باسمه تعالی

دانشکده مهندسی مکانیک

پردیس دانشکده‌های فنی

دانشگاه تهران

گزارش تکالیف

جلسه سوم

درس سیستم‌های اندازه‌گیری کارشناسی

دکتر صدیقی

مهدی عبداله چالکی (۸۱۰۶۹۶۲۶۸)

نیم‌سال دوم

سال تحصیلی ۰۰-۹۹

۱- تکلیف اول

در تکلیف اول، هدف ما طراحی مداری است که بتواند یک سیگنال سینوسویید تولید کند به طوری که تنظیم فرکانس این سیگنال به صورت دستی و از طریق یک پتانسیومتر صورت گیرد. همچنین یک دکمه فشاری در مدار تعبیه می‌شود تا تنها در صورت فشردن این دکمه، فرکانس تنظیمی جدید اعمال شود. در این مدار، یک فیلتر پایین‌گذر RC وجود دارد که تاثیر آن را نیز بر روی سیگنال‌های با فرکانس مختلف بررسی کرده و رسم می‌نماییم.

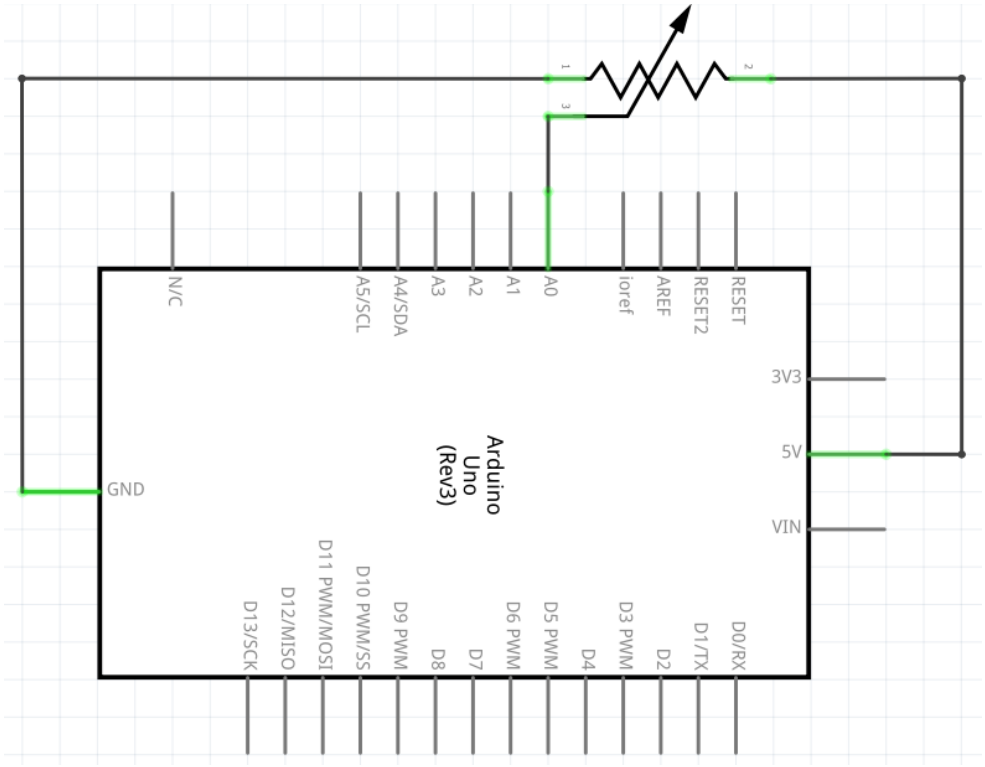
اولین گام در طراحی این مدار، مشخص نمودن مقادیر خازن و مقاومت فیلتر پایین‌گذر است. در صورت سوال از ما خواسته شده تا فیلتر به گونه‌ای طراحی شود که در فرکانس ۱۷۵ هرتز، نهایتا سه دسی‌بل افت دامنه داشته باشیم. از روابط مربوط به این نوع از فیلترها داریم:

$$G(\omega) = \frac{1}{1 + jRC\omega}$$

در فرکانس ۱۷۵ هرتز، باید اندازه G برابر ۰.۷۰۷ باشد. همچنین اندازه خازن مشخص و برابر با ۱ میکرو فاراد است. پس مقدار مقاومت برابر است با:

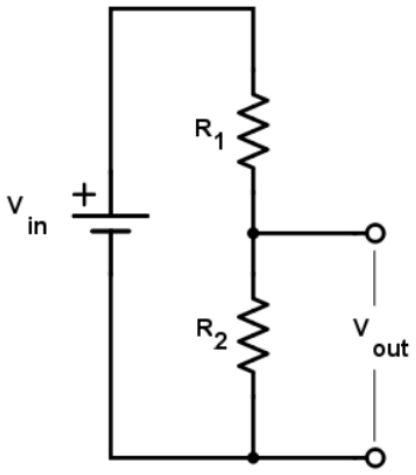
$$R = \sqrt{\frac{\frac{1}{0.707^2} - 1}{(C * \omega)\text{^}2}} \approx 909\ ohm$$

از آنجایی که مقاومتی با این مقدار در اختیار نداریم، باید به کمک یکی از مقاومت‌های متغیر موجود آن را بسازیم. بنابراین مدار تقسیم ولتاژ به صورت شکل ۱ طراحی کرده تا به کمک آن، مقاومت را تنظیم نماییم.



شکل ۱: مدار تنظیم‌کننده مقاومت

مدار تقسیم کننده ولتاژ به صورت شکل ۲ است:



شکل ۲: مدار تقسیم‌کننده ولتاژ

که مقدار ولتاژ خروجی برابر است با:

$$v_{out} = \frac{R_2}{R_1 + R_2} v_{in}$$

یک مقاومت متغیر ۱۰ کیلواهم انتخاب کرده و مقاومت بین پایه وسط و یکی از دو پایه دیگر را $R1$ و با پایه دیگری، $R2$ در نظر می‌گیریم. جمع این دو مقاومت برابر ۱۰ کیلو اهم است. ولتاژ ورودی مدار نیز ۵ ولت است. پس داریم:

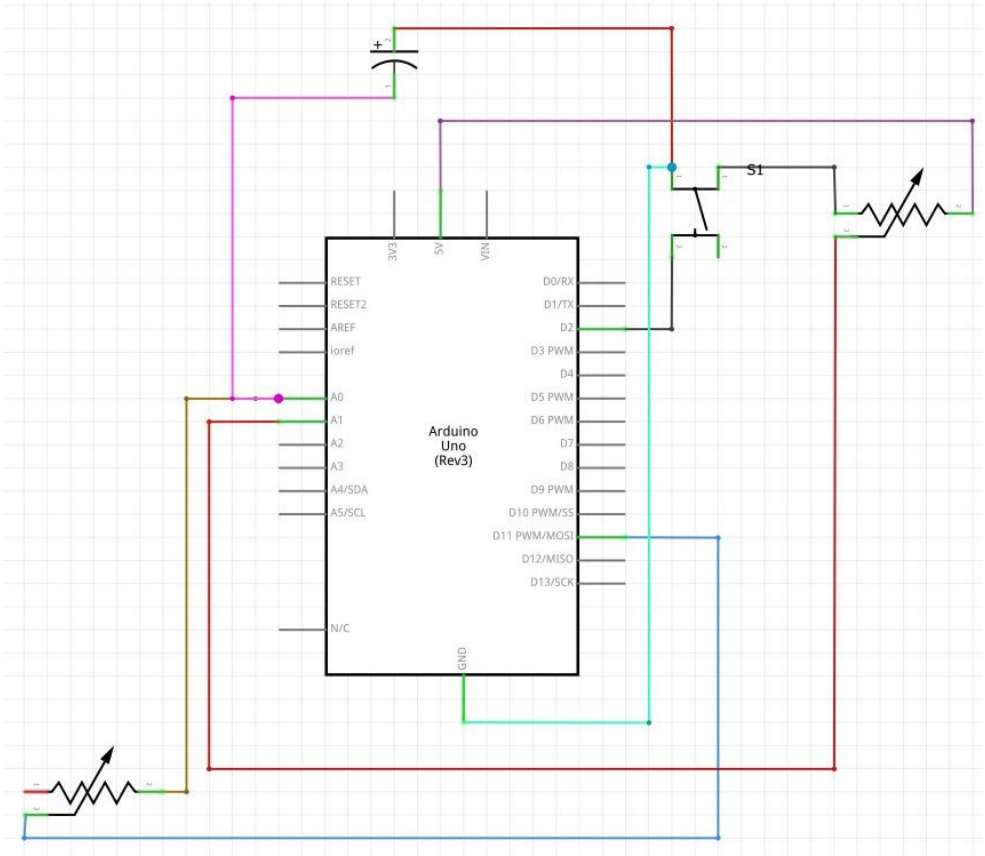
$$v_{out} = \frac{10000 - 909}{10000} * 5 = 4.545\ volt$$

پس آنقدر پیچ موجود بر روی مقاومت را چرخانده تا ولتاژ خروجی به ۴.۵۴ ولت برسد. سپس مقاومت را در مدار اصلی جایگذاری کرده و تنها از بخش $R1$ آن استفاده می‌کنیم.

