CL-20

Final Report

REVISION HISTORY AND APPROVAL RECORD

Revision	Date	Purpose
0	28/05/2020	Document creation
1	29/05/2020	Document revision
2	30/05/2020	Document revision
3	31/05/2020	Document revision
4	01/06/2020	Document improved and delivery

DOCUMENT DISTRIBUTION LIST

Name	E-mail
Laura Pérez Martínez	laura.cube98@gmail.com
Chaimae Fathallah Rhafil	chaima4fathallah@gmail.com
Elisa Sayrol Clols	elisa.sayrol@upc.edu
Albert Oliveras Verges	albert@tsc.upc.edu

WRITTEN BY:		REVIEWED AND APPROVED BY:	
Chaime Fathallah Laura Pérez		Chaime Fathallah Laura Pérez	
Date	28/05/2020	Date	01/06/2020
Name	Chaime Fathallah	Name	Laura Pérez
Position	Docum. Resp.	Position	Project leader

0. CONTENTS

DOCUMENT SCOPE 4

TIME PLAN UPDATED 4

SYSTEM DESIGN DOCUMENTATION 9

SYSTEM IMPLEMENTATION DOCUMENTATION 23

SYSTEM CHARACTERIZATION 26

BUDGET 38

CONCLUSION 39

REFLECTION DOCUMENT 39

ANNEX 40

1. DOCUMENT SCOPE

The goal of this document is to compare the results obtained in this project between the objectives planned in the project plan.

In this document, you can find the results obtained showing graphs and numerical results of every parameter implemented in the application and a brief explanation of the code implementation.

First, we are going to make a reflection about the time planned to do the work packages of this project and the real time we expended to finish every task. We are going to make a programmer's manual where we explain the script made to implement every parameter and some specific results in order to understand better the code made to help anyone who wants to extend or improve our project. Then, we are going to make a user's manual to show you how you can use the application without having any problem, we show some general results of every parameter.

Finally, you can find a reflection about things that our team could have done better and the difficulties we had to carry on this project.

2. TIME PLAN UPDATED

In the following table we can see what period of time has been dedicated to each of the tasks described at the beginning of the project.

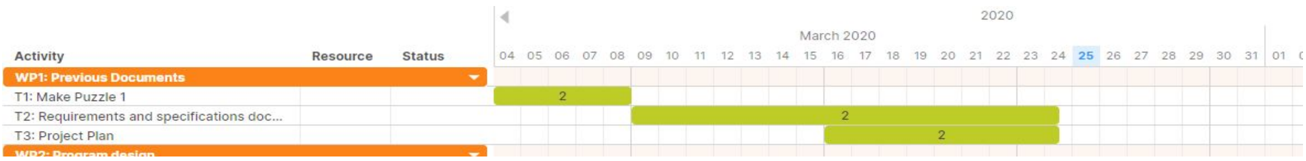
WP#	Task#	Short title	Duration	Start date	End date
1	1	Puzzle 1	5 days	04/03/2020	08/03/2020
	2	Requirements and specifications	17 days	08/03/2020	24/03/2020
	3	Project Plan	9 days	16/03/2020	24/03/2020
2	1	Define parameters	2 days	24/03/2020	25/04/2020
	2	Define program distribution	4 days	25/03/2020	28/03/2020
3	1	Signals definition	2 days	28/03/2020	29/03/2020
	2	Signals generation	2 days	28/03/2020	29/03/2020
	3	Parameters extraction	2 days	30/03/2020	31/03/2020
4	1	THD code	7 days	13/04/2020	19/04/2020
	2	THD + N code	7 days	20/04/2020	26/04/2020
	3	IMD code	7 days	20/04/2020	26/04/2020
	4	Frequency response code	7 days	27/04/2020	03/05/2020
	5	Gain and power code	7 days	27/04/2020	03/05/2020
4*	1	THD code	14 days	04/05/2020	17/05/2020
	2	THD + N code	14 days	04/05/2020	17/05/2020
	3	IMD code	14 days	04/05/2020	17/05/2020
	4	DF code	14 days	04/05/2020	17/05/2020
	5	Frequency response code	8 days	11/05/2020	18/05/2020
	6	Gain and power code	8 days	11/05/2020	18/05/2020

5	1	Interface implementation	3 days	16/05/2020	18/05/2020
6	1	Test all the parameters	-	-	-
7	1	User's manual	1 day	29/05/2020	29/05/2020
	2	Programmer's manual	1 day	30/05/2020	30/05/2020
	3	Final Report	5 days	28/05/2020	02/06/2020

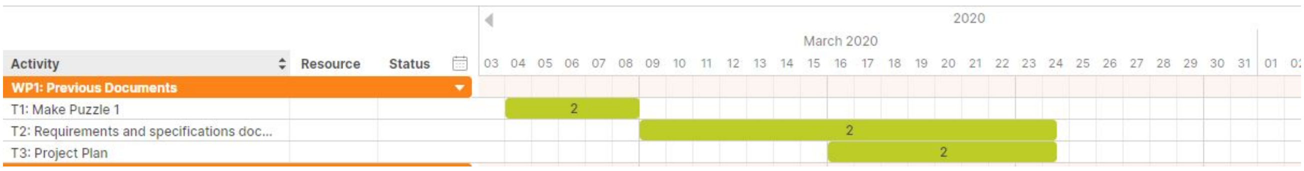
Next we can see a Gantt Diagram where we can observe the initially planned time with the one actually dedicated.

WP1

Planned

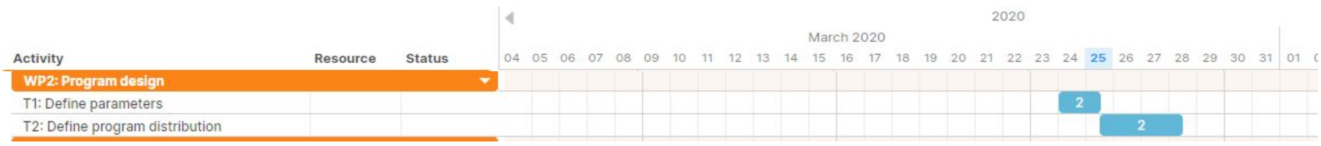


Dedicated

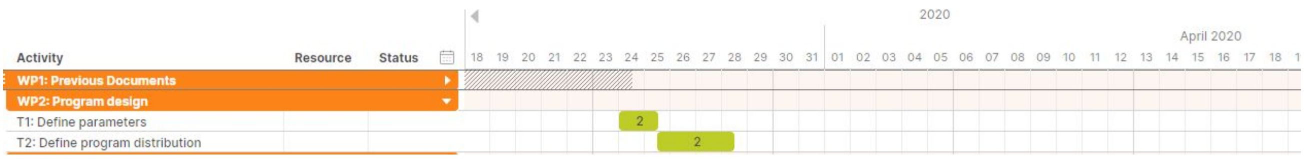


WP2

Planned

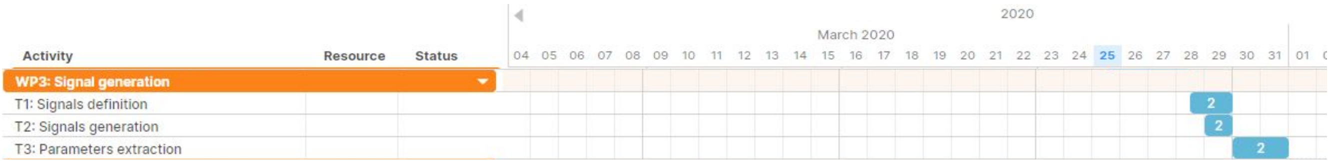


Dedicated

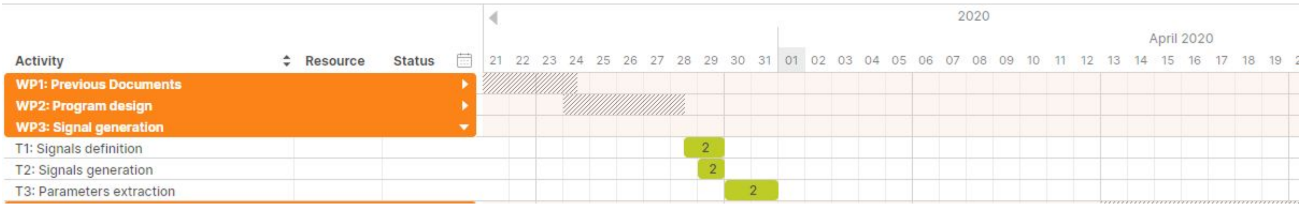


WP3

Planned

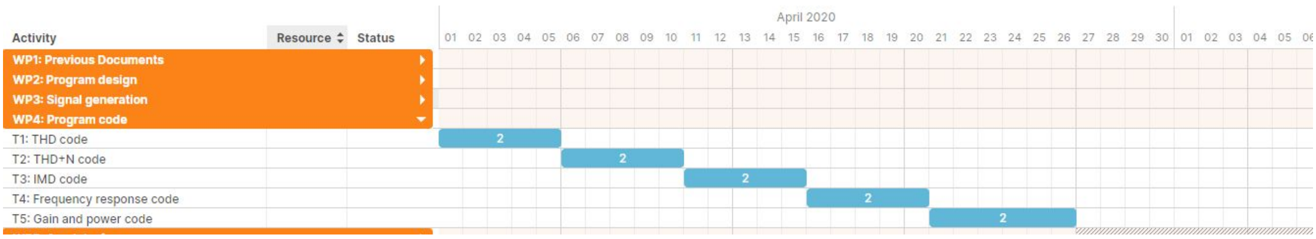


Dedicated

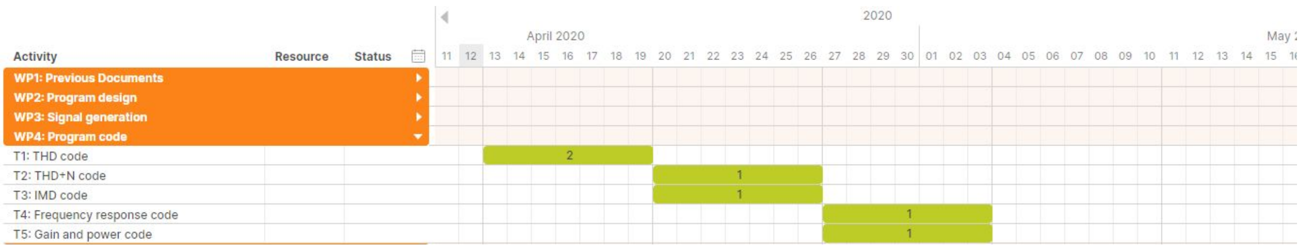


WP4

Planned

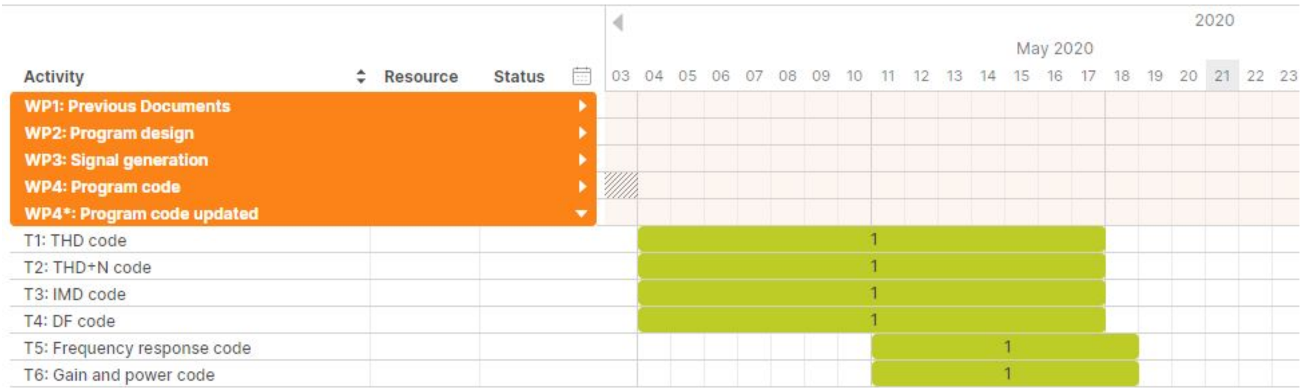


Dedicated



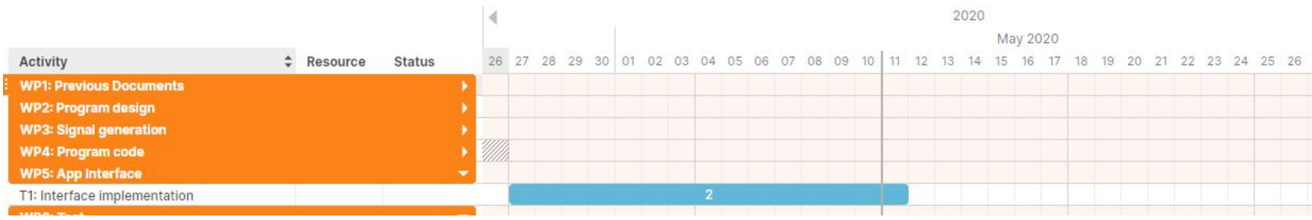
WP4*:

We don't have a planned gantt for this work package because it's a modification (we add a new parameter and modify some implementations of code) of WP4.

