Proyecto Fin de Grado
Grado en Ingeniería Aeroespacial

Puesta a punto de un sistema inalámbrico de bajo coste para análisis vibratorios: MPU6050 y módulo ESP8266 .

Autor: Víctor Javier Blanco González

Tutor: Pedro Galvín Barrera

Dpto. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Grado
Grado en Ingeniería Aeroespacial

Puesta a punto de un sistema inalámbrico de bajo coste para análisis vibratorios: MPU6050 y módulo ESP8266 .

Autor:
Víctor Javier Blanco González

Tutor:
Pedro Galvín Barrera
Catedrático

Dpto. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Grado: Puesta a punto de un sistema inalámbrico de bajo coste para análisis vibratorios: MPU6050 y módulo ESP8266 .

Autor: Víctor Javier Blanco González
Tutor: Pedro Galvín Barrera

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

A mi familia, amigos y compañeros.

Al profesorado.

Víctor Javier Blanco González.
Sevilla, 2021.

Resumen

Este proyecto fin de grado se centrará en la puesta a punto de un sistema de bajo coste para la elaboración de análisis vibratorios, realizando la adquisición de datos de manera inalámbrica. Para llevar a cabo dicho objetivo se ha contado con acelerómetros MPU6050 y módulos ESP8266.

En el presente documento, se realizará un seguimiento cronológico por las diferentes configuraciones que han sido probadas en lo que respecta a la electrónica, hasta alcanzar la óptima teniendo como objetivo la frecuencia de adquisición de datos, principal característica a la hora de realizar un análisis vibratorio.

Finalmente, se comprobará el funcionamiento del sistema mediante la comparación con equipos más sofisticado con los que cuenta el departamento de Mecánica de Medios Continuos y Teoría de Estructuras en los laboratorios de la escuela.

Abstract

This final degree project will focus on the tuning of a low-cost system for the development of vibratory analysis, acquiring data wirelessly. MPU6050 accelerometers and ESP8266 modules have been used to carry out this goal.

In this document, a chronological follow-up will be accomplished through the different configurations that have been tested, concerning electronics, until the optimum is reached, aiming at the frequency of data acquisition, which is the main characteristic when a vibratory analysis is executed.

Finally, the operation of the system will be checked by comparing it with more sophisticated equipment that the Department of Mecánica de Medios Continuos y Teoría de Estructuras has in the school laboratories.

Índice Abreviado

Índice

1 Introducción y objetivos

Este proyecto se centrará en la puesta a punto de un sistema inalámbrico para análisis vibratorios. Con este objetivo se analizará cual es el estado actual de la metodología, sistemas empleados, recurriendo al sector industrial donde los análisis vibratorios en la maquinaria juegan un papel esenciales. Finalmente, se expondrán los objetivos del proyecto y los medios a analizar para su cumplimiento.

1.1 Estado del arte

El Análisis de Vibración es una técnica empleada para identificar y predecir anomalías mecánicas en en una determinada estructura, midiendo la vibración e identificando las frecuencias involucradas. Esta vibración es registrada por un acelerómetro y los datos son procesados por un analizador de espectro. La aplicación de esta técnica en el mantenimiento predictivo mejora en gran medida la eficiencia y la fiabilidad.

Los equipos encargados de realizar dichos análisis vibratorios son instrumentos utilizados para medir, almacenar y diagnosticar la vibración producida por una determinada excitación. Estos equipos emplean herramientas basadas en el FFT.

En los últimos 20 años se han mejorado los procesos de mantenimiento predictivo y las herramientas utilizadas para realizarlo. En particular, la tecnología del análisis de vibraciones ha evolucionado a niveles inimaginables.

1.1.1 Estado actual de la metodología

Una de las últimas tendencias en la tecnología empleada para la realización de análisis vibratorios es la utilización de sistemas de Monitoreo de Vibraciones con transmisión inalámbrica, remplazando estos a los acelerómetros por cable convencionales. Dichos sistemas cuentan con un conjunto de elementos involucrados en la adquisición y análisis de uno o más parámetros, con el fin de identificar cambios en el comportamiento de la maquinaria. Monitorear estos parámetros ayuda a identificar futuras fallas como pueden ser: desbalance, fallas en engranes, holgura y muchas otras.

Figura 1.1 Monitoreo de Vibraciones con transmisión inalámbrica.

Entre los diferentes sistemas inalámbricos existentes en el mercado, se analizará el sistema principal con el que cuenta la empresa neoyorquina ERBESSD INSTRUMENTS [1]. Este es el sistema de monitoreo continuo de vibraciones denominado Phantom que integra otros parámetros como temperatura, corriente, RPM y velocidad en un solo sistema de diagnóstico. Los sensores empleados cuentan con una batería de alta autonomía pudiendo tomar hasta 100.000 muestras y además son de muy fácil instalación. Este sistema Phantom es capaz de enviar los datos a una base de datos local o al sistema de la empresa basado en la nube EIAnalytic. De este modo, se es capaz de realizar un seguimiento en todo momento de la maquinaria desde cualquier dispositivo: smartphone, ordenador o tablet. Esto último es posible gracias a que los sensores inalámbricos incluyen el protocolo de comunicación universal para aplicaciones industriales: Modbus TCP/IP.

Los Sensores envían gráficas de forma de onda y espectro a través del protocolo inalámbrico BLE 5.0 (Bluetooth Low Energy), alcanzado un ancho de banda de 10KHz. Además, consumen muy poca energía y alcanzan distancias de hasta 100m.

Figura 1.2 Protocolo BLE entre sensores y módulo.

Como ya se ha comentado, los acelerómetros empleados son sensores de Vibración por Bluetooth y a continuación se muestran algunos modelos:

- EPH-V11: El Acelerómetro Bluetooth Phantom Expert es un sensor de vibración inalámbrico diseñado para ser montado de manera permanente. El Phantom Expert graba FFT de vibración en tres ejes simultáneamente: X, Y y Z. Adicionalmente, incluye una batería que permite tomar hasta 100.000 muestras y puede ser remplazada al final de su vida útil.
- Phantom ATEX: es un Sensor de Vibración Triaxial capaz de enviar 3 señales de FFT. Este sensor de vibración también es capaz de activarse al detectar altos niveles de vibración con el fin de enviar datos de vibración y alarma.

Figura 1.3 Acelerómetros inalámbricos EPH-V11 y Phantom ATEX respectivamente.

En cuanto a la instalación, presentan ciertas ventajas como son la comodidad y la accesibilidad. En ciertas circunstancias es bastante complicado colocar los sensores y llevar los cables hasta el centro de control más próximo o puede darse el caso en el que, para alguna aplicación, no sea posible colocar el sensor con cable. Con lo cual, esta configuración permite instalaciones sencillas, rápidas y rentables.

Figura 1.4 Ventajas en la instalación.

1.1.2 Conclusiones de la metodología

Como se ha podido apreciar, la monitorización inalámbrica es una tecnología puntera que permite conocer a tiempo real cual es el estado de un determinado cuerpo. Además, al ser la transmisión sin cable, la instalación se simplifica bastante y permite el seguimiento desde cualquier dispositivo.

1.2 Objetivos y medios de cumplimiento

La tecnología anteriormente comentada ha sido analizada con la finalidad de emplearla para la monitorización de puentes ferroviarios. El objetivo a largo plazo es lograr una monitorización autosuficiente, es decir, un sistema inalámbrico de bajo consumo cuya alimentación eléctrica sea extraída de la propia vibración de la estructura, transformando la energía mecánica propia de la vibración en energía eléctrica. No obstante, en este trabajo fin de grado únicamente se realizarán los primeros estudios asociados al sistema de monitoreo inalámbrico.

Con los objetivos anteriores se analizarán los diferentes medios de cumplimiento para poner en funcionamiento un sistema de bajo coste formado por dos módulos ESP8266 y acelerómetros MPU6050. Uno de los principales requisitos, como ya se conoce, será que la adquisición de datos se haga de manera inalámbrica. Además, la frecuencia de adquisición de datos ha de cumplir una cierta especificación teniéndose en cuenta las frecuencias típicas para la aplicación en cuestión. Lograr unos 200 valores por segundo asegura tener un ancho de banda suficiente para contener las frecuencias a las que estará expuesta la estructura.

Para cumplir con dicha especificación de adquisición ha sido necesario estudiar diferentes configuraciones:

- Conexión del módulo ESP8266 a la red local y creación de una página web con lenguaje de programación HTML donde se vuelcan los datos medidos por el acelerómetro. Para esta primera configuración solo es necesario contar con un módulo ESP8266.
- Conexión de dos módulos ESP8266, uno como server (access point) y el otro como client (station).

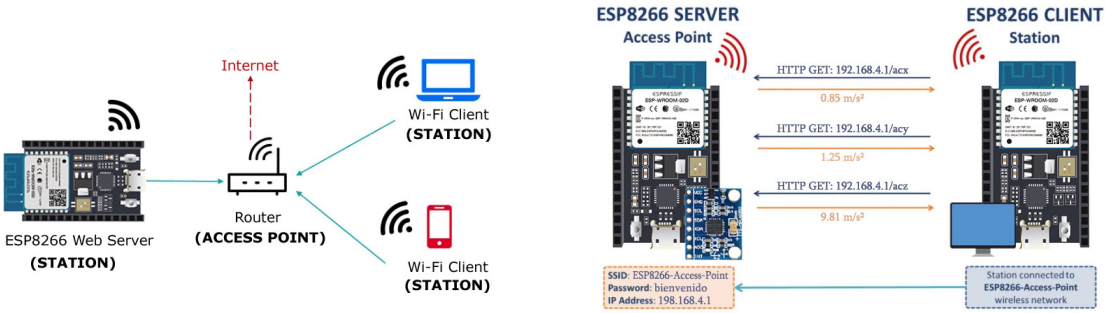


Figura 1.5 Esquemas de las configuraciones 1 y 2 respectivamente.

Para las dos configuraciones anteriores se han probado diferentes métodos de adquisición de datos con el objetivo de aumentar la frecuencia en dicha adquisición. Para la primera configuración se han realizado los siguientes métodos:

- Extracción de los valores volcados en la página web con Excel (web scraping).
- Extracción de los valores de la página web directamente con MATLAB.

Con la segunda configuración:

- Extracción de los valores por el puerto serie directamente desde MATLAB.
- Extracción de los valores con un software proporcionado por Roger Meier [2] que captura los datos del puerto serie en un archivo .txt, haciendo uso del lenguaje de programación Python.

Una vez se determine la configuración que mejores resultados presente, cumpliendo con los requisitos, se hará una comparativa con equipos profesionales en el sector. Finalmente, se expondrán las conclusiones y las líneas de trabajos futuros para alcanzar los objetivos iniciales.

2 Descripción de la electrónica y su funcionamiento

Este capítulo se centrará tanto en la descripción detallada de la electrónica usada: MPU6050 y ESP8266, como en la codificación de la misma en el IDE de Arduino.

2.1 Acelerómetro MPU6050

El MPU-6050 es una unidad de medidas inerciales (IMU) de seis grados de libertad, que combina un acelerómetro de tres ejes y un giroscopio de tres ejes. Este es uno de los IMUs más empleados por su gran calidad y precio. Normalmente, se encuentra integrado en módulos como el GY-521 que incorporan la electrónica necesaria para el correcto uso de la IMU y facilita el acceso a sus pines. Aunque la tensión de alimentación del PMU6050 es de bajo voltaje entre 2.4 y 3.6V, este módulo incluye un regulador de voltaje que permite alimentar directamente a 5V [6].

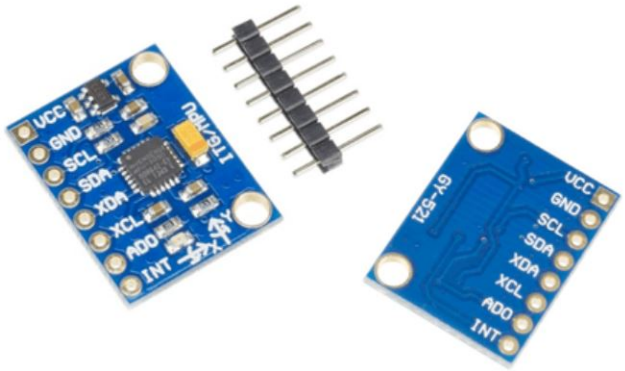


Figura 2.1 Imágene del módulo GY-521 que integra el MPU6050.

El MPU-6050 incorpora un procesador interno DMP (Digital Motion Processor) que ejecuta complejos algortimos de MotionFusion para combinar las mediciones de los sensores internos, evitando tener que realizar los filtros de forma exterior. Además, consta de convertidores anológico-digital, tres para los ejes del acelerómetro y otros tres para los del giroscopio. Como se verá más adelante, es posible ajustar la precisión de estos variando los rangos de salidas.
La transmisión de datos entre el módulo ESP8266 y la IMU se realizará a través del sistema de comunicación I2C con una frecuencia de 400 kHz [6].

2.1.1 Características del acelerómetro

Tabla 2.1 Características del acelerómetro [6].

Características	Medida
Rangos de salida ajustables	$\pm 2g, \pm 4g, \pm 8g \;\; y \;\; \pm 16g$
3 convertidores Analógico/Digital	16 bits
Intensidad nominal	500 μA

Con el objetivo de obtener la mayor sensibilidad, se ha configurado con el rango $\pm 2g$, más que suficiente para la labor que va a desempeñar.

2.1.2 Características del giroscopio

Tabla 2.2 Características del giroscopio [6].

Características	Medida
Rangos de salida ajustables	$\pm 250, \pm 500, \pm 1000 \;\; y \;\; \pm 2000^o/s$
3 convertidores Analógico/Digital	16 bits
Intensidad nominal	3.6 mA

No se hará uso del giroscopio en nuestro proyecto, con lo cual, se dejará la configuración que viene por defecto.

2.1.3 Protocolo de comunicación I2C

El estándar I2C (Inter-Integrated Circuit) fue desarrollado por Philips en 1982 para la comunicación interna de dispositivos electrónicos en sus artículos. Posteriormente, fue adoptado progresivamente por otros fabricantes hasta convertirse en un estándar del mercado [4].

Una de las principales características del bus I2C es que requiere únicamente dos cables para su funcionamiento, uno para la señal de reloj (CLK: Serial CLocK) y otro para el envío de datos (SDA: Serial DAta) [4].

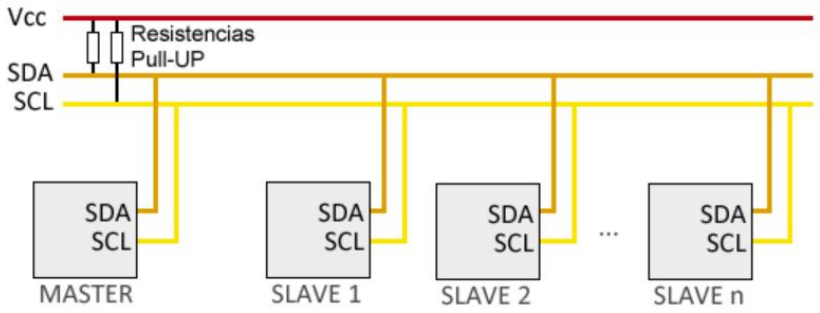


Figura 2.2 Esquema del bus I2C.

En el bus cada dispositivo dispone de una dirección, con el objetivo de acceder a cada uno de estos de manera individual. En general, cada dispositivo debe tener asignada una única dirección. Se tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro inicia la comunicación con los esclavos y puede mandar o recibir datos de estos.
El bus es síncrono: el maestro proporciona una señal de reloj que mantiene sincronizados a todos los dispositivos del bus. De este modo, se elimina la necesidad de que cada dispositivo contenga su propio reloj, tenga que acordar una velocidad de transmisión y mecanismos para mantener esta sincronizada [4].

Para poder realizar la comunicación con solo un cable de datos, el bus I2C emplea una trama amplia. La comunicación consta de:

- 7 bits para la dirección del dispositivo esclavo con el que queremos comunicar.
- Un bit restante reservado para indicar la intención: enviar o recibir información.
- Un bit de validación.
- Uno o más bytes son los datos enviados o recibidos del esclavo.
- Un bit de validación.

Con los 7 bits de de dirección es posible acceder a 112 dispositivos en un mismo bus, ya que de las 128 direcciones posibles se reservan 16 para usos especiales. EL incremento de datos enviados, 18bits por cada 8bits de datos, supone que, en general, la velocidad del bus I2C es reducida. La velocidad estándar de transmisión es de 100kHz con un modo de alta velocidad de 400kHz [4].

En la siguiente figura se muestra el sistema de codificación para el inicio de la comunicación y la transmisión de datos.

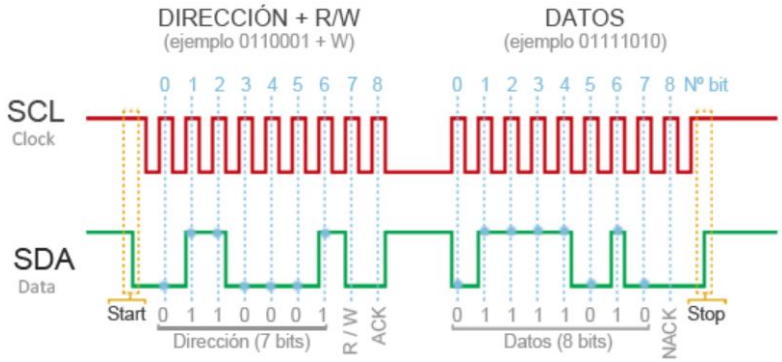


Figura 2.3 Esquema del protocolo de comunicación I2C.

Como se observa en el esquema, cuando no está iniciada la comunicación tanto la señal de reloj como la de datos se encuentran a uno. Tras esto, un flanco de bajada en SDA indica el inicio de la comunicación. Los siguientes 7 bits determinan la dirección del esclavo, luego, se indica si es lectura o escritura y el siguiente flanco de subida finaliza con el proceso de inicio.
Una vez conectado, la señal de reloj se encontrará a cero y la de datos a uno. De nuevo, un flanco de bajada en SDA inicia el proceso de comunicación, enviándose o recibiéndose paquetes de 8 bits. Se pondrá fin a la comunicación cuando haya un flanco de subida en SDA estando SCL activa y no encontrándose en mitad del proceso de transmisión o recepción de datos [4].

Como conclusión, se presentan las siguientes ventajas y desventajas de la comunicación I2C. Por un lado, requiere pocos cables: dispone de mecanismos para verificar que la señal ha llegado y, por otro lado, su velocidad es media-baja, no es full duplex y no hay verificación de que el contenido del mensaje sea correcto [4].

2.1.4 Implementación del MPU6050 en IDE de Arduino

Para la codificación del MPU6050 en IDE de Arduino se comenzó, en un principio, utilizando las librerías desarrolladas por Jeff Rowberg: "I2Cdev.h" y "MPU6050.h" y la librería "Wire.h" con la que cuenta el propio software de Arduino. Finalmente, se terminó usando únicamente la librería "Wire.h" ya que para nuestra aplicación, las desarrolladas por Jeff Rowberg no se han considerado necesarias [3].

En la siguiente imagen, se muestra como es la conexión entre el módulo ESP8266 y el MPU6050. Por un lado, los pines 4 y 5 están destinados para la comunicación I2C, siendo el 4(SDA) y el 5(SCL). Por otro lado, como se verá más adelante en el código, es necesario indicar la dirección del MPU6050. Esta puede ser 0x68 o 0x69, dependiendo del estado del pin AD0: 0x69(HIGH) o 0x68(LOW). Si no se especifica lo contrario, la dirección por defecto será 0x68.

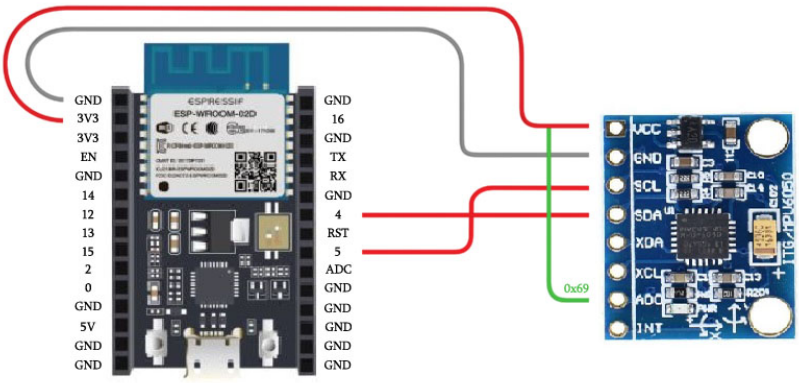


Figura 2.4 Esquema del protocolo de comunicación I2C.

