Team 07: Rayan Daod, Jean Chambras, and Ruben Ganansia

Transmission over a continuous-time channel with impairments

Introduction

Nowadays, no transmission by physical medium happens without impairments. Delay, noise, channel filtering, etc, need to be treated by good communication system designs. This project illustrates this situation and gives us a taste of the real world challenges in digital communications.

Code constellation:
We used **4-QAM** as a code constellation with the help of **Gray codes** to reduce the bit-error-probability. This allowed us to encode our message by groups of two bits.

Pulse characteristics:
For a pulse, we chose the **root-raised-cosine** because it satisfies the Nyquist criterion and is more interesting in communication designs (fast decay in time, and a box shape in frequency).
Thus, we implemented our own and chose the parameter T (symbol period) such that the pulse has the desired bandwidth (2000 Hz here).

Transmitter

Encoder:
We first convert the message into bytes, and **remove the first 0 in each byte** as it is useless in ASCII (we then have 160 bits less to send for a 160 characters message). After that, we group all the bits together, separate them into three arrays, and construct a parity check array. We then group the bits by groups of two in each array (because we used **4-QAM**, see above), map the grouped bits to integers which we interpret as indices of our code constellation, and map those indices to the corresponding symbols.
Finally, we used **Barker sequences** to synchronise our signal, and stick the corresponding symbols to the four symbol sequences, at the beginning, and the same symbols but flipped at the end (to detect the end of our signal).

Waveform former:
We first upsample each symbols array, filter the four obtained arrays with the pulse, modulate each of the samples array to the desired frequency, add those four arrays together, and scale the result to be an array of samples of maximum absolute value 0.85 (since the server clips the entering samples, and that noise is added above).

Receiver

N-tuple former:
After receiving the filtered samples, we deduce the removed frequency range, demodulate the samples and low-pass them with our pulse for each of the three available frequency ranges. We then correlate each of our signals with the stored preamble samples to find the delay with a maximum likelihood approach, and do the same with the flipped preamble samples to find the end of the data samples. We finally downsample the signal and obtain the received symbols.

Decoder:
We map the received symbols to the closest ones in the code constellation (**4-QAM**), and do the inverse process of the encoder except that we might have to reconstruct one array of bits with the **parity check** array if one of the three first frequency ranges is removed by the server.

Conclusion

*With this communication design, we use the allowed bandwidth at its full potential (no redundancy as each frequency range carries different information) and end up **sending only 2784 samples for a text of 160 characters**, which corresponds to **17.4 samples per character**.*

Thank you to the PDC team for this very interesting project!