بخش بعدی، دکمه فشاری است. این پایه را به صورت *Pull up* در کد تعریف می‌کنیم و همانند تکلیف جلسه گذشته به کمک اینتراپت زمانی هر موقع دکمه فشرده شد، مقدار فرکانس جدید خوانده شده و اعمال می‌شود. اما این مقدار فرکانس جدید باید توسط یک مقاومت متغیر دیگر ایجاد شود.

با استفاده از یک مقاومت متغیر و مدار تقسیم ولتاژ، ولتاژ خروجی مدار اندازه گرفته شده و برد آن را میان ۰ تا ۱۷۵ تصویر می‌شود. هر بار دکمه فشرده شد، این مقدار خروجی خوانده شده و فرکانس سیگنال تولیدی برابر با آن می‌شود.

در نهایت، مدار نهایی به صورت شکل ۳ خواهد بود.



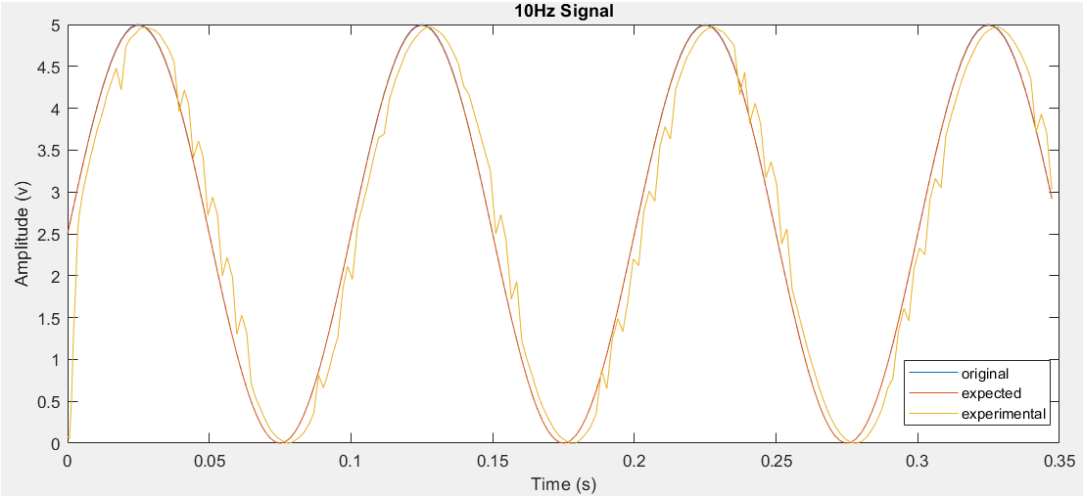
شکل ۳: مدار طراحی‌شده برای سوال اول

بخش بعدی، خواندن داده‌های درگاه سریال در متلب است. به کمک کدی که در پیوست آمده است، هر داده و زمان ثبت آن در متلب خوانده شده و به صورت یک ماتریس ذخیره می‌گردد. علاوه بر سیگنال دریافتی از مدار، در متلب یک سیگنال با فرکانس برابر با سیگنال مدار و بدون وجود فیلتر و یکی هم با در نظر گرفتن فیلتر رسم می‌شود تا تاثیر وجود فیلتر و همچنین تفاوت مدار عملی با روابط تئوری مشخص شود.

لازم به ذکر است که برای استفاده از رنج خروجی کامل آردوینو، سیگنال های سینوسی در بازه ۰ تا ۵ ولت و با میانگین ۲.۵ ولت ایجاد می‌شود.

در ۶ فرکانس مختلف (۱۰، ۴۵، ۸۰، ۱۱۵، ۱۵۰ و ۱۷۵ هرتز) سیگنال‌هایی تولید شده‌اند و در ادامه به بررسی آن‌ها پرداخته‌شده‌است.

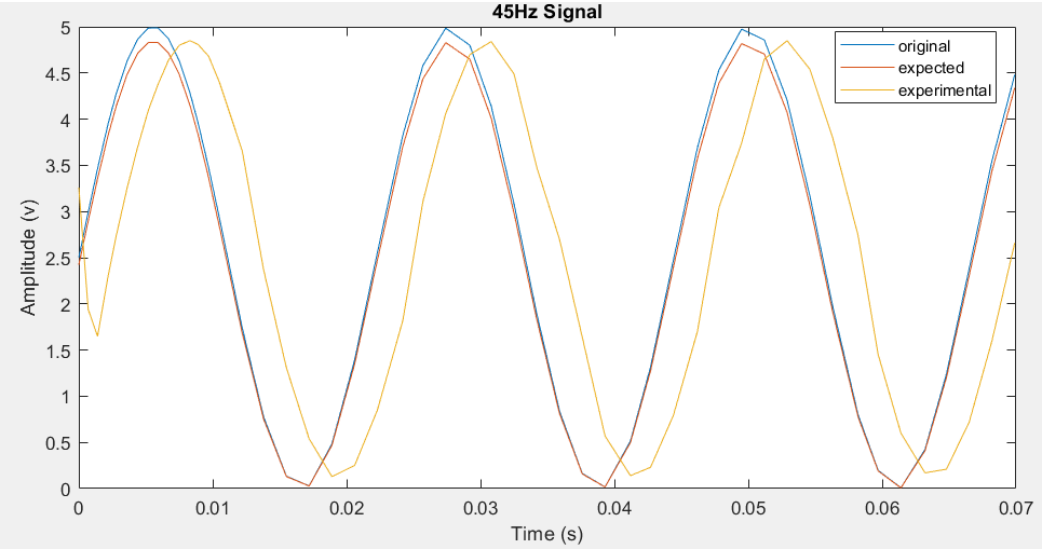
فرکانس ۱۰ هرتز:



شکل ۴: نمودار ولتاژ – زمان سیگنال با فرکانس ۱۰ هرتز

سه سیگنال در شکل ۴ رسم شده است. سیگنال آبی رنگ، سینوس ۱۰ هرتز بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است. مقدار دامنه در این فرکانس برابر ۰.۱۷ دسی‌بل است که در نتیجه بیشینه آن برابر ۴.۹۹ است. و در نهایت سیگنال زرد رنگ خروجی آردوینو است. با استفاده از دستور findpeaks متلب (که بیشینه‌های موضعی را پیدا می‌کند)، مقدار قله‌ی سیگنال خروجی برابر است با ۴.۹۸ (یعنی مقدار افت دامنه ۰.۰۳ دسی‌بل است).

