MCC125 Wireless link project
Final Report

Bingcheng Chen

December 22, 2023

Chalmers University of Technology
Wireless link project MCC125

1 Introduction

This report describes the software part of the project in which we implemented a one-way wireless communication system that can communicate over 100 meters.

The report begins by detailing the design of both the transmitter and receiver. Subsequently, the demonstration results are presented. Following this, we delve into a system enhancement endeavor by incorporating a neural network model into the receiver.

2 Software Design

2.1 Transmitter Design

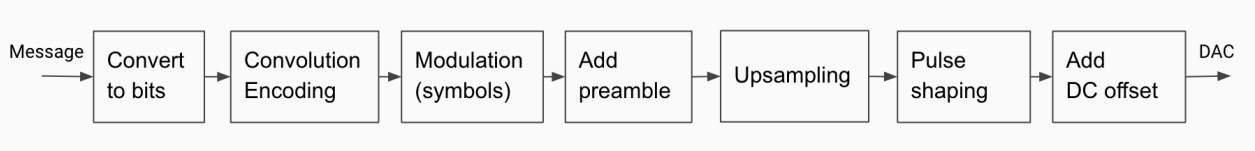


Figure 1: Block diagram of the transmitter

A block diagram depicting the transmitter design can be found in Figure 1, the design of the transmitter involves a series of critical steps to ensure reliable transmission of data. The first step in this process is the conversion of the message into bits using ASCII code, in this case we send a text message.

Following this, the transmitter employs convolution encoding, a technique used to enhance data reliability by introducing redundancy bits into the message bit stream, which can help in error detection and correction at the receiver end. We used a rate 2/3 feedforward convolutional encoder, as you can find the structure in Figure 2, the encoder has two input streams, three output streams, and seven shift registers.

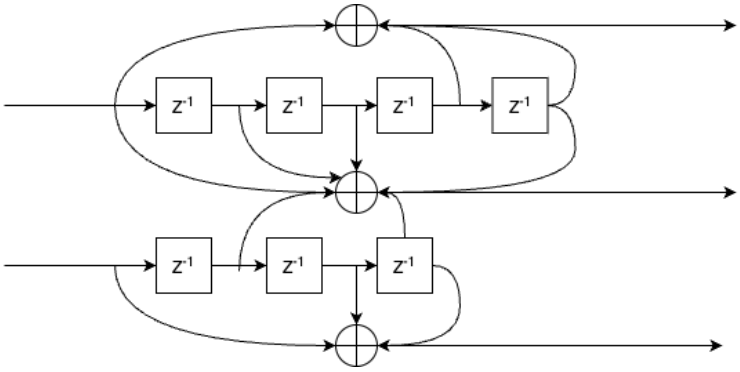


Figure 2: Trellis Structure for $\frac{2}{3}$ Feedforward Convolutional Encoder

Subsequently, the modulation process is employed to convert the encoded bits into symbols, QPSK and 16-QAM modulation were implemented during the demonstration. After modulation, the transmitter adds a preamble to the signal, a predefined sequence of [-1,1] that assists

the receiver in message detection and synchronization, we randomly generated a length of 150 sequence for preamble.

Following the adding preamble step, the transmitter exercuted upsampling to increase the number of samples in a signal. Then, pulse shaping(RRC filter) are implemented to change the shape of the transmitted pulses, it helps control the bandwidth of the transmitted signal.

Finally, a DC offset is added to the signal to facilitate coarse frequency correction at the receiver. In this specific instance, the employed DC offset consists of a predetermined sequence of fixed values, specifically set to 0.5.