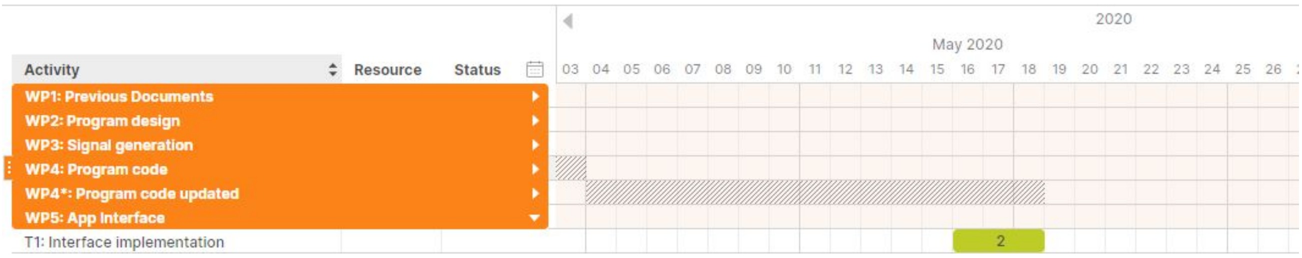


WP5

Planned

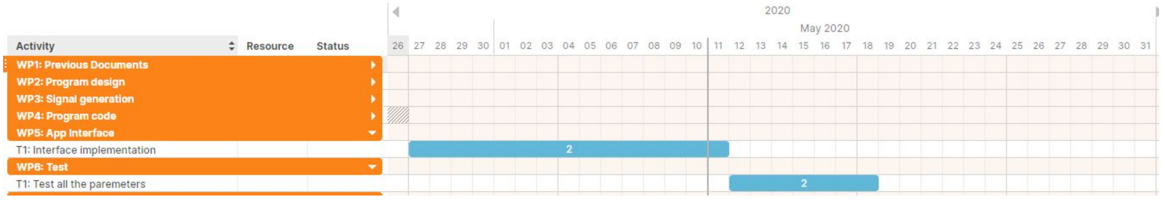


Dedicated



WP6

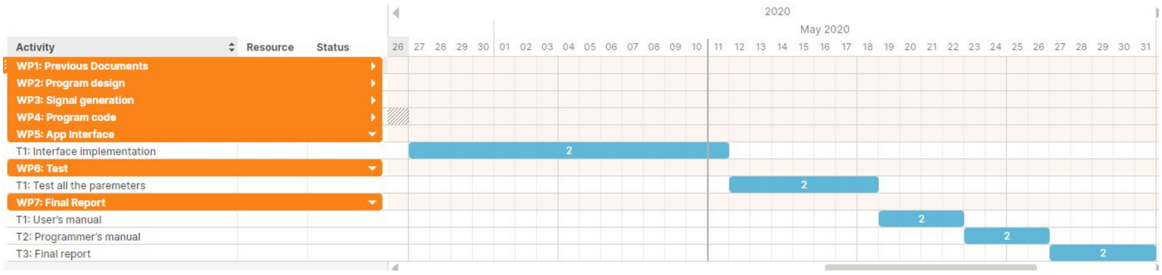
Planned



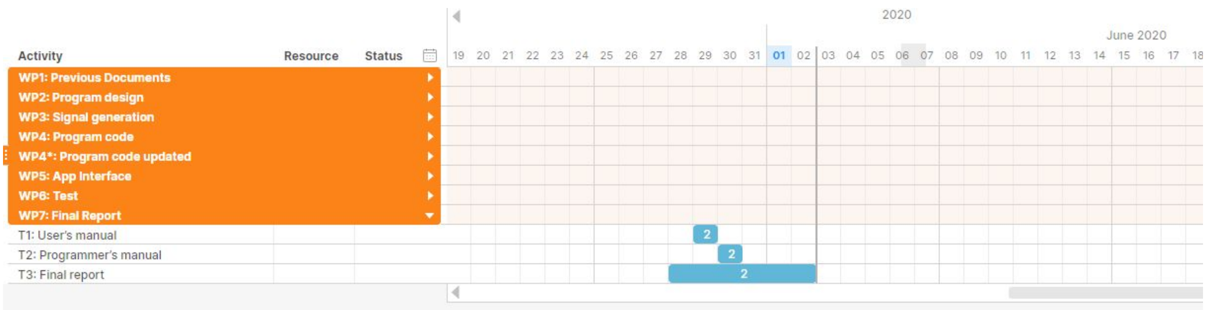
The testing of the parameters we have done throughout the project and we have not spent time at the end to do it.

WP7

Planned



Dedicated



In the first work packages we have followed the established timeline, but from WP 4 the start and end dates have varied, due to different reasons that we will explain below.

The main change has been the calculation of each of the parameters. We dedicate the initially planned time to calculate the parameters on real signals and add an extra period later to adapt this software to the calculation of real signals, reproduced and recorded by the computer. This extra time that we have used has not caused a delay in the project, since the implementation of the interface and the test period (WP5 and WP6) have finally been much faster.

When we finished WP 2, we saw that it was necessary to spend 2 weeks looking for information on the parameters calculated by our software and the implementation of them. Therefore, in the first two weeks of April we did this additional task, which we did not have planned.

Another change has been the WP6. Initially, we planned a testing period of all the parameters to corroborate their correct operation. We have realized when doing the project that we have done those tests weekly with the work done so far, so we have not had to spend extra time at the end.

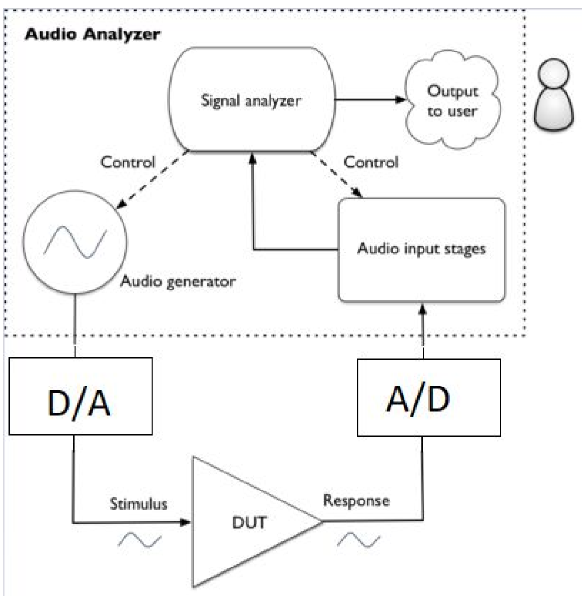
We have also done extra tasks that we had not originally planned. When looking for information about the distortion parameters that we calculated, we found a new parameter, the distortion factor, that could add more features to our software. We decided to add it to our project. Another of the extra tasks have been the weekly reports. At the end of each week we have made a report where we have included the work of the last 7 days, the doubts we had, the hours spent and occasionally bibliography consulted, especially the first weeks

In summary, we have managed to do everything we set out to do at the start of the project within the established time, and even some additional tasks. Although we have made small variations in the time dedicated to work packages, mainly due to exams or work that they have put us over the weeks, we have finish the project on time.

3. SYSTEM DESIGN DOCUMENTATION

Virtual Instrumentation

In the virtual instrumentation app, we measure some features to estimate the quality of audio amplifiers. Our program can measure THD (Total Harmonic Distortion), THD+N (Total Harmonic Distortion and Noise), IMD (Intermodulation distortion), DF (distortion factor), gain, power estimation and frequency response.



Next we will explain each one of them:

THD and THD+N

In this features we measure the harmonics produced by the imperfections of the amplifiers.

The principal harmonic is at the frequency of the sinusoid at the input and in the output we can found secondary harmonics in multiples of that frequency.

So we can use the following formulas:

$$\text{THD + N} = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + \ldots + V_n^2 + V_{noise}^2}}{V_s}$$

$$\text{THD} = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + \ldots + V_n^2}}{V_s}$$

V_s = Signal Amplitude (RMS Volts)
V_2 = Second Harmonic Amplitude (RMS Volts)
V_n = nth Harmonic Amplitude (RMS Volts)
V_{noise} = RMS value of noise over measurement bandwidth

If we consider noise we calculate THD+N, and without considering noise and considering only the harmonics we calculate THD.

DF

This parameter shows the power ratio of the main harmonic with respect to the total harmonics. When the secondary harmonics are minimal, the distortion factor tends to 100%. This allows us to know what percentage of the signal power corresponds to the main harmonic.

IMD

To measure this feature we consider the non-linearities produced by two signals at different frequencies . So, the purpose of this measure is to calculate the sub-harmonics generated by two signals at different frequencies. We can measure the distortion of any order but typically second and third order is measured.

Gain

This measure is the magnitude that express the relation between the amplitude at the input of the amplifier and the amplitude at the output of the amplifier. We can express the results in dB.
We can measure this feature as :

$$G(dB)=20\log_{10}\left(\frac{Vout}{Vin}\right)$$

or if we use the power :

$$G(dB)=10\log_{10}\left(\frac{Pout}{Pin}\right)$$

Frequency response

We measure the curve of gain typically between 20Hz and 20kHz of an audio amplifier.

To measure this feature we are going to generate a sinusoid at 20Hz. Then, we are going to increase the frequency of the sinusoid with the same amplitude and calculate for each frequency the corresponding gain. Like this, we can represent the frequency response of the element under test.

Scripts explanation

Now we are going to explain the code used in the application to measure these features:

THD code:

```
Fs = app.SamplingRateHzEditField.Value;
duration = app.SignalDurationsEditField.Value;

F = app.FrequencyHzEditField.Value;
Fd = F*duration;
t = 0:(1/Fs):duration-(1/Fs);
n = 0:1:duration*Fs-(1/Fs);
x = sin(2*pi*n*(Fd/Fs))';
```

First of all we obtain the sampling rate, the frequency and the duration of the signal introduced by the user.

Then, we generate a sinusoid considering it as a input.

```
player   = audioplayer(x, Fs, 16);
recorder = audiorecorder(Fs, 16, 2);

record(recorder,duration)
playblocking(player);

stop(recorder)
signal = getaudiodata(recorder, 'double');
[N,M] = size(signal);
```

We play the sinusoid to record it and then we get the signal of the sinusoid recorded.

