



รายงานการทดลอง

Pre-Project

Embedded Device for Audio Signal Analysis and Classification of Musical Notes Using Machine Learning

จัดทำโดย

นางสาวปุณชรัสมี วงศ์คำ	รหัสนักศึกษา	6301012620146
นายม.โนนพศ ยานศสกุลวัฒนา	รหัสนักศึกษา	6301012630141

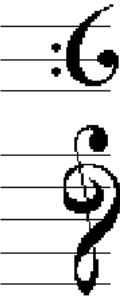
เสนอ

ดร.เรวัต ศิริ โภคากิริมย์

สาขาวิชาวิศวกรรมคอมพิวเตอร์
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
ปีการศึกษา พ.ศ.2566

ในการทดลองนี้ ได้เลือกใช้เปียโนเป็นเครื่องดนตรีที่ใช้ในการทดลอง ซึ่งจะมีความถี่ในแต่ละคีย์ดังนี้

MIDI number	Note name	Keyboard	Frequency Hz	Period ms
21 22	A0		27.500	36.36
23	B0		30.868	32.40 34.32
24 25	C1		32.703	30.58
26 27	D1		36.708	34.648 27.24 28.86
28	E1		41.203	38.891 24.27 25.71
29 30	F1		43.654	22.91
31 32	G1		48.999	46.249 20.41 21.62
33 34	A1		55.000	51.913 18.18 19.26
35	B1		61.735	58.270 16.20 17.16
36	C2		65.406	15.29
38 37	D2		73.416	69.296 13.62 14.29
40	E2		82.407	77.782 12.13 12.86
41 42	F2		87.307	11.45
43 44	G2		97.999	92.499 10.20 10.81
45 46	A2		110.00	103.83 9.091 9.631
47	B2		123.47	116.54 8.099 8.581
48	C3		130.81	7.645
50 51	D3		146.83	138.59 6.811 7.216
52	E3		164.81	155.56 6.068 6.428
53 54	F3		174.61	5.727
55 56	G3		196.00	185.00 5.102 5.405
57 58	A3		220.00	207.65 4.545 4.816
59	B3		246.94	233.08 4.050 4.290
60	C4	261.63	3.822
62 63	D4		293.67	277.18 3.405 3.608
64	E4		329.63	311.13 3.034 3.214
65 66	F4		349.23	2.863
67 68	G4		392.00	369.99 2.551 2.703
69	A4	440.00	415.30 2.273 2.408
71	B4		493.88	466.16 2.025 2.145
72	C5		523.25	1.910
74 75	D5		587.33	554.37 1.703 1.804
76	E5		659.26	622.25 1.517 1.607
77 78	F5		698.46	1.432
79 80	G5		783.99	739.99 1.276 1.351
81 82	A5		880.00	830.61 1.136 1.204
83	B5		987.77	932.33 1.012 1.073
84 85	C6		1046.5	0.9556
86 87	D6		1174.7	1108.7 0.8513 0.9020
88	E6		1318.5	1244.5 0.7584 0.8034
89 90	F6		1396.9	0.7159
91 92	G6		1568.0	1480.0 0.6378 0.6757
93 94	A6		1760.0	1661.2 0.5682 0.6020
95	B6		1975.5	1864.7 0.5062 0.5363
96 97	C7		2093.0	0.4778
98 99	D7		2349.3	2217.5 0.4257 0.4510
100	E7		2637.0	2489.0 0.3792 0.4018
101 102	F7		2793.0	0.3580
103 104	G7		3136.0	2960.0 0.3189 0.3378
105 106	A7		3520.0	3322.4 0.2841 0.3010
107	B7		3951.1	3729.3 0.2531 0.2681
108	C8		4186.0	0.2389



