# Implementation of Baseband Communication

To help achieve the project goal of developing a baseband communication visualisation, I split the baseband communication process into six key stages. Those stages are:

* Binary signal generation
* Encoding the binary signal
* Sampling and performing a DFT on the encoded signal
* Subjecting the frequency content to a low-pass filter
* Performing an IDFT on the filtered frequency content
* Demodulating and decoding the signal

Of these six stages the first four occur before the transmission of the signal and the last two occur at the receiver. Each of these stages required displaying a graph to the student; these graphs are vertically aligned and are ordered to indicate the progression of a real-world baseband communication system.

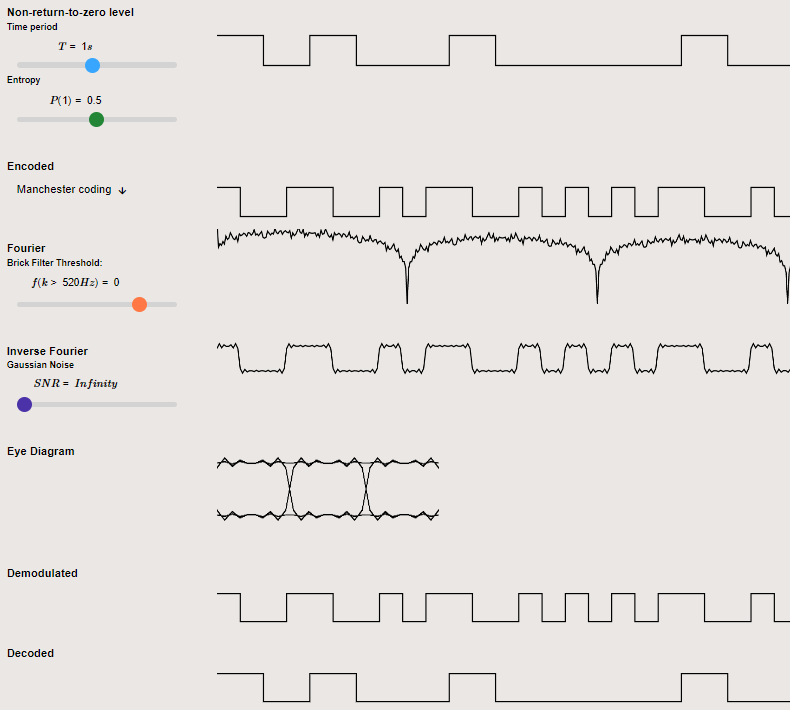


Figure – Screenshot of the visualisation displaying the key stages of Baseband Communication.

## Binary Signal Generator

The first of the key stages was the binary signal generator. It can generate a large, random, sample of data to be coded and transformed. This data is then clearly displayed to the user using non-return-to-zero level encoding.

This is the initial value of the data that will be transformed as it progresses through the baseband communication protocol. The data can be modified by the user in three ways:

* The user can modify the time period of the bits. This alters the bit rate and thus the energy per bit to noise power spectral density ratio of the signal. This is a useful metric for comparing the bit error rate of different encoding schemes.
* The user can manually set the probability of a 1-bit being generated by altering the ‘entropy’ slider between to .  
  This alters the entropy of the bit steam between 0 and 1 according to Shannon’s entropy equation [X].
* This can demonstrate to students how different encoding schemes can handle edge cases, such as a large sequence of binary zeros being transmitted. If the encoding scheme does not vary the signal then the receiver may be unable to maintain a consistent clock frequency to sample the signal correctly.
* The user can invert an individual bit by clicking on the bit position on the graph. This allows the user to manually set a bit pattern so they can see how any specific sequence will be encoded.

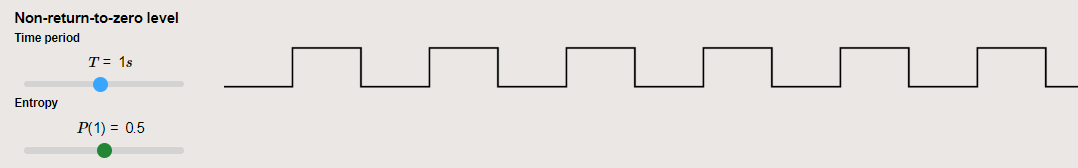


Figure – Screenshot of the binary signal generator.   
Generating a signal with T=1s and P (1) = 0.5.

