به نام خدا

سیگنال ها و سیستم ها

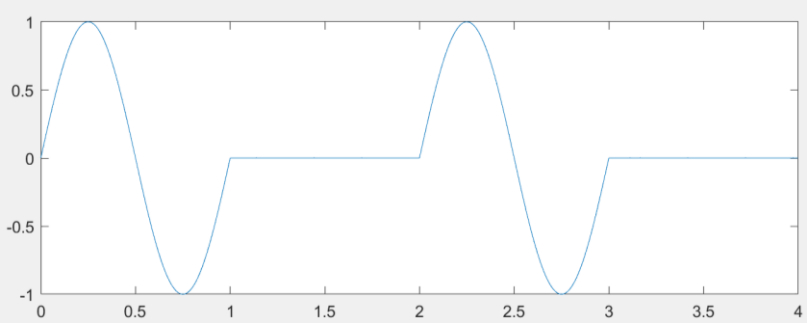
تمرین کامپیوتری دوم

مهلت تحویل: جمعه ۱۸ فروردین ساعت ۱۷:۰۰

لطفا صبورانه، گام به گام در این تمرین کامپیوتری همراه ما باشید. قرار است هم آموزش ببینیم و هم پیاده سازی عملی را تجربه کنیم. به هیچ وجه این تمرین را به روز آخر موکول نکنید و سعی کنید آهسته و پیوسته آن را انجام دهید.

بخش اول:

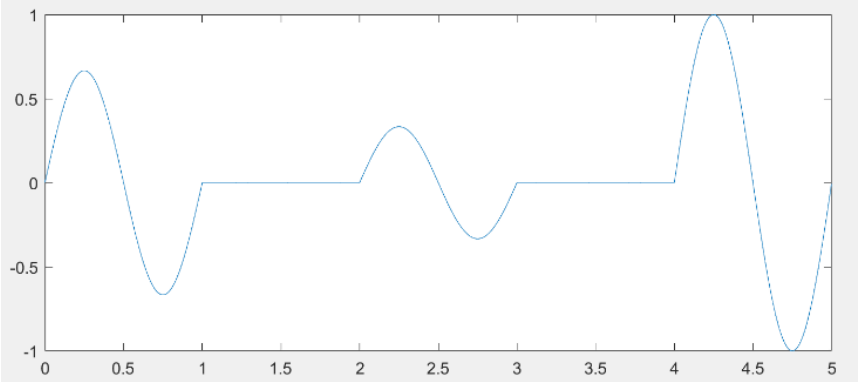
فرض کنید می خواهیم پیامی را به صورت بی سیم از طریق امواج برای شخصی ارسال کنیم. ابتدا پیام را در کامپیوتر به صورت یک رشته بیت باینری در آورده و سپس بیت ها را کدگذاری می کنیم. مثلا به جای هر بیت یک سیگنال خاص را ارسال می کنیم. فرض کنید به جای بیت صفر، سیگنال $x_0(t) = 0$ را به مدت یک ثانیه و به جای بیت یک، سیگنال $x_1(t) = \sin(2\pi t)$ را به مدت یک ثانیه ارسال کنیم. مثلاً رشته بیت 1010 به صورت زیر ارسال می شود:



فرض کنید قدرت فرستنده به گونه ای است که حداکثر دامنه ای که به سیگنال می تواند بدهد، دامنه ی یک است. همچنین فرکانس نمونه برداری سیگنال را $f_s = 100\ Hz$ در نظر بگیرید که باعث می شود هر سیگنال یک ثانیه ای شامل ۱۰۰ سمپل باشد. این فرض را تا آخر تمرین کامپیوتری در نظر داشته باشید.

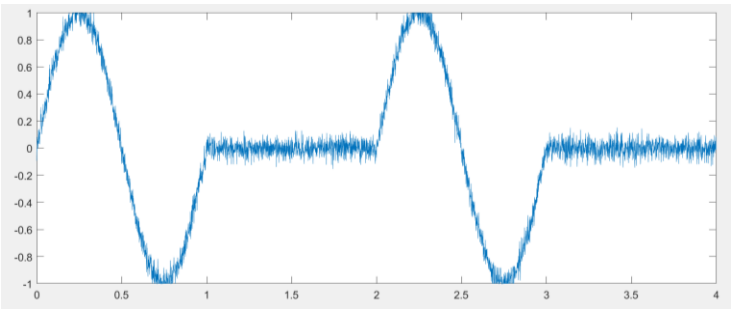
در گیرنده این سیگنال دریافت می شود. با فرض این که فرستنده و گیرنده از نظر زمانی کاملاً سنکرون باشند، برای رمزگشایی از پیام در گیرنده کافی است در هر یک ثانیه، correlation سیگنال دریافتی را با $2\sin(2\pi t)$ حساب کنیم. هر جایی correlation مقدار داشته باشد (مقدار $\int 2sin^2(2\pi t)\mathrm{dt} = 1$) می فهمیم بیت ارسالی برابر یک بوده و هر جا مقدار correlation صفر شود می فهمیم بیت ارسالی برابر صفر بوده است. توجه داشته باشید عدد 2 در دامنه ی $2\sin(2\pi t)$ هیچ اهمیتی ندارد و فقط برای رُند شدن مقدار correlation است.

