankan code ini (hanya bagian EEG) akan mengunduh sample data MNE sebesar 2.95GB. Boleh di skip jika anda tidak punya bandwidth / internet yang memadai

[87]:
```
eeg_signal = nk.eeg_simulate(duration=10, sampling_rate=1000, noise=0.1,␣
↪random_state=2025)
```

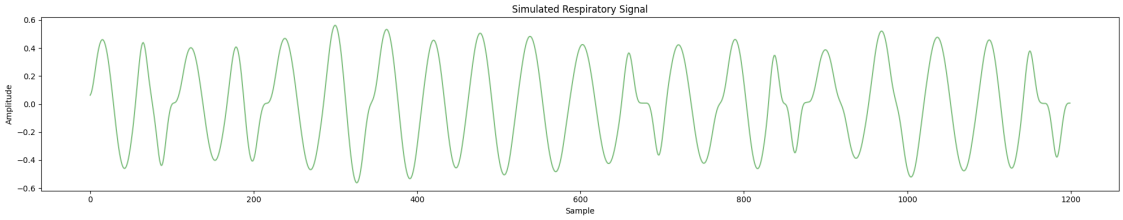
[88]:
```
plt.figure(figsize=(20, 4))
plt.plot(eeg_signal, color="blue", alpha=0.5)
plt.title("Simulated EEG Signal")
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.tight_layout()
plt.show()
```



Sinyal Respirasi (Respiration) Sinyal respirasi adalah sinyal yang dihasilkan oleh aktivitas pernapasan. Sinyal ini dapat digunakan untuk mendiagnosis berbagai kondisi pernapasan, seperti asma, penyakit paru obstruktif kronis (PPOK), dan sleep apnea. Sinyal respirasi biasanya direkam menggunakan sensor yang ditempatkan di dada atau perut (respitaroy belt).

[89]:
```
resp_signal = nk.rsp_simulate(duration=60, sampling_rate=20,␣
↪respiratory_rate=20, noise=0.1, random_state=2025)
```

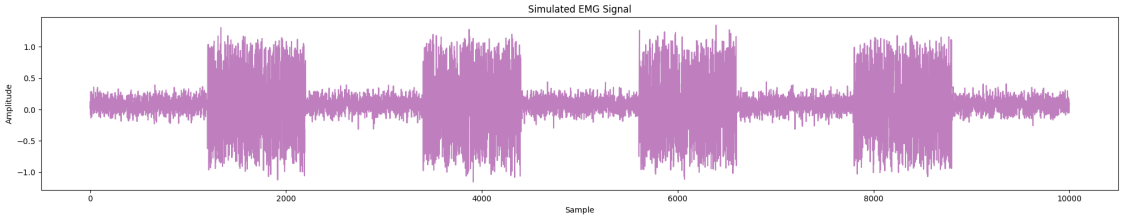
[90]:
```
plt.figure(figsize=(20, 4))
plt.plot(resp_signal, color="green", alpha=0.5)
plt.title("Simulated Respiratory Signal")
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.tight_layout()
plt.show()
```



Sinyal EMG (Electromyogram) Sinyal EMG adalah sinyal listrik yang dihasilkan oleh aktivitas otot. Sinyal ini dapat digunakan untuk mendiagnosis berbagai kondisi otot, seperti distrofi otot, neuropati, dan cedera saraf. Sinyal EMG biasanya direkam menggunakan elektroda yang ditempatkan di kulit di atas otot yang akan dianalisis.

[91]:
```
emg_singal = nk.emg_simulate(duration=10, sampling_rate=1000, noise=0.1,␣
↪burst_number=4 ,random_state=2025)
# Menampilkan plot sinyal EMG
plt.figure(figsize=(20, 4))
plt.plot(emg_singal, color="purple", alpha=0.5)
plt.title("Simulated EMG Signal")
plt.xlabel("Sample")
```

```
plt.ylabel("Amplitude")
plt.tight_layout()
plt.show()
```



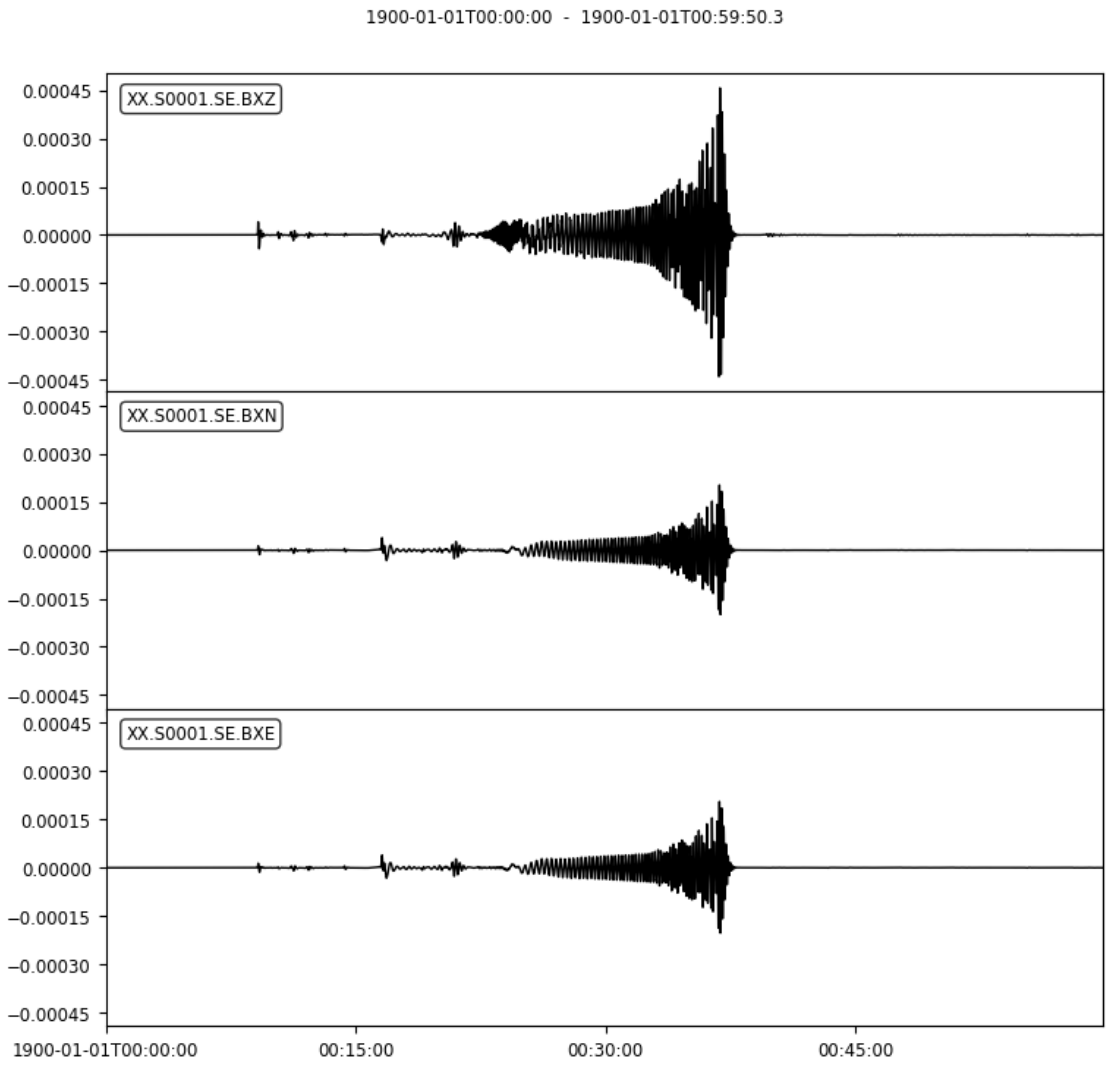
1.2.3 Sinyal Lingkungan (Seismic / Gempa Bumi)

