데이터 통신 과제 보고서

학과	컴퓨터공학과
학번	201602011
이름	서성덕

1. 과제 목표

python을 이용해 문자열을 처리하고 소리로 재생하는 코드를 구현합니다.

① 주파수 출력

② 소리 재생

③ YouTube에 demo영상 업로드

2. 코드 설명 및 과제 해결

과제해결방법: 저는 과제를 해결할 때 기존에 구현한 decode.py를 참고하여 그 과정을 반대로 구현해 문자열을 소리로 변환했습니다.

① 필요한 모듈 Import

```
1 import numpy as np
2 import pyaudio
3 from reedsolo import RSCodec
```

- 자료를 처리할 때 사용하는 numpy를 import합니다.
- 소리를 송출할 때 사용하는 pyaudio를 import합니다.
- FEC변환을 위해 사용하는 reedsolo를 import합니다.

② 필요한 Global변수 선언

```
5 HANDSHAKE_START_HZ = 8192        # select start hz
6 HANDSHAKE_END_HZ = 8192 + 512    # select start hz (higher than start hz)
7
8 START_HZ = 1024
9 STEP_HZ = 256
10 BITS = 4
```

첫 신호음인 HANDSHAKE_START_HZ는 8192로 설정합니다.
마지막 신호음인 HANDSHAKE_END_HZ는 8192 + 512로 설정합니다.
START_HZ / STEP_HZ / BITS / FEC_BYTES는
apk파일에서 확인하여 적절한 값을 설정해줍니다.

```
14 start_flag = False
15 end_flag = False
```

데이터를 주파수로 변경할 때 첫 신호음과 마지막 신호음을 데이터 리스트에 추가했는지 확인하기 위한 Flag, start_flag와 end_flag를 false로 선언합니다.

③ if __name__ == '__main__'

```
79 if __name__ == '__main__':
80     input_msg = input("Input : ")
81     play_sound(input_msg)
```

input을 통해 문자열을 입력받고 입력받은 문자열을
play_sound()를 통해 소리를 출력합니다.

④ **play_sound(msg)**

```
71 def play_sound(msg):
72     byte_array = msg
73     rs = RSCodec(FEC_BYTES)
74     fec_payload = rs.encode(byte_array.encode())
75 #     print('bs:', fec_payload)
76     fec_payload = encode_byte_data(fec_payload)
77     sound_code(fec_payload)
```

입력받은 msg를 RSCodec을 이용해 byte_array로 인코딩합니다.
이 때 새로 만든 함수, encode_byte_data()함수를 이용해 비트를 분할합니다.

저는 현재는 주석처리 되어있는 print문으로
인코딩이 정확히 작동하는지 확인하는 과정을 가졌습니다.

그 후 인코딩된 fec_payload를 sound_code함수를 이용하여
소리 출력을 처리하도록 구성했습니다.

⑤ **encode_byte_data(byte_data)**

```
35 def encode_byte_data(byte_data):
36     data = list()
37     for i in range(len(byte_data)):
38         data.append(byte_data[i] >> 4)
39         data.append(byte_data[i] & 15)
40 #     print('bc:', data)
41     return data
```

전송을 위해 비트스트림(4비트)로 분할하는 함수입니다.

저는 현재는 주석처리 되어있는 print문으로
비트 분할이 정확히 작동하는지 확인하는 과정을 가졌습니다.

⑥ sound_code(fec_payload)

```
60 def sound_code(fec_payload):
61     p = pyaudio.PyAudio()
62     stream = p.open(format=pyaudio.paFloat32,
63                     channels=1,
64                     rate=44100,
65                     output = True)
66
67     ######insert your code below######
68     feq_data = data_to_feq_data(fec_payload)
69     sound_generate(stream, feq_data)
```

먼저 Pyaudio객체를 만들어줍니다.
주어진 양식에 맞게 pyaudio의 open()함수를 이용해 stream을 만들어줍니다.