J. Wohl, UNSW

Signal Generation and Measurement

ສັງລະນະ Harmonic ທີ່ເລືອກໃຊ້

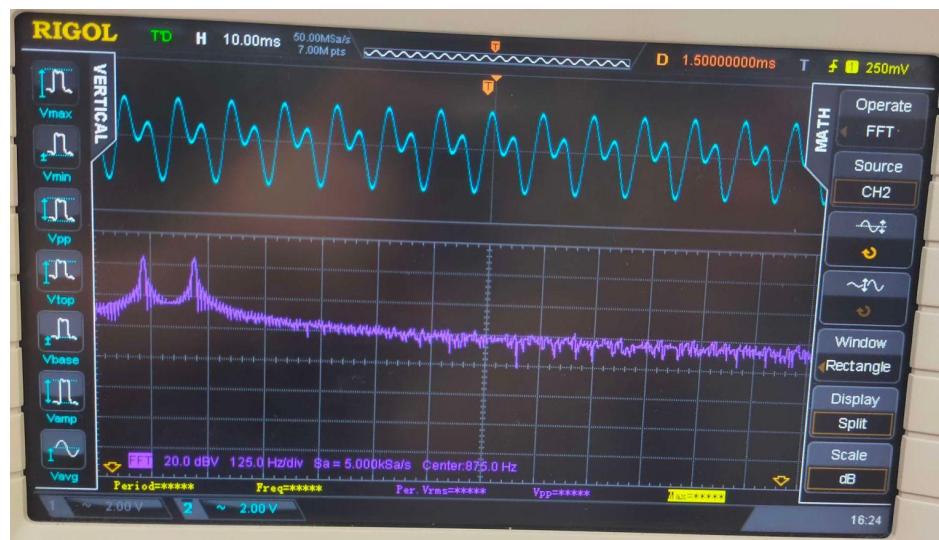
A. A2 : 110 Hz , A3 : 220 Hz

a. Ampl = 5 Vpp , No offset

ເສັ້ນສີເຫຼືອງແສດງຄົງ Input 110 Hz ແລະເສັ້ນກາຟິຟ້າແສດງຄົງ 220 Hz



b. FFT



B. A2 : 110 Hz , E4 : 330 Hz

a. FFT



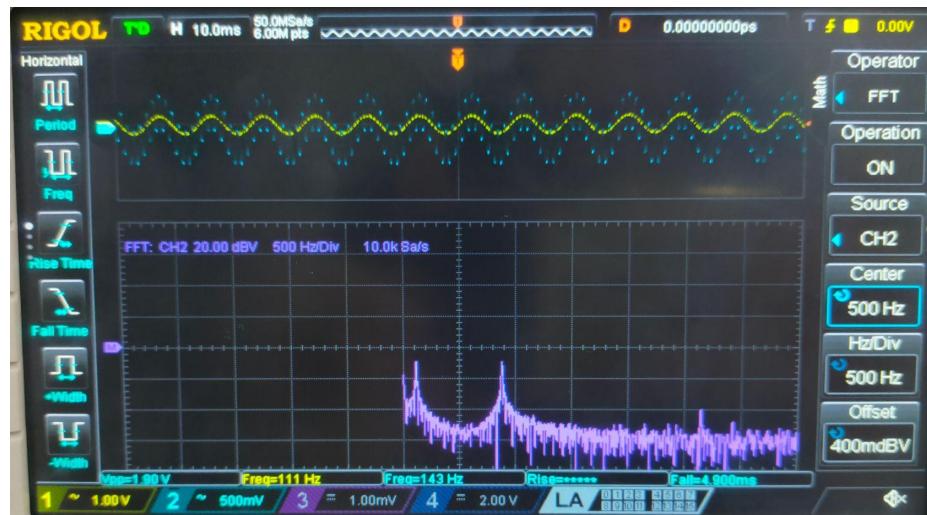
C. A2 : 110 Hz , A5 : 880 Hz

a. Ampl = 5 Vpp , No offset

เส้นสีเหลืองแสดงถึง Input 110 Hz และเส้นกราฟสีฟ้าแสดงถึง Summing ระหว่าง 110 Hz และ 880 Hz



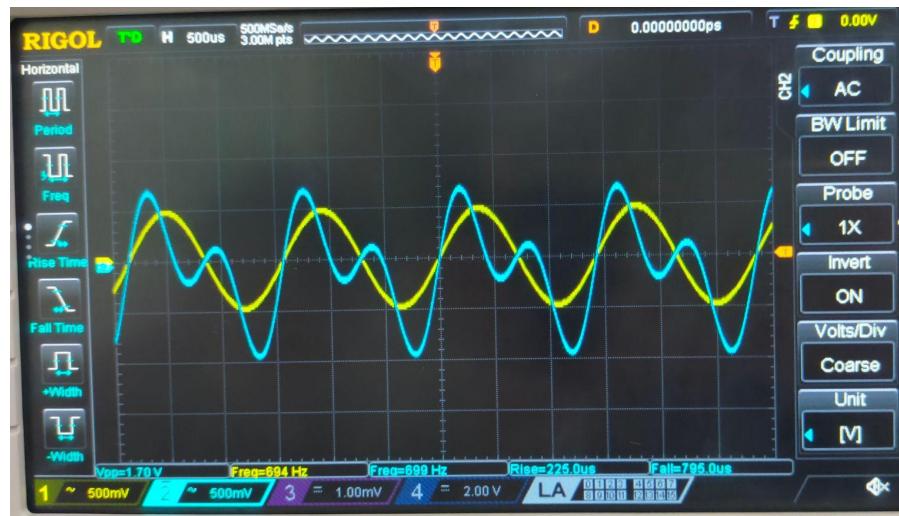
b. FFT



D. F5 : 700 Hz , F6 : 1400 Hz

a. Ampl = 5 Vpp , No offset

เส้นสีเหลืองแสดงถึง Input 700 Hz และเส้นกราฟสีฟ้าแสดงถึง Summing ระหว่าง 700 Hz และ 1400 Hz



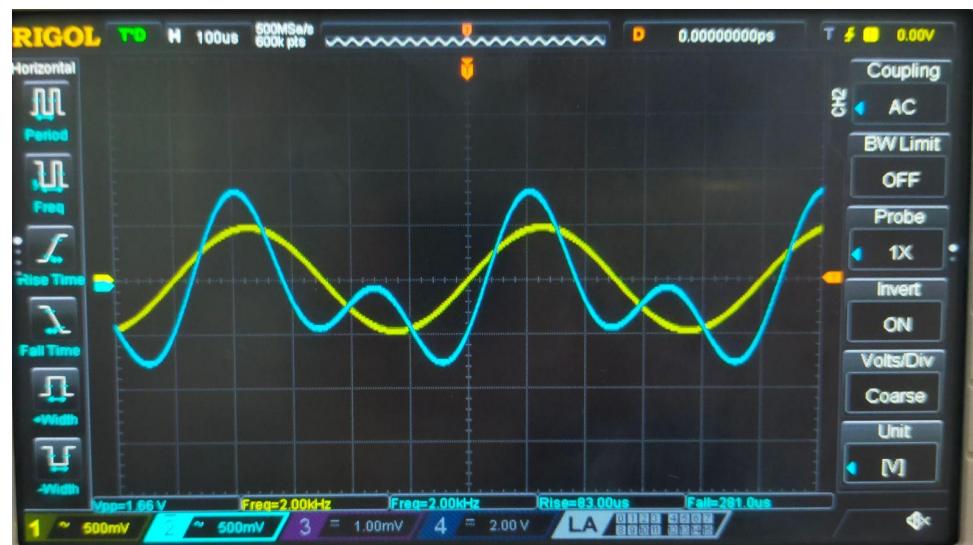
b. FFT



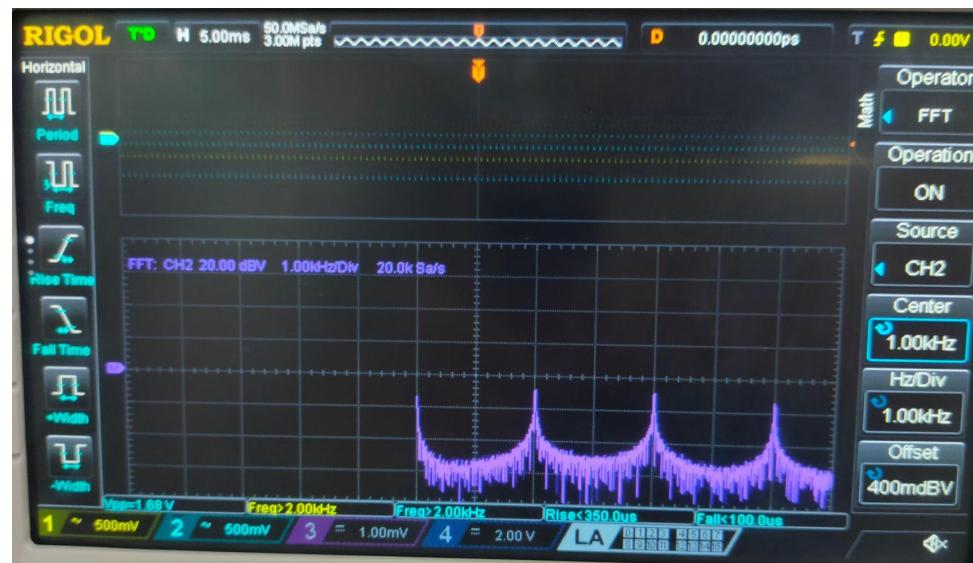
E. B6 : 2000 Hz , B7 : 4000 Hz

a. Ampl = 5 Vpp , No offset

เส้นสีเหลืองแสดงถึง Input 2000 Hz และเส้นกราฟสีฟ้าแสดงถึง Summing ระหว่าง 2000 Hz และ 4000 Hz



b. FFT



Write MATLAB/Python code to mathematically generate and visualize synthetic data, then compare it with the recorded data from the experiment.

