Proyecto de Graduación

Mitigación del ISI (Interferencia Intersimbólica) sobre un Sistema WiMAX Fijo

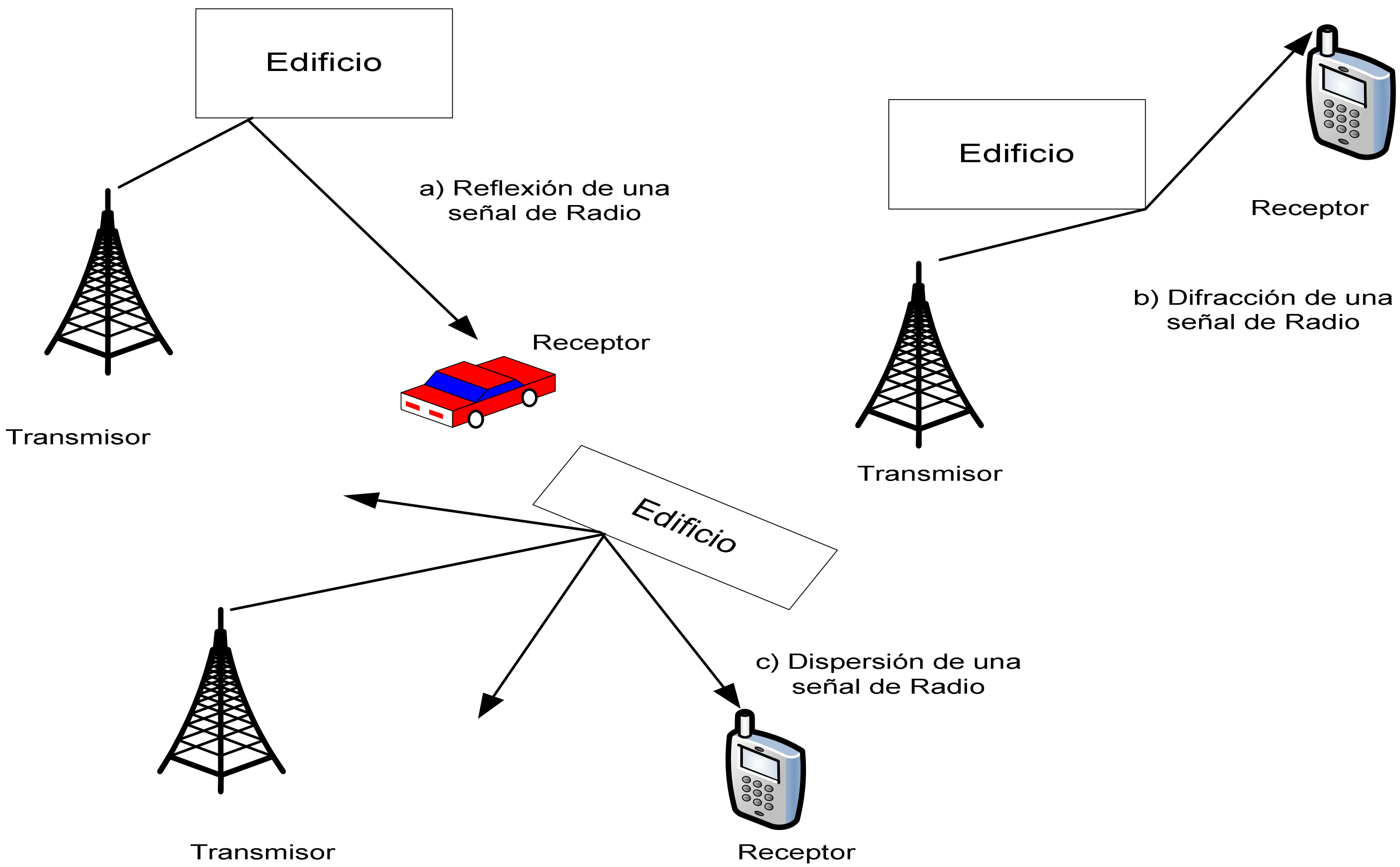
Byron Floreano – Santiago Jácome

Contenido

1) Comprensión del Problema.
2) Soluciones.
3) Modelamiento y Simulación.
4) Resultados y Conclusiones

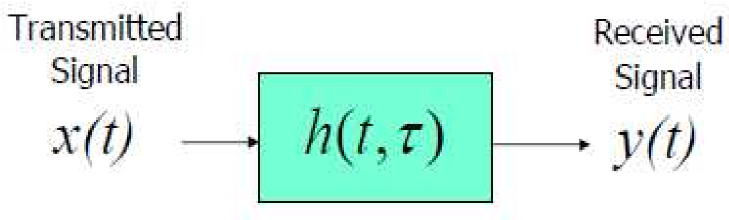
1) Comprension del problema

- Porque ocurre ISI?



Comprension del problema

- Canal Multicamino

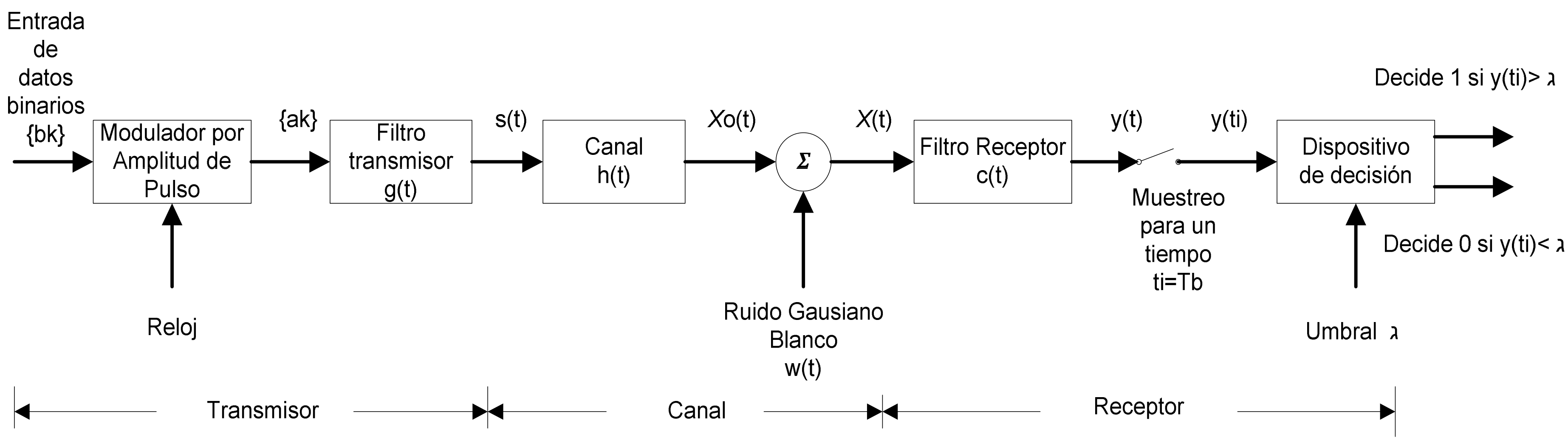


$$h(t,\tau) = \sum_{i=0}^{L-1} a_i(t)\,\delta(\tau - \tau_i)$$

$$y(t) = \int_{-\infty}^{+\infty} x(t-\tau)h(t,\tau)d\tau$$

Comprension del problema

- ISI



Comprensión del problema

$$a_k = \begin{cases} +1\ ; & Si\ el\ simbolo\ b_k = 1 \\ -1\ ; & Si\ el\ simbolo\ b_k = 0 \end{cases}$$

$$s(t) = \sum_k a_k g(t - kT_b)$$

$$y(t) = \mu \sum_k a_k p(t - kT_b) + n(t)$$

ISI

$$\mu p(t) = g(t) * h(t) * c(t)$$

$$\mu P(f) = G(f) \times H \times C(f)$$

$$y(t_i) = \mu \sum_{k=-\infty}^{\infty} a_k p[(i-k)T_b] + n(t_i) = \mu a_i + \mu \sum_{\substack{k=-\infty \\ k \neq i}}^{\infty} a_k p[(i-k)T_b] + n(t_i)$$

Escenario

- Sistema WiMAX Fijo.
- Canal NLOS tipo Rayleigh con desvanecimiento lento y selectivo en frecuencia en pequeña escala.
- Modulación MQAM(M=16 y M=32).