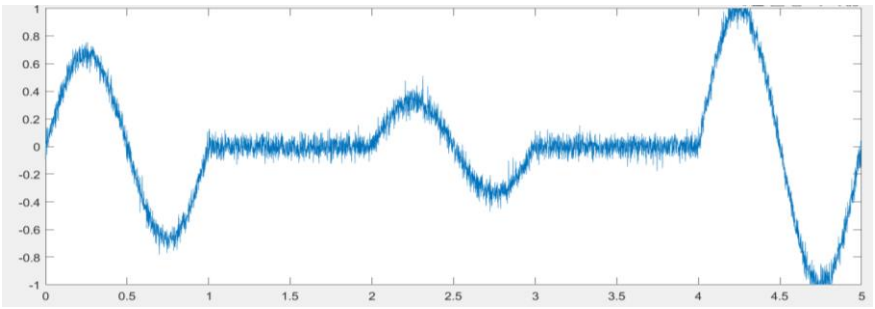
در سناریوی بالا سرعت ارسال پیام (bit rate) $1\ \frac{bit}{sec}$ است. حال فرض کنید بخواهیم bit rate را افزایش دهیم. برای این کار باید کدگذاری را روی بیت های بیشتری انجام دهیم. مثلا به جای هر دو بیت، یک سیگنال خاص را ارسال کنیم. برای ارسال 00 سیگنال $x_0(t) = 0$ را به مدت یک ثانیه، برای ارسال 01 سیگنال $x_1(t) = \frac{1}{3}\sin(2\pi t)$ را به مدت یک ثانیه، برای ارسال 10 سیگنال $x_2(t) = \frac{2}{3}\sin(2\pi t)$ را به مدت یک ثانیه و برای ارسال 11 سیگنال $x_3(t) = \sin(2\pi t)$ را به مدت یک ثانیه ارسال می کنیم. مثلا رشته بیت 1000010011 به صورت زیر ارسال می شود.



در گیرنده این سیگنال دریافت می شود. مجدداً برای رمزگشایی از پیام در گیرنده کافی است در هر ثانیه، correlation سیگنال دریافتی را با $2\sin(2\pi t)$ حساب کنیم و بسته به خروجی correlation که می تواند مقادیر صفر، $\frac{1}{3}$ ، $\frac{2}{3}$ و یک بگیرد راجع به بیت های ارسالی تصمیم گیری کنیم. در این سناریو سرعت ارسال پیام (bit rate) $2\ \frac{bit}{sec}$ است.

این روند را می توان به همین ترتیب ادامه داد و سرعت ارسال پیام را افزایش داد. یعنی هر سه بیت را کدگذاری کرد تا سرعت ارسال سه برابر شود و ... اما این اضافه شدن سرعت در عمل یک مشکل اساسی دارد که به ما اجازه نمی دهد سرعت را از حدی بیشتر کنیم. مشکل نویز است! وقتی فرستنده سیگنالی را به گیرنده ارسال می کند، در گیرنده علاوه بر سیگنال اصلی، نویز نیز به سیگنال اصلی اضافه می شود. مثلا در سناریوی اول و دوم سیگنال های زیر به گیرنده می رسد:





حال اگر correlation سیگنال دریافتی را با $2\sin(2\pi t)$ حساب کنیم، خروجی ها دقیقاً اعدادی که انتظار داشتیم نمی شوند و بنابراین ممکن است به اشتباه رمز گشایی کنیم. هر چه سرعت ارسال بیت (bit rate) بیشتر باشد این خطا بیشتر خواهد بود. چرا؟ قدری فکر کنید و بعد پاسخ این سوال را که در ادامه آمده است بخوانید.

به عنوان مثال سناریوی اول که سرعت ارسال $1\ \frac{bit}{sec}$ بود را در نظر بگیرید. انتظار داشتیم مقادیر خروجی correlation عدد صفر یا عدد یک باشد ولی در حضور نویز این اعداد کمی تغییر می کنند. پس به نظر می رسد باید عدد 0.5 را به عنوان threshold (آستانه) تعریف کنیم. اگر correlation بیش از 0.5 شد اعلام کنیم بیت مورد نظر یک بوده و اگر کمتر از 0.5 شد اعلام کنیم بیت مورد نظر صفر بوده است.

در سناریوی دوم که سرعت ارسال $2\ \frac{bit}{sec}$ بود، انتظار داشتیم مقادیر خروجی correlation عدد صفر، $\frac{1}{3}$ ، $\frac{2}{3}$ و یک شود، اما به خاطر وجود نویز این اعداد ممکن است هر مقداری بگیرند. بنابراین باید سه آستانه ی $\frac{1}{6}$ ، $\frac{3}{6}$ و $\frac{5}{6}$ را تعریف کنیم و با توجه به آنها راجع به بیت ها تصمیم گیری کنیم. چون فاصله ی این اعداد از هم کمتر شده است بنابراین احتمال خطا در تصمیم گیری بیشتر می شود.

پس در محیط عملیاتی، در روش پیشنهاد شده، یک trade off بین افزایش سرعت انتقال پیام و مقاوم بودن نسبت به نویز وجود دارد. حال می خواهیم مفاهیم بیان شده در این بخش را شبیه سازی کنیم.

هدف این تمرین ارسال پیام به زبان انگلیسی از فرستنده به گیرنده است. هر پیام فقط شامل حروف کوچک انگلیسی، فاصله، نقطه، ویرگول، علامت تعجب، سمی کالن (;) و کوتیشن (") است. بنابراین در مجموع ۳۲ کاراکتر داریم. به هر کاراکتر ۵ بیت مرتبط می کنیم.

تمرین ۱-۱) یک سلول به اسم Mapset با ابعاد 32×2 درست کنید. در سطر اول خود کارکترها را قرار دهید و در سطر دوم ۵ بیتی که به آنها مرتبط کردید را قرار دهید. به عنوان مثال شکل زیر قسمت ابتدایی Mapset را نشان می دهد. (دستور dec2bin برای باینری کردن اعداد کمک کننده خواهد بود.)

	1	2	3
1	'a'	'b'	'c'
2	'00000'	'00001'	'00010'

تمرین ۱-۲) تابعی (*function*) به نام *coding_amp* بنویسید که ورودی های آن ۱) پیام مورد نظر برای ارسال و ۲) سرعت ارسال اطلاعات باشد و خروجی آن پیام کدگذاری شده باشد.