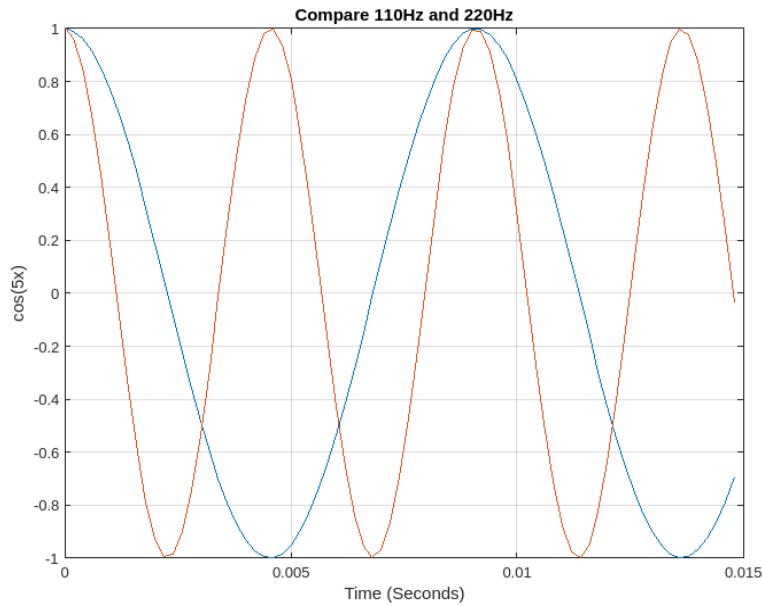
Matlab code

FFT

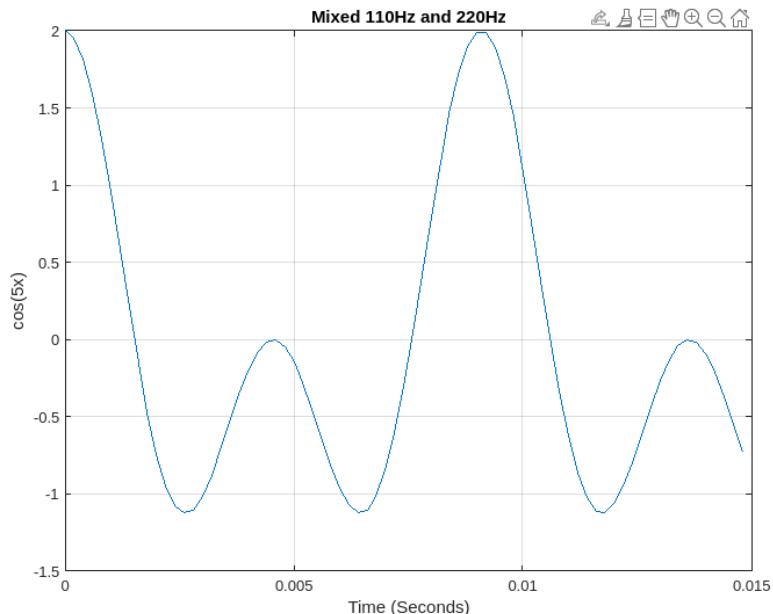
```
Fs = 8000; % Sampling frequency  
T = 1/Fs; % Sampling period  
L = 1500; % Length of signal  
f1 = 110;  
f2 = 220;  
t = (0:L-1)*T; % Time vector  
S = 0.7*sin(2*pi*f1*t) + sin(2*pi*f2*t);  
X = S + 2*randn(size(t));  
Y = fft(X);  
P2 = abs(Y/L);  
P1 = P2(1:L/2+1);  
P1(2:end-1) = 2*P1(2:end-1);  
f = Fs*(0:(L/2))/L;  
plot(f,P1)  
title("Single-Sided Amplitude Spectrum of X(t)")  
xlabel("f (Hz)")  
ylabel("|P1(f)|")
```

รูปถูกคลื่นแสดงผล

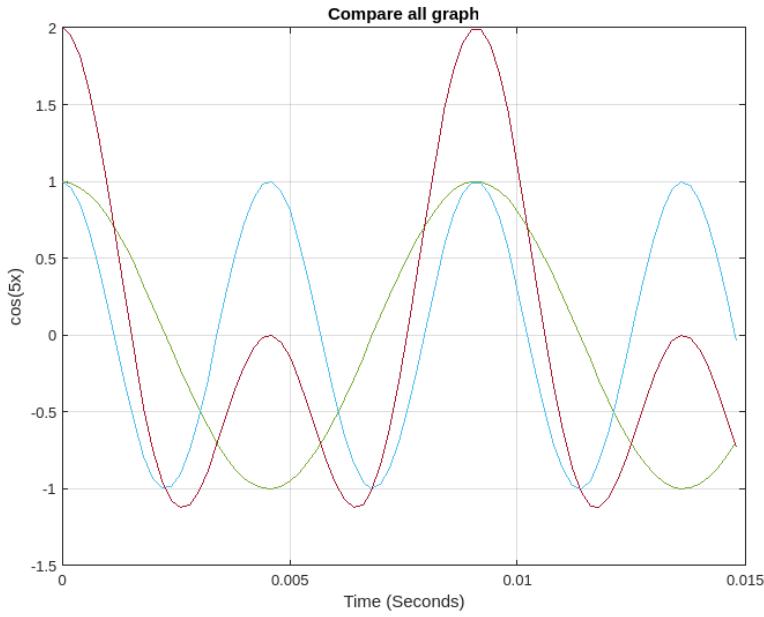
เลือกใช้ความถี่ที่ 110 Hz และ 220 Hz



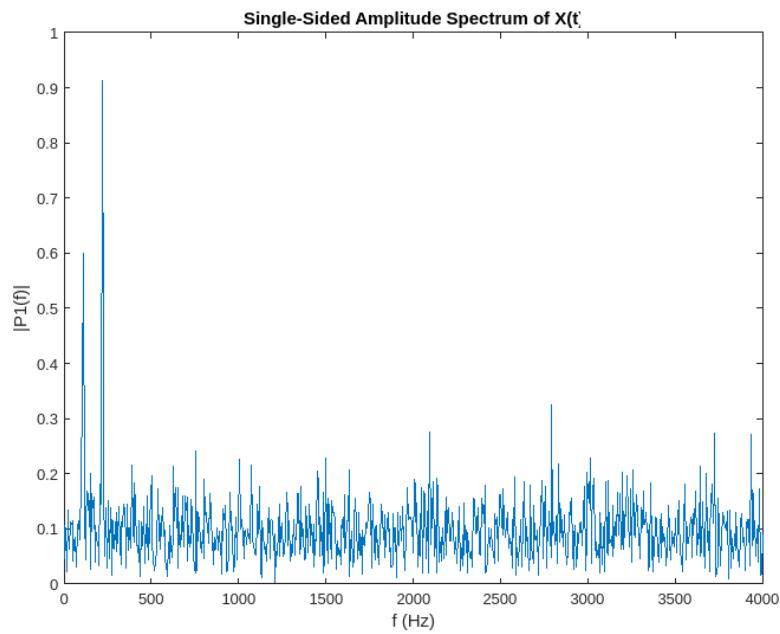
เปรียบเทียบรูปถูกคลื่นความถี่ที่ 110 Hz และ 220 Hz



