

1º Semestre 16/17 SI

Comunicação e Processamento de Sinais

**Relatório do 3º Trabalho Prático**

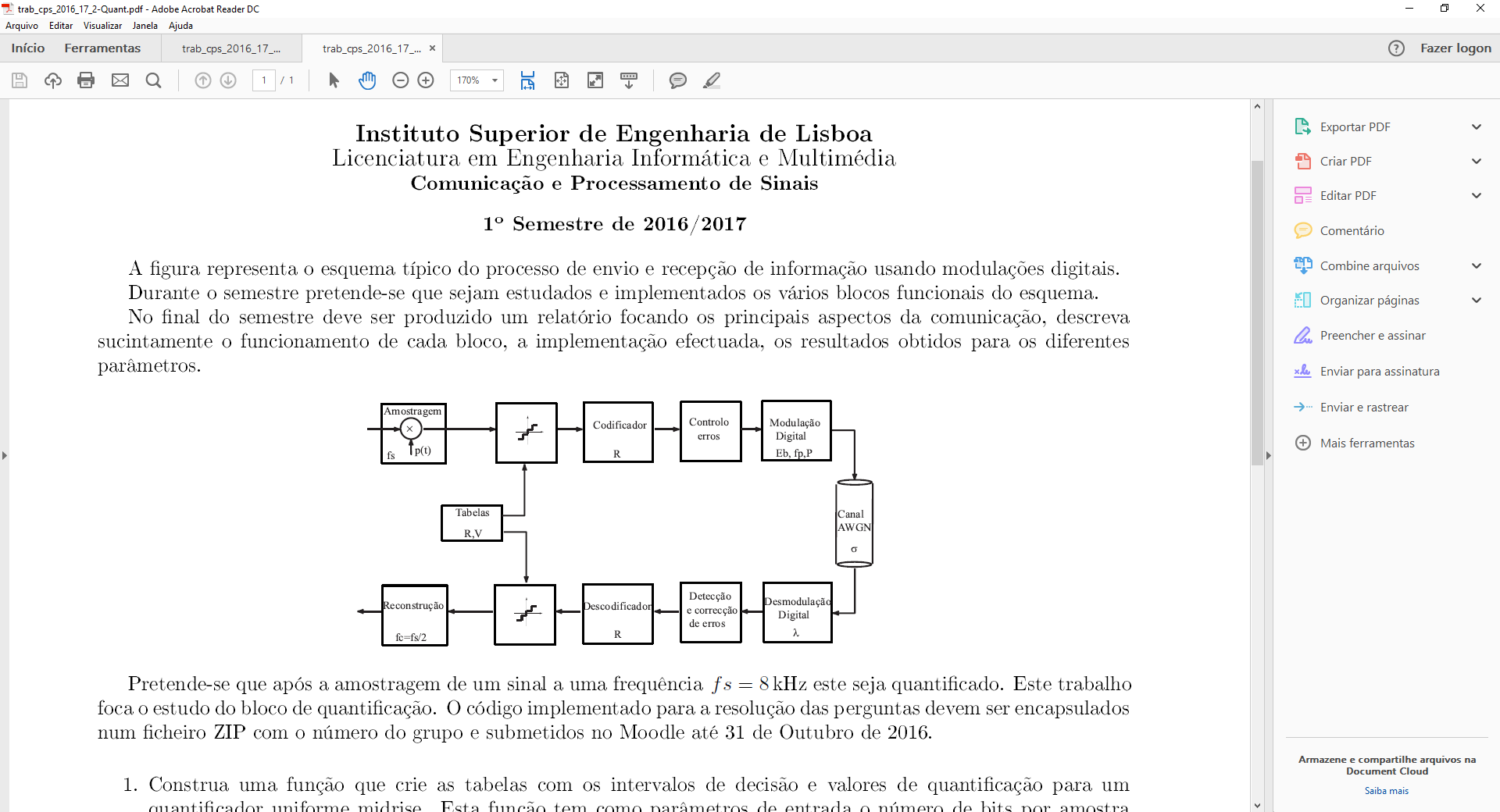
Eng. Pedro Fazenda

**Trabalho realizado por:**

Diogo Fernandes nº39205

Rui Santos nº39286

**Introdução**



Este terceiro trabalho foca-se nos blocos de Controlo e Correção de erros e suas operações inversas, iremos utilizar código desenvolvido no primeiro e segundo trabalho para executar testes que pretendemos neste trabalho.

Exercícios

1. Construa uma função que dado um array de N bits, para cada 4 bits gere 7 bits, sendo 3 bits resultado do código de Hamming(7,4).

def hamming(message):  
 currentMessage = message  
 #verificacao do tamanho da mensagem  
 resto = len(currentMessage)%messageBits  
 #extenca do array inicializada  
 if(not (resto == 0)):  
 finalSize = len(currentMessage)+(messageBits-resto)  
 currentMessage = np.zeros(finalSize)  
 #preenchimento binario correto  
 for i in range(len(message)):  
 currentMessage[i] = message[i]  
 #matriz identidade  
 identity = np.eye(messageBits)  
 #matriz geradora  
 G = np.hstack((identity,P))  
 #array de bits final  
 size = len(currentMessage)+((len(currentMessage)/messageBits)\*controlBits)  
 finalBits = np.zeros(int(size))  
 count = 0  
 for i in range(0,len(currentMessage),messageBits):  
 currentSlice = currentMessage[i:i+messageBits]  
 #bits de paridade resultantes  
 addControl = np.mod(np.dot(currentSlice,G),2)  
 for x in range(0,len(addControl)):  
 finalBits[count] = addControl[x]  
 count += 1

return finalBits.astype('int16')

**2.**Construa uma função que dado um array de M bits para cada 7 bits detete e corrige possiveis erros, sendo o resultado 4 bits de mensagem.

def sindrome(message):  
 #possiveis sindromas  
 sindromeTable = np.array([[0,1,1],[1,1,0],[1,0,1],[1,1,1],[1,0,0],[0,1,0],[0,0,1]])  
 #percorre o sinal  
 for i in range(0, len(message), totalBits):  
 currentMessage = message[i:i+totalBits]  
 #resultado do sindroma multiplicado por matriz de teste de paridade  
 sindromeResult = np.mod(np.dot(currentMessage,sindromeTable),2)  
 size = (len(message)/totalBits)\*messageBits  
 correctedMessage = np.zeros(int(size))  
 index = 0  
 count = 0  
 errors = 0  
 for x in range(0,len(sindromeTable)):  
 if(sum(sindromeTable[x]==sindromeResult)==controlBits):  
 message[i+count] = (message[i+count]+1)%2  
 #apenas corrige um erro  
 break  
 else:  
 count+=1  
 if(sum(sindromeResult)!=0):  
 errors += 1  
 count =0  
 correctedMessage[index:index+messageBits] = message[i:i+messageBits]  
 index += messageBits

return errors, correctedMessage.astype('int16')

**3.**Com as funções anteriores e admitindo que os restantes blocos do emissor, canal e receptor possam ser simulados pela expressão:

Onde BERt, contém o valor pretendido para o bit error rate. Meça a SNR na recepção, o BER antes e após a correção de erros, para diferentes valores de BERt.