Sinyal Seismik (Gempa Bumi) Sinyal seismik adalah sinyal yang dihasilkan oleh aktivitas gempa bumi. Sinyal ini dapat digunakan untuk mendiagnosis berbagai kondisi geologi, seperti gempa bumi, letusan gunung berapi, dan pergeseran tanah. Sinyal seismik biasanya direkam menggunakan sensor yang ditempatkan di permukaan tanah atau di dalam tanah (seismometer).

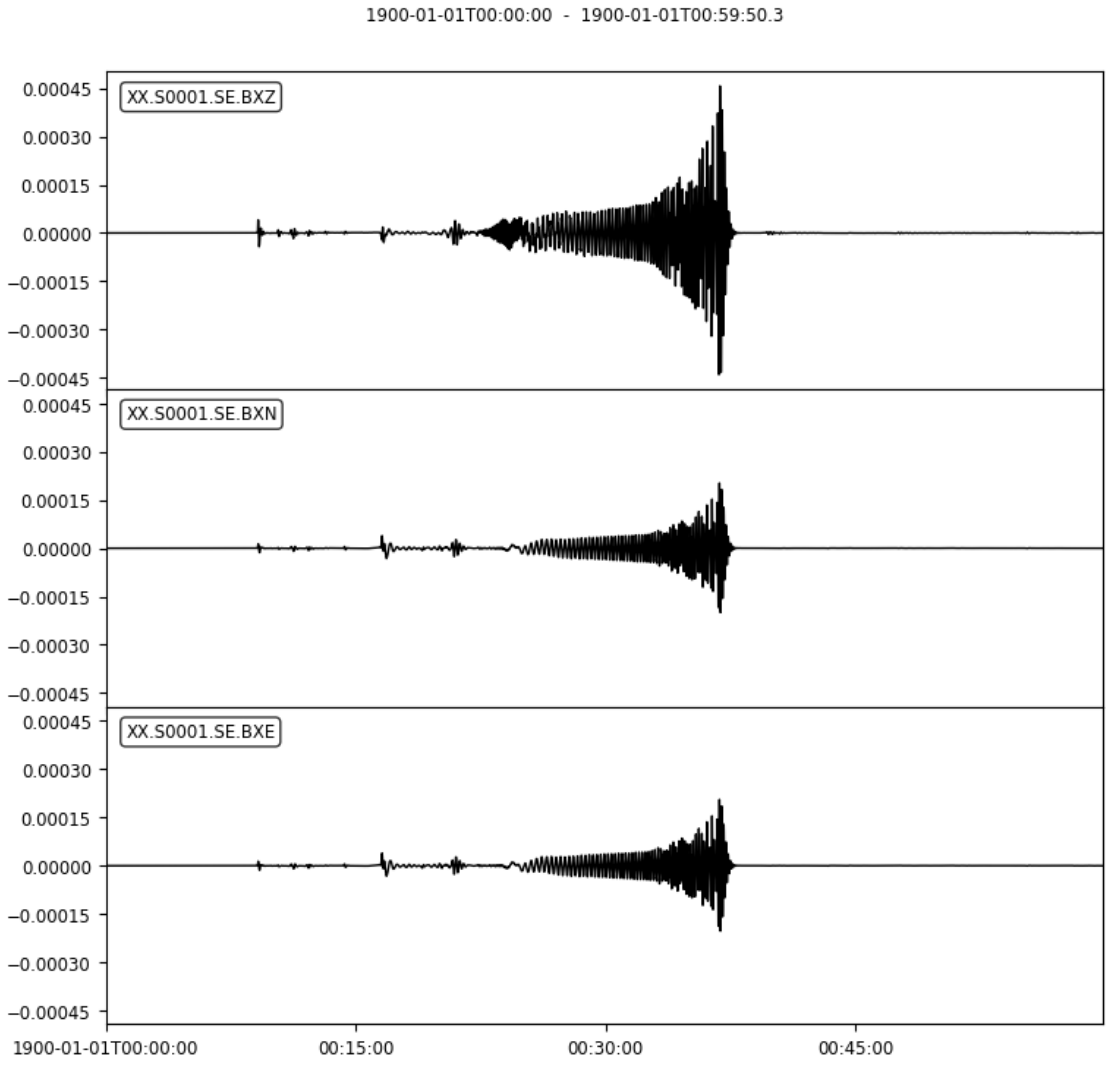
[ ]:
```
client = Client()
origin_time = UTCDateTime("2023-10-01T00:00:00")
# seismogram = client.get_waveforms(model="ak135f_2s", sourcelatitude=-7.
 ↪8007839033133575, sourcelongitude=110.37098235071159,␣
 ↪sourcedepthinmeters=10000, receiverlatitude=35.6895, receiverlongitude=139.
 ↪6917, components="ZNE", units="displacement") # Jogjakarta

seismogram = client.get_waveforms(model="ak135f_2s", sourcelatitude=-5.
 ↪440599180710982, sourcelongitude=105.27711959083427,␣
 ↪sourcedepthinmeters=10000, receiverlatitude=35.6895, receiverlongitude=139.
 ↪6917, components="ZNE", units="displacement") # Lampung
```

[92]:
```
# Menampilkan plot sinyal seismogram
plt.figure(figsize=(20, 4))
seismogram.plot()
```

```
<Figure size 2000x400 with 0 Axes>
```


1900-01-01T00:00:00 - 1900-01-01T00:59:50.3
XX.S0001.SE.BXZ
XX.S0001.SE.BXN
XX.S0001.SE.BXE
0.00045
0.00030
0.00015
0.00000
−0.00015
−0.00030
−0.00045
1900-01-01T00:00:00
00:15:00
00:30:00
00:45:00