นำความถี่ที่ 110 Hz และ 220 Hz มารวมกัน

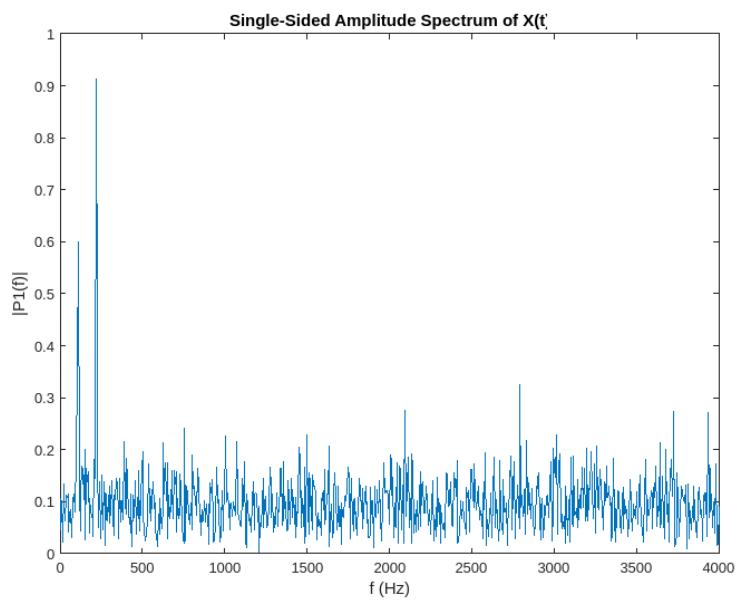
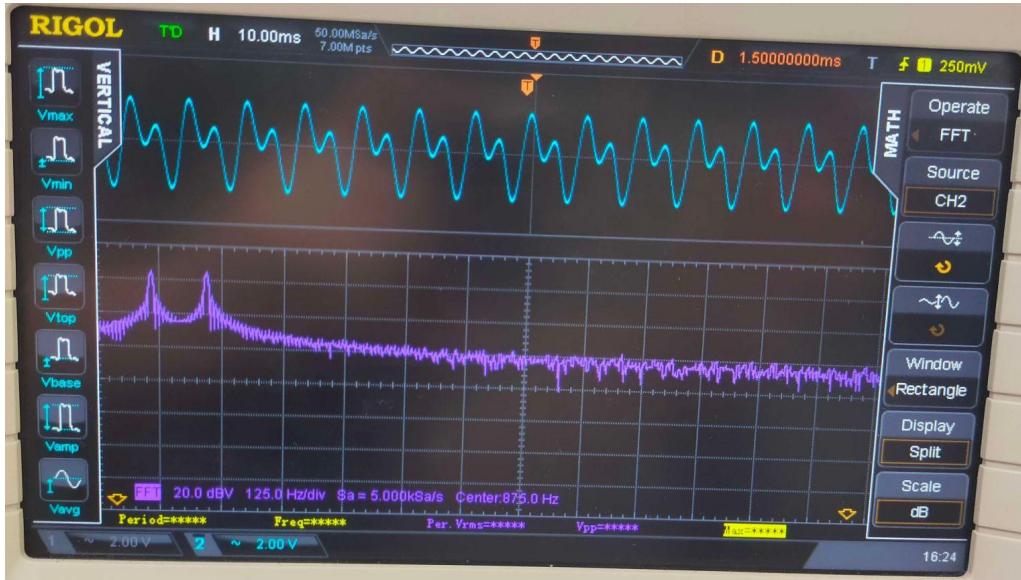


เปรียบเทียบรูปคลื่นที่ความถี่ 110 Hz , 220 Hz และผลรวมของทั้งสองคลื่น



รูปคลื่นที่ผ่าน FFT

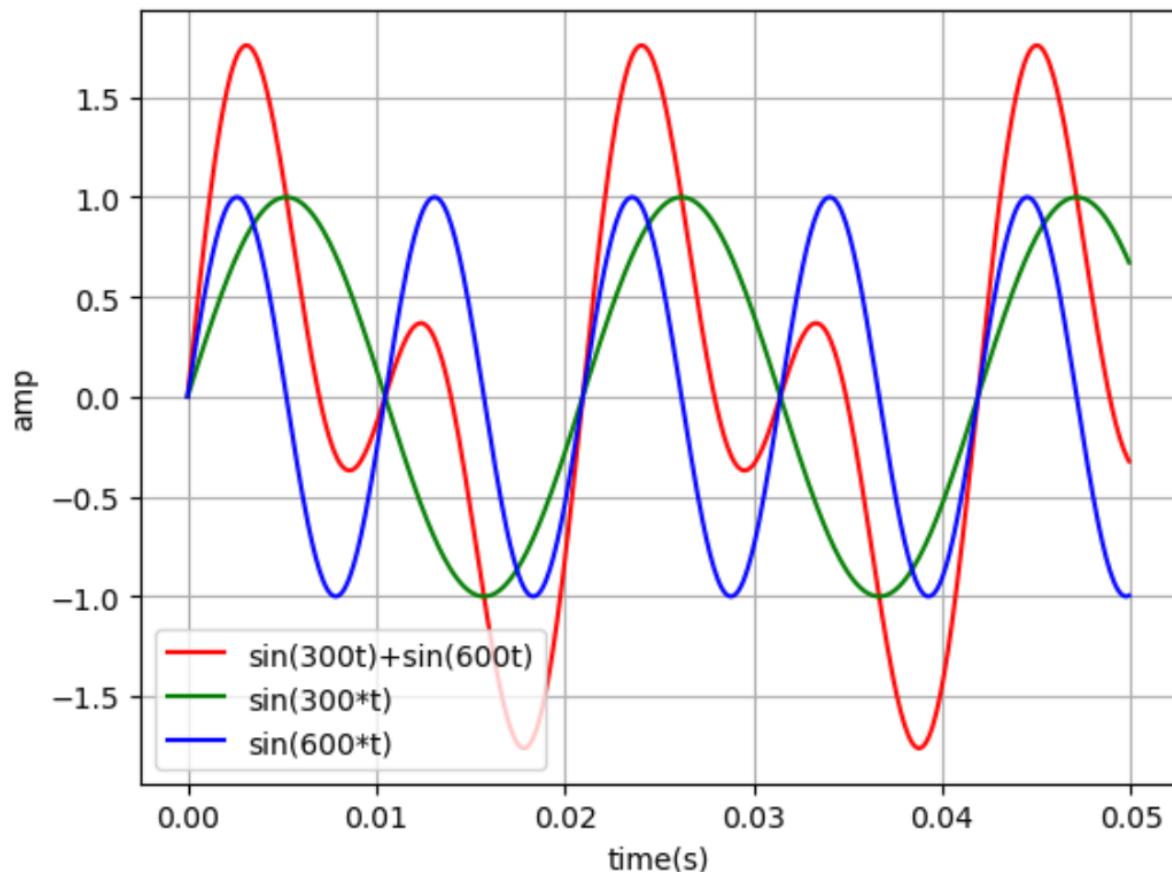
นำรูปกราฟที่ได้จากการทดลองจริงและรูปคลื่นที่ได้จากการเขียนโปรแกรมมาเปรียบเทียบกัน ดังนี้



