Decoder Design:

The decoder is designed to take in the transmitted Wifi signal and recover the original transmitted message along with the length and an integer representing the length of the zero padding at the start of the received signal. The decoder operates in multiple levels depending on the complexity and processing steps. These stages includes synchronization, demodulation, deinterleaving, convolutional decoding, and holistic reconstruction of the message.

1. Synchronization
    a. We use the same 128 bit preamble as in the Transmitter and modulate it with the 4-QAM modulation from the commpy package also used in the Transmitter. From there, we transform this into the time domain through an inverse Fourier transform which gives us OFDM symbols. We find the correlation array between the received signal and the preamble from the Transmitter, taking the max value to find the start of the data. Thus, the decoder can determine the length of the zero padding by taking the start index of the actual data as anything before will be zero padding.
2. Chunking
    a. We then divide the relevant portions of the signal into chunks of 64 as defined by nfft. We apply a Fourier transform in order to convert back to the frequency domain. We remove the preamble symbols to achieve the symbols from the actual message.
3. Demodulation
    a. We demodulate the symbols in the frequency domain with 4-QAM demodulation. The symbols are mapped back to bits to produce the bitstream. The decoder then extracts the length field by calling the helper function undo_length_encoding which decodes a length value that was encoded in the first 2*nfft (128) bits. By then applying majority voting to each group of 3 bits, we recover the original length bits
4. Deinterleaving
    a. As we now know the length, we proceed to deinterleaving. The decoder reverses the interleaving process using the inverse permutation, which I extract to a helper method undo_interleave.
5. Convolutional Decoding
    a. The length is decoded and we take the output without the length and pass it into our custom Viterbi decoder. Without access to the library decoder, we write our own which initializes path metrics, processes received bits, calculates metrics with hamming distance or euclidean distance, selects the most probable path, reconstructs and outputs the transmitted bits.
6. Reconstruction
    a. The decoder reconstructs the original message, trimming the decoded bits to ensure the length is the same as the original message. After converting the bits to bytes and the bytes to characters, we output the recovered message.

Inference of Packet Length:

The Transmitter encodes the message length by converting to binary and repeating each bit three times. At the receiver, the length is decoded by majority voting on each group of 3 bits to correct single bit errors. From there, the binary sequence is converted back to an int which corresponds to the message length. We use this to extract the correct number of bits for the message reconstruction. This is done in my undo_length_encoding function.