راهنمایی: مثلا فرض کنید می خواهیم پیام *bc* را با سرعت $1\ \frac{bit}{sec}$ ارسال کنیم. ابتدا حروف پیام را به صورت باینری در آورید که (با توجه به شکل بالا) می شود: *0000100010* ، سپس رشته باینری به دست آمده را مطابق آنچه در مقدمه توضیح داده شد کدگذاری کنید. توجه داشته باشید پیام ممکن است هر کلمه یا جمله ای با طول دلخواه شامل ۳۲ کاراکتر ذکر شده باشد. در بخش های بعدی بیشترین سرعت ارسال اطلاعات را ۳ بیت برثانیه در نظر خواهیم گرفت گرچه کد شما می تواند هر عدد طبیعی دلخواهی را پشتیبانی کند! (دستورات *extract* و *strlength* برای جداسازی کاراکترهای پیام کمک کننده خواهند بود.)

توجه: فرض کنید طول رشته بیت با سرعت اطلاعات همخوانی دارد به این معنی که پیش نمی آید طول رشته بیت بر تعداد بیت ارسالی در هر ثانیه تقسیم پذیر نباشد!

تمرین ۱-۳) خروجی تابع *coding_amp* را برای پیام (کلمه‌ی) *signal* با سرعت ارسال اطلاعات یک، دو و سه بیت بر ثانیه به صورت جداگانه رسم کنید.

تمرین ۱-۴) تابعی به نام *decoding_amp* بنویسید که ورودی های آن ۱) پیام کدگذاری شده (سیگنال زمانی تولید شده در قسمت قبل) و ۲) سرعت ارسال اطلاعات باشد و خروجی آن پیام رمز گشایی شده باشد. تابعی که نوشتید را روی همان پیام *signal* که در قسمت ۱-۳ با سرعت ارسال های مختلف کُد کردید تست کنید تا مطمئن شوید کدتان درست کار می کند.

توجه: اگر می خواهید نتایج *correlation* کاملاً شبیه آنچه که در مقدمه بیان شد باشد، عدد *0.01* را پشت *correlation* ضرب کنید. چون به صورت گسسته *correlation* را حساب می کنید و تعداد نمونه ها در هر بازه‌ی *correlation* گیری ۱۰۰ نمونه است این اتفاق رخ می دهد.

تمرین ۱-۵) در این قسمت می خواهیم به سیگنال دریافتی در گیرنده نویز اضافه کنیم تا شبیه سازی مشابه شرایط واقعی شود. برای تولید نویز از دستور *randn* استفاده کنید. این دستور نویز گوسی با میانگین صفر و واریانس یک تولید می کند. نشان دهید خروجی این دستور یک نویز الف) گوسی، ب) با میانگین صفر و ج) با واریانس یک است. برای مثال این سه نکته را روی خروجی *randn(1,3000)* نشان دهید.

تمرین ۱-۶) برای زیاد یا کم کردن قدرت نویز کافی است در پشت دستور *randn* عدد دلخواه σ (بدون توان ۲) را ضرب کنید. این عدد کاری می کند تا واریانس نویز σ^2 شود. به پیام *signal* بعد از این که کدگذاری شد نویز گوسی با واریانس *0.0001* و میانگین صفر اضافه کنید و بعد آن را *decode* کنید. آیا بازهم پیام *signal* استخراج شد؟ نتایج *decoding* را برای هر سه *bit rate* یک، دو و سه به صورت جداگانه گزارش کنید.

تمرین ۱-۷) قدرت نویز را کم کم و طی چندین مرحله افزایش دهید و هر بار قسمت ۱-۶ را تکرار کنید. مشاهدات خود را گزارش کنید. کدام یک از سه *bit rate* به نویز مقاوم تر بودند؟ آیا نتایج با آنچه در مقدمه بیان شد همخوانی داشتند؟

تمرین ۱-۸) طی شبیه سازی هایی که در قسمت ۱-۷ انجام دادید، بیشترین واریانس نویز که *bit rate* سه به آن مقاوم بود به صورت تقریبی چند بود؟ برای *bit rate* دو و یک چه طور؟

تمرین ۱-۹) به نظر شما با چه راهکاری می توان *bit rate* را نسبت به نویز مقاوم تر کرد؟ پاسخ خود را در هایلات قرمز جستجو کنید. حال در ادامه پاسخ را مطالعه کنید.

اگر قدرت فرستنده ی ما بیشتر می بود می توانستیم دامنه ی سیگنال بیشتری داشته باشیم و بنابراین فاصله ی *threshold* هایی که برای تصمیم گیری در نظر گرفته بودیم بیشتر می شد و در نتیجه به نویز حساسیت کمتری ایجاد می شد. مثلا برای ارسال اطلاعات با سرعت $2\ \frac{bit}{sec}$ ، اگر قدرت فرستنده به گونه ای می بود که بیشینه ی دامنه ی سیگنال سه می شد، شرایط بهتر می شد. به عبارت دیگر، برای ارسال 00 سیگنال $x_0(t) = 0$ را به مدت یک ثانیه، برای ارسال 01 سیگنال $x_1(t) = \sin(2\pi t)$ را به مدت یک ثانیه، برای ارسال 10 سیگنال $x_2(t) = 2\sin(2\pi t)$ را به مدت یک ثانیه و برای ارسال 11 سیگنال $x_3(t) = 3\sin(2\pi t)$ را به مدت یک ثانیه ارسال می کردیم.

پس در کل می توان نتیجه گرفت در روش کدگذاری دامنه اگر می خواهیم سرعت ارسال اطلاعات را افزایش دهیم باید *power* بیشتری نیز مصرف کنیم تا در برابر نویز مقاوم باشیم.