## Binary Line Encoder

The second stage is a binary line coder, which encodes the data generated by the binary signal generator using several distinct line coding techniques. When deciding which line coding techniques to include I wanted to ensure I was selecting those most useful to the students who were going to use the visualiser. This meant they had to be distinct, each introducing new concepts. They should be used in the real world whilst being easy to understand for students who had previously never been introduced to the concept of line coding. Because of this, I implemented the following six line coding techniques:

* Non-return-to-zero level. For basic display of the generated signal.
* Non-return-to-zero mark. This introduces the concept that data may not just be represented by a single voltage level but may be represented with a bit transition.
* Return to zero. This introduces the concept of return to zero coding to the students, showing that the data does not need to remain at a single level for the entire bit period.
* Biphase-L. Commonly referred to as Manchester Coding. I implemented the line coding technique defined by IEEE 802.3[1], which is implemented in some wired Ethernet standards. This technique can be self-clocked due to each bit pattern generating a bit transition.
* Bipolar, Duobinary signal [2]. This concept can introduce the advantages of a line coding signal having little or no DC component to the students. This is useful to help avoid electro-static discharge.
* Multi-Level Transmit 3 [3]. This introduces the concept of encoding a signal with three voltage levels. Due to the three voltage levels, it requires less bandwidth and emits less electromagnetic interference. Due to its symmetrical nature, it can be connected to a twisted pair of cables and regardless of which way the cables are connected the same signal can be received. This makes it a technique which is used in real-world applications, such as in the FDDI TP-PMD standard that is used in 100BASE-TX as a fast ethernet standard [4].

The student can select the line coding scheme they wish to use so they can see how the encoding scheme changes the frequency content of the wave. These encoding schemes may also be impacted less by noise, or have a greater bit error rate when subject to a low-pass filter. The student will be able to observe these effects further down the visualisation.

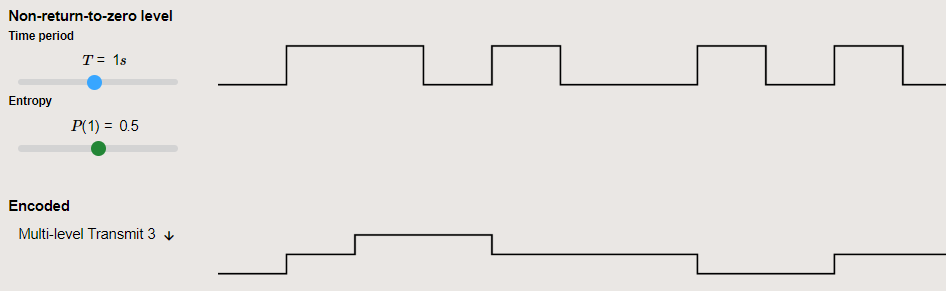


Figure – Screenshot of the visualisation showing a binary signal being  
encoded with MLT-3 Line Coding.

## Discrete Fourier Transform

This key stage passes the encoded signal through a DFT algorithm and displays the frequency content of the student's selected encoding scheme as a spectrogram.

The visualisation first performs a DFT on the student's encoded data. The data from this DFT is then passed down to the subsequent stages. However, to display a more useful depiction of the frequency content of the selected encoding scheme to the student the DFT will continue to process more sets of data. The program then averages the output of all iterations to display a more generalised depiction of the frequency content to the student.

The code for both the DFT and the above implementation can be found in the DFT section of the appendix.



Figure – Screenshot of the visualisation displaying a Spectrogram of a binary signal.

## Low-Pass Filter

When trying to load a cable with a signal that changes instantaneously from zero volts to a higher voltage, extremely high frequencies will be produced. These frequencies can cause undesirable capacitive coupling and crosstalk with other nearby cables, so these frequencies must be filtered out before transmission.