2.2 Receiver Design

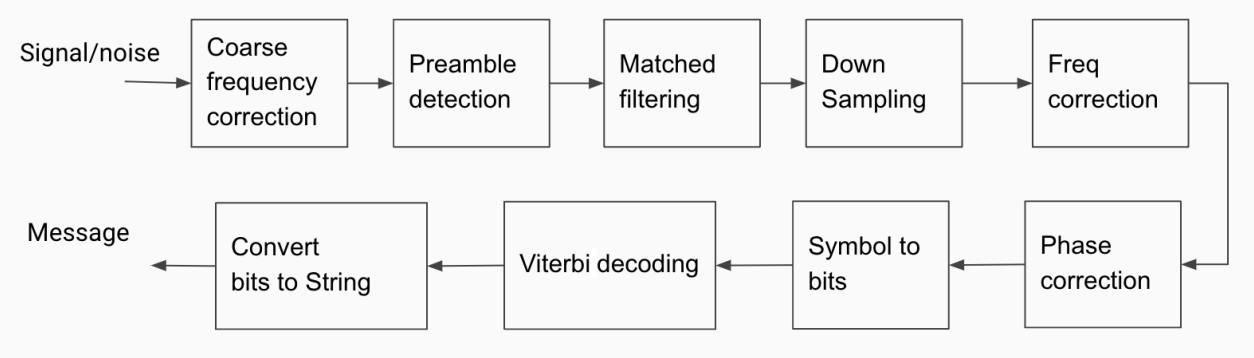


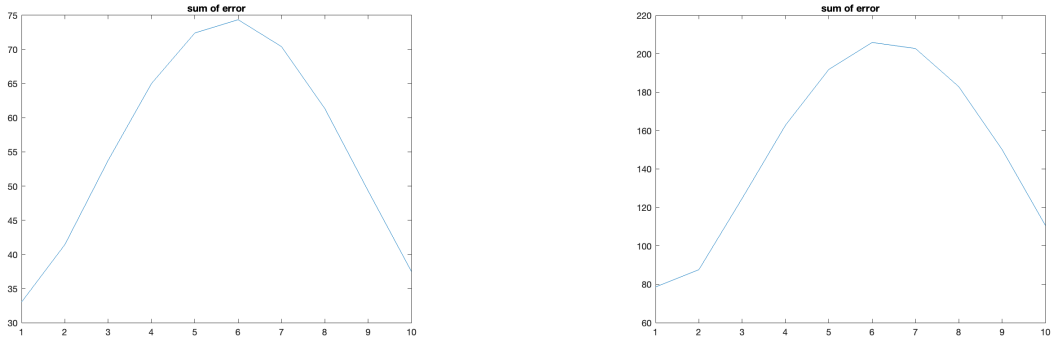
Figure 3: Block diagram of the receiver

A block diagram depicting the receiver design can be found in Figure 3, the first step involves coarse frequency correction by finding the peak in the power spectrum of the received signal, typically it is necessary due to the potential introduction of a certain degree of DC offset in the DAC. Additionally, it's important to note that we deliberately introduced extra DC offset in the transmitter to help finding its localization. By coarse frequency correction, it would make it easier to detect the preamble in the second step.

In preamble detection, the correlation between the received signal and the upsampling and filtered preamble is calculated. The highest peak in the correlation that surpass the predefined threshold indicates the start of the message.

The third step is matched filtering (the same RRC filter we used in the transmitter), it is employed to enhance the signal-to-noise ratio and mitigate the effects of channel noise.

Subsequently, downsampling is executed to reduce the sample rate (decrease the amount of data that needs to be processed) by picking the optimum sample which leads to a minimum of error sum, as illustrated in Figure 4.



a) Sum of error (QPSK, 100m wireless demo) b) Sum of error (16-QAM, 100m wireless demo)

Figure 4: Example of picking the optimum sample (sps=10), in both case we pick the 1st sample from the received signal

Following these above steps, the receiver now can make frequency correction and then phase correction, this is done by exploiting the difference between the received preamble and transmitted preamble which we both have now. The adopted methodology involves initially calculating the angle difference between the two preambles. Subsequently, a linear regression model is applied to the 150 angle points (the length of preamble we use in this case is 150), yielding the frequency offset in Hertz (determined by the slope divided by 2π) and the phase offset in radians (represented by the intercept).

The latter steps of the receiver involve demodulation, converting the modulated signal back into bits. Viterbi decoding is then applied to efficiently correct errors introduced during the transmission, enhancing the overall reliability of the communication system.

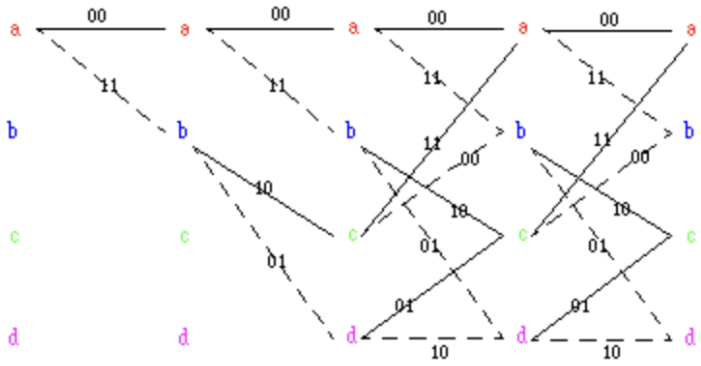


Figure 5: Viterbi Algorithm [1]

Finally, the receiver converts the decoded bits into a string, reconstructing the message transmitted.

