# DTSim

## Data Transmission Simulator

## 1. Introduction

When it comes to transmitting a message, there are two main issues that concern both the transmitter and the receiver:  The reliability and the security.

### 1.1. Reliability

We want the message to arrive to the destiny without any error so when it is received after the channel the receiver looks for any possible error. If there is any, it tries to fix it (Forward error correction FEC), and if it is not possible it asks for the repetition of the transmission (Automatic Repeat reQuest ARQ).

The technique used to increment the reliability of the transmission is known as channel coding, which consists on adding some redundancy to the message in order to be able to use that to know whether the message has been received correctly or not.

### 1.2. Security

If, by any chance, an undesired third person manages to get the message we don't want him or her to be able to know its contain. What we do is: Instead of transmitting it clearly, we modify the message in a secret way that only the transmitter and the receiver know, so if an attacker gets the message he will only read what seems to be random symbols. This technique is known as cryptography.

The Data Transmission Simulator (DTSym from now) is, as its name suggest, a simulator with which the user can load or write a message and navigate through the whole process of encryption, channel coding, transmission, channel decoding and decryption.

# 2. Theoretical Background

There are several ways of channel coding and encryption but in this project, I have focused in one of each. For the channel coding part I have used a convolutional code and for the cryptography part I have used the Vigenère algorithm.

## 2.1. Convolutional Codes

First of all, let's make a brief introduction on the math's that explain the channel coding.

Having a user word X, with a size of *k* bits, a linear function is applied on it to get the code word Y, with a size of *n* bits, being n > k.

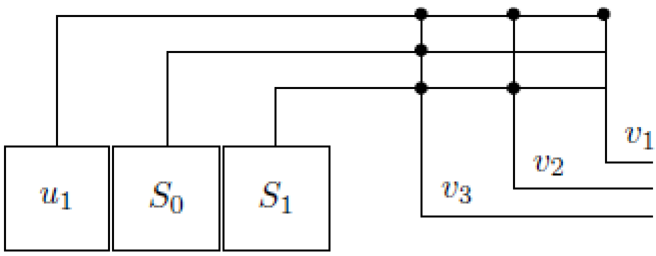
$$Y = f(X)$$

Convolutional codes are lineal codes but, unlike block codes, they have memory, which means that the code word bits do not only depend on the corresponding user word bit but also on the previous ones.

$$y(i) = f(x(i), x(i-1), x(i-2), \dots)$$

Another characteristic is that for every *n* input bits, *m* output bits are generated.

An example of convolutional code is this one with n = 1 and m = 3:




It starts with S0 and S1 both 0, and u1 = x(1), and every time the outputs are computed the next input enters the function. The second time will be u1 = x(2), S0 = x(1) and S1 = 0, and so on until u1 = 0, S0 = 0 and S1 = x(k).

In this case the functions would be:

$$v_1 = u_1$$
$$v_2 = u_1 + S_1$$

$$v_3 = u_1 + S_1 + S_0$$

The decoding of the convolutional code consists on using the Viterbi Algorithm, which will be explained in the final part of the work.

### 2.2. Vigenère algorithm

The Vigenère algorithm is a method of encrypting a text by using the method of polyalphabetic substitution. It consists on creating an alphanumeric key, known by the receiver and the transmitter, which is added word by word periodically to the clear text.

The mathematical function for each symbol of the cryptogram is:

$$C(i) = \big(M(i) + K(i)\big) mod(N)$$

Where M is the clear message and K is the key.

N is the size of the alphanumeric group you are using, for example, for the alphabet it will be 26. We will be using the extended ASCII code so N = 256.

This is an example of Vigenère algorithm where the clear message is “Attack at dawn” and the Key is “Lemon”, using N = 26.

