# Introduction

Teaching visualisations are a key tool for helping students to understand complex systems, they provide a way for students to not just to see how concepts they have been taught work but to interact with the simulation and understand how parameters effect the outcome and in doing so gain a deeper understanding of the concepts and why it may be implemented in such a way in the real world.

Dave Pearce has published over thirty Silverlight based visualisations helping students understand concepts including Basic Electronics, Communication Physical Layers and Communication Protocols and in doing so was awarded the [Insert full award name here]. This demonstrates the importance of these visualisations to students learning. However, as of 2019 Microsoft Silverlight, the application framework used to write and run these web applications has been discontinued, the software was only previous supported on Internet Explorer and as of June 2022 Internet Explorer has reached its end of life meaning these visualisations are no longer able to run on modern hardware making it difficult for students to be able to access this important learning resource.

Because of this it is important that these visualisations can be updated to be able to be ran on modern hardware that today’s students are able to access not just on university resources but also on their own devices whether that be a laptop or mobile phone.

# Overview of background reading

# Aim of the project

The aim of the project is to determine whether WebAssembly is an efficient tool for processing large data sets when developing teaching visualisations. It is important that teaching visualisations, particularly those which need to handle large data sets

# Objectives of the project

In order to achieve the aim of the project I have requirements that must be met

In order to achieve the aim of the project I have some must have requirements, these are essential to have in order to be able to thoroughly test my hypothesis.

## Must have requirements

These requirements must be met in order to adequately create a teaching visualisation tool that is able to determine whether WebAssembly is an efficient tool for processing large data when developing web-based teaching visualisations. I will list the requirements in the order that they would be applied in when implementing a Baseband Communication protocol

## Binary signal generator

The first of which is a binary signal generator able to generate a large, random, sample of data to be coded and transformed. This data must be able to clearly displayed to the user so they are able to follow the process of the data as it passes through the baseband visualiser. The user should be able to click on the signal in order to invert an individual bit. Doing this will allow the user to understand how each bit affects the final signal and how one-bit alteration could have large effects on the final output signal.

### Line coding

The second requirement is a line coder which must be able to encode the data generated by the binary signal generator using at least five distinct line coding techniques, it is important for the student to be able to select the line coding scheme they wish to use so they are able to see how the encoding scheme changes the output frequencies.

The must have line coding techniques are:

* Non-return-to-zero level

Where a binary zero is represented by zero volts and a binary one is represented by a high voltage.

* Non-return-to-zero mark

Where a binary zero is represented by maintaining the current voltage level and a binary one is represented by a transition from the current voltage level to either a high voltage or zero voltage.

* Return to zero

Where a binary zero is represented by zero volts for the entire bit period and a binary one is represented by half the bit period being high voltage before returning to zero volts for the remainder of the bit period.

* Biphase-L

Commonly referred to as Manchester Coding, I intend on implementing the line coding technique defined by IEEE 802.3[1], which is implemented in their wired Ethernet standards. This technique is commonly used as regardless of which symbol is generated there is always a bit transition, this means the signal is self- clocking.

A binary one is represented by the first half of the bit period being zero volts and the subsequent half of the bit period represented by a high voltage. A binary zero is the inverse, with the first half of the bit period represented by a high voltage and the second half of the bit period represented by zero volts.

* Bipolar, Duobinary signal [2]

Where a binary zero is represented by zero volts for the entire bit period and a binary one forces a transition to either a high voltage for half a bit period or a negative voltage for half a bit period, before returning to zero. Whether the voltage goes high or negative for the first half of the bit period alternates each time a binary one is encountered.

### Discrete Fourier Transformation

The third requirement is to be able to pass this encoded signal through a Discrete Fourier Transform (DFT) algorithm. This algorithm must be implemented in both JavaScript and WebAssembly and the user should be given the option to select which method to use. There should be an output of the time taken to compute the algorithm; this is to allow the user to visually understand how much more or less efficiently the data can be processed with these differing approaches.

### Low- pass filter

A low- pass filter must be applied to the Frequency domain signal returned from the DFT algorithm. This is essential in order to cut off the extremely high frequencies that would be generated when trying to load a cable with a signal that changes instantaneously from zero volts to a higher voltage. These extremely high frequencies could cause undesirable coupling and crosstalk with other nearby cables so it is important that these frequencies are filtered out before transmission.

### Inverse Discrete Fourier transform

The final must have requirement is to use the Inverse Discrete Fourier Transform algorithm to transform the filtered DFT signal back into 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.

## Nice to have requirements

Additionally, I will follow up these must have requirements with objectives that would be beneficial to include if time permits, in order to extend the scope and learning objectives for the project.