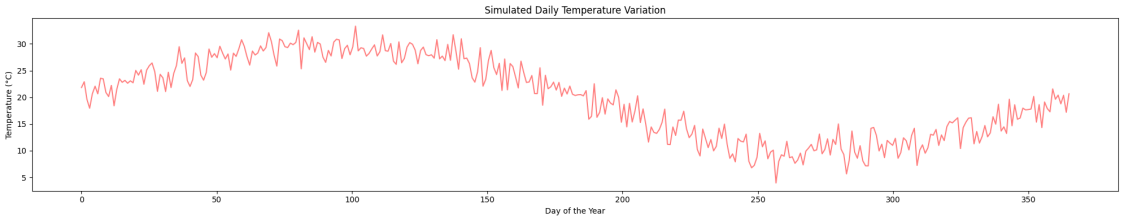
[92]:



Sinyal Perubahan Suhu Sinyal perubahan suhu adalah sinyal yang dihasilkan oleh perubahan suhu lingkungan. Sinyal ini dapat digunakan untuk mendiagnosis berbagai kondisi lingkungan, seperti perubahan iklim, cuaca ekstrem, dan polusi udara. Sinyal perubahan suhu biasanya direkam menggunakan sensor suhu yang ditempatkan di lingkungan sekitar.

[93]:
```
t = np.linspace(0, 365, 365) # Data selama 1 tahun

# Mensimulasikan variasi suhu harian dengan noise
temperature = 20 + 10 * np.sin(2 * np.pi * t / 365) + 2 * np.random.
 ↪normal(size=t.shape)

# Menvisualisasi data suhu
plt.figure(figsize=(20, 4))
plt.plot(t, temperature, color="red", alpha=0.5)
```

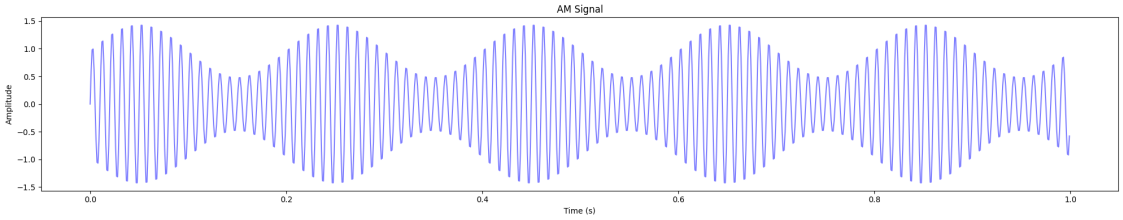
```
plt.title("Simulated Daily Temperature Variation")
plt.xlabel("Day of the Year")
plt.ylabel("Temperature (°C)")
plt.tight_layout()
plt.show()
```



1.2.4 Sinyal Komunikasi

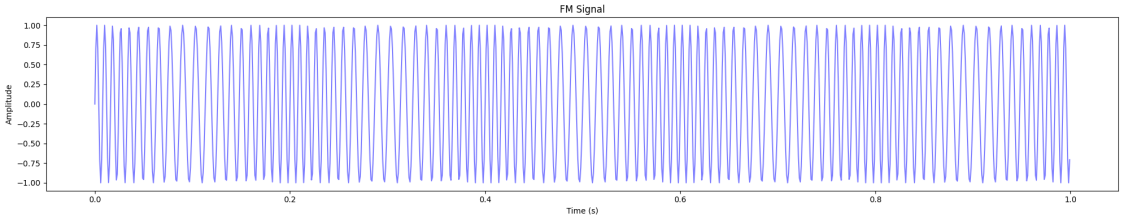
Sinyal Radio AM Sinyal radio AM bekerja dengan cara mengubah amplitudo dari sinyal pembawa sesuai dengan informasi yang ingin dikirimkan. Sinyal ini terdiri dari dua komponen utama: pembawa (carrier) dan modulasi. Pembawa adalah sinyal sinusoidal dengan frekuensi tetap, sedangkan modulasi adalah informasi yang ingin kita kirimkan. Dalam contoh ini, kita akan membuat sinyal radio AM sederhana.

[94]:
```
# Menginitialisasi parameter parameter yang digunakan untuk membuat sinyal␣
↪radio AM
fs = 1000 # Sampling frequency
carrier_frequency = 100 # Carrier frequency
modulating_frequency = 5 # Modulating frequency

t = np.linspace(0, 1, fs, endpoint=False) # Waktu dari 0 sampai 1 detik
carrier = np.sin(2 * np.pi * carrier_frequency * t) # Sinyal pembawa
modulating_signal = 1 + 0.5 * np.sin(2 * np.pi * modulating_frequency * t) #␣
↪Sinyal modulasi

# Membuat sinyal AM
am_signal = carrier * modulating_signal # Sinyal AM

# Menampilkan plot sinyal AM
plt.figure(figsize=(20, 4))
plt.plot(t, am_signal, color="blue", alpha=0.5)
plt.title("AM Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.tight_layout()
plt.show()
```



Sinyal Radio FM Berbeda dengan sinyal radio AM yang mengubah amplitudo dari sinyal pembawa, sinyal radio FM (Frequency Modulation) bekerja dengan cara mengubah frekuensi dari sinyal pembawa sesuai dengan informasi yang ingin dikirimkan. Sinyal ini terdiri dari dua komponen utama: pembawa (carrier) dan modulasi. Pembawa adalah sinyal sinusoidal dengan frekuensi tetap, sedangkan modulasi adalah informasi yang ingin kita kirimkan. Dalam contoh ini, kita akan membuat sinyal radio FM sederhana.