Ahora, se pasa a mostrar los dos códigos para la lectura del acelerómetro. Un primer código en el que se utilizan las tres librerías y un segundo donde únicamente se usa la librería "Wire.h" [3].

Código 2.1 Lectura del sensor con las tres librerías.

```
//La libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro en los ejes x,y,z y
   escalados
int ax, ay, az;
float ax_m_s2, ay_m_s2, az_m_s2;
```

```
void setup() {
  Serial.begin(57600);  //Iniciando puerto serial 57600 bits/s(=baudios)
  Wire.begin();         //Iniciando I2C
  sensor.initialize();  //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado
      correctamente");
  else Serial.println("Error al iniciar el sensor");
}

void loop() {
  // Lectura de las aceleraciones
  sensor.getAcceleration(&ax, &ay, &az);

  //Escalamos las lecturas para pasarlas a m/s^2
  ax_m_s2 = ax * (9.81/16384.0);
  ay_m_s2 = ay * (9.81/16384.0);
  az_m_s2 = az * (9.81/16384.0);

  //Mostrar las lecturas separadas por un [tab]
  Serial.print("a[x y z]:\t");
  Serial.print(ax_m_s2); Serial.print("\t");
  Serial.print(ay_m_s2); Serial.print("\t");
  Serial.println(az_m_s2);
}
```

Como se puede observar, "sensor.getAcceleration(ax, ay, az)" nos muestra el valor de la aceleración en el rango $[-32767, 32768]$ ya que se tiene un convertidor analógico-digital de 16bits como se ha comentado anteriormente. Puesto que se ha ajustado el rango de aceleraciones a $\pm 2g$, $1g$ equivale a $2^{16}/4 = 16384$.

Código 2.2 Lectura del sensor únicamente con la librería "Wire.h".

```
// la libreria Wire.h para el MPU6050
#include "Wire.h"

//Direccion I2C de la IMU
#define MPU 0x68

// Valores RAW (sin procesar) del acelerometro en los ejes x,y,z y
    escalados
int ax, ay, az;
float ax_m_s2, ay_m_s2, az_m_s2;


void setup(){
  Serial.begin(115200);

  Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL
```

```
  Wire.beginTransmission(MPU1);
  Wire.write(0x6B);// PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);

}

void loop(){
  // Lectura de las aceleraciones
  Wire.beginTransmission(MPU);
  Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,6,true); //A partir del 0x3B, se piden 6
      registros
  ax=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
  ay=Wire.read()<<8|Wire.read();
  az=Wire.read()<<8|Wire.read();

  //Escalamos las lecturas para pasarlas a m/s^2
  ax_m_s2=ax*(9.81/16384);
  ay_m_s2=ay*(9.81/16384);
  az_m_s2=az*(9.81/16384);

  //Mostrar las lecturas separadas por un [tab]
  Serial.print("a[x y z]:\t");
  Serial.print(ax_m_s2); Serial.print("\t");
  Serial.print(ay_m_s2); Serial.print("\t");
  Serial.println(az_m_s2);
}
```

2.1.5 Configuración de los parámetros del MPU6050

Como ya se ha comentado, existen varios parámetros que pueden ser configurados, como son la sensibilidad y escala del sensor o la frecuencia de corte del filtro LPF (low pass filter) que incorpora a la salida. En este proyecto se han dejado dichos parámetros como vienen por defecto. No obstante, a continuación se explicará como se realizaría la configuración en los casos en los que fuese necesario [5].

La configuración se hará con el siguiente caso práctico: $500^o/s$ para el giróscopo, $\pm 2g$ para el acelerómetro y 98Hz de frecuencia de corte para el filtro de salida. Será necesario disponer del datasheet del MPU6050 de donde se extraerán los diferentes registros [5].

Como se puede observar en el siguiente código, en primer lugar se hace uso del registro 6B para activar el modo configuración del sensor y poder modificar los parámetros internos. Seguidamente, los registros 1B y 1C para configurar el giróscopo y el acelerómetro respectivamente. Finalmente, el registro 1A asociado al filtro pasa bajos de la salida. Todas estas configuraciones se insertan dentro de una subrutina que se ejecutará una única vez al iniciar el programa [5].

Código 2.3 Configuración de los parámetros del MPU6050.

```
void init() {
  Wire.beginTransmission(MPU6050_address);
```

```
  Wire.write(0x6B);                       // PWR_MGMT_1 registro 6B hex
  Wire.write(0x00);                       // 00000000 para activar
  Wire.endTransmission();
  Wire.beginTransmission(MPU6050_address);
  Wire.write(0x1B);                       // GYRO_CONFIG registro 1B hex
  Wire.write(0x08);                       // 00001000: 500dps
  Wire.endTransmission();
  Wire.beginTransmission(MPU6050_address);
  Wire.write(0x1C);                      // ACCEL_CONFIG registro 1C hex
  Wire.write(0x10);                      // 00010000: +/- 8g
  Wire.endTransmission();

  Wire.beginTransmission(MPU6050_address);
  Wire.write(0x1A);                    // LPF registro 1A hex
  Wire.write(0x03);                    // 256Hz(0ms):0x00 - 188Hz(2ms):0
      x01 - 98Hz(3ms):0x02 - 42Hz(4.9ms):0x03 - 20Hz(8.5ms):0x04 - 10Hz
      (13.8ms):0x05 - 5Hz(19ms):0x06
  Wire.endTransmission();
}
```

Pinchando en el siguiente enlace se accede al datasheet del MPU6050 [**Consultar datasheet**]. La información anteriormente comentada se encuentra en las páginas 13-15.

Por último, es muy conveniente siempre calibrar el sensor y eliminar el offset. Esta tarea juega un papel realmente importante cuando se desea emplear el sensor para estimar la inclinación de un determinado objeto o vehículo. Generalmente la calibración del giróscopo y del acelerómetro se realizan de forma separada durante la inicialización del código. Para ello, se toman un número considerado de muestras tanto del giróscopo como del acelerómetro y se calcula el valor medio, siendo este valor medio el offset del sensor (es imprescindible mantener el sensor lo más estático posible durante la calibración) [5].
Para los objetivos de este proyecto será suficiente con eliminar los offset manualmente. Para ello, se coloca el sensor en una superficie horizontal y se ajustan las salidas para que únicamente indique una aceleración igual a la de la gravedad en el eje z.

2.2 Módulo wifi ESP8266

ESP8266 es un chip integrado con conexión WiFi y compatible con el protocolo TCP/IP(Protocolo de control de transmisión/Protocolo de Internet), creado y fabricado por Espressif, una empresa China situada en Shangai. El objetivo principal de su desarrollo fue el de dar acceso a cualquier microcontrolador a una red. La gran ventaja del ESP8266 es su bajo consumo. Se considera el producto ideal para wearable y dispositivos del IoT [8].

La versión básica ESP-01 salió al mercado en el año 2014 de la mano de AI-Thinker. Debido a su bajo precio y las posibilidades que tenía, poco a poco, comenzó a traducirse la documentación y crear firmwares, software de bajo nivel, pare el ESP8266. Tras esto, hubo un gran crecimiento y las aplicaciones se multiplicaron. En la actualidad, se puede encontrar multitud de módulos que incorporan este microcontrolador. Se debe distinguir entre los módulos y los microcontroladores. El ESP8266 es un microcontrolador pues se puede trabajar con él suelto o comprarlo integrado dentro de un PCB (Printed Circuit Board)[8].

En cuanto a las principales características técnicas se tienen las siguientes:

- Voltaje de operación entre $3V$ y $3.6V$
- Corriente de operación $80mA$
- Temperatura de operación -40^oC y 125^oC
- Tiene 17 puertos GPIO, aunque solo se pueden usar 9.
- Soporta los principales buses de comunicación (SPI, **I2C**, UART).

En cuanto al consumo, este dependerá de diferentes factores como el modo en el que esté trabajando, los protocolos que se estén utilizando, la calidad de la señal WiFi y, sobre todo, de si se envía o se recibe información a través de la WiFi. Los valores oscilan entre los $0.5\mu A$ cuando el dispositivo está apagado y $170mA$ cuando se transmite [8].

El módulo wifi con el que se ha trabajado ha sido concretamente el ESP8266-DevKitC-02D-F, cuyo fabricante es Espressif Systems. Estos módulos son pequeñas placas que ahorran espacio y están diseñadas para admitir varios módulos ESP8266, incluidos ESP-WROOM-02D y ESP-WROOM-02U. Estas herramientas de desarrollo ESP8266-DevKitC cuentan con 5V to 3.3V LDO, dual switch, USB-to-UART bridge, micro USB port, boot button e I/O connector [8].

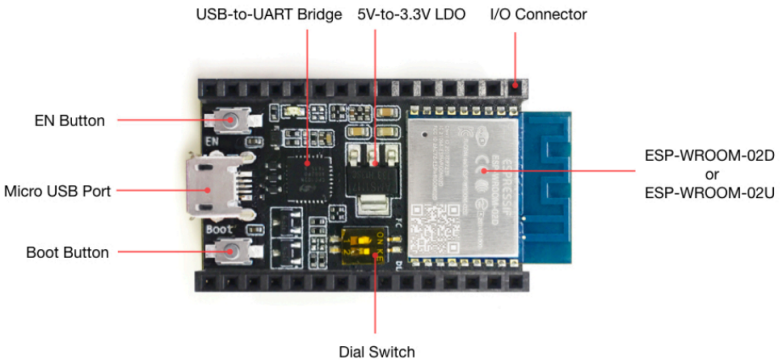


Figura 2.5 PLaca ESP8266-DevKitC-02D-F.

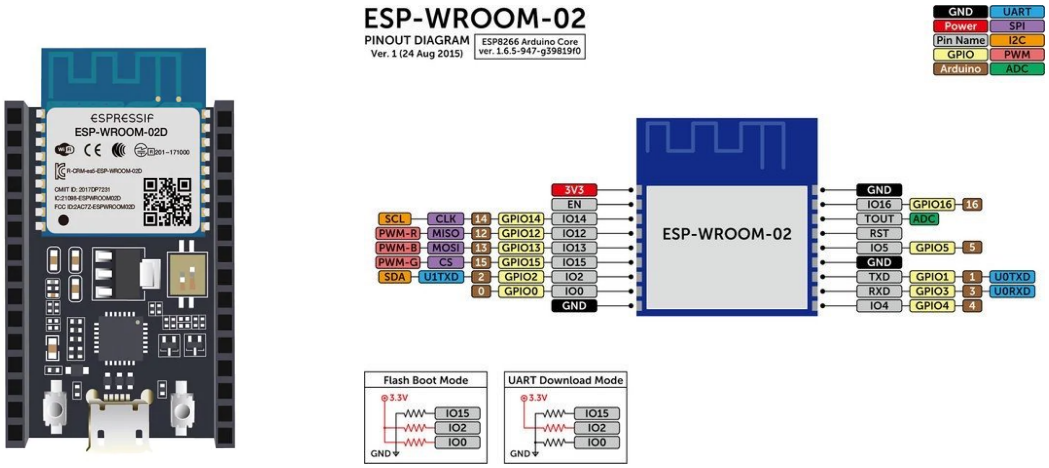


Figura 2.6 Imagen del módulo ESP8266 y esquema de los pines.

2.2.1 Implementación del ESP8266 en IDE de Arduino

El primer paso que se debe dar es la instalación de la placa ESP8266 en el IDE de Arduino. Esto es posible gracias a la comunidad ESP8266 que creó un complemento para el IDE Arduino que permite programar el ESP8266, utilizando este y su lenguaje de programación. Para realizar dicha instalación solo es necesario completar los siguientes pasos [7]:

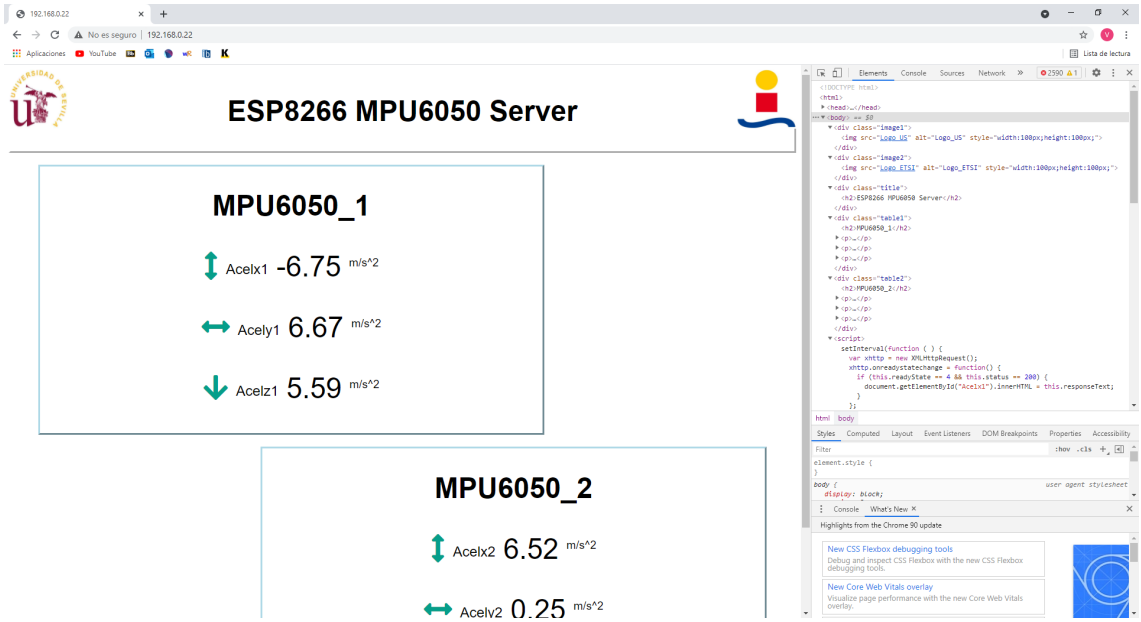
- 1. Una vez se está en el IDE de Arduino, se despliega la pestaña de Archivo y se hace clic en preferencia.
- 2. Se pega el siguiente enlace *hhttp://arduino.esp8266.com/stable/package_esp8266com_index.jso* en el campo "Gestor de URLs Adicionales de Tarjetas" y se acepta.
- 3. Se despliega la pestaña de Herramientas y se hace clic en Placa > Gestor de Tarjetas. Se busca ESP8266 y se pulsa instalar para la" ESP8266 por ESP8266 Comunidad".
- Después de unos segundos debe aparecer como instalado, dándose por terminada la instalación.

La codificación del ESP8266 en Arduino no es fija, por esta razón se detallará en función de la configuración en el siguiente capítulo.

3 Descripción de las diferentes configuraciones

3.1 Datos volcados en página web + adquisición con Excel o Matlab

En esta configuración solo es necesario contar con un módulo ESP8266. Este se conecta a la red local y vuelca los valores en una página web que se ha creado, programando en HTML. La actualización de dicha página web se hace de manera automática como se comentará más adelante [9].
Una vez se tienen los valores de las aceleraciones en la página web, se pasa a la extracción de estos con dos métodos diferentes. Un primer método, con el Editor de Visual Basic de Excel [12] y un segundo, con la función webread de Matlab [13].



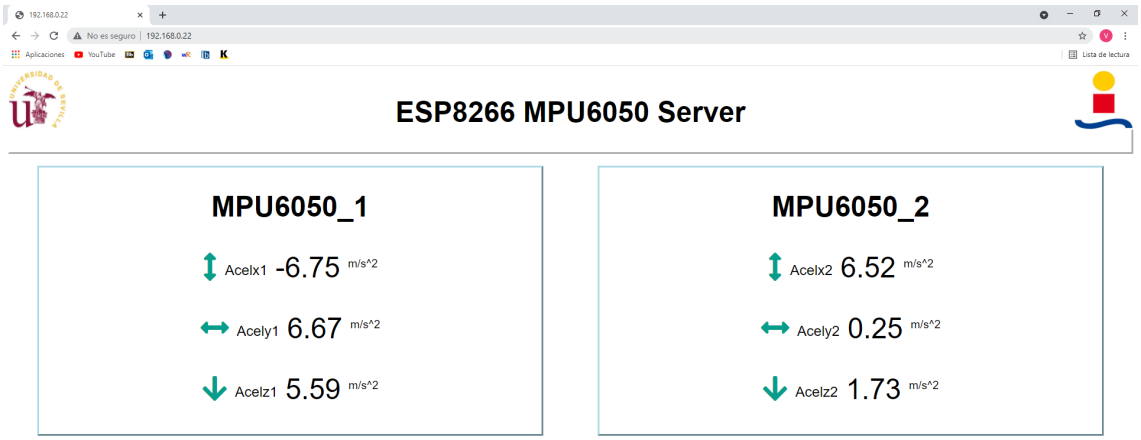


Figura 3.1 Página web y código HTML.

3.1.1 Implementación del código en IDE de Arduino

En esta sección se hará una descripción detallada del código. Código que es cargado en el módulo ESP8266 para poder llevar a cabo lo anteriormente comentado.

En primer lugar, se introducen las librerías necesarias. Por un lado, la librería Open Source "ESPAsyncWebServer" que permite crear un servidor asíncrono, es decir, que es capaz de atender a varios clientes de forma simultánea. Para que esta librería funcione se debe añadir también " Async TCP". Por otro lado, se incluyen las necesarias tanto para comunicarse con los acelerómetros como para identificar la placa en el IDE de Arduino. Por último, se añaden las librerías para usar el lenguaje HTML (CSS) y la inserción de los logos en la página web.

Código 3.1 Librerías necesarias.

```
// la libreria Wire.h
#include "Wire.h"
//Incluimos las librerias necesarias del módulo Wifi y lenguaje HTML (
    CSS)
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Hash.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "FS.h" //Para insertar los logos en la página web
```

En las dos líneas siguientes se le facilitan los credenciales para que la conexión con la red local sea satisfactoria.

Código 3.2 Definición de los credenciales de la red.

```
// Conección a la red Local
const char* ssid = "SSID";
const char* password = "PASSWORD";
```

Se pasa a definir tanto la dos direcciones para la comunicación I2C como las diferentes variables que se van a utilizar. Generalmente, las variables que almacenan tiempo se definen como "unsigned long", ya que su valor crece rápidamente. La variable "interval" determina el tiempo que transcurre

entre cada actualización automática de la página web.
Además, se crea el servidor en el puerto 80 con la siguientes sentencia: "AsyncWebServer server(80)"

Código 3.3 Direcciones I2C, variables y servidor en el puerto 80.

```
#define MPU1 0x68
#define MPU2 0x69

int16_t AcX1, AcY1, AcZ1;
int16_t AcX2, AcY2, AcZ2;
float axprint1, ayprint1, azprint1;
float axprint2, ayprint2, azprint2;

AsyncWebServer server(80);

unsigned long previousMillis = 0;

const long interval = 100; //tiempo en ms
```

Las siguientes líneas van asociadas tanto a la programación de la página web como a la configuración de la actualización automática cada X segundos de dicha página, con JavaScript. Con la etiqueta <meta> se consigue que nuestra página web responda en cualquier navegador. La etiqueta <Link> permite cargar los iconos del sitio web font awesome: *https://fontawesome.com/icons?d=gallery&p=2*. Entre las etiquetas <style></style> se agrega algo de Css para darle estilo a la página web: básicamente se está configurándola para que muestre el texto con fuente Arial en un bloque sin márgenes y centrado. Dentro de las etiquetas <body></body> es donde se establece el contenido de la página web: con las etiquetas <h2></h2> se agrega un encabezado a la misma y los párrafos están delimitados por las etiquetas y . Como se puede observar, las variables que se actualizan llevan asociado una etiqueta entre %, Ej: $\%ACELX1\%$. Resaltar que, a nuestro parecer, una de las mejores páginas para aprender HTML de forma dinámica y autodidacta es: *https://www.w3schools.com/*. Por último, las sentencias entre las etiquetas <script></script> configuran la actualización de las diferentes variables de aceleración en la página web cada $X(ms)$ definido en la variable "interval", como ya se ha comentado.

Código 3.4 Código HTML.

