Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Volum II
Annexes

TREBALL DE FI DE GRAU

ESTUDI DE TELEGESTIÓ APLICAT AL CONSUM D'ENERGIA ELÈCTRICA

TFG presentat per obtenir el títol de GRAU en
ENGINYERIA ELÈCTRICA
Per **Adriana Puigdevall Ducasi**

Barcelona, 14 d'Octubre de 2016

Tutor: Spartacus Gomáriz Castro
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya (UPC)

Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Annexes

ESTUDI DE TELEGESTIÓ APLICAT AL CONSUM D'ENERGIA ELÈCTRICA

TFG presentat per optar al títol de GRAU en ENGINYERIA ELÈCTRICA
per **Adriana Puigdevall Ducasi**

Barcelona, 14 d'Octubre de 2016

Tutor: Spartacus Gomáriz Castro
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya (UPC)

ÍNDEX ANNEXES

ANNEX A:

CODI

En aquest Annex es mostren els codis sencers tant de la modulació OFDM com de la modulació BPSK, que s'han introduït a la placa per tal de que executi ambdues modulacions. També s'adjunta el codi utilitzat per fer la simulació i extreure'n les gràfiques.

A.1. Codi simulat d'ambdues modulacions

```
import matplotlib.pyplot as plt
import random
import numpy as np
import timeit


"parametres d'entrada"

Fs                = 100e3; "sampling frequency (Hz)"
F0                = 25e3;  "center frequency (Hz)"
Fsym              = Fs/1; "symbol frequency ( bps  )"
Ts                = 1/Fs; "sample duration (s)"
interpRatio       = round(Fs/Fsym); "Interpolation ratio "
Fsym              = Fs / interpRatio; "sumbol frequency "
Tsym              = 1/Fsym; "symbol duration "



NFFT              = 1024; "OFDM FFT size "
K                 = int(2.*round((NFFT-(NFFT/10))/2)); "number of
modulated carriers 10% less of OFDM "
GIR               = 0.125; "GI ratio"
```

```
GI                = round(GIR*NFFT); "GI size in samples @ Fsym "
Nofdm             = NFFT + GI; " OFDM symbol size in samples @ Fsym"
NdataSymbPerFrame = 1; "Number of data symbol per frame"
Lguard            = 2; " Guard size in samples @ Fs"
dataSeed          = 1234; "data seed"
NdataBitsPerFrame = NdataSymbPerFrame*K*2; " Number of data bits per
Frame"
constellation     = 'QPSK'; " constellation mapping"
pilotSeed         = 456; " pilots seed"
rolloff           = 0.1; " roll-off factor"
bandwidth_OFDM        = Fsym*(1+rolloff); " Bandwidth "
T                 = NFFT*Tsym; "OFDM symbol duration (s) "
Tg                = GI*Tsym; "GI duration (s)"
Lf                = (NdataSymbPerFrame+1)*Nofdm*interpRatio; "Frame
length in samples @ Fs"


"Create data cells"


aux1=np.exp(1j*np.pi/4*np.array([1,3,5,7]))
aux2=np.random.choice(aux1,NdataBitsPerFrame/2)


b = np.zeros(((NFFT-K)/2,NdataSymbPerFrame))
c = np.reshape(aux2,(K,NdataSymbPerFrame))
Xd0 = np.concatenate((b,c,b), axis=0)



Z = np.zeros(((interpRatio-1)*(NFFT/2),NdataSymbPerFrame));


d = np.concatenate((Z,Xd0,Z), axis=0)
shift = np.fft.fftshift(d, -1)


Sd1 = np.fft.ifft(shift, axis=-1 )


Sd = (np.sqrt(NFFT))*(Sd1) #OFDM mod


"concatenate pilot symbol and datsymbols"
S0 = np.matrix(Sd)


aux3= S0[0:GI]
S = np.concatenate ((aux3,S0), axis=0)
s=S
```

```
guard = np.zeros((Lguard*interpRatio,1));
s_bb = np.concatenate((guard,s,guard), axis= 0)

"Up conversion"
g = np.shape(s_bb)
h=g[0]
t = np.arange(0,h,1)*Ts
t4 =  np.arange(0,1024,1)
s_pb_OFDM = np.diag(np.real (s_bb*np.exp(1j*2*np.pi*F0*t)))

"Plot data"

plt.plot(t[1:400], s_pb_OFDM[1:400], linewidth=2.0)
#plt.plot(t, s_pb_OFDM, linewidth=2.0)

print(NdataBitsPerFrame,'OFDM signal transmitting %d bits using a BW= %d
Hz ','bandwidth_OFDM\n')
plt.xlabel("Time (s)", fontsize = 15)
plt.ylabel("Amplitude", fontsize = 15)
plt.title("OFDM", fontsize = 20)
plt.savefig('OFDM', dpi=1000)
plt.show()

plt.plot(np.real(np.random.choice(aux1,NdataBitsPerFrame/2)),np.imag(np.ra
ndom.choice(aux1,NdataBitsPerFrame/2)),'bo')

print(NdataBitsPerFrame,'OFDM signal transmitting %d bits using a BW= %d
Hz ','bandwidth_OFDM\n')
plt.xlabel("Q", fontsize = 15)
plt.ylabel("I", fontsize = 15)
plt.title("CONSTELLATION QPSK", fontsize = 20)
plt.savefig('CONSTELLATION QPSK.png', dpi=1000)
plt.show()

###BPSK

lista = []
```

```
for n in range(0, int(NdataBitsPerFrame) ):
    lista.append(random.randint(0, 1))

t1= np.arange(0,(1156/Fsym),(1/Fs))

k= np.shape(lista)
s_bb_BPSK= []

for idx in range(0, k[0]):

    bitsignal=np.exp(1j*2*np.pi*F0*t1+lista[idx]);

s_bb_BPSK=bitsignal

"Up conversion"
l = np.shape(s_bb_BPSK)
m= l[0]
t2= np.arange(0,m,1)*Ts

t = np.arange(0,h,1)*5*Ts
s_pb_BPSK = np.real (s_bb_BPSK*np.exp(1j*2*np.pi*F0*t));



"Plot data"

plt.plot(t[1:50], np.real(s_pb_BPSK[1:50]), linewidth=2.0)
#plt.plot(t, np.real(s_pb_BPSK), linewidth=2.0)

print(NdataBitsPerFrame,'BPSK signal transmitting %d bits using a BW= %d
Hz ','bandwidth_BPSK\n')
plt.xlabel("Time (s)", fontsize = 15)
plt.ylabel("Amplitude", fontsize = 15)
plt.title("BPSK", fontsize = 20)
plt.savefig('BPSK', dpi=1000)
```

```
plt.show()


plt.plot(np.real(lista),np.imag(lista), 'bo')
plt.savefig('BPSK Constellation', dpi=1000)


timeit.timeit()
```