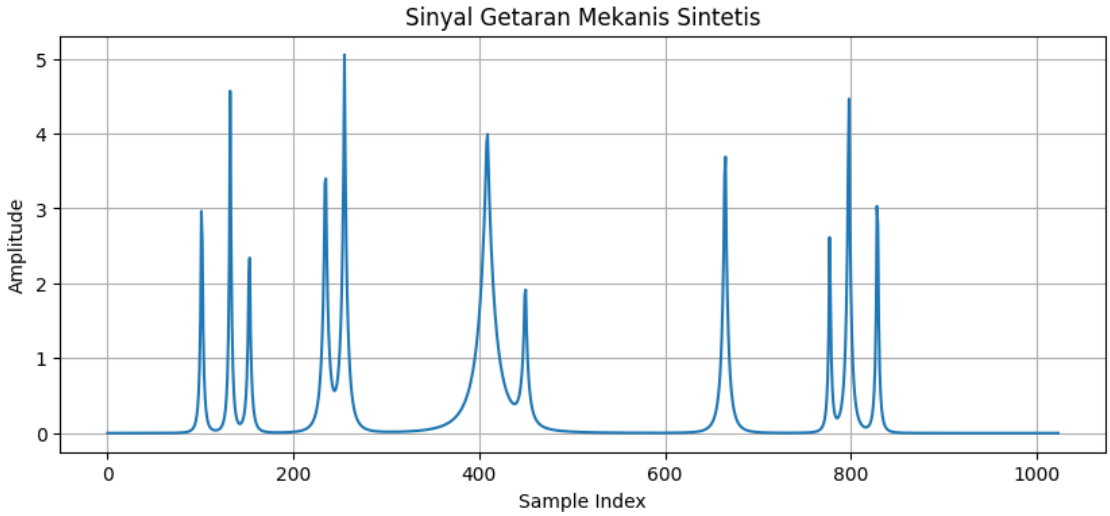
[95]:
```
#Menginitialisasi parameter parameter yang digunakan untuk membuat sinyal radio␣
 ↪FM
fs = 1000 # Sampling frequency
carrier_frequency = 100 # Carrier frequency
modulating_frequency = 5 # Modulating frequency
t = np.linspace(0, 1, fs, endpoint=False) # Waktu dari 0 sampai 1 detik

# Membuat sinyal pembawa
carrier = np.sin(2 * np.pi * carrier_frequency * t) # Sinyal pembawa
# Membuat sinyal modulasi
modulating_signal = np.sin(2 * np.pi * modulating_frequency * t) # Sinyal␣
 ↪modulasi
# Membuat sinyal FM
fm_signal = np.sin(2 * np.pi * carrier_frequency * t + 5 * modulating_signal) #␣
 ↪Sinyal FM

# Menampilkan plot sinyal FM
plt.figure(figsize=(20, 4))
plt.plot(t, fm_signal, color="blue", alpha=0.5)
plt.title("FM Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.tight_layout()
plt.show()
```



1.2.5 Sinyal Mekanik

Sinyal Getaran Mekanik Berbeda dengan sinyal **periodik** seperti **sinyal fisiologis tubuh (ECG)** atau sinyal sinusoidal, ada beberapa sinyal yang tidak terjadi secara periodik dan bergerak secara **acak / random**, sama seperti sinyal ISHG di atas

[24]:
```
import pywt

vibration_signal = pywt.data.demo_signal('Bumps', n=1024)

# Plot sinyal getaran
plt.figure(figsize=(10, 4))
plt.plot(vibration_signal)
plt.title('Sinyal Getaran Mekanis Sintetis')
plt.xlabel('Sample Index')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```



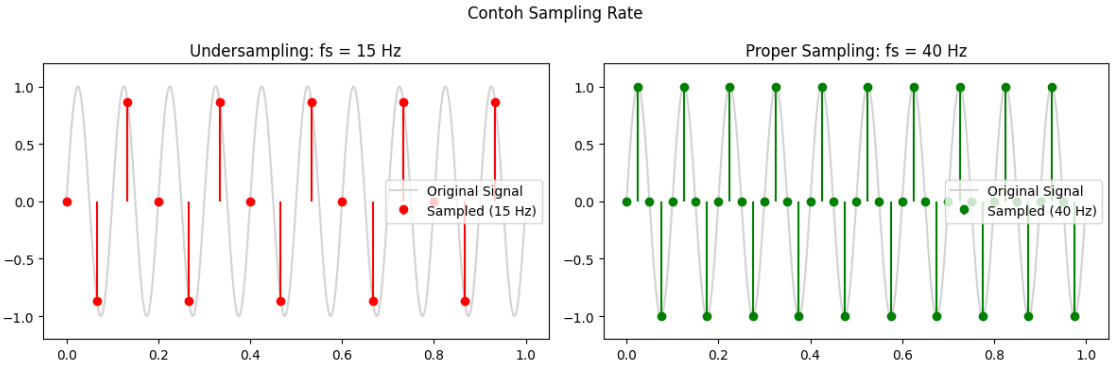
1.3 Sampling Rate - Sinyal Analog to Digital

1.3.1 Sampling = Kapan harus menangkap nilai

Sampling adalah proses untuk mengubah sinyal analog menjadi sinyal digital dengan menangkap nilai sinyal pada interval waktu tertentu.

Dibawah ini bakal ada dua contoh sinyal yang akan ditangkap nilainya dengan sampling rates yang berbeda, yakni dengan sampling rate `15` dan `40` Hz, dengan frekuensi sinyal utama sebesar `10` Hz.

[25]:
```
import numpy as np
import matplotlib.pyplot as plt

# Sinyal Utama
f_signal = 10  # Frekuensi Sinyal Utama
t_cont = np.linspace(0, 1, 1000, endpoint=False)
x_cont = np.sin(2 * np.pi * f_signal * t_cont)

# Two sampling rates: one below and one above Nyquist rate
fs_low = 15  # Below Nyquist (Nyquist = 2*10 = 20 Hz)
fs_high = 40  # Above Nyquist

def sample_signal(fs, t_max=1):
    t_sampled = np.arange(0, t_max, 1/fs)
    x_sampled = np.sin(2 * np.pi * f_signal * t_sampled)
    return t_sampled, x_sampled

t_low, x_low = sample_signal(fs_low)
t_high, x_high = sample_signal(fs_high)

# Plotting
fig, axs = plt.subplots(1, 2, figsize=(12, 4))

# Undersampling (Aliasing)
axs[0].plot(t_cont, x_cont, 'lightgray', label='Original Signal')
axs[0].stem(t_low, x_low, linefmt='r', markerfmt='ro', basefmt=" ",␣
 ↪label='Sampled (15 Hz)')
axs[0].set_title('Undersampling: fs = 15 Hz')
axs[0].legend()
axs[0].set_ylim(-1.2, 1.2)

# Proper sampling (No aliasing)
axs[1].plot(t_cont, x_cont, 'lightgray', label='Original Signal')
axs[1].stem(t_high, x_high, linefmt='g', markerfmt='go', basefmt=" ",␣
 ↪label='Sampled (40 Hz)')
axs[1].set_title('Proper Sampling: fs = 40 Hz')
axs[1].legend()
axs[1].set_ylim(-1.2, 1.2)

