Signal&Systems CA#3

SMahdiHajiSeyedHossein

810100118

Part 1

در این قسمت پروژه باید ، یک پیام را رمزگذاری کنیم . و در تصویر قرار دهیم و سپس آن را رمزگشایی کنیم .

Part1_1(MapSet)

در این قسمت پروژه باید یک مپست برای هر کاراکتر که شامل حروف a-z و فاصله و علامت تعجب و کاما و دابل کوتیشن و در آخر پیام ها سمیکولن هست تولید کنیم .

چون تعداد کل کاراکتر ها 32 تاست بنابراین با یک عدد باینتری به طول 5 بیت میتوان به هر یک به طور اختصاصی یک عدد باینری ۵ بیتی تخصیص داد .

```
function Mapset=MapSet()
    Mapset=cell(2,32);
    alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";';
    for i = 1:32
        Mapset{1,i} = alphabet(i);
        Mapset{2,i} = dec2bin(i-1, 5);
    end

end
```

	1	2	3	4	5	6	7	8	9	10	11	
1	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'	'k'	'l'
2	'00000'	'00001'	'00010'	'00011'	'00100'	'00101'	'00110'	'00111'	'01000'	'01001'	'01010'	'0101

Part1_2(Coding)

مشابه آنچه در کلاس توضیح داده شد ، وقتی یک پیام را رمزگذاری کردیم ، اگر طول پیام ما n باشه یک رشته باینری به طول 5*n خواهیم داشت .

حالا برای تصویر سیاه سفید هم هر پیکسل از ۰ تا ۲۵۵ میتواند باشد ، بنابراین درواقع هر پیکسل تصویر یک عدد باینری به طول ۸ بیت هست .

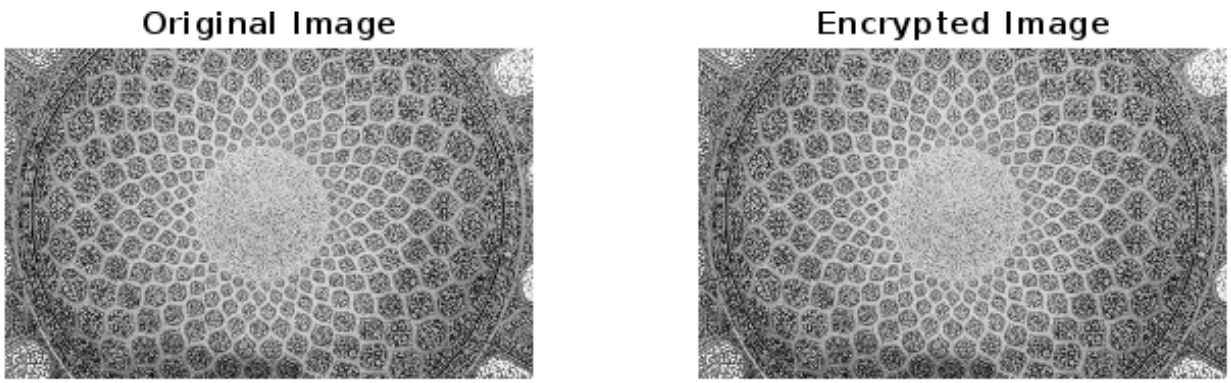
برای اینکه خیلی تغییری در تصویر ایجاد نشود باید بیت آخر هر پیکسل تصویر را به یک بیت از رشته باینری به طول 5*n اختصاص دهیم . (مشابه آنچه که در کلاس مطرح شد)

```
2
3   function EncodedPic = coding(message , pic , mapset)
4
5       [a , b ] = size(pic);
6       if length(message) > a*b
7           error("Cant coded");
8
9       message_binary=cell(1,length(message));
10
11      for i = 1:length(message)
12          current_char=message(i);
13          index=strcmp(current_char,mapset(1,:));
14          message_binary(i) = mapset(2,index);
15      end
16
17
18      binarymessage_encoded=cell2mat(message_binary);
19      binarymessage_len=length(binarymessage_encoded);
20      encoded_image =pic;
21
22      for i = 1:binarymessage_len
23          vals=pic(i);
24          valsbin1=dec2bin(vals);
25          valsbin1(end)=binarymessage_encoded(i);
26          encoded_image (i)=bin2dec(valsbin1);
27      end
28      EncodedPic = encoded_image;
29
30  end
31
32
33
34
```

Part1_3(Coding test)

