Signal&Systems for Comp.Eng.

BLG 354E

Project 1 Report

Mustafa Can Çalışkan
caliskanmu20@itu.edu.tr

Faculty of Computer and Informatics Engineering
Department of Computer Engineering

1. Introduction

The project addressed four different problems. In the first one, techniques from audio processing and signal processing were employed to extract a hidden message from a given audio file. The second problem involved filtering the extracted message to enhance its clarity, resulting in the creation of a refined output file. For the third problem, a scenario was simulated where live microphone data was analyzed to control a specific movement, demonstrating an application of real-time signal analysis for control purposes. Lastly, in the fourth problem, Bode diagrams were generated for given system functions, exploring the frequency-domain behavior of the systems. This project encompassed a wide range of skills including audio processing, signal analysis, control systems, and frequency-domain analysis.

2. Solutions

The solution steps for each question are provided below.

2.1. Question 1

1. The audio file is loaded using the `librosa` library, and the sampling rate is obtained.
2. The length of each frame is defined, and the number of frames in the audio file is calculated.
3. Iterating through each frame:
   - The start and end indices for the current frame are defined.
   - The current frame is extracted.
   - Fourier transform is applied to the frame.
   - The Fourier-transformed frame is split, and the first half is discarded.
   - Inverse Fourier transform is applied to the modified frame.
   - The real part of the inverse transformed frame is appended to the list of message frames.
4. The hidden message is reconstructed by concatenating all message frames.
5. The extracted message is written to a new audio file.
6. Finally, a message indicating the successful extraction of the hidden message is printed.

2.2. Question 2

1. The relevant audio file is loaded using the ‘soundfile‘ library, and the sampling rate is obtained.
2. Filter coefficients necessary for the filtering operation are determined.
3. A linear filtering operation is applied to the audio file using the ‘lfilter‘ function of the determined filter.
4. The filtered audio file is written to a file along with the specified sampling rate.
5. Finally, a message indicating the successful completion of the filtering operation on the extracted message is printed.

2.3. Question 3

2.3.1. `capture_audio()`

- This function initializes the microphone and continuously records audio data.
- It uses the ‘soundcard‘ library to access the microphone and records audio data in chunks.
- The recorded audio data is then yielded as the mean value of each chunk.

2.3.2. `analyze_heartbeat(signal)`

- This function takes a signal as input and calculates its derivative, representing the change between consecutive samples.
- It identifies the index of the minimum and maximum values of the derivative and calculates the difference between them.
- Based on the difference, it classifies the heartbeat type as "square" or "triangular".
- The result is printed, indicating the detected heartbeat type.

Even if I cannot navigate the monster to the exit correctly, I can still distinguish the signals correctly. The monster can be manually brought close to the mined area and the analysis function can be controlled.

2.3.3. `control_monster(heartbeat_type)`

- This function is intended to control the movements of a virtual entity based on the detected heartbeat type.
- Currently, the function is commented out, and no actual control logic is implemented.

2.3.4. Execution

- The script waits for 3 seconds before starting the audio stream.
- It continuously captures audio data from the microphone and analyzes it for heartbeat detection.
- The detected heartbeat type is used to control a virtual entity, although the control logic is not currently implemented.

2.4. Question 4

2.4.1. Transfer Functions

- The first transfer function $H_1(s)$ is defined by the numerator $10s + 10$ and the denominator $s^2 + 20s + 100$.
- The second transfer function $H_2(s)$ is defined by the numerator s and the denominator $s + 10$.
- The cascade transfer function $H_c(s) = H_1(s) \cdot H_2(s)$ is defined as the product of $H_1(s)$ and $H_2(s)$.

2.4.2. Bode Plots

- Frequency range is defined logarithmically from 10^{-2} to 10^2 with 500 points.
- Bode plots for each transfer function are generated using the `bode` function from the `scipy.signal` module.
- Each Bode plot consists of magnitude and phase plots.
- Plots are displayed in a 3×2 grid using `matplotlib`.
- Finally, the plots are saved as an image file named `q4Graphs.png` and displayed.