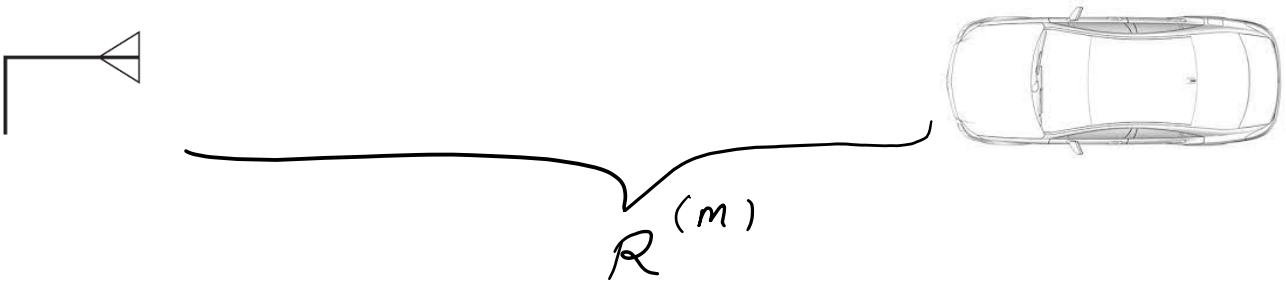
تمرین ۱-۱۰) به نظر شما در کدینگ دامنه، اگر نویز نباشد، بیت ریت را تا کجا می توان افزایش داد؟

تمرین ۱-۱۱) در کدینگ دامنه با فرض این که همچنان بیشترین دامنه ی ارسالی فرستنده برابر یک است، اگر سیگنال دریافتی را به جای $2\sin(2\pi t)$ در $10\sin(2\pi t)$ ضرب کنیم، آیا عملکرد روش به نویز مقاوم تر می شود؟ توضیح دهید.

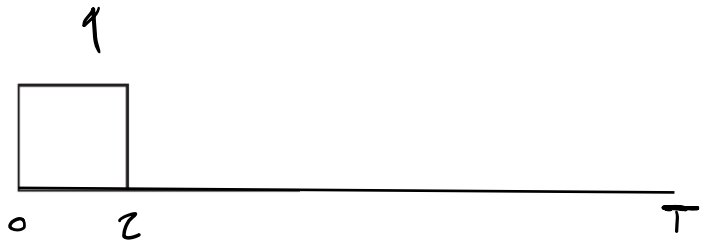
تمرین ۱-۱۲) سرعت اینترنت ADSL خانگی معمولاً چند $\frac{bit}{sec}$ است؟ ما در این تمرین با چه سرعتی اطلاعات را ارسال کردیم ؟!!!

بخش دوم:

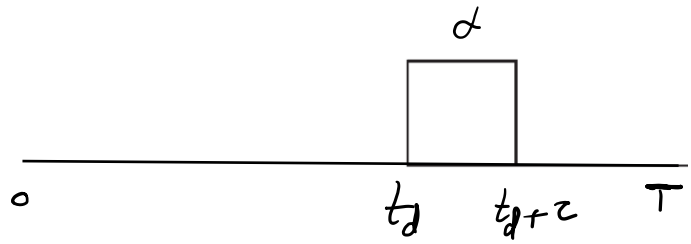
می خواهیم توسط یک رادار، فاصله ی یک جسم از رادار را بیابیم.



سیگنال ارسالی رادار به صورت زیر است:



این سیگنال پس از برخورد به جسم و بازگشت به سمت رادار، به صورت زیر دریافت می شود:



که در شکل بالا $t_d = \frac{2R}{C}$ است و مقدار $\alpha < 1$ اهمیتی ندارد.

پارامتر مجهول مساله t_d می باشد زیرا با به دست آوردن آن، فاصله نیز مشخص می شود. برای به دست آوردن این پارامتر، از همان ایده ی correlation یا template matching استفاده می کنیم.

تمرین ۲-۱) سیگنال ارسالی را با فرضیات زیر تولید کرده و رسم کنید (t_s فاصله ی دو نمونه ی زمانی و یا عکس فرکانس نمونه برداری f_s است).

```
ts=1e-9;T=1e-5;tau=1e-6;
```

تمرین ۲-۲) سیگنال دریافتی را با فرض $R = 450\ m$ تولید کرده و رسم کنید ($\alpha = 0.5$).

تمرین ۲-۳) حال به صورت برعکس به مساله نگاه کنید. یعنی فرض کنید سیگنال دریافتی را داریم و می خواهیم از روی آن فاصله را محاسبه کنیم. با ایده ی correlation یا template matching، این کار را انجام دهید.

تمرین ۲-۴) آیا می توان با استفاده از مفهوم کانوولشون روش پیشنهادی در تمرین ۲-۳ را پیاده سازی کرد؟ با استفاده از دستور conv ، آن را پیاده سازی کنید.

تمرین ۲-۵) حال به سیگنال دریافتی کمی نویز اضافه کنید و تمرین ۲-۳ را تکرار کنید. قدرت نویز را کم کم و طی چندین مرحله افزایش دهید و ببینید تا کجا همچنان می توانید فاصله را به درستی تشخیص دهید.