plt.suptitle("Contoh Sampling Rate")
plt.tight_layout()
plt.show()
```

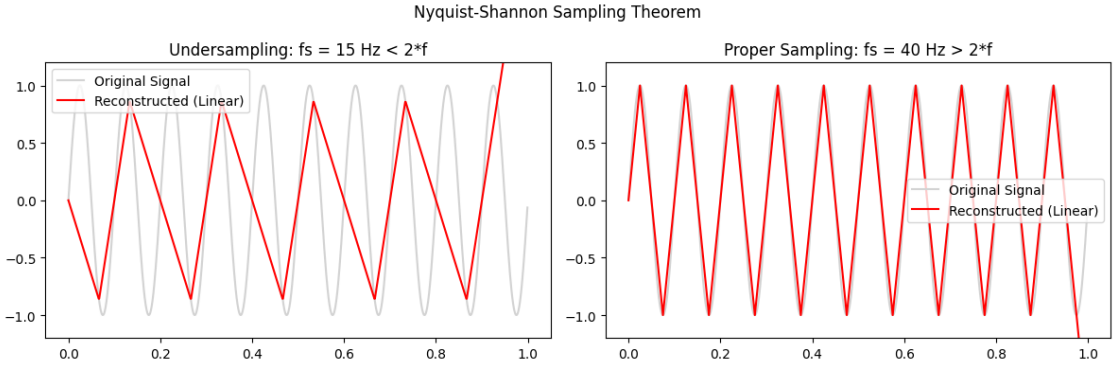


1.3.2 Teorema Sampling Nyquist–Shannon

Berdasakan hasil grafik di atas, bisa dilihat bahwa ketika misalnya dibuat sebuah garis antar titik, sinyal dengan `sampling rate` 40 Hz akan menghasilkan sinyal yang lebih baik di banding 15 Hz.

[26]:
```
from scipy.interpolate import interp1d

def interpolate_sampled(t_sampled, x_sampled, t_cont):
    """
    Interpolates the sampled signal to match the continuous time vector.
    """
    x_interp = interp1d(t_sampled, x_sampled, kind='linear',␣
 ↪fill_value="extrapolate")
    return x_interp(t_cont)

# Interpolate the sampled signals to the continuous time vector
x_low_interp = interpolate_sampled(t_low, x_low, t_cont)
x_high_interp = interpolate_sampled(t_high, x_high, t_cont)

## Plot the Undersampled and Properly Sampled Signals
fig, axs = plt.subplots(1, 2, figsize=(12, 4))

# Undersampling (Aliasing)
axs[0].plot(t_cont, x_cont, 'lightgray', label='Original Signal')
axs[0].plot(t_cont, x_low_interp, 'r', label='Reconstructed (Linear)')
axs[0].set_title('Undersampling: fs = 15 Hz < 2*f')
axs[0].legend()
axs[0].set_ylim(-1.2, 1.2)

# Proper sampling (No aliasing)
axs[1].plot(t_cont, x_cont, 'lightgray', label='Original Signal')
axs[1].plot(t_cont, x_high_interp, 'r', label='Reconstructed (Linear)')
axs[1].set_title('Proper Sampling: fs = 40 Hz > 2*f')
axs[1].legend()
```

```
axs[1].set_ylim(-1.2, 1.2)

plt.suptitle("Nyquist-Shannon Sampling Theorem")
plt.tight_layout()
plt.show()
```



Hal ini disebut dengan teorema `Sampling Nyquist-Shannon`, dimana

Untuk merekonstruksi secara akurat sinyal kontinu dari sampel:

$$f_s \geq 2 * f_{max}$$

Dimana: - : Laju sampling (sampel per detik) - : Frekuensi maksimum komponen sinyal

Jika $f_s \leq 2 * f_{max}$, akan terjadi aliasing — frekuensi palsu atau terdistorsi. Hal yang terjadi pada sinyal yang di sampling dengan `sampling rate` 15 Hz.

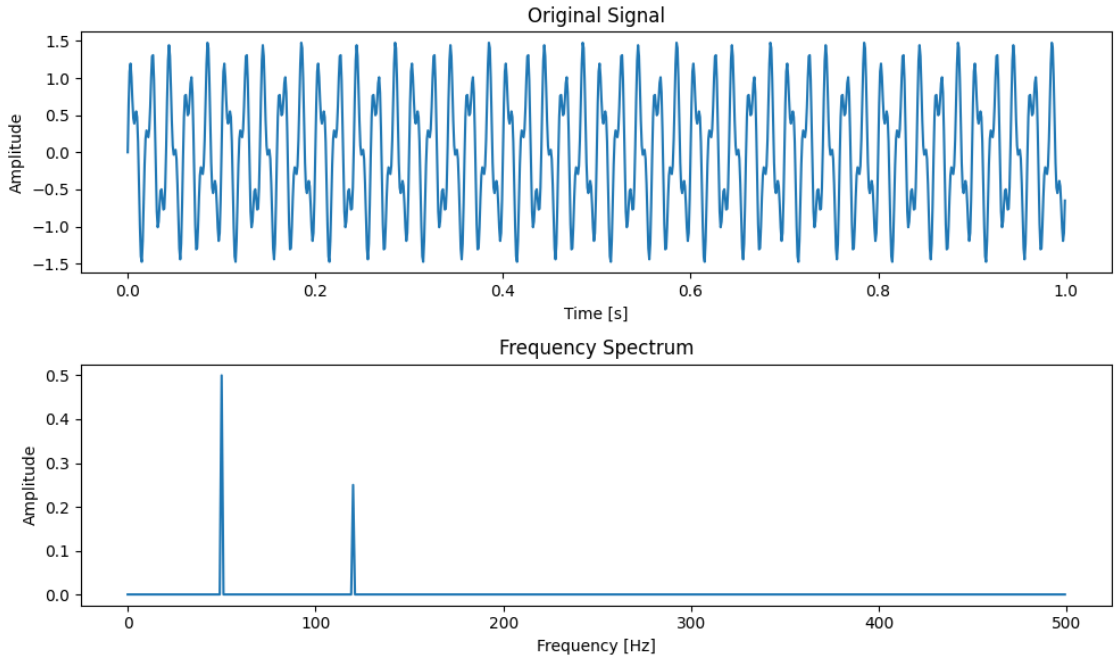
Apa maksud dari Frekuensi maksimum?? Secara sederhana, sinyal yang ada di sekitar kita merupakan gabungan dari `beberapa sinyal` dengan frekuensi yang berbeda menjadi satu. Kita bisa memecah sinyal tersebut menjadi `beberapa bagian` untuk mengetahui komponen frekuensi penysun sinyal tersebut.

Disini kita akan menggunakan Fourier Transform (Fungsi untuk mengubah sinyal dari `Time` domain ke `Frequenies` domain). > It's fine jika belum tau seputar Fourier transform, kamu bisa melihat tentang overview dari apa itu Fourier transform pada video berikut

Misalkan, jika dimiliki sebuah sinyal dengan frekuensi yang berbeda

[27]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq

# Simulate a signal (mix of 50 Hz and 120 Hz)
fs = 1000  # original sampling rate
T = 1.0    # duration in seconds
t = np.linspace(0.0, T, int(fs * T), endpoint=False)
```

```
## Gabungan dari dua sinyal
x = np.sin(2.0*np.pi*50.0*t) + 0.5*np.sin(2.0*np.pi*120.0*t)

# FFT
N = len(x)
xf = fftfreq(N, 1 / fs)
yf = np.abs(fft(x)) / N  # Normalize

## Plot Signal
fig, axs = plt.subplots(2, 1, figsize=(10, 6))

axs[0].plot(t, x)
axs[0].set_title("Original Signal")
axs[0].set_xlabel("Time [s]")
axs[0].set_ylabel("Amplitude")

axs[1].plot(xf[:N//2], yf[:N//2])
axs[1].set_title("Frequency Spectrum")
axs[1].set_xlabel("Frequency [Hz]")
axs[1].set_ylabel("Amplitude")

fig.tight_layout()
plt.show()
```