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5
      .7.2/css/all.css" integrity="sha384-
      fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
      crossorigin="anonymous">
  <style>
    html {
     font-family: Arial;
     display: inline-block;
     margin: 0px auto;
     text-align: center;
```

```
    }
    h2 { font-size: 3.0rem; }
    p { font-size: 3.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;
      vertical-align:middle;
      padding-bottom: 15px;
    }

    .image1 {
    float:left;
    text-align: center;
    }
    .image2 {
    float:right;
    text-align: center;
    }
    .title {
    border: 3px outset white;
    background-color: white;
    text-align: center;
    }
    .table1 {
    float:left;
    margin-top:20px;
    margin-left:50px;
    border: 3px outset lightblue;
    background-color: white;
    width:850px;
    text-align: center;
    }
    .table2 {
    float:right;
    margin-top:20px;
    margin-right:50px;
    border: 3px outset lightblue;
    background-color: white;
    width:850px;
    text-align: center;
    }
  </style>
</head>
<body>

  <div class="image1">
  <img src="Logo_US" alt="Logo_US" style="width:100px;height:100px;">
  </div>

  <div class="image2">
```

```
<img src="Logo_ETSI" alt="Logo_ETSI" style="width:100px;height:100px
    ;">
</div>

<div class="title">
<h2>ESP8266 MPU6050 Server</h2>
</div>

<div class="table1">
<h2>MPU6050_1</h2>
<p>
  <i class="fas fa-arrows-alt-v" style="color:#059e8a;"></i>
  <span class="dht-labels">Acelx1</span>
  <span id="Acelx1">%ACELX1%</span>
  <sup class="units">m/s^2</sup>
</p>
<p>
  <i class="fas fa-arrows-alt-h" style="color:#059e8a;"></i>
  <span class="dht-labels">Acely1</span>
  <span id="Acely1">%ACELY1%</span>
  <sup class="units">m/s^2</sup>
</p>
<p>
  <i class="fas fa-arrow-down" style="color:#059e8a;"></i>
  <span class="dht-labels">Acelz1</span>
  <span id="Acelz1">%ACELZ1%</span>
  <sup class="units">m/s^2</sup>
</p>
</div>

<div class="table2">
<h2>MPU6050_2</h2>
<p>
  <i class="fas fa-arrows-alt-v" style="color:#059e8a;"></i>
  <span class="dht-labels">Acelx2</span>
  <span id="Acelx2">%ACELX2%</span>
  <sup class="units">m/s^2</sup>
</p>
<p>
  <i class="fas fa-arrows-alt-h" style="color:#059e8a;"></i>
  <span class="dht-labels">Acely2</span>
  <span id="Acely2">%ACELY2%</span>
  <sup class="units">m/s^2</sup>
</p>
<p>
  <i class="fas fa-arrow-down" style="color:#059e8a;"></i>
  <span class="dht-labels">Acelz2</span>
  <span id="Acelz2">%ACELZ2%</span>
  <sup class="units">m/s^2</sup>
</p>
```

```
  </div>

</body>
```

Para actualizar los valores de las aceleraciones se dispone de la función "setInterval()" que se ejecuta en el intervalo especificado. Esta función lo que hace es realizar unas solicitudes en las URL/Acel- -, esto para cada eje y sensor, para obtener las últimas lecturas de las aceleraciones. Una vez que recibe los valores, actualiza los elementos HTML cuyo id son Acel- -.

Código 3.5 Actualización automática con JavaScript.

```
<script>
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Acelx1").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/Acelx1", true);
  xhttp.send();
}, 100 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Acely1").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/Acely1", true);
  xhttp.send();
}, 100 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Acelz1").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/Acelz1", true);
  xhttp.send();
}, 100 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Acelx2").innerHTML = this.responseText;
```

```
    }
  };
  xhttp.open("GET", "/Acelx2", true);
  xhttp.send();
}, 100 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Acely2").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/Acely2", true);
  xhttp.send();
}, 100 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Acelz2").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/Acelz2", true);
  xhttp.send();
}, 100 ) ;
</script>
</html>)rawliteral";
```

Es necesario crear la función processor() que reemplazará las etiquetas del texto HTML con los valores actualizados de aceleraciones.

Código 3.6 Función processor().

```
String processor(const String& var){
  //Serial.println(var);
  if(var == "ACELX1"){
    return String(axprint1);
  }
  else if(var == "ACELY1"){
    return String(ayprint1);
  }
  else if(var == "ACELZ1"){
    return String(azprint1);
  }
  else if(var == "ACELX2"){
    return String(axprint2);
  }
  else if(var == "ACELY2"){
    return String(ayprint2);
```

```
  }
  else if(var == "ACELZ2"){
    return String(azprint2);
  }
  return String();
}
```

Se procede a definir el setup(). En primer lugar, se inicializa el monitor serial y ambos acelerómetros. En segundo lugar, se conecta el módulo a la red local y se saca por el puerto serie la dirección IP del ESP8266 a la que se deberá acceder por el navegador web. Por último, se incluye una condición if necesaria para la inserción de los logos.

Código 3.7 void setup()(1).

```
void setup() {
  Serial.begin(115200);  //Iniciando puerto serial

   Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL
   Wire.beginTransmission(MPU1);
   Wire.write(0x6B);
   Wire.write(0);
   Wire.endTransmission(true);

   Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL
   Wire.beginTransmission(MPU2);
   Wire.write(0x6B);
   Wire.write(0);
   Wire.endTransmission(true);

  // Nos conectamos a la Wifi Local
  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println(".");
  }

  // Mostramos por el puerto serie la dirección IP del ESP8266
  Serial.println(WiFi.localIP());

    //Necesario para los logos
  if(!SPIFFS.begin()){
       Serial.println("An Error has occurred while mounting SPIFFS");
       return;
  }
```

Para concluir con el setup, se incluyen las siguientes líneas de código para gestionar el servidor web.
Cuando se hace una solicitud en la URL, se envía el texto HTML que se almacena en la variable "*index_html*". También es necesario pasar por la función processor() que reemplazará todas las etiquetas de posición con los valores correctos.

Se necesitan agregar dos controladores adicionales para actualizar las lecturas de las aceleraciones. Cuando se recibe una solicitud en la URL de /Acel- - simplemente se debe enviar el valor de aceleración actualizado. Es texto sin formato y debe enviarse como un carácter, por lo que se usa el "$c\_str()$".
Por último, se inicializa el servidor.

Código 3.8 void setup()(2).

```
//Primero que nada cargamos los logos:
  server.on("/Logo_US", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/Logo_US.JPG", "image/JPG");
  });

  server.on("/Logo_ETSI", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/Logo_ETSI.JPG", "image/JPG");
  });
//////////////////////////

  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/Acelx1", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(axprint1).c_str());
  });
  server.on("/Acely1", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(ayprint1).c_str());
  });
  server.on("/Acelz1", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(azprint1).c_str());
  });
  server.on("/Acelx2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(axprint2).c_str());
  });
  server.on("/Acely2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(ayprint2).c_str());
  });
  server.on("/Acelz2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", String(azprint2).c_str());
  });

  server.begin();

}
```

En el loop(), se introduce un if que permite hacer lecturas de los sensores en el periodo de tiempo definido en la variable "interval". Con esta condición, se leen los valores únicamente cuando se van a mostrar (actualizar) en la página web.

Código 3.9 void loop().

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the MPU6050 values
    previousMillis = currentMillis;

    //Leer los valores del Acelerometro de la IMU1
    Wire.beginTransmission(MPU1);
    Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
    Wire.endTransmission(false);
    Wire.requestFrom(MPU1,6,true); //A partir del 0x3B, se piden 6
        registros
    AcX1=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
    AcY1=Wire.read()<<8|Wire.read();
    AcZ1=Wire.read()<<8|Wire.read();

    //Leer los valores del Acelerometro de la IMU2
    Wire.beginTransmission(MPU2);
    Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
    Wire.endTransmission(false);
    Wire.requestFrom(MPU2,6,true); //A partir del 0x3B, se piden 6
        registros
    AcX2=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
    AcY2=Wire.read()<<8|Wire.read();
    AcZ2=Wire.read()<<8|Wire.read();

    //Escalado de las aceleraciones
    //MPU6050_1
    axprint1=AcX1*(9.81/16384)-2.3;
    ayprint1=AcY1*(9.81/16384)+0.15;
    azprint1=AcZ1*(9.81/16384);
    //MPU6050_2
    axprint2=AcX2*(9.81/8192)-3.4;
    ayprint2=AcY2*(9.81/8192)+0.2;
    azprint2=AcZ2*(9.81/8192)-4.25;
  }
}
```

3.1.2 Adquisición con Excel

El Web Scraping es una técnica usada para extraer información desde páginas Web. Se puede extraer tanto texto, emails o links como descargar ficheros o imágenes. También se puede extraer información adicional que no está visible en la Web. Para esta técnica se usan lenguajes de programación como Python, no obstante, también se puede usar el lenguaje VBA (Visual Basic para aplicaciones) que es el que se utilizará en este proyecto. Como se puede observar en el siguiente esquema, el método consiste en la extracción de los valores de la página web con el editor de Visual Basic y el muestreo en las celdas de Excel [12].

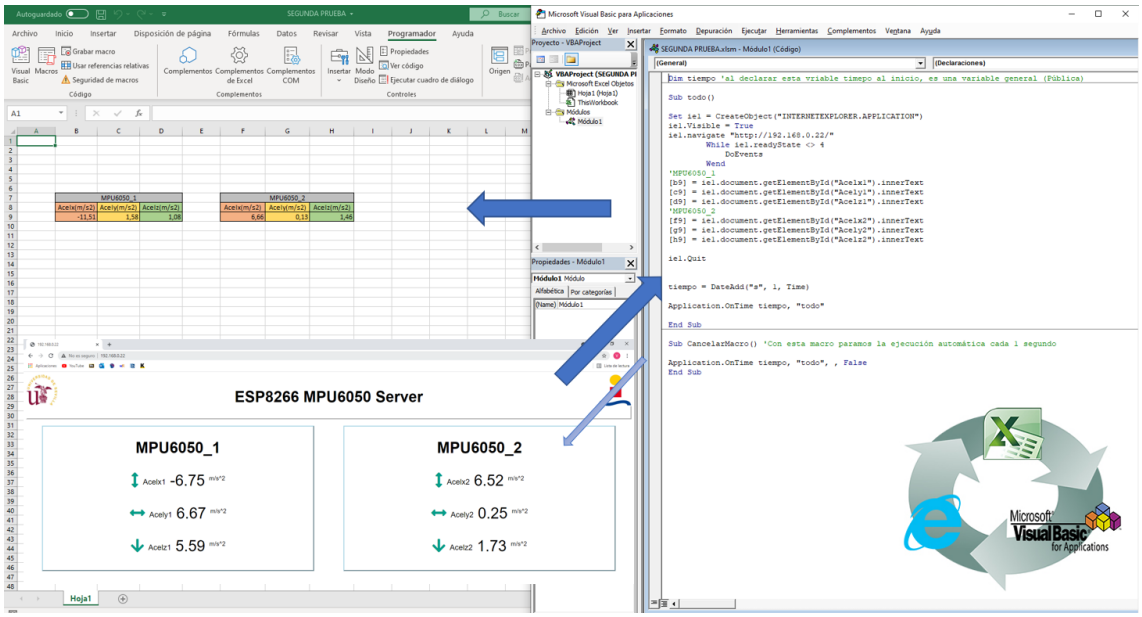


Figura 3.2 Esquema del funcionamiento.

Se pasa a explicar las sentencias del siguiente código que permiten extraer los valores de la página web cada X segundos. En primer lugar, se declara la variable "tiempo" como una variable general(pública). En segundo lugar, se ha creado dos subrutinas o procedimientos. Una cuyo objetivo es la extracción de los valores de aceleraciones cada X segundos de manera automática y otra para parar, cuando se desee, dicha ejecución automática.
En la primera subrutina, como se puede observar, se habilita el Internet Explorer, se comprueba que se ha hecho de manera correcta y se dirige a la URL especificada entre comillas. Una vez se ha realizado esta apertura, es posible la extracción de los elementos por id de la página web con el comando "ie1.document.getElementById("Acel- -").innerText". Además, se especifica entre corchetes las celdas de Excel en las que se quiere mostrar los valores extraídos. Por último, las dos sentencias siguientes: "tiempo = DateAdd("s", 1, Time)" y "Application.OnTime tiempo, "todo"" ejecutan de manera automática, en el periodo de tiempo impuesto, la subrutina todo(). El menor tiempo que permite introducir es $1s$, periodo muy grande para la frecuencia de adquisición que se desea conseguir.
En la segunda subrutina, como ya se ha comentado, simplemente se cancela la ejecución automática de la subrutina todo().

Código 3.10 Editor de Visual Basic.

```
Dim tiempo 'al declarar esta variable tiempo al inicio, es una variable
    general (Pública)
```

```
---------------------------------------------------------------------
Sub todo()

Set ie1 = CreateObject("INTERNETEXPLORER.APPLICATION")
ie1.Visible = True
ie1.navigate "http://192.168.0.22/"
       While ie1.readyState <> 4
           DoEvents
       Wend
'MPU6050_1
[b9] = ie1.document.getElementById("Acelx1").innerText
[c9] = ie1.document.getElementById("Acely1").innerText
[d9] = ie1.document.getElementById("Acelz1").innerText
'MPU6050_2
[f9] = ie1.document.getElementById("Acelx2").innerText
[g9] = ie1.document.getElementById("Acely2").innerText
[h9] = ie1.document.getElementById("Acelz2").innerText

ie1.Quit


tiempo = DateAdd("s", 1, Time)

Application.OnTime tiempo, "todo"

End Sub
---------------------------------------------------------------------
Sub CancelarMacro() 'Con esta macro paramos la ejecución automática cada
    1 segundo

Application.OnTime tiempo, "todo", , False
End Sub
```

Calificación del método: Como ya se ha visto, se ha hecho Web Scraping con el objeto IE Explorer que lo que hace es simular una navegación por una web en concreto. Este método es demasiado lento e inestable. Además, el periodo de ejecución automática del VBA no puede ser inferior a un segundo, con lo cual, este método de adquisición se ha descartado debido a que ha sido imposible conseguir la frecuencia de adquisición de datos impuesta en los objetivos de partida.

3.1.3 Adquisición con Matlab

El segundo método de adquisición para la primera configuración consiste en la utilización de la función **webread** de Matlab que nos permite cargar el código HTML (estructura) de la página web cuya URL se especifica entre paréntesis. Una vez se dispone de esa cadena de caracteres, es posible extraer, recorriendo dicha cadena, los valores de las aceleraciones.
Como se muestra remarcado en la imagen, se obtienen valores de aceleración en cada eje en una media de unos 300ms. Lo que da una frecuencia de adquisición de unos 3-4 valores por segundo. Frecuencia que se encuentra muy por debajo de la deseada.

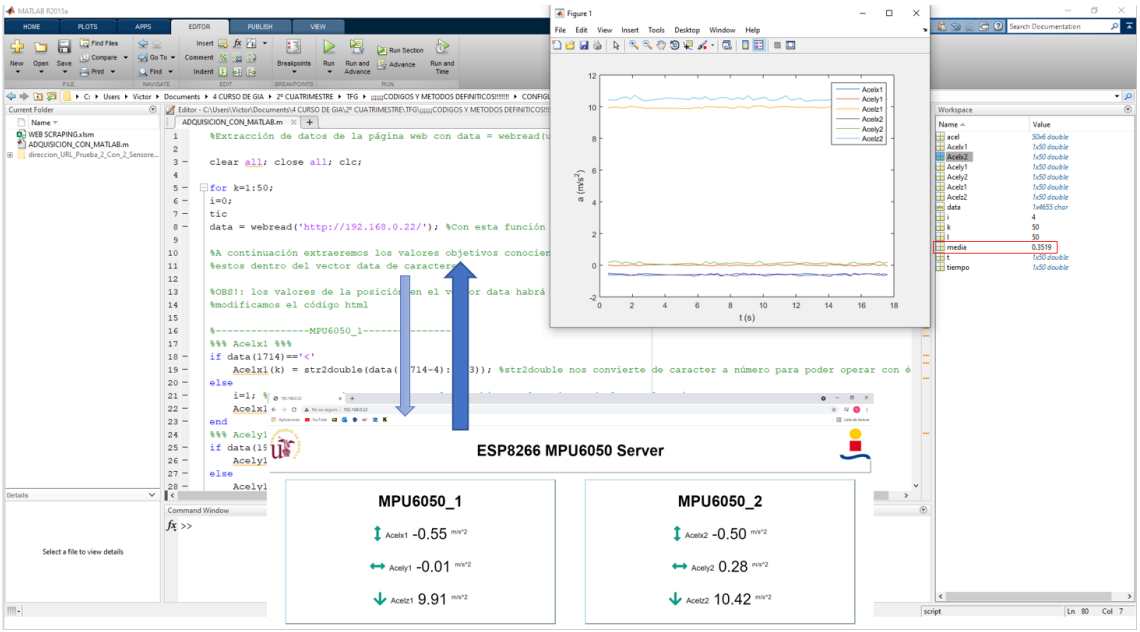


Figura 3.3 Esquema del funcionamiento.

Tal como se observa en el siguiente código, lo que se ha hecho principalmente ha sido una manipulación de la cadena que contiene el código HTML de la página web, usando webread, con el fin de extraer de este los valores de las aceleraciones. Por último, se representan dichas aceleraciones frente al tiempo.

Código 3.11 Adquisición con Matlab.

```
%Extracción de datos de la página web con data = webread(url)%

clear all; close all; clc;

for k=1:50;
i=0;
tic
data = webread('http://192.168.0.22/'); %Con esta función copiamos el có
    digo html (estructura de la página web)

%A continuación extraeremos los valores objetivos conociendo la posición
     de
%estos dentro del vector data de caracteres
```

```
%OBS!: los valores de la posición en el vector data habrá que cambiarlos
     si
%modificamos el código html

%----------------MPU6050_1-----------------%
%%% Acelx1 %%%
if data(1714)=='<'
    Acelx1(k) = str2double(data((1714-4):1713)); %str2double nos
       convierte de caracter a número %para poder operar con él
else
    i=1; %Con esto tendremos en cuenta los cambios en los signos de las
       aceleraciones
    Acelx1(k) = str2double(data((1714-4):1714));
end
%%% Acely1 %%%
if data(1904+i)=='<'
    Acely1(k) = str2double(data((1904+i-4):1903+i));
else
    Acely1(k) = str2double(data((1904+i-4):1904+i));
    i=i+1;
end
%%% Acelz1 %%%
if data(2091+i)=='<'
    Acelz1(k) = str2double(data((2091+i-4):2090+i));
else
    Acelz1(k) = str2double(data((2091+i-4):2091+i));
    i=i+1;
end

%----------------MPU6050_2-----------------%
%%% Acelx2 %%%
if data(2338+i)=='<'
    Acelx2(k) = str2double(data((2338+i-4):2337+i));
else
    Acelx2(k) = str2double(data((2338+i-4):2338+i));
    i=i+1;
end
%%% Acely2 %%%
if data(2528+i)=='<'
    Acely2(k) = str2double(data((2528+i-4):2527+i));
else
    Acely2(k) = str2double(data((2528+i-4):2528+i));
    i=i+1;
end
%%% Acelz2 %%%
if data(2715+i)=='<'
    Acelz2(k) = str2double(data((2715+i-4):2714+i));
else
    Acelz2(k) = str2double(data((2715+i-4):2715+i));
    i=i+1;
```

```
end

t(k)=toc; %Estamos midiendo cada unos (en media) 0.3s = 300ms

end

%Cálculo del tiempo medio de medición
media=0;
for l=1:50
    media=media+t(l);
end
media=media/l;
%----------------------------------%

tiempo(1)=t(1);
for k=2:50
    tiempo(k)=tiempo(k-1)+t(k);
end

acel=[Acelx1' Acely1' Acelz1' Acelx2' Acely2' Acelz2'];
figure
title('Representación de las aceleraciones')
plot(tiempo,acel)
legend('Acelx1','Acely1','Acelz1','Acelx2','Acely2','Acelz2')
xlabel('t (s)')
ylabel('a (m/s^2)')
```

Calificación del método: Como se puede observar, se ha mejorado el periodo de muestreo con respecto al método anterior. No obstante, como ya se ha comentado, la frecuencia de adquisición lograda está muy por debajo de la deseada, en torno a 3-4 valores en cada eje por segundo por lo que este método también ha sido descartado.

3.1.4 Conclusiones de esta primera configuración

Tras realizar los dos métodos de adquisición con esta primera configuración se han detectado las siguientes limitaciones o impedimentos:

- Frecuencias de adquisición resultantes muy por debajo de la necesaria.
- Dependencia de los protocolos de Internet que nos impiden tener un control total del tiempo.
- Gran dependencia con la velocidad de Internet de la que se disponga, respuesta del navegador frente a las actualizaciones...
- Pérdida o desecho de valores medidos por los acelerómetros. Esto ocurre por la gran dificultad o incluso imposibilidad de sincronizar las actualizaciones con las adquisiciones o extracciones de los valores de la página web.

