UNIVERSIDAD DE GRANADA

Máster Universitario en Estructuras
Curso 2020-2021

Práctica de laboratorio: Identificación Modal Operacional de marco experimental

Módulo: MÓDULO FUNDAMENTAL: CALIDAD Y DAÑO
Materia: **Análisis Modal y Detección de Defectos**

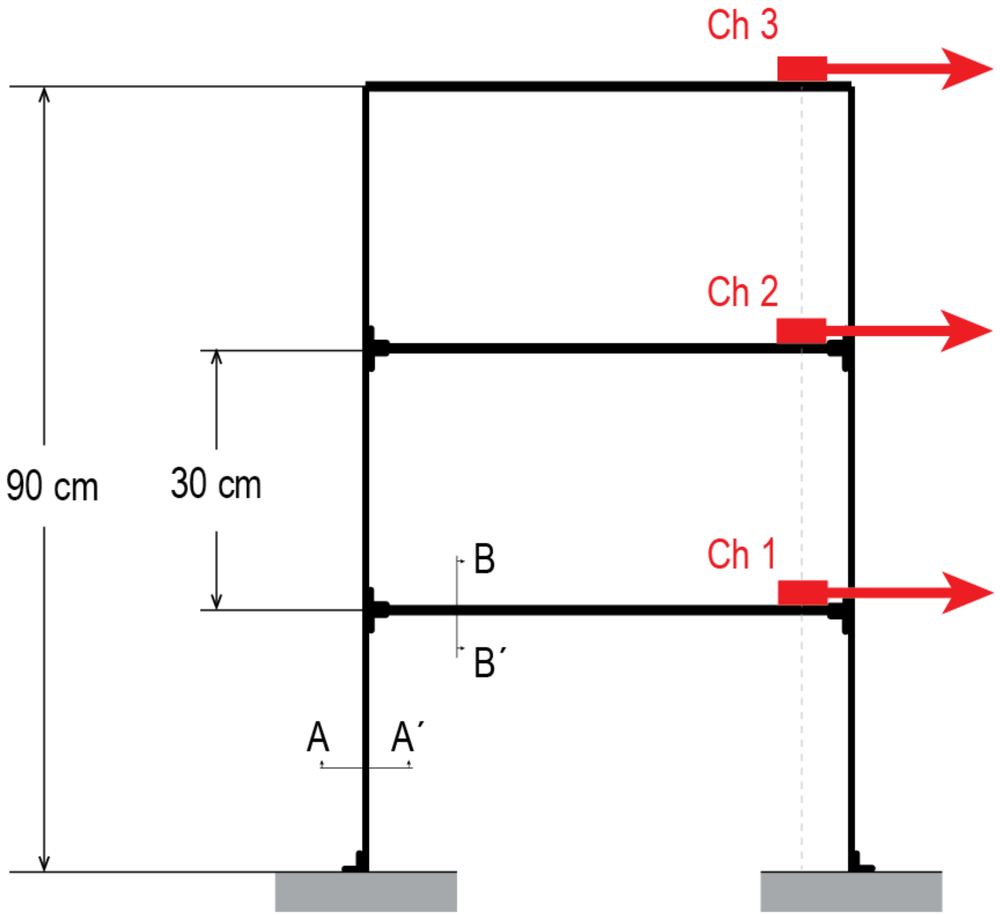
Enrique García Macías
enriquegm@ugr.es
Departamento de Mecánica de Estructuras e Ingeniería Hidráulica

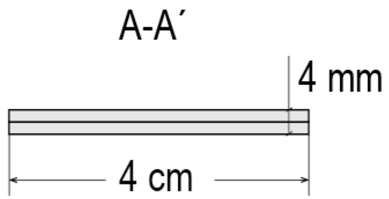
Desarrollo del curso

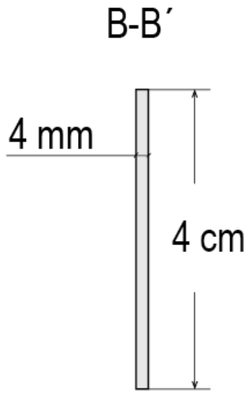
		FECHA		HORA	PROFESOR	TEMA	
Clase 1	Lunes	1	febrero	9:30-11:30	EGM	1	Introducción: Análisis modal dentro del marco del mantenimiento de la salud estructural.
Clase 2	Lunes	8	febrero	9:30-11:30	EGM	2	Fuentes de deterioro, patologías estructurales, y tecnologías de monitorización.
Clase 3	Lunes	15	febrero	9:30-11:30	EGM	3	Taller: procesamiento de señales.
Clase 4	Lunes	22	febrero	9:30-11:30	EGM	4	Análisis modal experimental.
Clase 5	Lunes	15	marzo	9:30-11:30	EGM	5	Análisis modal operacional.
Clase 6	Lunes	12	abril	9:30-11:30	EGM	6	Análisis modal operacional automatizado.
Clase 7	**Lunes**	**19**	**abril**	**9:30-11:30**	**EGM**	**7**	**Práctica de labotatorio I.** **Taller: Identificación del daño estructural.**
Clase 8	Lunes	26	abril	9:30-11:30	RCT	8	Técnicas de identificación dinámica basadas en análisis modal operacional.
Clase 9	Lunes	26	abril	12:00-14:00	RCT	9	Práctica de laboratorio II: Test de vibración ambiental.
Clase 10	Martes	27	abril	9:30-11:30	RCT	10	Casos de estudio.
Clase 11	Martes	27	abril	12:00-14:00	RCT		Presentación de trabajos.

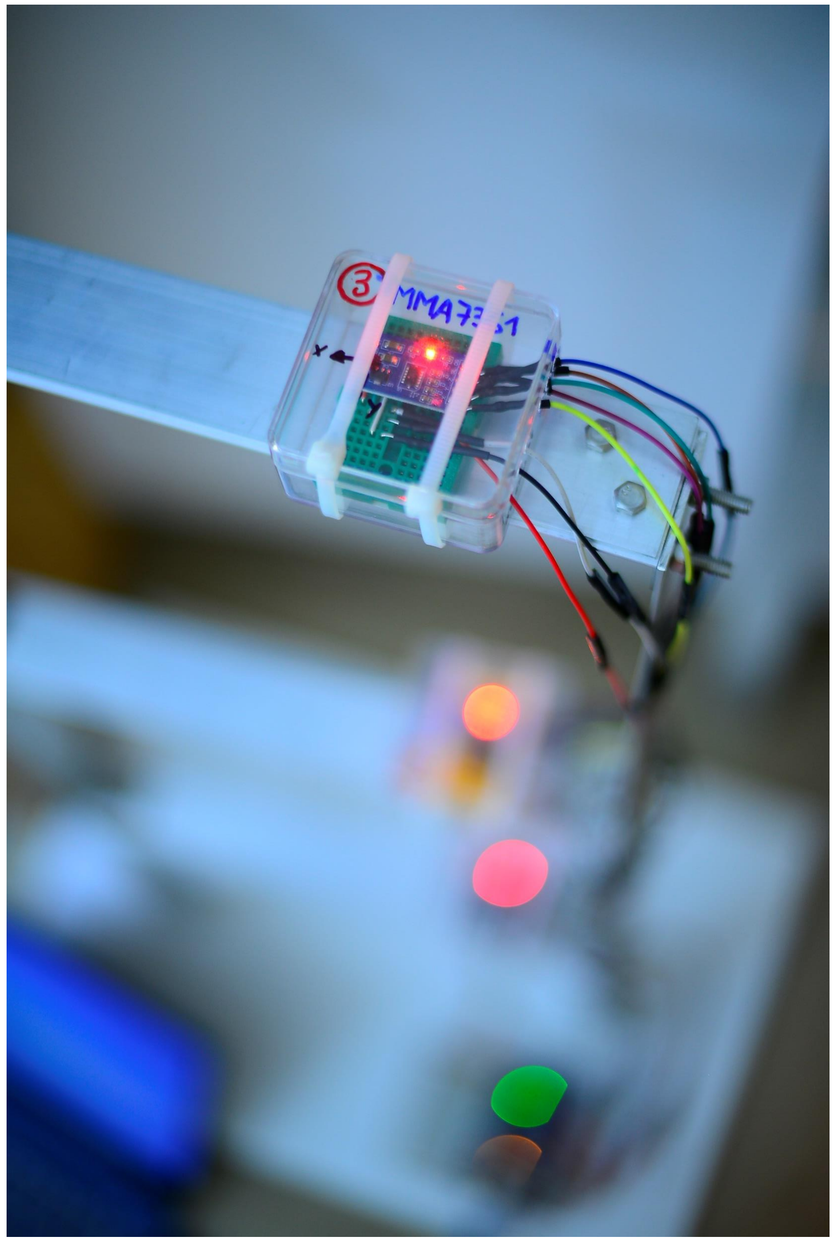
ENTREGA DE TRABAJOS Y EVALUACIÓN Del 3 al 28 de mayo