จะเห็นว่ารูปกราฟของ FFT สามารถแสดงผลการจำแนกคลื่นความถี่ได้ใกล้เคียงกับสัญญาณที่ป้อนเข้าไป รวมถึงทั้งสองรูปมีการแสดงผลรูปคลื่นที่ใกล้เคียงกัน แสดงให้เห็นว่าการทดลองจริงไม่ได้มีความคลาดเคลื่อนที่มากจนไม่สามารถแยกออกได้

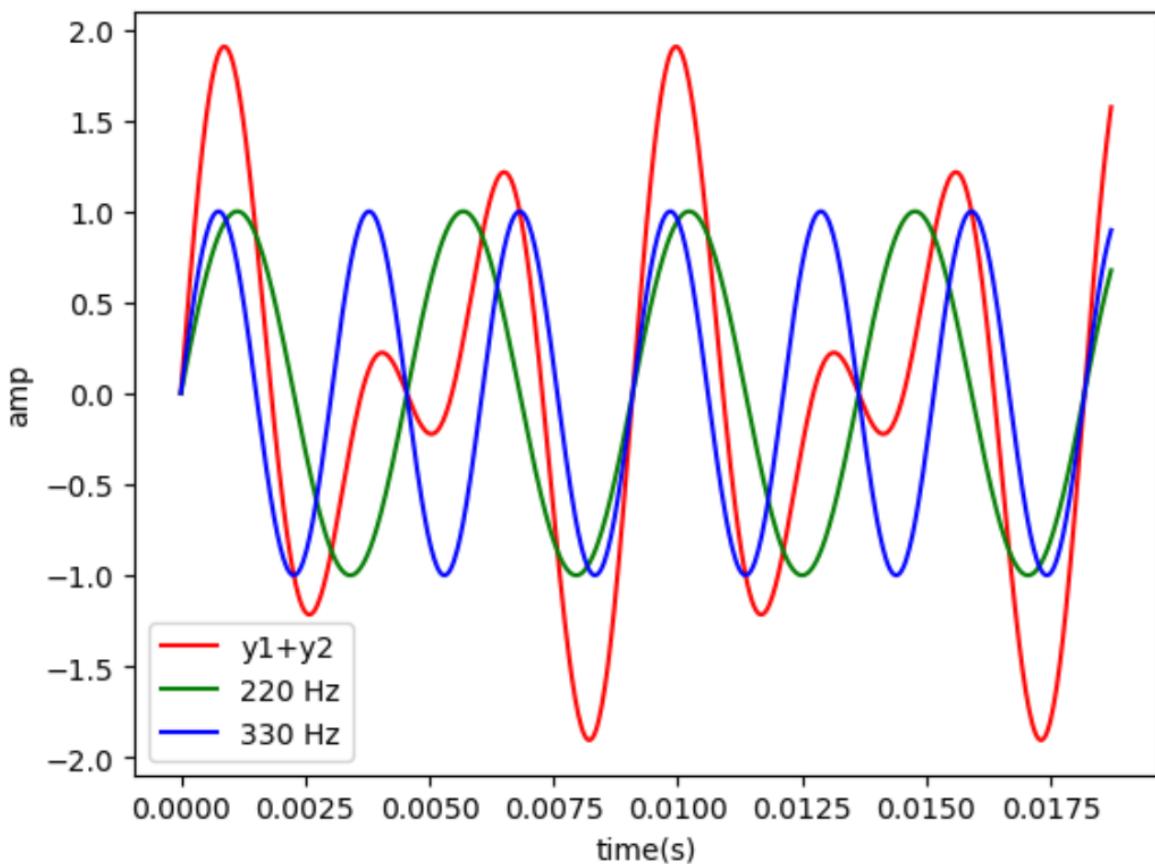
Code Python ทำการบวกความถี่ 300 Hz และ 600 Hz

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,0.05,0.0001) # start,stop,step
plt.grid(True)
y1 = np.sin(300*x)
y2 = np.sin(600*x)
y3 = y1+y2
plt.xlabel('time(s)')
plt.ylabel('amp')
plt.plot(x, y3, color='r', label='sin(300t)+sin(600t)')
plt.plot(x, y1, color='g', label='sin(300*t)')
plt.plot(x, y2, color='b', label='sin(600*t)')
plt.legend()
plt.show()
```



ตัวอย่างที่ 2 ทำการรวมคลื่นความถี่ 220 Hz และ 330 Hz

```
# Number of sample points
N = 600
# sample spacing
T = 1.0 / 800.0
x = np.linspace(0.0, 15*T, 600, endpoint=False)
y1 = np.sin(220 *2.0*np.pi*x)
y2 = np.sin(330 *2.0*np.pi*x)
y3 = y1+y2
plt.xlabel('time(s)')
plt.ylabel('amp')
plt.plot(x, y3, color='r', label='y1+y2')
plt.plot(x, y1, color='g', label='220 Hz')
plt.plot(x, y2, color='b', label='330 Hz')
plt.legend()
plt.show()
```



FFT

```
from scipy.fft import fft, fftfreq

import numpy as np

# Number of sample points

N = 600

# sample spacing

T = 1.0 / 800.0

x = np.linspace(0.0, N*T, N, endpoint=False)

y1 = np.sin(220 *2.0*np.pi*x)
y2 = np.sin(330 *2.0*np.pi*x)
y3 = y1+y2

yf = fft(y3)

