Báo cáo thuyết minh
Thiết kế ứng dụng bộ cân bằng âm thanh số (Equalizer)

1. Tóm tắt báo cáo

Trong báo cáo này, chúng tôi trình bày quá trình thiết kế và triển khai bộ cân bằng số (Equalizer) sử dụng bộ lọc FIR nhằm cải thiện chất lượng âm thanh đầu ra của loa bluetooth. Bộ cân bằng số này cho phép điều chỉnh các dải tần số (bass, mid, treble) theo thời gian thực thông qua giao diện người dùng trên máy tính, tích hợp bộ lọc FIR với cửa sổ Hamming. Kết quả cho thấy sự hiệu quả của việc điều chỉnh dải tần với đáp ứng tốt ở các tần số cắt mong muốn.

2. Giới thiệu

Bộ lọc số ngày càng được ứng dụng rộng rãi trong các thiết bị âm thanh và hệ thống xử lý tín hiệu số. Bộ lọc FIR được lựa chọn trong dự án này nhờ đặc tính ổn định và khả năng triển khai dễ dàng trên phần mềm mà không yêu cầu tài nguyên lớn. Với mục tiêu thiết kế một bộ cân bằng số, chúng tôi chia các dải tần thành ba băng chính: bass (tần số thấp), mid (tần số trung bình) và treble (tần số cao), từ đó giúp người dùng điều chỉnh các thành phần âm thanh theo nhu cầu.

3. Môi trường hoạt động

- **Nền tảng**: Chạy trên máy tính sử dụng Python với giao diện đồ họa dựa trên thư viện Tkinter.
- **Tính năng**:
    - Điều chỉnh các dải tần âm thanh theo thời gian thực.
    - Tương tác thông qua các thanh trượt (sliders) cho phép tăng giảm độ lợi của từng băng tần.
    - Visualizer nhảy theo tín hiệu âm thanh.

4. Cơ sở lý thuyết

a. Bộ lọc FIR

Bộ lọc FIR (Finite Impulse Response) là bộ lọc có đáp ứng xung hữu hạn và được thiết kế dễ dàng thông qua các kỹ thuật cửa sổ, phương pháp chuyển đổi tần số hoặc tối ưu hóa trực tiếp. FIR có thể được cấu hình như một bộ lọc thông thấp, thông cao, thông dải, hay chặn dải bằng cách chọn tần số cắt và kiểu cửa sổ phù hợp.

b. Cửa sổ Hamming

Cửa sổ Hamming được sử dụng để giảm nhiễu biên của tín hiệu và kiểm soát hiện tượng rò rỉ phổ, là yếu tố quan trọng để thiết kế bộ lọc FIR với đáp ứng tần số sắc nét. Công thức của cửa sổ Hamming được mô tả như sau:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right)$$

c. Số lượng hệ số (Taps)

Số lượng hệ số (taps) càng lớn sẽ tạo ra bộ lọc với đáp ứng tần số càng sắc nét nhưng lại làm tăng độ trễ và phức tạp tính toán. Dự án này sử dụng số lượng taps là 101 cho từng dải tần.

d. Tần số lấy mẫu

Chọn tần số lấy mẫu là 44.1 kHz (tương tự tần số lấy mẫu của đĩa CD) để đảm bảo chất lượng âm thanh và đủ dải tần cần xử lý.

e. Phân chia băng tần

Thiết kế chia thành 3 băng:

- Bass: 0 - 200 Hz
- Mid: 200 Hz - 2 kHz
- Treble: 2 kHz - 20 kHz

5. Thiết kế

a. Thiết kế bộ lọc cho các băng tần

Đây là một thiết kế equalizer 3 băng tần cơ bản cho phép người dùng điều chỉnh bass, mid và treble một cách độc lập, nghe theo thời gian thực hoặc lưu về.