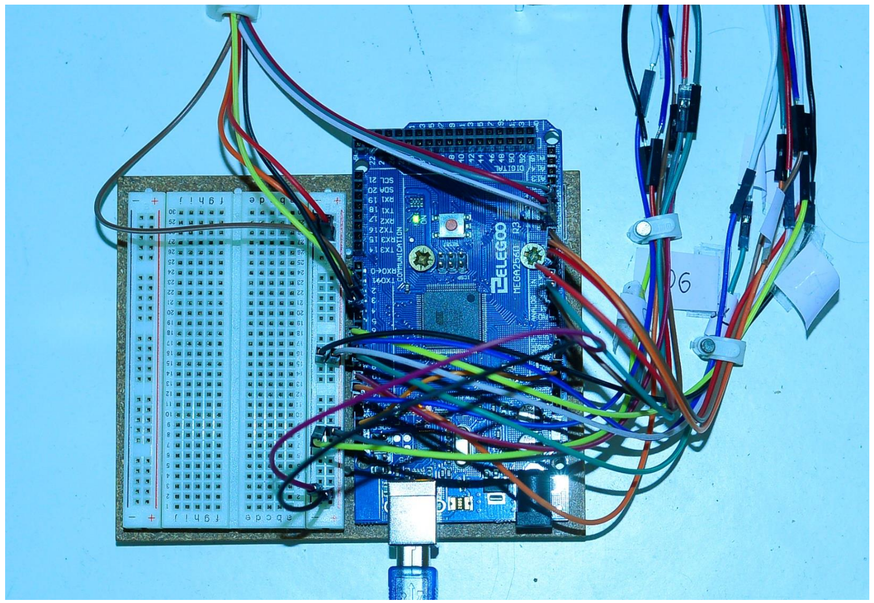
Geometría y set-up experimental

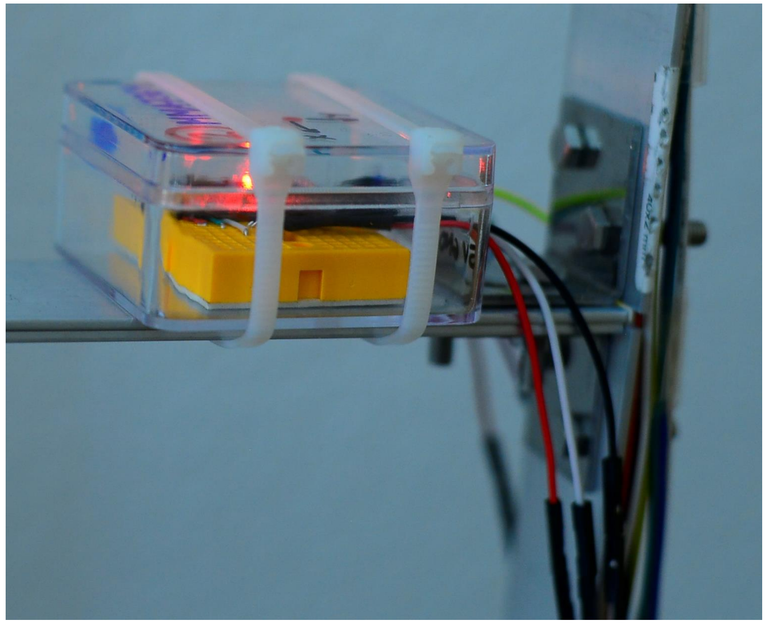






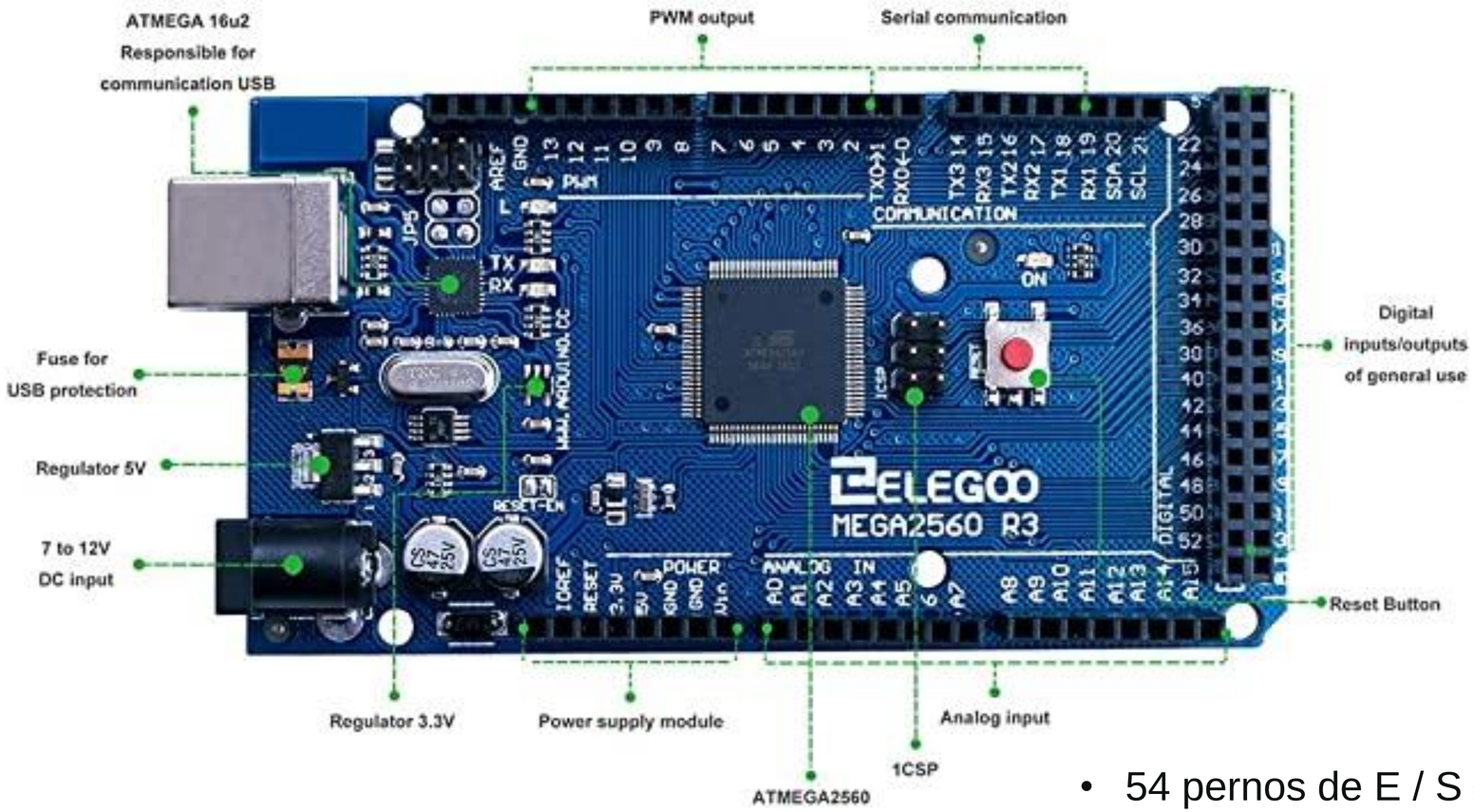






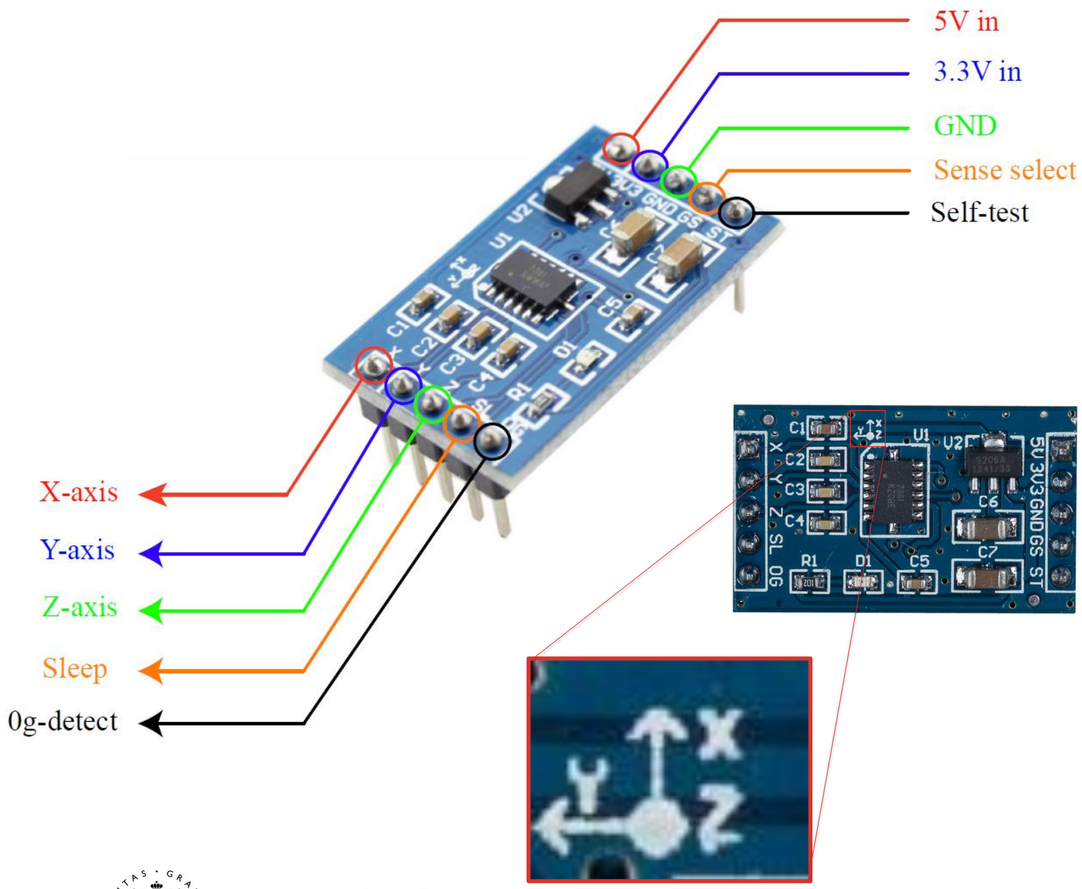
Micro-controlador

Arduino MEGA



- 54 pernos de E / S digitales y 16 entradas analógicas
- 256 KB de memoria flash para almacenar código
- 10-bits resolution

Acelerómetro Accelerometer MMA7361



Freescale Semiconductor
Technical Data

Document Number: MMA7361L
Rev 0, 04/2008

RoHS

±1.5g, ±6g Three Axis Low-g Micromachined Accelerometer

The MMA7361L is a low power, low profile capacitive micromachined accelerometer featuring signal conditioning, a 1-pole low pass filter, temperature compensation, self test, 0g-Detect which detects linear freefall, and g-Select which allows for the selection between 2 sensitivities. Zero-g offset and sensitivity are factory set and require no external devices. The MMA7361L includes a Sleep Mode that makes it ideal for handheld battery powered electronics.