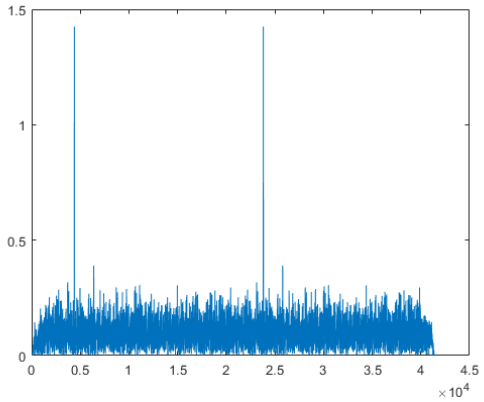
3 Result

3.1 Result - 256-QAM, Cable Connection

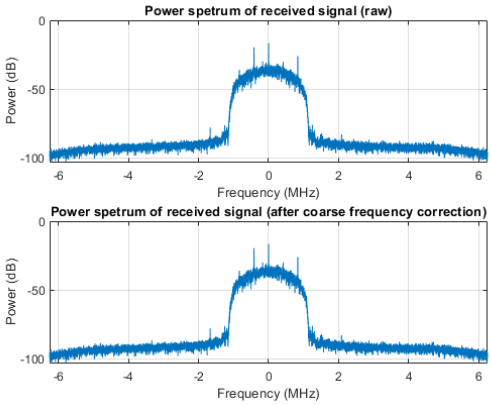
As illustrated in Figure 6, the system underwent testing on the USRP via a cable connection, resulting in the accurate reception of the transmitted message. The configuration parameters employed in this test are listed in Table 1.

Parameter	Value
Bitrate R_b	10 Mbits/s
Modulation format	256-QAM
Symbol rate R_s	1.25 MBaud/s
Rolloff factor / span	0.8 / 6
Sampling frequency	12.5MHz
sps (samples per symbol)	10
Bandwidth	1.25MHz
Interpolation factor	8

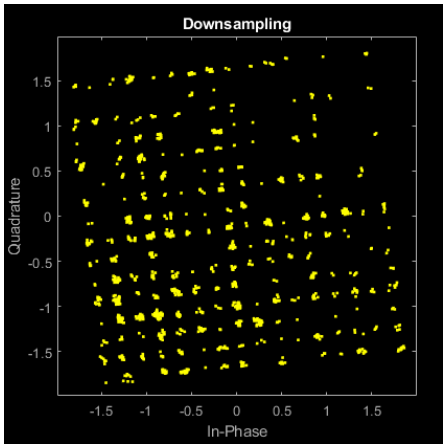
Table 1: Parameters set for 256-QAM



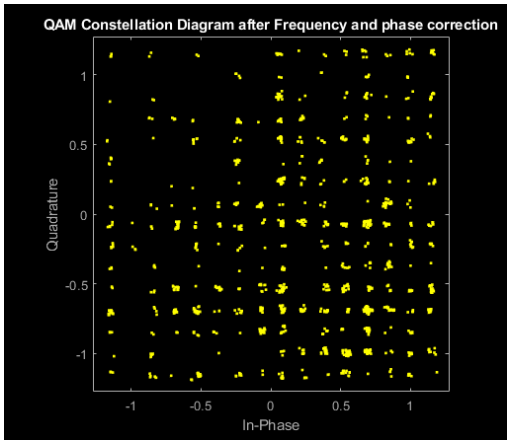
a) Preamble detection



b) Coarse frequency correction



c) After Downsampling



d) After frequency and phase correction

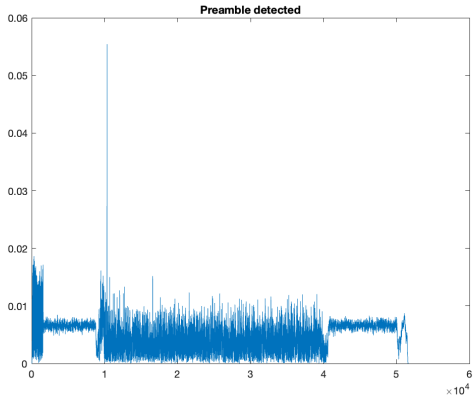
Figure 6: Plots of the received signal in different stages - 256-QAM, Cable connection

3.2 Result - QPSK, Wireless Link, 100m distance

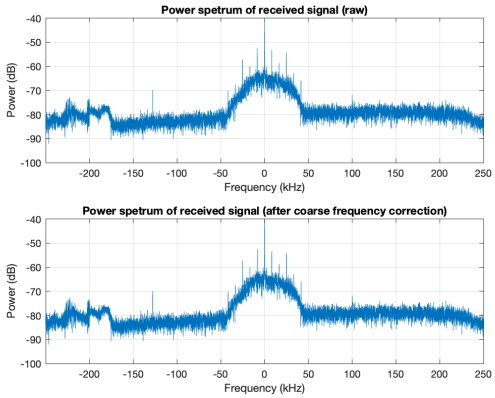
Similarly, Figure 7 demonstrates the system's successful testing on the USRP through the wireless link (100m). In this scenario, the transmitted message was received without error, and the parameter settings are outlined in Table 2, the SNR is roughly 15 dB.