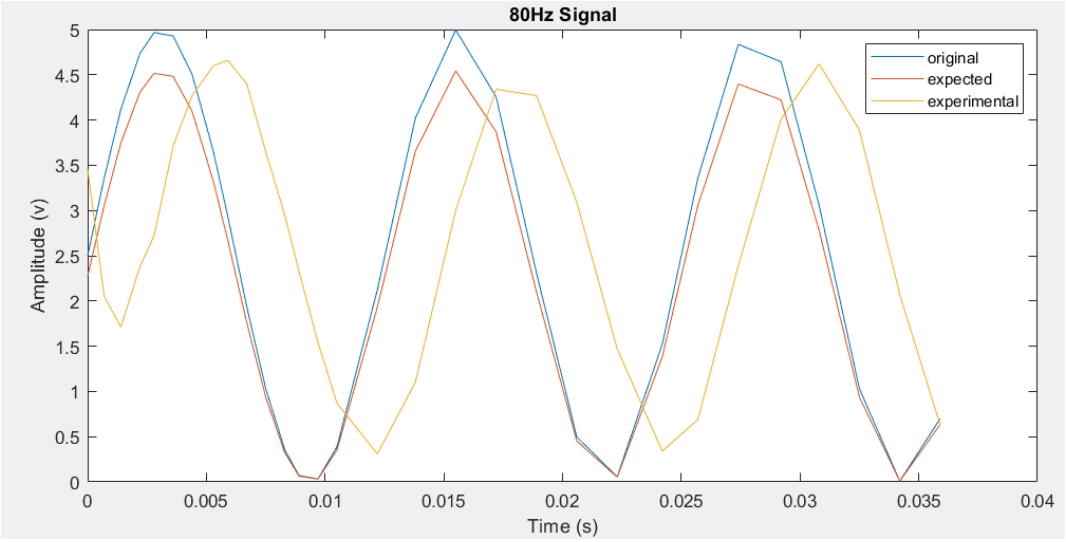
فرکانس ۴۵ هرتز:



شکل ۵: نمودار ولتاژ – زمان سیگنال با فرکانس ۴۵ هرتز

سه سیگنال در شکل ۵ رسم شده است. سیگنال آبی رنگ، سینوس ۴۵ هرتز بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است. مقدار دامنه در این فرکانس برابر ۰.۲۷۸ دسی‌بل است که در نتیجه بیشینه آن برابر ۴.۸۴ است. و در نهایت سیگنال زرد رنگ خروجی آردوینو است. با استفاده از دستور findpeaks متلب (که بیشینه‌های موضعی را پیدا می‌کند)، مقدار قله‌ی سیگنال خروجی برابر است با ۴.۸۵ (یعنی مقدار افت دامنه ۰.۲۶۴ دسی‌بل است).

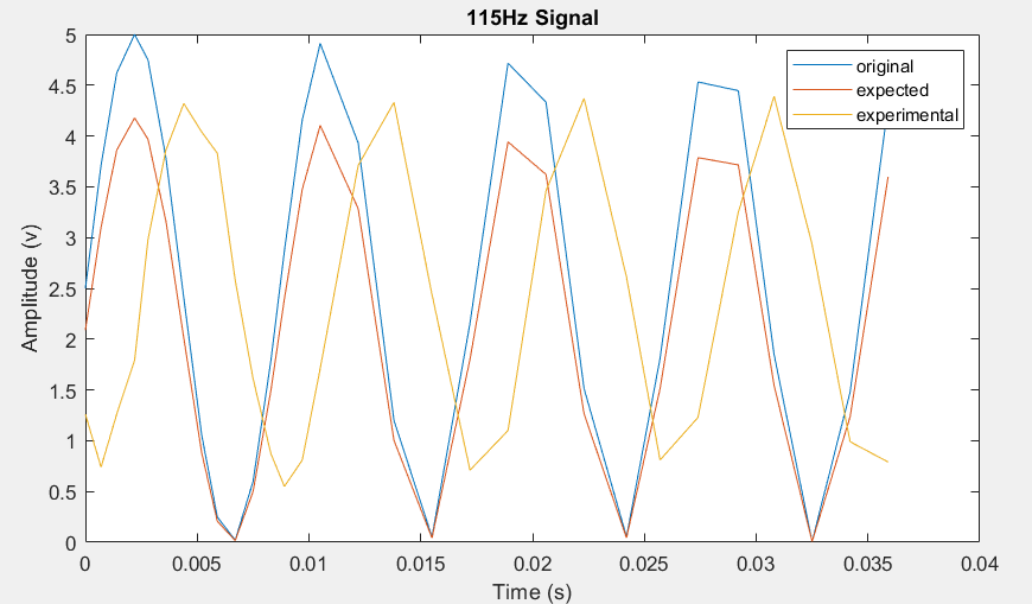
فرکانس ۸۰ هرتز:



شکل ۶: نمودار ولتاژ – زمان سیگنال با فرکانس ۸۰ هرتز

سه سیگنال در شکل۶ رسم شده است. سیگنال آبی رنگ، سینوس ۸۰ هرتز بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است. مقدار دامنه در این فرکانس برابر ۰.۸۲۵ دسی‌بل است که در نتیجه بیشینه آن برابر ۴.۵۴۷ است. و در نهایت سیگنال زرد رنگ خروجی آردوینو است. با استفاده از دستور findpeaks متلب (که بیشینه‌های موضعی را پیدا می‌کند)، مقدار قله‌ی سیگنال خروجی برابر است با ۴.۶۲ (یعنی مقدار افت دامنه ۰.۶۸۶ دسی‌بل است).

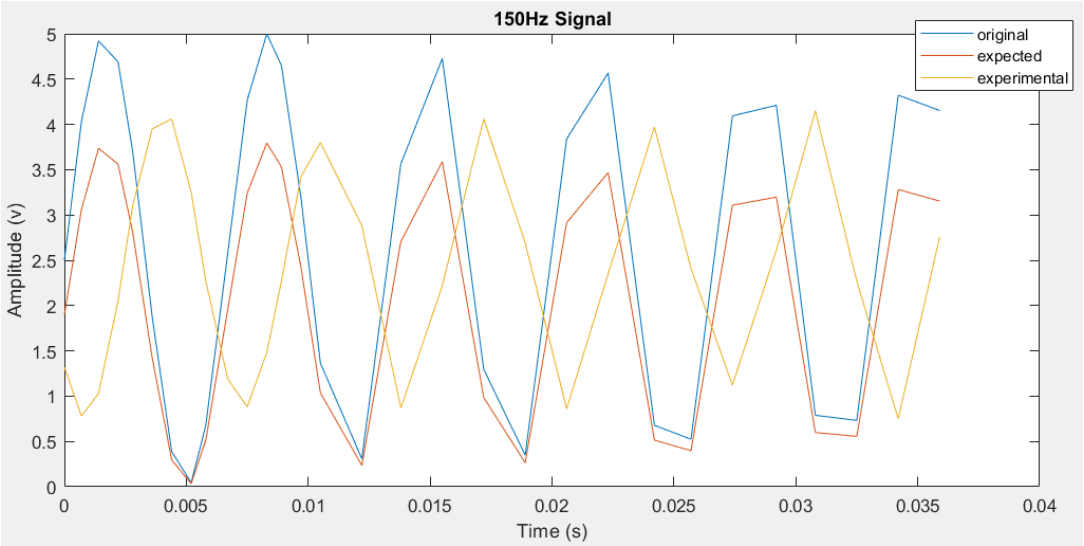
فرکانس ۱۱۵ هرتز:



شکل ۷: نمودار ولتاژ – زمان سیگنال با فرکانس ۱۱۵ هرتز

سه سیگنال در شکل ۷ رسم شده است. سیگنال آبی رنگ، سینوس ۱۱۵ هرتز بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است. مقدار دامنه در این فرکانس برابر ۱.۵ دسی‌بل است که در نتیجه بیشینه آن برابر ۴.۱۷۸ است. و در نهایت سیگنال زرد رنگ خروجی آردوینو است. با استفاده از دستور findpeaks متلب (که بیشینه‌های موضعی را پیدا می‌کند)، مقدار قله‌ی سیگنال خروجی برابر است با ۴.۳۲ (یعنی مقدار افت دامنه ۱.۲۷ دسی‌بل است).