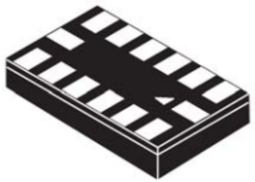
Features

- 3mm x 5mm x 1.0mm LGA-14 Package
- Low Current Consumption: 400 μA
- Sleep Mode: 3 μA
- Low Voltage Operation: 2.2 V – 3.6 V
- High Sensitivity (800 mV/g @ 1.5g)
- Selectable Sensitivity (±1.5g, ±6g)
- Fast Turn On Time (0.5 ms Enable Response Time)
- Self Test for Freefall Detect Diagnosis
- 0g-Detect for Freefall Protection
- Signal Conditioning with Low Pass Filter
- Robust Design, High Shocks Survivability
- RoHS Compliant
- Environmentally Preferred Product
- Low Cost

MMA7361L

MMA7361L: XYZ AXIS ACCELEROMETER ±1.5g, ±6g

Bottom View



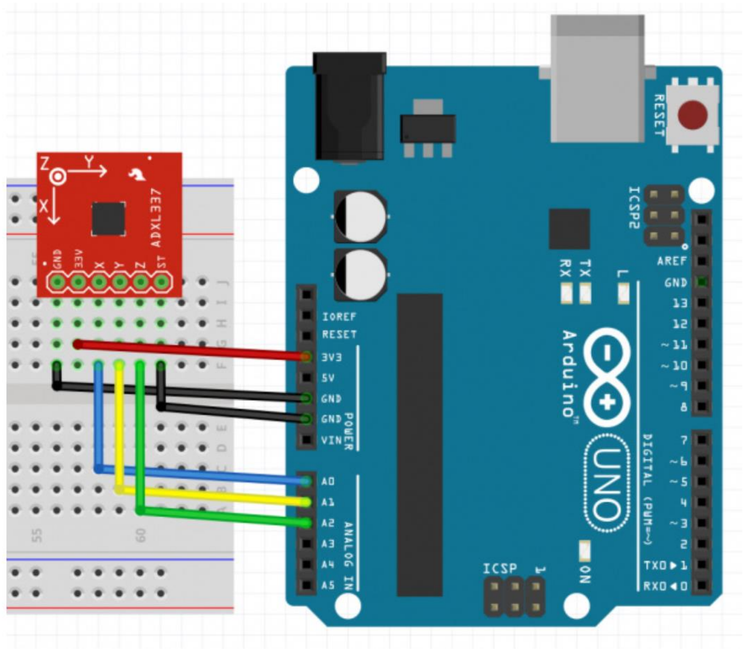
14 LEAD LGA CASE 1977-01

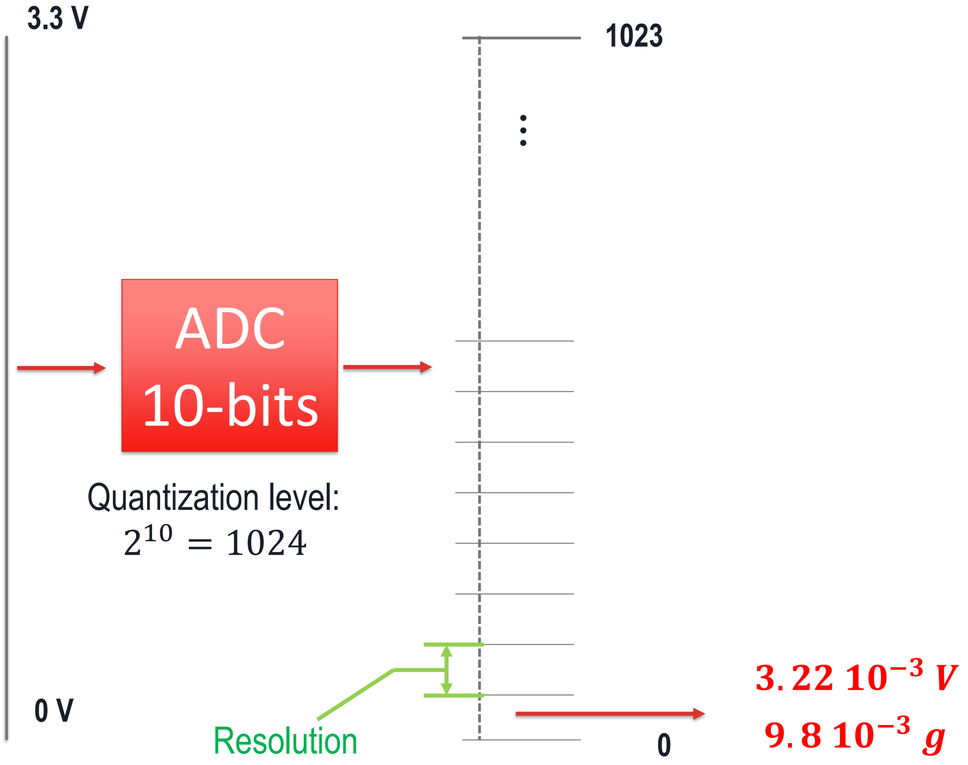
Datasheet

Acelerómetro Accelerometer MMA7361

Sensitivity
800 mV/g (3,3 V)

$$s_g = \pm 1 \cdot \frac{R}{sens_{sensor}} = \pm 4.025\ 10^{-3} \text{g}$$





$$R = \frac{V_{ref}}{N_q - 1} = \frac{3.3}{1024 - 1}\ V = 3.22\ 10^{-3}\ V$$

Acelerómetro

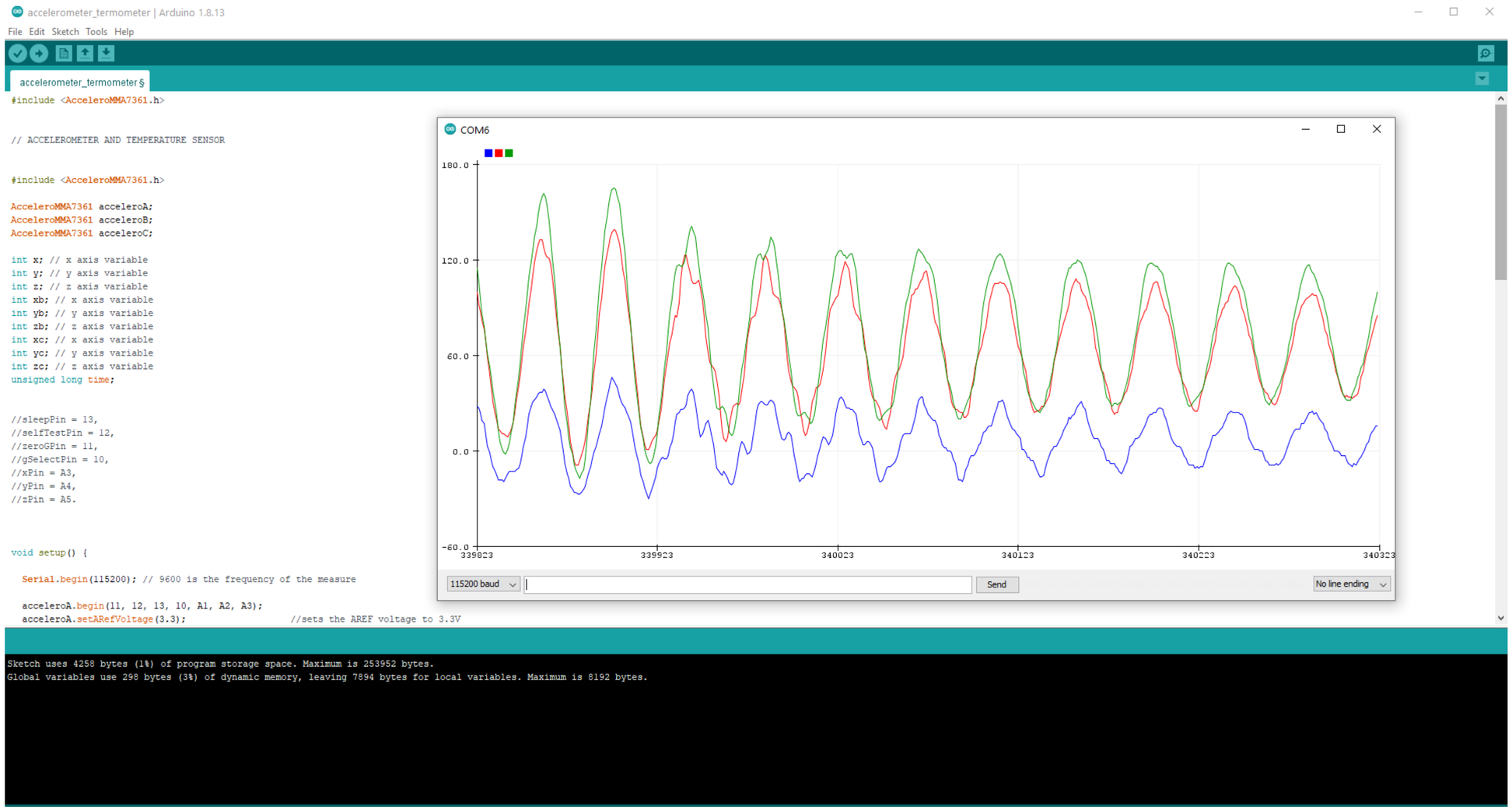
Programación acelerómetros

```
#include <AcceleroMMA7361.h>


// EXPERIMENTAL FRAME – 3
ACCELEROMETERS


#include <AcceleroMMA7361.h>

AcceleroMMA7361 acceleroA;
AcceleroMMA7361 acceleroB;
AcceleroMMA7361 acceleroC;

int x; // x axis variable
int y; // y axis variable
int z; // z axis variable
int xb; // x axis variable
int yb; // y axis variable
int zb; // z axis variable
int xc; // x axis variable
int yc; // y axis variable
int zc; // z axis variable
unsigned long time;
```