Bisa dilihat bahwa f_{max} berada pada `120` Hz, Jadi sesuai dengan `Teorema Nyquist`, minimum sampling rate harus dua kali dari frekuensi maksimum jadi $$ Minimum sampling rate = 2 *

f_{max} = 2 * 125 = 250 Hz

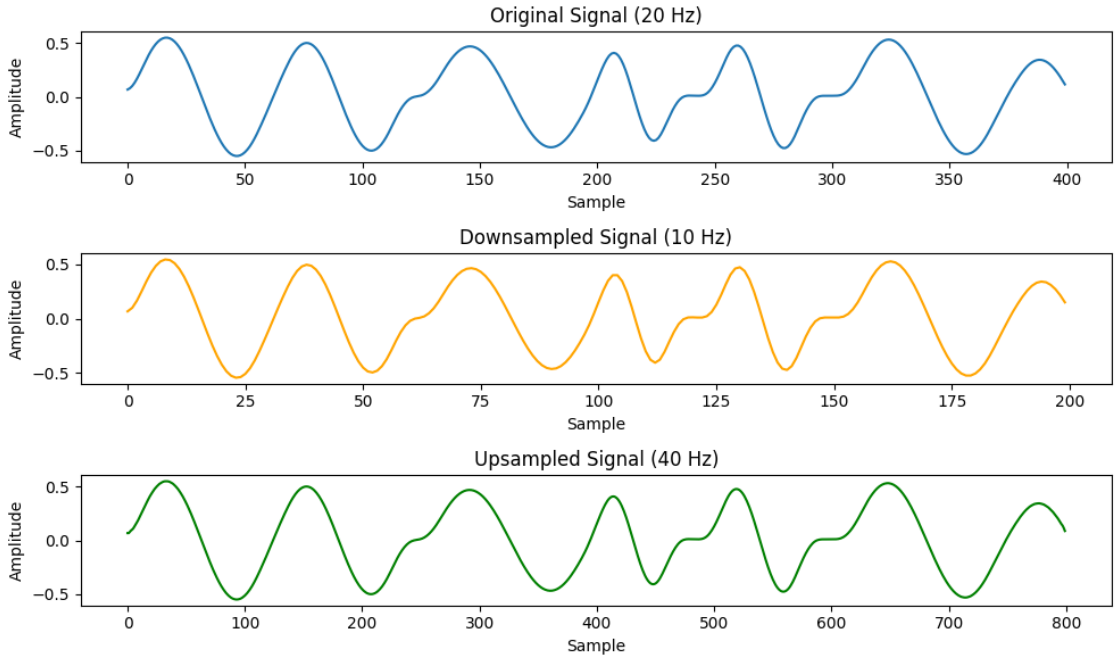
1.3.3 Resampling, Downsampling, Upsampling

Misalkan Kita punya sinyal respirasi dengan: - frekuensi sampling 20 Hz (Artinya ada 20 nilai dalam satu detik) - Total elemen (total sampel) dari sinyal ini adalah 400 sampel

Kita mempunya dua opsi 1. Menambah frekuensi sampling (menambah titik nilai diantara titik yang sudah ada, benefit sinyal semakin jelas, namun menambah proses komputasi dan ukuran sinyal) 2. Mengurangi frekuensi sampling (mengurangi nilai pada sinyal, membuat sinyal semakin pendek, namun proses komputasi yang lebih cepat)

[28]:
```
from scipy.signal import decimate, resample
import neurokit2 as nk

# Sinyal asli
fs = 20 ## Sampling rate 20 Hz
time = 20 ## 20 detik
resp_signal = nk.rsp_simulate(duration=time, sampling_rate=fs,␣
 ↪respiratory_rate=20, noise=0.01, random_state=2024)

# Upsampling
resp_upsampling = resample(resp_signal, 2*len(resp_signal))  # 20 Hz → 40 Hz␣
 ↪(dua kali lipat)

# Downsampling
resp_downsampling = decimate(resp_signal, 2)  # 20 Hz → 10 Hz

## Plot original and downsampled signals
fig, axs = plt.subplots(3, 1, figsize=(10, 6))

axs[0].plot(resp_signal, label='Original Signal')
axs[0].set_title("Original Signal (20 Hz)")
axs[0].set_xlabel("Sample")
axs[0].set_ylabel("Amplitude")

axs[1].plot(resp_downsampling, label='Downsampled Signal (10 Hz)',␣
 ↪color='orange')
axs[1].set_title("Downsampled Signal (10 Hz)")
axs[1].set_xlabel("Sample")
axs[1].set_ylabel("Amplitude")

axs[2].plot(resp_upsampling, label='Upsampled Signal (40 Hz)', color='green')
axs[2].set_title("Upsampled Signal (40 Hz)")
axs[2].set_xlabel("Sample")
axs[2].set_ylabel("Amplitude")

plt.tight_layout()
```

```
plt.show()
```



Bisa dilihat bahwa secara tidak langsung, `menambah / mengurangi sampling rate` dari sebuah sinyal akan mempengaruhi `jumlah sampel` dari sinyal itu sendiri

Question Apa yang bakal terjadi jika kamu melakukan downsampling secara extreme (misal dari 20 Hz ke 2 Hz), tulis analisis kamu.

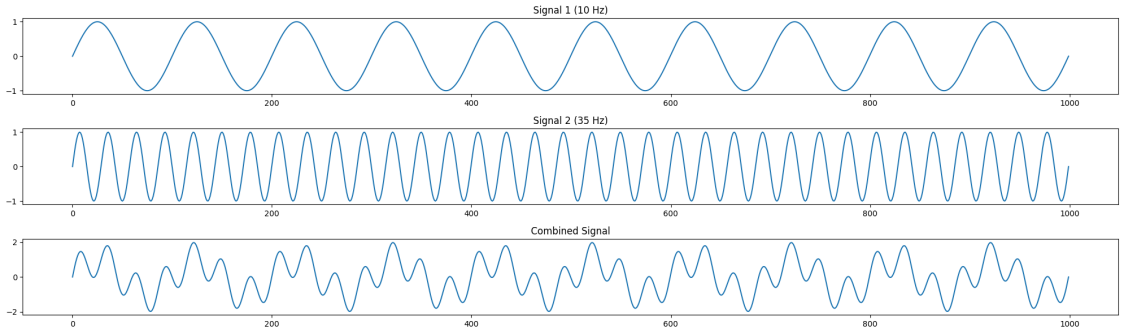
Filtering Filter digital membantu memilih atau menghapus komponen frekuensi tertentu dari sinyal.