Parameter	Value
Bitrate R_b	0.1 Mbits/s
Modulation format	QPSK
Symbol rate R_s	50 KBaud/s
Rolloff factor / span	0.8 / 6
Sampling frequency	25 MHz
sps (samples per symbol)	10
Bandwidth	50 KHz
Interpolation factor	4

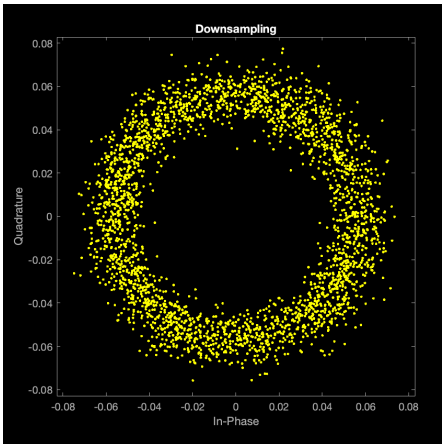
Table 2: Parameters set for QPSK



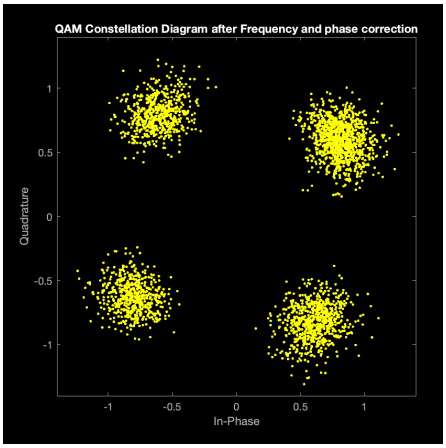
a) Preamble detection



b) Coarse frequency correction



c) After Downsampling



d) After frequency and phase correction

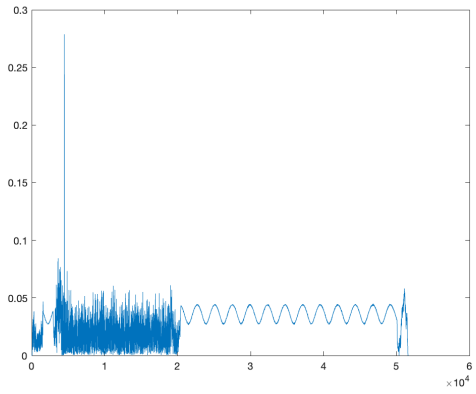
Figure 7: Plots of the received signal in different stages - QPSK, Wireless Link, 100m

3.3 Result - 16-QAM, Wireless Link, 100m distance

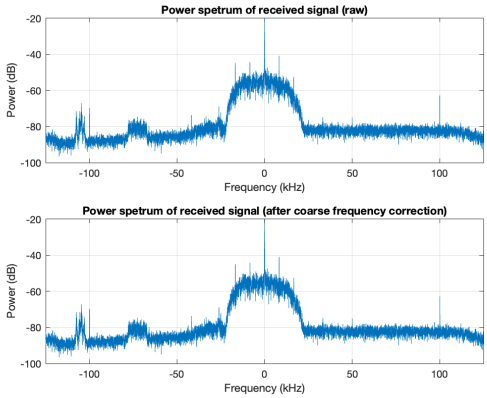
Furthermore, as shown in Figure 8, 16-QAM is tested on the USRP via a wireless link, with the received message accurately captured. We increased the magnitude factor of the transmitted signal to 0.4, the SNR is about 20 dB, The specific parameter values utilized in this test are provided in Table 3.

Parameter	Value
Bitrate R_b	0.1 Mbits/s
Modulation format	16-QAM
Symbol rate R_s	25 KBaud/s
Rolloff factor / span	0.8 / 6
Sampling frequency	25 MHz
sps (samples per symbol)	10
Bandwidth	25 KHz
Interpolation factor	4

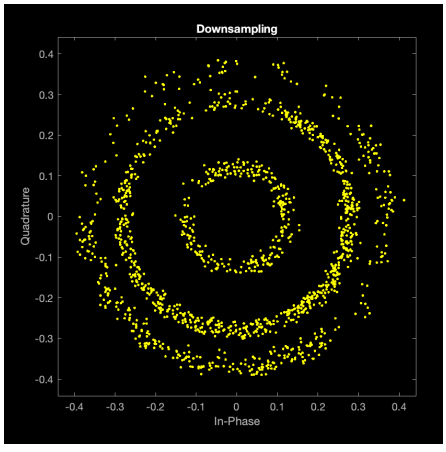
Table 3: Parameters set for 16-QAM



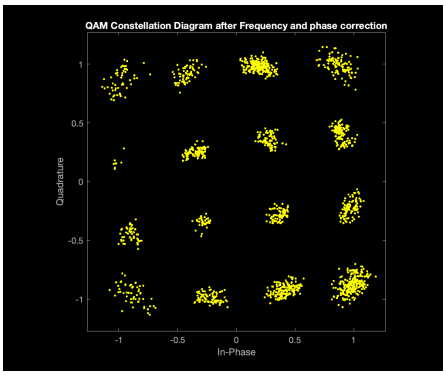
a) Preamble detection



b) Coarse frequency correction



c) After Downsampling



d) After frequency and phase correction

Figure 8: Plots of the received signal in different stages - 16-QAM, Wireless Link, 100m

4 System Improvement Attempt

By analysing the outcomes in part 3, it is evident that decoding the received signal accurately becomes more challenging when confronted with a real wireless channel as opposed to a cable connection.