- Định nghĩa băng tần trong phương thức **init**:
- Các băng tần được chia như sau:

```
class ThreeBandEqualizer:
    def __init__(self):
        self.sample_rate = 44100
        self.frame_size = 1024
        self.audio_queue = queue.Queue(maxsize=10)

        # Define frequency bands
        self.nyquist = self.sample_rate / 2
        self.low_cutoff = [20 / self.nyquist, 200 / self.nyquist]
        self.mid_cutoff = [200 / self.nyquist, 2000 / self.nyquist]
        self.high_cutoff = [2000 / self.nyquist, 20000 / self.nyquist]

        # Initialize gains (in linear scale)
        self.gains = np.ones(3)

        # Number of taps for FIR filters
        self.num_taps = 101

        self.filters = self._design_filters()
        self.player = AudioPlayer(self.sample_rate, self.frame_size)
```

- Băng tần thấp (Bass): 20 Hz - 200 Hz
- Băng tần trung (Mid): 200 Hz - 2000 Hz
- Băng tần cao (Treble): 2000 Hz - 20000 Hz

- Thiết kế bộ lọc trong phương thức **design_filters()**

```
def _design_filters(self):
    """Design FIR filters for each band using the improved method"""
    filters = []

    # Low-band filter (band-pass)
    filters.append(signal.firwin(self.num_taps, self.low_cutoff,
                                 pass_zero=False, window='hamming'))

    # Mid-band filter (band-pass)
    filters.append(signal.firwin(self.num_taps, self.mid_cutoff,
                                 pass_zero=False, window='hamming'))

    # High-band filter (band-pass)
    filters.append(signal.firwin(self.num_taps, self.high_cutoff,
                                 pass_zero=False, window='hamming'))

    return filters
```

- Code sử dụng hàm firwin từ thư viện scipy.signal để tạo các bộ lọc FIR (Finite Impulse Response) bandpass cho mỗi băng tần. Mỗi bộ lọc được thiết kế với:
    - Num_taps: 101 (số lượng hệ số của bộ lọc).
    - window=’hamming’: sử dựng cửa sổ hamming để giảm thiểu hiện tượng ringing.
    - pass_zero = False: chỉ định đây là bộ lọc bandpass.

- Xử lí âm thanh trong phương thức process_audio():

```python
def process_audio(self, audio_data):
    """Process audio through the three-band equalizer"""
    output = np.zeros_like(audio_data)

    for i in range(3):
        # Apply band-specific filter and gain
        filtered = signal.lfilter(self.filters[i], [1.0], audio_data)
        output += filtered * self.gains[i]

    # Normalize output to prevent clipping
    max_val = np.max(np.abs(output))
    if max_val > 1.0:
        output = output / max_val

    return output
```

- Quá trình xử lý:
    - Tạo mảng output rỗng.
    - Với mỗi băng tần:
        - Áp dụng bộ lọc bằng signal.lfilter
        - Nhân kết quả với hệ số gain của băng tần đó.
        - Cộng vào output.
    - Chuẩn hóa output để tránh clipping.
- Người dùng có thể điều chỉnh gain của từng băng tần thông qua thanh trượt trong GUI, với phương thức set_gain()

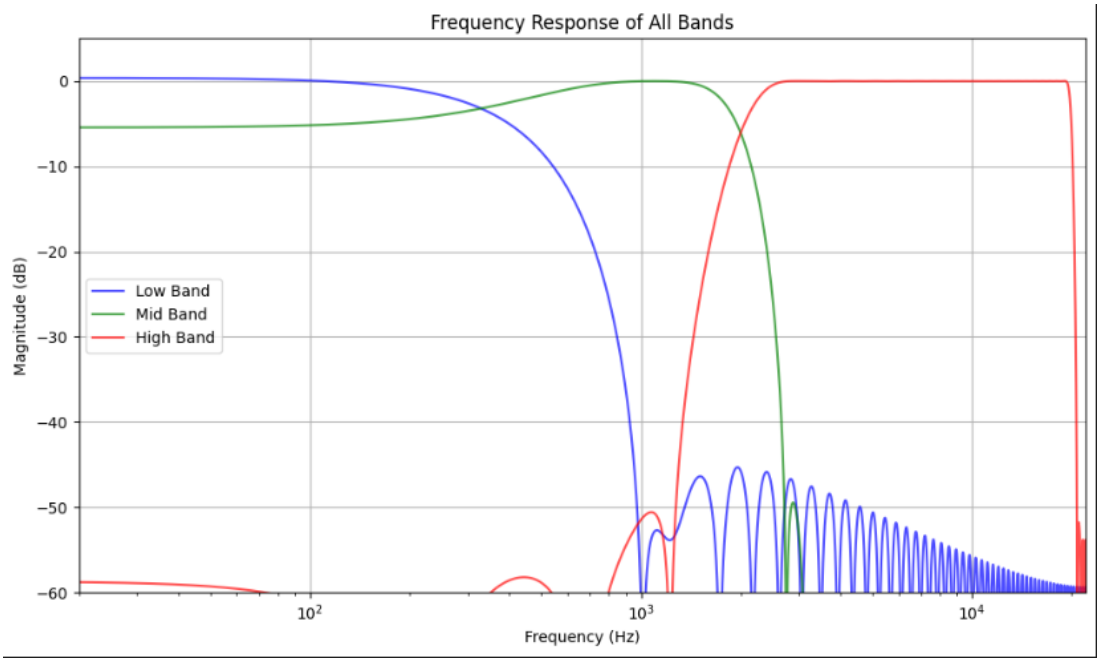
```python
def set_gain(self, band_idx, gain_db):
    """Set gain for a specific band (in dB)"""
    self.gains[band_idx] = 10 ** (gain_db / 20)
```