Ada 3 jenis filter yang umum digunakan - (Lowpass) Melewatkan frekuensi rendah, blok tinggi - (Highpass) Melewatkan frekuensi tinggi, blok rendah - (Bandpass) Melewatkan rentang tertentu saja

Misalnya, kita punya dua buah sinyal yang digabungkan

[29]:
```
fs = 1000
time_axis = np.linspace(0, 1, fs)

signal_1 = np.sin(2 * np.pi * 10 * time_axis)
signal_2 = np.sin(2 * np.pi * 35 * time_axis)
signal_combine = signal_1 + signal_2

fig, ax = plt.subplots(3, 1, figsize=(20, 6))
ax[0].plot(signal_1)
```

```
ax[0].set_title("Signal 1 (10 Hz)")
ax[1].plot(signal_2)
ax[1].set_title("Signal 2 (35 Hz)")
ax[2].plot(signal_combine)
ax[2].set_title("Combined Signal")
plt.tight_layout()
plt.show()
```



Misalnya, sinyal 1 adalah sinyal yang informasinya kita butuhkan, sedangkan sinyal 2 adalah noise. Bagaimana cara memisahkan sinyal 1 dari sinyal 2?

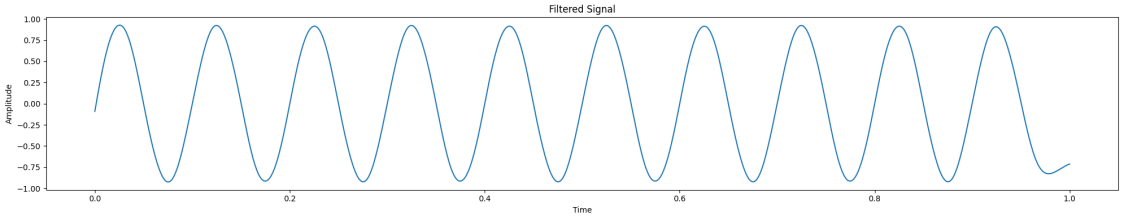
Disinilah kita butuh filter.

Question : Untuk mengetahui komponen frekuensi mana yang akan dipotong, kamu bisa menggunakan FFT untuk melihat komponen frekuensi dari sinyal. Coba buat FFT untuk sinyal di atas.

Mendesain Filter Kamu bisa menggunakan library `scipy.signal.butter` untuk membuat filter `bandpass`, `highpass` dan `lowpass`, untuk contoh kali ini akan menggunakan contoh `lowpass` filtering

[30]:
```
from scipy import signal

cutoff = 15 # Frekuensi cutoff dalam Hz
order = 3 # Orde filter

b, a = signal.butter(order, cutoff, fs=fs, btype='low', analog=False)

signal_filtered = signal.filtfilt(b, a, signal_combine)

plt.figure(figsize=(20, 4))
plt.plot(time_axis, signal_filtered)
plt.title("Filtered Signal")
plt.ylabel("Amplitude")
plt.xlabel("Time")
```

```
plt.tight_layout()
plt.show()
```



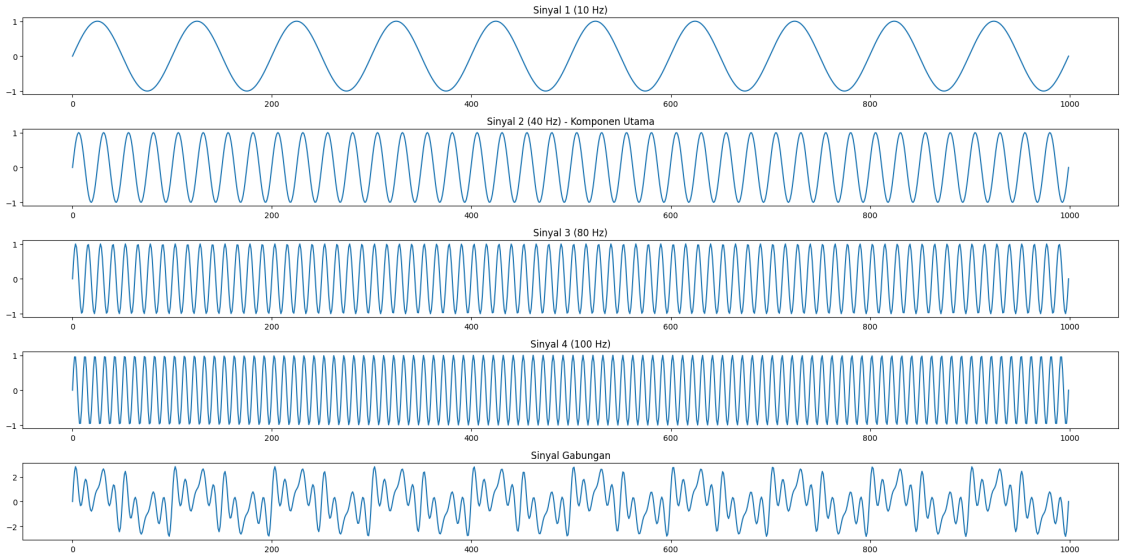
Yup sudah cukup mirip dengan sinyal 1. Cobalah untuk melakukan hal yang sama, namun kali ini gunakan highpass untuk mendapatkan sinyal 2.

1.4 Bagaimana kalau ada dua buah sinyal yang kita ingin buang?

Dimana, informasi yang kita butuhkan, dihimpit oleh dua buah sinyal yang tidak kita butuhkan.