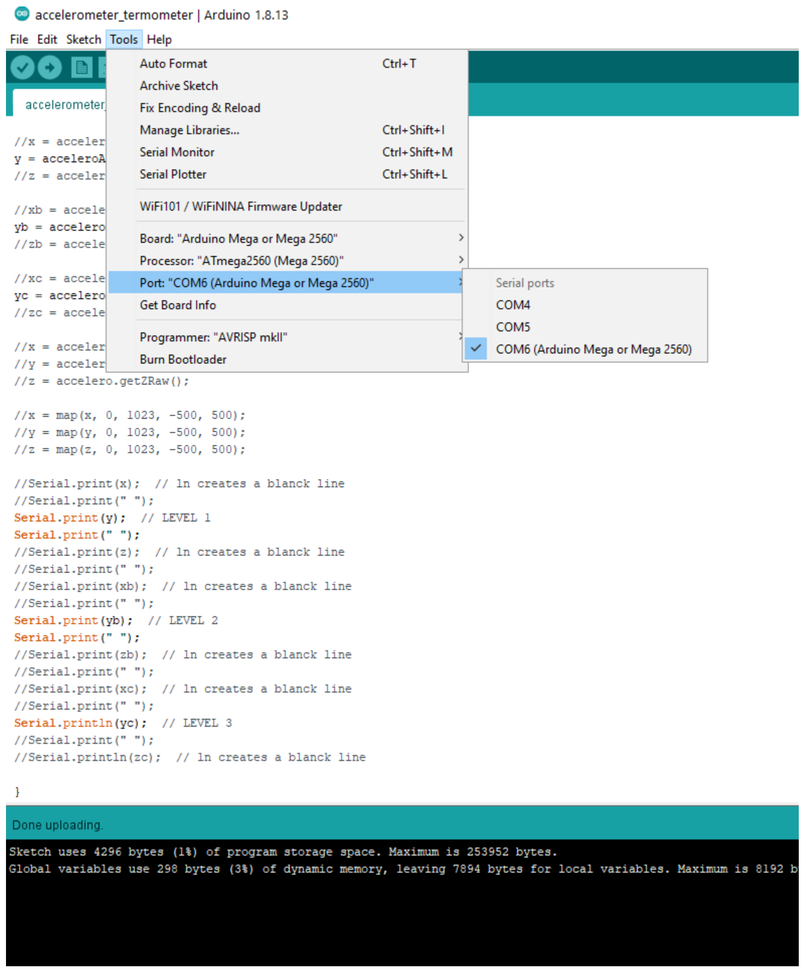
```
void setup() {

 Serial.begin(115200); // frequency of the measure

 acceleroA.begin(11, 12, 13, 10, A1, A2, A3);
 acceleroA.setARefVoltage(3.3);          //sets the AREF voltage to 3.3V
 acceleroA.setSensitivity(HIGH);         //sets the sensitivity to +/-1.5G
 analogReference(EXTERNAL);

 acceleroB.begin(7, 8, 9, 10, A5, A6, A7);
 acceleroB.setARefVoltage(3.3);          //sets the AREF voltage to 3.3V
 acceleroB.setSensitivity(HIGH);         //sets the sensitivity to +/-1.5G
 analogReference(EXTERNAL);

 acceleroC.begin(2, 3, 4, 10, A8, A9, A10);
 acceleroC.setARefVoltage(3.3);          //sets the AREF voltage to 3.3V
 acceleroC.setSensitivity(HIGH);         //sets the sensitivity to +/-1.5G
 analogReference(EXTERNAL);
}

void loop() {

time = micros();Serial.print(time);
Serial.print(" ");

y = acceleroA.getYAccel();
yb = acceleroB.getYAccel();
yc = acceleroC.getYAccel();

Serial.print(y);  //  LEVEL 1
Serial.print(" ");
Serial.print(yb);  // LEVEL 2
Serial.print(" ");
Serial.println(yc);  // LEVEL 3
}
```

Acelerómetro

Programación acelerómetros

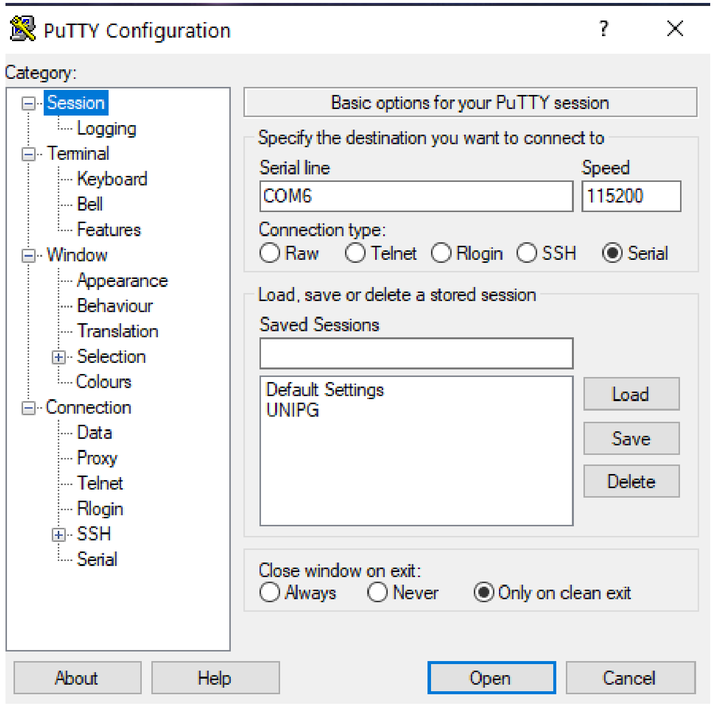


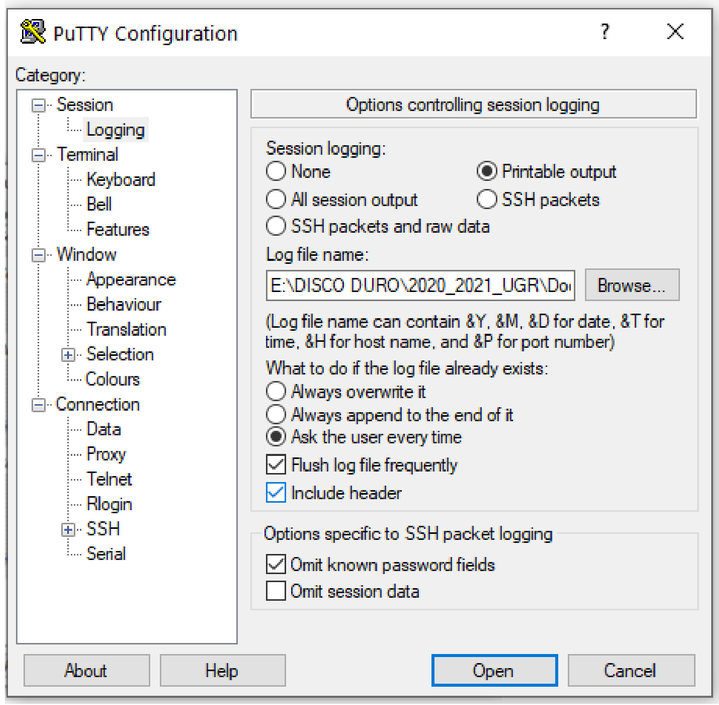
Toma de medidas



Puerto de conexión

PuTTY





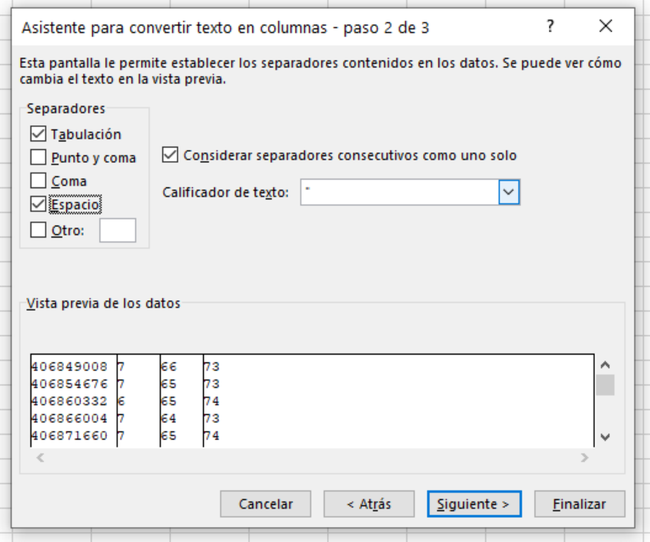
Toma de medidas

/// getXAccel(): Returns the acceleration of the X-axis as a int (1 G = 100.00)

