Development of an OFDM modulator with frame synchronization, equalization in the frequency domain and detection

Leonardo Stefanini - 285129@studenti.unimore.it
Alessandro Neri - 293434@studenti.unimore.it
Vincenzo Garofalo - 282517@studenti.unimore.it

January 17, 2024

Abstract

Development and implementation of an Orthogonal Frequency Division Multiplexing (OFDM) modulator with a focus on frame synchronization, frequency equalization, and symbol detection, using MATLAB code environment and a single Adalm Pluto SDR.

1 Introduction

The objective of this project is to develop an OFDM system and implement it in a single Adalm Pluto software defined radio (SDR). The code part has been entirely done in MATLAB environment.

In this scenario, the main issues to address are: the frame synchronization, referring to the correct estimation of the start of the received frame; equalization, whose operation is to mitigate the detrimental effect introduced by the channel (frequency and time selectivity); and detection, in order to assign a correct estimation of the symbol given the information available at the receiver.

2 Algorithms

2.1 Frame Synchronization

The technique employed for the frame synchronization is cross-correlation based [1] and the goal consists on finding the maximum of a metric which marks the start of the useful part of the OFDM symbol. In general terms, the OFDM symbol is built with Np samples representing the cyclic prefix followed by other N samples, where N is the order of the IFFT. The training symbol is made with two identical halves, each having $L = N/2$ samples. These training data are usually built with a PN (Pseudo-Random Noise) sequence.

The start of the frame is found calculating:

$$\max_d M(d) = \max_d \frac{1}{Np+1} \sum_{k=-Np}^{0} M_f(d+k)$$

where d is the time index and the metric $M(d)$ is an average over a window large $Np+1$ samples of the function:

$$M_f(d) = \frac{|P(d)|^2}{R_f^2(d)}$$

If the received samples at baseband are denoted with $r(n)$, the cross-correlation $P(d)$ is computed as follows:

$$P(d) = \sum_{m=0}^{L-1} r^*(d+m)\, r(d+m+L)$$

The term $R_f(d)$ that divides $P(d)$ into $M_f(d)$ is defined as

$$R_f(d) = \frac{1}{2} \sum_{m=0}^{N-1} |r(d+m)|^2$$

and represent half of the training symbol's energy if d is at the start of the frame and, more precisely, at position Np+1.

Under the assumption of zero noise and channel distortion, the metric reaches a peak of almost 1 when the index d respects the condition mentioned before.

2.2 Frequency Equalization

Wireless communication channels in real environments exhibit a non-flat frequency response so that distinct spectral components of the transmitted signal are attenuated differently.

The main idea behind equalization is to erase or at least mitigate the channel behaviour by using a known training sequence to extract an estimate of the channel response.

If each subcarrier has a bandwidth smaller than the coherence bandwidth of the communication channel, the frequency response of the channel can be approximated with constant amplitude within the bandwidth of the subcarrier.

Having defined $H(i)$ as the frequency response of the channel, the following signal model is expected at the output of the FFT:

$$\begin{cases} Y[0] = H_0 c_0 + W[0] \\ \vdots \\ Y[i] = H_i c_i + W[i] \end{cases} \tag{2.1}$$

Where W is the additive white Gaussian noise, c_i is the i-th information symbol and i is the index referring to the i-th useful sub-carrier, which means

$$i \in 0, \ldots, N_\alpha + 1, N - N_\alpha, \ldots N - 1$$

As a result, an estimation of $H(i)$ in the different sub-carriers can be obtained by dividing the received symbols by a known training sequence.

Assuming the noise to have relative importance, it is shown that:

$$H_i \approx Y[i] c_i^{-1} \tag{2.2}$$

given c_i known transmitted symbols (training sequence).

For the sake of this implementation, the training sequence length has been put to N_u (where N_u denotes the number of useful sub-carriers) symbols so that it is possible to estimate the effect of the channel on each sub-carrier.