Pertama-tama, mari kita buat sinyal-sinyalnya

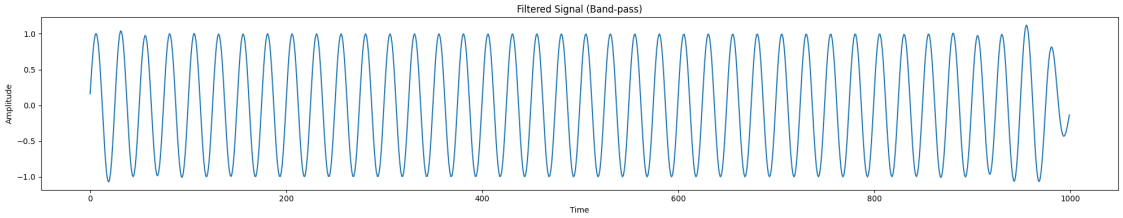
[31]:
```
fs = 1000
time_axis = np.linspace(0, 1, fs)

sinyal_1 = np.sin(2 * np.pi * 10 * time_axis) # 10 Hz
sinyal_2 = np.sin(2 * np.pi * 40 * time_axis) # 40 Hz
sinyal_3 = np.sin(2 * np.pi * 80 * time_axis) # 80 Hz
sinyal_4 = np.sin(2 * np.pi * 100 * time_axis) # 100 Hz
sinyal_gabungan = sinyal_1 + sinyal_2 + sinyal_3 + sinyal_4

fig, ax = plt.subplots(5, 1, figsize=(20, 10))
ax[0].plot(sinyal_1)
ax[0].set_title("Sinyal 1 (10 Hz)")
ax[1].plot(sinyal_2)
ax[1].set_title("Sinyal 2 (40 Hz) - Komponen Utama")
ax[2].plot(sinyal_3)
ax[2].set_title("Sinyal 3 (80 Hz)")
ax[3].plot(sinyal_4)
ax[3].set_title("Sinyal 4 (100 Hz)")
ax[4].plot(sinyal_gabungan)
ax[4].set_title("Sinyal Gabungan")
plt.tight_layout()
plt.show()
```



1.4.1 Melakukan Bandpass (Menghilangkan frekuensi rendah dan tinggi)

Maka kita akan lakukan bandpass di cutoff 27 dan 50 Hz untuk mengambil sinyal yang kita butuhkan (40 Hz)

[32]:
```
cutoff_low = 27 # Frekuensi cutoff dalam Hz
cutoff_high = 50 # Frekuensi cutoff dalam Hz
order = 3 # Orde filter

b, a = signal.butter(order, [cutoff_low, cutoff_high], fs=fs, btype='band',␣
↪analog=False)
signal_filt_band = signal.filtfilt(b, a, sinyal_gabungan)

plt.figure(figsize=(20, 4))
plt.plot(signal_filt_band)
plt.title("Filtered Signal (Band-pass)")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.tight_layout()
plt.show()
```



1.5 ## Tugas `Ho2`

1. Kamu adalah seorang intern pada sebuah perusahaan teknologi kesehatan. Perusahaan tersebut sedang mengembangkan teknologi wearable untuk mendeteksi detak jantung berdasarkan sinyal PPG (*Photoplethysmography*) yakni sinyal aliran darah dari jantung ke tangan. Kamu akan mensimulasikan sinyal PPG menggunakan library `neurokit2.ppg_simulate` (Ref.) dengan parameter sebagai berikut
   - Durasi: `Berdasarkan 3 digit terakhir nim anda`
   - Sampling Rate: `150 Hz`
   - Noise Level: `0.<2 digit nim terakhir>`
   - Heart Rate: `80` Beat Per Minute (BPM)
   - Random State: `tanggal bulan tahun lahir anda` dengan format YYMMDD misalnya `240925`

   a. Buktikanlah bahwa proses downsampling pada sinyal PPG akan menghilangkan informasi dari sinyal asi. Lakukan downsampling dari 150Hz ke 100Hz, 50Hz, 25Hz, 10Hz, hingga 5Hz. Jelaskan apa yang terjadi dan buktikan bahwa semakin rendah sampling frequency (fs) maka sinyal akan semakin terdistorsi dan terdapat `Aliasing` pada sinyal hasil downsampling. Jelaskan apa itu `Aliasing`

   b. Ketika kamu melakukan filtering, terdapat parameter `order` saat kamu melakukan filtering. Apa maksudnya dari `order` dan apa yang akan terjadi jika kamu mengganti nilai order terhadap sinyal?

2. Lakukan eksperimen dengan merancang filter band-pass menggunakan `signal.butter`. Pada sinyal respirasi (pernapasan). Generate sebuah signal dengan ketentuan:
   - Durasi: `Berdasarkan 3 digit terakhir nim anda`
   - Sampling Rate: `100 Hz`
   - Noise level: `0.<2 digit nim terakhir>`
   - `respiratory_rate`: `18` Breaths Per Minute (BPM)
   - `random_state`: `tanggal bulan tahun lahir anda` dengan format YYMMDD misalnya `240925`

Anda dapat dengan bebas bereksperimen menentukan frekuensi cutoff yang diinginkan namun jelaskan latar belakang penentuan letak frekuensi cutoff mengapa anda memilih di nilai tersebut.

3. Kamu sedang mencoba membuat filter audio sederhana, maka kamu mencoba untuk merekam suara audio menggunakan handphone / laptop dengan deskripsi berikut:
   - Anda harus berbicara selama 20 detik.
   - Bacaan / percakapan yang anda rekam haruslah berisi informasi mengenai diri anda, seperti nama, asal daerah, hobi, dll.
   - Lakukanlah perekaman di dekat sumber noise statis seperti kipas angin, AC, atau kipas laptop anda (atau apapun yang merupakan noise frekuensi tinggi)
   - Lakukanlah proses filtering pada audio tersebut dengan menggunakan bandpass. Temukan frekuensi cutoff untuk bandpass yang paling sesuai dengan karakteristik audio yang anda rekam.
   - Visualisasikan spektrum frekuensi menggunakan FFT dari audio sebelum di filter dan setelah di filter (dengan ketiga filter yang telah anda buat).

1.6 Notes:

- Link tugas dan informasi tentang batas waktu pengumpulan akan diumumkan di halaman web perkuliahan. Pastikan untuk rutin mengeceknya.
- Berkas yang dikumpulkan terdiri dari hasil render dalam format `PDF`. Jika hasil render notebook langsung ke PDF kurang rapih, disarankan untuk render terlebih dahulu dalam format `HTML`, lalu buka file HTML tersebut melalui browser dan simpan sebagai PDF. Selain `PDF`, unggah juga file notebook (`.ipynb`).
- Sebelum mengunggah file notebook (.ipynb), pastikan semua cell sudah dijalankan dengan jelas dan hasilnya terlihat dengan baik. Gunakan format penamaan file sebagai berikut: `nim.pdf` untuk file PDF, dan `nim.ipynb` untuk file notebook.
- Anda diperbolehkan berdiskusi dengan teman, meminta bantuan tutor, atau menggunakan AI LLM. Namun, Anda wajib mencantumkan atribusi kepada setiap pihak yang membantu, mencantumkan sumber referensi yang digunakan, serta melampirkan transkrip lengkap percakapan dengan AI LLM di bagian paling bawah notebook.
- Gunakan format wav untuk filetype dari audio recording, kamu bisa menggunakan converter mp3 ke wav melalui link berikut, dengan opsi buat Channel `mono` dan Sample-rate `44.1 Khz`.
- Enjoy :)

```
<img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRBGueNG1NScY4kuwTPSb2wrS0N_Tsgl
<p><em></em></p>
```