Estas limitaciones han impulsado el análisis una segunda configuración que no depende de Internet sino, simplemente, de una comunicación inalámbrica entres dos módulos ESP8266.

3.2 Servidor y clientes + adquisición con Matlab o con aplicación Coolterm

En esta segunda configuración será necesario disponer de dos módulos ESP8266. Lo que se hará será establecer una comunicación Wi-Fi (HTTP) entre los dos ESP8266 para intercambiar datos sin necesidad de conectarse a Internet. Para ello, se va a configurar un ESP8266 como punto de acceso (servidor) y el otro ESP8266 como estación (cliente). Los intercambios de datos y lecturas de los sensores se harán a través de solicitudes HTTP [14].

Para comprender mejor cómo funciona se muestra el siguiente esquema y las respectivas explicaciones.

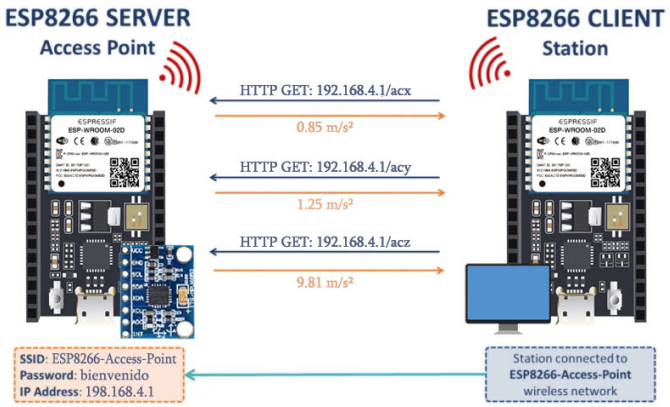


Figura 3.4 Esquema explicativo de la configuración servidor-ciente.

- El servidor ESP8266 crea su propia red inalámbrica (ESP8266 Soft-Access Point). Por lo tanto, otros dispositivos Wi-Fi pueden conectarse a esa red(SSID: ESP8266-Access-Point, Password: bienvenido).
- El cliente ESP8266 se fija como estación por lo que puede conectarse a la red inalámbrica del servidor ESP8266.
- El cliente puede realizar peticiones HTTP GET al servidor para solicitar datos del sensor. Solo tiene que utilizar la dirección IP del servidor para hacer una solicitud en una determinada ruta: /acx , /acy o /acz.
- El servidor recibe las solicitudes entrantes y envía una respuesta con las lecturas.
- El cliente recibe las lecturas y las muestra por el puerto serie.

3.2.1 Implementación del código del servidor en IDE de Arduino

Una vez se ha entendido el funcionamiento, se procede a implementar el código necesario a cargar en el ESP8266 para establecerlo como servidor.

Antes que nada, se cargan las librerías necesarias. Como ya se comentó en la primera configuración, se tiene la librería "Wire.h" para la lectura del MPU6050, se declara la placa y por último la librería Open Source "ESPAsyncWebServer" que permite crear un servidor asíncrono. Por último, se define la dirección $0x68$ para la comunicación I2C con el acelerómetro.

Código 3.12 Librerías necesarias.

```
#include "Wire.h"
#include <ESP8266WiFi.h>
```

```
#include "ESPAsyncWebServer.h"

//Direccion I2C de la IMU
#define MPU1 0x68
```

A continuación, se definen los credenciales de la red para que la conexión se haga de manera satisfactoria. Además, se crea el servidor en el puerto 80 con la siguiente sentencia: "AsyncWebServer server(80)"

Código 3.13 Credenciales de la red + creación del servidor.

```
const char* ssid = "ESP8266-Access-Point";
const char* password = "bienvenido";

AsyncWebServer server(80);
```

Se definen las diferentes variables que se van a usar.

Código 3.14 Definición de variables.

```
int16_t AcX1, AcY1, AcZ1;
float axprint1, ayprint1, azprint1;

String vecaxprint1_1, vecayprint1_1, vecazprint1_1, myaxprint1,
    myayprint1, myazprint1;
String memoryax1, memoryay1, memoryaz1;

String vecaxprint1_2, vecayprint1_2, vecazprint1_2;

int ITER;

unsigned long cont, t1, t2;
float t1s, t2s;
String cadcont, cadt1, cadt2;
```

En las siguientes sentencias se crean tres funciones que devolverán las aceleraciones en los tres ejes como variables String.

Código 3.15 Funciones para devolver String tras peticiones.

```
String readAX1_2() {
  return String(vecaxprint1_2);
}

String readAY1_2() {
  return String(vecayprint1_2);
}

String readAZ1_2() {
  return String(vecazprint1_2);
}
```

En el setup(), primero que nada, se inicializa el monitor serial. Después, se establece el ESP8266 como punto de acceso con el SSID y la PASSWORD definidos anteriormente. A continuación, se gestionan las rutas donde el ESP8266 estará escuchando las solicitudes entrantes. Por ejemplo, cuando el servidor ESP8266 recibe una solicitud $/acx1\_2$, este envía la aceleración en el eje x ,devuelta por la función $readAX1\_2()$, como un caracter (es por ello que se use el método $c\_str()$). Las siguientes sentencias van destinadas a inicializar el sensor MPU6050 y el servidor.

Código 3.16 void setup().

```
void setup(){

  Serial.begin(115200);

  Serial.print("Setting AP (Access Point)");
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.println(IP);

  server.on("/acx1_2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readAX1_2().c_str());
  });
  server.on("/acy1_2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readAY1_2().c_str());
  });
  server.on("/acz1_2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readAZ1_2().c_str());
  });

  Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL
  Wire.beginTransmission(MPU1);
  Wire.write(0x6B);// PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);

  server.begin();
}
```

En el loop(), lo primero que se hace es utilizar el comando "WiFi.softAPgetStationNum()" que permite conocer si el otro módulo(cliente) está conectado o no. Con lo cual, al verificar la condición, se empiezan a leer las aceleraciones en los tres ejes y se crean cadenas para cada eje con 100 valores cada una. Esto se ha hecho con el objetivo de aumentar la frecuencia de adquisición de datos, puesto que si se hicieran peticiones para valores individuales, por un lado, la frecuencia disminuiría notoriamente ya que en lo que más demora hay es en las peticiones http y, por otro lado, mientras se está haciendo una petición se están desechando los valores de aceleraciones que sigue midiendo el sensor. Con las cadenas no se están desechando valores ya que, como se puede observar, al final del código se almacenan las cadenas "$veca-print1\_1$" en otras cadenas "$veca-print1\_2$", quedando estas últimas invariables mientras se crean las siguientes. Se dispone de tiempo más que suficiente para hacer la petición antes de que se terminen de crear las nuevas cadenas con los 100 valores en cada una. Asimismo, en las cadenas "$veca-print1\_2$" se le añade el tiempo de inicio y fin de la

creación de esa cadena y dos contadores. Por una parte, el tiempo es necesario para conocer cual es el periodo de lectura y con ello la frecuencia de adquisición, por otra, los contadores también son necesarios , como se verá, para tener claro a qué iteración y a qué eje corresponde cada cadena mostrada por el puerto serie.

Código 3.17 void loop().

```
void loop(){

    if(WiFi.softAPgetStationNum() == 1){

    t1=millis();
      for(ITER=0; ITER<100; ITER++){
    Wire.beginTransmission(MPU1);
    Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
    Wire.endTransmission(false);
    Wire.requestFrom(MPU1,6,true); //A partir del 0x3B, se piden 6
        registros
    AcX1=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
    AcY1=Wire.read()<<8|Wire.read();
    AcZ1=Wire.read()<<8|Wire.read();

    //Escalado de las aceleraciones
    //MPU6050_1
    axprint1=AcX1*(9.81/16384)-2.7;
    ayprint1=AcY1*(9.81/16384);
    azprint1=AcZ1*(9.81/16384);

    myaxprint1 = String(axprint1);
    vecaxprint1_1= memoryax1 + " " + myaxprint1;
    memoryax1=vecaxprint1_1;
    myayprint1 = String(ayprint1);
    vecayprint1_1= memoryay1 + " " + myayprint1;
    memoryay1=vecayprint1_1;
    myazprint1 = String(azprint1);
    vecazprint1_1= memoryaz1 + " " + myazprint1;
    memoryaz1=vecazprint1_1;
    delay(2.3);
      }
      memoryax1="";
      memoryay1="";
      memoryaz1="";
      t2=millis();
      t1s = t1/1000.0;
      t2s = t2/1000.0;
      cadt1=String(t1s);
      cadt2=String(t2s);
      cadcont=String(cont);
      vecaxprint1_2 = cadt1 + " " + cadcont + " " + "1" + vecaxprint1_1 +
          " " + cadt2;
```

```
      vecayprint1_2 = cadt1 + " " + cadcont + " " + "2" + vecayprint1_1 +
          " " + cadt2;
      vecazprint1_2 = cadt1 + " " + cadcont + " " + "3" + vecazprint1_1 +
          " " + cadt2;
      cont++; //en convertir estas cadenas se invierte en torno a 1ms
    }
}
```

3.2.2 Implementación del código del cliente en IDE de Arduino

Antes que nada, se incluyen las bibliotecas necesarias para la conexión Wi-Fi y para realizar solicitudes HTTP. También es necesario crear una WiFiMulti instancia. Por último, se insertan las credenciales de red del servidor ESP8266.

Código 3.18 Librerías y credenciales de red.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti WiFiMulti;

const char* ssid = "ESP8266-Access-Point";
const char* password = "bienvenido";
```

A continuación, se guardan las direcciones URL donde el cliente realizará las solicitudes HTTP. El servidor ESP8266 tiene la dirección IP 192.168.4.1, y estaremos haciendo solicitudes en el URL /*acx*1_2, /*acy*1_2 y /*acz*1_2. Además, se definen las tres cadenas que utilizaremos en el código.

Código 3.19 Definición de variables.

```
const char* serverNameax1_2 = "http://192.168.4.1/acx1_2";
const char* serverNameay1_2 = "http://192.168.4.1/acy1_2";
const char* serverNameaz1_2 = "http://192.168.4.1/acz1_2";

String axprint1_2, ayprint1_2, azprint1_2;
```

En el void(), se conecta el ESP8266 cliente a la red del servidor ESP8266.

Código 3.20 void setup().

```
void setup() {
  Serial.begin(115200);
  Serial.println();

  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP(ssid, password);
  while((WiFiMulti.run() == WL_CONNECTED)) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("");
  Serial.println("Connected to WiFi");
}
```

En el loop() se impone la condición de que se esté conectado al servidor antes de hacer ninguna petición. A continuación, se pasan a hacer las tres peticiones HTTP llamando a la función "httpGETRequest()". Por último, el "delay(2.3)" está puesto de manera que le dé tiempo al servidor a crear las nuevas cadenas y así, pedir lo mismo lo menos posible.

Código 3.21 void loop().

```
void loop() {
    if ((WiFiMulti.run() == WL_CONNECTED)) {
      axprint1_2 = httpGETRequest(serverNameax1_2);
      ayprint1_2 = httpGETRequest(serverNameay1_2);
      azprint1_2 = httpGETRequest(serverNameaz1_2);
      Serial.println(axprint1_2);
      Serial.println(ayprint1_2);
      Serial.println(azprint1_2);
      delay(2.3);
    }
}
String httpGETRequest(const char* serverName) {
  WiFiClient client;

  HTTPClient http;

  http.begin(client, serverName);

  int httpResponseCode = http.GET();

  String payload = "--";

  http.end();

  return payload;
}
```

3.2.3 Adquisición con Matlab

Este primer método de adquisición para la segunda configuración no presentó un buen comportamiento desde el principio. La idea consistía en leer directamente el puerto serie haciendo uso de la función "dato = fscanf(canalserie,' %f',[1,100])" de Matlab. No obstante, los resultados obtenido no eran los adecuados debido a la complicada coordinación entre las capturas del puerto serie y los muestreos por el mismo, lo que daba lugar a una gran pérdida de valores enviados por el puerto serie. Además, la estructura con la que se extraían los valores no coincidía, generalmente, con la que se habían guardado ya que se grababa en un instante que no tenía por que coincidir con el inicio de muestreo por el puerto serie. Por estas razones y el descubrimiento del siguiente método se decidió dejar de trabajar con él.

3.2.4 Adquisición con la aplicación Coolterm

En principio, se hablará acerca de la aplicación. CoolTerm es una sencilla aplicación terminal de puerto serie que brinda la posibilidad de intercambiar datos con hardware conectado a puertos serie como servo controladores, kits robóticos, receptores GPS, microcontroladores, etc. Este software ha sido proporcionado por el programador Roger Meier [2] y en la bibliografía se puede encontrar el enlace donde poder descargar dicha aplicación, además de otros software gratuitos.

Tras haber comentado la aplicación que se usará, se pasa a explicar el siguiente esquema que muestra el procedimiento necesario para alcanzar los objetivos.

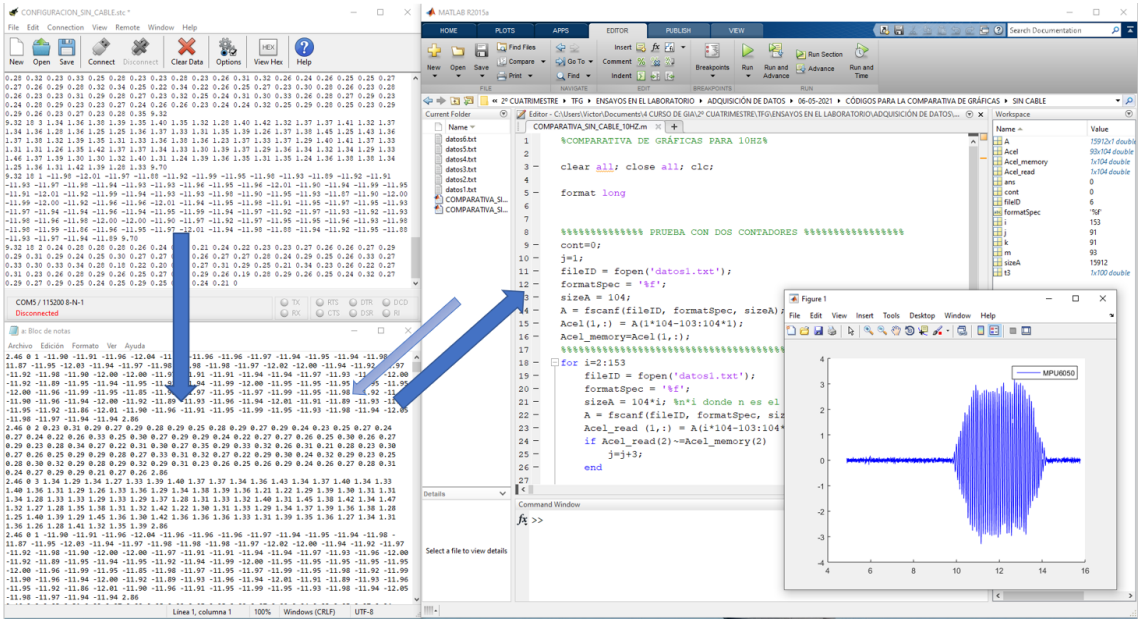


Figura 3.5 Esquema explicativo de la adquisición con el software + matlab.

Una vez se tienen cargados los códigos en ambos módulos, el módulo servidor **desconectado** y el cliente **conectado**, se conecta con la aplicación el módulo cliente.

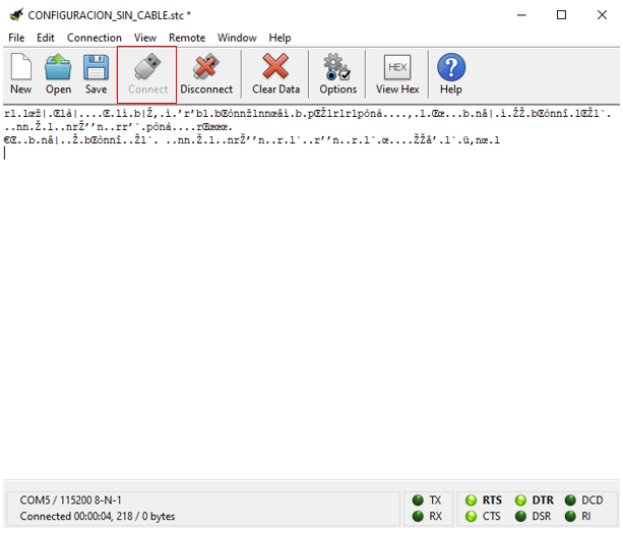


Figura 3.6 Conexión del módulo cliente con Coolterm.

A continuación, se configura la aplicación Coolterm. Para ello, como se muestra en la siguiente captura, se indica en las configuraciones cuál es el puerto por el que se transferirán los datos y los baudios de dicha transmisión por el puerto serie.

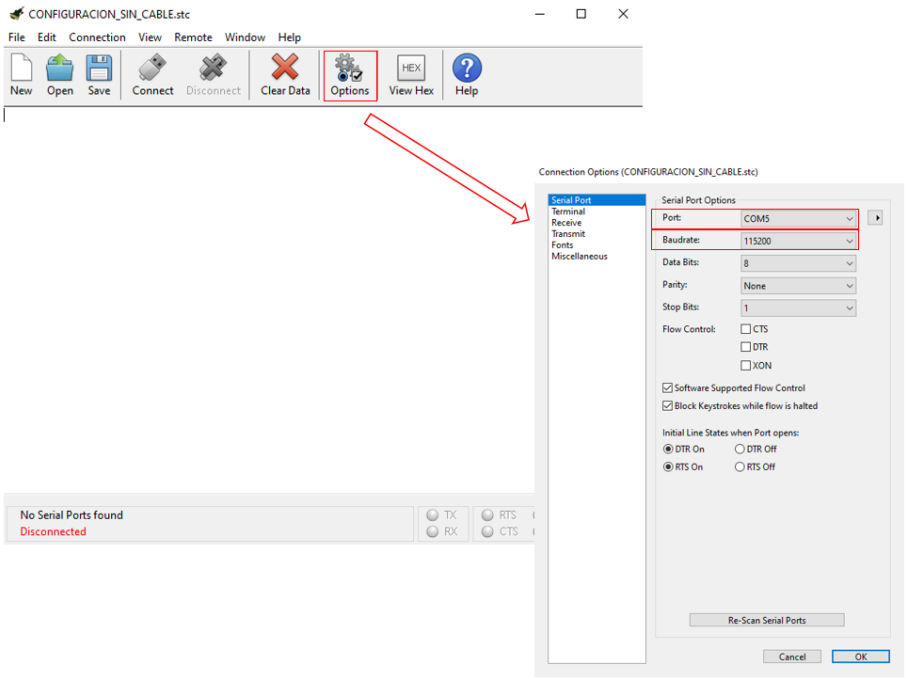


Figura 3.7 Configuraciones del Coolterm.

Se define ahora cuál va a ser el archivo .txt donde se quieren capturar las cadenas de las aceleraciones arrojadas por el servidor.

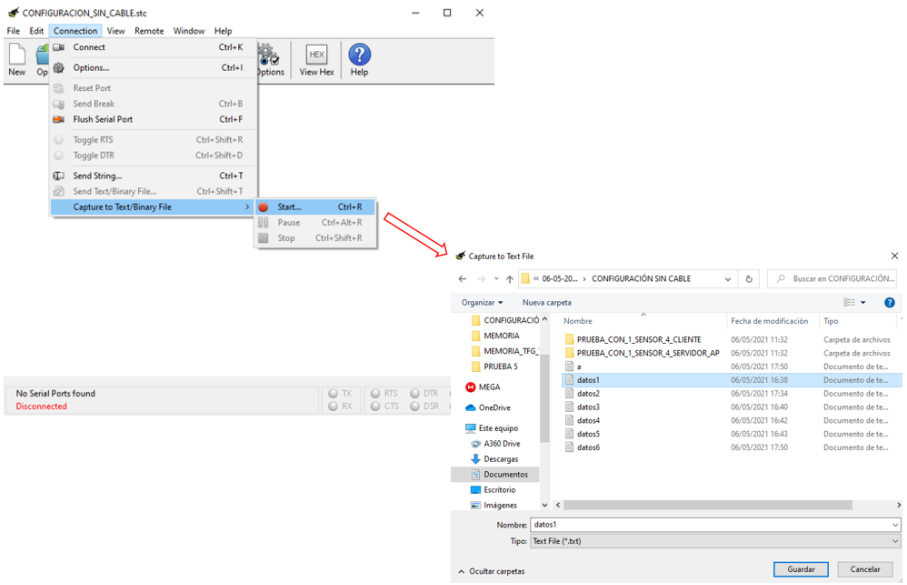


Figura 3.8 Selección del .txt donde queremos capturar los datos.

Seguidamente, se conecta el servidor a la alimentación. Una vez se ha establecido la conexión inalámbrica entre ambos, se empezarán a mostrar las cadenas con los valores de las aceleraciones. Cuando se desee terminar la transmisión, se desconectarán ambos módulos y se parará la captura que activamos en el .txt.

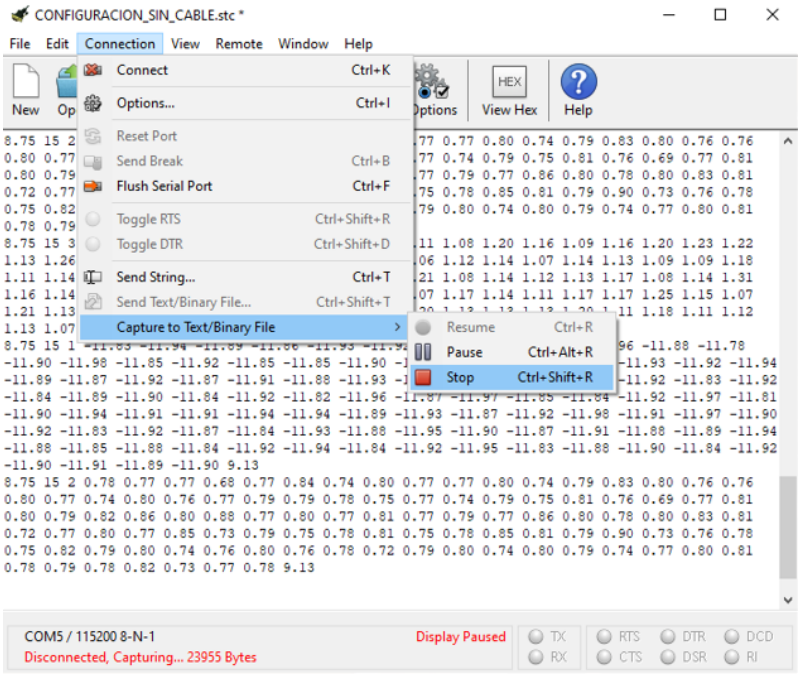


Figura 3.9 Fin de la captura en el .txt.

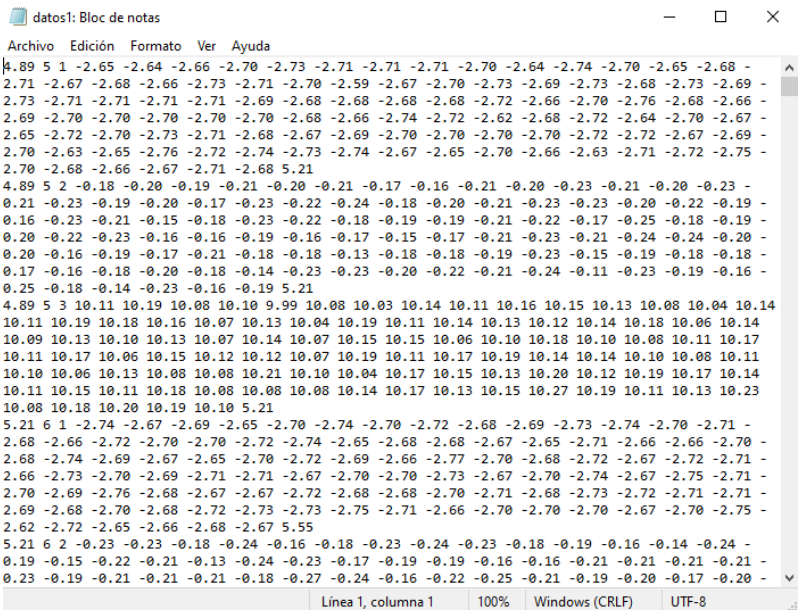


Figura 3.10 Valores guardados en el .txt.

Finalmente, una vez se disponen de los valores leídos, se hace uso de Matlab para hacer una correcta lectura del .txt. Se procede a continuación a detallar el código empleado.

En primer lugar, se debe recordar cuál era la estructura de cada cadena de aceleraciones almacenada, ya que, de esta manera, se podrá comprender mejor la lectura con Matlab.
Como se puede observar en la siguiente cadena mostrada, tanto la primera como la última componente van destinadas a definir cuál es el tiempo de inicio y fin de la creación de la cadena. A partir de ambos tiempos se podrá conocer de manera aproximada, bastante exacta, el instante de tiempo en el que se hace una determinada lectura en el acelerómetro. Y así, poder hacer una representación con respecto al tiempo de las aceleraciones, además de determinar la frecuencia de adquisición en la que se tiene tanto interés.
La segunda componente es un contador que indica en qué iteración ha sido creada la cadena. De esta forma se asegura que no se están desechando valores a lo largo del proceso de adquisición.
La tercera componente es un segundo contador que indica a qué eje corresponde la cadena: 1 corresponde con el eje x , 2 con el y y 3 con el z. Este segundo contador ha sido necesario introducirlo debido a que las peticiones y las creaciones de las cadenas no se hacen de manera sincronizada, como ya se ha comentado en apartados anteriores. Se detalla un poco más esto último: el servidor crea las cadenas de 100 componentes para cada eje de manera independiente y el cliente hace peticiones de estas cadenas cada X tiempo definido en el código. Puesto que el tiempo de petición y respuesta no es del todo estable, puede darse el caso en el que el cliente haga la primera petición, asociada al eje x, de la nueva iteración y al servidor no le haya dado tiempo de crear las cadenas de dicha nueva iteración. Con lo cual, la primera cadena será repetida de la iteración anterior y las dos siguientes son de la nueva iteración. De esta manera, con los dos contadores se pueden desechar, por un lado, las cadenas repetidas y ordenar, por otro, las cadenas de manera que siempre el orden sea el siguiente: eje x, y y z. Se debe aclarar que el tiempo entre peticiones ha sido elegido tanto para que no se pierdan cadenas, ya que si el tiempo de petición es demasiado grande al cliente no le da tiempo a hacer las peticiones antes de que el servidor actualice las cadenas, como para que el tiempo de petición no puede ser muy corto ya que si no, se harían demasiadas peticiones repetidas, empeorando así la adquisición de datos.
Por último, desde la componente 4 a la 103, ambas incluidas, se almacenan las 100 lecturas de aceleraciones, en el eje correspondiente de la cadena, realizadas al sensor.

Código 3.22 Una cadena de aceleraciones.