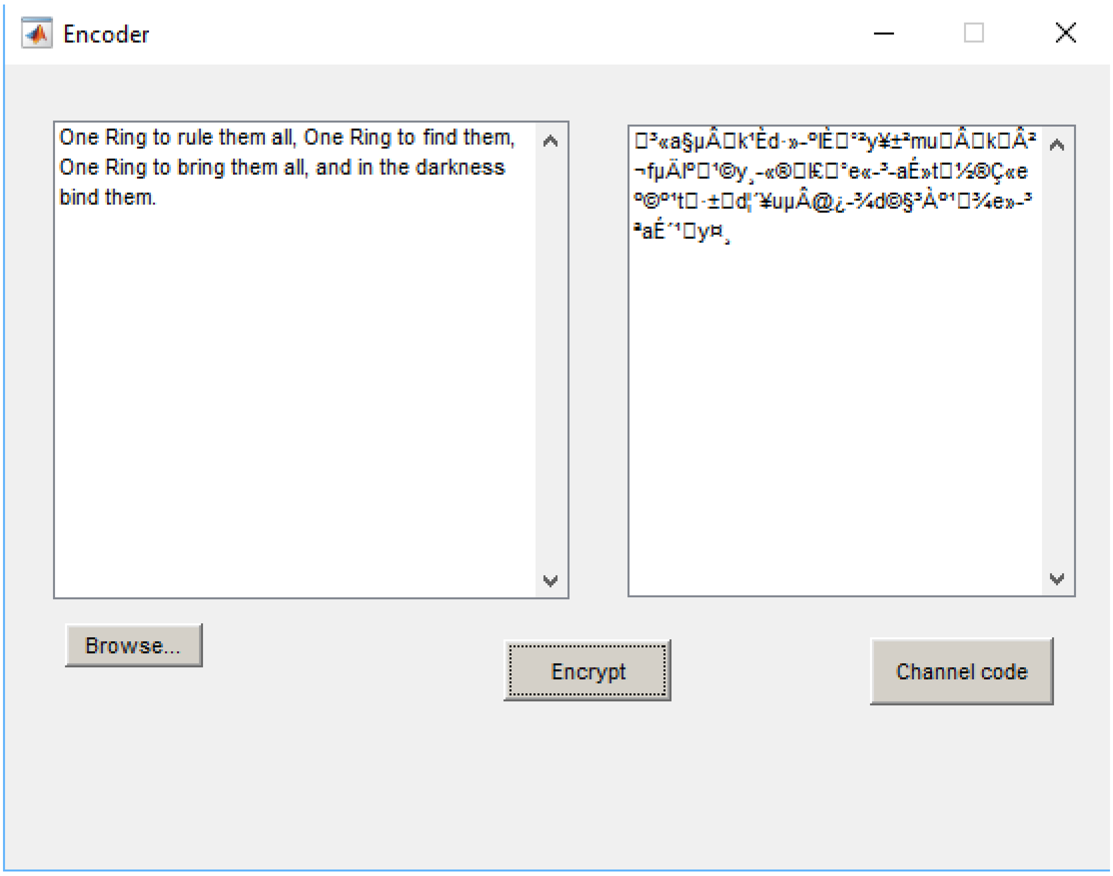
```
Plaintext:  ATTACKATDAWN

Key:        LEMONLEMONLE

Ciphertext: LXFOPVEFRNHR
```

# 3. Work done

I have already done the three main GUI figures (which need to be put on a “cooler” way):