COM6 - PuTTY

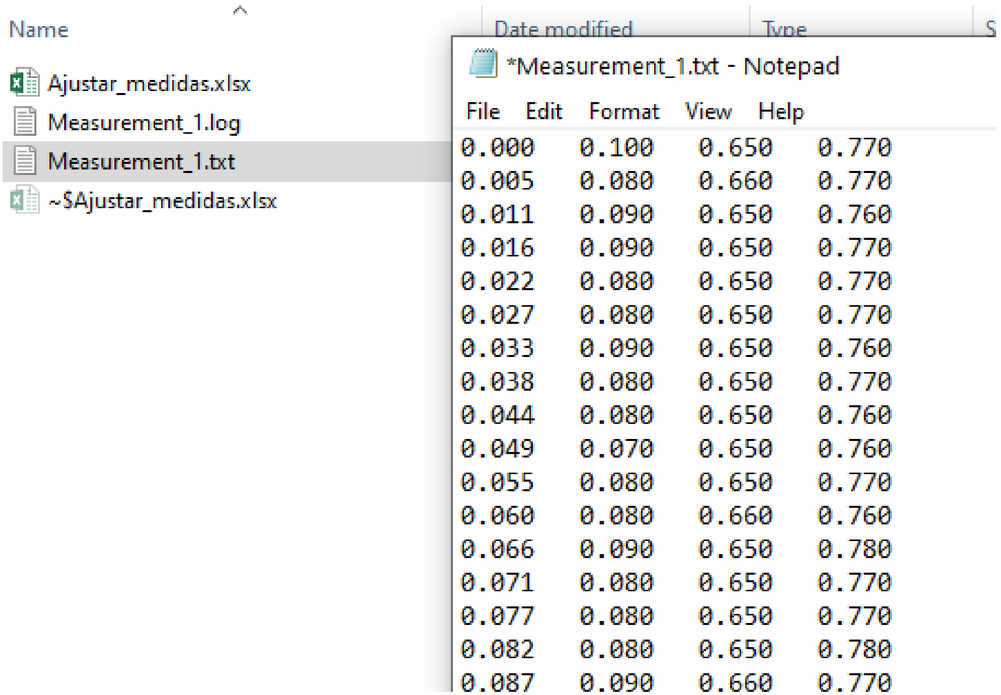
```
6732448 -4 41 46
6738064 -1 45 52
6743664 -1 50 58
6749268 3 55 66
6754852 4 59 73
6760420 6 66 80
6766004 8 71 87
6771572 10 77 93
6777212 12 82 99
6782828 16 85 104
6788512 19 90 110
6794184 21 93 113
6799872 23 95 115
6805568 24 98 118
6811248 26 100 118
6816988 25 102 117
6822700 25 101 117
6828420 24 99 116
6834080 22 95 115
6839768 21 91 112
6845464 18 88 106
6851144 16 84 100
6856832 14 81 94
6862460
```

			Delta t [s]	Time	Acceleration level 1	Acceleration level 2	Acceleration level 3
			#¡VALOR!	[s]	[g]	[g]	[g]
406849008	4	64	74	0.000	0.040	0.640	0.740
406854676	7 65 73			#¡VALOR!	0.000	0.000	0.000
406860332	6 65 74			#¡VALOR!	0.000	0.000	0.000
406866004	7 64 73			#¡VALOR!	0.000	0.000	0.000
406871660	7 65 74			#¡VALOR!	0.000	0.000	0.000
406877332	7 65 73			#¡VALOR!	0.000	0.000	0.000
406882988	7 6849008 7 66 73			#¡VALOR!	0.000	0.000	0.000
406854676	7 65 73			#¡VALOR!	0.000	0.000	0.000
406860332	6 65 74			#¡VALOR!	0.000	0.000	0.000
406866004	7 64 73			#¡VALOR!	0.000	0.000	0.000
406871660	7 65 74			#¡VALOR!	0.000	0.000	0.000
406877332	7 65 73			#¡VALOR!	0.000	0.000	0.000
406882988	7 6204 42 65 77			#¡VALOR!	0.000	0.000	0.000
5684 10 65	77			#¡VALOR!	0.000	0.000	0.000
11132 8 66	77			#¡VALOR!	0.000	0.000	0.000
16612 9 65	76			#¡VALOR!	0.000	0.000	0.000
22080 9 65	77			#¡VALOR!	0.000	0.000	0.000
27548 8 65	77			#¡VALOR!	0.000	0.000	0.000
33028 8 65	77			#¡VALOR!	0.000	0.000	0.000
38504 9 65	76			#¡VALOR!	0.000	0.000	0.000
43980 8 65	77			#¡VALOR!	0.000	0.000	0.000
49460 8 65	76			#¡VALOR!	0.000	0.000	0.000
54928 7 65	76			#¡VALOR!	0.000	0.000	0.000
60388 8 65	77			#¡VALOR!	0.000	0.000	0.000
65868 8 66	76			#¡VALOR!	0.000	0.000	0.000

