2025년 1학기 데이터통신 6주차 과제 보고서

- MFSK 통신-

202001156 정보통계학과 김수영

과제 해결

영상링크

https://youtu.be/69BQeFrH6w0

송신기능

```
26  def sender(text):
27      string_hex=text.encode('utf-8').hex().upper()
28      print(f'Unicode: {string_hex}')
29
30      audio = []
31      for i in range(int(unit*samplerate*2)):
32          audio.append(int(INTMAX*math.sin(2*math.pi*rules['START']*i/samplerate)))
33      for s in string_hex:
34          for i in range(int(unit*samplerate*1)):
35              audio.append(int(INTMAX*math.sin(2*math.pi*rules[s]*i/samplerate)))
36      for i in range(int(unit*samplerate*2)):
37          audio.append(int(INTMAX*math.sin(2*math.pi*rules['END']*i/samplerate)))
38
39      p = pyaudio.PyAudio()
40
41      stream = p.open(format=pyaudio.paInt16,
42                      channels=channels,
43                      rate=samplerate,
44                      output=True)
45
46      chunk_size = 1024
47      for i in range(0,len(audio), chunk_size):
48          chunk = audio[i:i+chunk_size]
49          stream.write(struct.pack('<'+('h'*len(chunk)), *chunk))
```

송신함수이다. 입력받은 텍스트를 유니코드로 변환하여 해당 문자를 rule에 따라 적절한 주파수로 변환하여 pyaudio로 출력한다. 스피커의 지원 비트가 16bit이기에 paInt16으로 설정하였다.

morse2audio 함수의 경우 요구사항에 맞춰 잘 frequency와 rate, unit(코드에서 t로 정의된 것)을 정의하였으며 스피커와 마이크의 스펙 한계로 인해 pyaudio의 포맷은 paint16으로 설정하였다. 따라서 2바이트를 정보로 갖는 포맷의 특성상 INTMAX 역시 2^15 -1 인 32767로 설정하였다. 모스부호 송신 간격 역시 국제 규정에 맞게 설정하였다. 또한, 실습코드에서 제시하였듯이 START와 END를 2unit만큼 보내는 부분을 넣어 동기화 지점과 녹음 종료 지점을 구분할 수 있게 하였다.

수신 기능

```python
def receive():
    word = ''
    p = pyaudio.PyAudio()
    threshold = 510
    samplerate = 48000
    stream = p.open(format = pyaudio.paInt16,
                    channels = 1,
                    rate = 48000,
                    input = True
                    )

    unit_samples = int(0.1 * 48000)

    count = 0

    while True:
        data = stream.read(unit_samples,exception_on_overflow=False)
        samples = struct.unpack('<' + ('h' * unit_samples), data)

        freq = scipy.fftpack.fftfreq(len(samples), d=1/samplerate)
        fourier = scipy.fftpack.fft(samples)
        freq_max = freq[np.argmax(abs(fourier))]

        print(f'[Start]: {freq_max}')

        if freq_max == threshold:
            count += 1
            if count == 2:
                break
        else:
            count = 0

    while True:
        data = stream.read(unit_samples, exception_on_overflow=False)
        samples = struct.unpack('<' + ('h' * unit_samples), data)
        freq = scipy.fftpack.fftfreq(len(samples), d=1/samplerate)
        fourier = scipy.fftpack.fft(samples)
        freq_max = freq[np.argmax(abs(fourier))]
        print(f'[Data]: {freq_max}')
        spell = converter(freq_max)
        if spell:
            word += spell

        print(f'Current Data: {word}')

        if freq_max == 2940 or freq_max == 2950:
            count += 1
            print(f'[END]: {freq_max}')
            if count == 2:
                break
        else:
            count = 0

    stream.stop_stream()
    stream.close()
    p.terminate()

    text = uni.uni2str(word)
    print(f'Receive Message: {text}')
```

수신함수이다. 과제에서 요구한 목표를 어떻게 해결하는지 알아보자. 추가적으로 해당 코드에서 얻은 주파수는 10Hz 단위로 주파수의 값이 표현되기에 이를 고려하여 코드를 작성하였다.

목표2 - 동기화

신호의 시작점을 인식하는 부분이다. 스트림을 통해 계속 마이크 신호를 받다가 특정 스트림이 START의 주파수와 동일한 경우(=512, 따라서 Threshold == 510으로 설정) count를 늘려 이 count값이 2가 되면 시작 부분임을 인지하게 하였다. 이를 통해 혹여나 시작 주파수가 같은 510Hz 주파수가 소음으로 들어와도 연속적이지 않을 경우 이를 탐지하지 않을 수 있는 기능도 수행할 수 있다.

목표1 - 소음 처리

소음처리에 대한 부분이다. 소음처리는 간단하게 스트림의 주파수의 최댓값을 인식하는 형식으로 구현하였다. Rule에 의해 인코딩된 주파수 대부분이 일상 소음에서는 접하기 힘든 고음역대라 생각하여 해당 스트림에서의 주파수의 최댓값으로 처리하는 것이 합리적이라 판단하였다.

```
32    def converter(freq):
33        for key, value in rules.items():
34            if freq >= value-10 and freq <= value+10:
35                if key == 'END' or key == 'START':
36                    return ''
37                else:
38                    return key
```

추가적으로 receive에 사용된 converter란 함수는 주파수를 받으면 rule에 명시된 문자로 decoding하는 함수이다. 이때, fft로 얻은 주파수의 값이 10Hz단위이기에 실제 주파수와는 약간 다름으로 이를 커버하기 위해 ±10정도 범위를 주어 인식을 좀 더 쉽게 할 수 있도록 하였다. 이를 통해 음성신호 자체에 약간의 소음이 생겨도 이를 정상적으로 탐지할 수 있으며 혹여 rule에 위배되는 신호가 들어올 경우 이를 필터링할 수 있는 기능도 겸할 수 있다.

목표3 - 신호의 끝

신호의 끝을 정의하는 부분이다. END에 해당하는 신호가 두 번 연속으로 탐지될 경우 녹음을 종료할 수 있게 구현하였다.

목표를 전부 해결한 후 음성으로 입력받은 유니코드를 최종적으로 복호화하여 원래 텍스트를 얻어낸다.

결론

최종적으로 일반적인 텍스트를 유니코드로 변환하고 변환된 유니코드를 MFSC 방식으로 송신하여 마이크와 스피커를 통해 통신할 수 있는 코드를 작성하였으며 정상적으로 작동함을 영상을 통해 확인할 수 있다.

영상링크

https://youtu.be/69BQeFrH6w0