Small Scale Fading

Desvanecimiento a Pequeña Escala
(Basado en el esparcimiento de retardo multicamino)

Desvanecimiento Plano

1. BW de la señal < BW del canal
2. Esparcimiento de retardo< Periodo de símbolo.

Desvanecimiento de Frecuencia Selectiva

1. BW de la señal > BW del canal
2. Esparcimiento de retardo>Periodo de símbolo

Desvanecimiento a Pequeña Escala
(Basado en el esparcimiento Doppler)

Desvanecimiento Rápido

1. Esparcimiento Doppler alto
2. Tiempo coherente<Periodo de símbolo.
3. La variación del canal es más rápido que la variación de la señal en banda base.

Desvanecimiento lento

1. Esparcimiento Doppler bajo
2. Tiempo coherente>Periodo de símbolo.
3. La variación del canal es más lento que la variación de la señal en banda base.

2) Soluciones

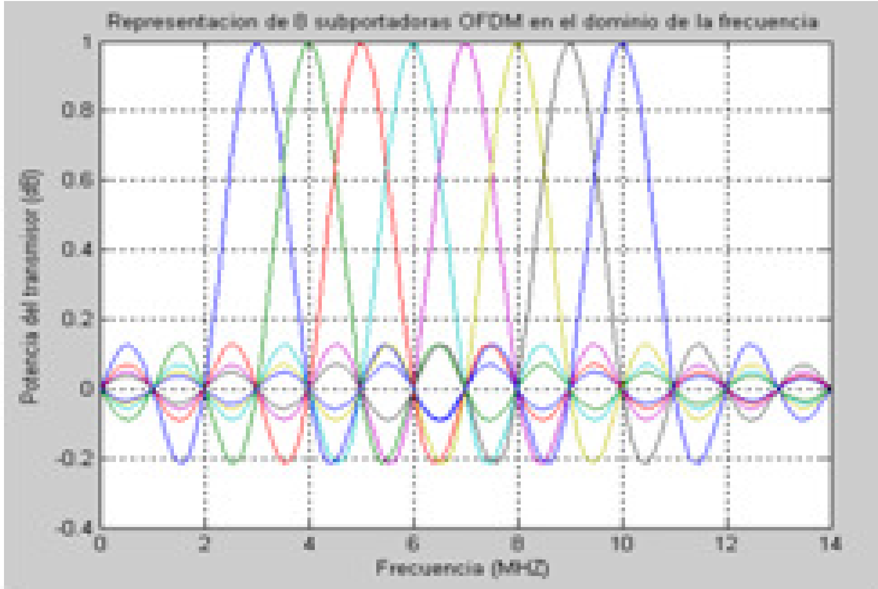
- Dentro de las posibles soluciones, para mitigar el problema del ISI están:
  - OFDM
  - Ecualización Adaptiva, Zero-Forcing y MMSE.
  - Direct Sequence and Frequency Hopping pread Spectrum (DSSS y FHSS).
  - Señal Piloto

OFDM

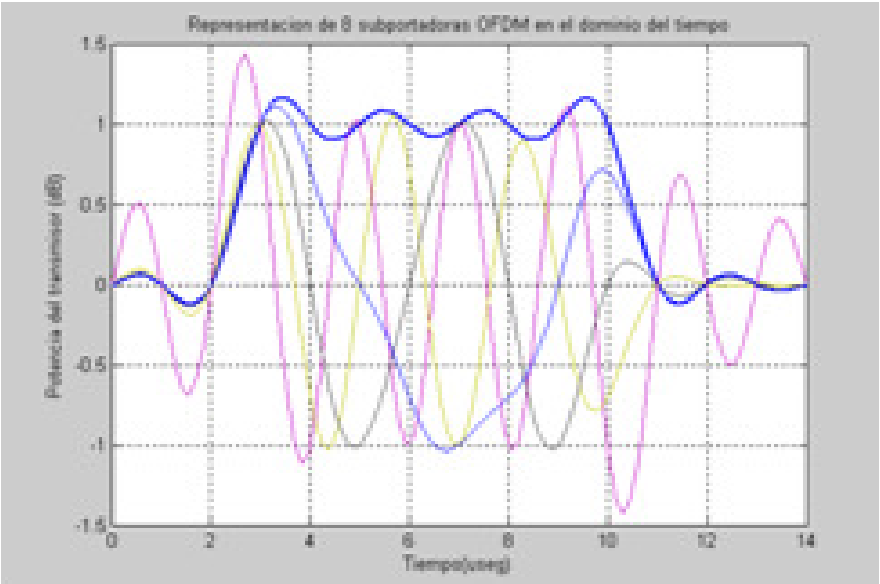
- Puede ser usado en canales con desvanecimiento de frecuencia selectiva y lo que hacen es particionar la señal banda base en varias sub-bandas, y transmitirlas bajo múltiples portadoras ortogonales.

OFDM

- Representación de 8 subportadoras OFDM a) en el dominio de la frecuencia y b) en el dominio del tiempo