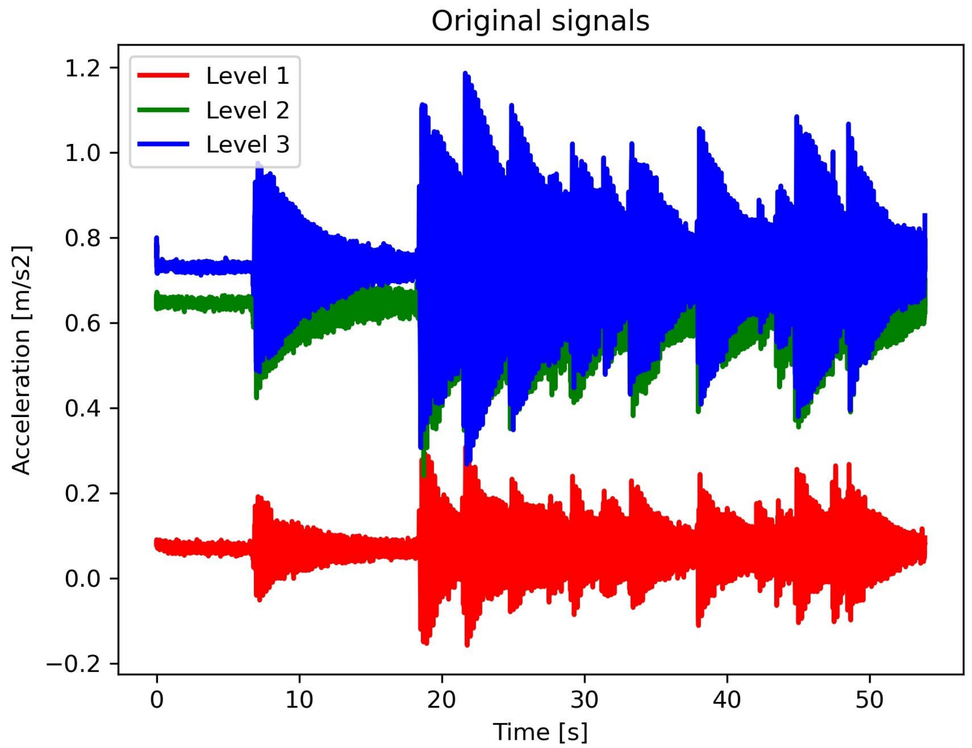


Ajustar_medidas.xlsx

Toma de medidas

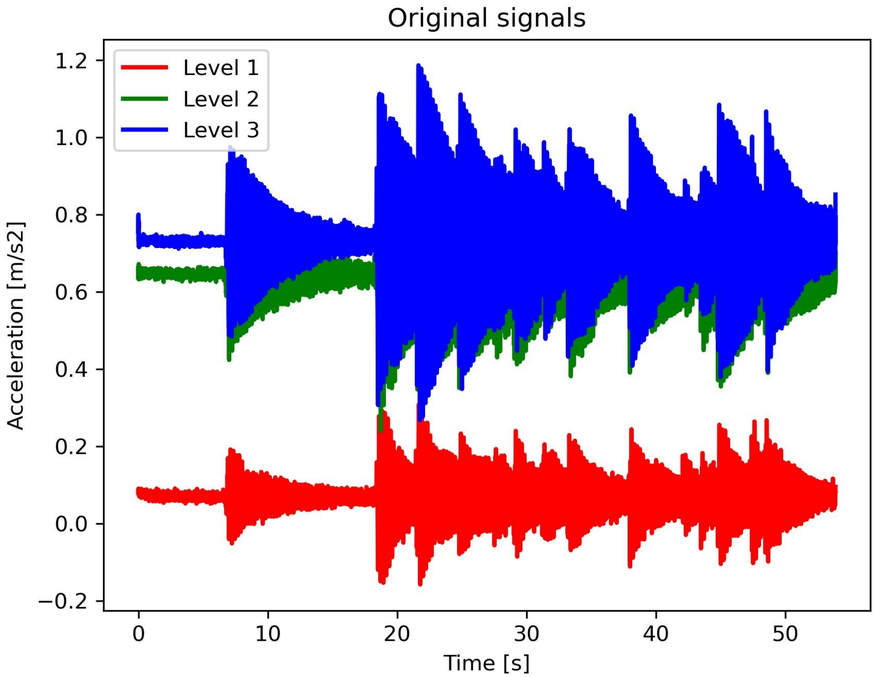


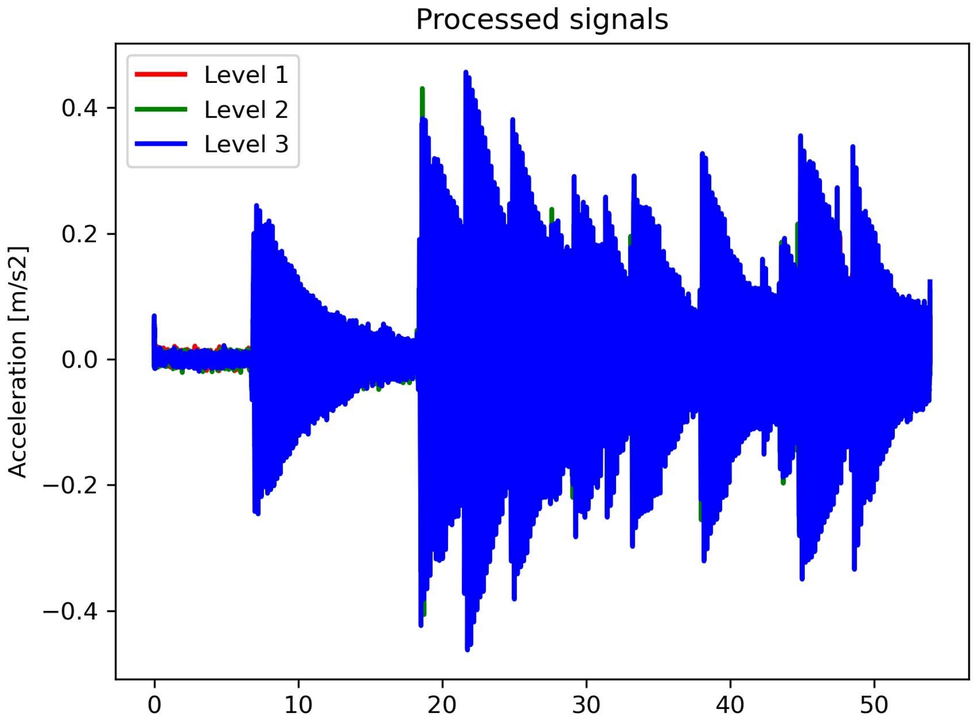
Generate_measurements.py

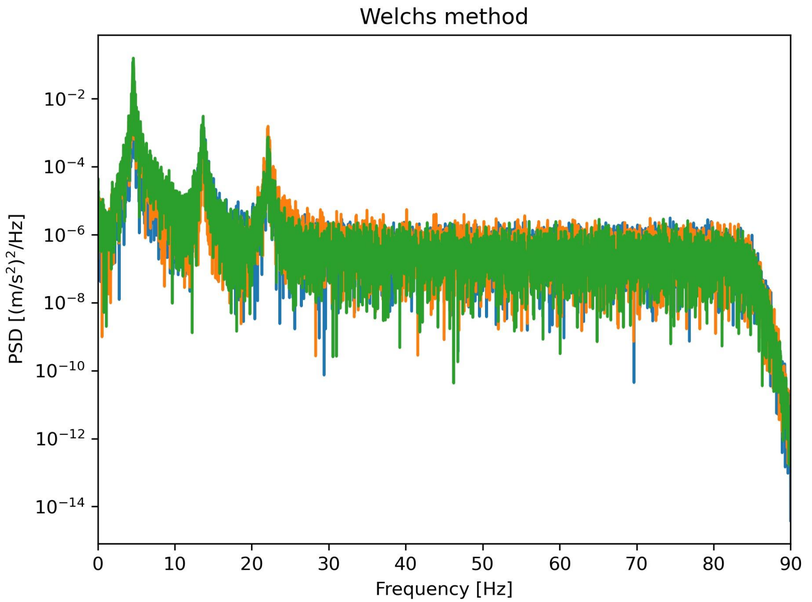


Toma de medidas

Generate_measurements.py

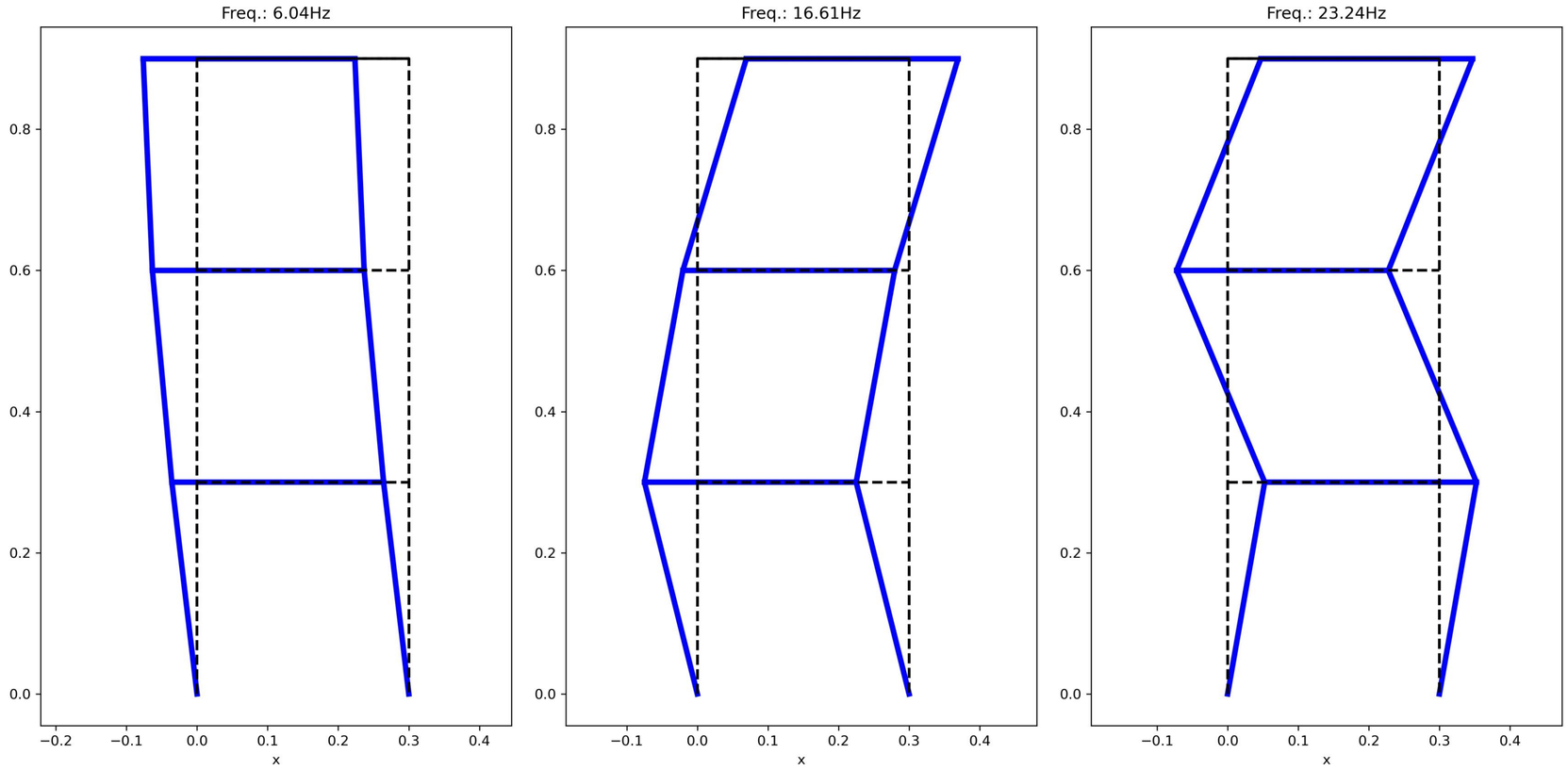






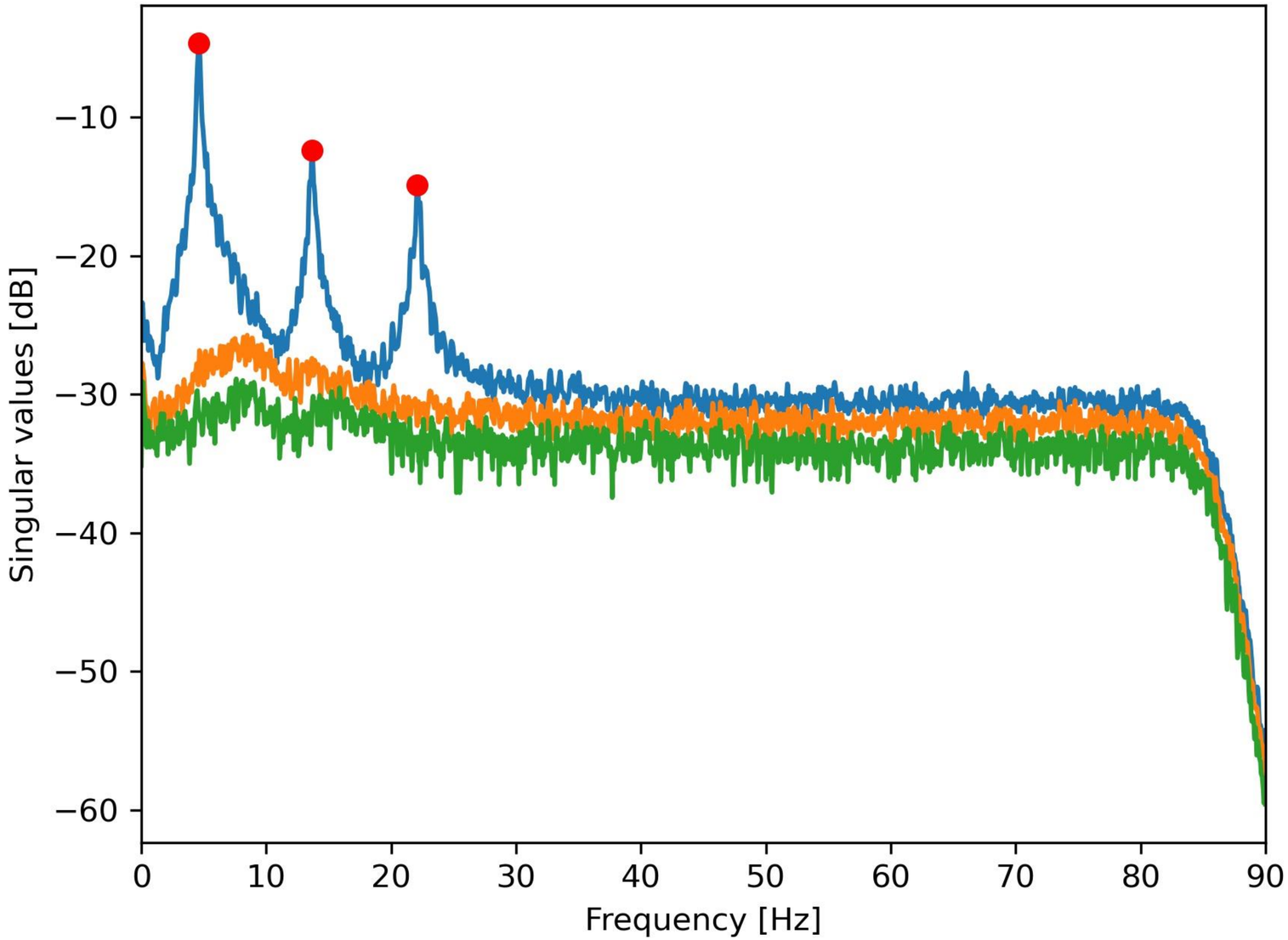
Identificación Modal

```
Eyo = 69*1E+9;  # N/m2
dens = 2700;  # kg/m3
Ar = 0.04*0.002;
Ine = (1/12)*0.04*0.002**3; # m^4
h = 0.3;  # m
Le = 0.3;  # m

k1 = 2*12*Eyo*Ine/h**3;
k2 = k1;
k3 = k1;
m1 = 2*h*Ar*dens+2*Ar*Le*dens;
m2 = m1;
m3 = h*Ar*dens+2*Ar*Le*dens;

M = np.array([[m1,0,0],[0,m2,0],[0,0,m3]])
K = np.array([[k1+k2,-k2,0],[-k2,k2+k3,-k3],[0,-k3,k3]])
```