در قسمت از پروژه به تست قسمت coding پروژه میپردازیم .

```
mapset = MapSet();
inputImage = imread("image.jpeg");
inputImage = rgb2gray(inputImage);
encodedImage = coding ('signal;' , inputImage , mapset);


subplot(1,2,1);
imshow(inputImage);
title("Original Image")
subplot(1,2,2);
imshow(encodedImage);
title("Encrypted Image")
```



Part1_4(DeCoding)

در این بخش ابتدا تصویری که به ورودی فانکشن داده ایم را بررسی میکنیم و تا زمانی که به رشته ۵ بیتی ۱۱۱۱۱ نرسیده ایم که به معنی ; است ادامه میدهیم و ۵ تا ۵ تا بررسی میکنیم .

```
2
3   function message=decoding(image , mapset)
4
5
6
7       message_bin = '';
8       for i = 1:size(image,1)
9           pixel = image(i);
10          pixel_bin = dec2bin(pixel);
11          message_bin = [message_bin pixel_bin(end)];
12      end
13
14      message_decoded = '';
15      while length(message_bin) > 4
16          char_bin = message_bin(1:5);
17          message_bin = message_bin(6:end);
18
19          index = find(strcmp(char_bin, mapset(2,:)));
20          char = mapset{1,index};
21
22          if char == ';'
23              break;
24          end
25          message_decoded = [message_decoded char];
26      end
27
28      disp(message_decoded)
29      |
30  end
```

فانکشن دیکودینگ را تست میکنیم :

```
2
3   mapset = MapSet();
4   inputImage = imread("image.jpeg");
5   inputImage = rgb2gray(inputImage);
6   encodedImage = coding ('signal;' , inputImage , mapset);
7
8
9   decoding(encodedImage , mapset);
```

```
Command Window
>> P1_4
signal
>>
```

Part1_5(Q1)

میتوان اگر یک حرف نویز داشت با توجه به معنی جمله آن را حدس بزنیم و مثلا در دیکشنری کلمات موجود در آن زبان چک کنیم که چه کلماتی ممکن است که شبیه ان باشند .

مثلا اگر با نویز کلمه ی signal را به صورت qignal تشخیص دهیم با توجه به دیکشنری ، میتوانیم بفهمیم که مثلا کلمه احتمالا signal بوده است .

اگر هم نویز در پیکسل های بعد از آن پیکسل هایی که رمزگذاری شده اند باشد مشکلی نداریم .

Part 2

در این قسمت باید یک شماره تماس را سنتز و آنالیز کنیم .

Descroption

در قسمت سنتز باید به هریک از کاراکتر های ABCD#*1234567890 یک wave خاصی اختصاص دهیم .

و این wave ها را با چسباندن بهم دیگر به شکل یک صورت تولید کنیم . حالا در بخش Analisys باید این صدای تولید شده را Decode کنیم و بتوانیم کاراکتر های صوت را تشخیص دهیم .

Part 1 (سنتز)

```
1
2   testNumber='1234567890*#ABCD1234567890**##ABCD';
3
4   fs=8000;
5   Ts=1/fs;
6   Ton=0.1;
7   Toff=0.1;
8   t=0:Ts:Ton;
9
10
11  fLow=[697 ,770 ,852 ,941];
12  fUp=[1209 ,1336 ,1477 ,1633];
13
14  silence=zeros(1,size(t,2)-1);
15
```

در بخش بالا یک شماره تستی تعریف کردم سپس با استفاده از تکه کدی که در صورت پروژه بود سیگنال را تشکیل دادم . حالا با توجه به فرکانس های تعریف شده در دکمه های تلفن به هر یک از کاراکتر ها باید یک شماره سطر و یک شماره ستون تخصیص دهیم .

این کار را با استفاده از الگوریتم زیر انجام میدهیم که به هریک از کاراکتر ها یک ستون و سطر اختصاص میدهیم .

```
for n=1:length(testNumber)

    switch testNumber(n)
        case 'A'
            row=1;
            column=4;
        case 'B'
            row=2;
            column=4;
        case 'C'
            row=3;
            column=4;
        case 'D'
            row=4;
            column=4;
        case '*'
            row=4;
            column=1;
        case '0'
            row=4;
            column=2;
        case '#'
            row=4;
            column=3;

        otherwise
            num=str2num(testNumber(n));
            row=ceil((num)/3);
            column=rem(num,3);

            if column==0
                column=3;
            end
    end
```

این الگوریتم برای اعداد با توجه به مقدار انها میتواند با استفاده از mod و خارج قسمت تقسیم انها بر 3 جایگاه انها را تشخیص دهد .