**<u>Encoder:</u>**

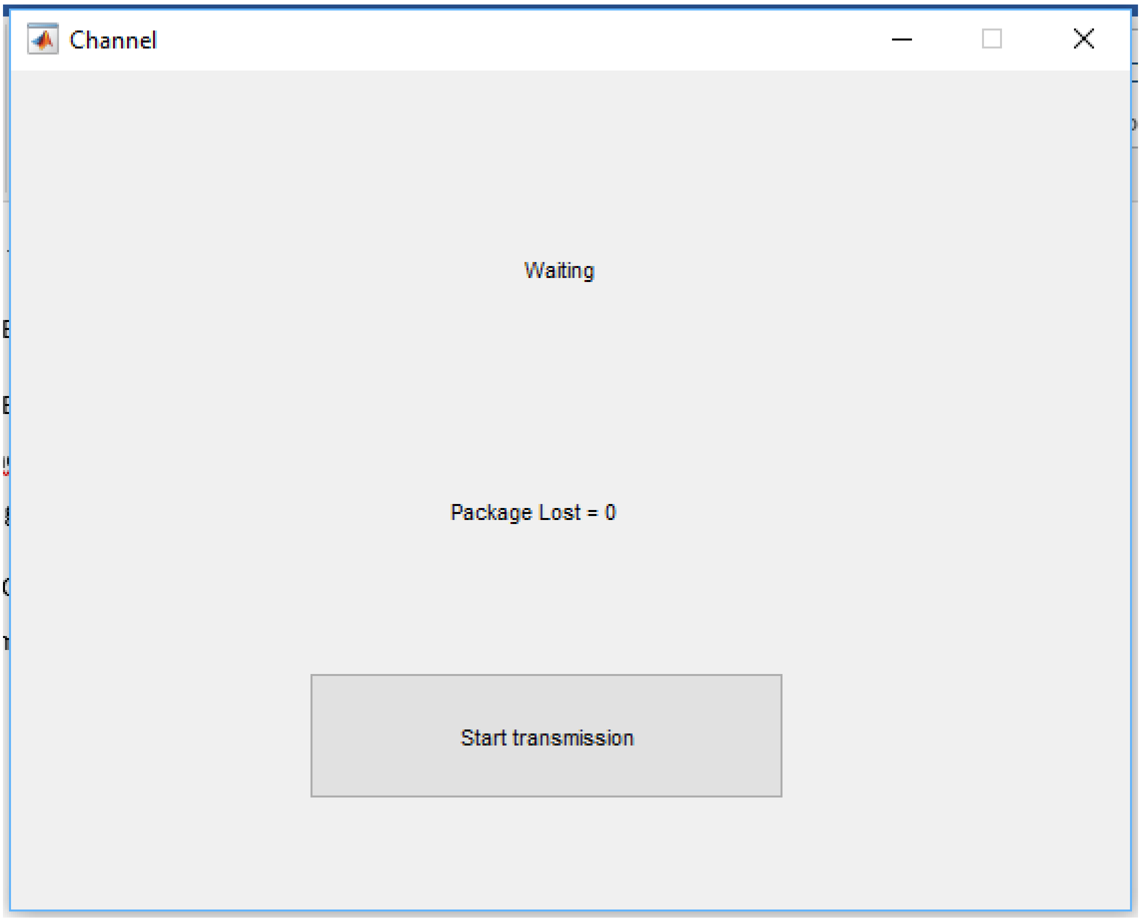
It consists on:

-Two text displays, the left one for the clear message and the right one for the cryptogram.

-The Browse... button with which you can load a .txt file from your computer.

-The Encrypt button which takes the left display text, calls the created function “Vignere_encipherment” with it as an input parameter, and gets the cryptogram, writing it in the right display.

-The Channel code button which opens the Channel GUI figure with the cryptogram as input parameter.