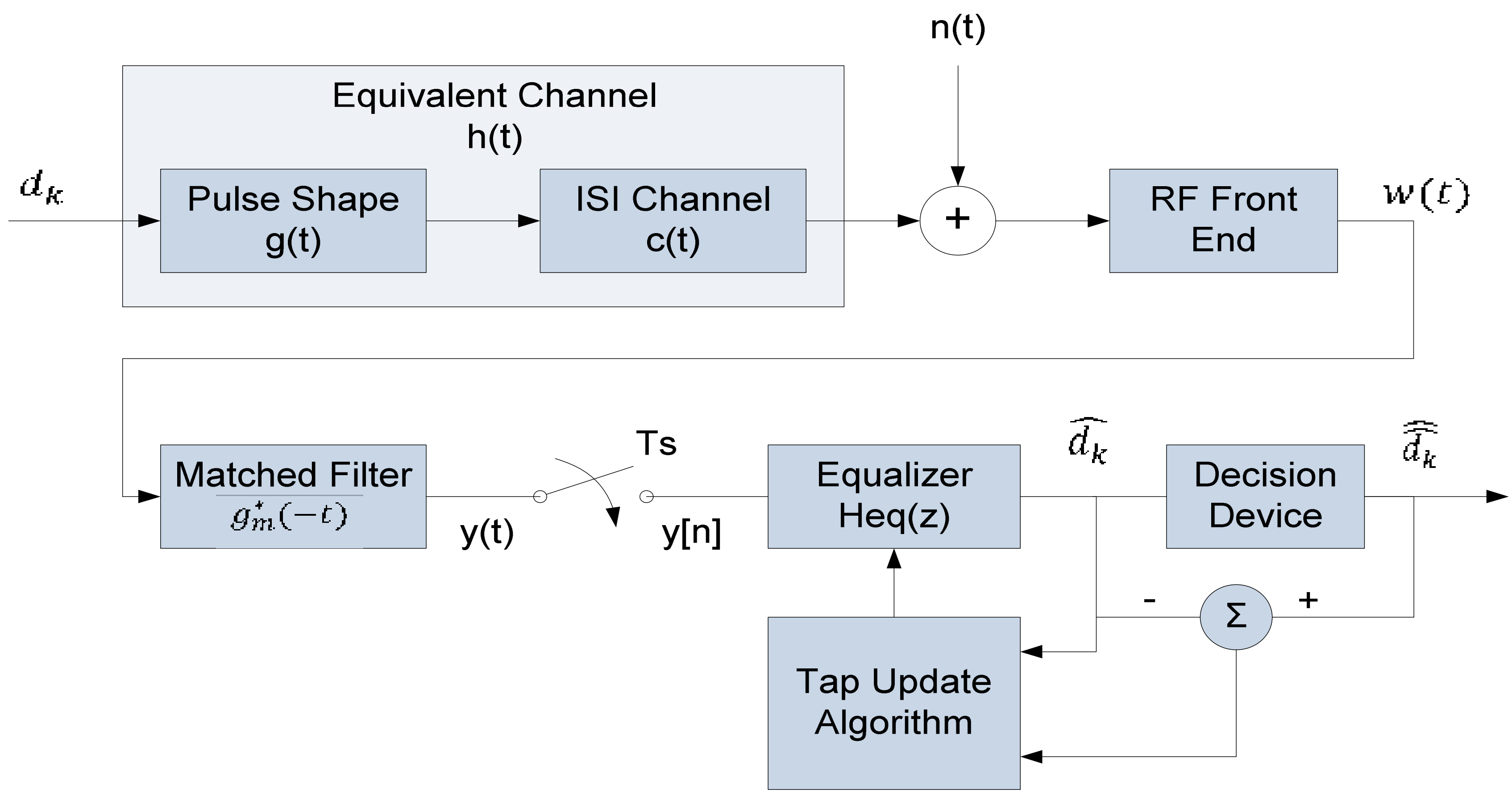


a)



b)

Ecualizadores



Sistema de un Ecualizador

Ecualizador Zero-Forcing

$$f(t) = g(t) * c(t) * g_m^*(-t)$$

$$y(t) = d(t) * f(t) + n_g(t) = \sum d_k f(t-kT) + n_g(t)$$

$$n_g(t) = n(t) * g_m^*(-t)$$

$$f[n] = f(nT_s) \qquad\qquad y[n] = y(nT_s)$$

$$y[n] = \sum_{K=-\infty}^{\infty} d_k f(nT_s - kT_s) + n_g(nT_s)$$

$$= \sum_{K=-\infty}^{\infty} d_k f[n-k] + v[n]$$

$$= d_n f[0] + \sum_{k \neq n} d_k f[n-k] + v[n]$$

Ecualizador Zero-Forcing

$$f(t) = h(t) * g^*(-t)$$

$$Y(z) = D(z)F(z) + N_g(z)$$

$$G^*(1/z^*)$$

$$F(z) = H(z)G_m^*(1/_{z^*}) = \sum_n f(nT_s)z^{-n}$$

$$H_{ZF}(z) = \frac{1}{F(z)}$$

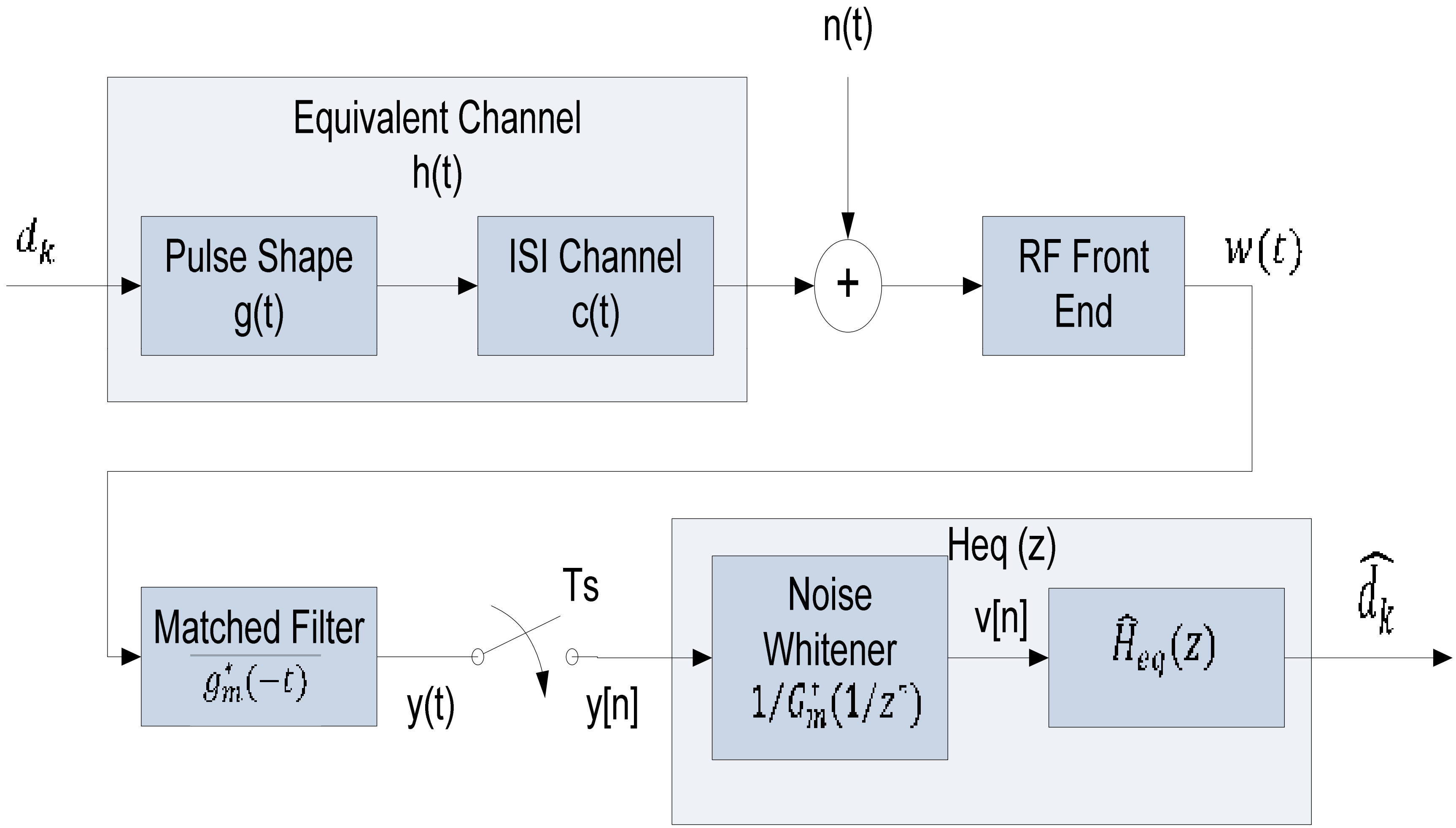
Ecualizador MMSE

- En el ecualizador MMSE el tema de diseño del ecualizador es para minimizar el error cuadrático medio entre la transmisión de símbolo y el estimado $\hat{d}_k$ para la salida del ecualizador. En otras palabras los $\{w_i\}$

  son escogidos para minimizar el valor $E[d_k - \hat{d}_k]^2$

  $\hat{d}_k$ es una combinación lineal de las muestras de entrada $y[k]$

Ecualizador MMSE



Ecualizador MMSE

$$\hat{d}_k = \sum_{i=-L}^{L} w_i y[k-i]$$

Los coeficientes óptimos de filtro $\{w_i\}$ conlleva a un problema estándar en la estimación lineal.

$$\hat{H}_{eq}(z) = \sum_{i=-L}^{L} w_i z^{-i}$$

$$V = (v[k+L], v[k+L-1] \ldots, v[k-L]) = (v_{k+L}, v_{k+L-1}, \ldots, v_{k-L})$$

$$W = (w_{-L}, \ldots, w_L)$$

$$\hat{d}_k = W^T V = V^T W.$$

$$J = E[d_k - \hat{d}_k]^2 = E[W^T V V^H W^* - 2R(V^H W^* d_k) + |d_k|^2]$$

.