매개변수로 받아온 fec_payload를 data_to_feq_data() 함수를 이용해
기존의 리스트를 주파수 값으로 변환하도록 구성합니다.

주파수 값으로 구성된 list, feq_data를 sound_generate() 함수를 이용해
소리를 출력하도록 구성합니다.

⑦ data_to_feq_data(fec_payload)

```
43 def data_to_feq_data(fec_payload):
44     global start_flag
45     global end_flag
46     data = list()
47     for i in range(len(fec_payload)):
48         freq = to_freq(fec_payload[i])
49         data.append(freq)
50         if freq == HANDSHAKE_START_HZ:
51             start_flag = True
52             data.append(to_freq(fec_payload[i]))
53         if i == len(fec_payload) - 1:
54             end_flag = True
55             data.append(to_freq(fec_payload[i]))
56
57     print('list:', data)
58     return data
```

이 함수에서 전역변수로 설정한 start_flag와 end_flag를 변경해주기 위해 globla로 변수를 선언해줍니다.
주파수로 변환한 값을 저장할 리스트, data를 선언합니다.

fec_payload의 모든 값에 대해 to_freq()함수를 이용해
주파수 값으로 변환하고 data 리스트에 추가해줍니다.

이 때 리스트의 첫 신호음과 마지막 신호음을 설정하기 위해
flag와 HANDSHAKE_START_HZ를 이용해 조건을 만들어줍니다.

과제의 ① 목표 주파수 출력을 위해 저장한 data, 리스트를 출력합니다.

⑧ to_freq(step)

```
27 def to_freq(step):
28     ######insert your code in function######
29     if not start_flag:
30         return HANDSHAKE_START_HZ
31     if end_flag:
32         return HANDSHAKE_END_HZ
33     return START_HZ + step * STEP_HZ
```

비트를 주파수로 변환하는 함수입니다.
start_flag가 false이면 HANDSHAKE_START_HZ를 반환합니다.
end_flag가 true이면 HANDSHAKE_END_HZ를 반환합니다.

⑨ sound_generate(stream, freq)

```
60 def sound_code(fec_payload):
61     p = pyaudio.PyAudio()
62     stream = p.open(format=pyaudio.paFloat32,
63                     channels=1,
64                     rate=44100,
65                     output = True)
66
67     ######insert your code below######
68     feq_data = data_to_feq_data(fec_payload)
69     sound_generate(stream, feq_data)
```

sound_code() 함수에서 주파수로 이루어진 리스트를
sound_generate() 함수를 이용해서 소리를 출력합니다.

```
17 def sound_generate(stream, freq):
18     ######insert your code in function######
19     for i in range(len(freq)):
20         samples = divide_by_tone(freq[i])
21         stream.write(samples)
```

매개변수로 받아온 주파수 값의 리스트에서 각각의 값을
divided_by_tone() 함수를 이용해 출력하기 위한 samples로 만들어줍니다.

만들어낸 samples를 pyaudio의 write함수를 이용해 소리로 출력합니다.

⑩ divide_by_tone(each_data)

```
23 def divide_by_tone(each_data):
24     ######insert your code in function######
25     return (np.sin(2 * np.pi * np.arange(44100 * 0.5) * each_data / 44100)).astype(np.float32)
```

주파수 값인 each_data를 sin함수의 여러 값을 가진 배열로 저장합니다.

이 때 np.arrange()안에 0.5를 곱해줘 소리의 길이를
기존 apk파일에서와 비슷하게 만들어주었습니다.

3. YouTube 링크

Demo영상 업로드

[과제]
soundEncode.py를 실행해 packet을 입력
입력받은 문자열을 처리해 소리로 출력

Link: https://youtu.be/mrvnS4-M7gg

. 느낀 점 및 하고 싶은 말

virtual-box / ubuntu 16.04 / python3 환경에서 소리 출력에 문제가 있어
window10 / pycharm - python3.7 환경에서 영상을 촬영했습니다.

기본 코딩은 virtual-box / ubuntu 16.04 / vim에서 작성했음을 알려드립니다.