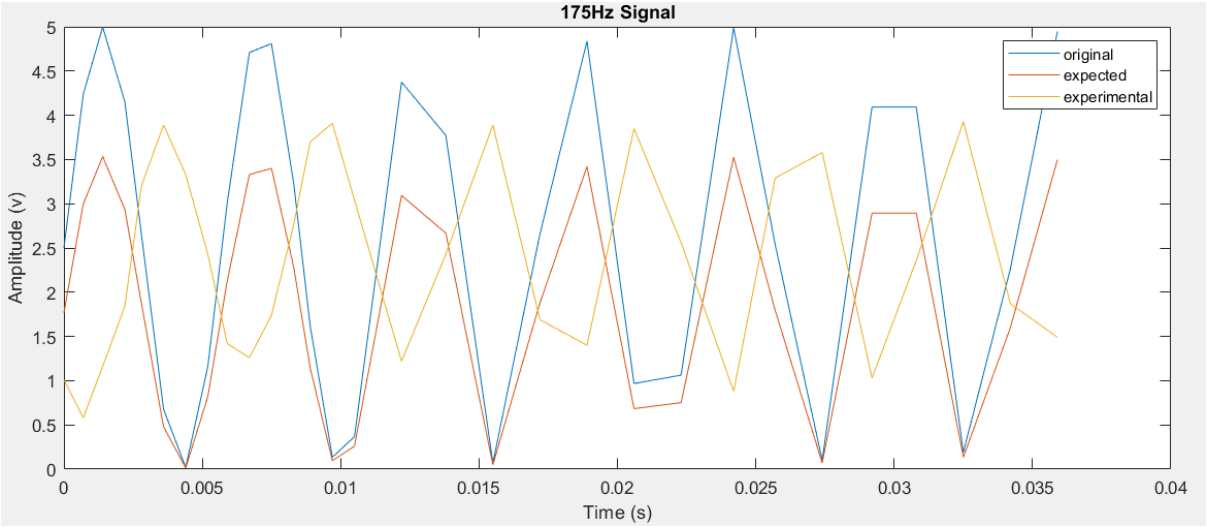
فرکانس ۱۵۰ هرتز:



شکل ۸: نمودار ولتاژ – زمان سیگنال با فرکانس ۱۵۰ هرتز

سه سیگنال در شکل ۸ رسم شده است. سیگنال آبی رنگ، سینوس ۱۵۰ هرتز بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است. مقدار دامنه در این فرکانس برابر ۲.۳۹ دسی‌بل است که در نتیجه بیشینه آن برابر ۳.۷۹۶ است. و در نهایت سیگنال زرد رنگ خروجی آردوینو است. با استفاده از دستور findpeaks متلب (که بیشینه‌های موضعی را پیدا می‌کند)، مقدار قله‌ی سیگنال خروجی برابر است با ۴.۰۶ (یعنی مقدار افت دامنه ۱.۸ دسی‌بل است).

فرکانس ۱۷۵ هرتز:



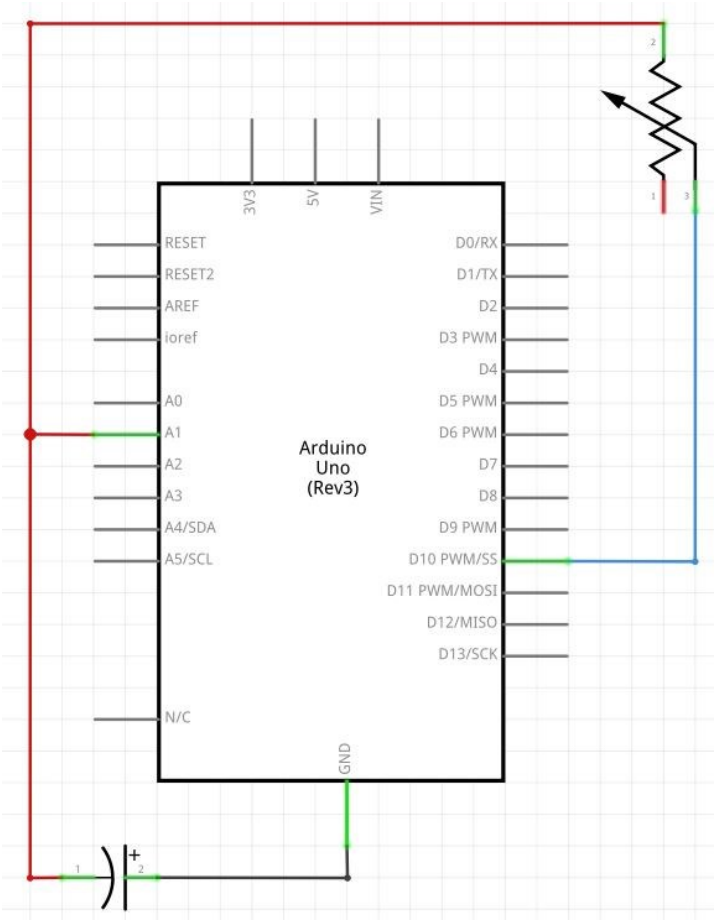
شکل ۹: نمودار ولتاژ – زمان سیگنال با فرکانس ۱۷۵ هرتز

سه سیگنال در شکل ۹ رسم شده است. سیگنال آبی رنگ، سینوس ۱۵۰ هرتز بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است. مقدار دامنه در این فرکانس برابر ۳ دسی‌بل است که در نتیجه بیشینه آن برابر ۳.۵۳۵ است. و در نهایت سیگنال زرد رنگ خروجی آردوینو است. با استفاده از دستور findpeaks متلب (که بیشینه‌های موضعی را پیدا می‌کند)، مقدار قله‌ی سیگنال خروجی برابر است با ۳.۸۸ (یعنی مقدار افت دامنه ۲.۲ دسی‌بل است).

۲- تکلیف دوم

در تکلیف دوم، هدف آن است تا دو سیگنال سینوسی (که یکی فرکانس کم و دیگری فرکانس بالاتری دارد) که بر روی یکدیگر قرار دارند، تولید شود و با کمک فیلتر پایین‌گذر، مولفه فرکانس بالای آن تا حد امکان تضعیف شود.

مدار طراحی شده برای این سوال، به صورت شکل ۱۰ است:



شکل ۱۰: مدار طراحی‌شده برای سوال دوم

ابتدا با توجه به شماره دانشجویی، فرکانس‌های مورد نیاز محاسبه شده است.

$$x = 810696268$$

$$Frequency_{Low}(Hz) = \frac{x \bmod 17}{3} + 1$$

$$Frequency_{High}(Hz) = Frequency_{Low}(Hz) \times 10^{1.146}$$

$$\rightarrow \begin{cases} Frequency_{Low} \approx 5.34\ Hz \\ Frequency_{High} \approx 74.64\ Hz \end{cases}$$

طبق خواسته سوال، باید دو فرکانس گوشه متفاوت تعیین و فیلتر متناظر طراحی شود. با توجه به مقادیر فرکانس‌ها، یک فیلتر با فرکانس گوشه ۶ هرتز و دیگری با فرکانس گوشه ۵۰ هرتز در نظر گرفته می‌شود. مقدار خازن برای فیلتر اول ۱۰۰ میکروفاراد و برای دومی، ۱ میکرو فاراد در نظر گرفته شده تا مقادیر مقاومت با مقاومت متغیر ۱۰ کیلواهمی قابل دسترسی باشد.

مقادیر مقاومت‌ها باید تعیین شود. داریم:

$$R = \frac{1}{2 * \pi * f_0 * C}$$

که بر این اساس، برای فیلتر با فرکانس گوشه ۶ هرتز مقدار مقاومت تقریبا برابر با ۲۶۵ اهم و برای فیلتر با فرکانس گوشه ۵۰ هرتز، مقدار مقاومت تقریبا برابر با ۳۱۸۳ اهم بدست می‌آید. این مقاومت‌ها مانند بخش قبل به کمک مدار تقسیم ولتاژ درست می‌شود.

برای ایجاد دو فرکانس به صورت همزمان، کافی است تا در اینتراپت زمانی، دو سیگنال سینوسی با فرکانس‌های مد نظر با هم جمع شوند و برای آنکه سیگنال خروجی در بازه ۰ تا ۵ ولت قرار گیرد، دامنه هر سیگنال در عدد ۱.۲۵ ضرب شده و نهایتا سیگنال حاصل با ۲.۵ جمع می‌شود.