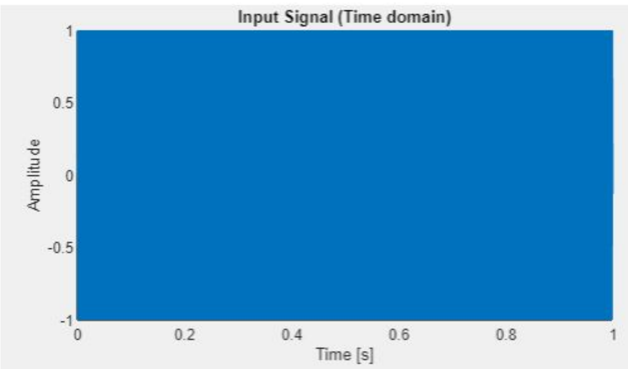
```
f = abs(fft(signal));

x_max = 1;

% Busquem a quina posició (x) es troba el màxim
for i = 1:1:N/2
    if (f(i) > f(x_max))
        x_max = i;
    end
end

harmonics = x_max;                  % Vector amb les posicions dels harmònics
delta_x = round(x_max*0.2);
num_harm = fix((N/2)/x_max);        % Càlcul del número d'harmònics generats

% Cerca de la posició de cada harmònic
for i = 2:1:num_harm
    hs = i*x_max;
    for j = (i*x_max-delta_x):1:(i*x_max+delta_x)
        if (f(j) > f(hs))
            hs = j;
        end
    end
    % Només agafem els harmònics amb una potència superior al 0.01%
    % respecte al principal
    if (f(hs)/f(x_max) > 0.01)
        harmonics(end+1) = hs;
    end
end
```

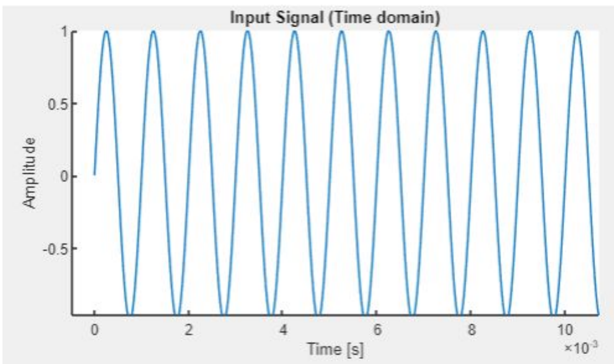
```
num = 0;
[m,n] = size(harmonics);

% Càlcul de la potència dels harmònics secundàris
for i = 2:1:n
    num = num + (f(harmonics(i)))^2;
end

thd = (sqrt(num)/f(harmonics(1)))*100;
app.THDEditField.Value = sprintf('%.2f', thd);

plot(app.In_time, t, x);
app.In_time.Title.String = 'Input Signal (Time domain)';

plot(app.Out_time, t, signal);
app.Out_time.Title.String = 'Output Signal (Time domain)';

app.fft_fr.XLim = [0 (harmonics(end)+harmonics(1))];
plot(app.fft_fr, abs(fft(signal)));
app.fft_fr.Title.String = 'FFT (frequency domain)';
```

We search the position of the harmonics and then we calculate the thd expressing the result in percentage.

In general, we plot for each measure the input signal and the output signal in the time domain and the fft of the output signal showing the resulting harmonics.

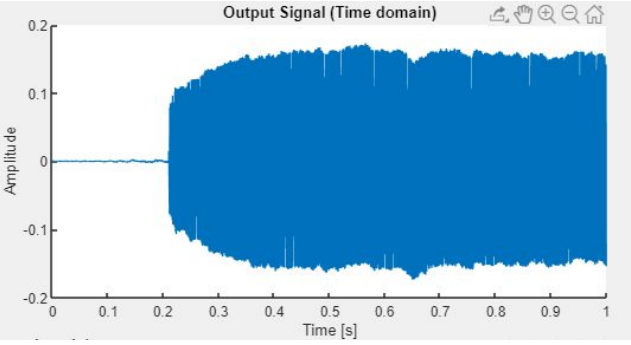
For example, for a input sinusoid of 1 kHz we obtain:

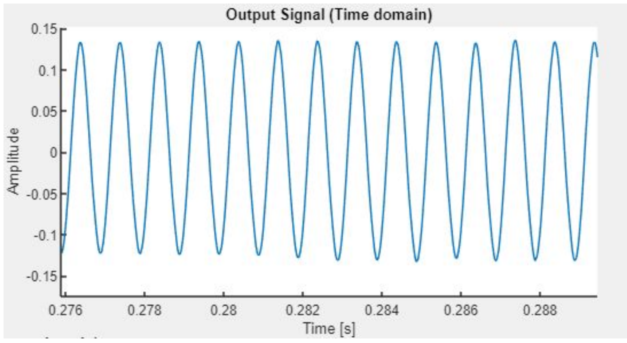




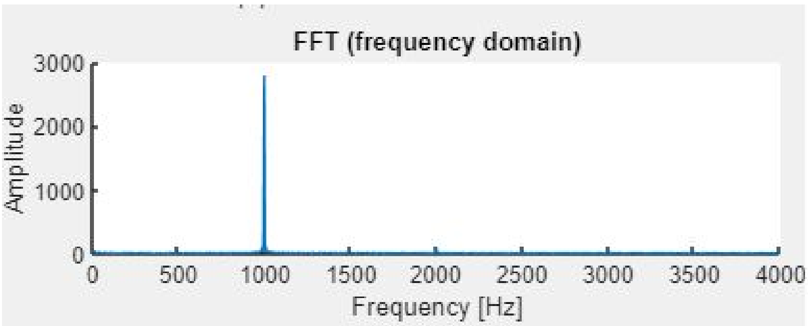
In the second image we can see an enlargement of the first to verify that it really is a sinusoid.

At the output we obtained:





The fft of the output signal is:



This graphs are represented for all parameters (except FR where we show only the curve of frequency response).

THD+N code

The difference between this measure and the previous one is that in this case we consider the noise produced by amplifier.

```
200         f = abs(fft(signal));
201
202         x_max = 1;
203
204         % Busquem a quina posició (x) es troba el màxim
205         for i = 1:1:N/2
206             if (f(i) > f(x_max))
207                 x_max = i;
208             end
209         end
210
211         % Càlcul de la potència de tot el senyal menys de l'harmonic
212         % principal
213
214         f_harm_prin = f(:,1);
215         f_seny_sense_harm = f(:,1);
216
217         rang = round(0.1*x_max);
```

```
219     for i = 1:1:(x_max - rang)
220         f_harm_prin(i) = 0;
221     end
222     for i = (x_max - rang):1:(x_max + rang)
223         f_seny_sense_harm(i) = 0;
224     end
225     for i = (x_max + rang):1:(Fs - x_max - rang)
226         f_harm_prin(i) = 0;
227     end
228     for i = (Fs - x_max - rang):1:(Fs - x_max + rang)
229         f_seny_sense_harm(i) = 0;
230     end
231     for i = (Fs - x_max + rang):1:Fs-1
232         f_harm_prin(i) = 0;
233     end
234
235     a = ifft(f_seny_sense_harm);
236     b = ifft(f_harm_prin);
237
238     P = sum((abs(a)).^2)/Fs;
239     Pa = sum((abs(b)).^2)/Fs;
240
241     harmonics = x_max;              % Vector amb les posicions dels harmònics
242     delta_x = round(x_max*0.2);
243     num_harm = fix((N/2)/x_max);    % Càlcul del número d'harmònics generats
```

We transform the signal received to the frequency domain to find the principal harmonic and the secondary harmonics. Then we apply the inverse transform of fft to obtain two signals (line 235 and 236) in the time domain; the first one corresponds only to the main harmonic and the second one to the entire signal except the main harmonic. The sum of both give the original signal.

```
260                 % Càlcul del THD + N
261 -               thdn = (P/Pa)*100;
262 -               app.THDNEditField.Value = sprintf('%.2f', thdn);
```

Then, in the line 262 we show the result in the app.

DF code

```
df = (f(harmonics(1))/sqrt(num))*100;
app.DFEditField.Value = sprintf('%.2f', df);
```

The procedure to calculate the distortion factor is similar to the calculation of the THD (creation and recording of the signal, search for the main harmonic and all the secondary harmonics). The difference is that the distortion factor is calculated as the ratio of the main harmonic to all harmonics of the signal.

IMD code

```
Fs = app.SamplingRateHzEditField.Value;
duration = app.SignalDurationsEditField.Value;

F1 = app.FrequencyHzEditField.Value;
F2 = app.AEditField.Value;
Fd1 = F1*duration;
Fd2 = F2*duration;
t = 0:(1/Fs):duration-(1/Fs);
n = 0:1:duration*Fs-(1/Fs);
x1 = sin(2*pi*n*(Fd1/Fs))';
x2 = sin(2*pi*n*(Fd2/Fs))';
f01 = abs(fft(x1));
f02 = abs(fft(x2));
x = x1 + x2;
```

To measure this feature we obtain two different frequencies that the user introduce and generate for each one a sinusoid and then we sum the two sinusoids.

Final Report
[CL - 20]

```
f = abs(fft(signal));

s = f;

x_max_1 = 1;
x_max_2 = 2;

% Busquem a quina posició (x) es troba el màxim
for i = 1:1:N/2
    if (s(i) > s(x_max_1))
        x_max_1 = i;
    end
end

rang = round(0.1*x_max_1);

% Valor del primer màxim a zero per no generar problemes en la cerca del segon
for i = (x_max_1 - rang):1:(x_max_1 + rang)
    s(i) = 0;
end

% Càlcul del segon màxim
for i = 1:1:N/2
    if (s(i) > s(x_max_2))
        x_max_2 = i;
    end
end
```

First, we search the two principals harmonics.

```
% Càlcul de la posició dels harmònics secundàris
harmonics = 1;

if (x_max_1 > x_max_2)
    max_absolut = x_max_1;
else
    max_absolut = x_max_2;
end

delta_x = round(((x_max_1+x_max_2)/2)*0.03);
harmonics(end+1) = x_max_1;
harmonics(end+1) = x_max_2;
ordre = app.BEditField.Value;
```

```
% Cerca de la posició dels harmònics secundaris
for i = -ordre:1:ordre
    for j = -ordre:1:ordre
        if (abs(i) + abs(j) == ordre)
            pos = i*x_max_1 + j*x_max_2;
            if (pos > 0 && pos < (N/2))
                for k = (pos - delta_x):1:(pos + delta_x)
                    if (f(k) > f(pos))
                        pos = k;
                    end
                end
                if (f(pos)/f(max_absolut) > 0.01)
                    harmonics(end+1) = pos;
                end
            end
        end
    end
end

%% IMD

num = 0;
den = 0;
[m,n] = size(harmonics);
```

Then, we search the secondary harmonics depending on the order the user choose.

```
% Càlcul de la potència dels harmònics secundàris
for i = 3:1:n
    num = num + (f(harmonics(i)))^2;
end

% Càlcul de la potència dels harmònics principals
for i = 1:1:2
    den = den + (f(harmonics(i)))^2;
end

imd = (sqrt(num)/sqrt(den))*100;
app.IMDEditField.Value = sprintf('%.2f', imd);
```

Finally, we calculate the imd and we show the result in the app.

Gain code

```
%% Calculem la fft de la senyal de sortida
f = abs(fft(signal(:,1)));

%% Busquem a quina posició (x) es troba el màxim
x_max = 1;

for i = 1:1:N/2
    if (f(i) > f(x_max))
        x_max = i;
    end
end

harmonics = x_max;
delta_x = round(x_max*0.2);
num_harm = round((N/2)/x_max);

% Cerca de la posició de cada harmònic
for i = 2:1:num_harm
    hs = i*x_max;
    for j = (i*x_max-delta_x):1:(i*x_max+delta_x)
        if (f(j) > f(hs))
            hs = j;
        end
    end
    % Només agafem els harmònics amb una potència superior al 0.01%
    % respecte al principal
    if (f(hs)/f(x_max) > 0.01)
        harmonics(end+1) = hs;
    end
end

%% Calculem el guany
gain = f(harmonics(1))/f0(F+1);
app.GainEditField.Value = sprintf('%.2f', gain);
```

First, we search the harmonics as we do to calculate thd and then we divide the first harmonic of the output signal between the first harmonic of the input signal.

Power code

```
s = 0;
Pot=0;

% Busquem l'amplitud mitja de les primeres mostres per establir un llindar a partir del
% cual considerarem que comença la grabació
for i = 1:1:1000
    s = s + abs(signal(i,1));
end

s = s/1000;

% Càlcul de la potència mitja del senyal
for j = 1000:N
    if(abs(signal(j,1))>2*s)
        Pot = Pot + abs(signal(j,1)).^2;
    end
end

%Calculem la potència
pwr = Pot/(N-1000);
app.PwrEditField.Value = sprintf('%f', pwr);
```

We calculate the power considering a threshold. To calculate the threshold we used the first 1000 samples to estimate the amplitude of noise and then we calculate the power considering two times the threshold.

Frequency Response code

```
Fs = app.SamplingRateHzEditField.Value;
duration = app.SignalDurationsEditField.Value;

n = 0:1:duration*Fs-(1/Fs);

fMin = log10(app.AEditField.Value);
fMax = log10(app.BEditField.Value);
nSinusoids = app.CEditFile.Value;

freq = logspace(fMin, fMax, nSinusoids);
gain = linspace(0, 0, nSinusoids);

%% Reproducció i grabació de la sinusoide a diferents freqüències

for j=1:nSinusoids
    x = 2*sin(2*pi*n*((freq(j)*duration)/Fs))';
    f0 = abs(fft(x));

    x_max_in = 1;

    for r = 1:1:Fs/2
        if (f0(r) > f0(x_max_in))
            x_max_in = r;
        end
    end
```

In this case to calculate the frequency response we define fmax and a fmin in order to calculate the gain over this range. The user has the option to introduce the number of sinusoids and the range of frequencies.