اما برای کاراکتر های ABCD#* باید به صورت دستی به انها یک شماره ستون و سطر تخصیص دهیم .

حالا در پارت بعدی این قسمت باید این امواج تولید شده را به هم بچسبانیم و تولید یک صدا کنیم که تمام کاراکتر ها را شامل شود .

```
52          disp([row , column]);
53          y1=sin(2*pi*fLow(row)*t);
54          y2=sin(2*pi*fUp(column)*t);
55          y=(y1+y2)/2;
56          on=Ton*fs;
57          out=[out y(1:on) silence];
58      end
59
60      %sound(out,fs)
61      audiowrite('./y.wav',out,fs)
```

که موج تولید شده از جمع دو تابع sin ای با توجه به فرکانس بالا و پایین تخصیص داده شده به هر کاراکتر تولید میشود (مشابه آنچه در صورت پروژه ذکر شده است)

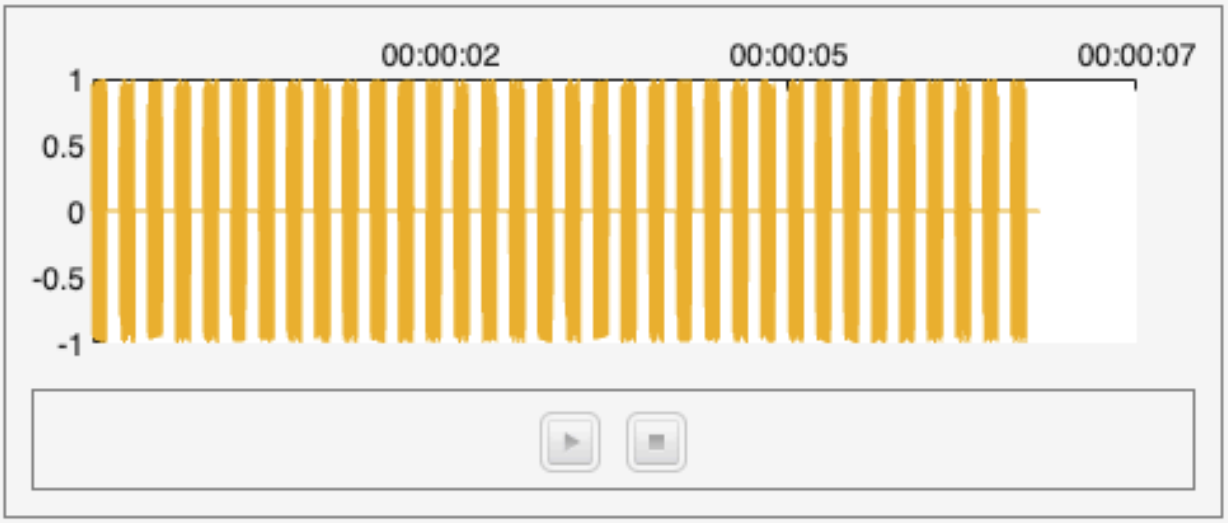
Import Data

Import y.wav

Variables to be imported:

Name	Size	Class	Value
data	54400x1	double	54400x1 double
fs	1x1	double	8000

Preview:



☐ Don't show this dialog again for importing audio

Import Cancel

Part 2 (آنالیز)

```
[a,Fs]=audioread("y.wav");
data=cell(2,16);
fLow=[697 ,770 ,852 ,941];
fUp=[1209 ,1336 ,1477 ,1633];
fs=8000;
Ts=1/fs;

Ton=0.1;
t=0:Ts:Ton;

Toff=0.1;
on=Ton*fs;

s=size(a);
dataLabel=['1','2','3','A','4','5','6','B','7','8','9','C','*','0','#','D'];

output=[];

```

در این بخش باید باتوجه به قرار گیری کاراکتر ها در تلفن یک دیتالیبل تولید کنیم که مشابه ترتیب زیر است :

1 , 2 , 3 , A

4 , 5 , 6 , B

7 , 8 , 9 , C

* , 0 , # , D

سپس برای هر یک از کاراکتر های موجود در تلفن با توجه به فرکانس بالا و فرکانس پایین انها برای هر کدام یک موج تولید میکنیم که نمایانگر آن کاراکتر است .

پس در نهایت یک مپ ست ساخته ایم که هر کاراکتر به همراه موج آن کاراکتر موجود است .