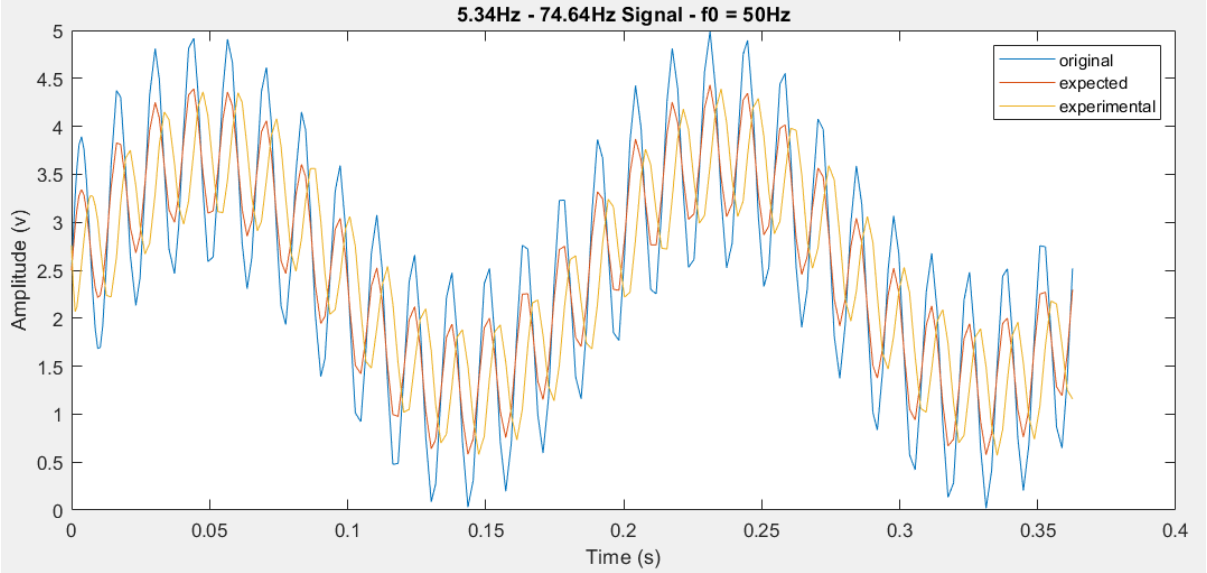
بخش بعدی، خواندن و ذخیره سیگنال تولیدی توسط متلب است. علاوه بر این سیگنال، یک سیگنال سینوسی بدون اعمال فیلتر و یک سیگنال سینوسی دیگر با اعمال فیلتری که توسط روابط تئوری محاسبه شده‌است، تولید و نمایش داده می‌شود تا بتوان مقایسه‌ای انجام داد.

فیلتر پایین‌گذر با فرکانس گوشه‌ی ۵۰ هرتز

برای طراحی این فیلتر، از مقاومت ۱۰ کیلواهم استفاده شده‌است و مقدار آن توسط مدار تقسیم ولتاژ، بر روی عدد ۳۱۸۳ اهم قرار دارد. همچنین از خازن ۱ میکروفاراد استفاده می‌شود. پس داریم:

$$f_0 = \frac{1}{2 * \pi * R * C} \approx 50\ Hz$$

سه سیگنال در شکل ۱۱ رسم شده است. سیگنال آبی رنگ، بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است و در نهایت سیگنال زرد رنگ خروجی آردوینو است.



شکل ۱۱: نمودار ولتاژ – زمان مجموع دو سیگنال با فرکانس‌های ۵.۳۴ و ۷۴.۶ هرتز و فرکانس گوشه‌ی ۵۰ هرتز

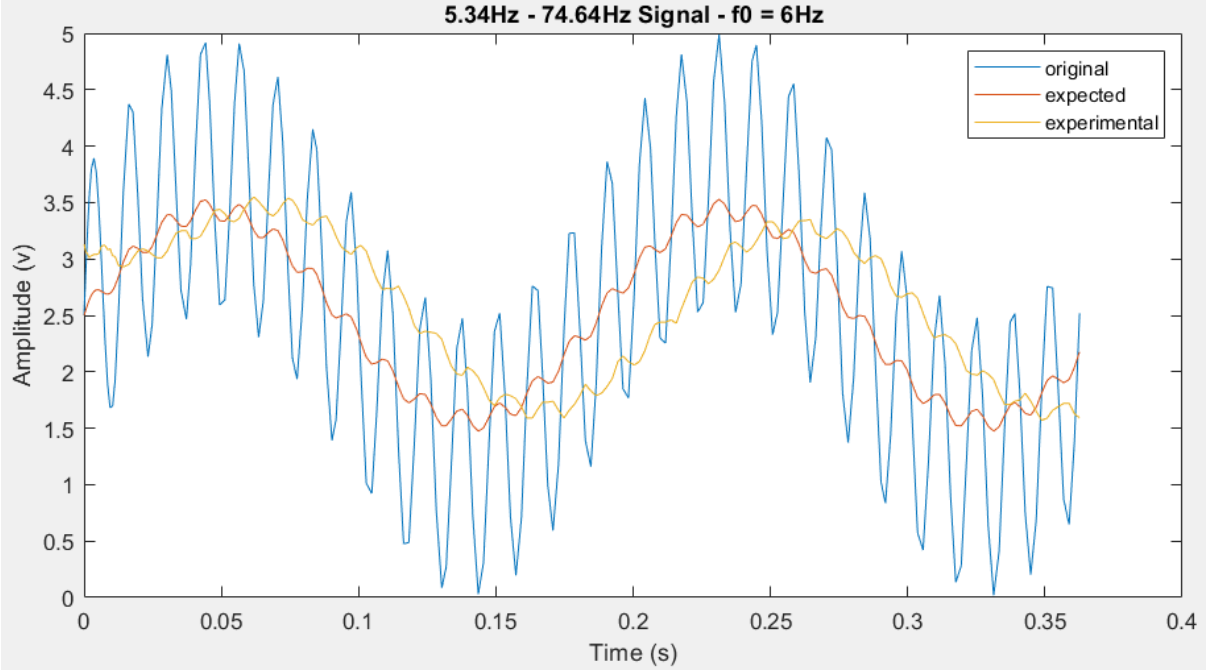
مقدار تئوری تضعیف دامنه‌ای که این فیلتر ایجاد می‌کند، به ترتیب برای سیگنال ۵.۳۴ و ۷۶.۶۴ هرتز برابر با ۰.۰۵ و ۵.۱ دسی‌بل است. این مقادیر برای سیگنال خروجی از مدار به ترتیب ۰.۰۶۹ و ۵.۳۵ دسی‌بل است. این اعداد به کمک سعی و خطا و با ضرب مقادیر مختلف در فرکانس تئوری بدست آمده‌اند.

فیلتر پایین‌گذر با فرکانس گوشه‌ی ۶ هرتز

برای طراحی این فیلتر، از مقاومت ۱۰ کیلواهم استفاده شده‌است و مقدار آن توسط مدار تقسیم ولتاژ، بر روی عدد ۲۶۵ اهم قرار دارد. همچنین از خازن ۱۰۰ میکروفاراد استفاده می‌شود. پس داریم:

$$f_0 = \frac{1}{2 * \pi * R * C} \approx 6\ Hz$$

سه سیگنال در شکل ۱۲ رسم شده است. سیگنال آبی رنگ، بدون فیلتر تصویر شده در بازه ۰ تا ۵ ولت است. سیگنال نارنجی رنگ همان موج اما با اعمال فیلتر است و در نهایت سیگنال زرد رنگ خروجی آردوینو است.



