2025년 1학기 데이터통신 7주차 과제 보고서

-Reed Solomon-

202001156 정보통계학과 김수영

과제 해결

정상 wav 파일 생성 및 랜덤 에러 생성

```python
def make_audio(text):
    uni_data = text.encode('utf-8')
    # RS 인코딩
    rsc = reedsolo.RSCodec(8)    "reedsolo": Unknown word.
    encoded_bytes = rsc.encode(uni_data)
    encoded_hex = encoded_bytes.hex().upper()
    print(f'RS 인코딩된 16진수: {encoded_hex}')
    hex_length = len(encoded_hex)
    print(f'RS 인코딩된 16진수의 길이: {hex_length}')
    # 오디오 신호 생성
    audio = []
    for s in encoded_hex:
        for i in range(int(unit*samplerate)):    "samplerate": Unknown word.
            audio.append(int(INTMAX*math.sin(2*math.pi*rules[s]*i/samplerate)))    "samplerate": Unknown w
    audio2wav(audio, "normal.wav")

    # make error

    #error_count = random.randint(0, bytes_length/2) # 오류 개수, reedsolo에서는 n/2개의 오류를 복구할 수 있음
    error_count = random.randint(0, 8)
    print(f"주입할 오류 개수 (byte단위 오류): {error_count}")

    encoded_bytes_with_errors = bytearray(encoded_bytes)
    indices = random.sample(range(len(encoded_bytes_with_errors)), error_count)

    for idx in indices:
        original_byte = encoded_bytes_with_errors[idx]
        new_byte = random.randint(0, 255)
        while new_byte == original_byte:
            new_byte = random.randint(0, 255)
        encoded_bytes_with_errors[idx] = new_byte
    error_hex = encoded_bytes_with_errors.hex().upper()
    print(f"오류가 주입된 RS 데이터 16진수: {error_hex}")

    audio = []
    for s in error_hex:
        for i in range(int(unit*samplerate)):    "samplerate": Unknown word.
            audio.append(int(INTMAX*math.sin(2*math.pi*rules[s]*i/samplerate)))    "samplerate": Unknown w

    audio2wav(audio, "error.wav")

def audio2wav(audio,filename):
    with wave.open(filename, 'wb') as w:
        w.setnchannels(1)
        w.setsampwidth(4)
        w.setframerate(48000)
        for a in audio:
            w.writeframes(struct.pack('<l', a))
    print(f"{filename} 파일이 생성되었습니다.\n")
```

wav 파일을 만드는 함수이다. 입력 받은 텍스트를 변환한 후 오디오 신호를 생성하여 우선 정상적인 wav 파일을 생성한다. 이후 byte 단위로 오류를 랜덤하게 첨가하여 error wav 파일을 만든다. 이때, RSCodec은 8로 설정하였으며 디코딩이 정말 잘 되는지 추후 확인하기 위해 RS코드가 검출할 수 있는 양인 4개보다 높은 값도 샘플링되게 하기 위하여 random함수의 범위를 8까지 늘렸다.