حالا باید برای ویس ورودی ، با استفاده از کورولیشین گیری تشخیص دهیم که یک تکه از صدای ورودی مشابه ترین به کدام موج برای کاراکتر است .

```
21      for n=1:length(dataLabel)
22          num=n;
23          row=ceil(num/4);
24          column=rem(num,4);
25          if column==0
26              column=4;
27          end
28
29          disp([row, column]);
30          y1=sin(2*pi*fLow(row)*t);
31          y2=sin(2*pi*fUp(column)*t);
32
33          y=(y1+y2)/2;
34
35          data(1,n)={dataLabel(n)};
36          data(2,n)={y(1:on)};
37      end
38
```

```
for n=0:(s(1)/(0.2*Fs))-1
    samples=[(2*n*Fs*0.1)+1,(2*n*Fs*0.1)+on];

    [b,Fs]=audioread("y.wav",samples);
    ro=zeros(1,16);

    for i=1:16
        ro(i)=corr2(data{2,i},transpose(b));
    end
    [MAXRO,pos]=max(ro);
    na=cell2mat(data(1,pos));

    output=[output na];
end
disp(output);
```

در کد بالا تکه ی کورولیشن گیری انجام شده و ماکسیم کورولیشن را به آن کاراکتر تخصیص میدهیم .

و در نهایت کاراکتر های تشخیصی را نمایش میدهیم که خروجی برای ورودی سنتر اولیه که شامل کاراکتر های زیر بود :

```
testNumber='1234567890*#ABCD1234567890**##ABCD';
```

```
1234567890*#ABCD1234567890**##ABCD
>>
```

Part 3

در این قسمت باید تصویر IC با استفاده از Corrolation در یک PCB شناسایی شود :

main function (ICRecognization)

```
function ICrecognition(ICpath , PCBpath)
    IC = imread(ICpath);
    PCB = imread(PCBpath);
    grayIC = rgb2gray(IC);
    grayPCB = rgb2gray(PCB);
    rotatedGrayIC = imrotate(grayIC , 180);
    out = twoImageCorrolation(grayPCB , grayIC);
    outRotated = twoImageCorrolation(grayPCB , rotatedGrayIC);
    plotRect(PCB , IC , {out , outRotated});

end
```

همان طور که در صورت پروژه ذکر شده ، تابع ICRecognition دو ورودی میگیرد که آدرس های ذخیره سازی تصاوری PCB , IC هستند .

سپس دو عکس را میخواند و در متغیر های PCB , IC ذخیره میکند .

حالا برای ساده سازی دو عکس را با تابع rgb2gray تبدیل به عکس های خاکستری میکنیم که corrolation گیری در آنها راحتتر شود .

و سپس با توجه به اینکه در صورت پروژه ذکر شده که IC میتواند در حالت ۱۸۰ درجه هم چرخیده باشد بنابراین با استفاده از تابع imroatate یک عکس سیاه سفید مشابه قبلی درست میکنیم با این تفاوت که ۱۸۰ در جه .IC ما چرخیده و این کورولیشن گیری رو هم روی عکس ۱۸۰ درجه چرخیده باید لحاظ کنیم تا بتوانیم IC های برعکس را تشخیص بدهیم .

تابع twoImageCorrolation دو ورودی PCB , IC میگیرد و روی اینها هر بار به اندازه ی اندازه تصویر IC کورولیشن گیری میکند تا بتواند IC ها را در PCB تشخیص دهد .

```
function out=twoImageCorrolation (PCB,IC)
    [ICRow, ICCol] = size(IC);
    [PCBRow, PCBCol] = size(PCB);

    IC = double(IC);
    out=zeros(PCBRow-ICRow+1,PCBCol-ICCol+1);


    for i=1:(PCBRow-ICRow+1)
        for j=1:(PCBCol-ICCol+1)
            littlePCB=double(PCB(i:i + ICRow - 1, j:j + ICCol - 1));
            out(i,j)=corr(IC,littlePCB);
        end
    end



end
```

در تصویر بالا میتوان پیاده سازی corrollation را مشاهده کرد .

با توجه به صورت پروژه باید این corrolation نسبت به دو عکس Normalized شود .