A.2. Codi OFDM implementat a la placa

```
import numpy as np

import timeit


"parametres d'entrada"

Fs              = 100e3; "sampling frequency (Hz)"
F0              = 25e3;  "center frequency (Hz)"
Fsym            = Fs/1; "symbol frequency ( bps  )"
Ts              = 1/Fs; "sample duration (s)"
interpRatio     = round(Fs/Fsym); "Interpolation ratio "
Fsym            = Fs / interpRatio; "sumbol frequency "
Tsym            = 1/Fsym; "symbol duration "



NFFT            = 1024; "OFDM FFT size "
K               = int(2.*round((NFFT-(NFFT/10))/2)); "number of
modulated carriers 10% less of OFDM "
GIR             = 0.125; "GI ratio"
GI              = round(GIR*NFFT); "GI size in samples @ Fsym "
Nofdm           = NFFT + GI; " OFDM symbol size in samples @ Fsym"
NdataSymbPerFrame = 1; "Number of data symbol per frame"
Lguard          = 2; " Guard size in samples @ Fs"
dataSeed        = 1234; "data seed"
NdataBitsPerFrame = NdataSymbPerFrame*K*2; " Number of data bits per
Frame"
constellation   = 'QPSK'; " constellation mapping"
pilotSeed       = 456; " pilots seed"
rolloff         = 0.1; " roll-off factor"
bandwidth_OFDM       = Fsym*(1+rolloff); " Bandwidth "
T               = NFFT*Tsym; "OFDM symbol duration (s) "
Tg              = GI*Tsym; "GI duration (s)"
Lf              = (NdataSymbPerFrame+1)*Nofdm*interpRatio; "Frame
length in samples @ Fs"


aux1=np.exp(1j*np.pi/4*np.array([1,3,5,7]))
aux2=np.random.choice(aux1,NdataBitsPerFrame/2)
```

```
b = np.zeros(((NFFT-K)/2,NdataSymbPerFrame))
c = np.reshape(aux2,(K,NdataSymbPerFrame))
Xd0 = np.concatenate((b,c,b), axis=0)

Z = np.zeros(((interpRatio-1)*(NFFT/2),NdataSymbPerFrame));
d = np.concatenate((Z,Xd0,Z), axis=0)


shift = np.fft.fftshift(d, -1)
Sd1 = np.fft.ifft(shift, axis=-1 )
Sd = (np.sqrt(NFFT))*(Sd1) #OFDM mod

"concatenate pilot symbol and datsymbols"

S0 = np.matrix(Sd)
#S0  = Sd;
aux3= S0[0:GI]
S = np.concatenate ((aux3,S0), axis=0)


s=S

guard = np.zeros((Lguard*interpRatio,1));
s_bb = np.concatenate((guard,s,guard), axis= 0)



# Up conversion
g = np.shape(s_bb)
h=g[0]
t = np.arange(0,h,1)*Ts
t4 =  np.arange(0,1024,1)
s_pb_OFDM = np.diag(np.real (s_bb*np.exp(1j*2*np.pi*F0*t)))

timeit.timeit()

print(timeit.timeit)
```