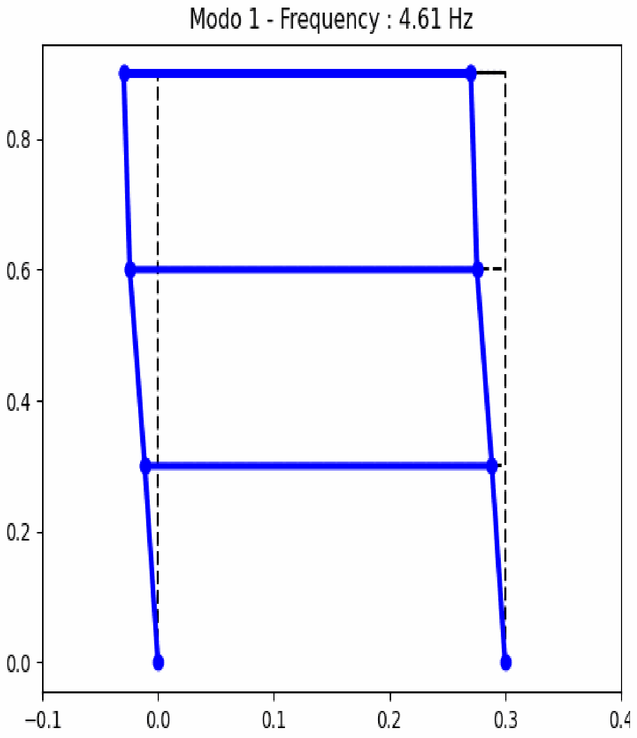


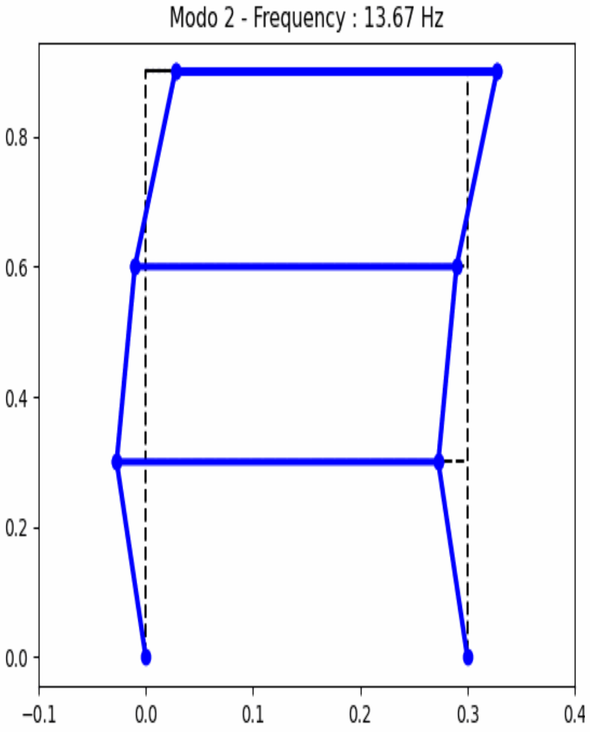
FDD

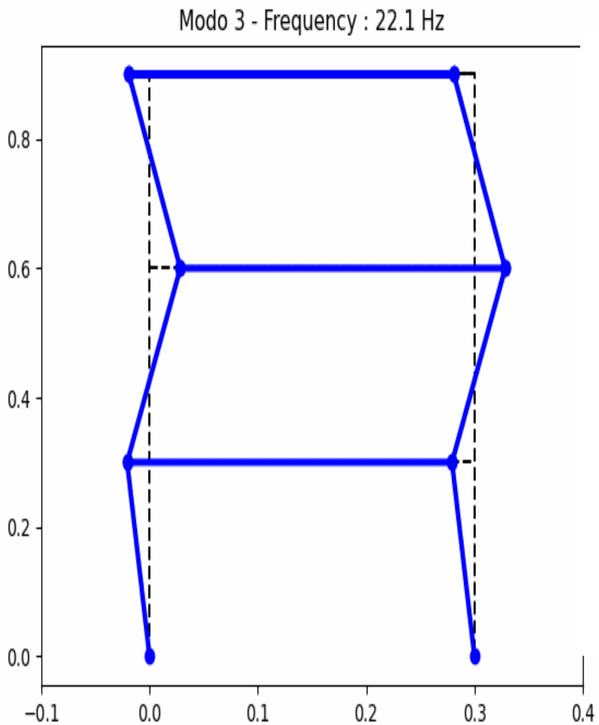
OMA_FDD.py

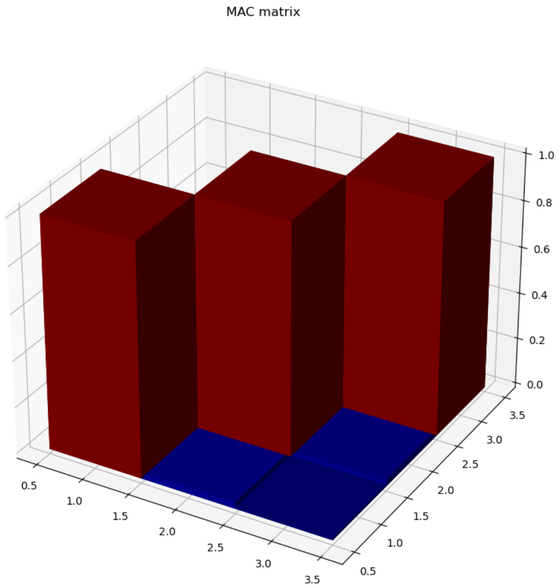


FDD


Modo 1 - Frequency : 4.61 Hz
0.8
0.6
0.4
0.2
0.0
−0.1
0.0
0.1
0.2
0.3
0.4

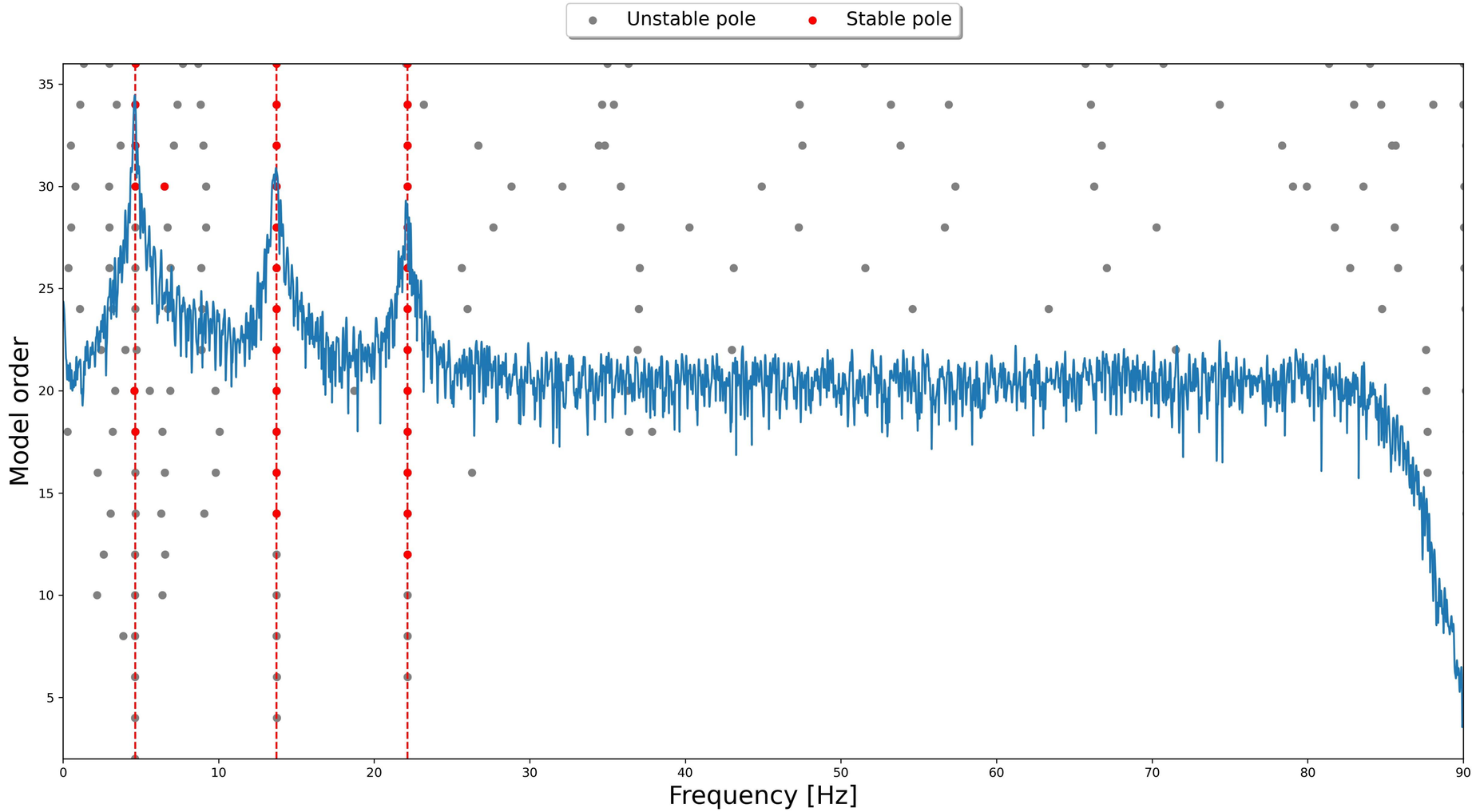

Modo 2 - Frequency : 13.67 Hz
0.8
0.6
0.4
0.2
0.0
−0.1
0.0
0.1
0.2
0.3
0.4


Modo 3 - Frequency : 22.1 Hz
0.8
0.6
0.4
0.2
0.0
−0.1
0.0
0.1
0.2
0.3
0.4

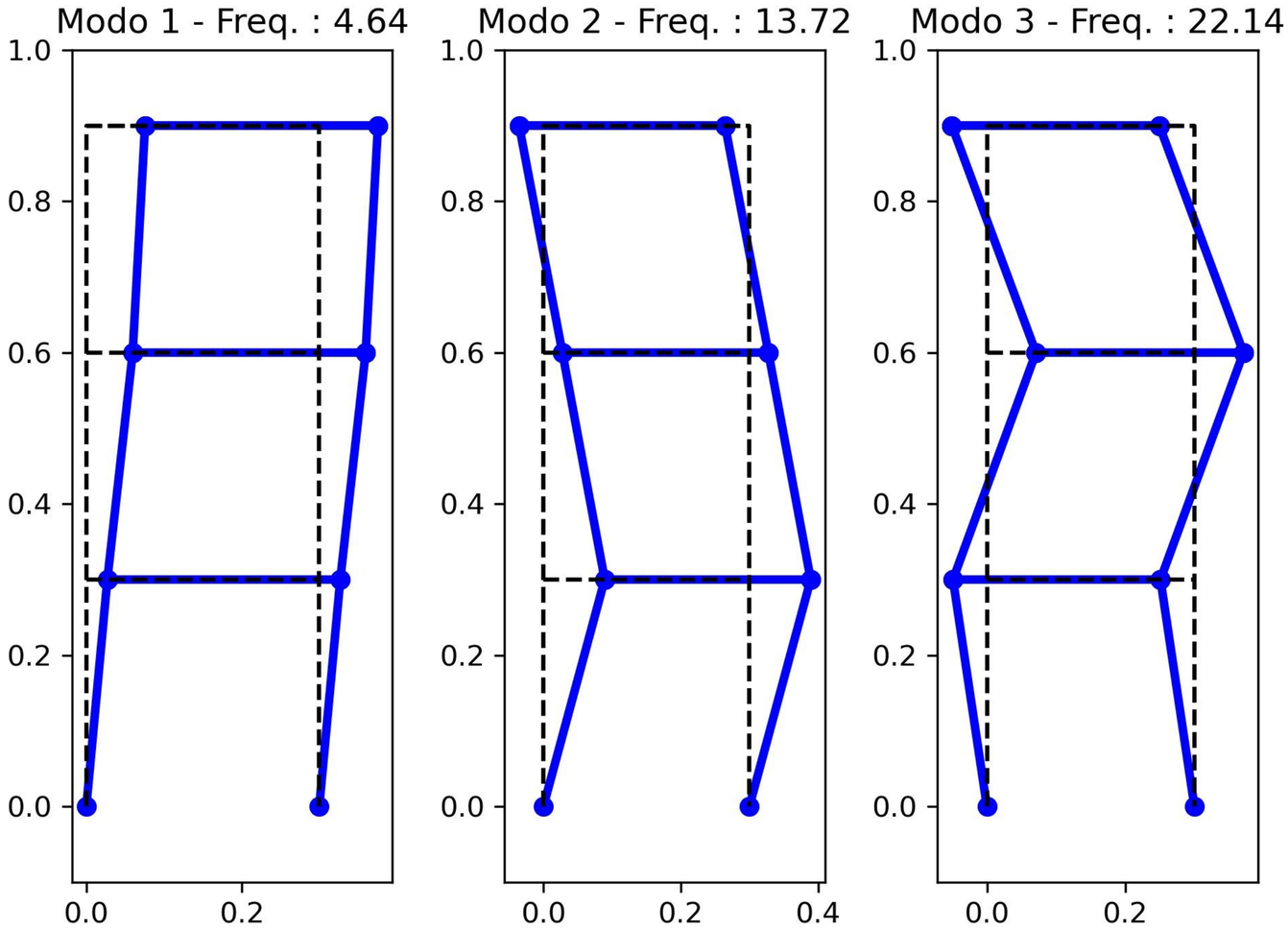

MAC matrix
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

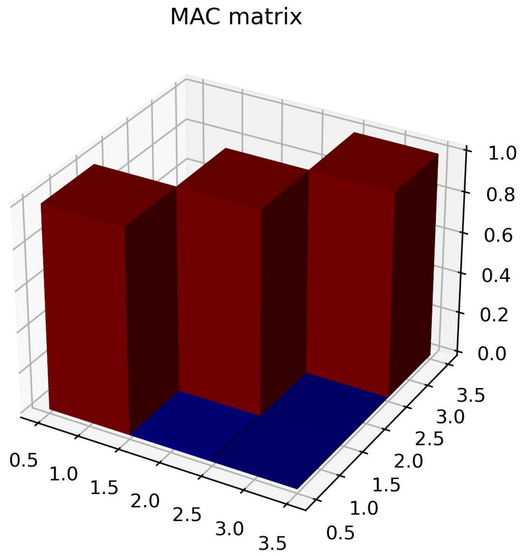
COV-SSI

OMA_COV_SSI.py



COV-SSI


Modo 1 - Freq. : 4.64
Modo 2 - Freq. : 13.72
Modo 3 - Freq. : 22.14
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4


MAC matrix
1.0
0.8
0.6
0.4
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

COMPARISON RESULTS

Mode N.	FDD		COV-SSI	
	f_i [Hz]	ξ_i [%]	f_i [Hz]	ξ_i [%]
1				
2				
3				

COV-SSI