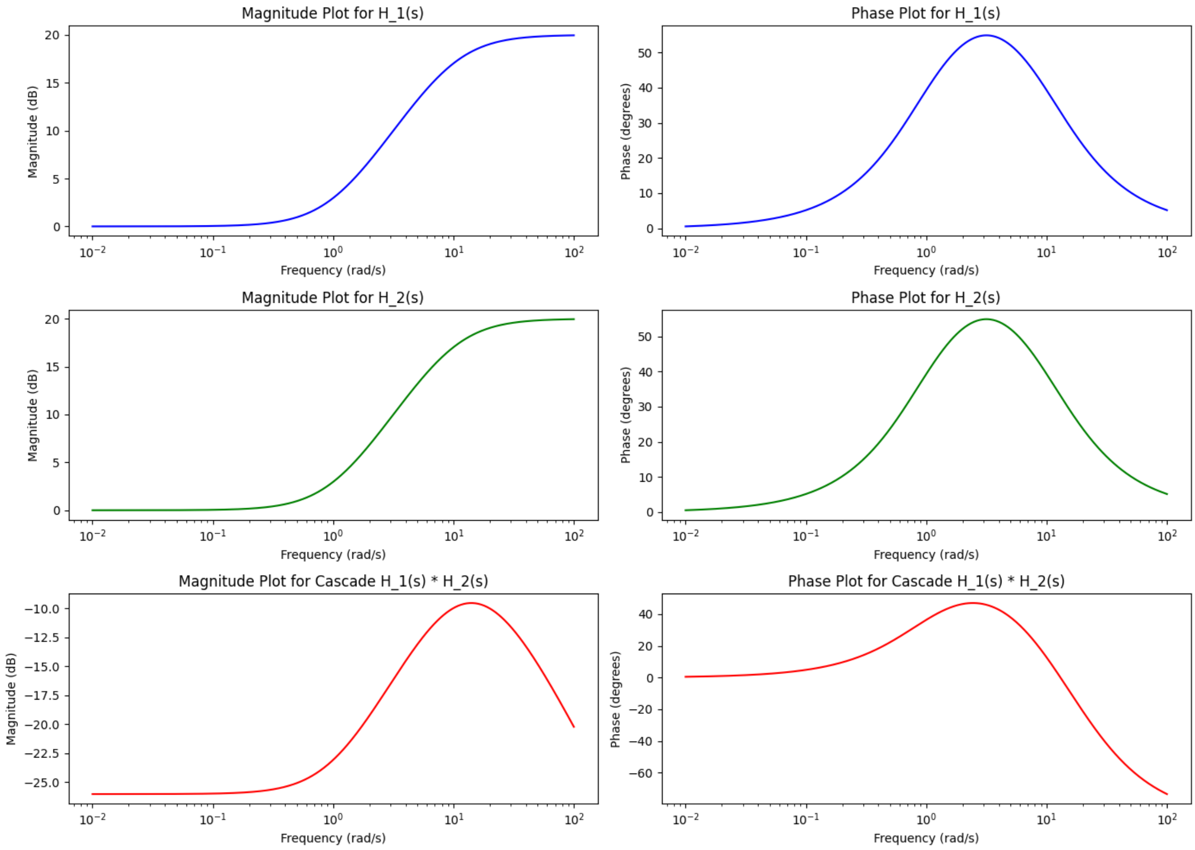

Figure 2.1: Bode Plots

3. Conclusion

The project encompassed diverse challenges in audio processing, signal analysis, real-time control, and frequency-domain analysis. We employed Fourier transformation and signal manipulation to extract a hidden message, followed by filtering techniques to enhance its clarity. Real-time microphone data analysis enabled control of specific movements, demonstrating practical signal analysis for control purposes. Bode plots were utilized to visualize the frequency response of system functions, offering insights into their stability. Overall, the project provided a comprehensive learning experience, integrating fundamental concepts in signal processing and control systems.