```
[4.89 5 1 -2.65 -2.64 -2.66 -2.70 -2.73 -2.71 -2.71 -2.71 -2.70 -2.64
   -2.74 -2.70 -2.65 -2.68 -2.71 -2.67 -2.68 -2.66 -2.73 -2.71 -2.70
   -2.59 -2.67 -2.70 -2.73 -2.69 -2.73 -2.68 -2.73 -2.69 -2.73 -2.71
   -2.71 -2.71 -2.71 -2.69 -2.68 -2.68 -2.68 -2.68 -2.72 -2.66 -2.70
   -2.76 -2.68 -2.66 -2.69 -2.70 -2.70 -2.70 -2.70 -2.70 -2.68 -2.66
   -2.74 -2.72 -2.62 -2.68 -2.72 -2.64 -2.70 -2.67 -2.65 -2.72 -2.70
   -2.73 -2.71 -2.68 -2.67 -2.69 -2.70 -2.70 -2.70 -2.70 -2.72 -2.72
   -2.67 -2.69 -2.70 -2.63 -2.65 -2.76 -2.72 -2.74 -2.73 -2.74 -2.67
   -2.65 -2.70 -2.66 -2.63 -2.71 -2.72 -2.75 -2.70 -2.68 -2.66 -2.67
   -2.71 -2.68 5.21]
```

Dicha cadena es real para la configuración que se está comentando. Con lo cual, se tiene una frecuencia de adquisición de unos 300 valores por segundo. Estando esta por encima de los requisitos iniciales.

A continuación, se procede a detallar el código de Matlab que permite leer el .txt en el que se han almacenado las cadenas. Dicho código no solo ha de leer el .txt sino que debe ser capaz de desechar las repetidas y ordenarlas como ya se ha comentado anteriormente.

Código 3.23 Lectura del .txt + gestión de datos con Matlab 1.

```
clear all; close all; clc

A = importdata('datos1.txt');
[i,~]=find(isnan(A));
if isempty(i)==0 %returns logical 1 if A is empty, and logical 0
    otherwise.
    datos = A(1:(i(1)-1),:);
else
    datos = A;
end
[n,~]=size(datos);
```

En estas líneas del código iniciales, lo primero que se hace es cargar el archivo que se especifica entres comillas. Seguidamente, se determina si la última fila, cadena, grabada en el .txt se ha rellenado completamente o si se ha cortado la transmisión antes. Con esto se puede eliminar la última fila, en caso de no estar rellena por completo, para evitar futuros problemas con las dimensiones de la matriz datos.

Código 3.24 Lectura del .txt + gestión de datos con Matlab 2.

```
%%%% DOS CONTADORES PARA ELIMINAR VALORES REPETIDOS %%%%
j=1;
Acel(1,:) = datos(1,:);
Acel_memory=Acel(1,:);
for i=2:n
    Acel_read (1,:) = datos(i,:);
    if Acel_read(2)~=Acel_memory(2)
        j=j+3;
    end
    if Acel_read(3)==1
        Acel(j,:)=Acel_read(1,:);
    elseif Acel_read(3)==2
        Acel(j+1,:)=Acel_read(1,:);
    else
        Acel(j+2,:)=Acel_read(1,:);
    end
    Acel_memory=Acel_read;
end
```

En esta segunda parte, haciendo uso de diferentes condiciones, se eliminan las cadenas repetidas y se ordenan estas según la siguiente secuencia: eje x, y y z.

Código 3.25 Lectura del .txt + gestión de datos con Matlab 3.

```
% REPRESENTACIÓN DE LOS TRES EJES %
```

```
figure
[m,n]=size(Acel);
acelx=[];
acely=[];
acelz=[];
t=[];
for k=1:3:(m-2)
    t1 = linspace(Acel(k,1),Acel(k,104),100);
    t = [t t1];
    acelx = [acelx Acel(k,4:103)];
    acely = [acely Acel(k+1,4:103)];
    acelz = [acelz Acel(k+2,4:103)];
end
plot(t,acelx,'r',t,acely,'g',t,acelz,'c')
xlabel('t(s)')
ylabel('a (m/s^2)')
title('Aceleraciones frente al tiempo')
legend('Acelx','Acely','Acelz')

