باسمه تعالی

دانشکده مهندسی مکانیک

پردیس دانشکده‌های فنی

دانشگاه تهران

گزارش تکالیف

جلسه چهارم

درس سیستم‌های اندازه‌گیری کارشناسی

دکتر صدیقی

مهدی عبداله چالکی (۸۱۰۶۹۶۲۶۸)

نیم‌سال دوم

سال تحصیلی ۰۰-۹۹

۱- تکلیف اول

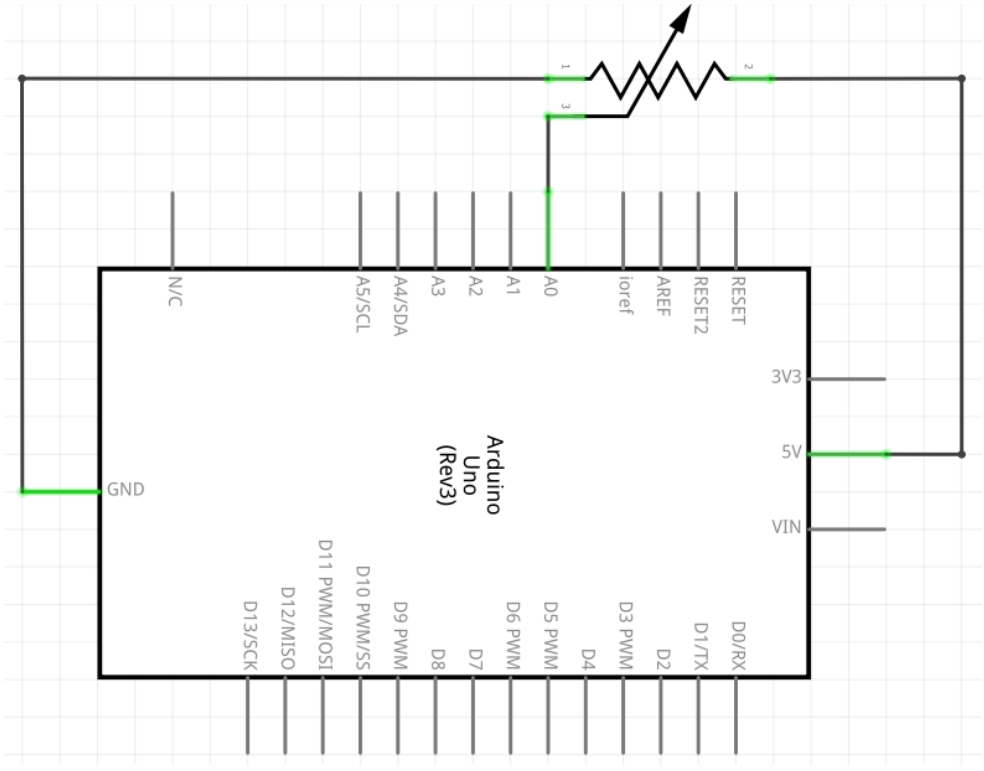
در تکلیف اول هدف طراحی مداری است که بتواند یک سیگنال ۲۳ هرتز تولید کرده و با فرکانس‌های مختلف از آن نمونه‌برداری کند تا بتوان اثر aliasing را مشاهده کرد.

اولین گام در طراحی این مدار، مشخص نمودن مقادیر خازن و مقاومت فیلتر پایین‌گذر است. با توجه به صورت سوال و اینکه فرکانس ۲۳ هرتز باید تولید شود، فرکانس گوشه بر روی ۳۰ هرتز قرار داده شده‌است.

در فرکانس ۳۰ هرتز، اندازه G باید برابر ۰.۷۰۷ باشد. همچنین اندازه‌ی خازن مشخص و برابر با ۱ میکرو فاراد است. پس مقدار مقاومت برابر است با:

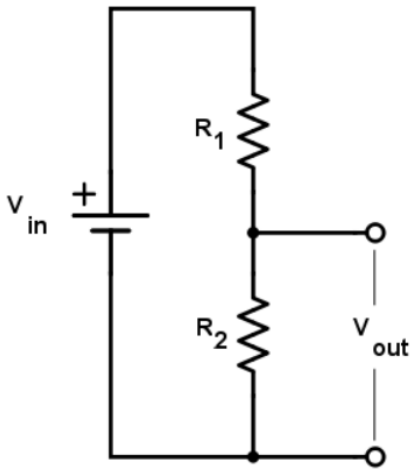
$$R = \sqrt{\frac{\frac{1}{0.707^2} - 1}{(C * \omega)\text{^}2}} \approx 5305\ ohm$$

از آنجایی که مقاومتی با این مقدار وجود ندارد، باید به کمک یکی از مقاومت‌های متغیر موجود، چنین مقاومتی ساخته‌شود. بنابراین مدار تقسیم ولتاژ به صورت شکل ۱ طراحی شده تا به کمک آن، مقاومت تنظیم شود.



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:

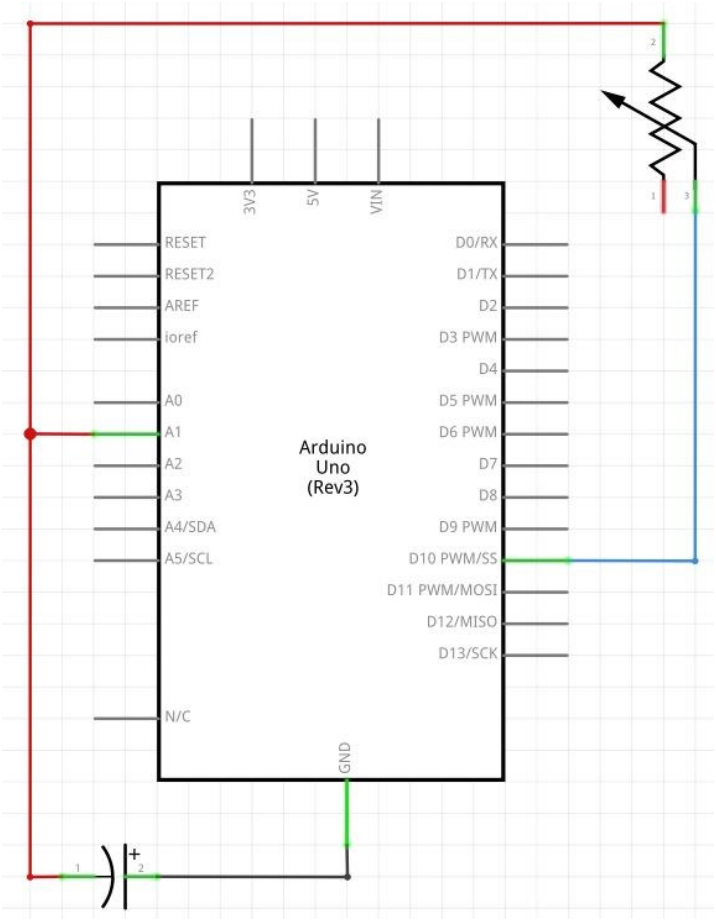
$$v_{out} = \frac{R_2}{R_1 + R_2} v_{in}$$

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

$$v_{out} = \frac{10000 - 5305}{10000} * 5 = 0.4695\ volt$$

آنقدر پیچ موجود بر روی مقاومت را چرخانده تا ولتاژ خروجی به ۰.۴۶۹۵ ولت برسد. سپس مقاومت در مدار اصلی جایگذاری شده و تنها از بخش $R1$ آن استفاده می‌شود.