    - Gain được chuyển đổi từ dB sang hệ số khuyêchs đại tuyến tính thông qua công thức: gain_linear=10^(gain_dB/20).
- Đáp ứng tần số của tất cả các dải tần: đồ thị thể hiện cách mỗi bộ lọc trong equalizer phản ứng với các tần số khác nhau trong âm thanh.
    - Trục X (frequency): thể hiện tần số từ thấp đến cao (20Hz-200Hz).
    - Trục Y (magnitude): thể hiện độ lợi/suy giảm của tín hiệu (dB).

```python
def plot_frequency_responses(self):
    """Plot frequency response of all filters"""
    fig, ax = plt.subplots(figsize=(10, 6))
    colors = ['b', 'g', 'r']
    labels = ['Low', 'Mid', 'High']

    for i, (filt, color, label) in enumerate(zip(self.filters, colors, labels)):
        w, h = signal.freqz(filt, worN=8000)
        freq = (w / np.pi) * self.nyquist
        response = 20 * np.log10(np.abs(h))
        ax.semilogx(freq, response, color, label=f'{label} Band', alpha=0.7)

    ax.set_title('Frequency Response of All Bands')
    ax.set_xlabel('Frequency (Hz)')
    ax.set_ylabel('Magnitude (dB)')
    ax.set_xlim(20, self.nyquist)
    ax.set_ylim(-60, 5)
    ax.grid(True)
    ax.legend()
    plt.tight_layout()
    plt.show()
```

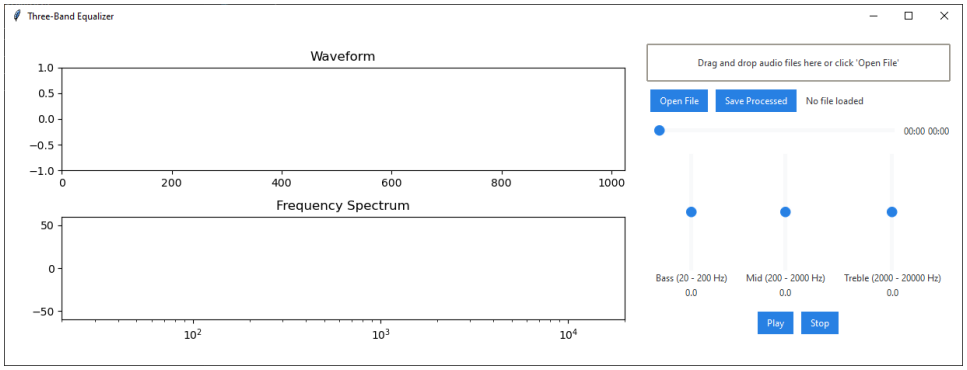


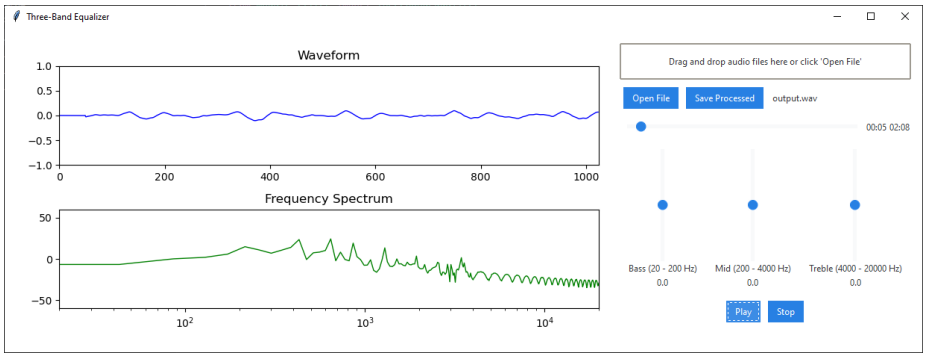
- Đồ thị cho phép ta thấy được: dải tần số mà mỗi bộ xử lí lọc.
- Độ mạnh của suy giảm tín hiệu ngoài dải.
- Độ chồng lấp giữa các dải tần.
- Chất lượng của bộ lọc (độ dốc của các đường cắt).