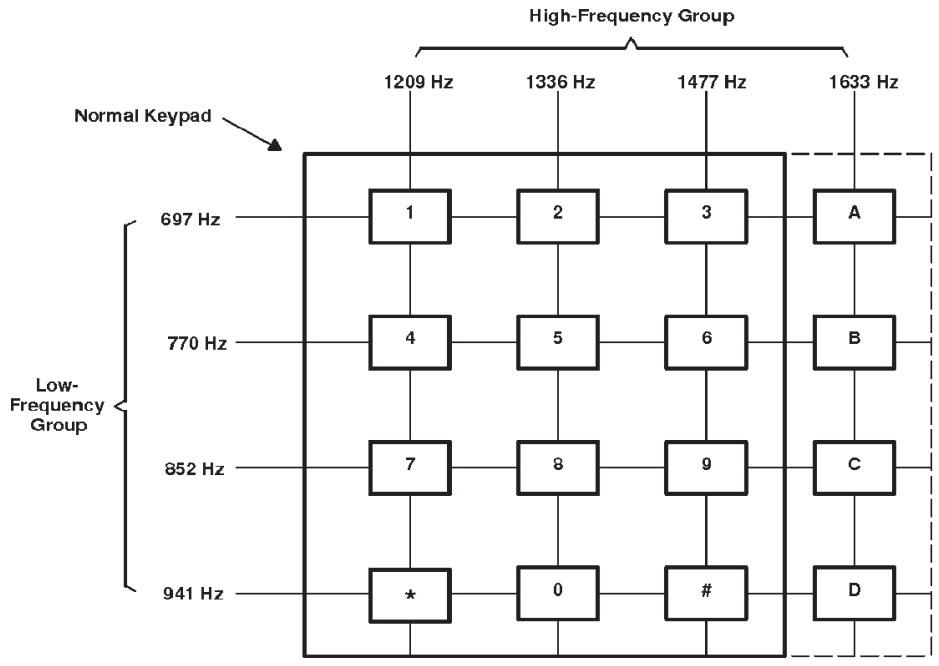
بخش سوم:

نوای دوگانه چند بسامدی (Dual-Tone Multi Frequency یا به اختصار DTMF)، اصطلاح فنی فرکانس‌های صوتیست که با فشردن کلیدهای تلفن ایجاد می‌شوند. این روش که بعضا touch tone نیز نامیده می‌شود، در ابتدا در ارسال سیگنال‌های تلفن به مراکز سوییچ محلی و دریافت سیگنال از آن‌ها کاربرد داشته است، اما امروزه کاربردهای متعدد دیگری نیز در حوزه مخابرات پیدا کرده است.

شرح رویکرد:

به هر کلید تلفن یک نوای دوگانه (دو نوا با فرکانس‌های مختلف) اختصاص داده می‌شود؛ یکی با فرکانس پایین و دیگری با فرکانس بالا که پس از فشردن کلید به صورت همزمان نواخته می‌شوند. (همان صدایی که پس از شماره‌گیری تلفن میشنوید!)

همانطور که در شکل زیر مشاهده میکنید، به هر یک از ۴ ردیف صفحه‌کلید، یکی از نواهای با فرکانس پایین و به هر یک از ۳ ستون یکی از نواهای با فرکانس بالا اختصاص داده میشود. کلیدهای ستون چهارم هم که با حروف A, B, C, D مشخص شده اند، دلخواه هستند و بیشتر کاربرد نظامی دارند.



مشخصات نوای اختصاصی هر سطر و ستون یک صفحه کلید در سیگنالینگ **DTMF**

پیاده سازی:

در این تمرین، شما باید هر دو بخش سنتز و آنالیز این نوع سیگنالینگ را، با توجه به توضیحات شکل بالا، پیاده سازی نمایید.

تمرین ۳-۱)

آ) سنتز:

منظور از سنتز، تولید تون‌های آنالوگ متناظر برای نشان دادن ارقام یک صفحه‌کلید تلفن است. در این قسمت، شما باید سیگنال متناظر با هر کلید را تولید کرده و با کنار هم قرار دادن آن‌ها، صوت متناظر با عدد ۴۳۲۱۸۷۶۵ را با استفاده از سیگنالینگ DTMF تولید کنید. مدت زمان هر سیگنال(T_{on}) را ۰.۱ ثانیه و فاصله زمانی بین پخش دو سیگنال (T_{off}) را هم ۰.۱ ثانیه در نظر بگیرید. فرکانس نمونه برداری را ۸ کیلوهرتز در نظر بگیرید و با استفاده از دستور audiowrite سیگنال تولیدی را در y.wave ذخیره کنید.

راهنمایی: کد زیر برای تولید سیگنال فقط یک شماره ی صفحه کلید نوشته شده است. با دستور sound می توانید سیگنال را بشنوید.

```
fr = [697 770 852 941];          % row frequencies
fc = [1209 1336 1477];           % column frequencies
fs = 8000;                       % signal sampling frequency
Ts = 1/fs;                       % signal sampling time
Ton = 0.1;                       % ON time for each DTMF signal (in second)
Toff = 0.1;                      % OFF time (gap) between DTMF signals (in second)
t = 0:Ts:Ton;
y1 = sin(2*pi*fr(k)*t);          % k is row index
y2 = sin(2*pi*fc(j)*t);          % j is column index
y = (y1 + y2)/2;
sound(y,fs)
```

تمرین ۳-۲)

ب)آنالیز:

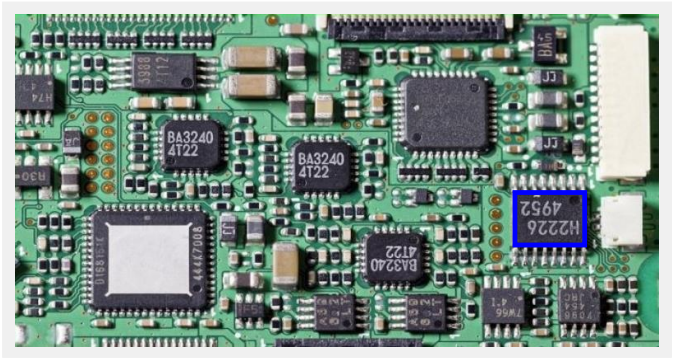
آنالیز به این معنی است که شما باید یک صوت را که به روش DTMF کد شده است، رمزگشایی یا دیکود کنید. به عبارت دیگر باید بگویید این سیگنال منتج از فشردن چه کلیدهایی بوده است.