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



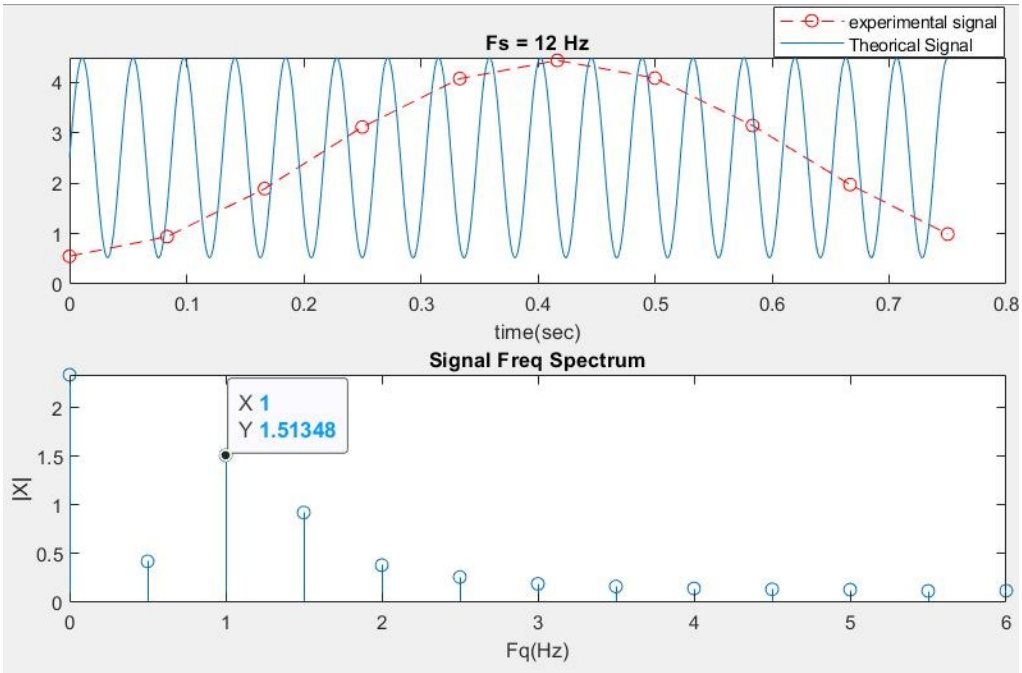
شکل ۳: مدار طراحی‌شده برای سوال اول

بخش بعدی، خواندن داده‌های درگاه سریال در متلب است. به کمک کدی که در پیوست آمده است، هر داده 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 با در نظر گرفتن فیلتر رسم می‌شود تا 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 (۱۲، ۲۰، ۲۸، ۴۶، ۹۲ و ۱۲۰ هرتز) از سیگنال تولید شده نمونه برداری شده‌است و در ادامه به بررسی آن‌ها پرداخته‌شده‌است.

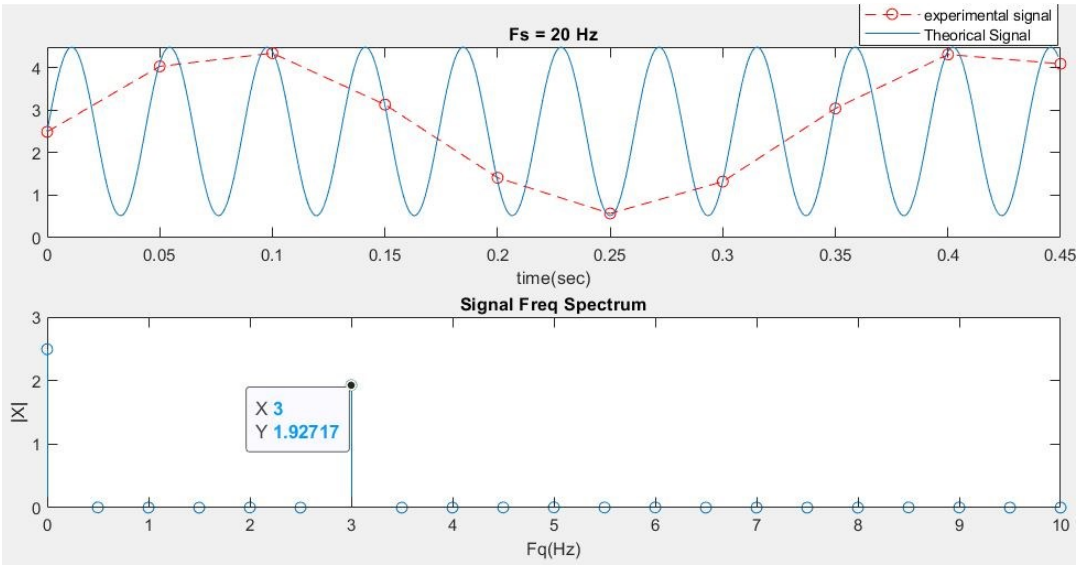
فرکانس ۱۲ هرتز:



شکل ۴: نمودار ولتاژ – زمان و فرکانس سیگنال نمونه برداری شده با فرکانس ۱۰ هرتز

در بخش بالایی شکل ۴، نمودار سیگنالی که از آردوینو گرفته شده‌است و نیز یک سیگنال سینوسی فیلترشده با فیلتر تئوری مشابه فیلتر مدار نشان داده شده‌است. در بخش پایینی نیز محتوای فرکانسی همان سیگنال رسم شده‌است که مشاهده می‌شود فرکانس سیگنال بدست آمده، ۱ هرتز است و پدیده‌ی *aliasing* رخ داده‌است.