Ecualizador MMSE

$$M_V = E[VV^H] \qquad V_d = E[V^H d_k]$$

M_V es una matriz Hermitiana de $N \; x \; N$ y V_d es un vector fila de tamaño N

$$E|d_k|^2 = 1$$

$$J = W^T M_V \; W^* - 2 \; R\{V_d W^*\} + 1$$

$$\nabla_W \, J = 0$$

$$\nabla_W \, J = \left(\frac{\partial J}{\partial w_{-L}}, \dots, \frac{\partial J}{\partial w_L}\right) = 2W^T \, M_v - 2V_d$$

$$W^T M_v = V_d$$

$$W_{opt} = (M_V{}^T)^{-1} \, V_d{}^T$$

Ecualizador MMSE

$$J_{min} = 1 - V_d M_V^{-1} V_d^H$$

$$V = (v_{n+\infty}, \ldots, v_n, v_{n-\infty}) \qquad W = (w_{-\infty}, \ldots, w_0, \ldots, \qquad w_\infty)$$

$$\sum_{i=-\infty}^{\infty} w_i (f[j-i] + N_0)\delta[j-i] = g^*{}_m[-j], \quad -\infty \leq j \leq \infty$$

$$\hat{H}_{eq}(z)(F(z) + N_0) = G^*(1/z^*)$$

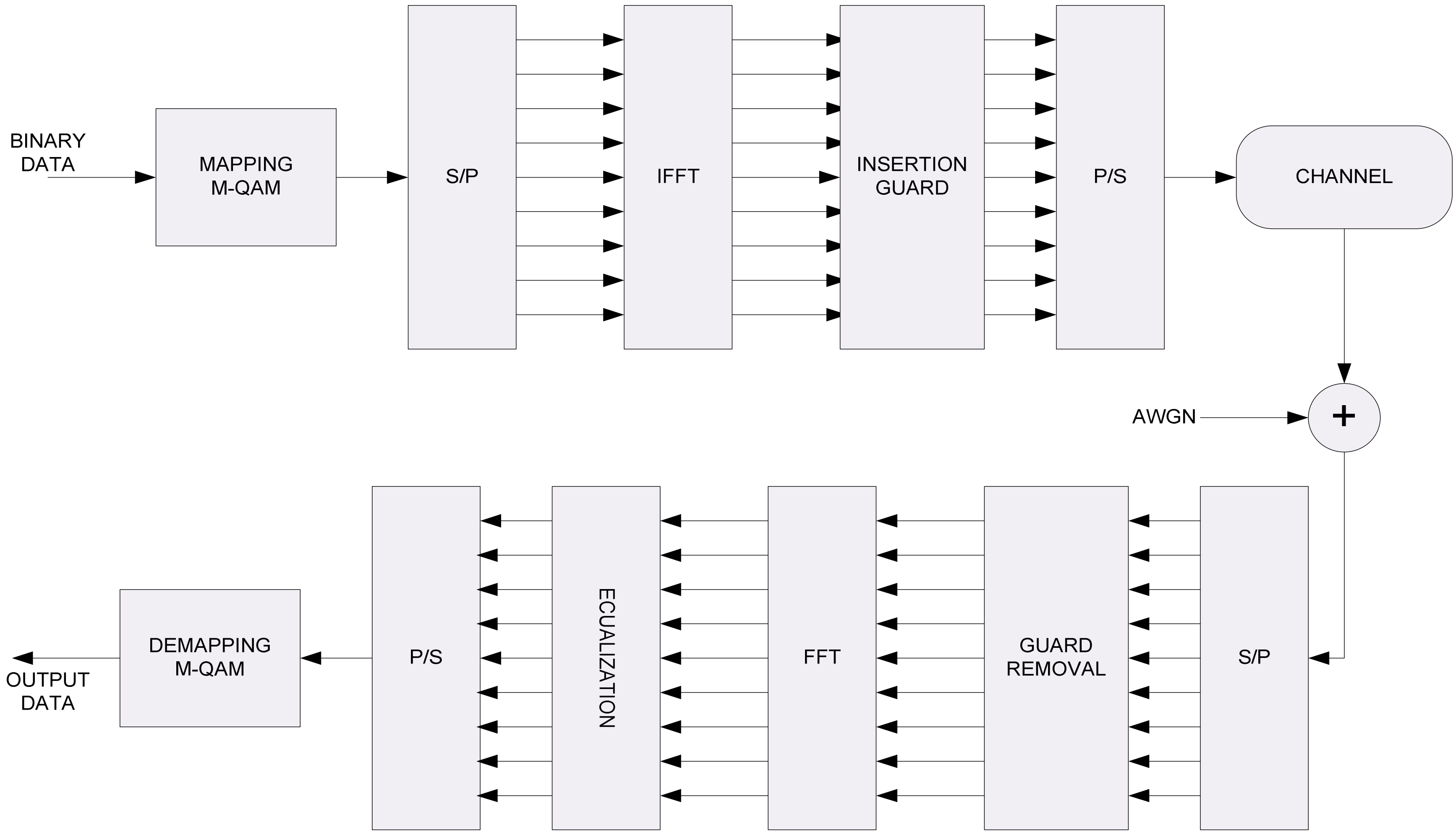
$$\hat{H}_{eq}(z) = \frac{G_m^*(1/z^*)}{F(z) + N_0}$$

Ecualizador MMSE

$$H_{eq}(z) = \frac{\tilde{H}_{eq}(z)}{G_m^*(1/z^*)} = \frac{1}{F(z) + N_0}$$

$$J_{min} = T_s \int_{-0.5/T_S}^{0.5/T_S} \frac{N_0}{F_\Sigma \quad (f) + N_0} \, df$$

3) Modelamiento y Simulación



Modelamiento y Simulación

Parámetros de Simulación

- Número de Subportadoras: 256
- Número de Subportadoras Dato: 256
- Número de Subportadoras Piloto: 0
- Numero de Subportadoras nulas/guardas: 0

- Prefijo Cíclico: [4, 8, 16, 32]

- Tipo de Canal: Rayleigh
- Tipo de Modulación: 16QAM y 32QAM

Modelamiento y Simulación

```
>> channel = rayleighchan(1/500000,200,[0 0.04 0.08 0.12]*1e-4,[0 -3 -6 -9])

channel =

             ChannelType: 'Rayleigh'
       InputSamplePeriod: 2.0000e-006
         DopplerSpectrum: [1x1 doppler.jakes]
         MaxDopplerShift: 200
              PathDelays: [0 4.0000e-006 8.0000e-006 1.2000e-005]
           AvgPathGaindB: [0 -3 -6 -9]
      NormalizePathGains: 1
            StoreHistory: 0
          StorePathGains: 0
               PathGains: [-0.6439 - 0.1968i 0.1296 - 0.3330i 0.5773 + 0.0894i -0.0332 - 0.0217i]
      ChannelFilterDelay: 0
    ResetBeforeFiltering: 1
     NumSamplesProcessed: 0

>>
```

Parámetros del Canal tipo Rayleigh.