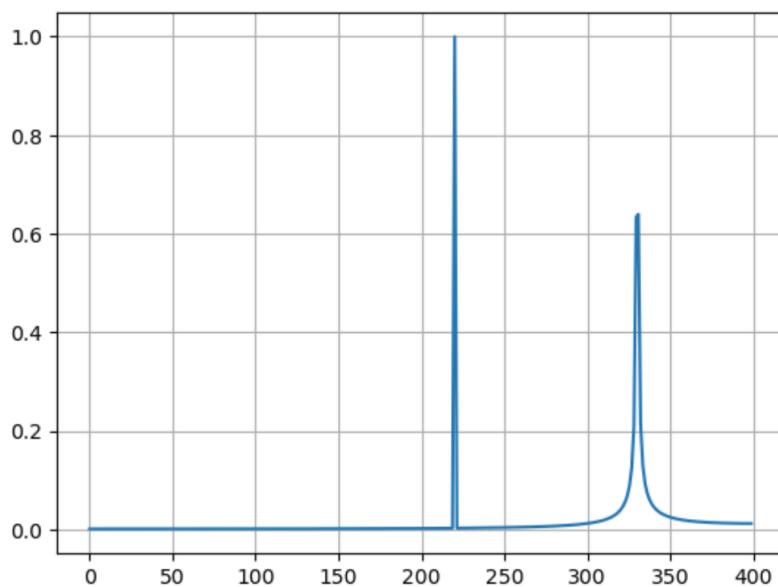
xf = fftfreq(N, T)[:N//2]

import matplotlib.pyplot as plt

plt.plot(xf, 2.0/N * np.abs(yf[0:N//2]))

plt.grid()