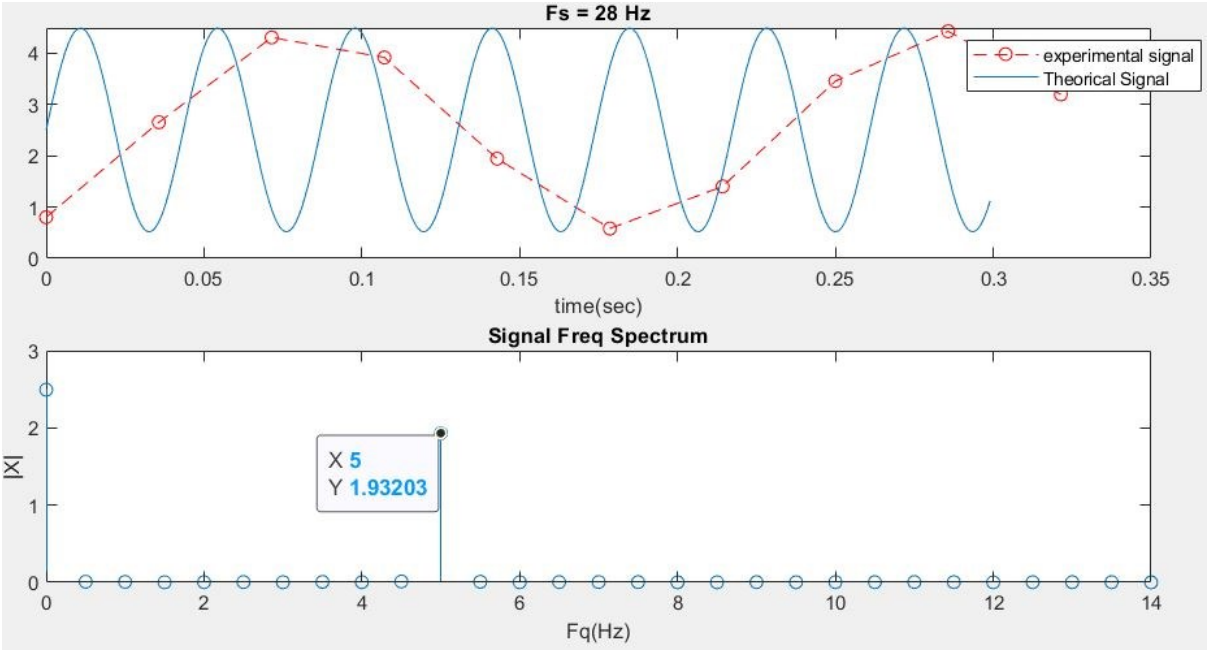
فرکانس ۲۰ هرتز:



شکل ۵: نمودار ولتاژ – زمان و فرکانس سیگنال نمونه برداری شده با فرکانس ۲۰ هرتز

در بخش بالایی شکل ۵، نمودار سیگنالی که از آردوینو گرفته شده‌است و نیز یک سیگنال سینوسی فیلترشده با فیلتر تئوری مشابه فیلتر مدار نشان داده شده‌است. در بخش پایینی نیز محتوای فرکانسی همان سیگنال رسم شده‌است که مشاهده می‌شود فرکانس سیگنال بدست آمده، ۳ هرتز است و پدیده‌ی *aliasing* رخ داده‌است.

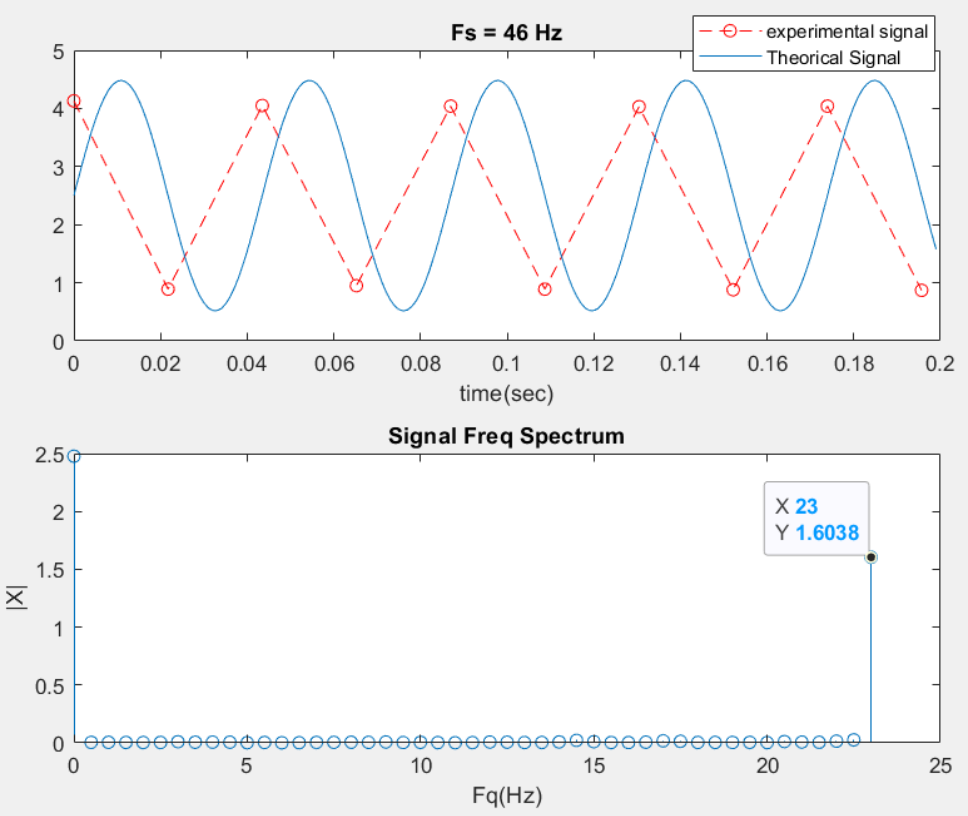
فرکانس ۲۸ هرتز:



شکل ۶: نمودار ولتاژ – زمان و فرکانس سیگنال نمونه برداری شده با فرکانس ۲۸ هرتز

در بخش بالایی شکل ۶، نمودار سیگنالی که از آردوینو گرفته شده‌است و نیز یک سیگنال سینوسی فیلترشده با فیلتر تئوری مشابه فیلتر مدار نشان داده شده‌است. در بخش پایینی نیز محتوای فرکانسی همان سیگنال رسم شده‌است که مشاهده می‌شود فرکانس سیگنال بدست آمده، ۵ هرتز است و پدیده‌ی *aliasing* رخ داده‌است.