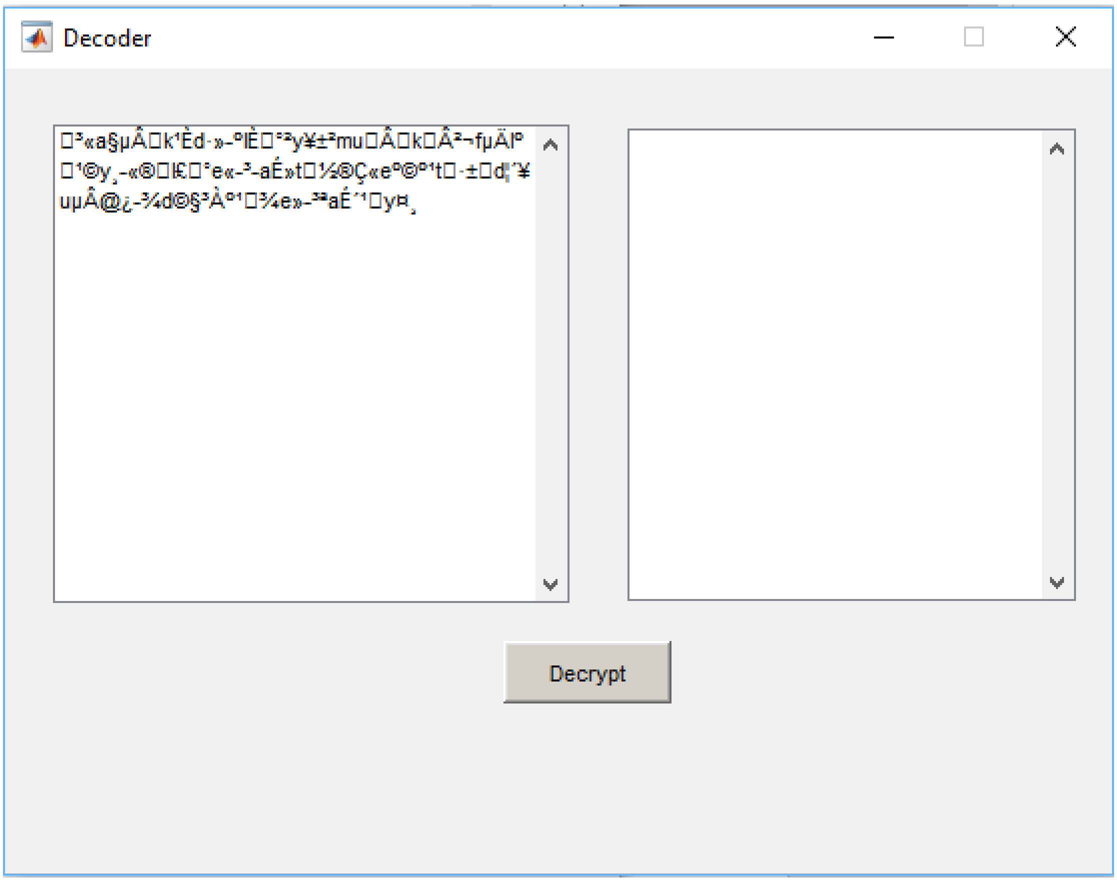

**Channel:**

It consists on:

-A text display that varies between “Waiting”, “Transmitting… (percentage of transmission)%” and “Transmission Completed”.

-A text display with the number of packages that have need a retransmission.

-A single button that codes the channel using the convolutional code described before, then using a ‘for’ loop it transmits the packages of size 1 char and retransmits them if needed. Once the transmission have finished, it calls the Decoder GUI figure with the received text as a parameter.




**Decoder**

It consists on:

-Two text displays, the left one for the cryptogram and the right one for the clear message.

-The Decrypt button which calls the “Vigenre_decipherment” function to get the clear message from the cryptogram.

# 4. Future Work

My plans for the future work are:

-Make a deployable figure in which the convolutional code can be chosen with some check boxes in the connections.

-Think about a way to generate a random key and transmit it to the receiver.

-Program the decodifying Viterbi algorithm for the convolutional code.

-Put the figures in a cooler way.

-Optimize as possible the codes and see if I can make some of the ‘for’ without loops.