오류 검출 및 디코딩

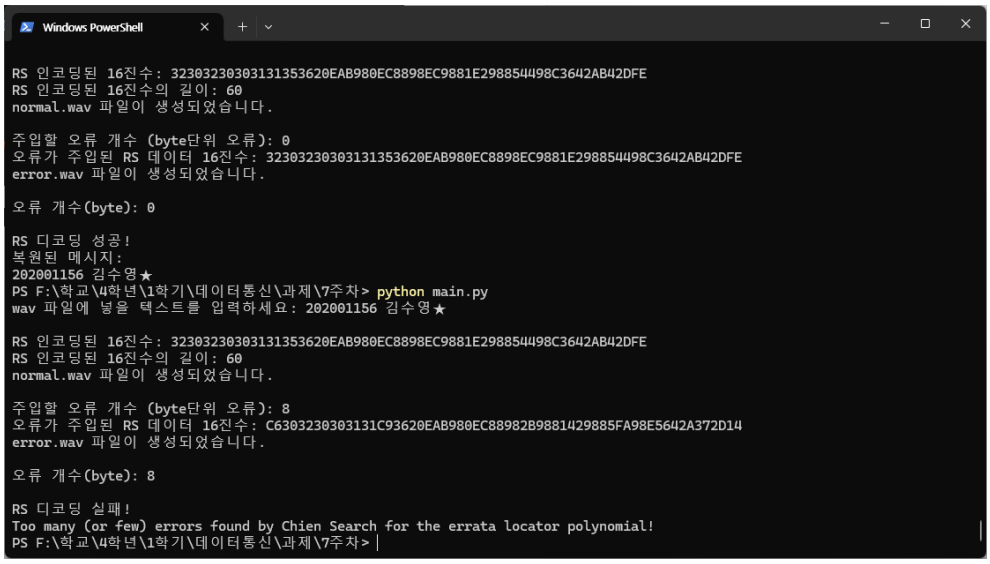
```python
def converter(freq):
    for key, value in rules.items():
        if freq >= value-10 and freq <= value+10:
                return key

def restore_error_wav(normal_wav, error_wav):
    # 정상 wav 파일
    with wave.open(normal_wav, 'rb') as w:
        frames = w.readframes(w.getnframes())
        normal_audio = np.array(struct.unpack('<' + 'i' * (len(frames)//4), frames), dtype=np.int32)

    # 오류 wav 파일
    with wave.open(error_wav, 'rb') as w:
        frames = w.readframes(w.getnframes())
        error_audio = np.array(struct.unpack('<' + 'i' * (len(frames)//4), frames), dtype=np.int32)

    # 원본의 주파수 분석
    samples_per_symbol = int(unit * samplerate)    "samplerate": Unknown word.
    num_symbols = len(normal_audio) // samples_per_symbol

    normal_hex_string = ""
    for i in range(num_symbols):
        samples = normal_audio[i*samples_per_symbol:(i+1)*samples_per_symbol]

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

        hex_char = converter(freq_max)
        normal_hex_string += hex_char

    # 오류의 주파수 분석
    error_sybol = len(error_audio) // samples_per_symbol    "sybol": Unknown word.
    error_hex_string = ""
    for i in range(error_sybol):    "sybol": Unknown word.
        samples = error_audio[i*samples_per_symbol:(i+1)*samples_per_symbol]

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

        hex_char = converter(freq_max)
        error_hex_string += hex_char

    # 바이트 단위 오류 개수 탐지
    normal_bytes = bytes.fromhex(normal_hex_string)
    error_bytes = bytes.fromhex(error_hex_string)

    error_count = 0
    min_len = min(len(normal_bytes), len(error_bytes))
    for i in range(min_len):
        if normal_bytes[i] != error_bytes[i]:
            error_count += 1

    print(f"오류 개수(byte): {error_count}")
```

이후 원본 파일과 오류 파일을 모두 입력받아 fft로 주파수를 분석하여 얻은 주파수들을 converter() 함수를 이용하여 원본 16진수 문자열로 mapping시킨다. 이때, 주파수의 규칙은 지난 6주차 MFSK 과제 때 수행했던 rule을 사용하였다. 이후, 바이트 단위로 오류를 첨가하였으므로 다시 바이트로 변환하여 두 바이트 문자열을 비교하여 오류의 개수를 출력한다.

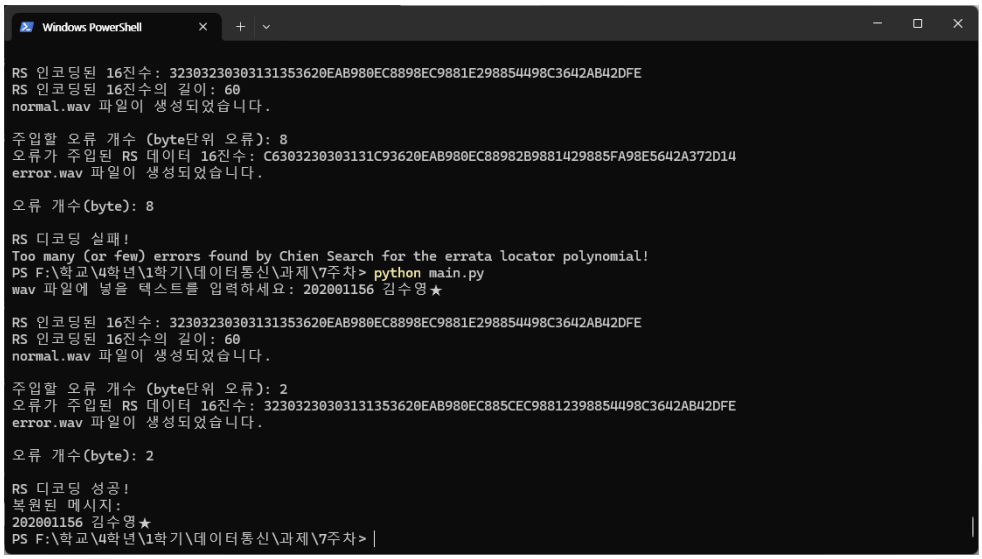
```python
    # RS 디코딩
    try:
        encoded_bytes_with_errors = bytes.fromhex(error_hex_string)
        rsc = reedsolo.RSCodec(8)    "reedsolo": Unknown word.
        decoded_bytes = rsc.decode(encoded_bytes_with_errors)[0]
        recovered_text = decoded_bytes.decode('utf-8')
        print("\nRS 디코딩 성공!")
        print("복원된 메시지:")
        print(recovered_text)
    except reedsolo.ReedSolomonError as e:    "reedsolo": Unknown wo
        print("\nRS 디코딩 실패!")
        print(e)
    except UnicodeDecodeError as e:
        print("\nRS 디코딩은 되었으나, UTF-8 디코딩에 실패했습니다.")
        print(e)
```

최종적으로 RS 디코딩을 실시하여 디코딩이 되는지 확인한다.