plt.show()
```



ทดสอบการใช้ไมโครโฟน

ทดสอบการใช้ไมโครโฟนรับคลื่นความถี่ 300 Hz และ 3000 Hz ตามลำดับ



ไมโครโฟนรับคลื่นความถี่ 300 Hz

จากการจะเห็นว่า รูปคลื่นแต่ละรูปนั้นมีสัญญาณรบกวนเข้ามา เนื่องจากในสถานที่ที่ทดลองนั้นไม่ได้เงียบ ทำให้รูปคลื่นไม่เรียบสวยงาม จากการทดลองใช้ไมโครโฟนรับคลื่นความถี่ 300 Hz เนื่องจากความถี่ที่ป้อนเข้ามานั้นเป็นความถี่ที่ต่ำ ทำให้มีสัญญาณรบกวนเข้ามา จึงแสดงผลอย่างชัดเจน

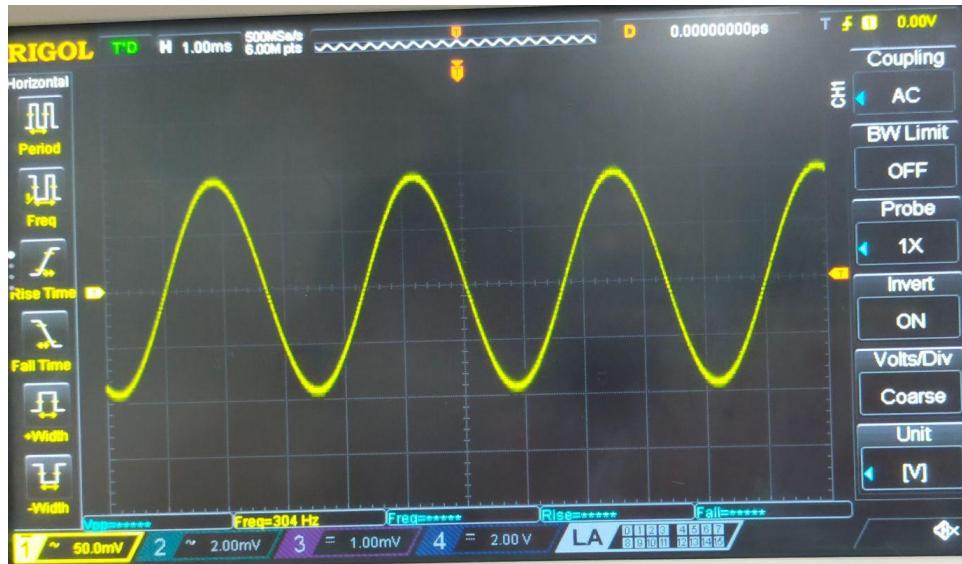


ไมโครโฟนรับคลื่นความถี่ 3000 Hz

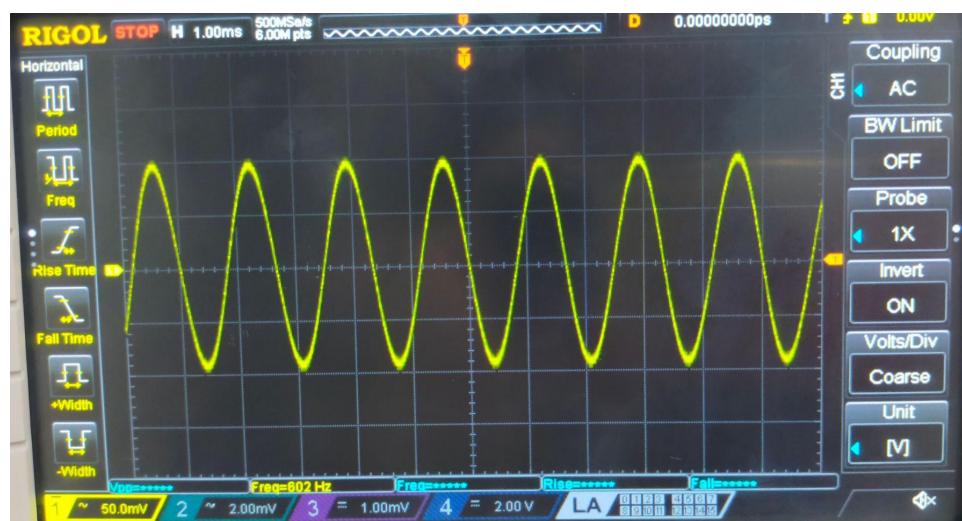
จากการจะเห็นว่า ความถี่ที่ป้อนเข้าไปมีระดับที่สูง ทำให้ไม่สามารถสังเกตเห็นสัญญาณรบกวนเข้ามาได้ เพราะสัญญาณรบกวนมีความถี่ไม่มากนักเมื่อเทียบกับความถี่ 3000 Hz แต่เพราะเข่นนั้น จึงทำให้รูปคลื่นมีลักษณะคลื่นไหวไปมาไม่คงที่ อันเป็นเพราความถี่ที่สูง

ทดสอบการใช้สายเก็บรับค่าความถี่เสียง

- ทดสอบโดยใช้ audio jack connectors ร่วมกับโทรศัพท์มือถือ



ความถี่ 300 Hz



ความถี่ 600 Hz



ความถี่ 10 kHz

2) ทดสอบโดยใช้ audio jack connectors ร่วมกับ Laptop



ความถี่ 300 Hz



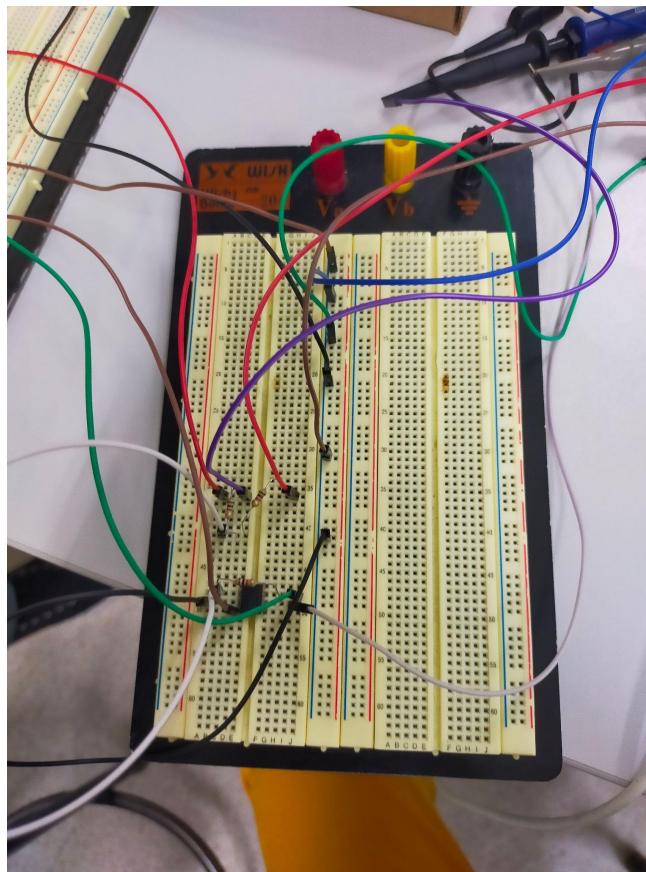
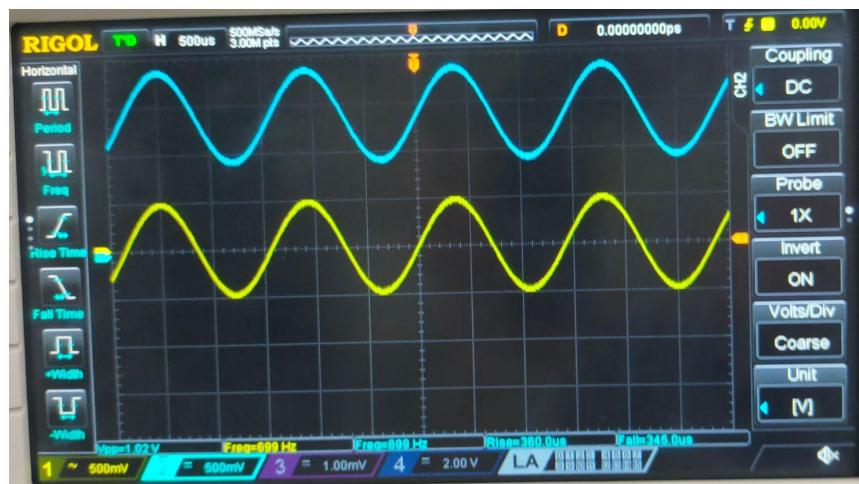
ความถี่ 600 Hz



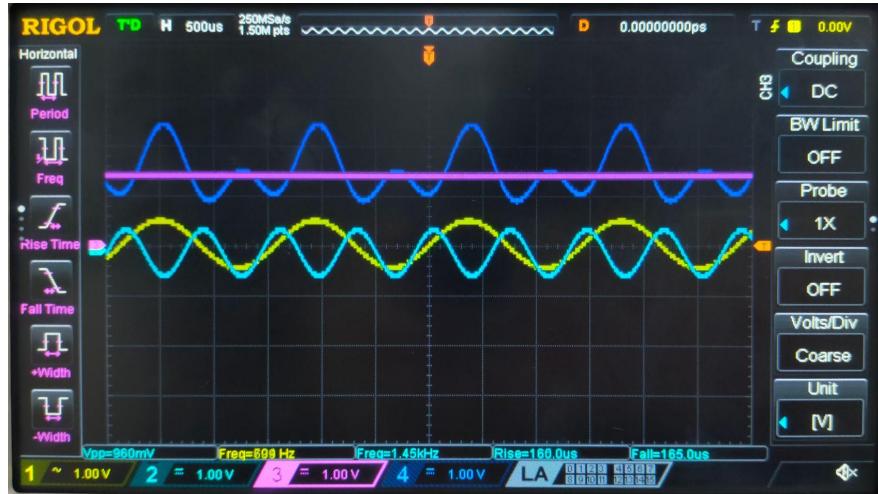
ความถี่ 10 kHz

Signal Level Shifting: Use a voltage level shifting circuit to convert the AC signal

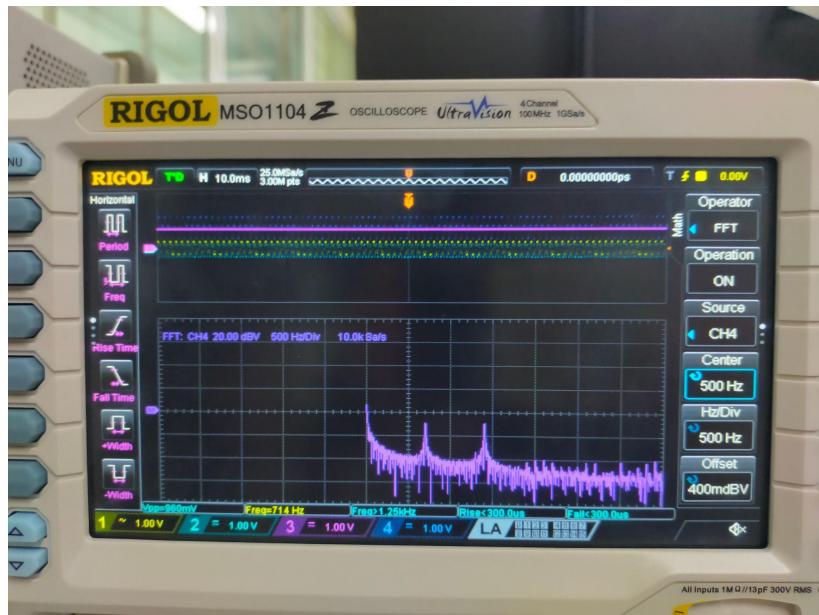
กำหนดให้ V_{pp} เป็น 1 V และมีความถี่ Input เท่ากับ 700 Hz ทำให้ V_p จะมีค่าอยู่ที่ 0.5 V และ -0.5 V เมื่อนำไป Offset ที่มีค่าเท่ากับ 1.5 V จะแสดงให้เห็นว่า V_p จะมีแรงดันอยู่ที่ราว 2 V และ 1 V



