Stage	Transmitter	Receiver
1	The message is converted into a set of bits through dec2bin. The reason why it is done 2 times and in reverse is something that I am unable to understand. These are 2 explanations given to me: 1. Professor Swarun explained that this is due to some limitation seen with dec2bin which was posing problems during the code. Which is why reshaping was done 2 times to get the code to conform to what was required for later stages. 2. Adishree told me that this was done because of the Endian difference being in the case of Networks and Systems. Once this is done, we append zeros to the message to make it a multiple of 64 so that the Trellis Encoder can work. This now passes through turboenc (which basically does trellis), to get a encoded message. It is then prepended with the length of the message as a 64-bit byte stream.	First from the output of stage 2, we remove the prepended length by recalculating output from 65^{th} bit to end stream. To this, Viterbi Decoding is carried out where in the tb value is kept to be 1 (again trial and error). This will give a bytestream consisting of 0s and 1s. Here we find out the first 1 and last 1 and calculate output as stream from first 1 to last1. It might be possible that our data would end with 0. Due to our previous approach this might have been removed. So, what we do now is that we pad 0s to the stream to make it a multiple of 8. Reshape is now done to counter the dec2bin shaping at the receiver. The reshaped output is converted into a string and then manipulations are done to get the string back.
2	The output from stage 1 is taken and is first split into 64-bit chunks. Each 64-bit chunk is first reshaped into 4 rows, n column vectors and further that is reshaped into n row, 1 column vector.	The output from stage 3 is taken and the complete reverse of the reshape that's done in the transmitter side is carried out in the receiver side. Once this has been done, the signal is again rounded so as to prevent any decimal values from BPSK demodulation ((output + 1)/2).
3	Here BPSK modulation is done where in the output message is made just less than the twice of itself. The preamble (64-bit identifier of any WiFi packet) is appended to the output.	To counter this stage, we first find out the length of the preamble and recalculate the output from the length(preamble)+1 till output. With this, we counter BPSK by adding 1 to the Output and halving (just opposite to what was done in the modulation stage).
4	The output is broken down into chunks of 64 bits and each chunk has its FFT found out.	The output here is divided into 64-bit chunks and the IFFT of each chunk is found out. Since the IFFT may/may not return discrete values, we round off all the values to the nearest 0s, thereby getting values either 0 or 1. If we get a 0 in the output, we need to make sure that the 0 is replaced by a -1, which is what is the next step done.

5	Output from Step 4 is taken and padded at the beginning as well as the end with random number of bits between 1 and 1000. After this, AWGN noise is added to the output depending on the SNR which is passed as argument to the Function.	The noise plus signal is passed to the receiver. Now with this, find the peaks of the signal (details explained below) from the signal. After finding the peaks, the first and the last peak greater than the threshold of 6 (calculated through trial and error) is extracted. The output between these 2 positions (first peak pos - 1, second peak pos) of the peak give the size of data + preamble. This data is now supposed to be made a multiple of 64 bits. For this, mod the length of the signal with 64 and pad 64-bitmod to the output. This is passed to the Stage 4.

Find Peaks of Signal:

1. Take the absolute value of the complex signal.
2. Use findpeaks function to find the location and the value of the peaks.
3. Check if each peak is greater than threshold and rewrite the vector to store only those values. Rewrite even the location vector to store the exact location of each of these points which cross the threshold.
4. Take the values between the 2 peaks. That will be the data + noise stream.

How to find length of packet:

Now in this I am confused among 2 things. Length of packet and length of message.

Length of Packet: (Presuming that it is the packet which is noise + signal stream)

1. Take the absolute value of the complex signal.
2. Use findpeaks function to find the location and the value of the peaks.
3. Check if each peak is greater than threshold and rewrite the vector to store only those values. Rewrite even the location vector to store the exact location of each of these points which cross the threshold.
4. Take the values between the 2 peaks. That will be the data + noise stream.

Length of Data:

This can be obtained by seeing the first 64 bits of the output that is fed into Stage 1 of the Receiver code.

This can also be obtained by finding the length of the output after all the stages have been complete (end of stage 1).