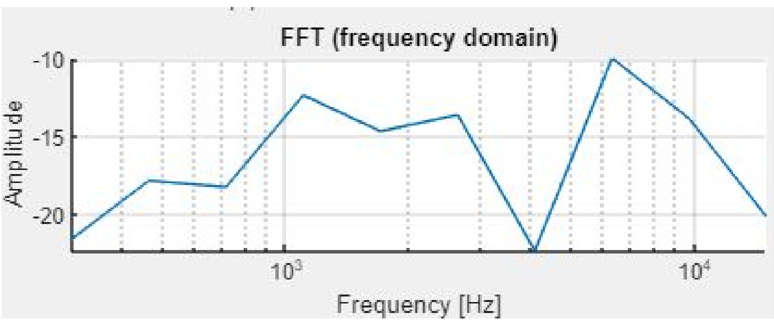
```
harmonics_in = x_max_in;
delta_x_in = round(x_max_in*0.2);
num_harm_in = fix((Fs/2)/x_max_in);

for r = 2:1:num_harm_in
    hs = r*x_max_in;
    for p = (r*x_max_in-delta_x_in):1:(r*x_max_in+delta_x_in)
        if (f0(p) > f0(hs))
            hs = p;
        end
    end
    if (f0(hs)/f0(x_max_in) > 0.01)
        harmonics_in(end+1) = hs;
    end
end

player = audioplayer(x, Fs, 16);
recorder = audiorecorder(Fs, 16, 2);

record(recorder,duration)
playblocking(player);

stop(recorder)
signal = getaudiodata(recorder, 'double');
[N,M]=size(signal);

f=abs(fft(signal(:,1)));
```

```
            x_max_out = 1;

            for i = 1:1:N/2
                if (f(i) > f(x_max_out))
                    x_max_out = i;
                end
            end

            harmonics_out = x_max_out;
            delta_x_out = round(x_max_out*0.2);
            num_harm_out = round((N/2)/x_max_out);

            for i = 2:1:num_harm_out
                hs = i*x_max_out;
                for k = (i*x_max_out-delta_x_out):1:(i*x_max_out+delta_x_out)
                    if (f(k) > f(hs))
                        hs = k;
                    end
                end
                if (f(hs)/f(x_max_out) > 0.01)
                    harmonics_out(end+1) = hs;
                end
            end

            % Càlcul del guany en dB

            gain(j) = 10*log10(f(harmonics_out(1))/f0(harmonics_in(1)));
        end
```

We play and record for each iteration and search the harmonics generated by each sinusoid. We calculate the gain using the same procedure uses to calculate the gain parameter.

```
            plot(app.In_time, 0)
            plot(app.Out_time, 0)

            app.fft_fr.XLim = [app.AEditField.Value app.BEditField.Value];
            app.fft_fr.YLim = [min(gain) max(gain)];
            app.fft_fr.XGrid = 'on';
            app.fft_fr.YGrid = 'on';
            semilogx(app.fft_fr, freq, gain);
            app.fft_fr.Title.String = 'FFT (frequency domain)';
```

Then we plot the frequency response and show the result in the app, and we obtain:



The result of this parameter is not optimal due to problems with the computer and the environment. These problems are discussed later.

4. SYSTEM IMPLEMENTATION DOCUMENTATION

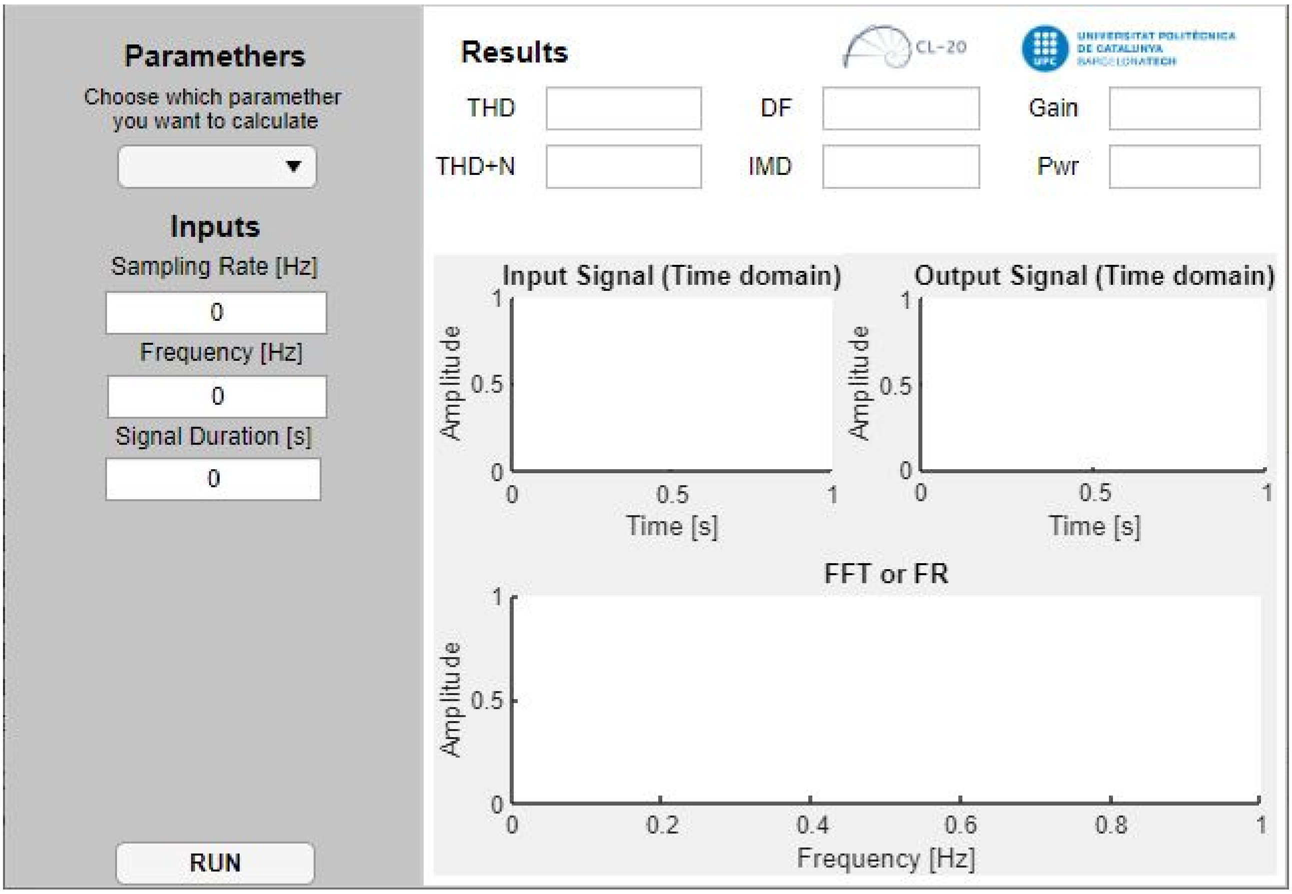
Installation:

The application can be downloaded from the project's GitHub repository. There you can found all the documents related to the project, all of them linked in the Readme.

When you have downloaded the file, it can be install and run from Matlab, in app section. You can run it without doing any additional material like external cables or connections.

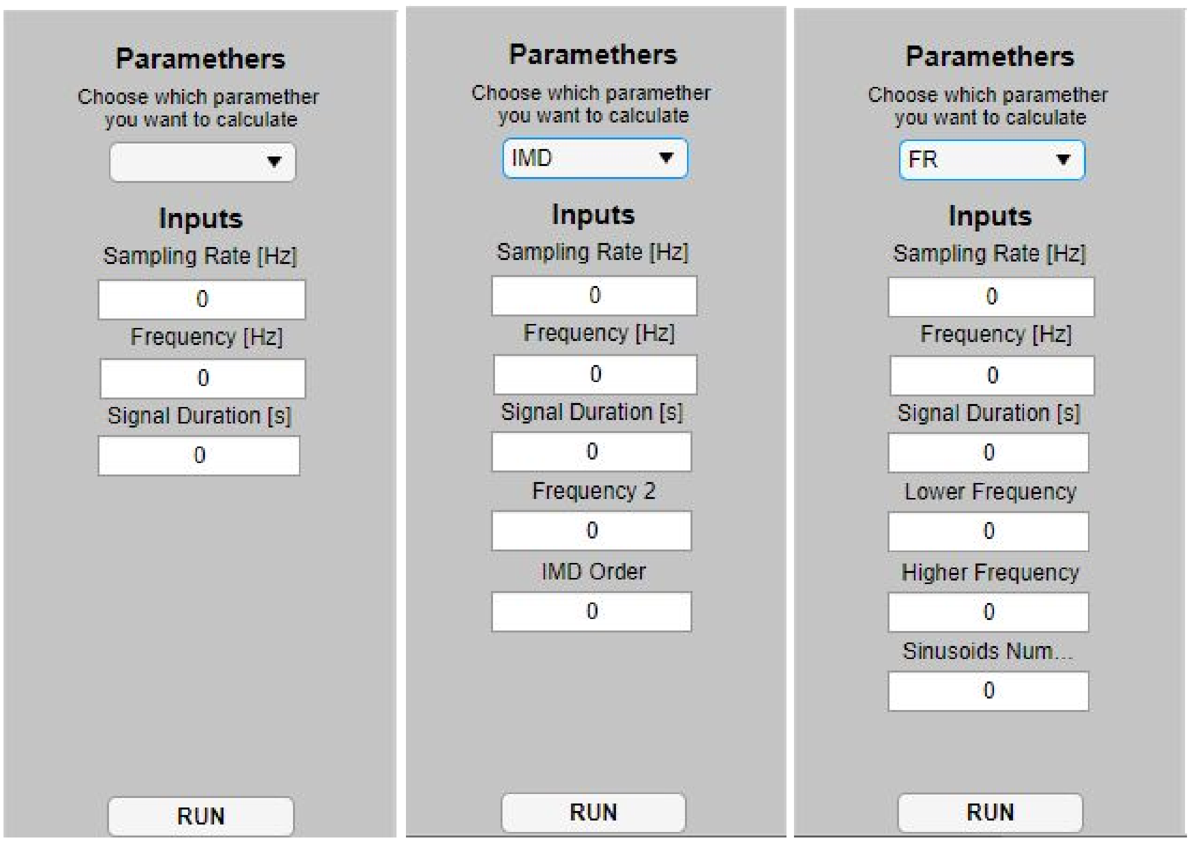
App structure:

The app is divided into two sections: the data panel and the results panel.



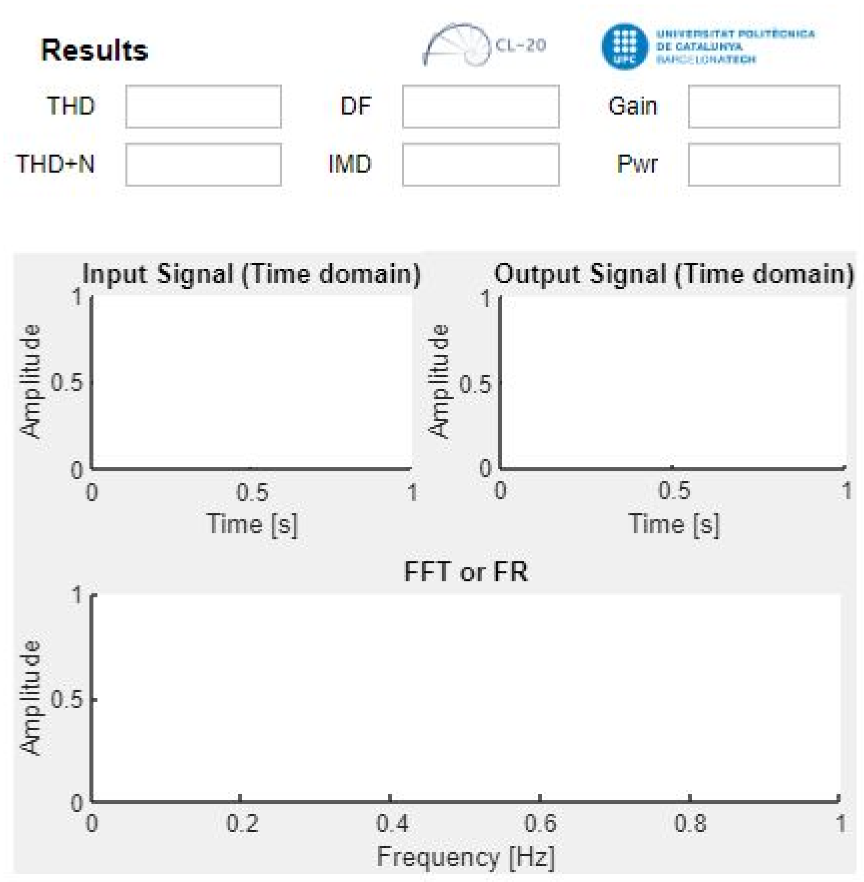
- **Data panel**

  In this section, the user indicates which parametre he wants to calculate and the values of the inputs. Once all the values have been entered, the calculation will be made by pressing the "Run" button.