### Eye Diagram

Firstly, if time permits, I would like to present the user with an eye diagram overlaying the filtered signal for a binary one and a binary zero. This would allow the student to be able to visually compare different line coding schemes by visually understanding where the receiver would need to sample the signal to receive the correct interpretation of the signals value. For example, in the diagram below, the receiver could read the signal at half of the time period and correctly interpret the desired signal value. However, if the receiver were to read the signal at one quarter of the time period then the receiver would not always interpret the correct desired signal value.

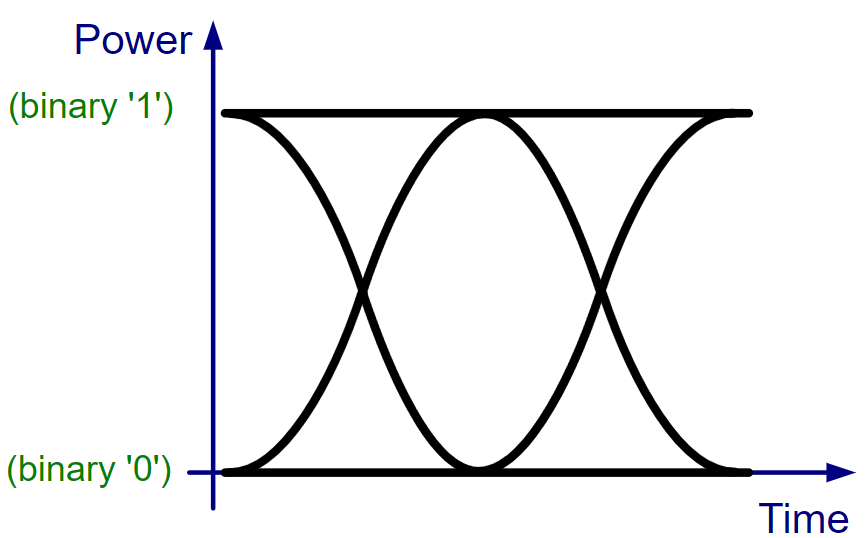


Figure 1 - Example of an eye diagram   
(Taken from <https://en.wikipedia.org/wiki/File:On-off_keying_eye_diagram.svg>)

### Noise

I would like to be able to allow the user to add variable amounts of gaussian noise to the original signal, this would allow the student to better be able to visualise how the signal would look in a real-world situation. And how different line coding techniques can be used to mitigate the effects of noise. Additionally, it would help the student visually understand how the eye diagram is an important tool when designing a communication protocol; As if there is no clear time period where the receiver can correctly sample the signal and interpret the correct signal value then there will be errors across the communication channel.

### Entropy

I would like to add the ability for the student to select the entropy of the original signal. They would do this by altering the probability that a binary one is generated. This would allow them to more easily understand how different line coding schemes affect the output signal. Additionally, by allowing the user to select the probability with a slider, the visualisation can provide feedback as to how their input affects the entropy of the signal by displaying an automatically updating entropy equation. Such as the one shown below.

Where is the probability of the symbol being generated. And, is the number of symbols used in the alphabet for the communication protocol; for the case of a binary signal this would be two.

### Additional Line Coding Techniques

If time permits I would wish to include additional line coding techniques for the student to be able to select from. In particular, I would wish to include techniques such as 4B5B line coding. This techniques maps for input bits onto 5 output bits for transmission [1]. This technique ensures that there will always be enough bit transitions to produce a self- clocking system regardless of the input bits. Due to this 4B5B encoding is used in the USB Power Delivery specification [2]

### Implement the Inverse Fourier Transform via WebAssembly

Finally, if time permits I would like to implement the Inverse Fourier Transform algorithm in WebAssembly, in addition to the implementation via JavaScript. By doing this, I would be able to provide an additional datapoint when evaluating whether WebAssembly is an appropriate tool for implementing algorithms that process large sets of data for web- based teaching visualisations.

[1] *IEEE Ethernet,* IEEE Standard 802.3

[2] Wikipedia. *“Line Coding”, Wikipedia.org* [Online]. Available: <https://en.wikipedia.org/wiki/Line_code> [Accessed: 20 January 2023]

[1] Wikipedia. *“4B5B”, Wikipedia.org.* [Online]. Available: https://en.wikipedia.org/wiki/4B5B[Accessed: 20 January 2023]

[2] USB Implementers Forum *USB Power Delivery* usb.org [Online]. Available: <https://www.usb.org/document-library/usb-power-delivery> [Accessed: 20 January 2023]