A.3. Codi BPSK implementat a la placa

```
import random

import numpy as np

import timeit1


"parametres d'entrada"

Fs              = 100e3; "sampling frequency (Hz)"
```

```
F0              = 25e3;  "center frequency (Hz)"
Fsym            = Fs/1; "symbol frequency ( bps  )"
Ts              = 1/Fs; "sample duration (s)"
interpRatio     = round(Fs/Fsym); "Interpolation ratio "
Fsym            = Fs / interpRatio; "sumbol frequency "
Tsym            = 1/Fsym; "symbol duration "



NFFT            = 1024; "OFDM FFT size "
K               = int(2.*round((NFFT-(NFFT/10))/2)); "number of
modulated carriers 10% less of OFDM "
GIR             = 0.125; "GI ratio"
GI              = round(GIR*NFFT); "GI size in samples @ Fsym "
Nofdm           = NFFT + GI; " OFDM symbol size in samples @ Fsym"
NdataSymbPerFrame = 1; "Number of data symbol per frame"
Lguard          = 2; " Guard size in samples @ Fs"
dataSeed        = 1234; "data seed"
NdataBitsPerFrame = NdataSymbPerFrame*K*2; " Number of data bits per
Frame"
constellation   = 'QPSK'; " constellation mapping"
pilotSeed       = 456; " pilots seed"
rolloff         = 0.1; " roll-off factor"
bandwidth_OFDM       = Fsym*(1+rolloff); " Bandwidth "
T               = NFFT*Tsym; "OFDM symbol duration (s) "
Tg              = GI*Tsym; "GI duration (s)"
Lf              = (NdataSymbPerFrame+1)*Nofdm*interpRatio; "Frame
length in samples @ Fs"
```

```
lista = []

for n in range(0, int(NdataBitsPerFrame) ):
    lista.append(random.randint(0, 1))



t1= np.arange(0,(1156/Fsym),(1/Fs))
k= np.shape(lista)
s_bb_BPSK= []

for idx in range(0, k[0]):

    bitsignal=np.exp(1j*2*np.pi*F0*t1+lista[idx]);

s_bb_BPSK=bitsignal
```

```
l = np.shape(s_bb_BPSK)
m= l[0]
t2= np.arange(0,m,1)*Ts

t = np.arange(0,h,1)*5*Ts

s_pb_BPSK = np.real (s_bb_BPSK*np.exp(1j*2*np.pi*F0*t));

timeit.timeit()
print(timeit.timeit)
```