- **Results panel**

  In this section, the user can see the value of each parameter that he has calculated, as well as three graphs. The upper two, show the signal generated in Matlab and the signal recorded with the computer. Both in temporal domain. The lower one, shows the absolute value of the fft in all cases except in the calculation of the frequency response.



Expected results:

For optimal operation of the amplifier we will look for the following values:

- Minimum values for THD and THD + N. Ideal value: 0%.
- Maximum values for DF and IMD. Ideal value: 100%.
- Minimum values for Gain. Ideal value: 0dB
- Frequency response. +/- 3 dB between 20 Hz and 20KHz

5. SYSTEM CHARACTERIZATION

When we made the application, we prioritized the intuitive aspect of the application. We wanted a simple, clean and intuitive application.

We divide the application into two sections. The first one, located on the left, is the part with which the user can interact and consists of three sections:

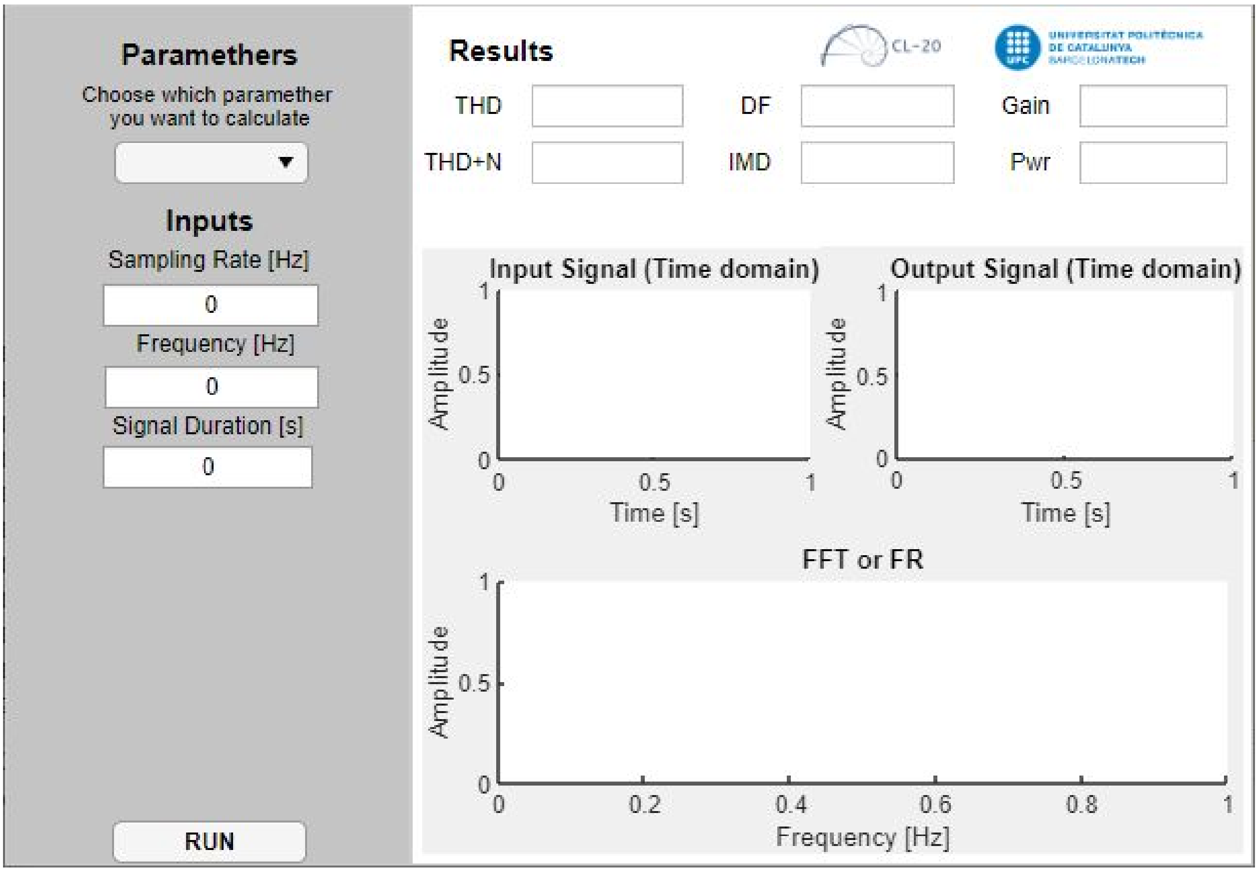
- Choice of the parameter to be calculated.
- Entering the values that define the signal.
- Run button.

The second part, in which the results are displayed, is made up of two sections:

- Boxes where the value of the parameter being calculated is displayed
- Graphics. The upper two, show the signal generated in Matlab and the signal recorded with the computer. Both in temporal domain. The lower one, shows the absolute value of the fft in all cases except in the calculation of the frequency response.

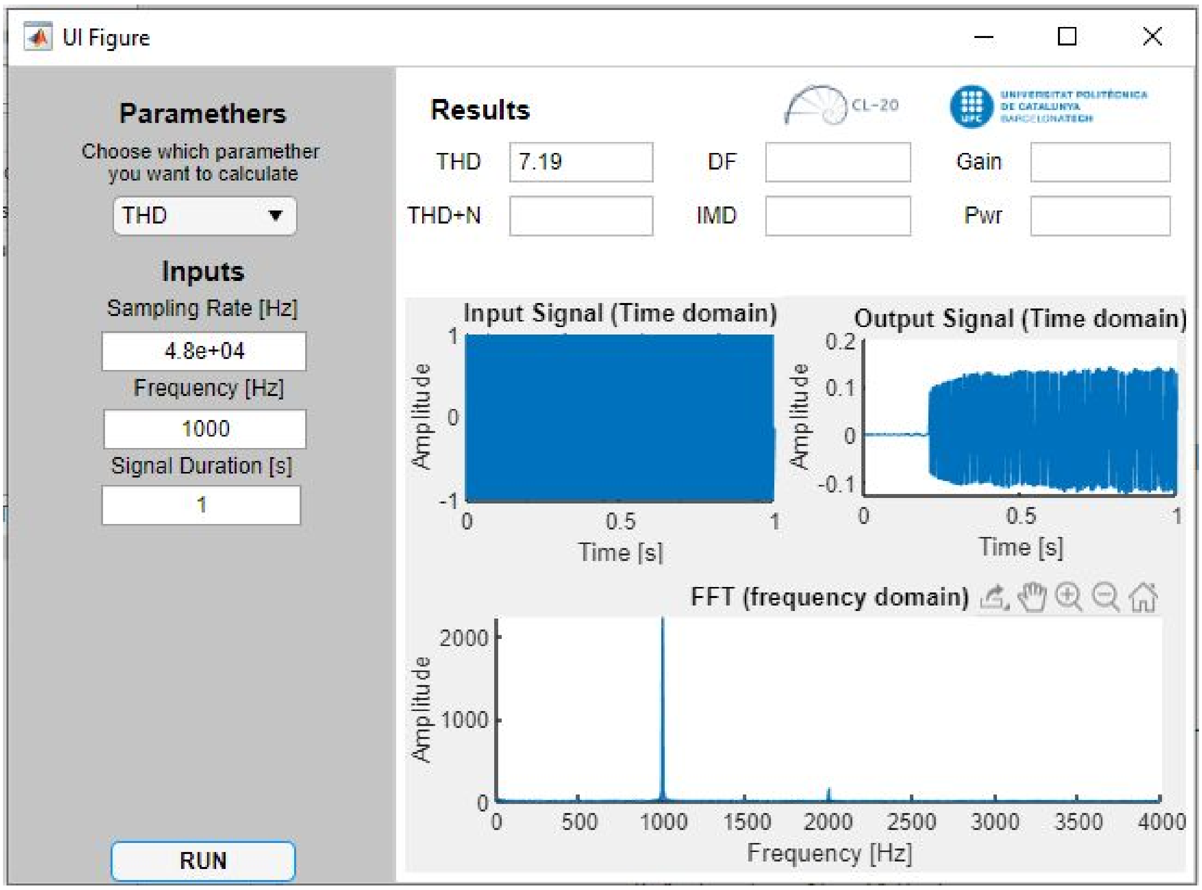
At the top right we include the logo of our product and that of the university.

This is what the application looks like:



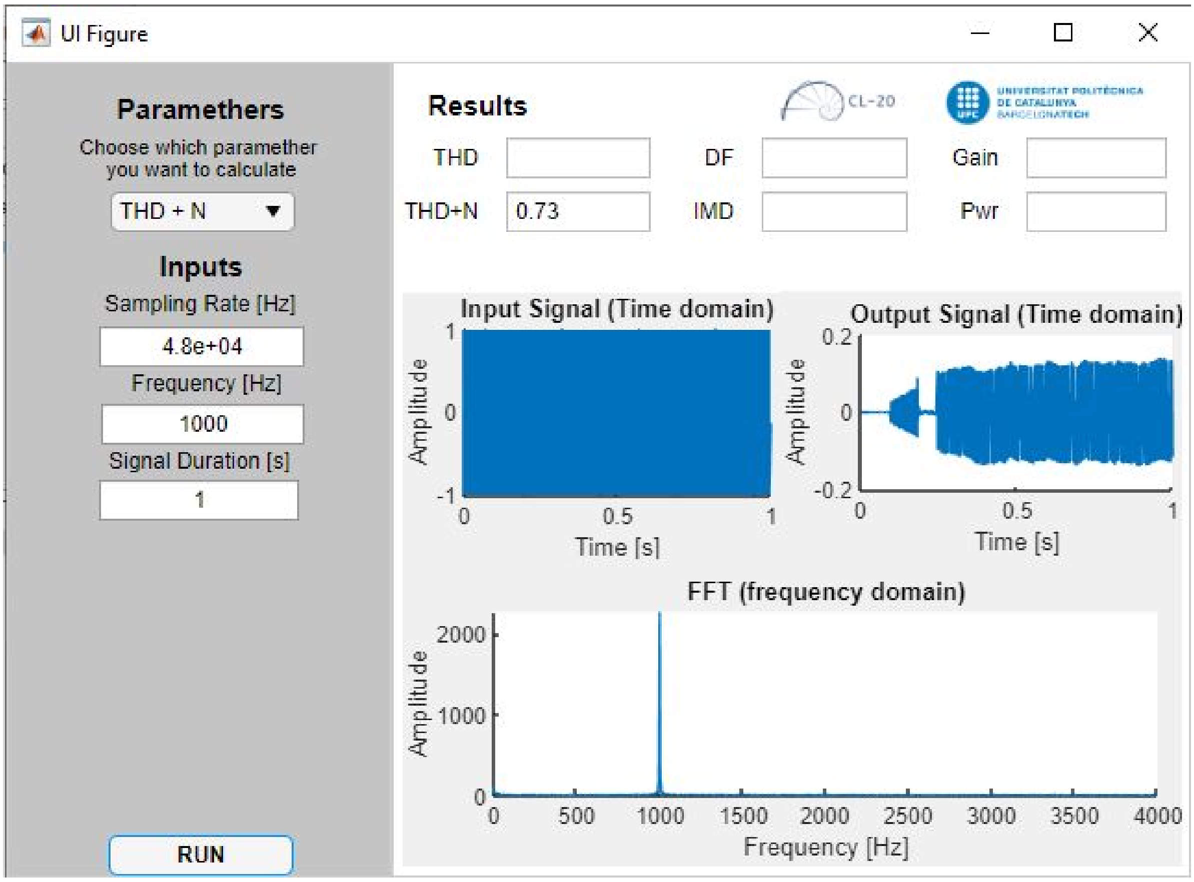
Next we will show the result we obtain for each of our parameters. In all the parameters (except in the frequency response) we will see the percentage value in its corresponding box and the signals described above. In the frequency response we will only see the graph of it.