فرکانس ۴۶ هرتز:

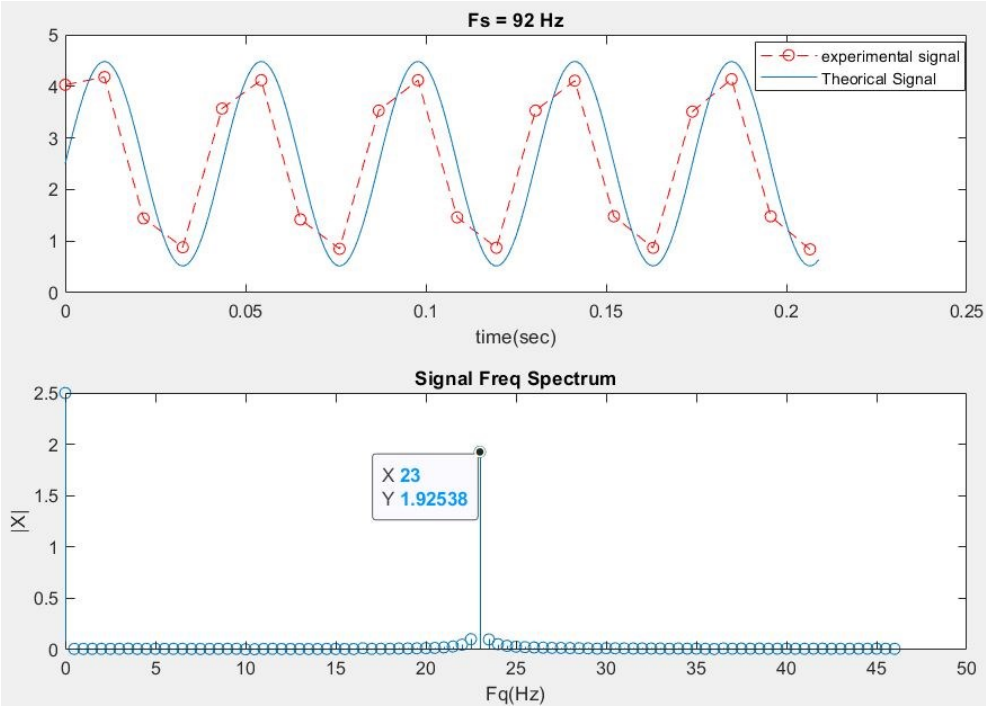


شکل ۷: نمودار ولتاژ – زمان و فرکانس سیگنال نمونه برداری شده با فرکانس ۴۶ هرتز

در بخش بالایی شکل ۷، نمودار سیگنالی که از آردوینو گرفته شده‌است و نیز یک سیگنال سینوسی فیلترشده با فیلتر تئوری مشابه فیلتر مدار نشان داده شده‌است. در بخش پایینی نیز محتوای فرکانسی همان سیگنال رسم شده‌است که مشاهده می‌شود فرکانس سیگنال بدست آمده، ۲۳ هرتز است و پدیده‌ی *aliasing* رخ داده‌است.

تنها در صورتی می‌توان از سیگنال درست نمونه‌برداری کرد که اختلاف فاز سیستم تولید سیگنال و سیستم نمونه‌برداری با یکدیگر ۹۰ درجه باشد. این امر تقریبا غیر ممکن است و به همین خاطر است که معمولا از فرکانس نمونه‌برداری بسیار بیشتر از فرکانس نایکوییست استفاده می‌شود.

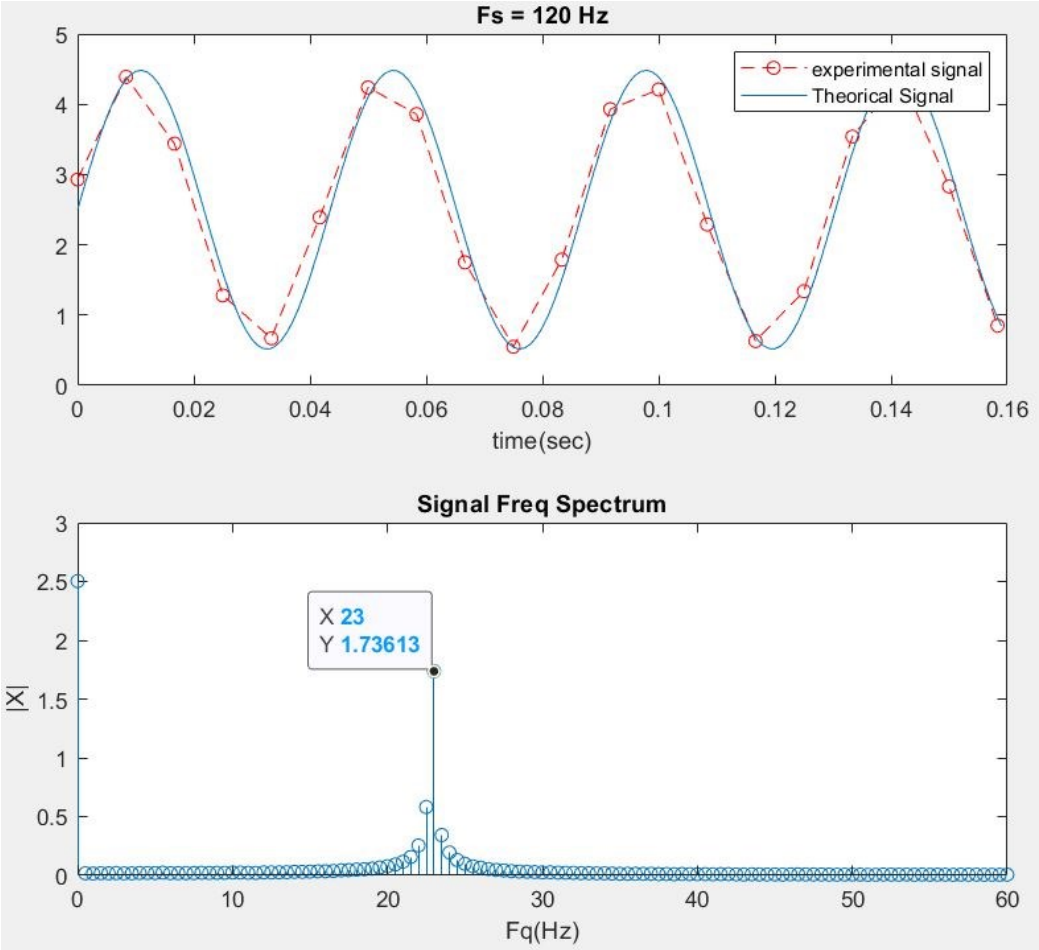
فرکانس ۹۲ هرتز:



شکل ۸: نمودار ولتاژ – زمان و فرکانس سیگنال نمونه برداری شده با فرکانس ۹۲ هرتز

در بخش بالایی شکل ۸، نمودار سیگنالی که از آردوینو گرفته شده‌است و نیز یک سیگنال سینوسی فیلترشده با فیلتر تئوری مشابه فیلتر مدار نشان داده شده‌است. در بخش پایینی نیز محتوای فرکانسی همان سیگنال رسم شده‌است که مشاهده می‌شود فرکانس سیگنال بدست آمده، ۲۳ هرتز است و پدیده‌ی *aliasing* رخ نداده‌است، چرا که ۹۲ هرتز از دو برابر ۲۳ هرتز (یعنی ۴۶ هرتز) بیشتر است.