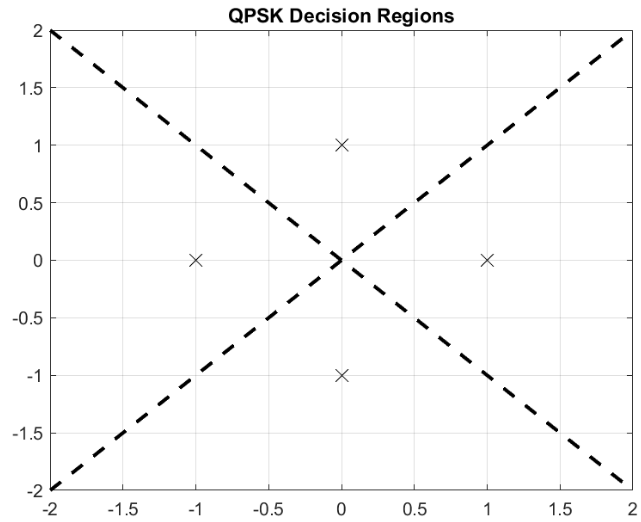
After obtaining an estimation of H, the equalization is done simply by dividing the received constellation points by the corresponding value of H_i, so that the frequency selectivity is mitigated:

$$Y_{eq}[i] = Y[i] H_i^{-1} \tag{2.3}$$

2.3 Detection

The detection algorithm is based on a minimum distance criterion which allows to take a decision on the received symbol by choosing the constellation point which has the least amount of distance from the point received.

In the case of a Q-PSK constellation, the decision regions that are shown in the figure below are defined.



In order to take the best decision, the following function needs to be minimized:

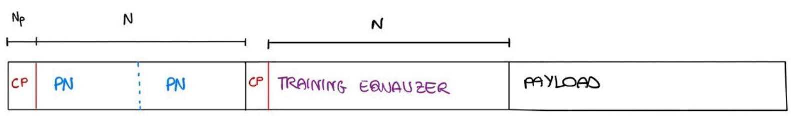
$$f(c_i) = |c_n - Y_{eq}|^2$$

So the constellation point which has the greatest probability of being the correct one is:

$$\underset{c_n}{min}\ |c_n - Y_{eq}|^2 \tag{2.4}$$

3 Frame structure

The following image summarizes the frame structure. The approach adopted involves one OFDM symbol for the synchronization and one OFDM symbol containing the equalizer training sequence.



4 Developed Software

The MATLAB code starts with common initializations such as the roll-off factor $\alpha = 0.25$, the oversampling factor $N_c = 10$, the IFFT order $N = 256$ and the TX and RX filter design.

In order to simulate what could be a frequency selective channel response, the following code is employed

```
% Amplitudes of the paths
a = [0.8; 0.4; 0.4];
```

```
% Delays of the paths
tau = [0; -1; 2];
% Frequency-dispersive channel
ptau = sum(a.*truncRRC(n, alpha, tau), 1)/(sum(a));
```

In this way, the properties of the channel are merged with the impulse response of the TX filter.

Like previously discussed in the algorithms section, in order to make an estimation of the channel effect, a training sequence is employed:

```
% Generation of data input and symbol mapping
symtx_data       = randi([0,M-1], nSym, 1);

% Generation of sequence of data used for equalization
training_equalizer = randi([0,M-1], Nu, 1);

symtx = [training_equalizer; symtx_data];
[symtxg, ~] = togray(symtx, mb);      % Gray mapping
ci          = psklev(symtxg+1);       % Symbol selection
```

The corresponding training symbols are converted to gray code and then mapped in order to be used later for equalization means:

```
[tr_eq_gray, ~] = togray(training_equalizer,mb);
known_ci = psklev(tr_eq_gray+1);
```

A supplementary training symbol to be used for timing estimation is now created by putting in series two identical random binary sequence (PN sequence) and by adding the cyclic prefix.

```
% Set the length of the PN sequence used for timing estimation
pn_length = N/2;

% Generate a random binary sequence
pn_sequence = -1 + 2.*randi([0, 1], 1, pn_length);

% Repeat the PN sequence to create the training symbol
training_symbol = repmat(pn_sequence, 1, 2);

% Add cyclic prefix to the training symbol
training_symbol_with_prefix = [training_symbol(end - Np + 1:end), training_symbol];
```