CODIGO PRINCIPAL

```
%===================== Parameters ============================
NFFT = 256;                          % Length of FFT.
K = 4;                               % Bits encoded in a M-QAM symbol.
M = 2^K;                             % Level of Modulation M-QAM
Ns = 10;                             % Number of Symbols for transmission.
w_ZF = ones(1, NFFT);                % Filter coefficients.
w_MMSE = ones(1, NFFT);              % Filter coefficients.
SNRdB = 0:30;                                  % SNR of AWGN in channel in dB.
a = 1;                               % Figure Index

%=============== Vaiables for ZF Equ ============================
ber_ZF = zeros(1,length(SNRdB));
bit_error_rate_ZF = zeros(1,length(SNRdB));

%=============== Vaiables for MMSE Equ ========================
ber_MMSE = zeros(1,length(SNRdB));
bit_error_rate_MMSE = zeros(1,length(SNRdB));
```

CODIGO PRINCIPAL

```
%================== Simulation =====================
for CPLen = [4 8 16 32] % Length of Cyclic Prefix.

  for sym=1:Ns
  %---------- Data Genration -----------------
  input = randint(1,NFFT,M);
  % Decimal to Binary
  input2bin = de2bi(input,'left-msb');
  %------------ Transmit Data ----------------
  [signal_tx,input_symbols] = transmitter(input, NFFT, CPLen, M);
  for snr = 1:length(SNRdB)
  %-------------------- Channel -------------------------
  [signal_rx,h] = channel(signal_tx,snr);
```

CODIGO PRINCIPAL

```
%-------------- Receiver Data ZF and MMSE ----------------
  % Receiver ZF
  [signal_recovered_ZF, w_ZF, output_symbols_ZF] = receiver_ZF(signal_rx, input_symbols, NFFT, CPLen, M,
     w_ZF);
  signal_recovered_ZF2bin = de2bi(signal_recovered_ZF,'left-msb');

  % Receiver MMSE
  [signal_recovered_MMSE, new_w_MMSE, output_symbols_MMSE] = receiver_mmse(signal_rx,
     input_symbols, NFFT, CPLen, M, w_MMSE, snr);
  w_MMSE = new_w_MMSE;
  % Decimal To Binary
  signal_recovered_MMSE2bin = de2bi(signal_recovered_MMSE,'left-msb');

  %------- Bit Error Rate Instantaneous --------
  % BER ZF
  [nerr,ber_ZF(snr)] = biterr(input2bin, signal_recovered_ZF2bin);
  % BER MMSE
  [nerr,ber_MMSE(snr)] = biterr(input2bin,signal_recovered_MMSE2bin);
  end

  bit_error_rate_ZF = bit_error_rate_ZF + ber_ZF;
  bit_error_rate_MMSE = bit_error_rate_MMSE + ber_MMSE;
  end
```

CODIGO PRINCIPAL

```
%==================== BER Calculation ========================

   bit_error_rate_ZF = bit_error_rate_ZF/Ns;

   bit_error_rate_MMSE = bit_error_rate_MMSE/Ns;

   figure(a)
   color_ZF = 'b.-';
   color_MMSE = 'r.-';

   semilogy(SNRdB,bit_error_rate_ZF,color_ZF);
   hold on;
   semilogy(SNRdB,bit_error_rate_MMSE,color_MMSE);
   a = a+1;
   axis([-Inf Inf 10^(-4) 1])
   xlabel('Signal to noise ratio ')
   ylabel('Bit Error Rate')
   title(['Bit Error Rate for ',num2str(2^K),'QAM, with Guard = 1/',num2str(CPLen)]);
   legend('Equalizer ZF','Equalizer MMSE');
end
```

TRANSMITTER

```
function[trans_signal, input_symbols] = transmitter(input, FFTLen, CPLen, M)

%================== Baseband Modulation =======================
input_symbols = modulate (modem.qammod(M),input);

%=================== Constellation 16QAM ======================
scatterplot(input_symbols);

%=========================== IFFT ============================
input_time_para = ifft(input_symbols);

%==================== Parallel to Serial =========================
input_time_serial = reshape(input_time_para,1,FFTLen);

% Calcululo:
inicio = FFTLen - (FFTLen/CPLen);

%=============== Gaurd Interval Insertion (GII) ===================
input_ext = [input_time_serial((inicio + 1):FFTLen),input_time_serial];

%==================== Return output ==========================
trans_signal = input_ext;
end
%=====================  END FILE =============================
```

CHANNEL

```
function[output,h] = channel(input, SNRdB)


%+------------------------------------------------------------+
%|                 CHANNEL MODEL                              |
%+------------------------------------------------------------+
%=============== Pass Through Rayleigh Channel ====================

h = rayleighchan(1/500000,200,[0 0.04 0.08 0.12]*1e-4,[0 -3 -6 -9]);
h.StoreHistory = 1;
h.ResetBeforeFiltering = 1;
signal_filtered = filter(h, input);

%=============== Pass Through AWGN Channel ====================
output = awgn(signal_filtered, SNRdB,'measured');
end
%====================  END FILE ==============================
```