b. Thiết kế giao diện người dùng

Giao diện điều khiển bao gồm các thanh trượt điều chỉnh độ lợi của từng băng tần. Visualizer hiển thị tín hiệu âm thanh theo thời gian thực, cập nhật ngay khi người dùng thay đổi bất kỳ thông số nào, các nút Play và Stop để phát và dừng file âm thanh đầu vào.

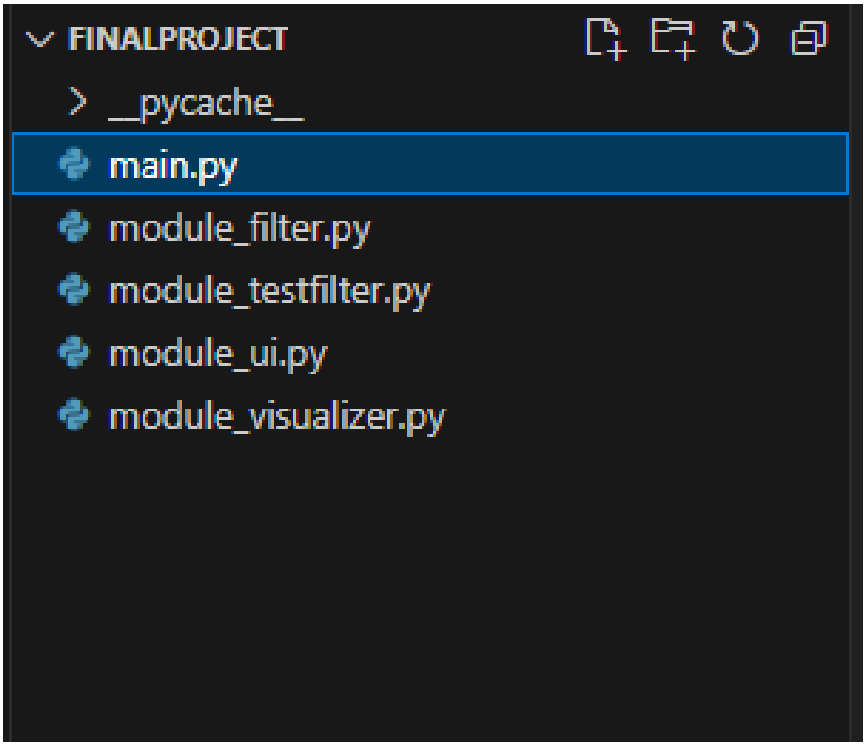
Giao diện tổng quan:





- Giao diện bao gồm các thành phần:
  - Nhập tệp âm thanh
  - Lưu âm thanh đã xử lý
  - Xem dạng sóng và phổ tần số
  - Xem thanh tiến trình nhạc
  - Điều chỉnh mức tăng âm trầm, âm trung, âm bổng
  - Phát/dừng nhạc

c. Tổ chức chương trình



Mã nguồn được chia thành các module chính:

- **module_filter.py**: Chứa các hàm xử lý lọc số và các hàm, các lớp hỗ trợ cho việc lọc.
- **module_ui.py**: Giao diện người dùng, kết nối với các thanh trượt điều chỉnh độ lợi.
- **module_visualizer.py**: Visualizer cho tín hiệu đầu vào và đầu ra.
- **module_testfilter.py**: Chứa các test case để kiểm tra các hàm chức năng lọc chính của bộ lọc trong module_filter.py.