% NOS CENTRAMOS EN EL EJE Z %
figure
plot(t,detrend(acelz),'c')
xlabel('t(s)')
ylabel('a (m/s^2)')
title('Aceleración en eje z frente al tiempo [detrend()]')
```

Para finalizar, se procede a plotear. En primer lugar, las aceleraciones en los tres ejes frente al tiempo. En segundo lugar, las amplitudes asociadas al eje z con la función **detrend()** de matlab ya que, como se comentará, las comparativas del sistema se harán excitando únicamente el eje z.

Se muestra en las siguientes imágenes las gráficas de salida de dicho código para el análisis almacenado en el archivo “datos1.txt".

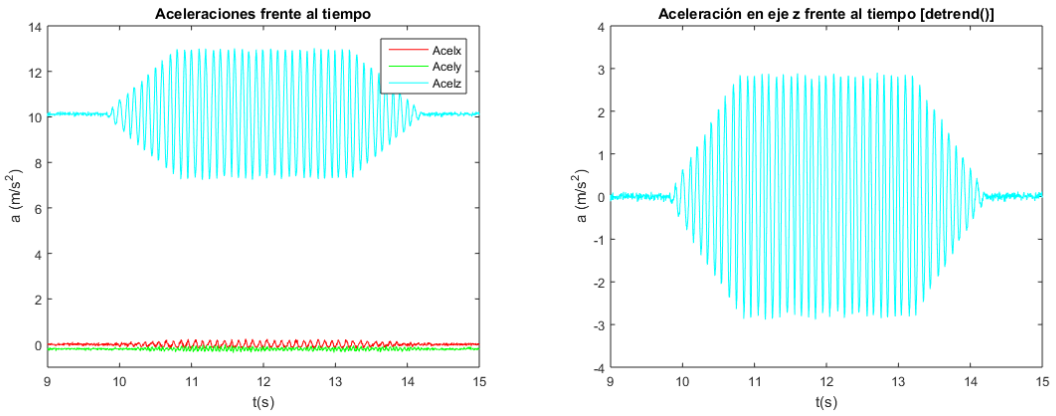


Figura 3.11 Representación de las aceleraciones en los tres ejes (Izq.). Representación de las amplitudes de las aceleraciones en el eje z (dcha.).

3.2.5 Conclusiones de la segunda configuración

Tras haberse probado diferentes configuraciones y métodos, se concluye con que esta segunda configuración basada en la comunicación inalámbrica de dos ESP8266 ha sido la que mejores resultados ha arrojado, cumpliendo con los dos requisitos impuestos desde el principio en el proyecto: frecuencia de adquisición y transmisión inalámbrica. Las prestaciones más notorias son las siguientes:

- Frecuencia de adquisición resultante de unos trecientos valores por segundo para cada eje. Estando esta por encima de los 200 valores por segundo que se tenía como requisito de partida.
- Transmisiones inalámbricas entre los dos módulos ESP8266 sin necesidad de disponer de acceso a internet, eliminando así las limitaciones encontradas en la primera configuración asociadas al complicado control temporal debido a: los protocolos de internet, la velocidad de internet de la que se disponga, respuesta del navegador frente a actualizaciones...
- Control total de las cadenas almacenadas, consiguiendo así no desechar valores de aceleraciones durante la transmisión.

Con estos últimos comentarios, se concluye este capítulo basado en el análisis de los diferentes tipos de configuración y métodos de adquisición.
El siguiente se centrará en la comparativa de la electrónica en su configuración definitiva con equipos profesionales en la materia con los que cuenta el departamento de Mecánica de Medios Continuos y Teoría de estructuras en los laboratorios de la escuela.

4 Comparativa del sistema con acelerómetro PCB PIEZOTRONICS 352C33

Este cuarto capítulo se centrará en la comparativa de la electrónica en su configuración definitiva, analizada en el capítulo anterior, con el acelerómetro PCB PIEZOTRONICS 352C33 [14]. Para poder realizar dicha comparación de los valores de aceleraciones facilitados por ambas electrónicas, se hará uso de una mesa excitadora, la APS 400 ELECTRO-SEIS [16].

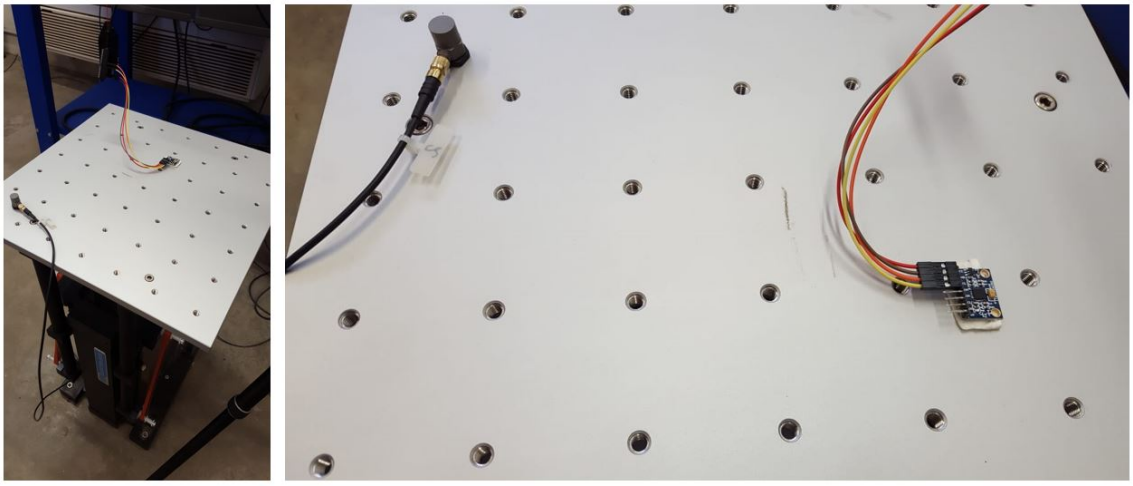


Figura 4.1 Fotos tomadas en los laboratorios de la escuela.

4.1 Acelerómetro PCB PIEZOTRONICS 352C33

PCB Piezotronics Inc. dispone de la más amplia gama de transductores para vibraciones del mercado. La base del éxito de PCB radica en el lanzamiento al mercado de la tecnología reconocida mundialmente como IPC (Integrated Circuit Piezoelectric). La inclusión de circuitos microelectrónicos en los propios sensores otorga a PCB un puesto privilegiado en el campo de la sensorización.

En cuanto al acelerómetro 352C33, este cumple con la especificaciones técnicas más exigentes, con un tamaño y masa reducido. Alguna de las especificaciones del sensor son las siguientes:

Tabla 4.1 Especificaciones del 352C33.

Especificación	Valor (S.I.)
Sensibilidad ($\pm 10\,\%$)	10.2 mV/(m/s^2)
Rango de frecuencia ($\pm 5\,\%$)	0.5 a 10000 Hz
Frecuencia de resonancia	$\geq$ 50 kHz

Para la adquisición de datos se ha usado el siguiente módulo de Brüel Kjaer, modelo 3160 [15].

Figura 4.2 Acelerómetro PCB PIEZOTRONICS 352C33 [14] y Módulo de adquisición de datos [15].

4.2 APS 400 ELECTRO-SEIS

El APS 400 ELECTRO-SEIS es un generador de fuerza electrodinámica de carrera larga, diseñado específicamente para usarse solo o en conjunto para estudiar las características de respuesta dinámica de varias estructuras. Se emplea en la excitación modal de estructuras complejas, particularmente, cuando se requieren bajas frecuencias. Además, se puede utilizar para ensayos de vibración de baja frecuencia de componentes y conjuntos.

El APS 400 ELECTRO-SEIS ha sido optimizado para llevar las estructuras a sus frecuencias naturales de resonancia. Es un electrodinámico generador de fuerza cuya salida es directamente proporcional al valor instantáneo de la corriente que se le aplica, independientemente de la frecuencia y respuesta de la carga. Puede excitar con señales random o transitorias de fuerza, como formas de onda sinusoidales, a la carga.

El excitador presenta diferentes modos de operación en función de su configuración física. En este proyecto se usará en la configuración de mesa excitadora.

Será necesario contar también con un amplificador, como el APS 124 - Power Amplifier, para alimentar al excitador.

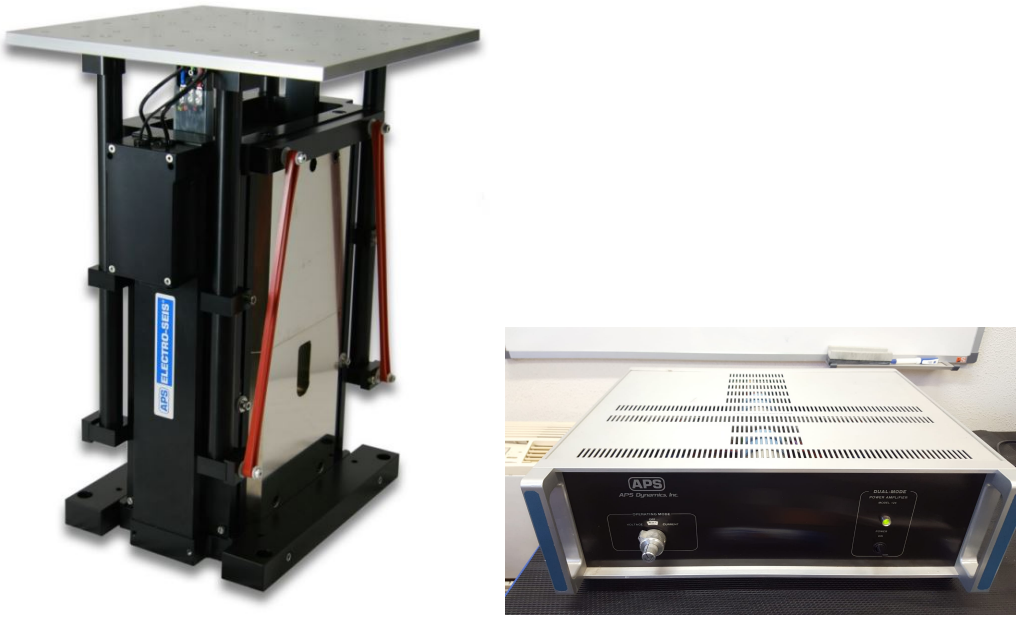


Figura 4.3 APS 400 ELECTRO-SEIS [16] y Amplificador APS 124 - Power Amplifier.

En la siguiente tabla se muestran algunas de las especificaciones del APS 400.

Tabla 4.2 Especificaciones del APS 400.

Specifications	Value
Fuerza máxima	445N
Velocidad máxima	1000 mm/s
Peso total del excitador	73.0 kg

4.3 Resultados

Una vez se han comentado los equipos empleados, se procede a presentar cuáles han sido los resultados obtenidos. En primer lugar, la comparativa se ha realizado con el sistema conectado por cable. Y en segundo lugar, con el sistema funcionando inalámbricamente, como se ha estudiado en este proyecto. De esta manera se podrá comparar, por un lado, cómo de acertados son los resultados con el sistema de bajo coste y, por otro, las limitaciones al hacer la transmisión inalámbrica con dicho sistema.

Antes de mostrar los resultados, en la siguiente tabla se detallan las características principales de las excitaciones ensayadas:

Tabla 4.3 Características principales de las excitaciones ensayadas.

Tipo de excitación	Intervalo frecuencial [Hz]	Resolución frecuencial [Hz]	Frecuencia [Hz]
Seno	100	0.5	10
Seno	100	0.5	20
Seno	100	0.5	30
Seno	100	0.5	40
Seno	100	0.5	50
Burst Random	50	0.0625	-

De la resolución frecuencial se puede calcular el tiempo de ensayo con la siguiente expresión:

$$\boxed{T_{\text{ensayo}} = \frac{1}{\text{Resolución frecuencial}}} \tag{4.1}$$

Por otro lado, los intervalos (períodos) de tiempo, δ_T, en los que se han registrado los datos han sido:

- Para los dos tipos de excitaciones con el 352C33 se tiene:
  - Seno: $\delta_T = 0.00195s \quad \Rightarrow \quad f_s \sim 500Hz$.
  - Burst Random: $\delta_T = 0.00195s \quad \Rightarrow \quad f_s \sim 500Hz$.
- Para ambas configuraciones con el MPU6050:
  - Con cable: $\delta_T \sim 0.0022s \quad \Rightarrow \quad f_s \sim 455Hz$.
  - Inalámbrico: $\delta_T \sim 0.0035s \quad \Rightarrow \quad f_s \sim 285Hz$.

donde f_s es la frecuencia de muestreo, número de datos por segundo registrados por el sistema. Dichas frecuencias son mayores para el sensor 352C33. No obstante, para la configuración con cable ambos sensores presentan frecuencias bastante cercanas. Y, como era de esperar, con la transmisión inalámbrica se tiene la frecuencia de muestreo más pequeña, frecuencia que condicionará la comparativa.

En cuanto a las sensibilidades de los sensores, la del PCB PIEZOTRONICS 352C33 es de 100.2 mV/g. Para el caso del MPU6050 se tiene una sensibilidad máxima de 16384 LSB / g. Esto equivale, haciendo uso de la siguiente fórmula, a 825 mV/g en una escala analógica [18].

$$\text{LSB} = \frac{\text{voltaje operación}}{2^{N^o bits ADC}} \tag{4.2}$$

Donde el voltaje de operación del MPU6050 es de 3.3 V y la escala ADC es de 16 bits. Por lo tanto, cada unidad de LSB equivale a 0.0504 mV y para los 16384 LSB/g se tienen los 825 mV/g.

Debe resaltarse que aunque la sensibilidad para el MPU6050 sea superior, esta es una sensibilidad teórica digital mientras que para el caso del PCB es una sensibilidad calibrada.

Para llevar a cabo la comparativa se han representado las aceleraciones en el dominio del tiempo y en el de la frecuencia. Con este último se detecta de manera más precisa cómo de parecidas son ambas señales, al disponer de los contenidos en frecuencia. Como se puede observar en la siguiente imagen, el objetivo del análisis en el dominio de la frecuencia es el de descomponer la señal compleja arrojada por los sensores en otras señales periódicas básicas que permiten un estudio más asequible.

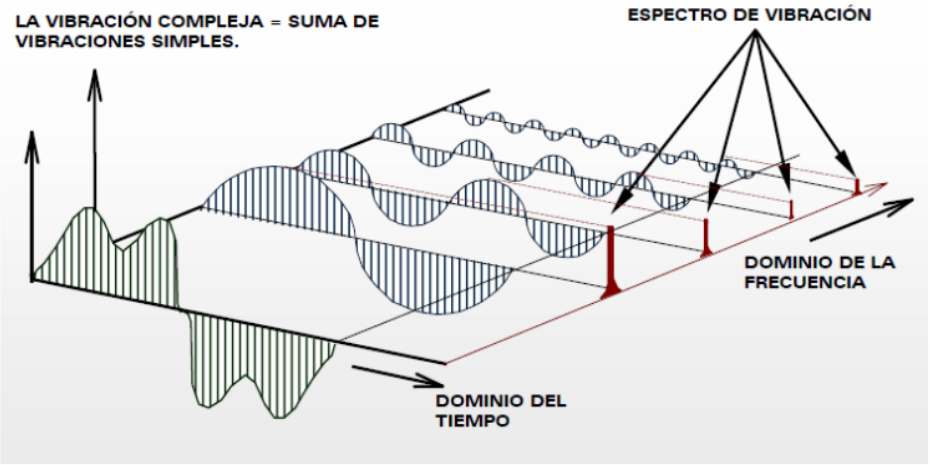


Figura 4.4 Señal en el dominio del tiempo y de la frecuencia [19].

La representación obtenida en el dominio de la frecuencia se le denomina espectro de frecuencias.

Como se ha comentado anteriormente, en este caso se excitará principalmente con señales tipo seno en un rango de frecuencia entre 10 y 50 Hz. Con lo cual, se conoce previamente cual deberá ser el contenido en frecuencia de las señales mostradas por los sensores (ver figura).

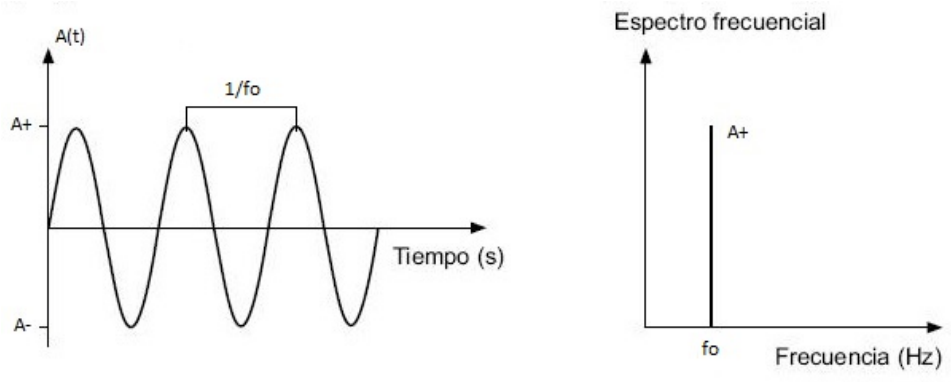


Figura 4.5 Espectro frecuencial de una señal periódica.

4.3.1 Comparativa excitación tipo seno de 10Hz [Con cable]

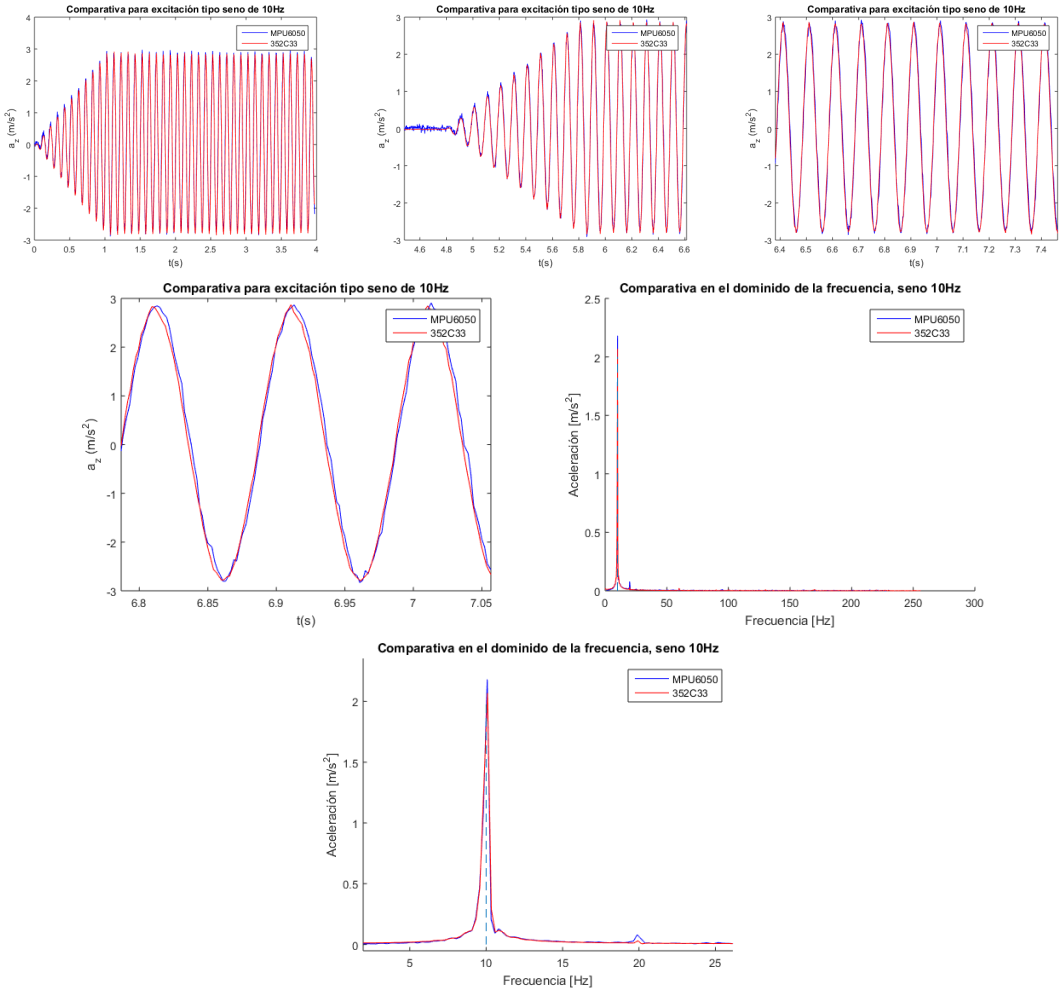
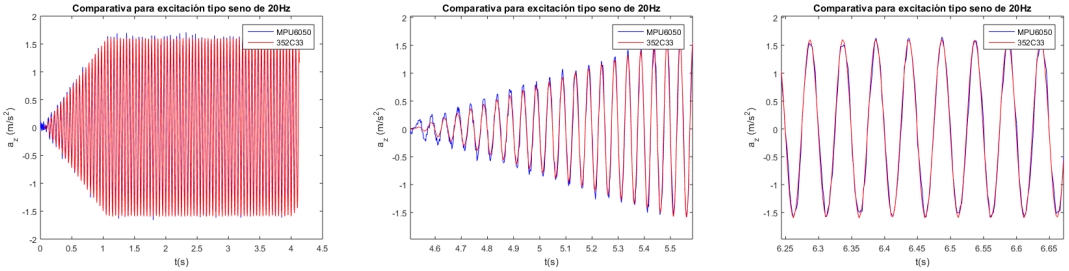


Figura 4.6 Comparativa para excitación tipo seno de 10Hz [Con cable].

Para la frecuencia de 10Hz las señales obtenidas por ambos sensores se asemejan bastante, tanto en el dominio del tiempo como en el de la frecuencia. Ambos armónicos se encuentran centrados en la frecuencia de excitación, 10Hz. Comentar que la amplitud en el dominio de la frecuencia no coincide exactamente con la del seno debido a la variación de amplitud en la fase transitoria.

4.3.2 Comparativa excitación tipo seno de 20Hz [Con cable]



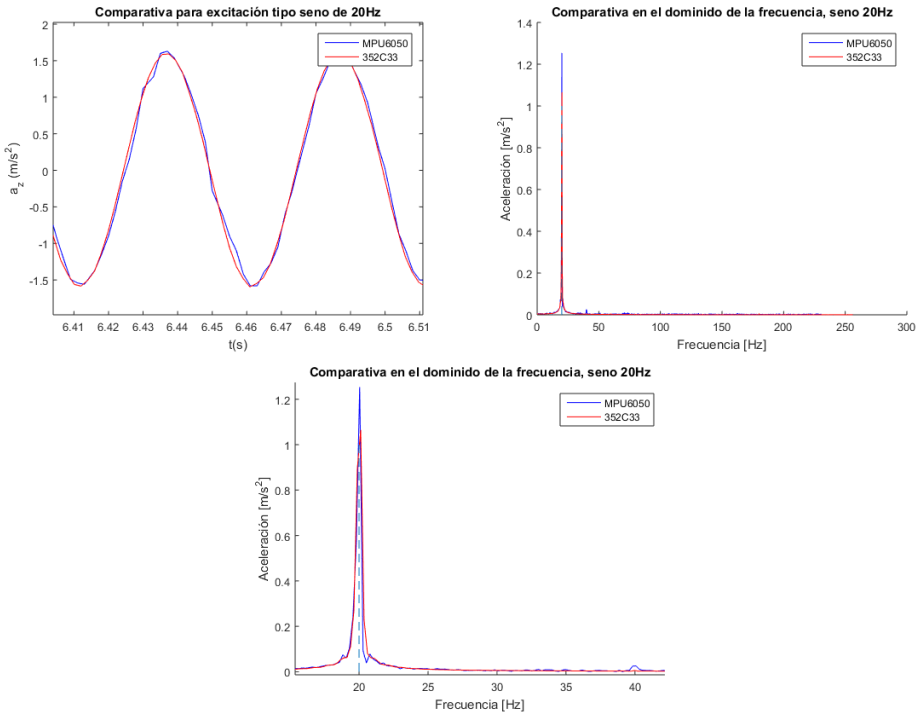
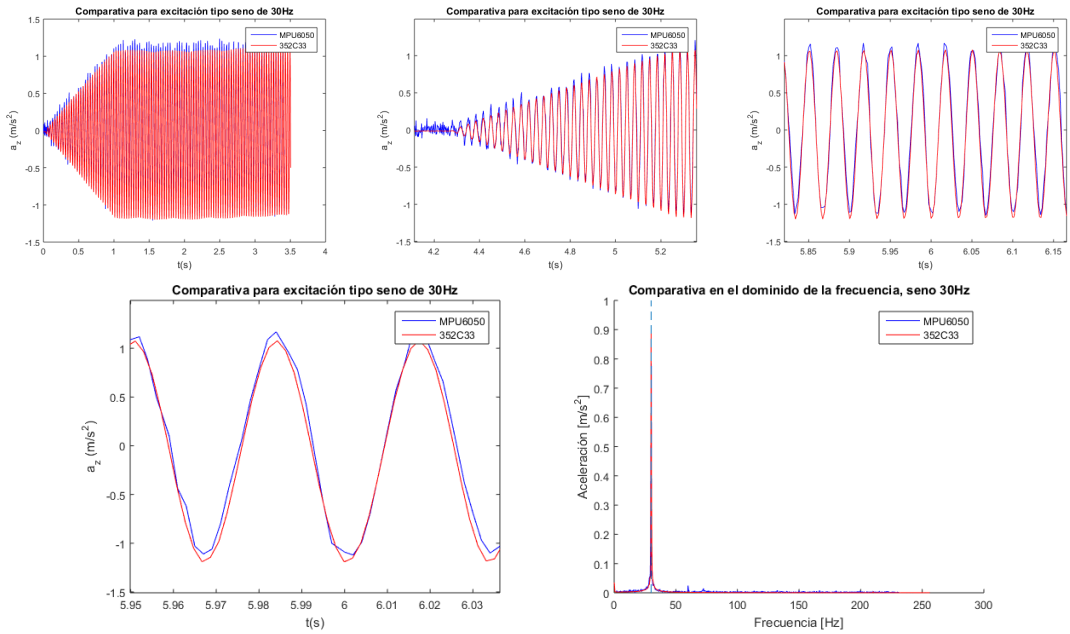


Figura 4.7 Comparativa para excitación tipo seno de 20Hz [Con cable].

Para este segundo ensayo, los resultados obtenidos siguen el mismo comportamiento que para el primero: espectros centrados en la frecuencia de excitación, 20Hz, y amplitudes ligeramente inferiores por el tramo transitorio.

4.3.3 Comparativa excitación tipo seno de 30Hz [Con cable]



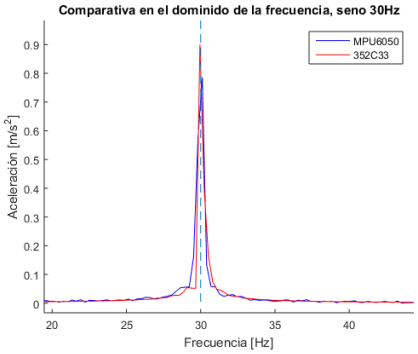


Figura 4.8 Comparativa para excitación tipo seno de 30Hz [Con cable].

En el caso de 30Hz el sistema sigue comportándose bastante bien, arrojando resultados muy semejantes a los del sensor PCB PIEZOTRONICS 352C33.

4.3.4 Comparativa excitación tipo seno de 40Hz [Con cable]

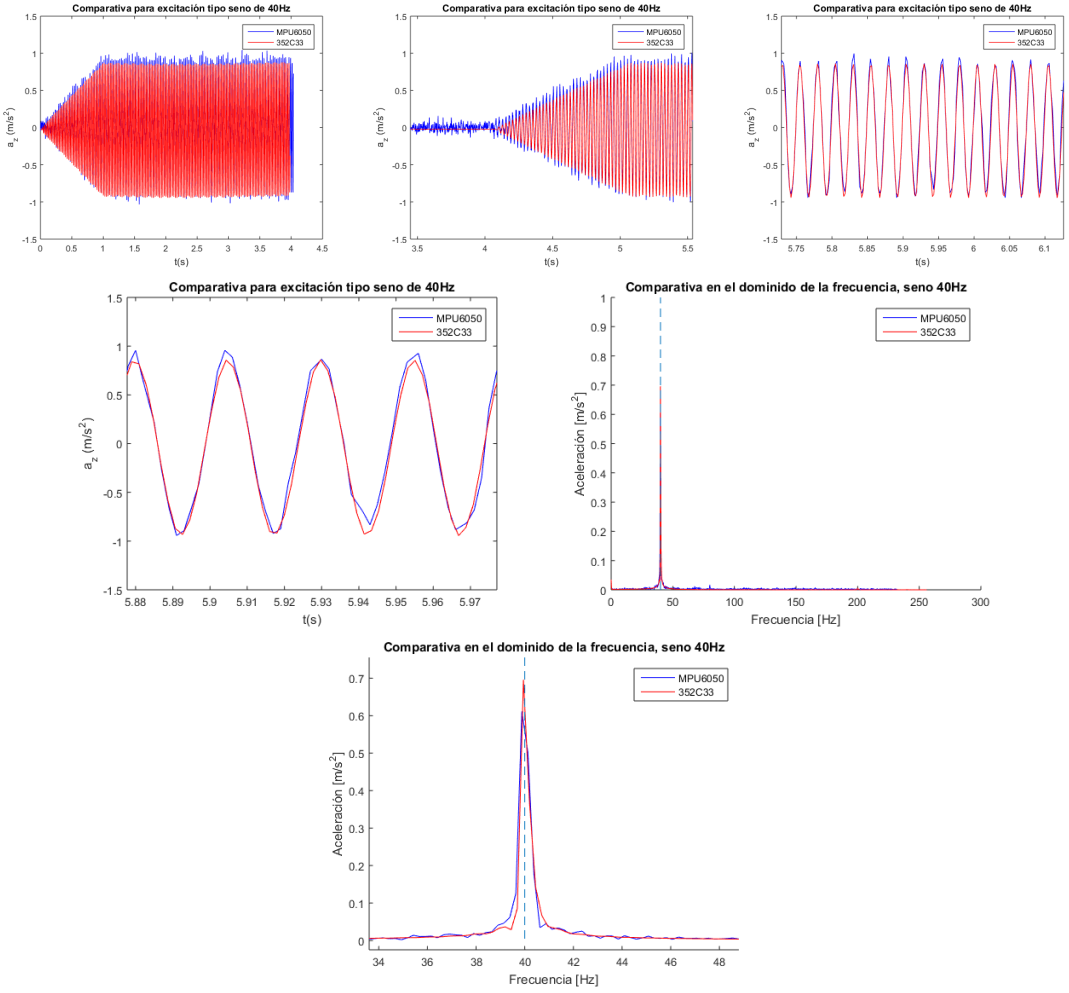


Figura 4.9 Comparativa para excitación tipo seno de 40Hz [Con cable].

Resultados similares a los casos anteriores, buen comportamiento del sistema.

4.3.5 Comparativa excitación tipo seno de 50Hz [Con cable]

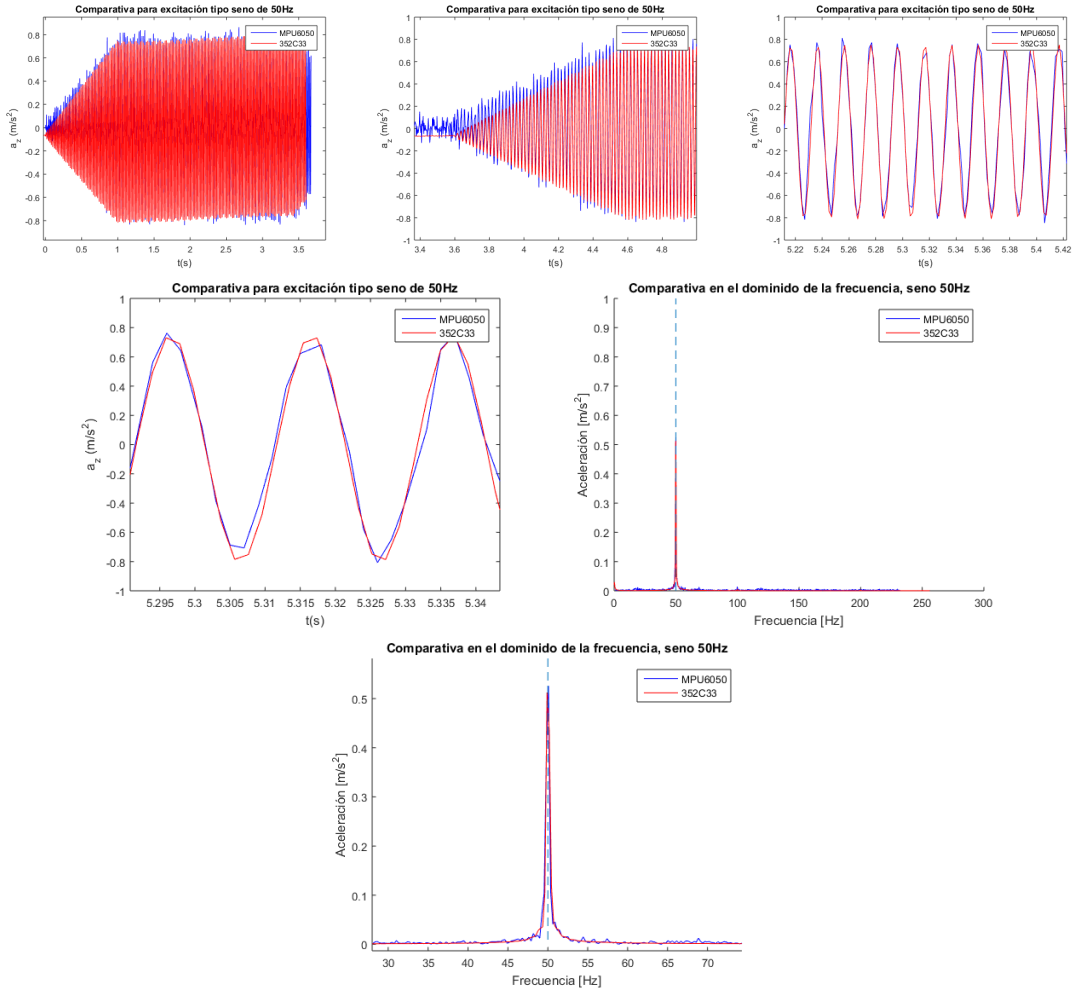
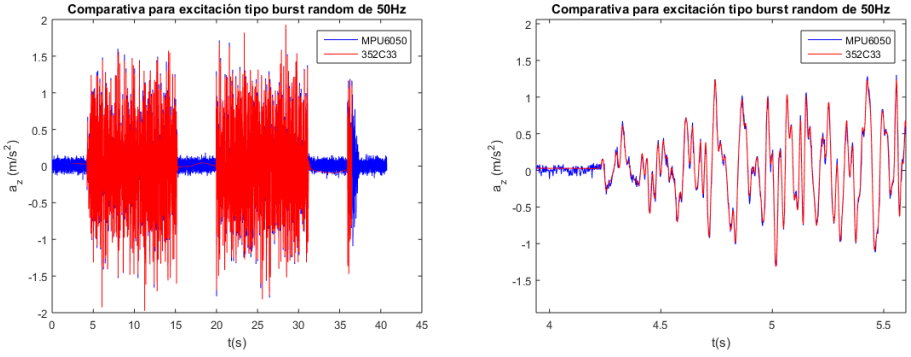


Figura 4.10 Comparativa para excitación tipo seno de 50Hz [Con cable].

Para la excitación tipo seno de 50Hz se observa una cierta anomalía en el dominio del tiempo. No obstante, dicha anomalía corresponde a la señal arrojada por el sensor del laboratorio al no encontrarse centrada en cero. Finalmente se ha decidido incluirla pues, como se puede apreciar, en permanente como en el espectro de frecuencias ambas señales son bastante similares.

4.3.6 Comparativa excitación tipo burst random de 50Hz [Con cable]



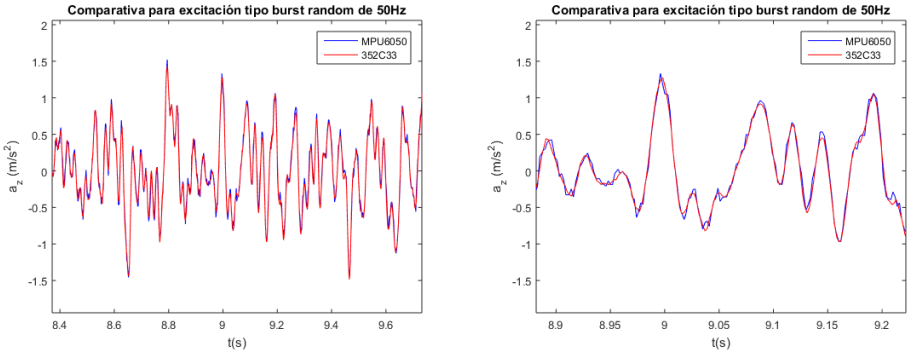


Figura 4.11 Comparativa para excitación tipo burst random de 50Hz [Con cable].

En este último ensayo únicamente se han representado las señales frente al tiempo, siendo estas muy similares. Se ha omitido el espectro de frecuencias pues al ser una señal tipo burst random el contenido en frecuencia está bastante saturado y se dificulta mucho la comparativa.

4.3.7 Comparativa excitación tipo seno de 10Hz [Inalámbrico]

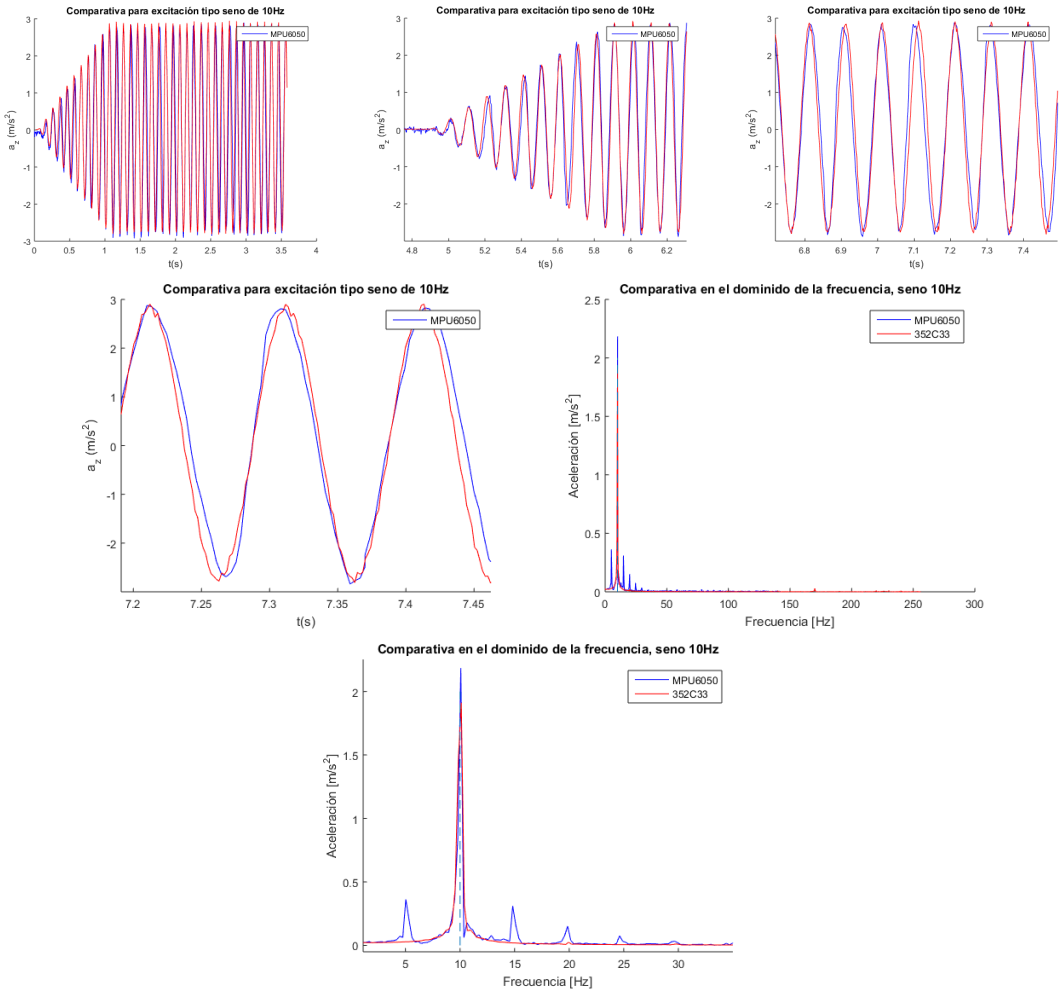


Figura 4.12 Comparativa para excitación tipo seno de 10Hz [Inalámbrico].

Como se puede observar, en la configuración inalámbrica la señal no es un seno puro sino que

presenta un ruido de unos 5Hz, teniéndose contenidos en 5, 10, 15, 20 y 25 Hz. No obstante, para esta frecuencia de 10 Hz el resto de armónicos presentan unas amplitudes bastante menores.

4.3.8 Comparativa excitación tipo seno de 20Hz [Inalámbrico]

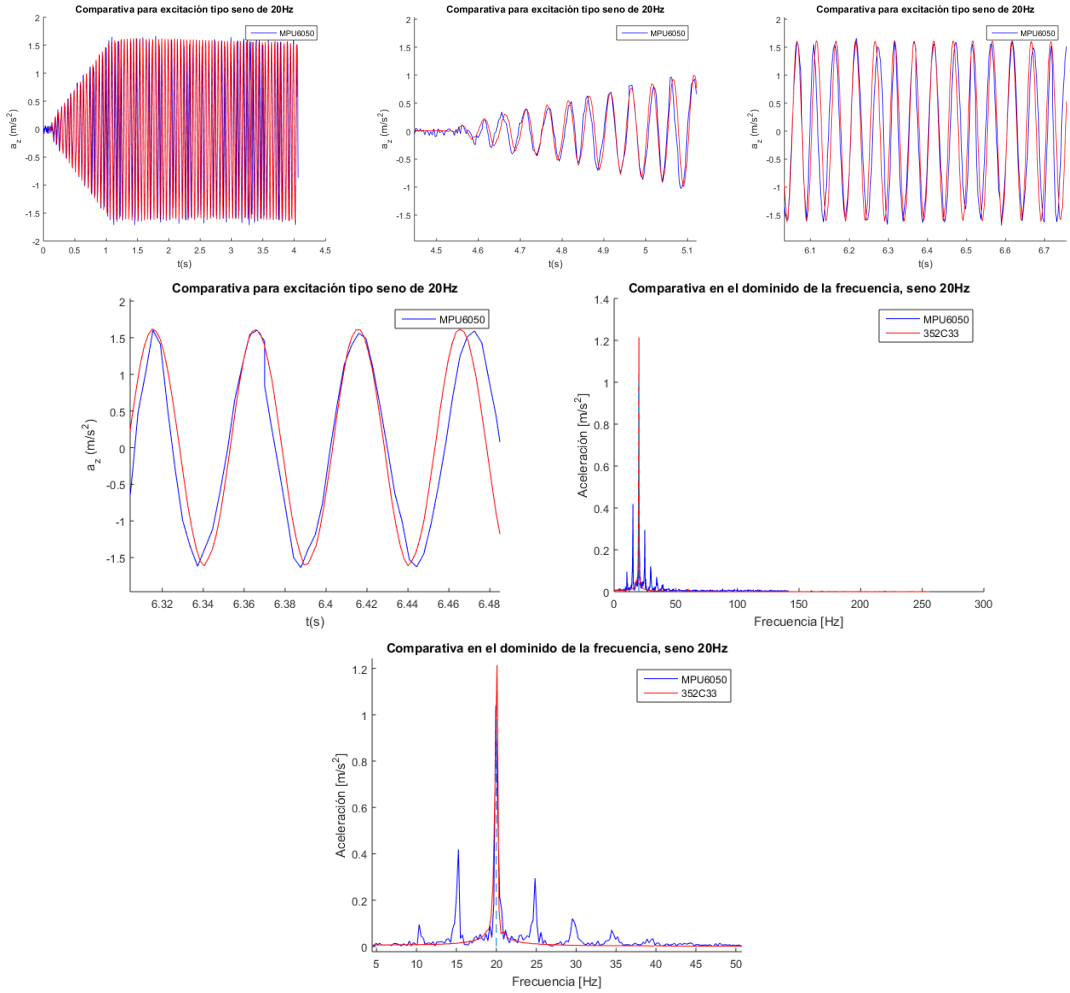
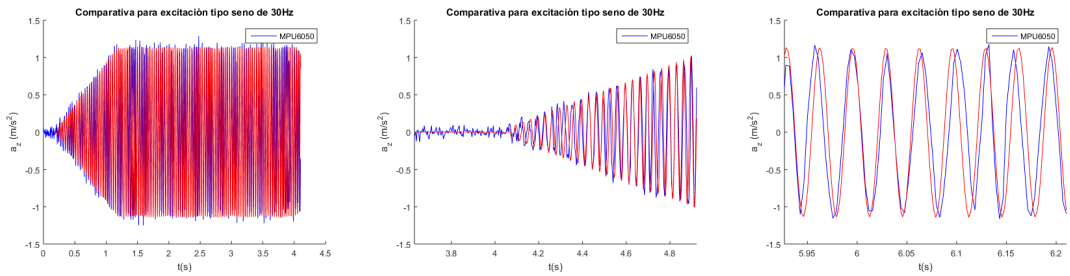


Figura 4.13 Comparativa para excitación tipo seno de 20Hz [Inalámbrico].

En la segunda excitación con la configuración inalámbrica se tienen comportamientos similares al primer ensayo. La señal tiene fundamentalmente un contenido en la frecuencia de excitación y como antes, unos armónicos con menor potencia separados en 5Hz.

4.3.9 Comparativa excitación tipo seno de 30Hz [Inalámbrico]



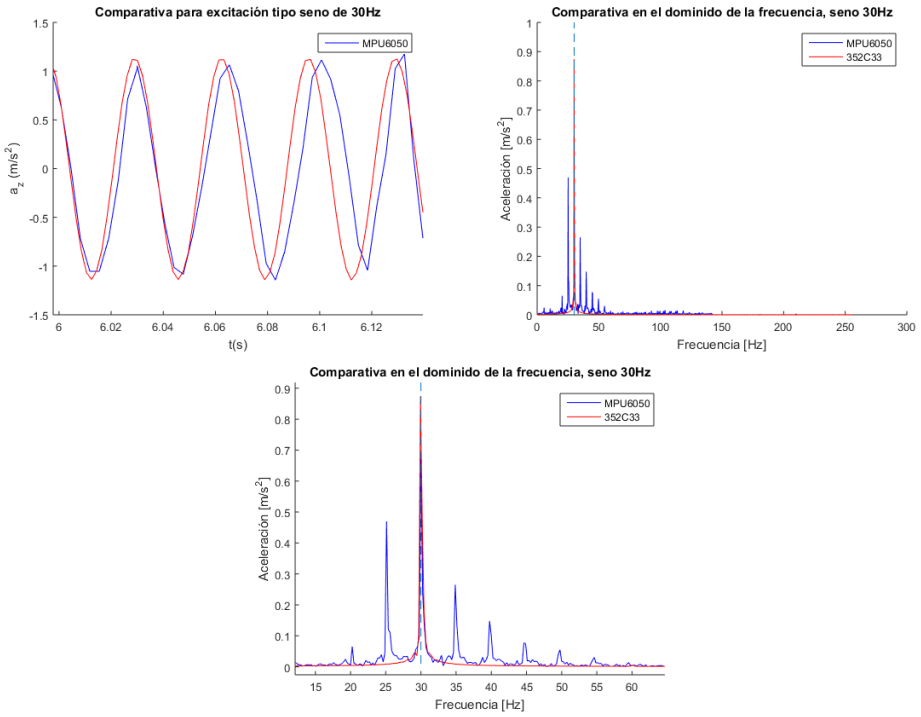
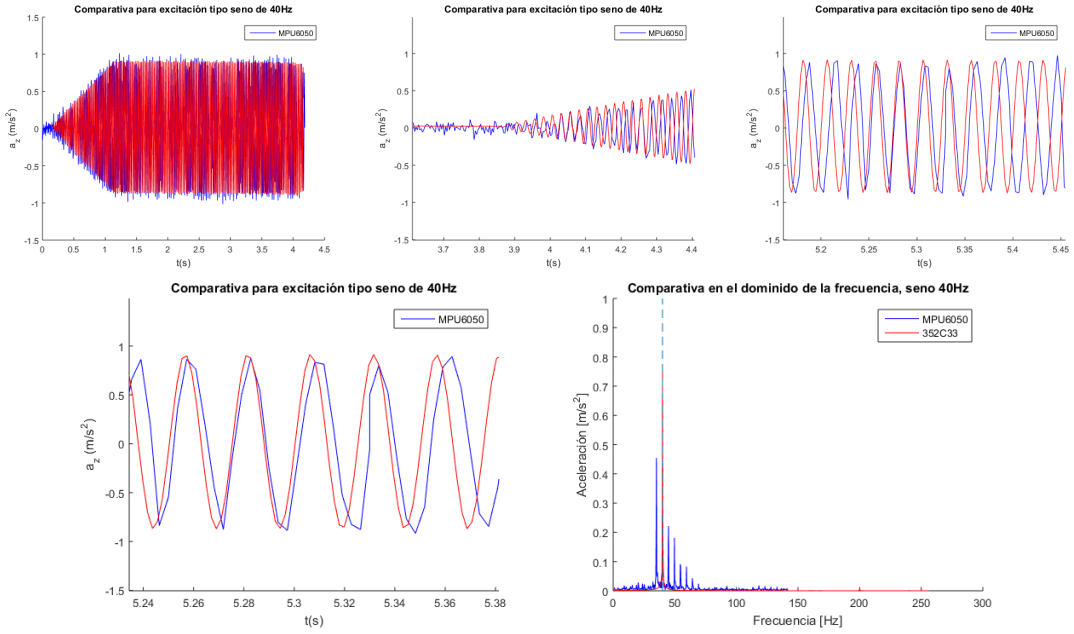


Figura 4.14 Comparativa para excitación tipo seno de 30Hz [Inalámbrico].

Al ir aumentando la frecuencia de excitación, los armónicos no centrados comienzan a presentar amplitudes similares al centrado.

4.3.10 Comparativa excitación tipo seno de 40Hz [Inalámbrico]



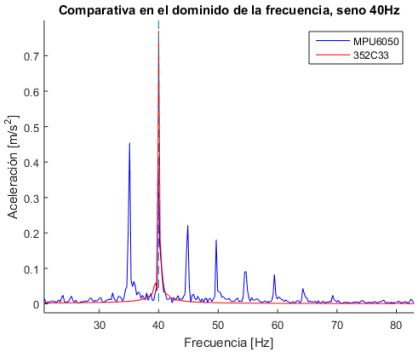


Figura 4.15 Comparativa para excitación tipo seno de 40Hz [Inalámbrico].

Para esta frecuencia de 40Hz el armónico de 35Hz presenta una amplitud superior a la que corresponde a la frecuencia de excitación.

4.3.11 Comparativa excitación tipo seno de 50Hz [Inalámbrico]

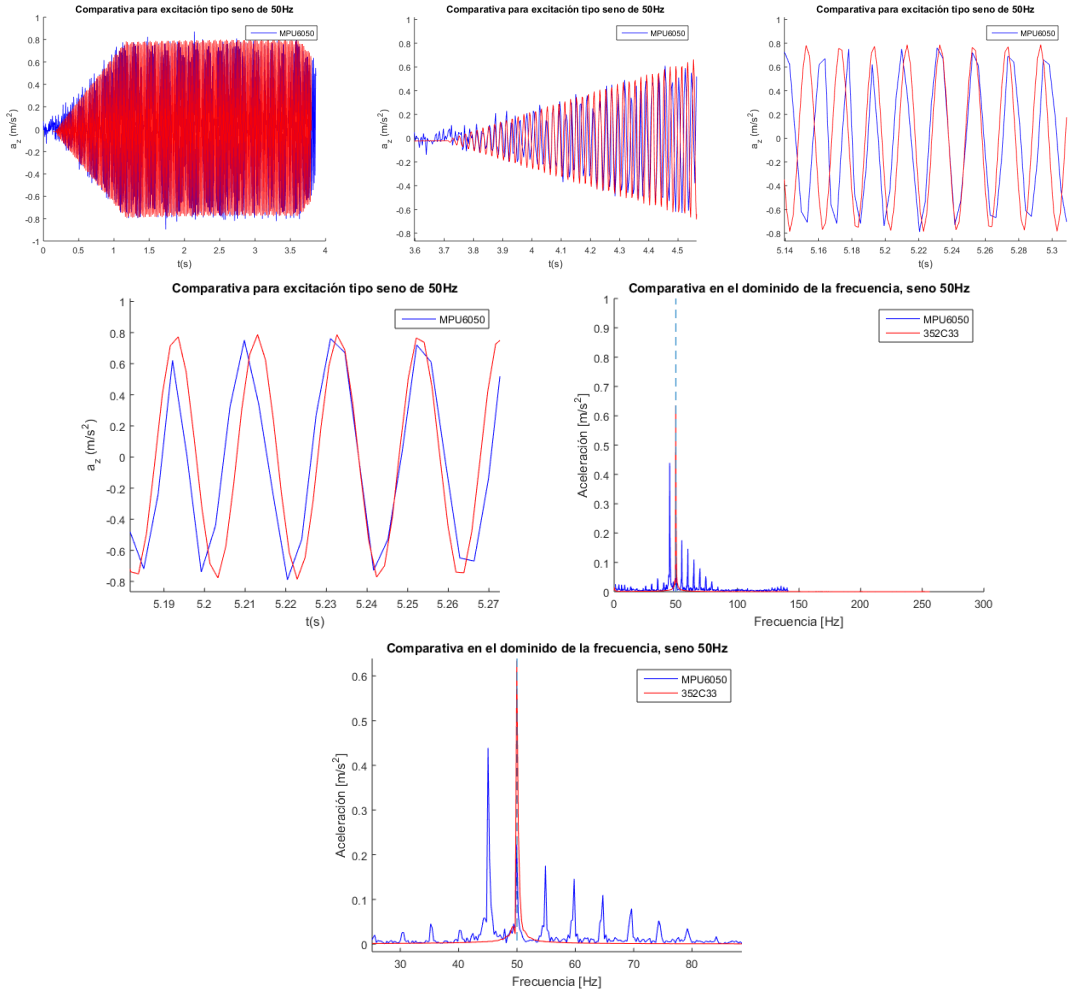


Figura 4.16 Comparativa para excitación tipo seno de 50Hz [Inalámbrico].