The code immediately following deals with essential operations in OFDM signal processing, such as reorganization of symbols into parallel blocks of length N_u, virtual carrier insertion, IDFT computation, cyclic prefix insertion, conversion of the signal from parallel to series and introduction of the syncronization training symbol at the beginning of the frame.

```
% Serial to parallel block
r = rem(nSym, Nu);
if r~=0
    txt = sprintf('discarding last %d symbols', r);
    disp(txt);
    ci = ci(1:end-r);
```

```
end

ci_par = reshape(ci, Nu, []);

Nbl = size(ci_par,2); % Overall number of blocks

% Virtual carrier insertion
ci_N = [ci_par(1:Na+1,:); zeros(Nsc, Nbl); ci_par(Na+2:end,:)];

% IDFT
a_N = N*ifft(ci_N);

% Cyclic prefix insertion
a_NT = [a_N(end-Np+1:end,:); a_N];

% Parallel to series conversion
a_NT = a_NT(:);

% Training symbol insertion
a_NT = [training_symbol_with_prefix.'; a_NT];
```

An up-sampling operation is done to the signal previously obtained and, finally, in order to obtain the transmitted signal *txSig*, the signal is filtered by the transmitting filter (which takes into account the behaviour of the channel).

```
% TX filtering: upsample and filter
a_up    = upsample(a_NT, Nc)/Nc;
txSig    = filter(ptau, 1, a_up);
```

This code section allows to connect MATLAB with the Adalm Pluto SDR.
Firstly the rate at which the device is configured to collect data (*SampleRate*), the size of the buffer transfer from the radio to MATLAB and the number of frame to collect are defined. It is important to correctly set these parameters in order to avoid overflows conditions.

```
%% Radio paramaters

SampleRate = 1e6;
SamplesPerRXFrame = 2^14; % Change to 2^16 to remove overflow
FramesToCollect = 5;
```

Subsequently the same Pluto SDR device has been initialized both as receiver (rx) and transmitter (tx), the function *transmitRepeat* ensures that the SDR transmits repeatedly the same frame over time and finally some data (precisely *FramesToCollect*SamplesPerRXFrame* samples) is captured through the receive antenna.

```
%% Set up radio and capture some data
rx = sdrrx('Pluto','SamplesPerFrame',SamplesPerRXFrame,...
    'BasebandSampleRate',SampleRate,'OutputDataType','double');
tx = sdrtx('Pluto','Gain',-30,...
    'BasebandSampleRate',SampleRate);
```

```
tx.CenterFrequency = tx.CenterFrequency + FrequencyOffset;
tx.transmitRepeat(txSig);

% Get data from radio
saved = zeros(FramesToCollect*SamplesPerRXFrame,1);
ofe = 0;
for g=1:SamplesPerRXFrame:FramesToCollect*SamplesPerRXFrame
    [data1,len,of] = rx();
    saved(g:g+SamplesPerRXFrame-1) = data1;
    if of % Count overflows
        ofe = ofe + 1;
    end
end
fprintf('Overflow events: %d of %d\n',ofe,FramesToCollect)

rxSig = saved;
```

At the receiver side all the samples of the received sequence pass through the matched filter and get down-sampled. The last operation of this section removes the filter transient.

```
%% Receiver Side
rxdem    = filter(pmt, 1, rxSig);
rx_ds    = downsample(rxdem, Nc);   % downsample demodulated
    signal
rx_sym = rx_ds(L+1:end);
```

$rxsym$ contains the repetition of the initial frame. At this point, the synchronization helps distinguishing the time indexes at which all the frames begin.

This code section starts with the definition of a window inside which a time index d evolves. This window should be large enough to include at least 2 frames, since the message is composed of 1000 symbols, it's been chosen a length of 4000 samples. Then the metric previously discussed is computed.

```
%% Timing Sync

Ltrain = N/2;

finestra = 4000;

Pgrande = zeros(1, finestra);
Rf = zeros(1, finestra);
Mf = zeros(1, finestra);
M1 = zeros(1, finestra);

 for d = 1: finestra

    r_asterisco = conj(rx_sym(d:d+Ltrain-1));
    r_base = rx_sym(d+Ltrain:d+2*Ltrain-1);

    Pgrande(d)= sum(r_asterisco .* r_base);


    r_N = rx_sym(d:d+N-1);
```

```

    Rf(d) = 0.5*sum(r_N.*conj(r_N));
    Mf(d) = (abs(Pgrande(d))^2) / (Rf(d)^2);

 end
 risultato = zeros(1, finestra);

 for d= 1:finestra
    for k = 0 : Np
        if d-k >= 1
            risultato(d) = risultato(d) + Mf(d-k);
        end
    end
 end

M1 = (1/(Np+1)) .* risultato;
```

In this project the radio transmits the same frame repeatedly, so in the following code lines an algorithm that finds two consecutive peaks of the metric is implemented. Between those two maximum values there are the samples of the useful part of the frame.