6. Phương pháp test ứng dụng

a. Test module lõi

- Kiểm tra đáp ứng băng tần của các bộ lọc băng tần thấp, trung, và cao.
- Kiểm tra độ suy giảm ngoài băng tần để đảm bảo rằng tín hiệu ngoài băng tần được lọc đủ mạnh.
- Kiểm tra bảo toàn năng lượng của tín hiệu trong băng tần sau khi lọc.
- Kiểm tra đáp ứng pha tuyến tính của bộ lọc để đảm bảo tín hiệu đầu ra không bị biến dạng pha.
- Kiểm tra độ ổn định và độ dài tín hiệu sau khi lọc.
- Kiểm tra độ lệch pha giữa tín hiệu đầu vào và đầu ra.
- Kiểm tra phản hồi với tín hiệu tần số đơn nằm trong băng thông.
- Kiểm tra phản hồi với tín hiệu tần số đơn nằm ngoài băng thông.
- Kiểm tra bộ lọc băng tần trung với tín hiệu white noise.
- Kiểm tra bộ lọc băng tần cao với tín hiệu tần số cao.

test_low_band_filter, test_mid_band_filter, test_high_band_filter: Kiểm tra đáp ứng tần số của các bộ lọc băng tần thấp, trung và cao. Đảm bảo rằng bộ lọc duy trì gain lớn trong băng tần và suy giảm ngoài băng tần.

test_apply_filter_energy_conservation: Đảm bảo rằng năng lượng của tín hiệu trong băng tần được bảo toàn sau khi lọc.

test_linear_phase_response: Kiểm tra đáp ứng pha tuyến tính của bộ lọc để đảm bảo bộ lọc không gây biến dạng pha.

test_phase_shift: Kiểm tra độ lệch pha giữa tín hiệu đầu vào và đầu ra, đảm bảo bộ lọc không gây ra sự thay đổi pha lớn.

test_apply_filter_output_stability: Đảm bảo tín hiệu đầu ra không vượt quá phạm vi giá trị của tín hiệu đầu vào, kiểm tra tính ổn định của bộ lọc.

test_apply_filter_output_length: Đảm bảo rằng độ dài tín hiệu sau khi lọc không thay đổi.

test_single_frequency_response_within_band: Kiểm tra rằng bộ lọc không làm mất năng lượng của tín hiệu tần số đơn nằm trong băng thông.

test_single_frequency_response_outside_band: Đảm bảo rằng tín hiệu tần số đơn nằm ngoài băng tần bị suy giảm đủ mạnh.

b. Test các module khác

Test giao diện bằng cách thay đổi các thanh trượt, tương tác với các nút và quan sát kết quả. Test visualizer bằng cách nhập các tín hiệu âm thanh và kiểm tra tính chính xác của hiển thị.

7. Kết quả thực hiện

a. Kết quả test module lõi và module khác

- Bộ lọc bảo toàn năng lượng tín hiệu trong băng tần mong muốn.
- Bộ lọc ổn định, không gây hiện tượng quá tải trong tín hiệu đầu ra.
- Bộ lọc giữ nguyên độ dài tín hiệu, duy trì tính toàn vẹn của dữ liệu.
- Bộ lọc duy trì năng lượng của tín hiệu trong băng thông, đảm bảo truyền tải chính xác.
- Bộ lọc hoạt động hiệu quả trong băng tần trung, giảm năng lượng nhiễu ngoài băng thông.
- Bộ lọc duy trì năng lượng của tín hiệu cao trong băng tần cao.

- Gain ngoài băng tần thấp quá cao (1.046 > 0.2), gây rò rỉ tín hiệu vào ngoài băng thông.
- Độ lệch pha tuyến tính là 0.0733, lớn hơn giá trị mong muốn, gây biến dạng pha.
- Độ lệch pha đo được là 1.9908, vượt ngưỡng cho phép (0.1), gây biến dạng pha đáng kể.
- Năng lượng tín hiệu ngoài băng tần là 21098.59, quá cao so với mức mong muốn (0.1), không suy giảm đủ mạnh.
- Các bộ lọc đáp ứng đúng tần số cắt và giảm nhiễu ngoài băng.
- Giao diện người dùng hoạt động mượt mà và dễ điều chỉnh.
- Visualizer hiển thị chính xác sự thay đổi của tín hiệu âm thanh khi thay đổi độ lợi của từng băng tần.
- Thay đổi được độ lợi của âm thanh đầu ra trực tiếp không cần xuất file.
- Chất lượng âm thanh đầu ra khi nghe trực tiếp bị nhiễu và mất mát tín hiệu dẫn đến âm thanh phát trực tiếp hơi rè, nhưng khi lưu lại kết quả xử lý thì chất lượng âm thanh vẫn được đảm bảo.

b. Kết quả đạt được và chưa đạt được

Kết quả đạt được: Bộ lọc ba băng tần đạt yêu cầu về bảo toàn năng lượng trong băng tần, ổn định đầu ra, giữ nguyên độ dài tín hiệu và hiệu quả khi lọc nhiễu trắng và tín hiệu tần số cao.