# 5. CODE

## 5.1. GUI Figures

**Encoder**

```
function varargout = Encoder(varargin)
% ENCODER MATLAB code for Encoder.fig
%      ENCODER, by itself, creates a new ENCODER or raises the 
existing
%      singleton*.
%
%      H = ENCODER returns the handle to a new ENCODER or the handle 
to
%      the existing singleton*.
%
%      ENCODER('CALLBACK',hObject,eventData,handles,...) calls the 
local
%      function named CALLBACK in ENCODER.M with the given input 
arguments.
%
%      ENCODER('Property','Value',...) creates a new ENCODER or raises 
the
%      existing singleton*.  Starting from the left, property value 
pairs are
%      applied to the GUI before Encoder_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property 
application
%      stop.  All inputs are passed to Encoder_OpeningFcn via 
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows 
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Encoder

% Last Modified by GUIDE v2.5 08-Dec-2016 19:57:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Encoder_OpeningFcn, ...
                   'gui_OutputFcn',  @Encoder_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Encoder is made visible.
function Encoder_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Encoder (see VARARGIN)

% Choose default command line output for Encoder
handles.output = hObject;
handles.text_clear = '';
handles.text_coded = '';
handles.key = 'DEFAULT KEY';

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Encoder wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Encoder_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```matlab
% --- Executes on button press in button_Channel_code.
function button_Channel_code_Callback(hObject, eventdata, handles)
% hObject    handle to button_Channel_code (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Channel(handles.text_coded);



function message_Callback(hObject, eventdata, handles)
% hObject    handle to message (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.text_clear = get(hObject,'String');
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of message as text
%        str2double(get(hObject,'String')) returns contents of message
as a double


% --- Executes during object creation, after setting all properties.
function message_CreateFcn(hObject, eventdata, handles)
% hObject    handle to message (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in button_Browse.
function button_Browse_Callback(hObject, eventdata, handles)
% hObject    handle to button_Browse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename, pathname] = uigetfile({'*.txt'},'File Selector');
%opens a browser tab to select a .txt file and saves its name and
pathname.
handles.text_clear = fileread(strcat(pathname,filename));
%loads the browsed file
set(handles.message,'String',handles.text_clear);
guidata(hObject, handles);


% --- Executes on button press in button_Encrypt.
function button_Encrypt_Callback(hObject, eventdata, handles)
% hObject    handle to button_Encrypt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
handles.text_coded =
Vignere_encipherment(handles.text_clear,handles.key);
set(handles.cryptogram,'String',handles.text_coded);
guidata(hObject, handles);
```

**Decoder**

```
function varargout = Decoder(varargin)
% DECODER MATLAB code for Decoder.fig
%      DECODER, by itself, creates a new DECODER or raises the
existing
%      singleton*.
%
%      H = DECODER returns the handle to a new DECODER or the handle
to
%      the existing singleton*.
%
%      DECODER('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in DECODER.M with the given input
arguments.
%
%      DECODER('Property','Value',...) creates a new DECODER or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Decoder_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Decoder_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Decoder

% Last Modified by GUIDE v2.5 10-Dec-2016 20:41:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```matlab
                   'gui_OpeningFcn', @Decoder_OpeningFcn, ...
                   'gui_OutputFcn',  @Decoder_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Decoder is made visible.
function Decoder_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Decoder (see VARARGIN)

% Choose default command line output for Decoder
handles.output = hObject;
handles.text_clear = '';
handles.text_coded = strjoin(varargin(1));
set(handles.cryptogram,'String',strjoin(varargin(1)));
handles.key = 'DEFAULT KEY';

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Decoder wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Decoder_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function message_Callback(hObject, eventdata, handles)
% hObject    handle to message (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.text_clear = get(hObject,'String');
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of message as text
%        str2double(get(hObject,'String')) returns contents of message
as a double


% --- Executes during object creation, after setting all properties.
function message_CreateFcn(hObject, eventdata, handles)
% hObject    handle to message (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in button_Decrypt.
function button_Decrypt_Callback(hObject, eventdata, handles)
% hObject    handle to button_Decrypt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.text_clear =
Vignere_decipherment(handles.text_coded,handles.key);
set(handles.message,'String',handles.text_clear);
guidata(hObject, handles);
```