เมื่อเพิ่มความถี่ที่มีค่า 1400 Hz เป็น Input ที่ 2 และนำความถี่ Input ทั้ง 2 Inputs มา Summing รวมกับ Voffset 1.5 V ทำให้ได้รูปกราฟดังนี้



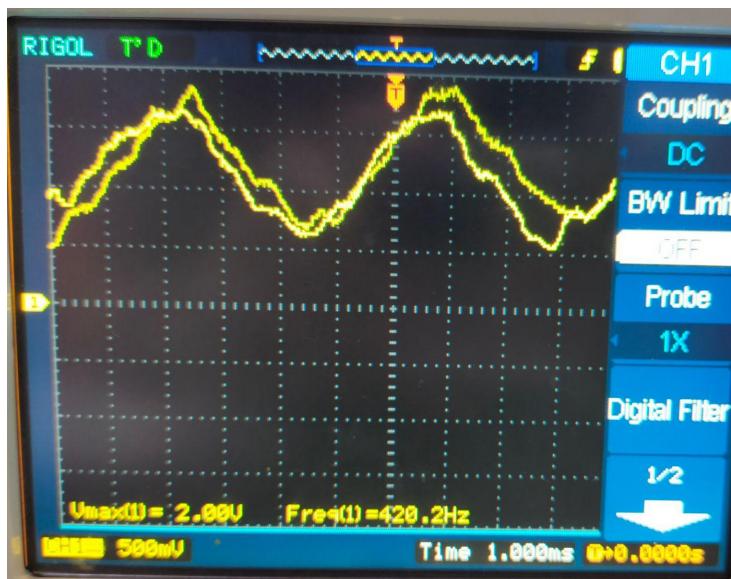
โดยเส้นสีเหลืองคือ Input ที่มีความถี่ 700 Hz, เส้นสีฟ้าคือ Input ที่มีความถี่ 1400 Hz, เส้นสีม่วงคือ Offset ที่มีค่า 1.5 V และสีน้ำเงินคือความถี่ที่เกิดจากการรวมกันของทั้ง 2 ความถี่ Input



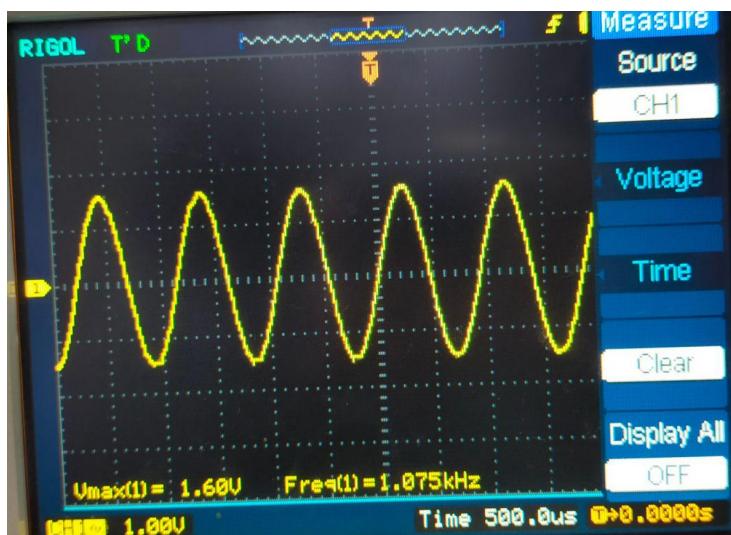
นำ output ที่ได้มามาทำ FFT ได้ความถี่ 700 Hz และ 1400 Hz

Task 2) Signal Generation and Measurement Using ESP32

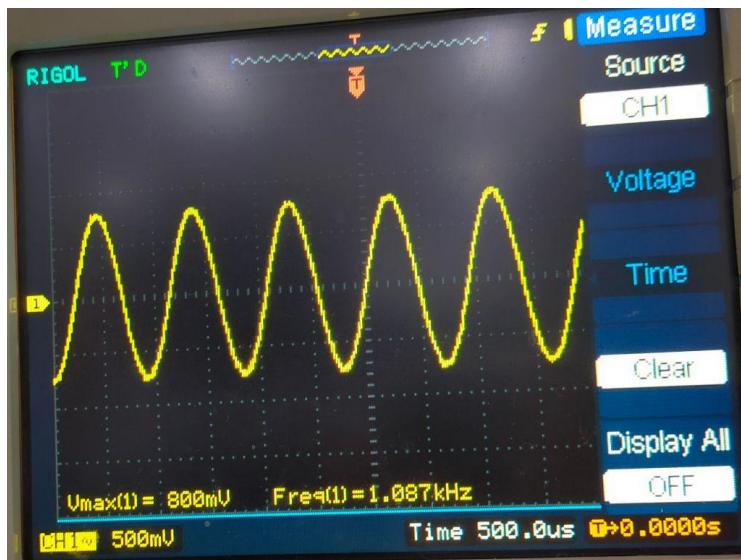
1. Utilize a digital function generator to generate a sine wave with $V_{pp}=1V$ and $V_{offset}=1.5V$ (i.e., $V_{min}=0.5V$ and $V_{max}=2.5V$) as a test signal.



4. Write an Arduino sketch for the ESP32 that utilizes the on-chip ADC to capture a sine wave signal of a specific frequency, which is created as described in Step 1) and inputted into an ADC input channel, at a fixed sampling rate.



5. Write an Arduino sketch for the ESP32 that utilizes the on-chip DAC to generate a sine wave signal as output.



7. Use an electret microphone module (with a built-in amplifier for analog output) together with the ESP32 (via ADC) to capture an audio signal.

- Write an Arduino sketch for the ESP32 that utilizes the on-chip ADC to capture the signal.

8. Use an I2S MEM microphone module with the ESP32 (via I2S) to capture an audio signal.

- Write an Arduino sketch to configure the I2S peripheral, read out the samples, send them through the serial port, and visualize the data.