در پوشه تمرین، یک فایل به نام ’a.wav‘ قرار دارد. آن را رمزگشایی کنید. این فایل را با استفاده از دستور audioread به شکل `[a,Fs] = audioread('a.wav')` لود کنید و سپس رمزگشایی کنید.

راهنمایی: ابتدا باید بازه‌های زمانی T_{on} را از سیگنال a پیدا و جدا کنیم. حال باید بفهمیم هر بازه ی زمانی منتج از فشردن کدام کلید بوده است. برای این کار از معیار correaltion استفاده می کنیم. بنابراین برای هر بازه ی زمانی جدا شده، correaltion سیگنال آن بازه را با سیگنال های DTMF کلیدهای مختلف حساب می کنیم. سپس آن کلیدی که سیگنالش correaltion بزرگتری را ایجاد می کند به عنوان کلیدی در نظر می گیریم که سیگنال صوتی را تولید کرده است. این کار را برای همه ی بازه های زمانی جدا شده تکرار می کنیم تا رمزگشایی کامل انجام شود. برای اطمینان از صحت کد خود، حتما فایل صوتی که در قسمت قبل تولید کردید را هم رمزگشایی کنید.

بخش چهارم:

در این بخش می‌خواهیم با روش template matching یا correlation، اسکریپت متلبی بنویسیم که مشابه تصویر زیر، قطعه‌ای که تصویر آن به عنوان ورودی داده می‌شود را بر روی تصویر یک برد مدار چاپی[1] شناسایی کند و در صورت وجود اطراف آن(ها) مستطیل رسم کند.



تصویر یک برد مدار چاپی که قطعه مورد نظر روی آن شناسایی و مشخص شده است.

ابتدایی‌ترین روش برای انجام این کار استفاده از ضریب همبستگی نرمالایز شده است که برای دو سیگنال تک بعدی x و y به صورت زیر تعریف می‌شود:

$$Correlation\ Coeff(x,y) = \frac{\sum_{n=1}^{L} x[n]y[n]}{\sqrt{(\sum_{n=1}^{L} x^2[n]) \times (\sum_{k=1}^{L} y^2[k])}}$$

توجه: مفهوم correlation گیری که در این بخش بیان شد قدری با آنچه که در کلاس مطرح شد و در تمارین قبلی انجام دادید تفاوت دارد. در این بخش ضریب correlation نرمالیزه در نظر گرفته شده اما قبلاً این کار را انجام نداده بودیم. درست تر و دقیق تر این است که این نرمالیزاسیون انجام شود. برای این که اثر یک مشاهده که صرفا دامنه ی زیادی دارد با مشاهده ی دیگری که دامنه ی زیادی ندارد مشابه باشد، این نرمالیزاسیون انجام می شود.

احتیاجی نیست بخش های قبل را با تعریف جدید تکرار کنید. فقط در این بخش، با این تعریف، مساله را حل کنید.

[1] Printed Circuit Board (PCB)

همان طور که بیان شد، در این بخش می‌خواهیم یک اسکریپت متلب بنویسیم که تصویر یک برد مدار چاپی و تصویر قطعه مورد نظر را گرفته و آن را در برد مدار چاپی شناسایی کند.

توجه: لطفاً هرکدام از توابع خواسته شده در ادامه را در یک فایل با پسوند m با همان نام تابع ذخیره کرده و در کنار گزارش خود ضمیمه کنید.

در ابتدا تابعی با نام select_image تعریف کنید که ورودی ندارد و با فراخوانی آن یک پنجره برای انتخاب فایل تصویر باز شود، بعد از انتخاب تصویر، محتوای آن به صورت یک ماتریس سه بعدی به عنوان خروجی تابع برگردانده شود. (**راهنمایی**: می‌توانید از توابع imgetfile و imread متلب برای این منظور استفاده کنید.)

در ماتریس سه بعدی بدست آمده، هر کانال تصویر مربوط به یکی از کانال‌های رنگ قرمز، سبز و آبی است؛ برای کاهش پیچیدگی محاسبات می‌توان تصویر رنگی را به تصویر خاکستری[۲] تبدیل کرد زیرا که اطلاعات موجود در تصویر خاکستری نیز برای شناسایی و تطابق الگو کافی است. برای این کار تابع rgb_to_gray را تعریف کنید که در آن با کمک ترکیب خطی کانال رنگ‌های قرمز، سبز و آبی تصویر ورودی به صورت زیر، تصویر رنگی سه کاناله را به یک تصویر تک کاناله خاکستری که تنها بیانگر شدت روشنایی برای هر پیکسل است تبدیل می‌کند و آن را به عنوان خروجی تابع قرار می‌دهد.

$$Gray_{channel} = 0.299 \times Red_{channel} + 0.578 \times Green_{channel} + 0.114 \times Blue_{channel}$$

حال تصویر برد مدار چاپی و قطعه BA3240 که در کنار این فایل ضمیمه شده‌است را با کمک توابعی که بالاتر نوشته‌اید در متلب وارد کرده و به تصویر خاکستری تبدیل کرده و درنهایت دو تصویر خاکستری شده را با دستور imshow نمایش داده و خروجی را در گزارش خود بیاورید.

با توجه به رابطه‌ای که برای ضریب همبستگی ارائه شد، تابع corr_2d را تعریف کنید که در ورودی دو تصویر (دو ماتریس) گرفته و ضریب همبستگی آن دو را محاسبه می‌کند. (**راهنمایی**: برای ضرب درایه به درایه دو ماتریس در متلب باید از اپراتور .* استفاده کرد؛ همچنین استفاده از توابع ریاضی داخلی متلب نظیر sum و sqrt نیز بلامانع است.)