THD



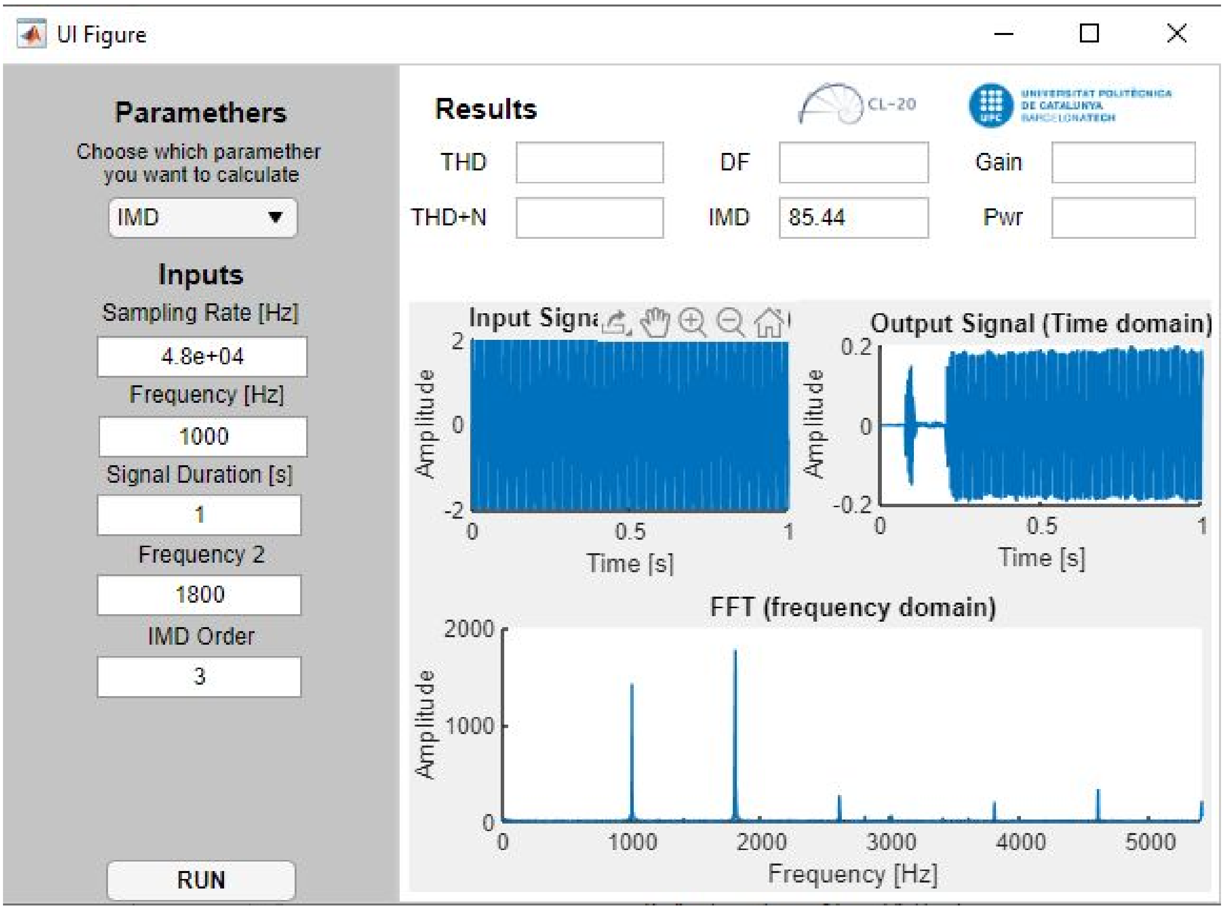
We get a THD value of 7.19 %. As we see in the fft, the first secondary harmonic is very small and the rest are negligible, which makes their relationship to the main harmonic very small.

THD + N



We obtain a THD + N value of 0.73%. In this case we can see that all the secondary harmonics are minimal and there is no noise in the signal, so the value of this parameter is almost zero.

IMD

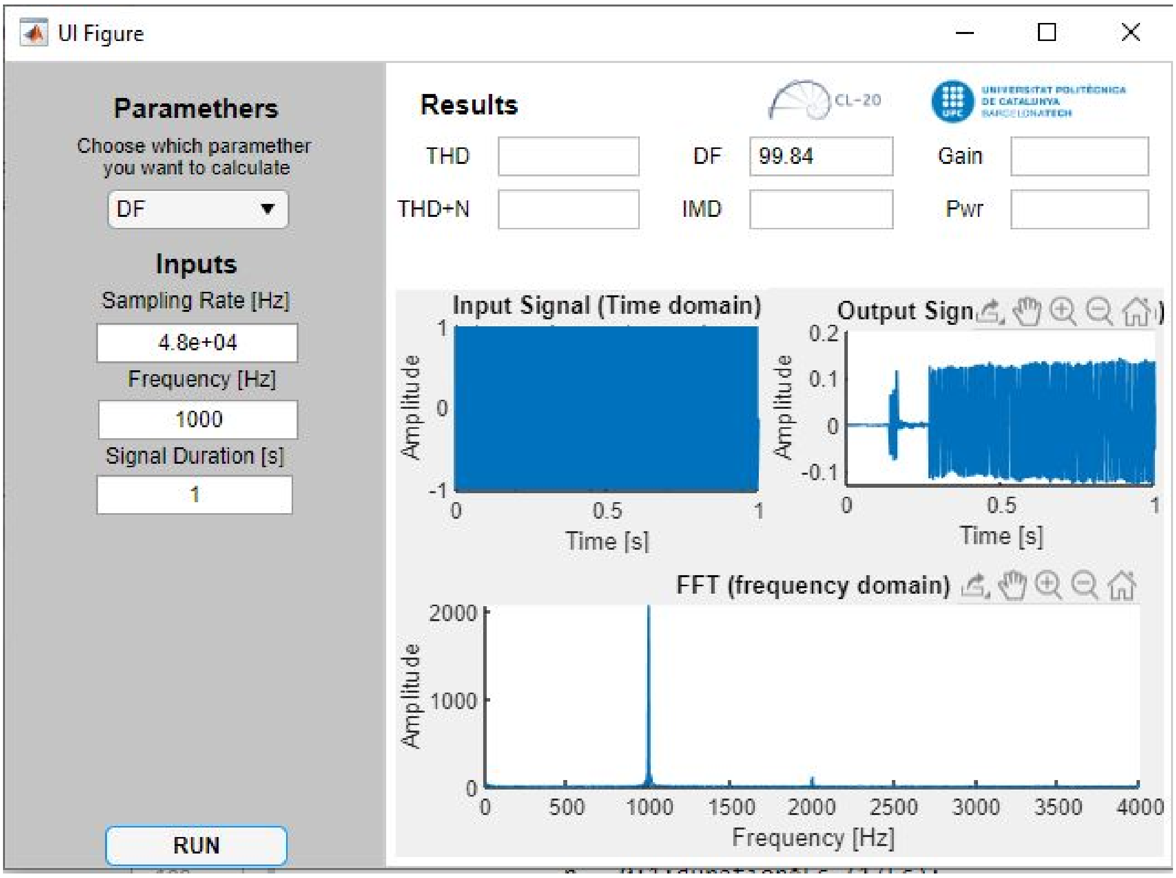


In the fft we can see three third order harmonics.

- The first one at the approximate frequency of 2600, which corresponds to twice the frequency 2 minus once the frequency 1: 2 * 1800 - 1000 = 2600.
- The second harmonic is found at a frequency close to 2800, which corresponds to twice the frequency 1 plus once the frequency 2: 2 * 1000 + 1800 = 3800.
- The third harmonic is found at a frequency close to 4600, which corresponds to once the frequency 1 plus twice the frequency 2: 1000 + 2 * 1800 = 4600.