An more advanced algorithm is expected to handle the complex effect of channel in the receiver. Three main limitations with the previous design are outlined below:

1. In the Downsampling step, a significant portion of the received signal is discarded. However, the discarded signal contains valuable information about the channel. Exploiting this information could be beneficial for improved decoding.
2. The frequency and phase correction step utilizes only a linear regression model, oversimplifying the complex effects introduced by the channel.
3. The previous design heavily relies on the preamble (for message detection / frequency correction / phase correction), yet the preamble's length is limited.

4.1 New receiver design

To overcome these limitations, a new receiver software design is proposed, as depicted in Figure 9. This design omits three steps from the previous version—Downsampling, Frequency Correction, and Phase Correction—replacing them with a neural network model, the Model is expected to capture a border characteristics of the channel than the previous design.

The challenge now is converted to how to train a model that can decoding received signals accurately.

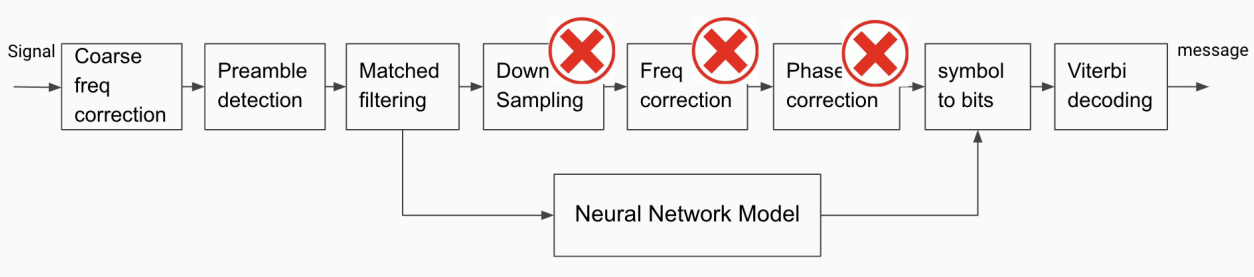


Figure 9: Block diagram of New receiver

Before delving into the specifics of model training, it is beneficial to clarify the three main steps on how the model will be well trained.

As you can see from Figure 10, the first step involves training a base neural network model using a extensive large data set sourced from diverse channels and hardware. The objective in this step is to make the model have a fundamental general capacity to decode received signals. Since the training data comes from different channel, so the model does not develop a specific understanding of any one channel during this step.

In step 2, the base neural model undergoes fine-tuning using data exclusively collected from a designated channel where the model is intended to operate. In this step, it is expected that the model learns the characteristics of a specific channel.

It is worth noting that we don't need to retrain a channel model from scratch for a new channel if a base model is well trained, but to fine tune the base model using data collected the specific channel. This strategy enhances computational efficiency and reduces the volume of required data. The expectation is that the model will surpass the performance of the previous design through the combination of steps 1 and 2.

In step 3, reinforcement learning technology can be involved here to make the model stronger, the underlying concept is to enable the model to learn dynamically as it encounters new information, but the challenge here is how to define a good reward/punishment policy.

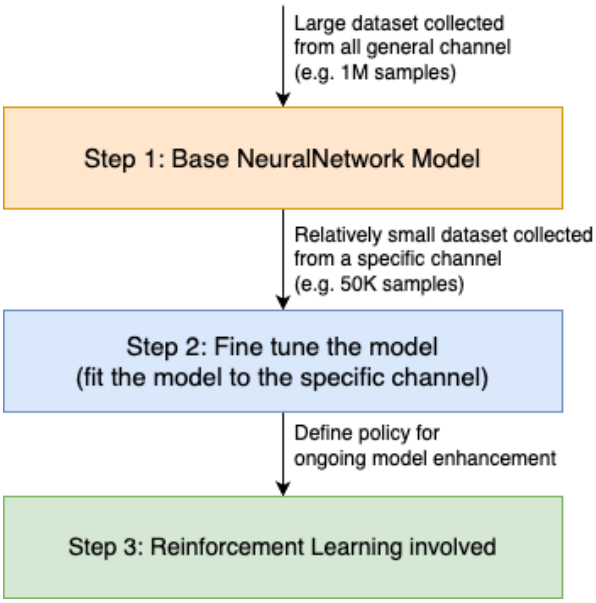


Figure 10: Three main steps on how to train a model

4.2 Collecting data

Utilizing the previous design system, depicted in Figure 11, allows us to collect data for training the model. as we can see, **X** represents the input data fed into the model, while **Y** corresponds to the expected output of the model. Therefore, **X, Y** constitute the data essential for training the model.