تابع corr_matrix را به این صورت تعریف کنید که تصویر خاکستریِ قطعه را از گوشه بالا سمت چپ، روی تصویر خاکستریِ برد مدار چاپی حرکت داده و ضریب همبستگی تصویر قطعه و بخشی از تصویر برد مدار چاپی که در هر مرحله زیر تصویر قطعه قرار گرفته را حساب کرده و مقدار آن را در ماتریسی ذخیره کنید و در پایان این ماتریس را در خروجی تابع قرار دهید.

---

[۲] Grayscale

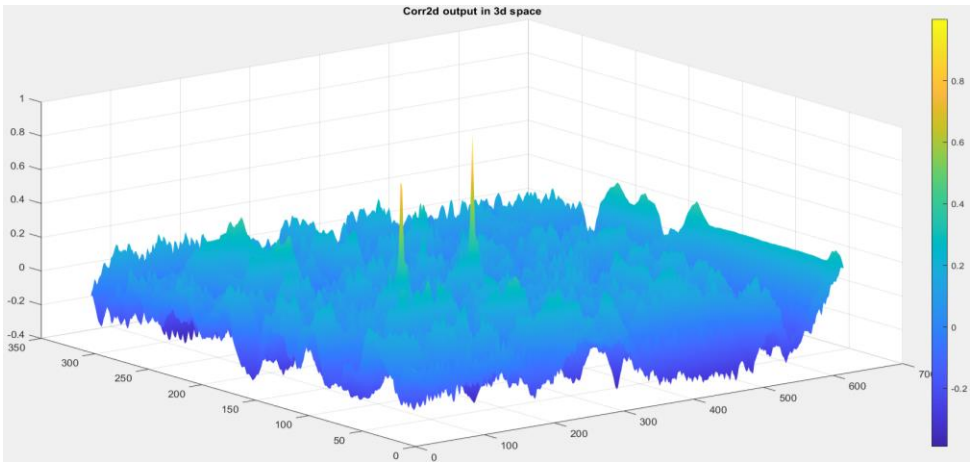
راهنمایی: می‌توانید تابع زیر را تکمیل کنید.

```
function M = corr_matrix(PCB, IC)

    [PCB_row, PCB_col] = size(PCB);
    [IC_row, IC_col] = size(IC);
    IC = double(IC);
                                        %کدی بنویسید که یک ماتریس با درایه‌های صفر به نام M با ابعاد
                                    [PCB_row - IC_row + 1, PCB_col - IC_col + 1] تعریف کند.

    for i=1:(PCB_row - IC_row + 1)
        for j=1:(PCB_col - IC_col + 1)
            PCB_cropped = double(PCB(i:i + IC_row - 1, j:j + IC_col - 1));
      %بعد کدی بنویسید که ضریب همبستگی بین PCB_cropped و IC را  محاسبه کرده و در درایهٔ i,j ماتریس M ذخیره کند.
        end
    end
end
```

تابع plot_surface را طوری تعریف کنید که خروجی تابع corr_matrix را گرفته و آن را به صورت سه بعدی (که برحسب ابعاد سیگنال تصویر برد مدار چاپی است) مشابه تصویر زیر رسم کند.



نمایش سه بعدی ماتریس ضرایب همبستگی

شکل خروجی کد خود که مشابه تصویر بالا است را برای تصویر برد مدار چاپی و قطعه BA3240 که در کنار این فایل ضمیمه شده را در گزارش خود آورده و شرح دهید، پستی و بلندی‌ها (قله‌ها) در این نمودار به چه علت ایجاد شده و نمایانگر چیست؟ (**راهنمایی**: تابع surf متلب برای رسم چنین سطوحی مناسب است.)

حال با توجه به نمودار فوق باید یک حد آستانه[۳] تعیین کنید تا اگر مقدار ضریب همبستگی بیشتر از آن بود، آن بخش به عنوان الگوی مورد نظر شناسایی شود (**توجه**: لطفاً مقداری که برای این حد آستانه تعیین کرده اید را در گزارش خود نیز ذکر کنید)؛ در ادامه تابع plot_box را تعریف کنید که تصویر برد مدار چاپی، تصویر خاکستری قطعه و ماتریس ضرایب

---

[۳] Threshold

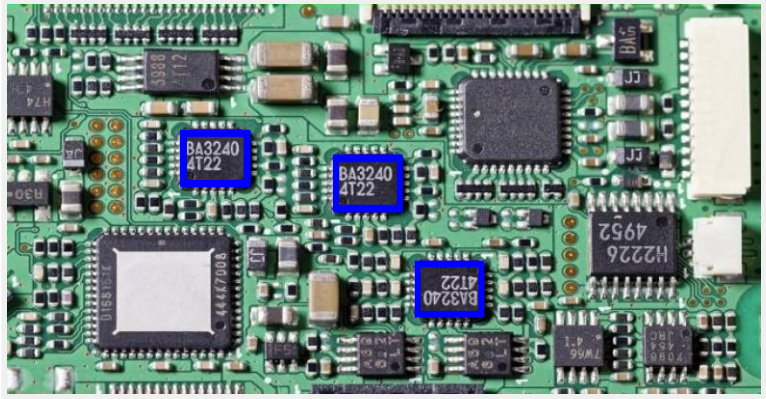
همبستگی (خروجی تابع corr_matrix) را به عنوان ورودی گرفته و در نقاطی که مقدار ضریب همبستگی از حد آستانه تعیین شده بیشتر است (نقاط متناظر با درایه‌های ماتریس همبستگی بدست آمده)، روی تصویر برد مدار چاپی یک مستطیل به عنوان مکان شناسایی شده برای قطعه رسم کند و این تصویر را خروجی دهد؛ توجه کنید که مقدار حد آستانه را باید طوری تعیین کنید که تمام قطعات مورد نظر در تصویر برد مدار چاپی شناسایی و اطراف آن‌ها مستطیل رسم شود و همچنین در مکان‌های دیگر نیز به اشتباه مستطیل رسم نشود. همچنین قطعات می‌توانند دوران داشته باشند همانطور که در تصویر ضمیمه شده برد مدار چاپی دیده می‌شود؛ برای سادگی فرض می‌کنیم قطعات فقط ۱۸۰ درجه می‌توانند دوران داشته باشند؛ پس در ورودی این تابع ماتریس‌های ضرایب همبستگی برای تصویر اصلی و نیز تصویر ۱۸۰ درجه دوران یافته به صورت یک آرایه سلولی داده می‌شود. یعنی این تابع باید به صورت زیر قابل فراخوانی باشد و خروجی آن (result) تصویری مشابه تصویر زیر باشد:

```
M = corr_matrix(PCB_image_gray, IC_image_gray);
M_rotated = corr_matrix(PCB_image, IC_image_gray_rotated);
M_cell = {M; M_rotated};
result = plot_box(PCB_image, IC_image_gray, M_cell);
```