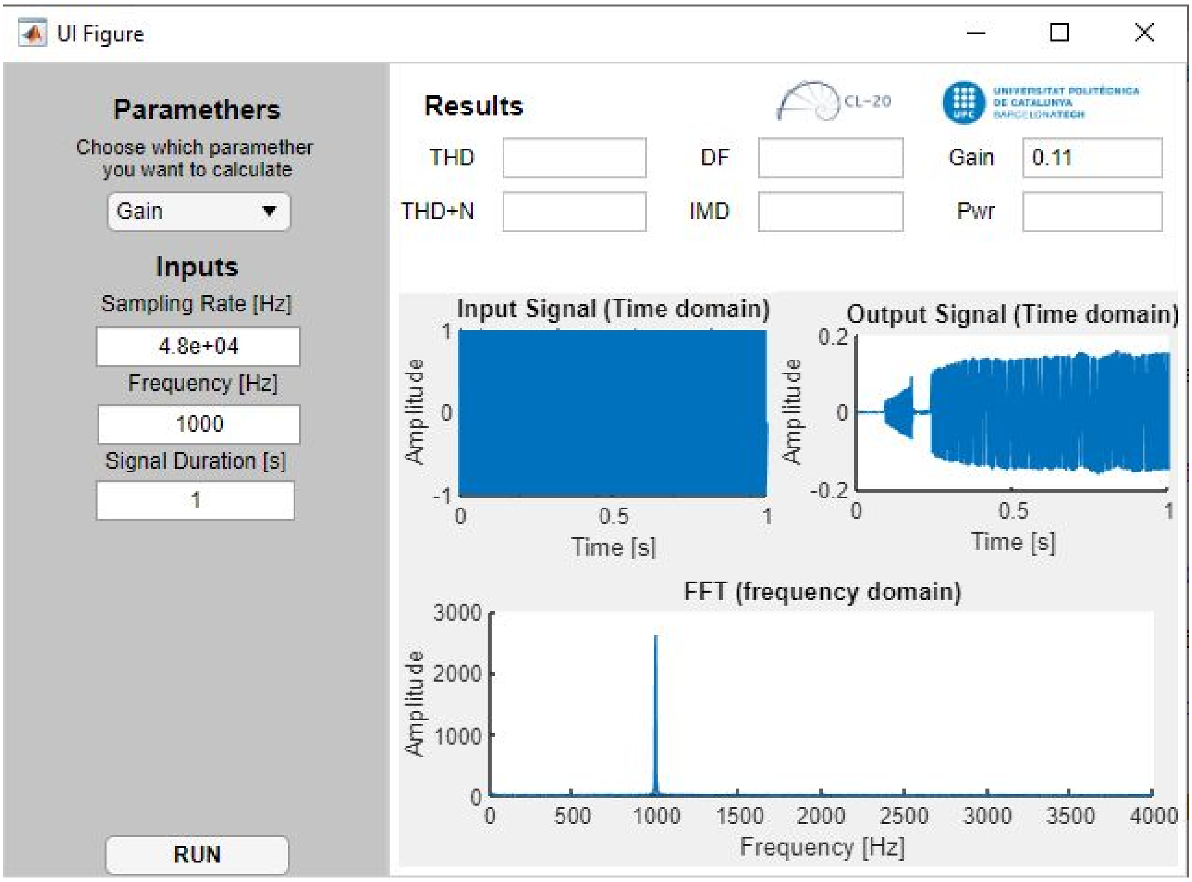
The three harmonics make the IMD 85.44%

DF



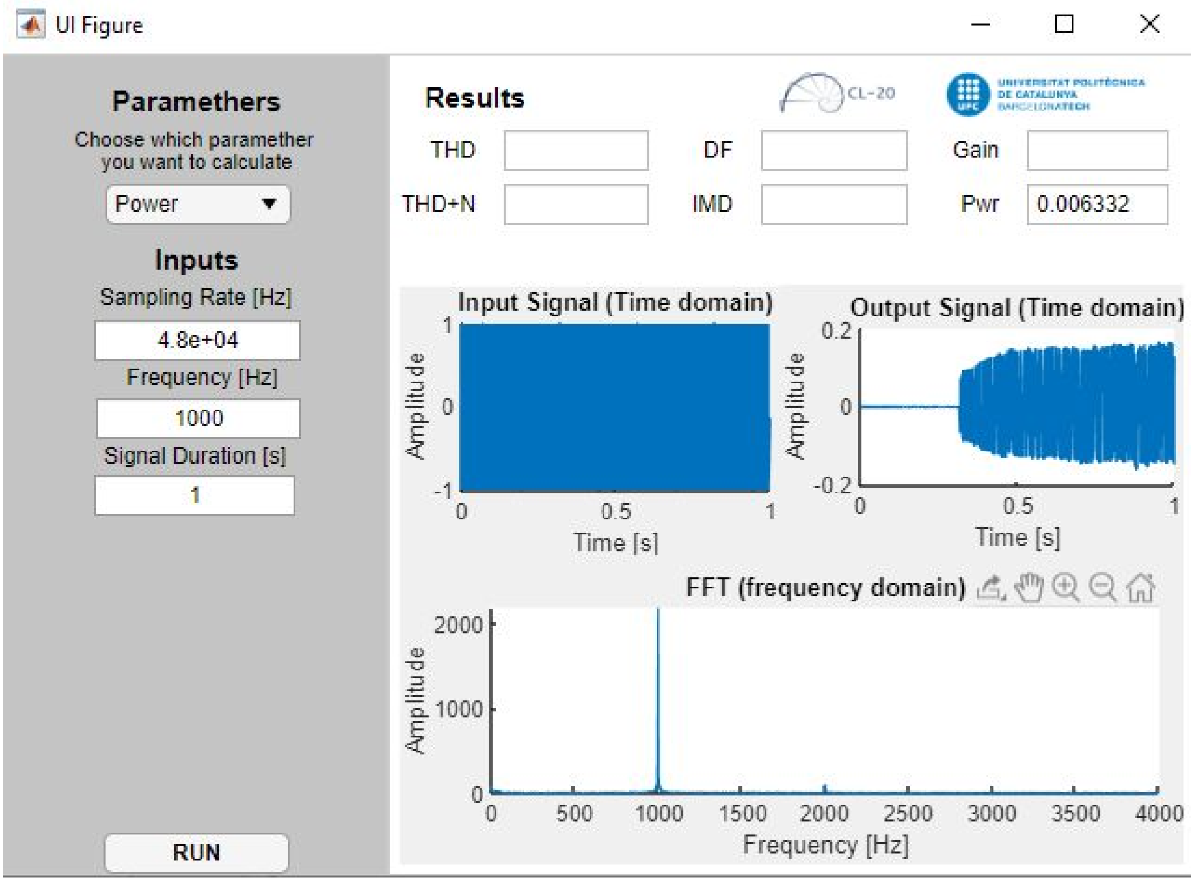
As we can see the secondary harmonics are practically negligible, so the value of the distortion factor is very close to 100%, it is 99.84%.

Gain



As we can see the gain obtained is approximately equal to the amplitude of the output signal because the amplitude of the input signal is 1.

Power



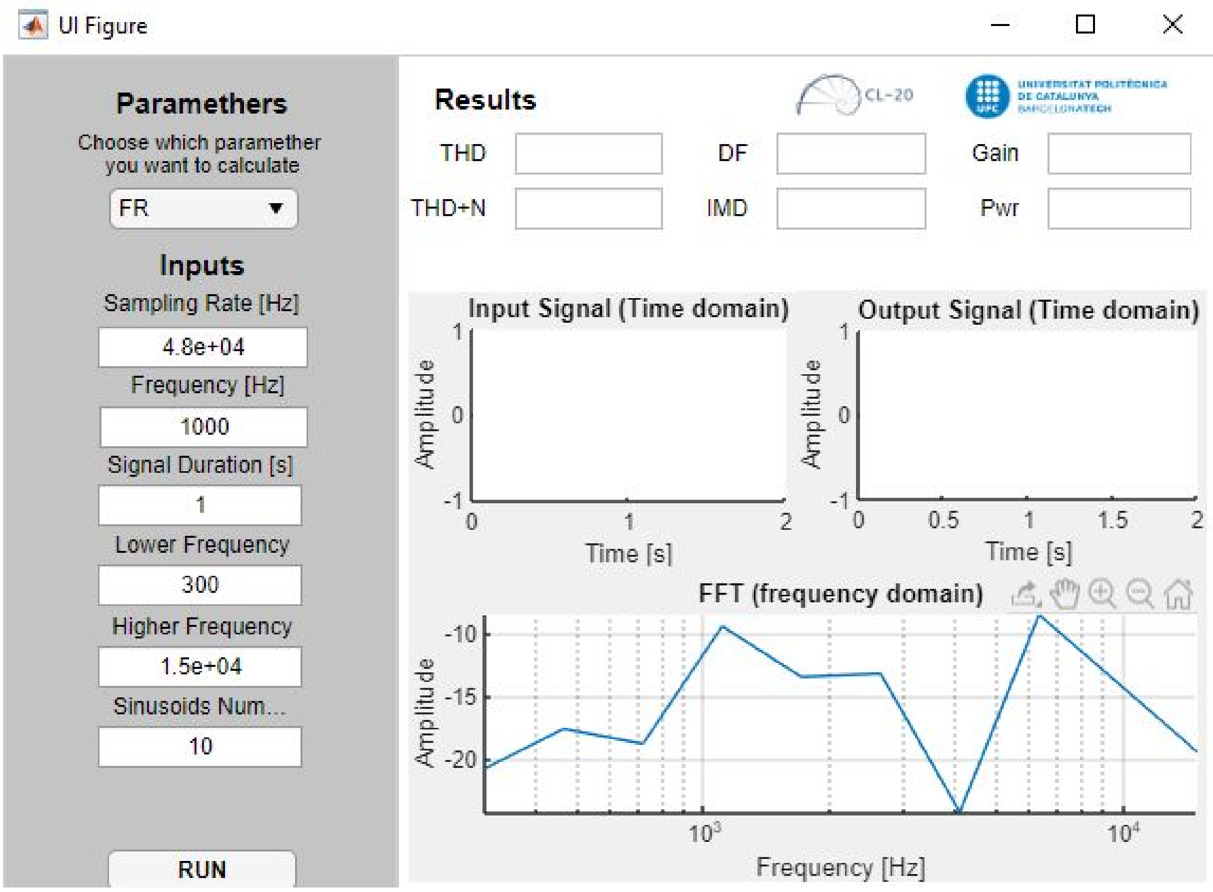
This power is lower than what we want but this is because the signal is not uniform as we can see and we have a variable delay so for this reason we cannot control this delay when we want to calculate any parameter because is not always the same value.

We have 48000 samples, if we consider a uniform signal without any delay and with an amplitude of 0.12, the power of signal would be:

$$(24000*0.12^2)/48000=0.0072 \text{ Watts}$$

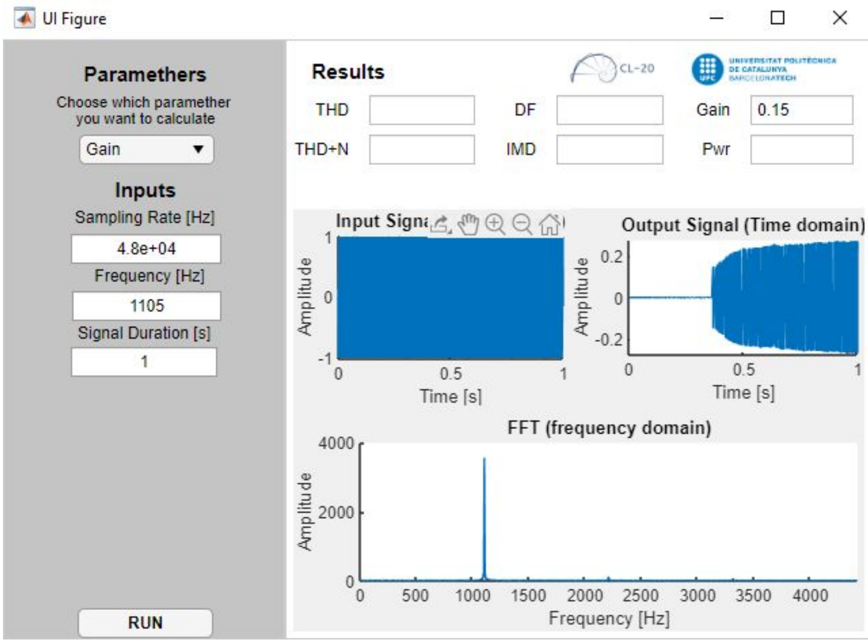
As we can see we obtain approximately the same result.

Frequency Response



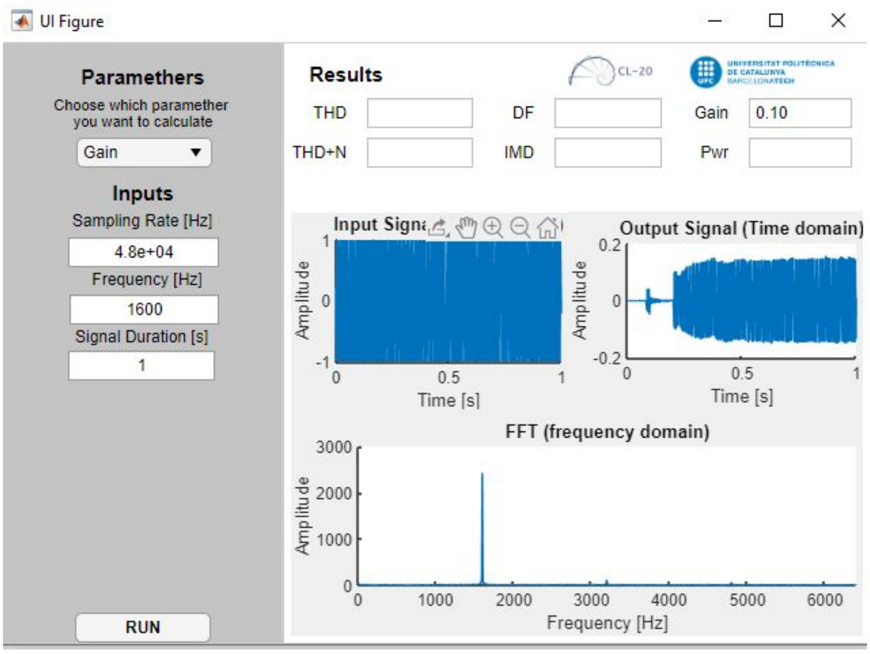
As we can see the frequency response is not the desired but this is because we don't have the necessary equipment prove our program correctly. Even this, we can prove that this result is a good estimation of the frequency response of our computer showing the signal generated in some frequencies and check that the result is correctly.

In the example above we can see that the gain at 1 kHz is 0.11 (-9.58 dB).
If we see the gain at 1105 Hz:



As we can see in the graph of the output signal the amplitude in this case is greater than for a 1 kHz case. The gain in dB in this case is -8.24 dB.

For 1600 Hz we obtain :



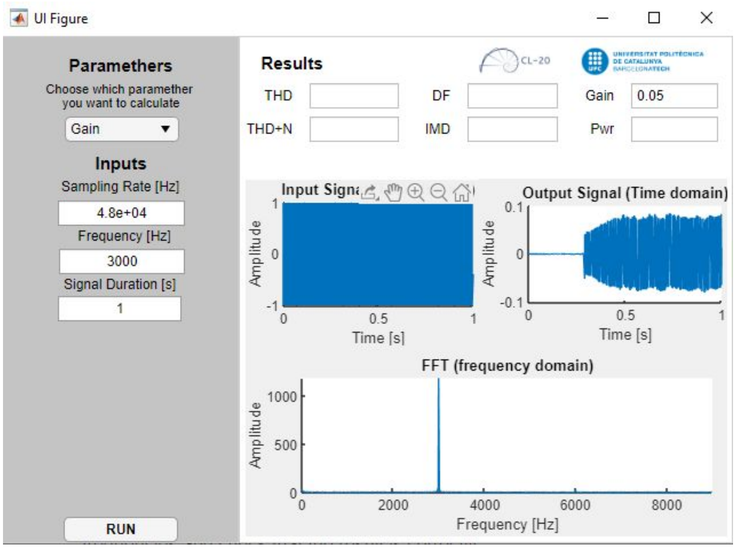
In dB is -10 dB.

So like this we can see and check the peak at 1105. Even without calculating the gain we can see in the graph of output signal how the amplitude increase at 1105 and decrease at 1600 Hz, like this we can justify this peak.