This can be achieved by utilising a low-pass filter. In the visualisation, I utilise a brick-wall filter with a frequency cut-off that can be defined by the student. This is applied to the frequency domain signal returned from the DFT algorithm. In a real-world system, this implementation would not be feasible. Rather, in a real-world analogue system, an LC T-Type filtering circuit could be used to introduce poles to the frequency response, lessening the frequency content after the break frequency [5]. Alternatively, in a digital real-world system, a FIR Filter [6] could be used.

This is an area of the project that, with time, I would like to improve, by implementing more realistic digital filtering techniques.



Figure - Screenshot of the visualisation showing a Spectrogram of a binary signal subject to a brick wall, low-pass, filter with f (k>100Hz) = 0

## Performing an IDFT

Now that the signal has been subjected to a low-pass filter the frequency content is processed using an IDFT. This will then display the filtered version of the signal to the student in the time domain. This is important as it allows the student to see the impact that the low-pass filter has had on the original signal and allows them to understand how the signal will be loaded onto the cable for transmission.

Below are two examples of applying a brick wall low-pass filter to a set of data then displaying the filtered result in the time domain. The first example has a frequency cut off at 300Hz and the second has a frequency cut off of 100Hz.

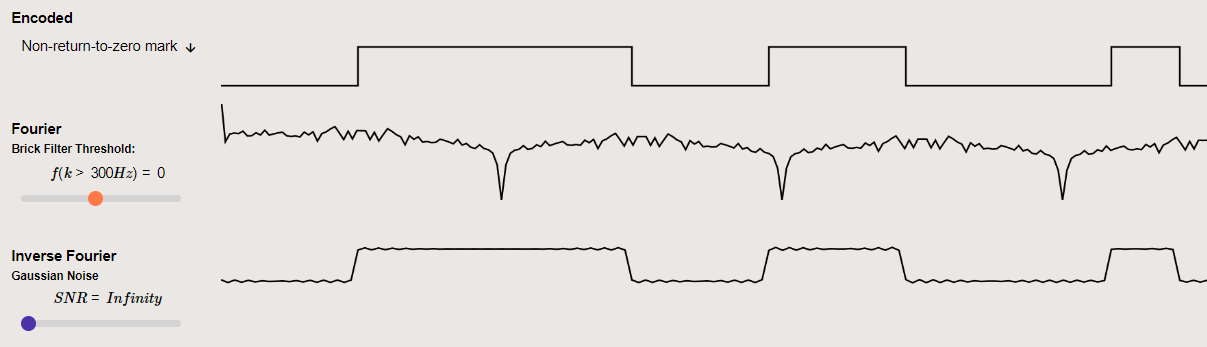


Figure – Screenshot of the visualisation showing the time response of   
a filtered signal with f(k>300Hz) = 0.

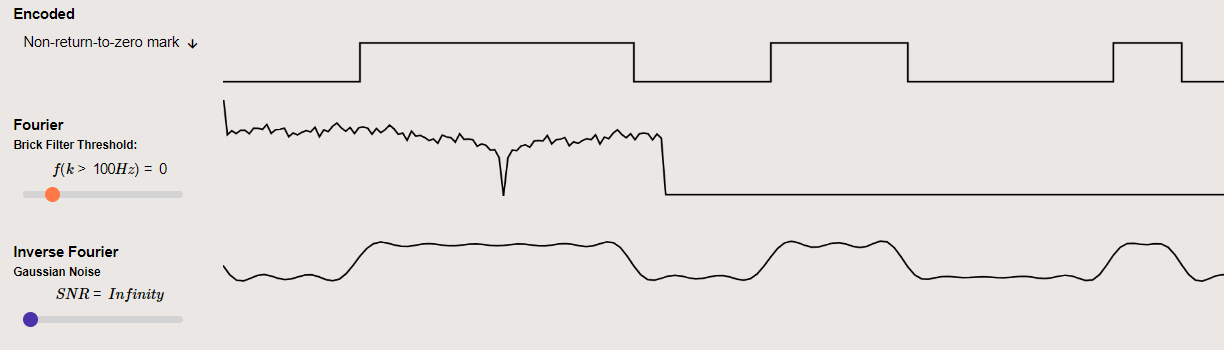


Figure – Screenshot of the visualisation showing the time response of   
the same filtered signal with f(k>=100Hz) = 0.