فرکانس ۱۲۰ هرتز:



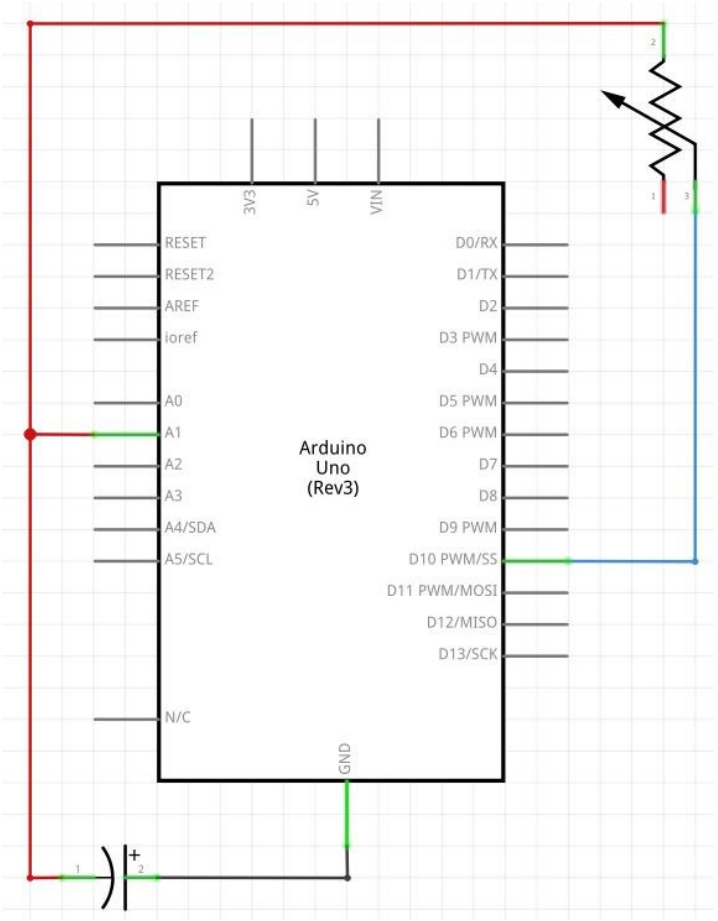
شکل ۹: نمودار ولتاژ – زمان و فرکانس سیگنال نمونه برداری شده با فرکانس ۱۲۰ هرتز

در بخش بالایی شکل ۹، نمودار سیگنالی که از آردوینو گرفته شده‌است و نیز یک سیگنال سینوسی فیلترشده با فیلتر تئوری مشابه فیلتر مدار نشان داده شده‌است. در بخش پایینی نیز محتوای فرکانسی همان سیگنال رسم شده‌است که مشاهده می‌شود فرکانس سیگنال بدست آمده، ۲۳ هرتز است و پدیده‌ی *aliasing* رخ نداده‌است، چرا که ۱۲۰ هرتز از دو برابر ۲۳ هرتز (یعنی ۴۶ هرتز) بیشتر 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، مولفه‌ی فرکانس بالای آن تا حد امکان تضعیف شود. سپس از این سیگنال با فرکانس ثابت نمونه‌برداری شده تا بتوان با استفاده از تبدیل فوریه‌ی سریع، فرکانس سیگنال‌های تشکیل‌دهنده‌ی آن را تعیین کرد. همچنین در این بخش و برای جلوگیری از نشت انرژی در نمودار فرکانس، از پنجره‌ی هنینگ[1] استفاده شده‌است.

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.

[1] Hanning window

$$x = 810696268$$

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

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

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

طبق خواسته سوال، باید یک فرکانس گوشه تعیین و فیلتر متناظر طراحی شود. با توجه به مقادیر فرکانس‌ها، یک فیلتر با فرکانس گوشه‌ی ۱۰ هرتز در نظر گرفته می‌شود. مقدار خازن برای این فیلتر، ۱ میکرو فاراد در نظر گرفته شده است.

مقدار مقاومت باید تعیین شود. داریم:

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

که بر این اساس، برای فیلتر با فرکانس گوشه‌ی ۱۰ هرتز مقدار مقاومت تقریبا برابر با ۱۵۹۱۵ اهم بدست می‌آید. این مقاومت مانند بخش قبل به کمک مدار تقسیم ولتاژ درست می‌شود.

برای ایجاد دو فرکانس به صورت همزمان، کافی است تا در اینتراپت[۲] زمانی، دو سیگنال سینوسی با فرکانس‌های مد نظر با هم جمع شوند و برای آنکه سیگنال خروجی در بازه‌ی ۰ تا ۵ ولت قرار گیرد، دامنه‌ی 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 با اعمال فیلتری که توسط روابط تئوری محاسبه شده‌است، تولید و نمایش داده می‌شود تا بتوان مقایسه‌ای انجام داد.

برای نمونه‌برداری از سیگنال، با توجه به بالاترین فرکانس موجود در مدار (۷۴.۶ هرتز) باید حداقل فرکانس نمونه‌برداری روی عدد ۱۴۹.۲ هرتز تنظیم شود. با در نظر گرفتن این نکته که حافظه‌ی آردوینو گنجایش تقریبا ۸۰۰ عدد را دارد، انتخاب فرکانس نمونه‌برداری بالاتر به زمان داده‌برداری کمتری منتهی می‌شود. پس از فرکانس ۱۵۰ هرتز استفاده می‌شود تا زمان بیشتری بتوان فرکانس را دریافت کرد و به نتایج دقیق‌تری رسید.