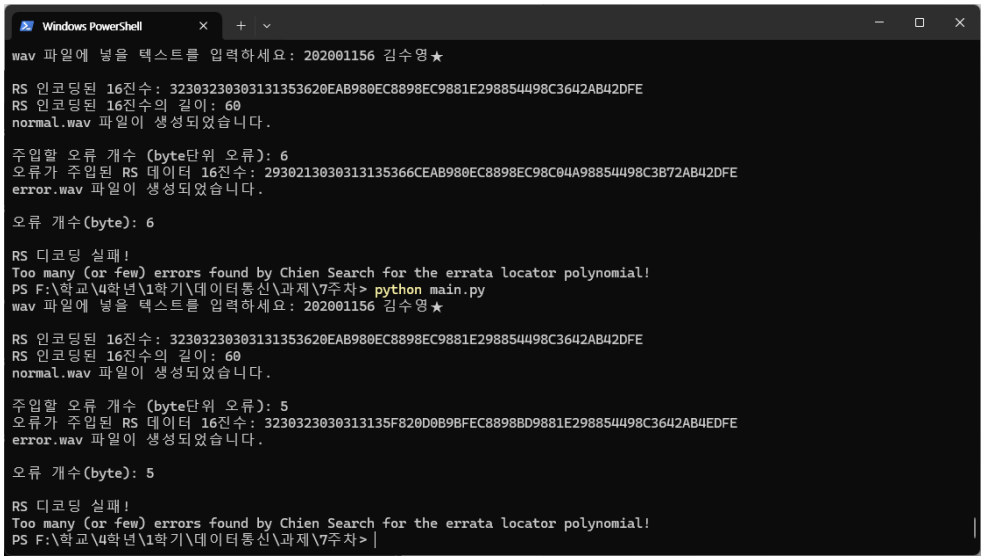
실험 결과

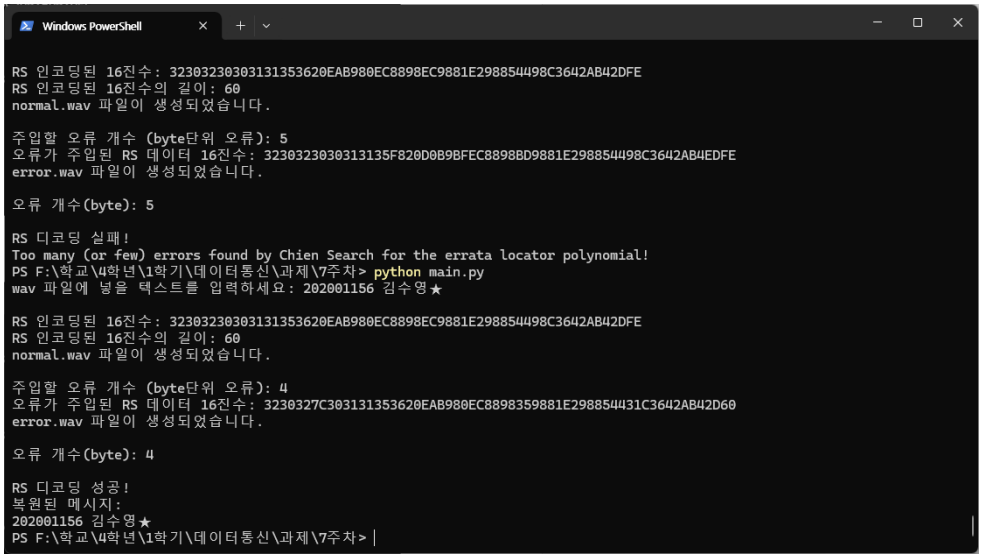


첫번째로, 오류가 8개 생성되었을 때에는 RS 디코딩이 처리할 수 있는 범위를 벗어났기에 디코딩이 실패함을 확인할 수 있다.



두번째로, RS 디코딩이 오류를 복구할 수 있는 수치인 2개의 오류는 제대로 디코딩이 됨을 확인할 수 있다.





마지막으로, RS코드가 최대 복구할 수 있는 오류인 4개까지는 제대로 디코딩을 수행함을 확인할 수 있으며 5개 이상부터는 디코딩이 안됨을 확인할 수 있다.

결론

최종적으로 오류가 있는 wav 파일과 원본 wav 파일을 비교하여 오류의 개수를 검출하고 RS Code를 이용하여 오류가 있는 wav 파일을 정상적으로 복구하는 작업을 수행하였다.

과제로 제출된 wav파일은 제일 마지막에 있는 오류의 개수가 4인 wav파일과 그 원본 파일이다.