شکل ۱۲: نمودار ولتاژ – زمان مجموع دو سیگنال با فرکانس‌های ۵.۳۴ و ۷۴.۶ هرتز و فرکانس گوشه‌ی ۶ هرتز

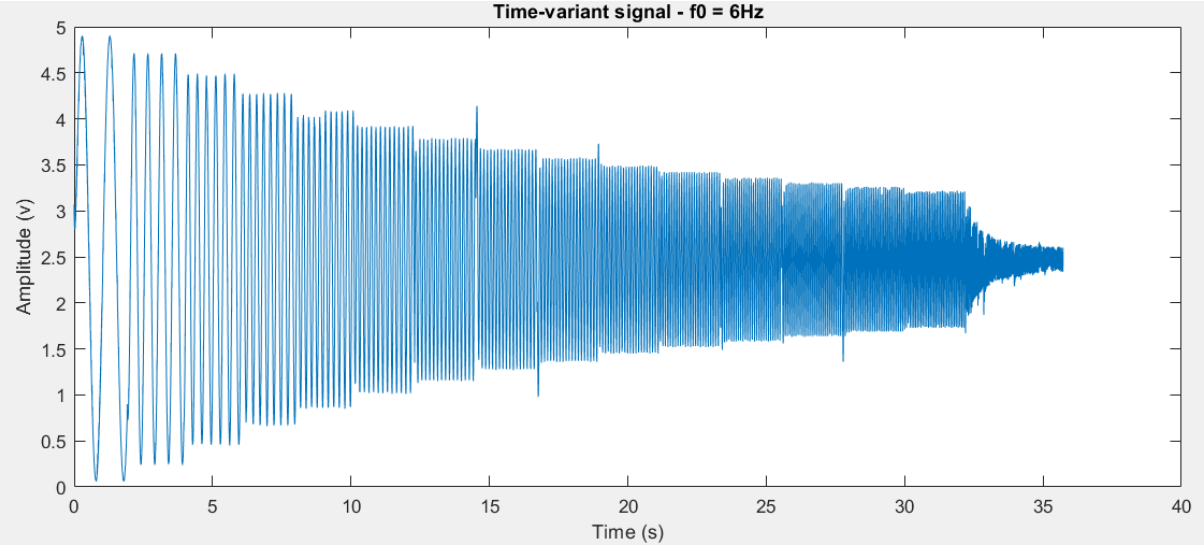
مقدار تئوری تضعیف دامنه‌ای که این فیلتر ایجاد می‌کند، به ترتیب برای سیگنال ۵.۳۴ و ۷۶.۶۴ هرتز برابر با ۲.۵۳ و ۲۱.۹ دسی‌بل است. این مقادیر برای سیگنال خروجی از مدار به ترتیب ۳.۳۴ و ۲۴.۴۳ دسی‌بل است. این اعداد به کمک سعی و خطا و با ضرب مقادیر مختلف در فرکانس تئوری بدست آمده‌اند.

۳- تکلیف سوم

در این تکلیف، باید یکی از فیلترهای طراحی شده در بخش اول یا دوم انتخاب شود و با بررسی پاسخ بدست آمده از حداقل ده سیگنال سینوسی با فرکانس‌های مختلف، مشخصات این فیلتر را بدست آورد.

مدار طراحی شده، مانند مدار بخش قبل (شکل ۱۰) است. مقاومت ۲۶۵ اهم و خازنی با ظرفیت ۱۰۰ میکروفاراد دارد و فرکانس گوشه آن، ۶ هرتز است.

برای تولید سیگنال‌ها بوسیله‌ی آردوینو، از یک حلقه استفاده می‌کنیم. نیاز است تا تعداد سیگنال‌های با فرکانس متفاوت در حوالی فرکانس گوشه بیشتر باشد. پس همه‌ی سیگنال‌های ۱ تا ۱۵ هرتز را تولید کرده و پس از آن از سیگنال ۲۰ تا ۱۷۰ هرتز، تنها مضارب ۱۰ را تولید می‌کنیم. همچنین این نکته در نظر گرفته شده‌است که سیگنال‌های با فرکانس پایین‌تر برای ایجاد چند موج کامل، زمان بیشتری نیاز دارند. ایجاد چند موج برای میانگین گرفتن و کمتر کردن خطا، ضروری است. در نتیجه، برای فرکانس‌های ۱ تا ۱۵ هرتز هر کدام ۱۰۰۰ نقطه تولید شده و برای فرکانس‌های دیگر، هر کدام ۱۰۰ نقطه کافی است. در نهایت ۱۶۶۰۰ نقطه برای این فرکانس‌ها تولید می‌شود. نمودار سیگنال‌های بدست آمده، مانند شکل ۱۳ است.

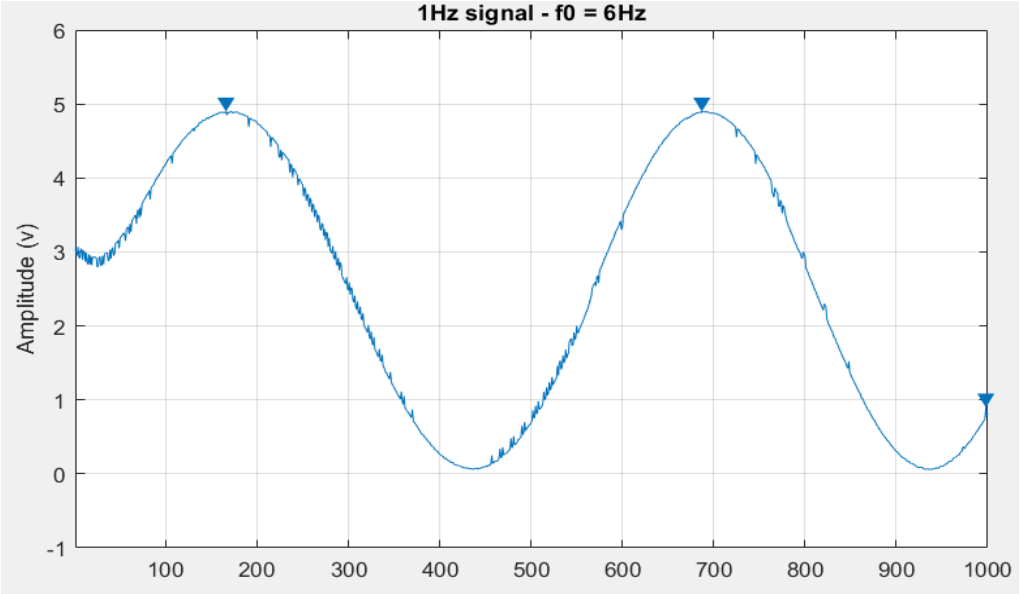


شکل ۱۳: سیگنال تولید شده با فرکانس ۱ تا ۱۷۰ هرتز و فیلتر شده با فرکانس گوشه‌ی ۶ هرتز

ادامه فرایند در متلب طی می‌شود. در بخش ابتدایی، سیگنال تولید شده توسط مدار به همراه زمان آن‌ها خوانده و ذخیره می‌شود. سپس برای هر یک از ۱۵ فرکانس اول که ۱۰۰۰ نقطه داشتند، توسط دستور findpeaks متلب (که نقاط بیشینه‌ی موضعی را پیدا می‌کند)، قله‌های سیگنال‌ها پیدا می‌شود. این تابع

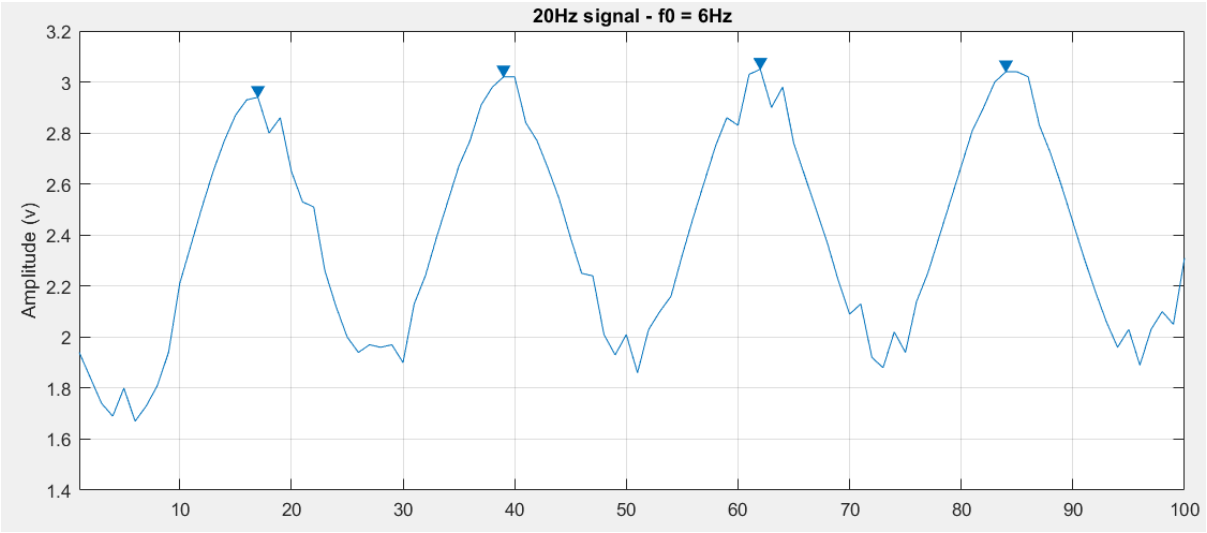
تمام نقاط بیشینه موضعی را پیدا می‌کند؛ پس برای جلوگیری از اشتباه و یافتن قله‌های مربوط به نویز، حداقل فاصله بین دو قله به صورت تجربی رو عدد ۳۰۰ نقطه در نظر گرفته شده‌است.