In general, the two peaks could have different heights so in order to obtain the values which correspond respectively with the start of the first frame and the following, this empirical method has been employed.

The algorithm draws a set number of maximum values from the metric (in this case 50) and if the maximum values are distant enough (so that they aren't related to the same frame) then the second maximum can be saved.

```
[picco,idx]=max(M1); %First peak detection

[piccok,idxk]=maxk(M1,50);
for indicemassimo = 1:50 %Gather 50 maximum values
    if abs(idxk(indicemassimo)-idx)>300 %If the peaks are distant enough
        secondomax = idxk(indicemassimo); %Take the index for the second peak found and exit the loop
        break;
    end
end

% The two peaks may not be in order so they are swapped in the case it happens

if idx > secondomax
    change = idx;
    idx = secondomax;
    secondomax=change;
end


% Selecting only the correct part of the received signal, corrisponding to the transmitted frame

rx_sym = rx_sym(primomax+N:secondomax-Np-1);
```

Within this code, the operations regarding the reception of the received signal are executed.

```matlab
%% S/P and removing CPs
r = rem(length(rx_sym), N+Np);
a_NTr = rx_sym(1:end-r);
a_NTr = reshape(a_NTr, N+Np, []);
a_Nr  = a_NTr(Np+1:end,:); % Removing all Cyclic prefixes

%% FFT
ci_Nr = 1/N*(fft(a_Nr));

%% Removing non useful subcarriers
ci_parr = [ci_Nr(1:Na+1,:); ci_Nr(end-Na+1:end,:)];
```

Having discharged the non useful sub-carriers, the following code proceeds to take an estimation of the channel response from the first block of data which it is known to be the training channel symbols, based on the equation (2.2).

```matlab
%% Equalization (assuming known channel)

Hstima = [ci_parr(:,1)].' ./ known_ci;
```

Following up, the number of blocks to equalize is computed and the equalized symbols are calculated.

```matlab
% Overall number of blocks to equalize
Nblock_eq = size(ci_parr,2);

equalized = zeros(Nu, Nblock_eq-1);

sym_da_eq_par = ci_parr(:,2:end);

equalized = sym_da_eq_par ./ Hstima.';

ci_eqserie = equalized(:);
```

The received symbols have to be associated to the nearest constellation points according to the minimum distance criterion.

Within these code lines the Euclidean distance between each possible transmitted symbol and the received symbols is calculated. Then, the index of the symbol in the constellation that has the minimum distance to the received Gray-coded symbols is extrapolated. Finally, the information contained in the symbol that we wanted to transmit to the receiver is obtained by reconverting the Gray-coded sequence.

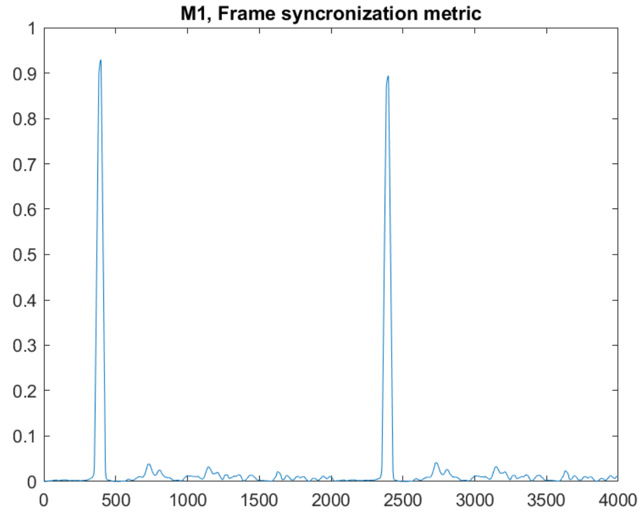
```matlab
%% DETECTION Algorithm
% Minumum distance criterion
dist_vec     = abs(psklev(:) - ci_eqserie.').^2;

[~, sym_idx] = min(dist_vec);

det_symg     = sym_idx - 1; % Gray-coded detected symbol

[det_sym, ~] = fromgray(det_symg, mb); % Detected symbol
```