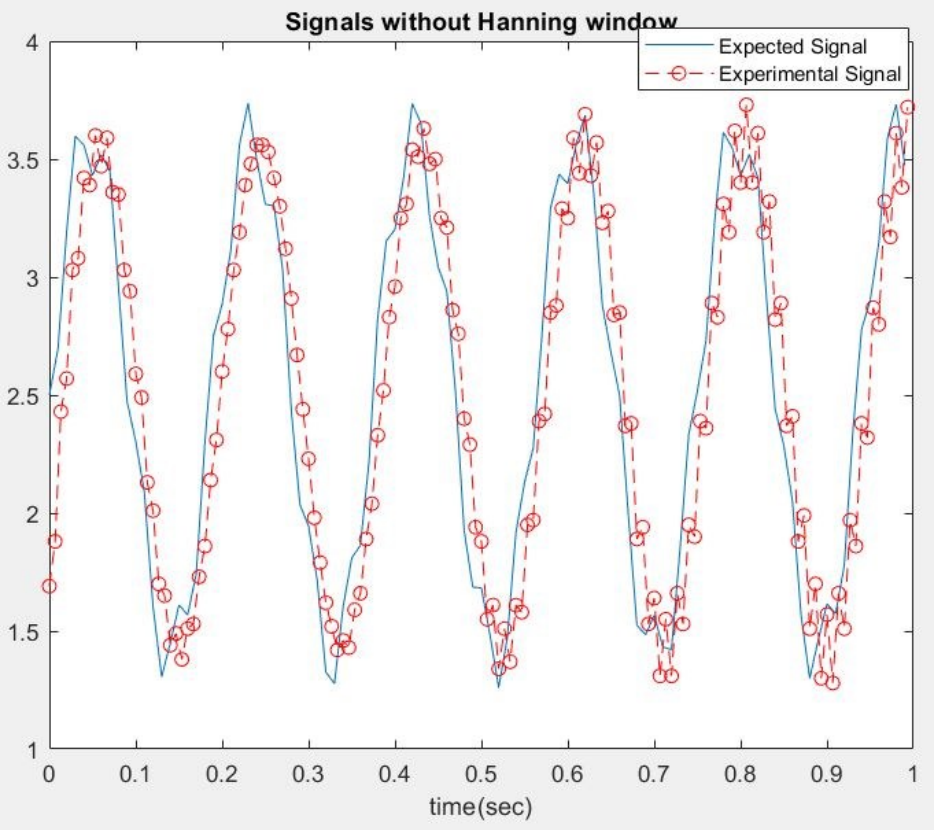
[2] Interrupt

فیلتر پایین‌گذر با فرکانس گوشه‌ی ۱۰ هرتز

برای طراحی این فیلتر، از مقاومت ۵۰ کیلواهم استفاده شده‌است و مقدار آن توسط مدار تقسیم ولتاژ، بر روی عدد ۱۵۹۱۵ اهم قرار دارد. همچنین از خازن ۱ میکروفاراد استفاده می‌شود. پس داریم:

$$f_0 = \frac{1}{2 * \pi * R * C} \approx 10\ Hz$$

دو سیگنال در شکل ۱۱ رسم شده است. سیگنال آبی رنگ، سیگنال تئوری تولیدشده توسط متلب است. سیگنال قرمز رنگ نیز خروجی آردوینو است.



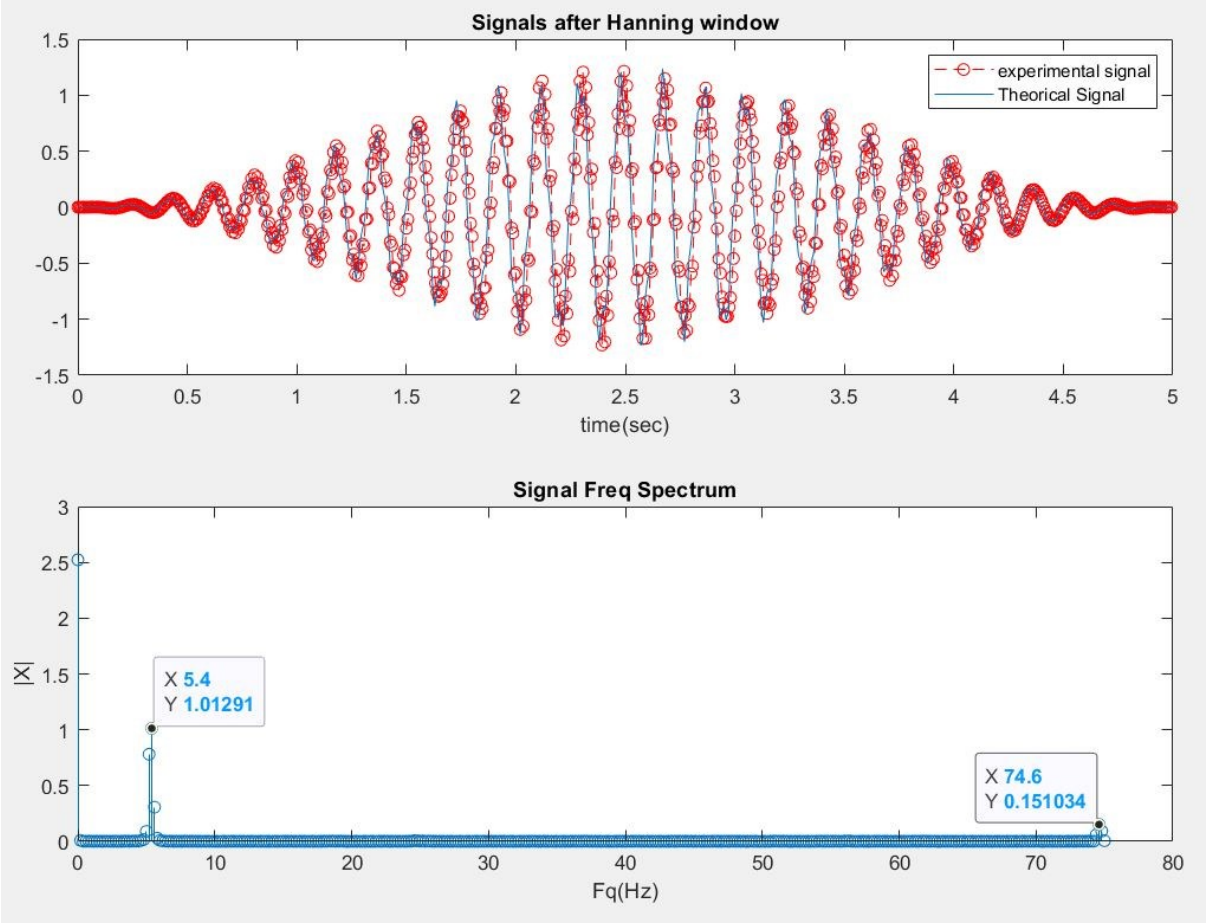
شکل ۱۱: نمودار ولتاژ – زمان مجموع دو سیگنال با فرکانس‌های ۵.۳۴ و ۷۴.۶ هرتز و فرکانس گوشه‌ی ۱۰ هرتز

مقدار تئوری تضعیف دامنه‌ای که این فیلتر ایجاد می‌کند، به ترتیب برای سیگنال ۵.۳۴ و ۷۶.۶۴ هرتز برابر با ۱.۰۸ و ۱۷.۵۳ دسی‌بل است.