همان‌طور از که از نمودار شکل ۱۳ مشاهده می‌شود، به هنگام عوض شدن فرکانس، به صورت ناگهانی افزایش داریم. از نقاط می‌تواند در تشخیص یافتن قله مشکل ایجاد کند. میانگین گرفتن را نیز مختل می‌کند. پس به‌جای میانگین، از میانه استفاده شده است. هر سیگنال حداقل چند قله دارد و میانه بهترین جواب را خواهد داد. در شکل ۱۴، قله‌های سیگنال ۱ هرتزی پیدا شده است.



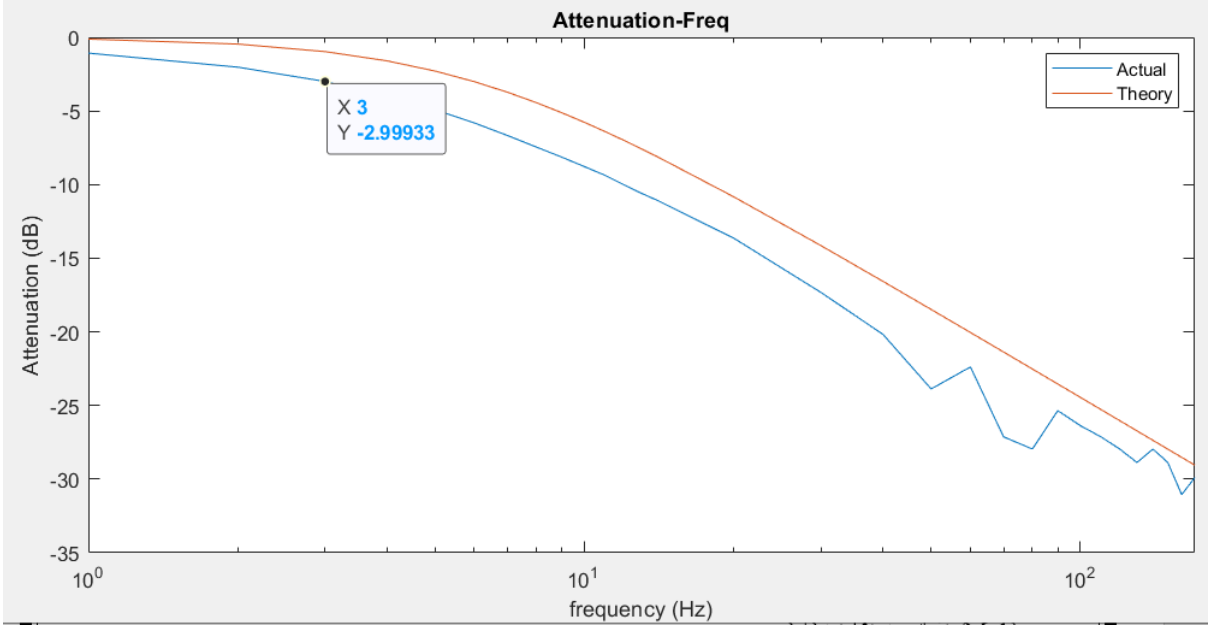
شکل ۱۴: یافتن نقاط پیک سیگنال ۱ هرتز

عمل مشابهی بر روی فرکانس‌های ۲۰ تا ۱۷۰ هرتز انجام شده‌است. با این تفاوت که حداقل فاصله نقاط بیشینه در دستور findpeaks، بر روی ۱۵ تنظیم شده‌است. تصویر ۱۵، این فرایند را نشان می‌دهد.



شکل ۱۵: یافتن نقاط پیک سیگنال ۲۰ هرتز

پس از یافتن قله‌های هر سیگنال با فرکانس متفاوت، نمودار لگاریتمی به شکل ۱۶ رسم می‌شود که محور افقی آن نمایانگر فرکانس سیگنال و محور عمودی نیز مقدار افت دامنه را در آن فرکانس نشان می‌دهد.



شکل ۱۶: پاسخ فرکانسی تئوری و عملی فیلتر مورد استفاده

همان‌طور که دیده می‌شود، فرکانس گوشه اندکی از ۳ هرتز بالاتر خواهد بود که با فرکانس گوشه‌ی سیستم که ۶ هرتز است، تفاوت دارد. همچنین پاسخ فرکانسی که از رابطه تئوری بدست می‌آید نیز رسم شده‌است.

علل وجود خطا در نتایج:

همان‌طور که مشاهده شد، در همه‌ی بخش‌ها میان مقادیر تئوری و عملی تفاوت‌هایی وجود دارد. به تعدادی از این علل اشاره شده‌است:

۱- مقادیر مقاومت‌ها دقیق نیستند؛ اولا هر مقاومت خطایی دارد. ثانیا در تنظیم آن‌ها توسط مدار مقسم ولتاژ، نویزی وجود دارد که مانع از رسیدن به مقدار صحیح مقاومت است.

۲- مقادیر خازن‌ها نیز کاملا دقیق نیست و خطا دارد.

۳- تایم‌استمپ‌هایی که گرفته می‌شود، از زمان کامپیوتر استفاده می‌کند و نه خود کلاک آردوینو.

۴- نویز همیشه وجود دارد و باعث ایجاد خطا است.

۵- در محاسبه مقادیر قله‌ها، میانگین یا میانه باعث ایجاد خطاست.

۴- ضمائم

Setting Resistor

```
/* This is a program for setting Resistor*/

// Defining pins
int res = A1;

// Value is output voltage , R is for R1 in voltage divider circuit
float Value , R;

void setup() {
  Serial.begin(9600);
  pinMode(res, INPUT);

}

void loop() {
  Value = analogRead(res);
  Value = Value*5.00/1023.00;    // measures output voltage
  R = 10000 - 2000 * Value;      // Measures R1 in voltage divider circuit
  Serial.println(R);
  delay(100);
}
```

First Assignment:

Arduino

```
#include <TimerOne.h>
#define PWMOUT 11
#define ADC_CHANNEL A0
#define res A1
#define record 2
float Value;

// Debouncing variables
volatile long t = 0;
int debouncingTime = 200;

float SINE_FREQ = 10;
// Output should be in 0-5 volt range
float SINE_COEFF, Amp=2.5/5*255, Offset=2.5/5*255;
int DUTYCYCLE, SAMPLE;
long TIMESTAMP, counter = 0;

void setup() {
  pinMode(res, INPUT);
  pinMode (record, INPUT_PULLUP);

  // Sets Timer 2 PWM frequency to 31KHz
  TCCR2B = TCCR2B & B11111000 | B00000001;

  pinMode(PWMOUT, OUTPUT);
  Serial.begin (57600);
  Timer1.initialize (1000);
  Timer1.attachInterrupt (PWM_DT);
  attachInterrupt (digitalPinToInterrupt(record), buttonISR, FALLING);
}
```

```
void loop() {
  SINE_COEFF = SINE_FREQ*TWO_PI/1000.000;
  TIMESTAMP = micros();                                    // Timestamp for
each sample to be analyzed later
  SAMPLE = analogRead(ADC_CHANNEL);                        // Using Arduino's
ADC to read the PWM output voltage
  Serial.print (TIMESTAMP/100); Serial.print("\t");        // Printing
timestamp
  Serial.println (SAMPLE/1023.00*5);
}


// Function for generating PWM signal
void PWM_DT(){
  DUTYCYCLE = Amp * sin(SINE_COEFF * counter) + Offset;
  analogWrite(PWMOUT, DUTYCYCLE);
  counter++;
}


// Reading signal everytime the button is pushed
void buttonISR() {
  //  Debouncing
  if (millis()-t>debouncingTime) {
  SINE_FREQ = analogRead(res);
  SINE_FREQ = SINE_FREQ/1023*175;
  t = millis();
  }
}
```