5 Results

5.1 Frame Synchronization

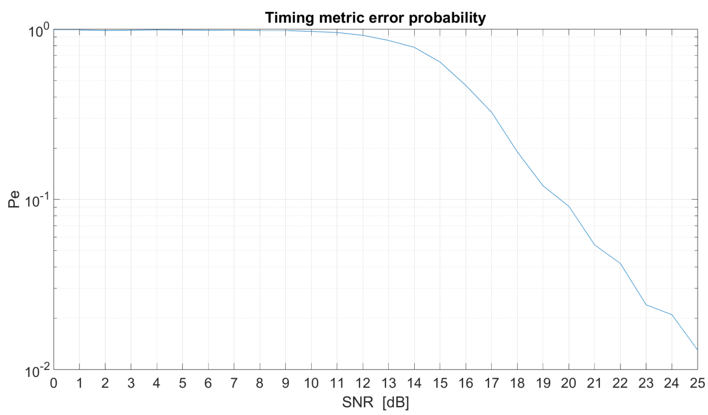
As previously stated, the metric yields the start of the useful part of the OFDM symbol. In the implementation chosen for this project, the single radio transmits the frame repeatedly, so that a finite amount of memory is needed to save a part of the received values, corresponding to a certain listening window.

Having set the window for the metric to be 4000 symbols, two of the peaks for the cross-correlation show on the plot.



After simulating the system, it turned out that the bottleneck is the metric. Infact, each time it fails, the symbol detection is completely wrong.

The following graph shows the error probability of the metric with respect to SNR.

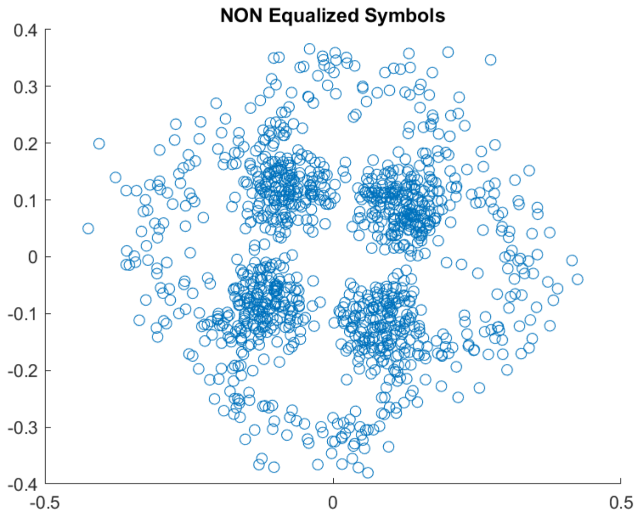


The downside of this synchronization method is the lack of robustness against noise (compared to other sophisticated methods), even with SNR = 20 dB the metric flops one time out of ten.

These measurements have been done without using the SDR, because of the impossibility to control the SNR.

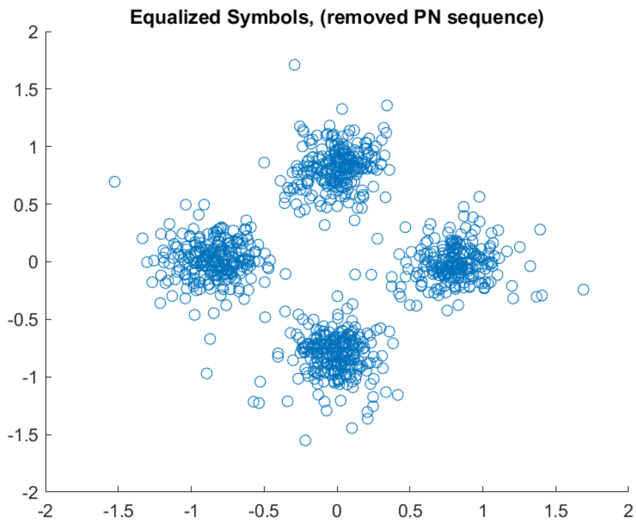
5.2 Equalization

The equalization algorithm is tested in the real scenario. By plotting the received constellation, it is shown that the channel introduces noise and a bit of phase rotation on the points, which would make the detection perform very poorly.



After equalizing, the symbols position are much more recognizable and take place closer to the original constellation than before.

This helps and improve correct detection, excluding negligible amount of symbols which could be ambiguous to detect due to the presence of approximations in the estimations and residual noise.