## Demodulating and decoding the signal

At this stage in a real-worlgd baseband communication system, the filtered signal will be transmitted to the receiver.

It is now the job of the receiver to interpret the incoming signal and attempt to convert it back to the original data stream, as generated by the binary signal generator.

I have implemented the demodulator by sampling the signal at regular intervals determined by the bit period. In a real-world system, the bit period may be standardised or have to be interpreted by the receiver. The current version of the simulation knows the bit period rather than attempting to calculate it. However, this is something that could be implemented with future versions of the visualisation.

Once the signal has been sampled it is then decoded. The demodulated signal may be unable to be decoded correctly due to errors introduced by filtering, or by noise. If this occurs the graph displays this as an error to the user.

If the decoded signal does not match that of the original signal it will be displayed with a red line. This indicates to the student that there was an error introduced when transmitting the data. This is then used to calculate the bit error rate of the system.

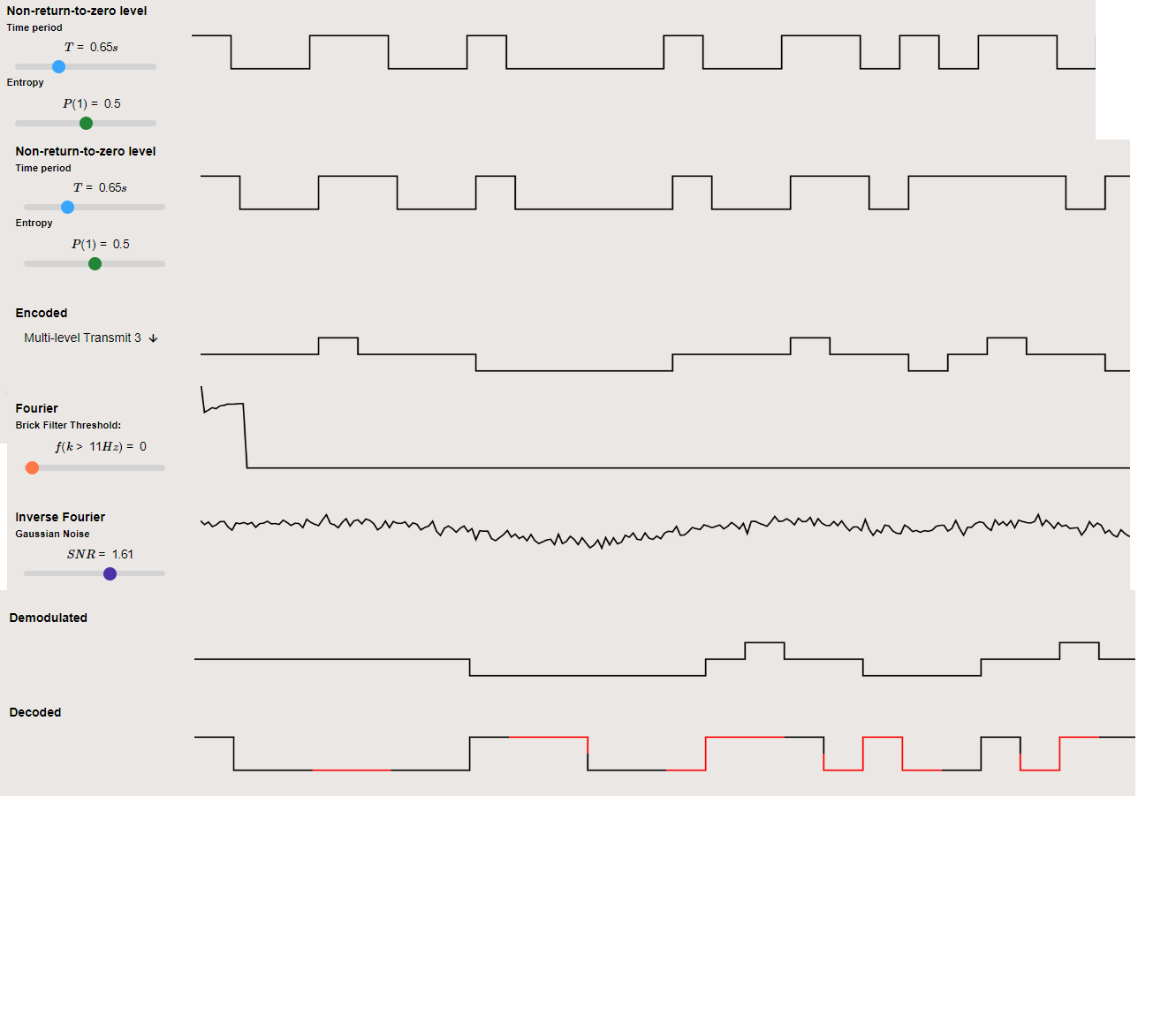


Figure – Screenshot of the visualisation showing a signal being encoded with MLT-3 line coding, filtered, then demodulated and decoded.

## Additional Features

To help demonstrate additional concepts related to baseband communication I have added two further features to the baseband visualisation. A Gaussian noise generator and an eye diagram.

### Gaussian Noise Generator

During transmission in a real-world system, signals will be subjected to noise and interference. This can introduce errors when attempting to demodulate the received signal. To help students understand how different line coding techniques are more or less susceptible to Gaussian noise, when allocated equal power, I added a slider that introduces a variable amount of Gaussian noise to the signal before the signal is demodulated.

### Eye Diagram

To further demonstrate both the effects of noise and the filtering of a signal I implemented an eye diagram. This diagram overlays the filtered signal of all possible sequences of binary ones and zeroes. This allows students to visually compare different line coding schemes by seeing if the receiver would be able to sample the signal at a regular period and receive the correct interpretation of the value of the signal.

