Signal processing: Exercise 2

Description of my application

My application reads a wav file, encodes its data using rice encoding and stores the encoded data to a binary file. Then it reads that file, decodes it and stores the decoded data as a wav file. Both the original file and the decoded one are able to play using a wav player as I show in the notebook that both sound the same.
The process I follow to encode the original file is:

1) Read the wav file as a binary file and stores all its bytes in a list of bytes using Bytearray object from python
2) Determine what's the longest possible encoded value, to do this I perform rice coding to all values between 0 and 255(all possible integers we can store per byte) and select the length of longest value
3) I iterate over all bytes in the Bytearray list and for each I encode it using rice encoding and pad it so that all encoded values have the same length.
4) I store all the bytes in output file
5) To decode the file I read it, store all its bytes in a bytearray
6) Then I iterate over every byte, I group the bytes in groups of n_bytes(value discovered in step 2), every group of n_bytes bytes from the encoded file is sent to a function that decodes it using rice decoding and translate the group into a decoded byte
7) Store all decoded bytes as a wav file.

Analysis of results

	Original size	Rice(K=4)	Rice(K=2)	% compression (K=4)	% compression (K=2)
Sound1.wav	1002.088 Kb	3006.264 Kb	9018.792 Kb	0.33	0.11
Sound2.wav	1008.044 Kb	3024.132 Kb	9072396 Kb	0.33	0.11

Using rice coding over the raw data, ie: without analyzing it as an audio file is very inefficient. For K=4 the encoded files become 3x larger in size and for K=2 they become 9X larger.The reason is because to store every encoded byte in the output file they must all take the same number of bytes to delimit the values. The length of every encoded byte must therefore be padded to be the same number of bytes as the longest possible encoded value. Because we are encoding raw bytes, the values we encode are in the range 0-255, 255 takes 3 bytes for K=4 and 9 bytes for K=2.

Further development

In search of improving the compression rate I experimented with different K values for encoding the wav files and found that using a value of K between 7 and 15 increases compression rates to 0.5, since the encoded version of 255 can be encoded in 2 bytes rather than 3 when using K=4.