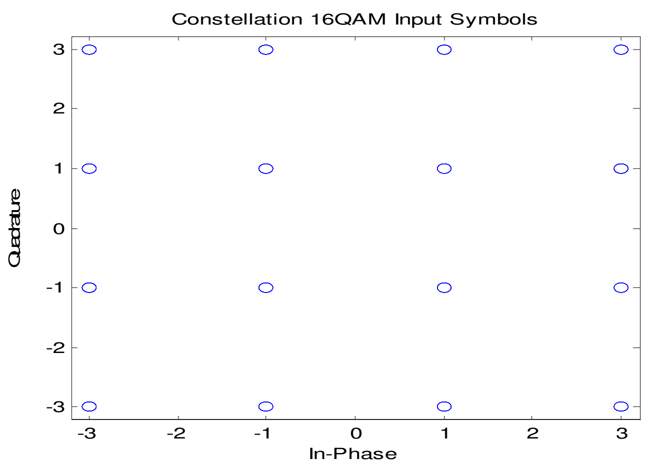
RECEIVER_ZF

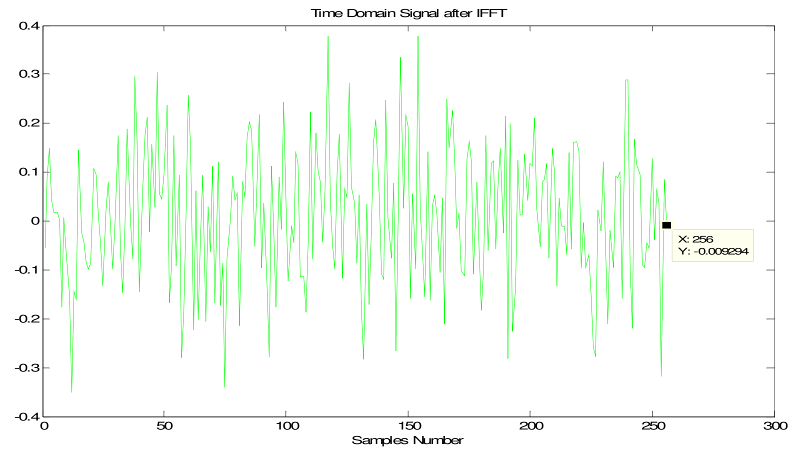
```
function[recovered_signal, w, output_symbols] =
    receiver_ZF(noisy_signal, input_symbols,
    FFTLen, CPLen, M, w)

%Calculo:
inicio_2 = FFTLen/CPLen + 1;

%==== Gaurd Interval Removal (GIR) ====
% Remove cyclic extension from noisy signal.
noisy_time_serial = noisy_signal(inicio_2:end);

%======= Serial To Parallel ===========
%output_time_para =
    reshape(noisy_time_serial,FFTLen,1);
output_time_para =
    reshape(noisy_time_serial,1,FFTLen);

%============== FFT ====================
output_noisy_symbols = fft(output_time_para);

%======== Zero Forcing Equalizer ======
% Channel estimator in frequency domain
h_est = output_noisy_symbols ./ input_symbols;
output_symbols = output_noisy_symbols ./ w;

%========= Baseband Demodulation =========
output_para =
    demodulate(modem.qamdemod(M),output_sy
    mbols);

%============ Returning output ===========
recovered_signal = output_para;
w = h_est;
end
%================  END FILE ============
```

RECEIVER_MMSE

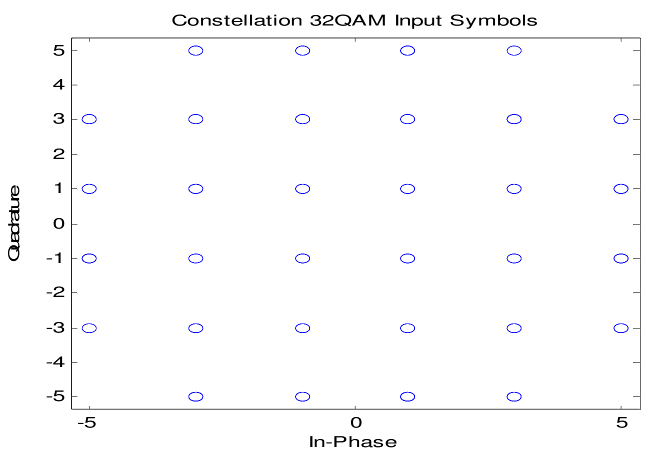
```
function[recovered_signal ,new_w, output_symbols] =
        receiver_mmse(noisy_signal, input_symbols, FFTLen,
        CPLen, M, w, SNRdB)

% Calculo:
inicio_3 = FFTLen/CPLen + 1;

%====Gaurd Interval Removal (GIR) ========
% Remove cyclic extension from noisy signal.
noisy_time_serial = noisy_signal(inicio_3:end);

%========== Serial To Parallel ============
        output_time_para =
        reshape(noisy_time_serial,1,FFTLen);

%================= FFT ===============
output_noisy_symbols = fft(output_time_para);

%========= Zero Forcing Equalizer ========
% LS channel estimator in frequency domain
h_est = output_noisy_symbols ./ input_symbols;

% MMSE equalizer
SNR = 10^(SNRdB/10);
new_w = conj(h_est) ./ ( abs(h_est).^2 + 1/SNR);
output_symbols = output_noisy_symbols .* w;

%=========== Baseband Demodulation =======
output_para =
        demodulate(modem.qamdemod(M),output_symbols);

%============ Returning output ========
recovered_signal = output_para;
end

%============== END FILE ===========
```

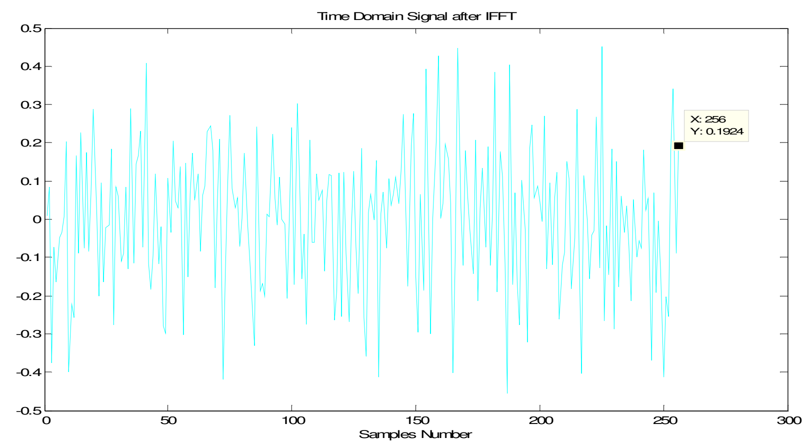
4) Resultados.





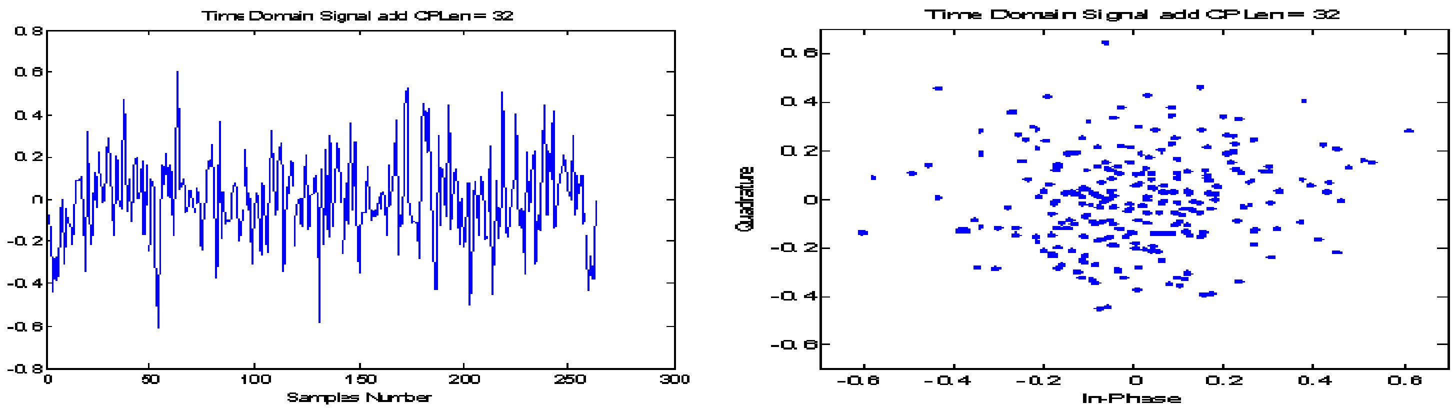
Constelación **16QAM** y Señal en el dominio del tiempo **16 QAM**