**Channel**

```
function varargout = Channel(varargin)
% CHANNEL MATLAB code for Channel.fig
%      CHANNEL, by itself, creates a new CHANNEL or raises the
existing
%      singleton*.
%
%      H = CHANNEL returns the handle to a new CHANNEL or the handle
to
%      the existing singleton*.
%
%      CHANNEL('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CHANNEL.M with the given input
arguments.
%
%      CHANNEL('Property','Value',...) creates a new CHANNEL or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Channel_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Channel_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Channel

% Last Modified by GUIDE v2.5 08-Dec-2016 20:05:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Channel_OpeningFcn, ...
                   'gui_OutputFcn',  @Channel_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Channel is made visible.
function Channel_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Channel (see VARARGIN)

% Choose default command line output for Channel
handles.output = hObject;
handles.text_send = strjoin(varargin(1));
handles.text_received = repmat(' ',1,length(strjoin(varargin(1))));
handles.package_lost = 0;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Channel wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Channel_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in button_Start_transmission.
function button_Start_transmission_Callback(hObject, eventdata,
handles)
% hObject    handle to button_Start_transmission (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
for i = 1:1:length(handles.text_send)

    package = Channel_coder(handles.text_send(i));
    %we generate the package, each package has an 8-bit length.

    ACK = false;
    while(ACK==false)
        ACK = AWGN_transmission(package);
        %We transmit it through a AWGN channel
        handles.package_lost = handles.package_lost + ~ACK ;
        guidata(hObject, handles);
        set(handles.text_package_lost,'String',['Package lost = '
num2str(handles.package_lost)])
        %If ACK == 0 it means that the package has been lost and we
        %transmit it again.
    end

%                             TBC
%
% For the final delivery I will program the Viterbi algorithm in order
to
% get the transmitted letter from the received package. Until I do
that I
% will just get the transmitted symbol as the received.
```

```
handles.text_received(i) = handles.text_send(i); %THIS IS JUST
PROVISIONAL
set(handles.text_state,'String',['Transmiting...     '
num2str(i/length(handles.text_send)*100) '%']);
guidata(hObject, handles);

pause(100e-3); %THIS IS PROVISIONAL, JUST TO SEE THE TEXT CHANGING
WITH SHORT EXAMPLES OF .TXT

end
set(handles.text_state,'String','Transmission finished');
guidata(hObject, handles);
Decoder(handles.text_received);
```

### 5.2. Functions

**Vignere_encipherment**

```
function [ cryptogram ] = Vignere_encipherment( message, key )
%Vignere_encipherment    Cryptography using Vigenère algorithm.
%
%Vignere_enchiperment(message, key) encrypts the string message with
the
%   Vignère algorithm using key. It returns the resulting cryptogram.

if (length(key)>length(message))
    key = key(1:length(message));
end
str = message;
if (mod(length(message),length(key))~=0)
    str = [message, char(95*ones(1,length(key)-
mod(length(message),length(key))))];
    %I use the 95 value which in the ASCII table corresponds to '_'
because the
    %space caused me some problems with the key
end
K = repmat(key,1,length(str)/length(key));

cryptogram = char(mod(str + K,256));
end
```

**Vignere_decipherment**

```
function [ message ] = Vignere_decipherment( cryptogram, key )
%Vignere_decipherment     Decrypts using Vigenère algorithm.
%
%Vignere_decipherment(cryptogram, key) decrypts the string cryptogram
with the
%   Vignère algorithm using key. It returns the resulting clear
message.

if (length(key)>length(cryptogram))
    key = key(1:length(cryptogram));
end
str = cryptogram;
if (mod(length(cryptogram),length(key))~=0)
    str = [cryptogram, char(95*ones(1,length(key)-
mod(length(cryptogram),length(key))))];
    %I use the 95 value which in the ASCII table corresponds to '_'
because the
    %space caused me some problems with the key
end
K = repmat(key,1,length(str)/length(key));

message = char(mod(str - K,256));
end
```

**Channel_coder**

```
function [ y ] = Channel_coder( letter )
%Channel_coder    Codifies the channel using a convolutional code
%
%Channel_coder(letter) transforms the letter into an 8 bits array and
%codifies it using a convolutional code. Returns y where each column
is the
%output vector of each of the bits of the letter.

x = flip(de2bi(1*letter,8));
%converts de letter into a decimal with 8 bits, then into a binary
number and finaly
%flips it to put in the order we work with.
S = [0 0 0];
%States vector.
a = zeros(3,length(x));
%Output vector
for i = 1:1:length(x)

    S = [x(i) S(1:2)];
    a(1,i) = S(1);
    a(2,i) = mod(S(1)+S(3),2);
    a(3,i) = mod(S(1)+S(2)+S(3),2);
end
y = a;
end
```

**AWGN_transmission**

```
function [ ACK ] = AWGN_transmission( package )
%AWGN_transmission    Transmission through an AWGN channel
%
%AWGN_transmission(package) simulates an AWGN channel and sends the
package
%through it, if the package is receiver correctly returns 1 and if
not,
%returns 0.

y = package;

z = awgn(y,15); %AWGN noise addition
yr = z>0.5; %Hard decision

ACK = isequal(y,yr);
% display(ACK);

end
```