بنابراین corr coeff از رابطه ی زیر بدست می آید :

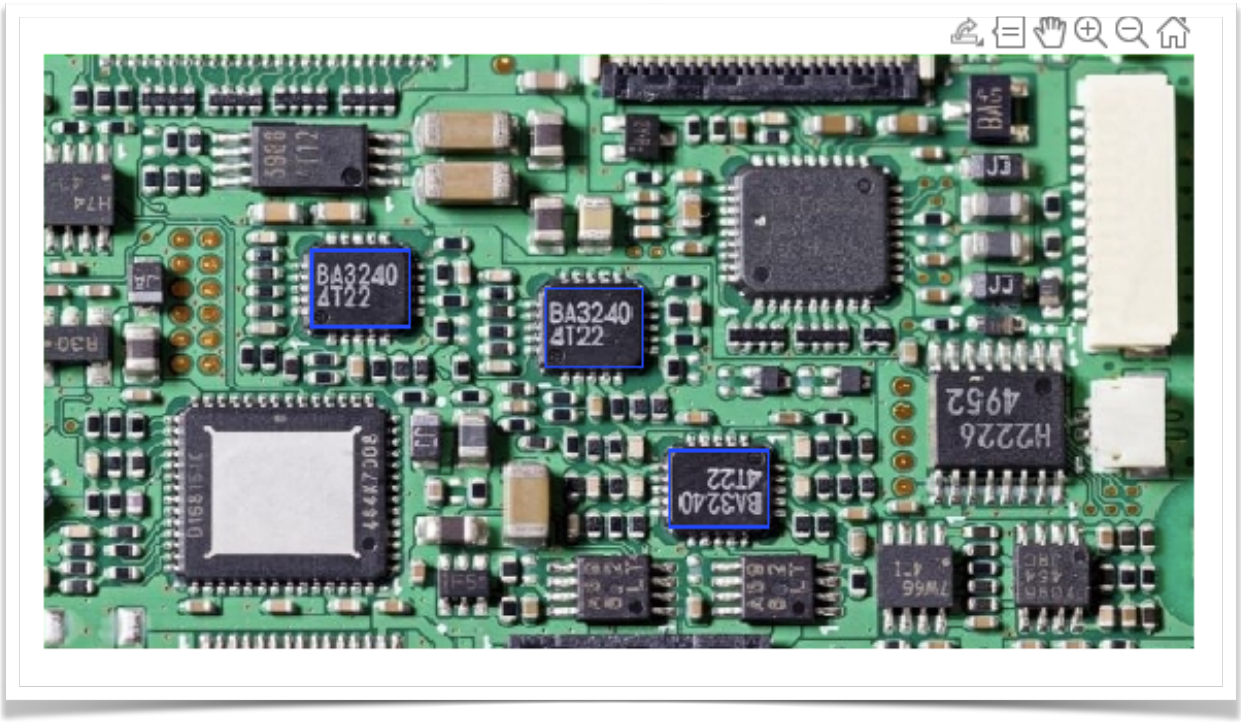
```
function coeff = corr(p1 , p2)
    coeff = sum((p1.*p2)) / sqrt( sum(p1.*p1) .* sum(p2.*p2) );

end

```

برای اینکه اثر یک مشاهده که صرفا دامنه زیادی دارد با مشاهده ی دیگری که دامنه ی زیادی دارد با مشاهده دیگری که دامنه زیادی ندارد مشابه باشد ، این نرمالیزاسیون را انجام میدهیم .

حالا پس از کورولیشین گیری باید نتیجه ی کورولیشین گیری را یک بار برای حالت عادی ic و یک بار برای ic با ۱۸۰ درجه دوران را در یک CELL به نام (matrix_cell) M_cell. میریزیم و با توجه به مختصات درون m_cell حالا کافیست مستطیل های دیتکت شده در M_cell را روی PCB نشان دهیم .

```
1
2   function result=plotRect(PCB,IC,M_cell)
3
4       [IC_row,IC_col , dim]=size(IC);
5       threshold=0.94;
6       figure,imshow(PCB);
7       hold on
8       for l=1:length(M_cell)
9           M_1=cell2mat(M_cell(1,l));
10          M=find(M_1>threshold);
11          [rows, columns]=ind2sub(size(M_1),M);
12          for k=1:length(rows)
13              disp([columns(k) rows(k) IC_col IC_row ])
14              rectangle('Position',[columns(k) rows(k) IC_col IC_row],'EdgeColor','b');
15          end
16      end
17      F=getframe(gcf);
18      result=frame2im(F);
19  end
```

output



نکته حائز اهمیت این است که در کورولیشن گیری چون تکه تکه به جلو حرکت میکنیم ممکن است برای یک IC درون pcb چندین جا دیتکت شود (چون مقدار کورولیشین گیری انها بیشتر از Thershhold میباشد)