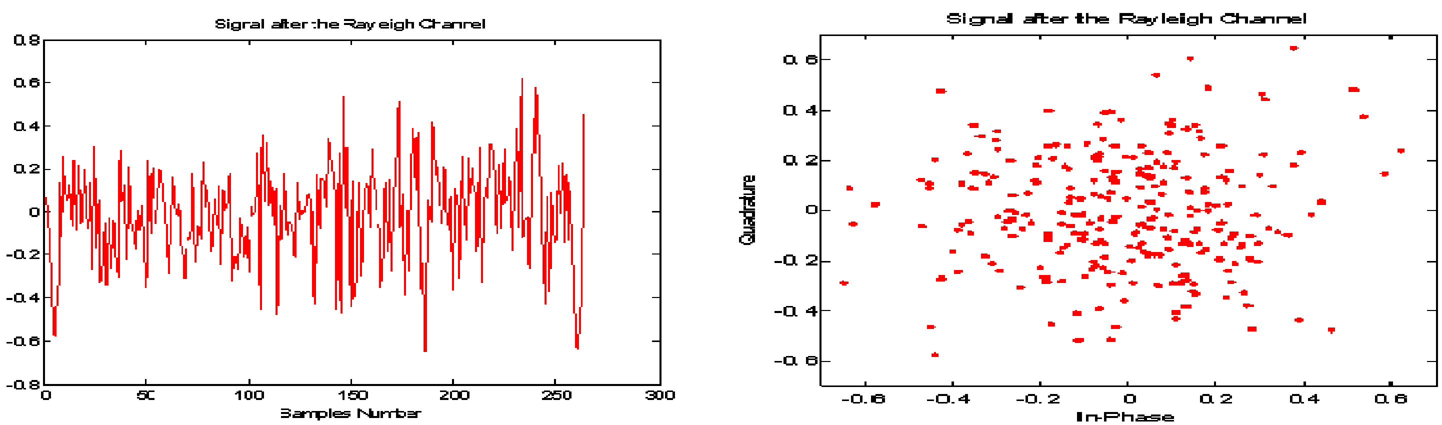


Constelación **32QAM** y Señal en el dominio del tiempo **32 QAM**

Resultados.

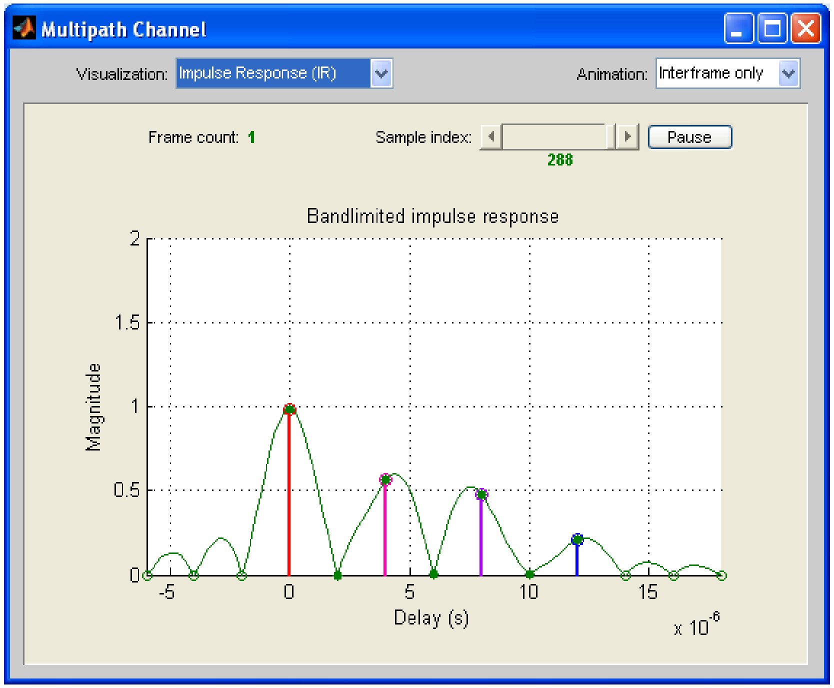


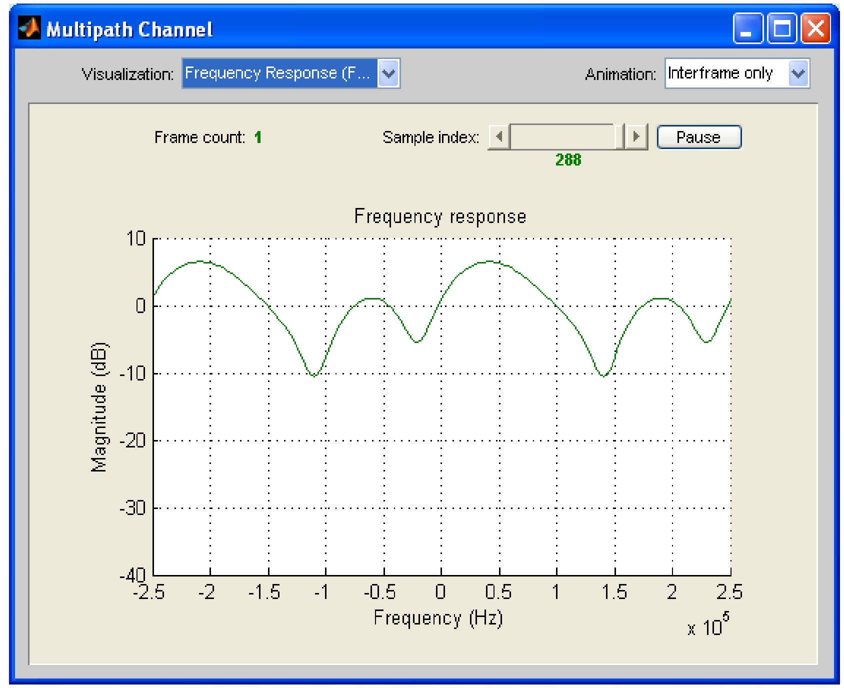
Señal antes de pasar por el canal tipo Rayleigh con Modulación 32QAM



Salida del canal tipo Rayleigh con Modulación 32 QAM

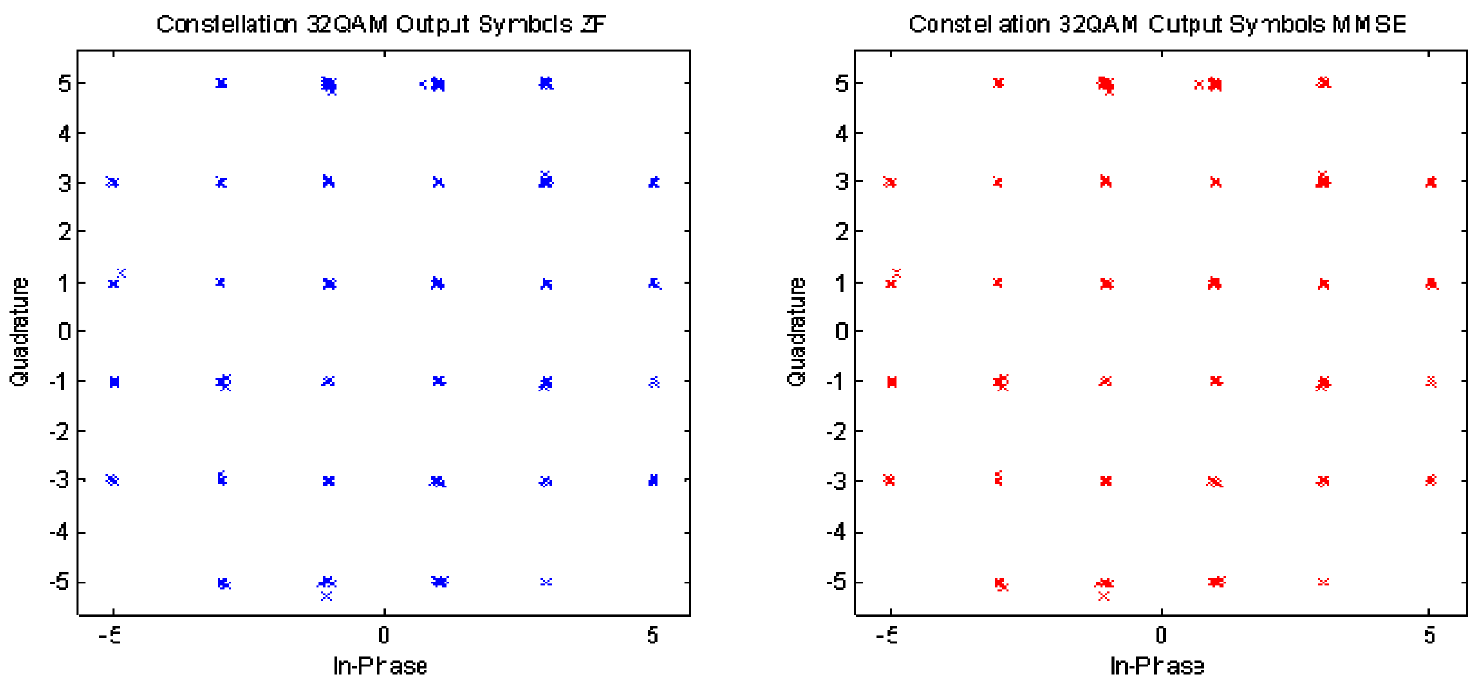
Resultados.