Kết quả chưa đạt: Bộ lọc cần cải thiện ở các mục đáp ứng băng tần và suy giảm ngoài băng tần, đặc biệt ở băng tần thấp và ngoài băng thông. Đáp ứng pha tuyến tính và độ lệch pha cũng cần được tối ưu hóa để giảm thiểu biến dạng pha.

c. Biện luận kết quả

Bộ lọc FIR cho kết quả đáp ứng tốt trong các dải tần, đáp ứng mong muốn của người dùng trong điều chỉnh âm thanh bass, mid và treble. Tuy nhiên, việc sử dụng FIR có thể làm tăng độ trễ khi số lượng taps lớn.

Bộ lọc băng tần thấp cho thấy đáp ứng gain cao trong dải tần rất thấp (dưới 200 Hz), sau đó suy giảm nhanh khi tần số tăng lên. Đáp ứng của bộ lọc cho thấy khả năng lọc các tín hiệu tần số thấp tốt và loại bỏ hầu hết các tần số cao hơn. Tuy nhiên, độ suy giảm ngoài băng tần không hoàn toàn tối ưu, với một số tín hiệu ở tần số cao hơn có gain nhỏ nhưng vẫn tồn tại. Điều này có thể dẫn đến hiện tượng rò rỉ một lượng rất nhỏ tín hiệu ngoài băng thông.

Bộ lọc băng tần trung cho thấy đáp ứng tốt trong khoảng tần số từ 200 Hz đến khoảng 2000 Hz, sau đó suy giảm mạnh ở cả hai phía. Điều này cho thấy bộ lọc hoạt động tốt trong việc giữ lại các tín hiệu trong băng tần trung và loại bỏ các tín hiệu thấp hơn và cao hơn. Mặc dù đáp ứng trong băng tần trung là khá tốt, nhưng có hiện tượng dao động nhỏ ở biên độ gain trong vùng cận băng tần (ripples). Điều này có thể ảnh hưởng một chút đến chất lượng tín hiệu trong một số ứng dụng nhạy cảm.

Bộ lọc băng tần cao có đáp ứng tốt từ khoảng 2000 Hz đến gần 20000 Hz, với độ gain ổn định. Khi tần số giảm dưới 2000 Hz, độ suy giảm mạnh, cho thấy bộ lọc hoạt động hiệu quả trong việc giữ lại các tín hiệu tần số cao và loại bỏ các tín hiệu tần số thấp hơn. Bộ lọc này có khả năng suy giảm mạnh các tín hiệu ngoài băng tần thấp, nhưng không đạt độ sắc nét hoàn toàn, với một số tín hiệu ở ngoài băng tần có gain nhỏ nhưng vẫn tồn tại.

Đánh giá tổng thể về bộ lọc FIR với cửa sổ Hamming:

- **Ưu điểm**: Cửa sổ Hamming mang lại đáp ứng tần số khá tốt, với độ suy giảm ngoài băng tần hợp lý và đáp ứng pha tương đối tuyến tính, đặc biệt hữu ích trong việc hạn chế biến dạng pha cho tín hiệu trong băng tần.
- **Nhược điểm**: Tuy nhiên, cửa sổ Hamming không cung cấp suy giảm ngoài băng tần mạnh như một số cửa sổ khác (ví dụ, Blackman hoặc Kaiser). Điều này có thể gây ra hiện tượng rò rỉ tín hiệu ngoài băng thông, nhất là ở tần số cao. Hiện tượng dao động nhỏ ở các biên tần số của các bộ lọc (ripples) có thể ảnh hưởng đến độ chính xác trong một số ứng dụng yêu cầu cao.

8. Kết luận

Dự án đạt được mục tiêu thiết kế bộ cân bằng số hoạt động theo thời gian thực với các đặc tính lọc số tốt. Khả năng điều chỉnh từng băng tần cho phép người dùng tối ưu hóa chất lượng âm thanh. Trong tương lai, việc tối ưu hóa mã nguồn cũng như tính năng và bên cạnh đó là tích hợp với phần cứng sẽ được thực hiện để giảm tải cho máy tính và tăng tính ứng dụng thực tiễn.