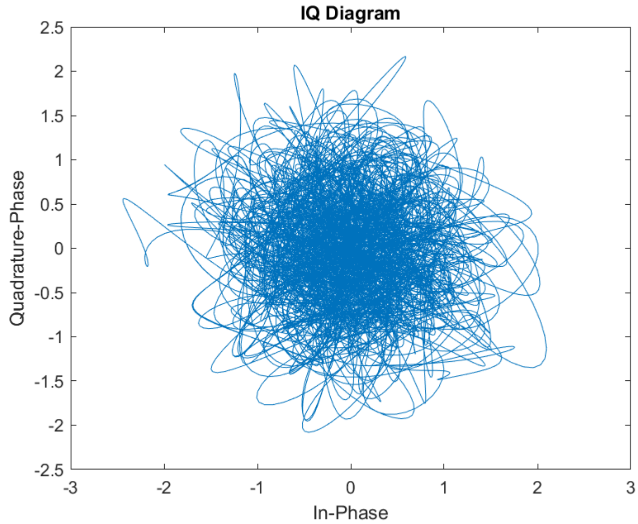


5.3 Detection

As previously established, finding the correct start of the frame is crucial for detection means. When the correct initial point of the frame is found, the performance of the detector is observable from the error rate. For this system, error rate is kept under 1%.

5.4 Other results

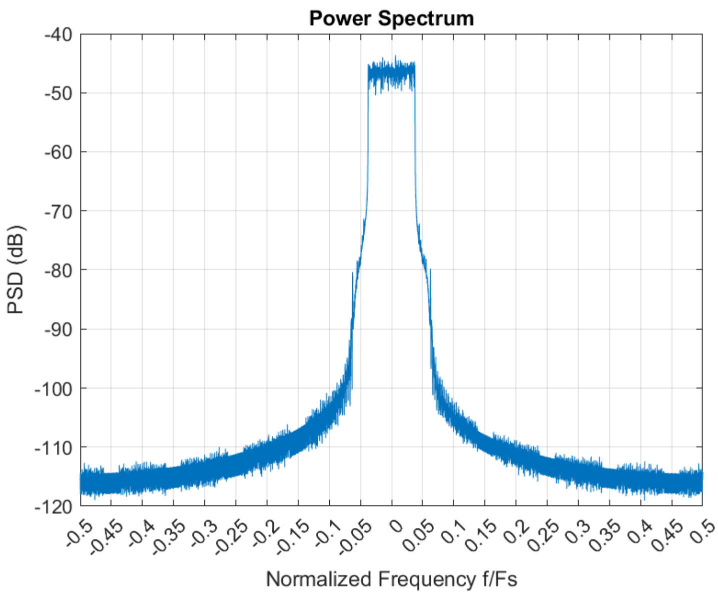
The following figure represents the I-Q diagram characterizing the complex envelope of an OFDM signal that has been transmitted.



One of the characteristics that distinguish OFDM from other modulation is the fact that the modulus of the complex envelope exhibits huge variation over time. As a result, a lot of energy is wasted because the power amplifier can't always operate into its maximum yield region.

The intensity of those fluctuations of the complex envelope is quantified by a parameter called peak-to-average-power ratio (PAPR) and the values calculated for our system fluctuate between 8 to 12 dB. A smaller value of PAPR is desirable in order to achieve better energy efficiency.

Here there is the representation of the normalized PSD on a logarithmic scale. It is important to notice that the signal transmitted is not band limited as expected because the impulse response of the transmitted filter has a finite duration. Nevertheless a significant portion of the power is between -0,05 and 0,05.



6 Conclusions

The implementation of the OFDM modulator with frame synchronization, equalization in the frequency domain, and detection has been successfully carried out, even though the frame synchronization algorithm isn't the most sophisticated, it achieves fair results given the low computational complexity.

The proposed equalizer has given good performance in mitigating the channel effects as it can be seen from the graphs, enhancing the recognition of symbols and reducing the impact of channel-induced noise.

The error rate of the system in non relatively noisy environments exhibit excellent outcome.

Future work should focus on refining the synchronization technique, exploring alternative algorithms for improved robustness and implementing the OFDM system in two SDR devices (one transmit and the other one receive) so that frequency selectivity can be observed without having to introduce it through software.

References

[1] H. Minn, M. Zeng, and V.K. Bhargava. On timing offset estimation for ofdm systems. *IEEE Communications Letters*, 4(7):242–244, July 2000.