Matlab

```
clear; close all; clc;

%% Circuit properties
freq = 175; %Hz
R = 909.73; %ohm
C = 1e-6;   %F
cte = 1/sqrt(1+(R*C*freq*2*pi)^2);

%% Receiving data
SampleNumber = 1000;
a=strings(SampleNumber,2);
s = serialport ("COM5", 57600);
configureTerminator(s,"CR/LF");

for i = 1:SampleNumber
a(i,:) = strsplit(readline(s));
end

b = double(a);
clear s

% timestamps
t = b(:,1)/10000;

% Theoritical signal without filter
original_signal = 2.5 * sin(freq*2*pi*t) + 2.5;

% Theoritical signal with filter
expected_signal = cte * (2.5 * sin(freq*2*pi*t) + 2.5);

% Signal read from Arduino
experimental_signal = b(:,2);


%% Plotting
figure;
plot(t,original_signal)
hold on
plot(t,expected_signal)
hold on
plot(t,experimental_signal)
hold off
legend('original','expected','experimental')
xlabel('Time (s)')
ylabel('Amplitude (v)')
title('175Hz Signal')
```

Second Assignment

Arduino

```
#include <TimerOne.h>

#define PWMOUT_1 11

#define ADC_CHANNEL A0

float SINE_FREQ_1 = 5.34;     // First signal Frequency

float SINE_FREQ_2 = 74.645;   // Second signal Frequency

float SINE_COEFF_1 = SINE_FREQ_1 * TWO_PI/1000.000 ,SINE_COEFF_2 =
SINE_FREQ_2 * TWO_PI/1000.000;


// Output should be in 0-5 volt range

float Amp=1.25/5*255, Offset=2.5/5*255;

int DUTYCYCLE , SAMPLE;

long TIMESTAMP, counter = 0;


void setup() {

// Sets Timer 2 PWM frequency to 31KHz

  TCCR2B = TCCR2B & B11111000 | B00000001;

  pinMode(PWMOUT_1, OUTPUT);

  Serial.begin (57600);

  Timer1.initialize (1000);

  Timer1.attachInterrupt (PWM_DT);

}


void loop() {

  TIMESTAMP = micros();                                  // Timestamp for
each sample to be analyzed later

  SAMPLE = analogRead(ADC_CHANNEL);                      // Using Arduino's
ADC to read the PWM output voltage

  Serial.print (TIMESTAMP/100); Serial.print("\t");       // Printing
timestamp

  Serial.println (SAMPLE/1023.00*5);

}
```

```
// Function for generating PWM signal
void PWM_DT(){
  // Adding two signals
  DUTYCYCLE = Amp * (sin(SINE_COEFF_1 * counter)+ sin(SINE_COEFF_2 *
counter)) + Offset;
  analogWrite(PWMOUT_1, DUTYCYCLE);
  counter++;
}
```

Matlab

```
clear; close all; clc;

%% Circuit properties
freq1 = 5.34;        %Hz
freq2 = 74.645;      %Hz
R = 265.258;         %Ohm
C = 1e-4;            %F
cte1 = 1/sqrt(1+(R*C*freq1*2*pi)^2);
cte2 = 1/sqrt(1+(R*C*freq2*2*pi)^2);

%% Receiving data
SampleNumber = 1000;
a=strings(SampleNumber,2);
s = serialport ("COM5", 57600);
configureTerminator(s,"CR/LF");

for i = 1:SampleNumber
a(i,:) = strsplit(readline(s));
end

b = double(a);
clear s

% timestamps
t = b(:,1)/10000;

% Signal read from Arduino
experimental_signal = b(:,2);


% Theoritical signal without filter
original_signal = 1.25 * (sin(freq1*2*pi*t) + sin(freq2*2*pi*t))
+ 2.5;

% Theoritical signal with filter
expected_signal = 1.25 * (cte1 * sin(freq1*2*pi*t) + cte2 *
sin(freq2*2*pi*t)) + 2.5;

figure;
plot(t,original_signal)
%hold on

%% Plotting
figure
plot(t,original_signal)
hold on
plot(t,expected_signal)
hold on
plot(t,experimental_signal)
```

```
hold off
legend('original','expected','experimental')
xlabel('Time (s)')
ylabel('Amplitude (v)')
title('5.34Hz - 74.64Hz Signal - f0 = 6Hz')
```

Third Assignment

Arduino

```
#include <TimerOne.h>
#define PWMOUT 11
#define ADC_CHANNEL A0

float SINE_FREQ;
float SINE_COEFF;

// Output should be in 0-5 volt range
float Amp = 2.5 /5*255, Offset = 2.5/5*255;

int DUTYCYCLE , SAMPLE;
long TIMESTAMP, counter = 0;

void setup() {

  TCCR2B = TCCR2B & B11111000 | B00000001;              // Sets Timer
2 PWM frequency to 31KHz

  pinMode(PWMOUT, OUTPUT);
  Serial.begin (57600);
  Timer1.initialize (1000);
  Timer1.attachInterrupt (PWM_DT);
}
```

```
void loop() {
  // For 1-15 Hz, generates 1000 data points for each
  for (int i = 1 ; i <= 15 ; i++)
  {
    SINE_FREQ = i;
    SINE_COEFF = SINE_FREQ * TWO_PI/1000.000;
    for (int j = 0 ; j < 1000 ; j++)
    {
      // Timestamp for each sample to be analyzed later
      TIMESTAMP = micros();
      // Using Arduino's ADC to read the PWM output voltage
      SAMPLE = analogRead(ADC_CHANNEL);
      Serial.print (TIMESTAMP/100); Serial.print("\t");
      Serial.println (SAMPLE/1023.00*5);
    }
  }
  // For 20:10:170 Hz, generates 100 data points for each
    for (int i = 20 ; i < 171 ; i+=10)
  {
    SINE_FREQ = i;
    SINE_COEFF = SINE_FREQ * TWO_PI/1000.000;
    for (int j = 0 ; j < 100 ; j++)
    {
      TIMESTAMP = micros();
      SAMPLE = analogRead(ADC_CHANNEL);
      Serial.print (TIMESTAMP/100); Serial.print("\t");
      Serial.println (SAMPLE/1023.00*5);
    }
  }
}
```

```
// Function for generating PWM signal
void PWM_DT(){
  DUTYCYCLE = Amp * sin(SINE_COEFF * counter) + Offset;
  analogWrite(PWMOUT, DUTYCYCLE);
  counter++;
}
```

Matlab

```
clear; close all; clc;

z = [];              %Holds Freqs and Responses
freq = 20:10:170;

%% Receiving data
SampleNumber = 16600;
a=strings(SampleNumber,2);
s = serialport ("COM5", 57600);
configureTerminator(s,"CR/LF");

for i = 1:SampleNumber
a(i,:) = strsplit(readline(s));
end

b = double(a);
clear s

% timestamps
t = b(:,1)/10000;

% Signal read from Arduino
experimental_signal = b(:,2);


%% Plotting
figure
plot(t,experimental_signal)
xlabel('Time (s)')
ylabel('Amplitude (v)')
title('Time-variant signal - f0 = 6Hz')

% For 1-15Hz, find peaks ans stores them
for j = 0:14
    s = experimental_signal(1000*j+1:1000*(j+1));
    pks = findpeaks(s,'MinPeakDistance',300);
    z = [z; [ j , median(pks) ]];
end


% For 20:10:170 Hz, find peaks ans stores them
for j = 1:numel(freq)
    s = experimental_signal(15001+100*(j-1):15000+100*j);
    pks = findpeaks(s,'MinPeakDistance',15);
    z = [z; [ freq(j) , median(pks) ]];
end
```

```
% Frequency response
figure
 %Actual signal freq response
semilogx(z(:,1),20*log10((z(:,2)-2.5)/2.5))
ylabel('Attenuation (dB)')
xlabel('frequency (Hz)')
title('Attenuation-Freq')

hold on
%Theoritical signal freq response
semilogx(z(:,1),20*log10((1./sqrt(1+(z(:,1)/6).^2))))
legend('Actual','Theory')
```