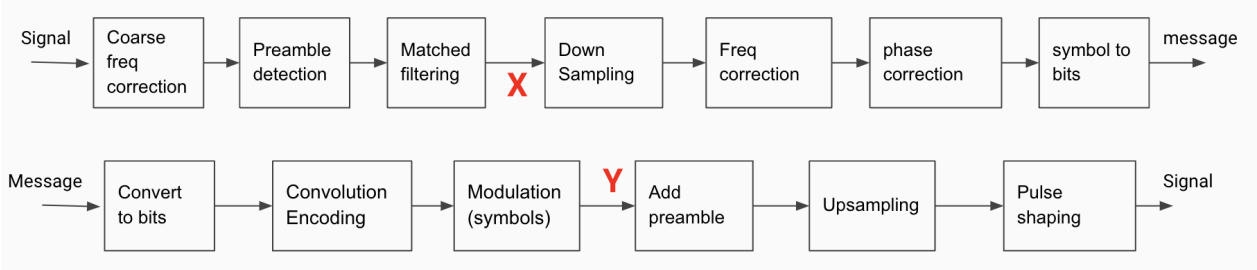


Figure 11: Collecting data using old system design

During the data collection in a real channel, hardware plays a crucial role. Therefore, ensuring the continuous operation of the hardware is important to collect enough data. In this scenario, depicted in Figure 12, we established an extra UDP connection between the transmitter and receiver. This leverages the timeout mechanism inherent in the UDP protocol, enabling the receiver to wait for an appropriate time to facilitate synchronization between two hardware.

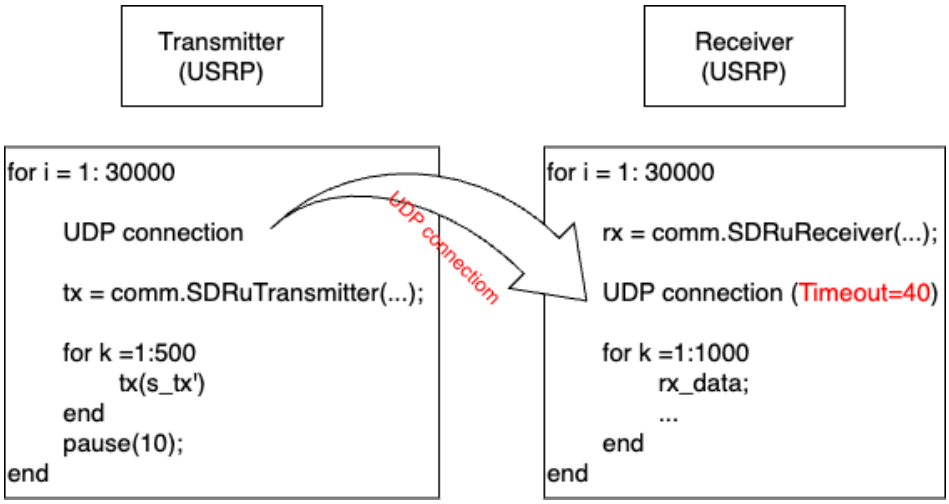


Figure 12: Collecting data using hardware

4.3 Model selecting and training

The neural network architecture selected for the channel model is the transformer architecture, it is first introduced in the paper "Attention is All You Need" by Vaswani et al [2]. in 2017. Since then it has been widely used in many areas such as natural language processing, computer vision, time series analysis, and more.

Considering the channel's features as a 3-D image in the space, then if the transformer works well on computer vision tasks, we might hypothesize its potential use as an architecture for channel modeling.

Transformer is characterized by its self-attention mechanism, which allows it to capture relationships between tokens in a sequence. Recall that in the earlier system design, our approach to frequency and phase correction involved leveraging relationships identified through

preamble. Subsequently, we applied the inverse relationship to the received signal to get the transmitting symbols, this exactly the similar pattern we want the model to learn from the data.

The self-attention mechanism in the Transformer architecture is commonly formulated as follows.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