راهنمایی: می‌توانید تابع زیر را تکمیل کنید.

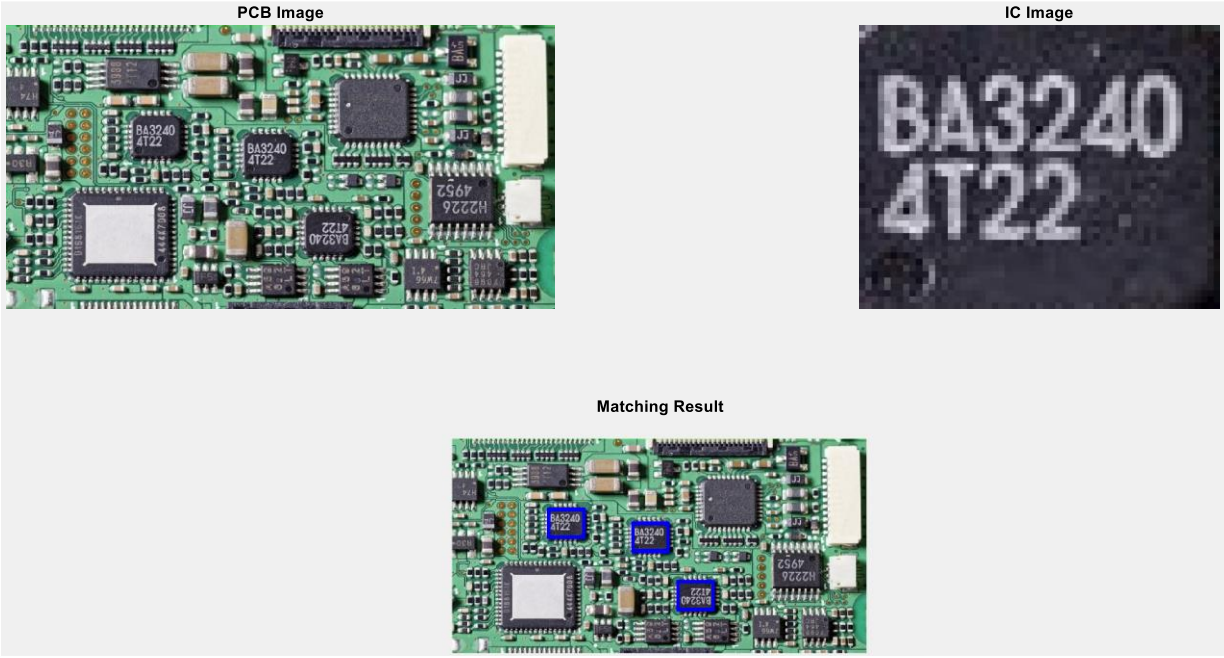
```
function result = plot_box(PCB, IC, M_cell)
      [IC_row, IC_col] = size(IC);
      threshold = 0;

      figure, imshow(PCB);
      hold on;
      for l=1:length(M_cell)
%کدی بنویسید که با کمک تابع find متلب، تمام درایه‌های l امین عنصر M_cell که مقدارشان از
threshold بیشتر است را خروجی دهد و بعد این خروجی را به صورت [rows, cols] در دو متغیر rows و cols
ذخیره کنید.
          for k=1:length(rows)
%کدی بنویسید که با کمک تابع rectangle متلب یک مستطیل در مکان شناسایی شده k امین قطعه
رسم کند.
          end
      end
      F = getframe(gcf);
      result = frame2im(F);
end
```



تصویر برد مدار چاپی داده شده که مکان قطعات **BA3240** روی آن مشخص شده است.

در نهایت یک تکه کد بنویسید که برای تصویر برد مدار چاپی و تصویر قطعه BA3240 که در کنار این فایل ضمیمه شده، با کمک توابعی که تعریف کردیم خروجی مشابه تصویر زیر ایجاد کند؛ این تکه کد را در فایلی به نام main.m ذخیره و در کنار گزارش خود ضمیمه کنید. این کد باید قابلیت اجرای مجدد داشته باشد یعنی با اجرای آن، پنجره انتخاب تصویر برد مدار چاپی و قطعه باز شود و بعد از انتخاب این تصاویر توسط کاربر، خروجی مشابه تصویر ۴ ایجاد و نمایش داده شود. (**راهنمایی**: برای دوران ۱۸۰ درجه تصویر قطعه می‌توانید از تابع imrotate استفاده کنید.)



خروجی مد نظر نهایی مربوط به یافتن قطعات **BA3240** در تصویر برد مدار چاپی داده شده.

نکات کلی:

• درصورت وجود هرگونه پرسش و ابهام به سید مهدی حسینی، نوید رزاقی و استاد ایمیل بزنید.

• فایل نهایی شما باید به صورت یک فایل زیپ شامل گزارشکار به فرمت PDF و کد های متلب باشد.