برای آنکه در محتوای فرکانسی نشت به کمترین میزان برسد، باید از پنجره‌ی هنینگ استفاده کنیم.

$$u_{\text{Hanning}}(t) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi t}{t_f}\right)\right)$$

که در این رابطه، t_f کل مدت زمان سیگنال است. با ضرب این پنجره در سیگنال‌های بخش قبل، بخش بالایی شکل ۱۲ بدست می‌آید. در این شکل سیگنال آبی تولید شده توسط روابط تئوری در متلب پس از اعمال پنجره‌ی هنینگ است و نیز سیگنال قرمز خروجی آردوینو پس از اعمال این پنجره است.



شکل ۱۲: (بالا) نمودار ولتاژ – زمان مجموع دو سیگنال پس از اعمال پنجره هنینگ – (پایین) نمودار محتوای فرکانسی

در بخش پایینی شکل ۱۲ نیز نمودار محتوای فرکانسی سیگنال مشاهده می‌شود. همان‌طور که پیش‌بینی می‌شد، دو فرکانس ۵.۴ و ۷۴.۶ هرتز تشخیص داده شده‌است و به دلیل عبور سیگنال از فیلتر پایین‌گذر، مولفه‌ی دارای فرکانس بالا به میزان زیادی تضعیف شده‌است اما سیگنال با فرکانس پایین، مقدار تضعیف کمتری دارد.

مقدار تضعیف سیگنال ۵.۴ هرتز برابر است با:

$$20 * log10\left(\frac{1.01}{1.25}\right) = 1.85 \ dB$$

مقدار تضعیف سیگنال ۷۴.۶ هرتز برابر است با:

$$20 * log10\left(\frac{0.151}{1.25}\right) = 18.35 \ dB$$

این مقادیر اندکی با مقادیر تضعیف تئوری تفاوت دارند.

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- نویز همیشه وجود دارد و باعث ایجاد خطا است.

۴- مقداری نشت حتی با وجود استفاده از پنجره‌ی هنینگ همچنان وجود دارد.

۳- 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
  //R = 50000 - 10000 * Value;  // In the case of 50e3 Ohm.
  Serial.println(R);
  delay(100);
}
```

First Assignment:

Arduino

```
#include <TimerOne.h>
#include <MsTimer2.h>


/***** Constantants ****/
#define PWMOUT 5
#define ADC_CHANNEL A0


// Sine wave generation
const int SINE_FREQ = 23;          // Sine wave's frequency to be generated.
Hz.
const int SINE_GEN_FREQ = 500;     // Frequency of the sine wave generation
system. Hz.
const float SINE_AMP = 2.5;        // Sine wave's amplitude to be generated.
Volts.
const float SINE_OFFSET = 2.5;     // Sine wave's offset to be generated.
Volts.


// Sampling
const int SAMPLING_FREQ = 120;      // Sampling frequency. Hz.
const float TOTAL_SAMPLE_TIME = 2;  // Total sampling time. Seconds.


/****** Auxiliary variables ******/
const float SINE_COEFF = SINE_FREQ*TWO_PI/SINE_GEN_FREQ;
const float SINE_AMP_A = SINE_AMP/5.00*255;
const float SINE_OFFSET_A = SINE_OFFSET/5.00*255;
const int NUMBER_OF_SAMPLES = TOTAL_SAMPLE_TIME*SAMPLING_FREQ;
int DUTY_CYCLE, SAMPLES [NUMBER_OF_SAMPLES];
long SINE_GEN_COUNTER, SAMPLING_COUNTER;
bool FINISHED = false;              // Flag to know if process is finished
```

```
/****** Initialization ******/
void setup() {
  for (int i = 0; i < NUMBER_OF_SAMPLES ; i++) {
    SAMPLES[i]=-1;
  }

TCCR0B = TCCR0B & B11111000 | B00000001;  // Sets Timer0 PWM frequency
to 62.5KHZ
pinMode (PWMOUT, OUTPUT);
Serial.begin (57600);
MsTimer2::set(1000/SINE_GEN_FREQ, PWM_DT);   // Sets Timer 2
MsTimer2::start();
Timer1.initialize(1000000/SAMPLING_FREQ);    // Initializes Timer 1
Timer1.attachInterrupt(SAMPLING);
}

// Prints data if the process is finished
// Stores data otherwise
void loop() {
  if(FINISHED) {
    for (int i = 0; i < NUMBER_OF_SAMPLES ; i++) {
      Serial.println (SAMPLES[i]/1023.00*5);
    }
    FINISHED=false;
  }
}
```

```
// Generates signal
void PWM_DT() {
DUTY_CYCLE = SINE_AMP_A * sin(SINE_COEFF*SINE_GEN_COUNTER) +
SINE_OFFSET_A;
analogWrite (PWMOUT, DUTY_CYCLE);
SINE_GEN_COUNTER++;
}

// Read samples and stops if it reaches to the number of samples
void SAMPLING(){
  if (SAMPLING_COUNTER == NUMBER_OF_SAMPLES){
    MsTimer2::stop();
    Timer1.detachInterrupt();
    FINISHED = true;
  } else {
    SAMPLES[SAMPLING_COUNTER] = analogRead(ADC_CHANNEL);
    SAMPLING_COUNTER++;
  }
}
```

Matlab

```
clear; close all; clc;

%% Circuit properties
freq = 23;  %Hz
R = 5305;   %ohm
C = 1e-6;   %F
cte = 1/sqrt(1+(R*C*freq*2*pi)^2); %attenuation

%% Sampling properties
Fs = 120;          % Sampling frequency
T = 1/Fs;          % Sampling period
SampleNumber = Fs * 2;
t = (0:SampleNumber-1)*T;     %Arduino's signal time vector
f = 0:Fs/SampleNumber:Fs/2;   %FFT x-axis
t1 = 0:0.001:2;               %Theory signal time vector

%% Receiving data

a = zeros(SampleNumber,1);
s = serialport ("COM5", 57600);
configureTerminator(s,"CR/LF");