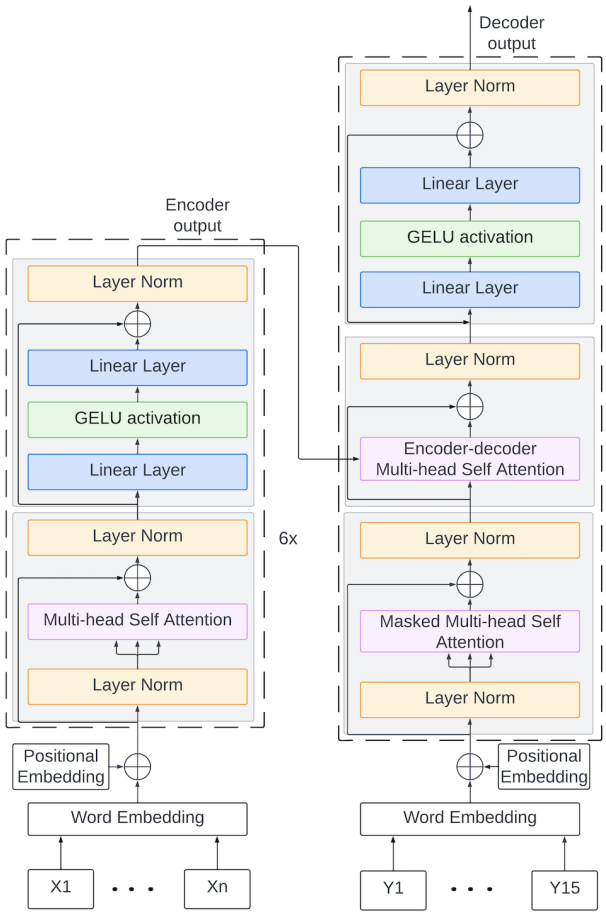


Figure 13: Transformer Architecture

To speed up the training process, since it takes more time to train the model if you have a long encoder input sequence, thus we adjusted the length of the preamble from 150 to 80, each transmitted message comprises 150 bits (training data, randomly generated), the samples per symbol (sps) is set to 5, and the employed modulation format is $1024 - QAM$.

To train a base neural network model, we generated a data set consisting of 30,000 samples simulately in matlab, where the channel is a Gaussian channel with a signal-to-noise ratio (SNR) of 30 dB, and random frequency and phase offsets are introduced.

Consequently, the length of the input sequence $\mathbf{X}$ is calculated as *sps* multiplied by the sum of the preamble length and the signal length. This results in an input $\mathbf{X}$ size of $(30000, 475, 2)$, while the target $\mathbf{Y}$ size is $(30000, 15, 2)$.

4.4 Model testing

As the model is trained using PyTorch, and the system is implemented in MATLAB, it becomes necessary to invoke the trained model within MATLAB after the training process.

Figure 14 depicts the procedure for calling the pre-trained model in MATLAB, with Python serving as an intermediary translator between the two ends.

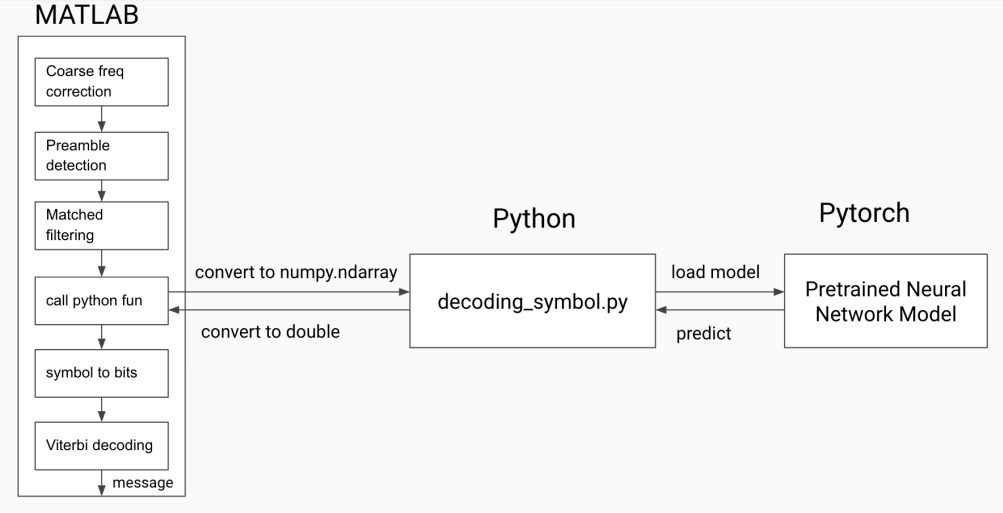
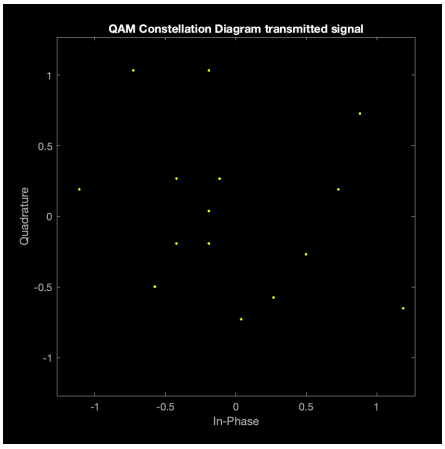


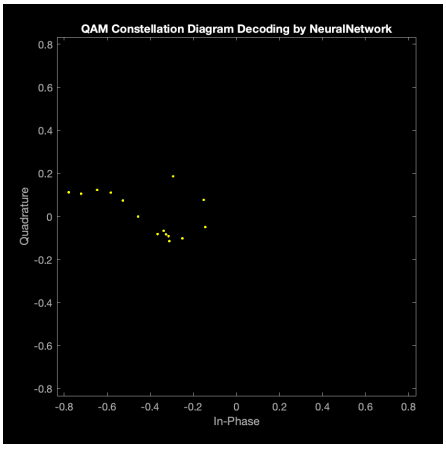
Figure 14: Data flow in the new designed receiver

Once the model is trained, we transmit a predefined message (testing data distinct from the training set) to the receiver for model testing. As depicted in Figure 15, we plot the constellation of the transmitted signal and the model predictions at epochs 10, 20, 50, 70, and 90.