If there is not a distinct “eye”, or gap, between the high and low signal values then there is too much intersymbol interference, or noise, present on the signal for a receiver to demodulate the signal correctly at a regular sampling interval; this would result in errors being introduced when demodulating the signal.

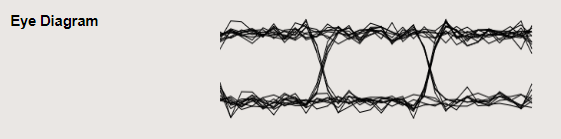


Figure – Eye diagram displaying a signal which is able to be demodulated at regular time intervals without error.

The above eye diagram indicates that it is possible for a receiver to sample and demodulate the signal at regular intervals without introducing errors.

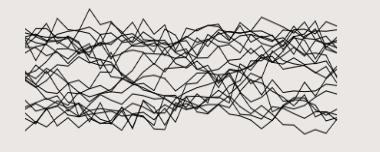


Figure – Eye diagram displaying a signal which is unable to be demodulated at regular time intervals without error.

This eye diagram does not have a clear “eye” between the high and low signal values. This indicates that there is too much noise and intersymbol interference present on the signal thus the receiver will be unable to demodulate the signal without error

# References

[1] *IEEE Standard for Ethernet,* IEEE Standard 802.3-2018

[2] Wikipedia contributors. *"Bipolar encoding"*.  Wikipedia, The Free Encyclopaedia. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Bipolar_encoding&oldid=1113660078> [Accessed: 29 April 2023].

[3] Wikipedia contributors. "MLT-3 encoding". Wikipedia, The Free Encyclopaedia. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Line\_code&oldid=1131753198 [Accessed: 29 April 2023].

[4] *IEEE Standards for Local and Metropolitan Area Networks: Supplement - Media Access Control (MAC) Parameters, Physical Layer, Medium Attachment Units, and Repeater for 100Mb/s Operation, Type 100BASE-T*,IEEE Standard 802.3, 1995.

[5] D. Pearce. “A Short Introduction to First-Order Responses”. University of York. [Online]. Available: <https://wiki.york.ac.uk/display/EE/Short+Introductions?preview=/206308218/251169152/24_Short_Intro_to_First_Order_Responses.pdf> [Accessed: 30 April 2023].

[6] D. Halliday. “FIR filter design”. University of York [Online]. Available: <https://wiki.york.ac.uk/download/attachments/198216152/DSP_OHP_Lect10.pdf?version=1&modificationDate=1676897975000&api=v2> [Accessed: 28 April 2023].