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

b = double(a);
clear s

experimental_signal = b;
%% FFT
xf = fft(experimental_signal);
Mf = abs(xf);
Mfs = Mf/(SampleNumber/2); %Scaling
Mfs(1) = Mfs(1)/2;
%% Plotting
figure
subplot (2,1,1)                       %Time-domain
plot(t(1:10),experimental_signal(1:10),'b--o','Color','r')
hold on
plot(t1(1:75),cte * 2.5*sin(2*pi*freq*t1(1:75))+2.5,'Color','#0072BD')
hold off
xlabel('time(sec)')
title('Fs = 120 Hz')
legend('experimental signal','Theorical Signal')
```

```
subplot (2,1,2)                          %FFT
stem (f,Mfs(1:SampleNumber/2+1))
xlabel('Fq(Hz)')
ylabel('|X|')
title('Signal Freq Spectrum')
```

Second Assignment

Arduino

```
#include <TimerOne.h>

#include <MsTimer2.h>


/***** Constantants ****/

#define PWMOUT 5

#define ADC_CHANNEL A0


// Sine wave generation

const float SINE_FREQ_1 = 5.34;     // First sine wave's frequency to be
generated. Hz.

const float SINE_FREQ_2 = 74.645;   // Second sine wave's frequency to be
generated. Hz.


const int SINE_GEN_FREQ = 500;    // Frequency of the sine wave generation
system. Hz.

const float SINE_AMP = 1.25;      // Sine wave's amplitude to be generated.
Volts.

const float SINE_OFFSET = 2.5;    // Sine wave's offset to be generated.
Volts.

// Sampling

const int SAMPLING_FREQ = 150;      // Sampling frequency. Hz.

const float TOTAL_SAMPLE_TIME = 5;  // Total sampling time. Seconds.


/****** Auxiliary variables ******/

// First sine wave's coefficient

const float SINE_COEFF_1 = SINE_FREQ_1*TWO_PI/SINE_GEN_FREQ;

// Second sine wave's coefficient

const float SINE_COEFF_2 = SINE_FREQ_2*TWO_PI/SINE_GEN_FREQ;

const float SINE_AMP_A = SINE_AMP/5.00*255;

const float SINE_OFFSET_A = SINE_OFFSET/5.00*255;

const int NUMBER_OF_SAMPLES = TOTAL_SAMPLE_TIME*SAMPLING_FREQ;

int DUTY_CYCLE, SAMPLES [NUMBER_OF_SAMPLES];

long SINE_GEN_COUNTER, SAMPLING_COUNTER;

bool FINISHED = false;
```

```
/****** Initialization ******/
void setup() {
  for (int i = 0; i < NUMBER_OF_SAMPLES ; i++) {
    SAMPLES[i]=-1;
  }

TCCR0B = TCCR0B & B11111000 | B00000001;  // Sets Timer0 PWM frequency to
62.5KHZ
pinMode (PWMOUT, OUTPUT);
Serial.begin (57600);
MsTimer2::set(1000/SINE_GEN_FREQ, PWM_DT);  // Sets Timer 2
MsTimer2::start();
Timer1.initialize(1000000/SAMPLING_FREQ);   // Initializes Timer 1
Timer1.attachInterrupt(SAMPLING);
}

// Prints data if the process is finished
// Stores data otherwise
void loop() {
  if(FINISHED) {
    for (int i = 0; i < NUMBER_OF_SAMPLES ; i++) {
      Serial.println (SAMPLES[i]/1023.00*5);
    }
    FINISHED=false;
  }
}

// Generates signal
void PWM_DT() {
DUTY_CYCLE = SINE_AMP_A * (sin(SINE_COEFF_1 * SINE_GEN_COUNTER) +
sin(SINE_COEFF_2 * SINE_GEN_COUNTER)) + SINE_OFFSET_A;
analogWrite (PWMOUT, DUTY_CYCLE);
SINE_GEN_COUNTER++;
}
```

```
// Read samples and stops if it reaches to the number of samples
void SAMPLING(){
  if (SAMPLING_COUNTER == NUMBER_OF_SAMPLES){
    MsTimer2::stop();
    Timer1.detachInterrupt();
    FINISHED = true;
  } else {
    SAMPLES[SAMPLING_COUNTER] = analogRead(ADC_CHANNEL);
    SAMPLING_COUNTER++;
  }
}
```

Matlab

```
clear; close all; clc;

%% Circuit properties
freq1 = 5.34;       %Hz
freq2 = 74.645;     %Hz
R = 15915;          %ohm
C = 1e-6;           %F
cte1 = 1/sqrt(1+(R*C*freq1*2*pi)^2);    %attenuation
cte2 = 1/sqrt(1+(R*C*freq2*2*pi)^2);

%% Sampling properties
Fs = 150;                   % Sampling frequency
T = 1/Fs;                   % Sampling period
SampleNumber = Fs * 5;
t = (0:SampleNumber-1)*T;     %Arduino's signal time vector
f = 0:Fs/SampleNumber:Fs/2;   %FFT x-axis
t1 = 0:0.01:5;                %Theory signal time vector

%% Receiving data

a = zeros(SampleNumber,1);
s = serialport ("COM5", 57600);
configureTerminator(s,"CR/LF");

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

b = double(a);
clear s

experimental_signal = b;
expected_signal = 1.25 * (cte1 * sin(freq1*2*pi*t1) + cte2 *
sin(freq2*2*pi*t1)) + 2.5;
%% Plotting without Hanning window
figure
plot(t1(1:100),expected_signal(1:100),'Color','#0072BD')
hold on
plot(t(1:150),experimental_signal(1:150),'b--o','Color','r')
hold off
xlabel('time(sec)')
title('Signals without Hanning window')
legend('Expected Signal','Experimental Signal')

%% FFT
% experimental_signal
x_m = mean(experimental_signal);
experimental_signal = experimental_signal - x_m;
w = hann(SampleNumber);
experimental_signal = w.*experimental_signal;
```

```
xf = fft(experimental_signal);
Mf = abs(xf);
Mfs = 2*Mf/(SampleNumber/2); %Scaling
Mfs(1) = Mfs(1) + x_m;

% Theory_signal
expected_signal = hann(length(t1)).*(expected_signal' - mean(ex-
pected_signal));

%% Plotting after Hanning window
figure
subplot (2,1,1) %Time-domain
plot(t,experimental_signal,'b--o','Color','r')
hold on
plot(t1,expected_signal,'Color','#0072BD')
hold off
xlabel('time(sec)')
title('Signals after Hanning window')
legend('experimental signal','Theorical Signal')

subplot (2,1,2) %FFT
stem (f,Mfs(1:SampleNumber/2+1))
xlabel('Fq(Hz)')
ylabel('|X|')
title('Signal Freq Spectrum')
```