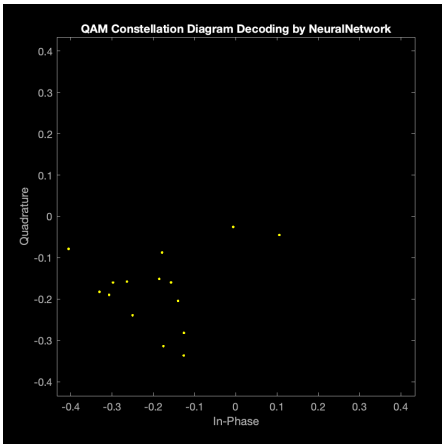
It can be found that even after 90 epochs of training, the model still can not capture the right pattern of the data, resulting in entirely inaccurate predictions.



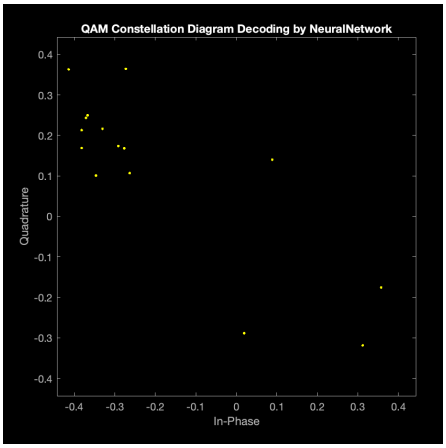
a) Transmitted Signal



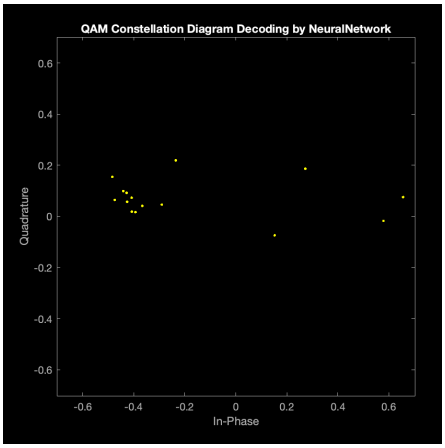
b) Model prediction - 10 epochs



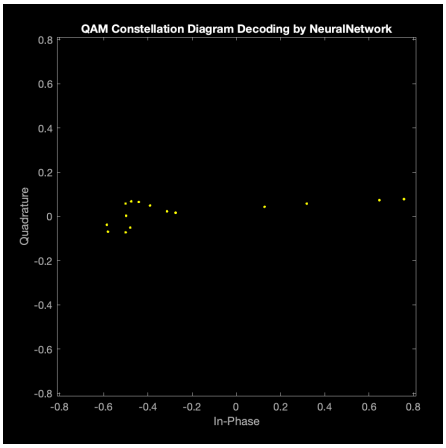
c) Model prediction - 20 epochs



d) Model prediction - 50 epochs



e) Model prediction - 70 epochs



f) Model prediction - 90 epochs

Figure 15: Constellation of the transmitted signal and the model predictions after different epochs of training

4.5 Reflection

By analysing the inaccuracies in the model predictions, several conclusions have been drawn:

1. The encoder sequence $\mathbf{X}$ feed into the Transformer model is long, this makes the attention mechanism harder to find the pattern of the data given limited number of training data.
2. The number of training samples (N=30000) appears far from sufficient for the Transformer network to effectively learn and generalize in this case.
3. Potential solutions: acquiring a larger dataset, simplifying features, or considering a modification in the neural architecture, such as convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN).

5 Conclusion

In this report, we described two different software design of a wireless communication system.

In the first design, we employed a conventional receiver, the system was tested on hardware in $256-QAM$ via cable connection, as well as $QPSK$ and $16-QAM$ through a 100m wireless link. In all three scenarios, the received messages were accurately decoded.

For the second design, we explored a new approach by incorporating a receiver with a neural network model. The model is based on transformer architecture, after training the model on 30000 samples for 90 epochs, however, the model's predictions was entirely inaccurate. Despite this setback, there is a firm belief in the high potential of this method to surpass the performance of the first design. Several possible solutions were discussed to enhance the accuracy of the model.

To access the code used in this report, it is available at https://github.com/chenbin234/MCC125-Wireless-link-project.

References

[1] 'The Viterbi algorithm', http://www.wirelesscommunication.nl/reference/chaptr05/receiver/viterbi/epfl_1/decoding.htm. Accessed 22 December. 2023.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need (2017), arXiv:1706.03762 [cs.CL].