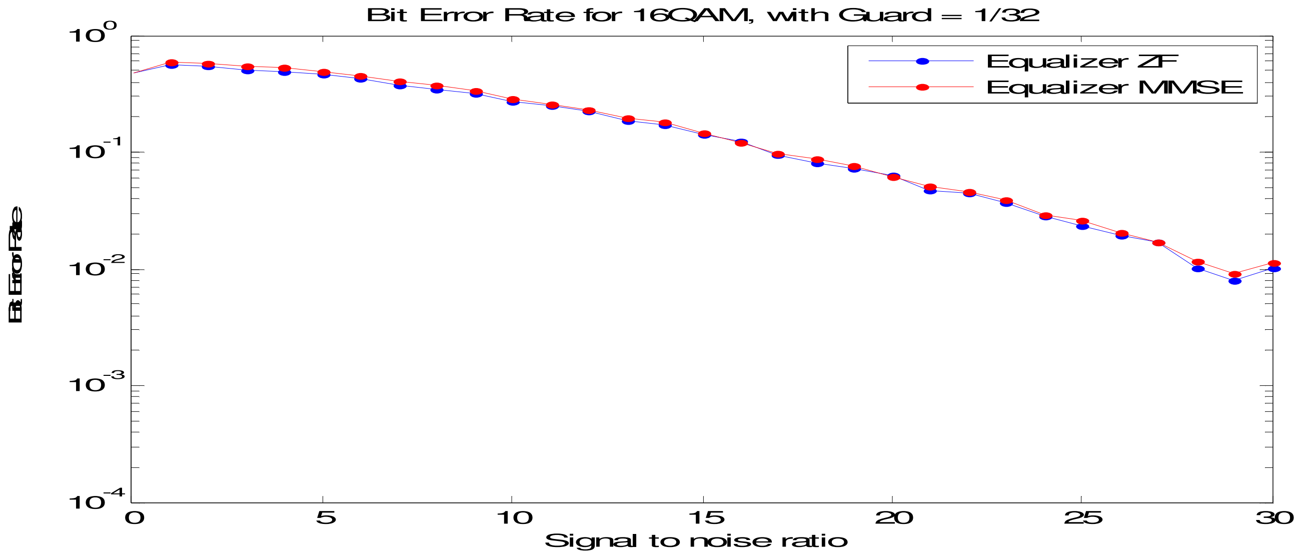


Respuesta al Impulso del Canal y Respuesta en Frecuencia del Canal.

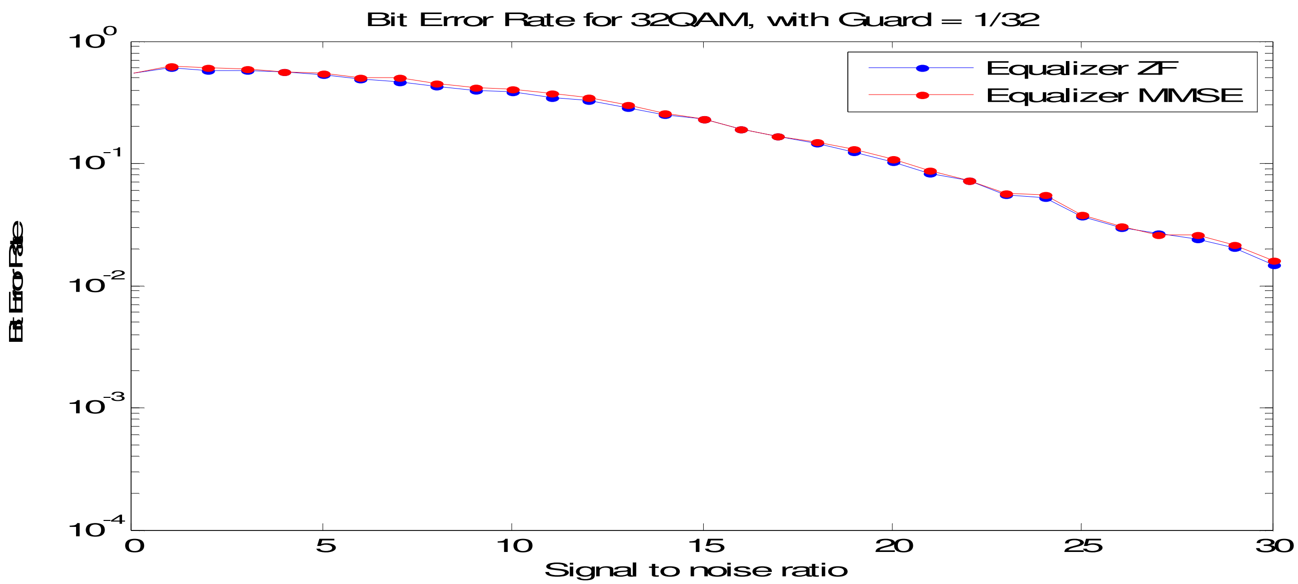
Resultados



Constelación de los símbolos recibidos para el receptor ZF y MMSE utilizando Modulación 32QAM



Tasa de error de bit, para un sistema WiMAX Fijo utilizando modulación **16QAM** e Intervalo de Guarda de un 1/32,



Tasa de error de bit, para un sistema WiMAX Fijo utilizando modulación **32QAM** e Intervalo de Guarda de un 1/32,

Conclusiones

- Como observamos en cada una de estas graficas tanto el ecualizador Zero-Forcing como el MMSE son muy similares en cuanto a los resultados finales sin embargo como se explico en la teoría de ecualización adaptiva, el ecualizador Zero-Forcing no es un ecualizador optimo debido a que engrandece el ruido de la señal por otro lado el MMSE utiliza un algoritmo de estimación del mínimo error cuadrático medio, en el cual el ruido es reducido al mínimo.

Conclusiones

- En el análisis de la tasa de error de bits, se puede concluir que entre más grande sea el prefijo cíclico, la curva del BER tiende a crecer.
- De igual manera se obtuvo un resultado similar a lo anterior en cuanto al BER, cuando se incrementaba el nivel de modulación QAM, la tasa de error de bit tiende a incrementarse

Conclusiones

- Con respecto al canal que se utilizo al simular el sistema WiMAX una señal que pase atreves de este sufrirá un desvanecimiento de frecuencia selectiva debido a que el periodo de muestreo es menor al camino con el mayor retraso. Además como el periodo de muestreo de mi sistema es mucho menor que el periodo de coherencia mi señal se atenuara por desvanecimiento de tipo lento.

Gracias por su atención.

Byron Floreano – Santiago Jácome