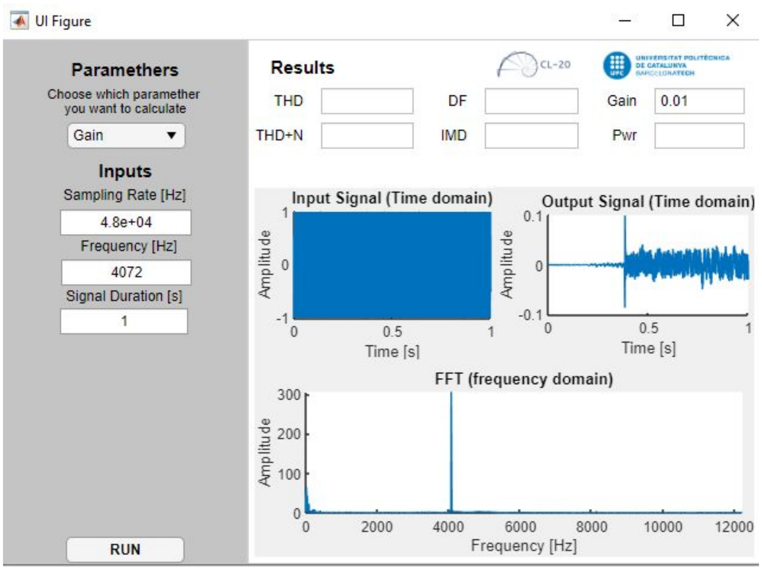
Let's see another case:

At 4072 Hz we can see a valley. To justify this case let's see frequencies below and above this value.

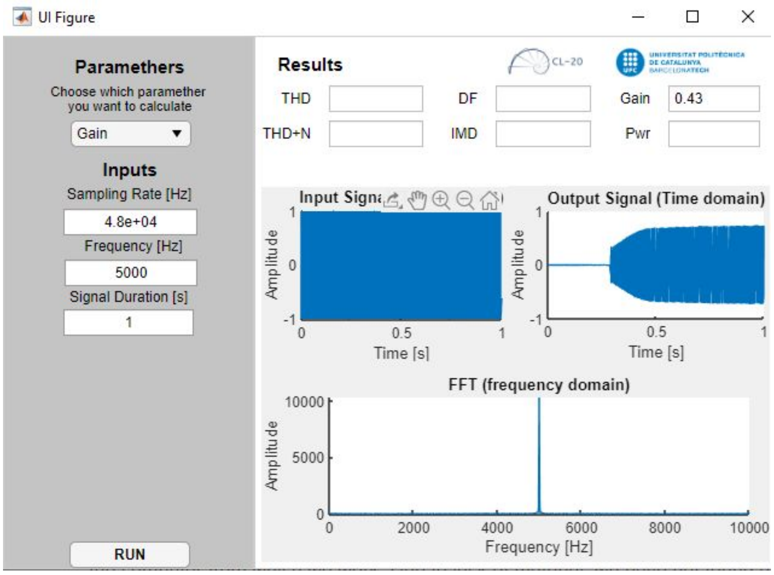
For 3000 Hz :



For 4072 Hz:



For 5000 Hz:

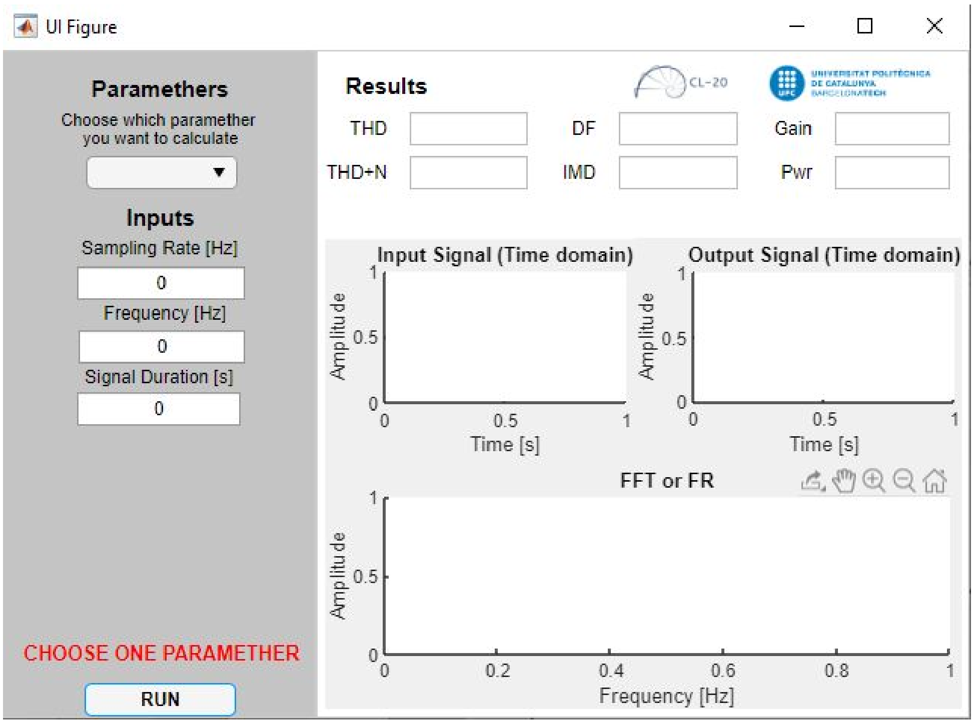


As we can see at 4072 we a decrease of the amplitude of the signal due to the properties of the audio card. Another important thing to comment is that we cannot obtain exactly the same value at a given frequency that's because we don't have an appropriate equipment but with an appropriate one we can check better that our program is functioning correctly.

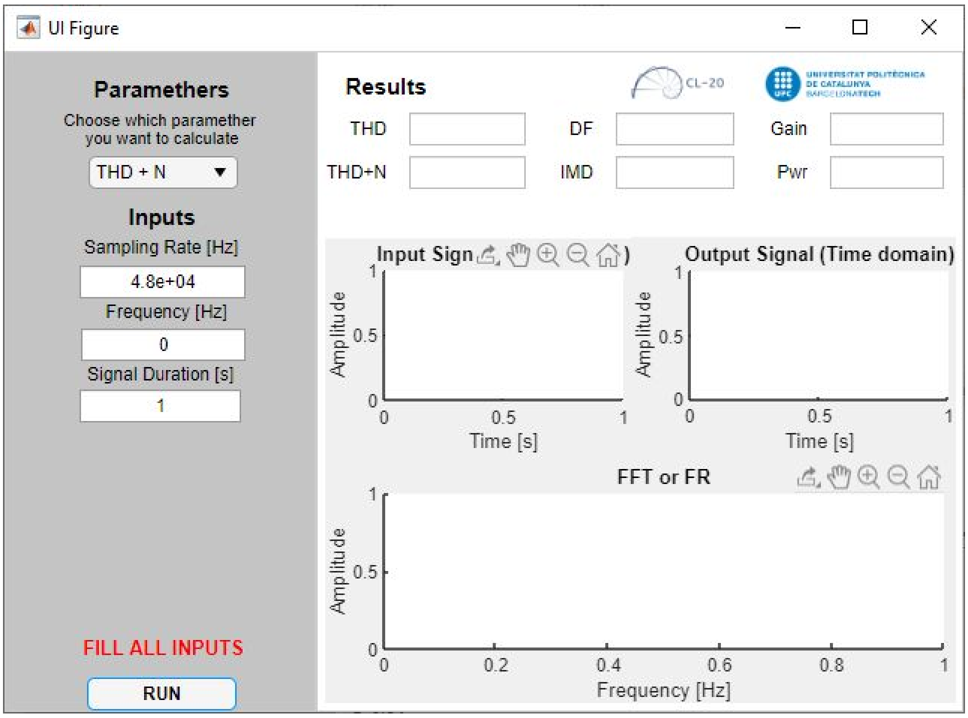
Most of the values we obtain are reasonable as we have seen. Still, some are different from theorists. We observed this problem as we developed the software. After a period of testing, in which we shuffled all possible sources of errors, we concluded that it was the cause of a problem in the computer from which we work. Due to lack of material, we have not found another computer to work and we have had to adapt to the situation. The parameters in which we find the greatest error are the gain, the estimation of the power and the frequency response. Even so, it must be borne in mind that this is an error on our computer and should not be seen on other devices.

In the application we have also made error control that allows that if the user has not correctly put the inputs, the application warns and does not close because of the error. There is an "Alert message" that alerts the user if they have not chosen any selected parameter to calculate or if they have not entered all the required parameters. In the images on the next page we can see these notices.

In this image we can see how there is no parameter selected, after pressing the run button the message "Choose one parameter" appears.



If any of the values that define the signal has not been filled, the message "Fill all inputs" appears.



6. BUDGET

Licensing Software Components
The user needs Matlab License. For an annual license the cost is 800 €.

Design and prototyping costs
Next we indicate the hours that we have dedicated to each work package. Keep in mind that the salary is 13 €/hour.

Work Package	Work Period	Hours * Person	Cost
WP 1	04/03/2020 - 24/03/2020	13 hours * 2	338 €
WP 2	24/03/2020 - 28/03/2020	5 hours * 2	130 €
WP 3	28/03/2020 - 31/03/2020	4 hours * 2	104 €
WP 4	13/04/2020 - 03/05/2020	24 hours * 2	624 €
WP 4*	04/05/2020 - 18/05/2020	19 hours * 2	494 €
WP 5	16/05/2020 - 18/05/2020	5 hours * 2	130 €
WP 6	-	0 hours	0 €
WP 7	29/05/2020 - 02/06/2020	11 hours * 2	286 €
TOTAL	**04/03/2020 - 02/06/2020**	**81 hours * 2**	**2106 €**

Thanks to the weekly reports that we have been making, where we indicate the hours worked each week, we know for sure that the count of hours per WP is exact.

In the project plan we calculate an approximate cost for salaries (without taking into account social security) of € 364 per month per person, which is a total of € 2,184.
The final total cost was 2106 €, a value very close to that expected.

Adding the costs of social security, the total would be 2808 €.

7. CONCLUSION

This course has given us the opportunity to do a project from scratch, without having any initial base as we are used to having in most laboratories that we do throughout the career. We have expanded our knowledge of signal processing, amplifiers, and application deployment in Matlab.
It has also allowed us to improve working with people we do not know, since before this laboratory the two members of the team did not know each other.

Due to the extraordinary situation that this course has experienced, different problems have arisen (inability to work in person, lack of laboratory material and insufficient resources, ...). The one that has influenced us most has been the lack of material, in this case computers, with a correct operation.

8. REFLECTION DOCUMENT

Our team is made up of two components, Chaimae Fathallah (Document Responsible and programmer) and Laura Pérez (Project Leader and programmer). The coordination between both team members has been very good, we have worked in equal parts throughout the entire project and we have not had any problems. Still, we believe there are things we could have improved upon.

One of the mistakes we have made has been that of not understanding the project from the first moment. Now reflecting on what the project has been, we see that it differs greatly from the initial idea we had. This has involved additional and unnecessary work that has increased the number of hours that we have subsequently had to dedicate to the project. This has also been the main cause of the failure we had in the mid-course demo. At that time we thought that what we were showing in the presentation was what we had to do.

Another mistake we have had has been the irregularity regarding weekly working hours. At first we plan 7 hours of weekly work, which we finally have not done. For academic and personal reasons, some weeks we have not been able to dedicate these hours, but most weeks we have done more.

Finally, we think that the lack of more members in the team has meant that we have had to distribute the work equally and both have assumed all the roles that there may be in a team (project leader, document responsible, programmer, lecturers, ...). We do not consider it to be a mistake, we have simply considered that it was the best way to work in our situation.

We don’t think that any of the members of the team has worked more than the other, for this reason we think that the score of the project should be shared equally between the two.

We have not encountered great difficulties in implementing the software. Even so, initially there were errors in the calculation of the parameters when we started working with signals recorded by the computer. After a study of the possible factors that could cause these errors, we saw that one of the two computers we had for the project had a problem that did not allow us to record the signals well and caused the application to not work correctly. From that moment (at the end of April) we were only able to work with a computer, which made the work difficult. The computer that if it worked, gave problems when recording the signals, since the recording was choppy. Not having the complete signal available, we have had problems mainly in calculating the frequency response. Other parameters have also been altered by this error, such as profit or power estimation.

Another problem caused by the computer has been that being in a meet session, the playback and recording behavior of the computer were altered. For this reason, the final presentation of the project could not be done at the time. To solve the problem, we recorded a video where we show the calculation of the different parameters by varying the inputs. For tests required by the teachers, one of the members of the group entered the meet session from the phone and recorded the computer screen while running the program. We would have liked to have made a better presentation, with greater quality and fluidity, but given the circumstances, we believe that we did the best we could.

9. ANNEX

All the code regarding the project can be found in our GitHub repository.

You can find the scripts that calculate each of the parameters of the application, as well as some test that we have done throughout the project and all the documents generated.