Por último, para una excitación de 50Hz el comportamiento empeora bastante. Por un lado la amplitud máxima no se encuentra centrada en la frecuencia de excitación y, por otro, el número de

armónicos que representan en el espectro de frecuencias la señal arrojada es mayor. Como se ha podido apreciar, se tiene la necesidad de incorporar filtros que permitan desechar los armónicos asociados a la alteración de la señal tipo seno puro por la transmisión inalámbrica.

4.3.12 Comparativa excitación tipo burst random de 50Hz [Inalámbrico]

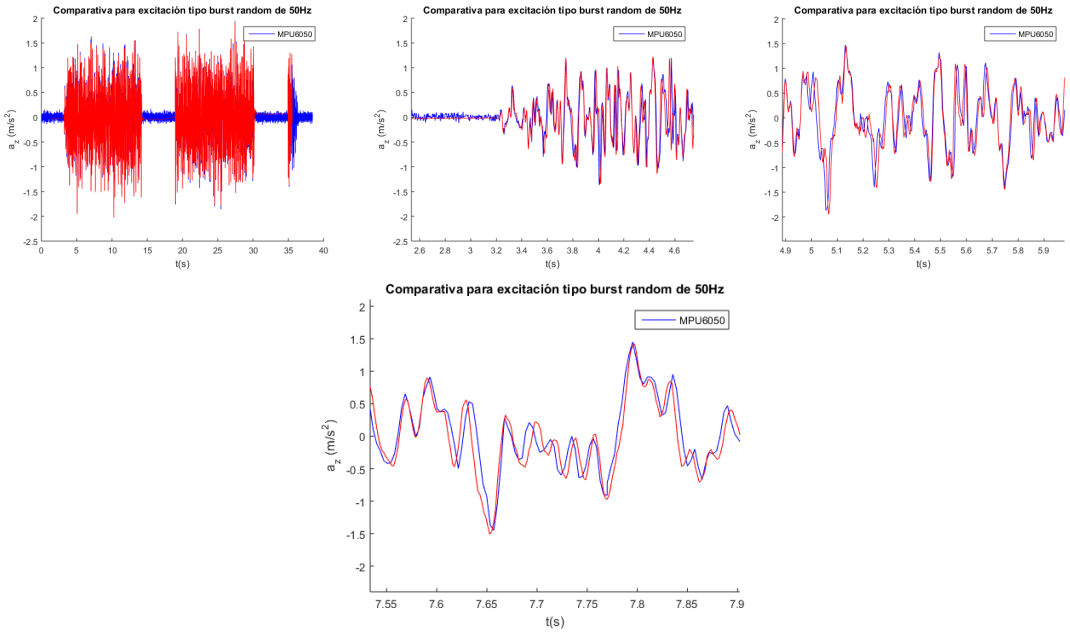


Figura 4.17 Comparativa para excitación tipo burst random de 50Hz [Inalámbrico].

Al igual que para la configuración con cable, únicamente se han representado las señales frente al tiempo, siendo estas bastante similares teniéndose en cuenta el tipo de excitación.

4.3.13 Filtrado de las señales arrojadas por el MPU6050

Como se ha observado en los resultados mostrados para la configuración inalámbrica, la señal facilitada por el sensor MPU6050 ha de ser filtrada. Para realizar dicha tarea se ha aplicado el filtro de Chebyshev haciendo uso de las siguientes funciones de Matlab: "[b,a] = cheby1(n,Rp,Wp,ftype)" y "y = filter(b,a,x)" que generan el numerador y denominador de la función de transferencia asociado al filtro y se la aplican a la señal x, siendo y la señal filtrada (ver código).

En un primer momento, se aplicaron dos filtros, uno pasa bajos y otro pasa altos, cuyas características se encuentran recogidas en la siguiente tabla.

Tabla 4.4 Características de los filtros implementados.

	Filtro pasa bajos			Filtro pasa altos		
F. excitación [Hz]	Orden del filtro	Rizado	Frecuencia de corte [Hz]	Orden del filtro	Rizado	Frecuencia de corte [Hz]
10 -50	3	0.1	100	11	0.1	8

En estas gráficas se observan las respuestas en frecuencia de los dos filtros:

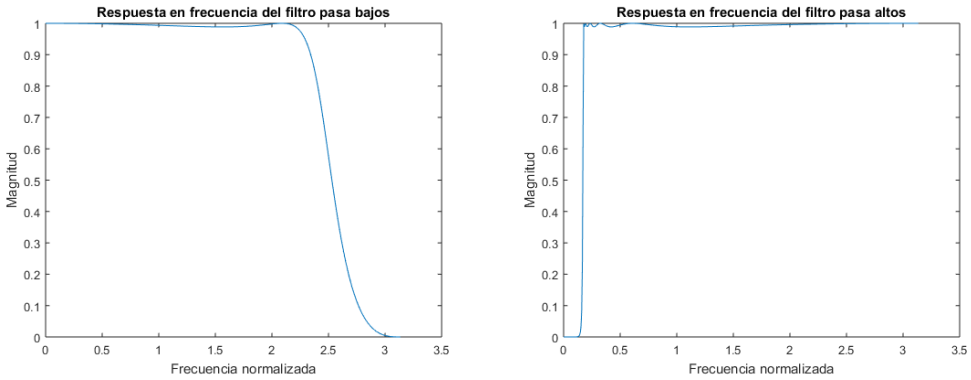
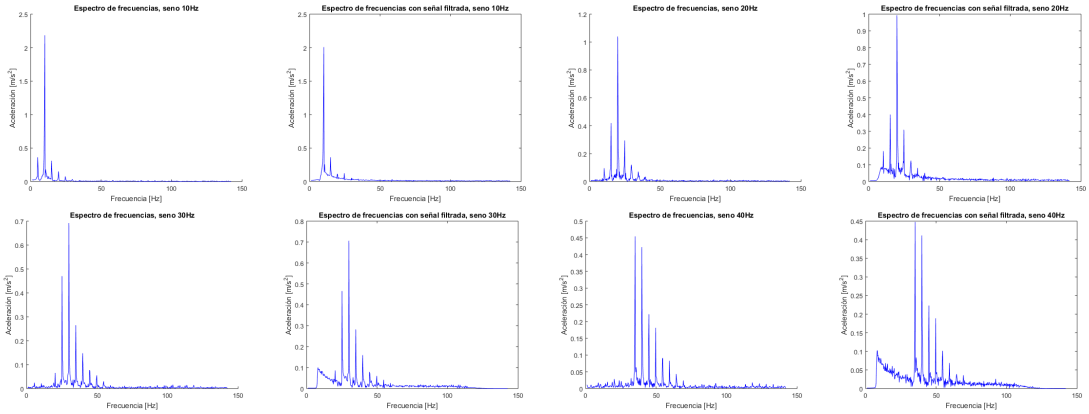


Figura 4.18 Respuestas en frecuencia de los filtros pasa bajos y pasa altos.

El objetivo de la implementación de ambos filtros era el de atenuar el ruido de 5Hz que presentan las señales arrojadas por el MPU6050. No obstante, como se puede observar, los cambios logrados no fueron satisfactorios.



Tras los resultados obtenidos para los filtros anteriormente comentados, se decidió aplicar filtros con diferentes frecuencias de corte según la frecuencia de excitación, logrando así eliminar los armónicos no deseados. No obstante, se debe tener presente que con esta segunda forma se está eliminando gran parte señal.
En la siguiente tabla se recogen las características de los diferentes filtros empleados según la frecuencia de excitación.

Tabla 4.5 Características de los filtros implementados.

	Filtro pasa bajos			Filtro pasa altos		
F. excitación [Hz]	Orden del filtro	Rizado	Frecuencia de corte [Hz]	Orden del filtro	Rizado	Frecuencia de corte [Hz]
10	6	1	10.3	6	1	9.7
20	6	1	20.3	6	1	19.7
30	8	1	30.1	8	1	29.9
40	9	1	40.1	9	1	39.9
50	10	1	50.1	10	1	49.9

Como se puede apreciar en las siguientes gráficas, se han eliminado aquellos armónicos asociados a la perturbación del seno de excitación por la transmisión inalámbrica. Para frecuencias de excitación inferiores a 30Hz se obtienen señales bastante similares para ambos sensores. No obstante, aunque para 40 y 50 Hz los contenidos en frecuencia siguen estando centrados, estos presentan una amplitud bastante inferior, como se ha comentado en la comparativa anterior.

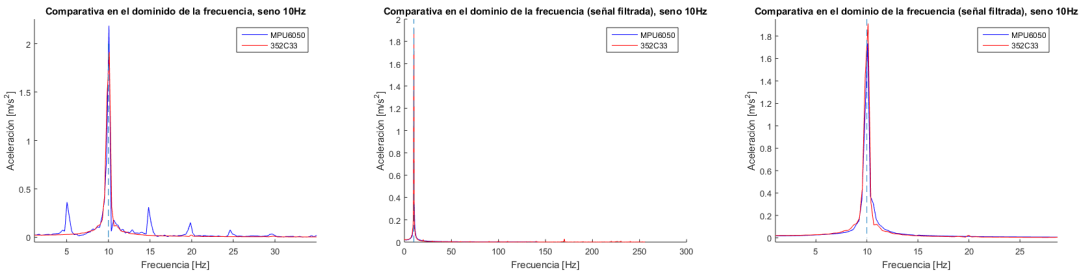


Figura 4.19 Comparativa para excitación tipo seno de 10Hz con señal filtrada[Inalámbrico].

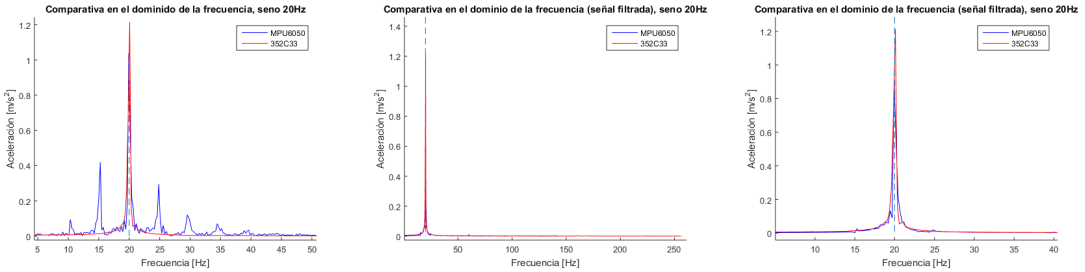


Figura 4.20 Comparativa para excitación tipo seno de 20Hz con señal filtrada[Inalámbrico].

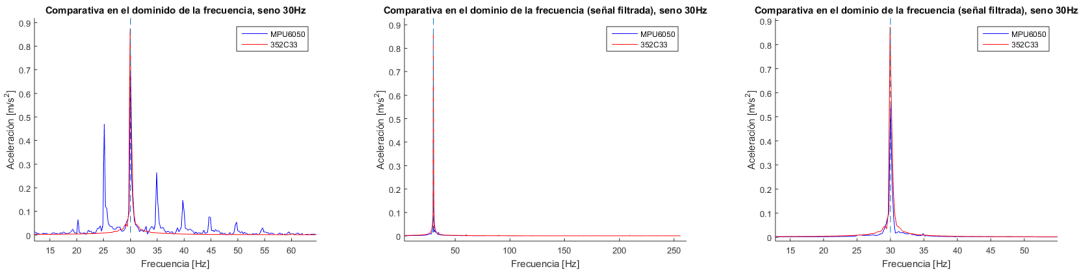


Figura 4.21 Comparativa para excitación tipo seno de 30Hz con señal filtrada[Inalámbrico].

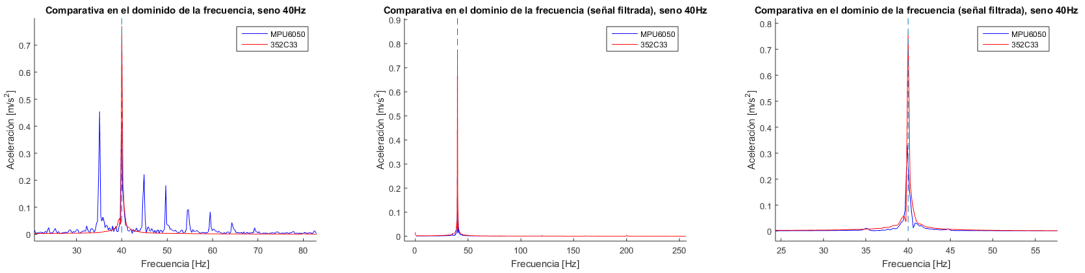


Figura 4.22 Comparativa para excitación tipo seno de 40Hz con señal filtrada[Inalámbrico].

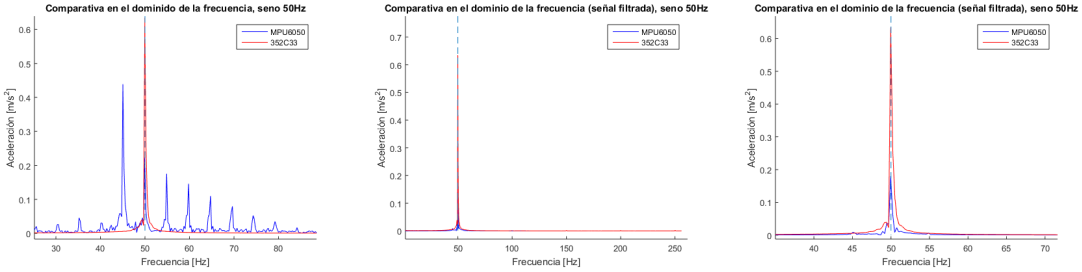


Figura 4.23 Comparativa para excitación tipo seno de 50Hz con señal filtrada[Inalámbrico].

4.4 Conclusiones de los resultados obtenidos

En primer lugar, en cuanto a la configuración con cable, los resultados arrojados por el sistema de bajo coste se asemejan bastante a los facilitados por los equipos del laboratorio. Únicamente hay una anomalía en la excitación tipo seno de 50Hz que, como se comentó, se produce en la respuesta del PCB 352C33, ya que no se encuentra centrada en cero. En cuanto a las representaciones en el dominio de la frecuencia, se observa un único armónico en la frecuencia correspondiente al seno de excitación. Las amplitudes en el espectro de frecuencia no coinciden con la amplitud del seno en permanente, esto se debe, como ya se comentó, al tramo transitorio de la excitación. Teniéndose todo esto presente, se puede concluir que el sistema de bajo coste conectado por cable ofrece unos resultados bastante buenos para estos tipos de excitación.

En segundo lugar, en lo que respecta a la configuración objeto de estudio, se puede observar como las medidas para una frecuencia superior a 20Hz empiezan a diferir de las arrojadas por el PCB 352C33. Esto se observa perfectamente en el dominio de la frecuencia con la incorporación de armónicos con cada vez más amplitud a frecuencias diferentes a la de la excitación, armónicos que han sido filtrados posteriormente. Para el caso de 50Hz el armónico con mayor amplitud no se encuentra centrado en la frecuencia de excitación. No obstante, siempre que la aplicación en la que se emplee no exponga al sensor a altas frecuencias, los resultados obtenidos se pueden considerar bastante aceptables teniéndose en cuenta el bajo coste de los equipos y la comunicación inalámbrica.

¿Por qué se produce esta diferencia entre la transmisión con cable y la inalámbrica? La causa principal radica en las lecturas realizadas al MPU6050 en ambas configuraciones. En la realizada con cable las lecturas del sensor se hacen de forma ininterrumpida, mientras que en la inalámbrica, como ya se ha comentado con mayor detalle, ha sido necesario introducir unos delays de unos 2ms en los códigos con el objetivo de sincronizar las peticiones y solicitudes entre módulos. En ese corto periodo de tiempo no se están leyendo los valores medidos por el acelerómetro.
Como se observa en los gráficos y es lógico, el comportamiento de los resultados empeora al ir aumentando la frecuencia de excitación.

4.5 Código elaboración de gráficas

Con el siguiente código y los respectivos archivos de datos se han generado las gráficas anteriormente mostradas.

Código 4.1 Elaboración de las gráficas con Matlab en el dominio del tiepo y de la frecuencia + aplicación del filtro de Chebyshev.

```
clear all; close all; clc

format long

%%%COMPARATIVA PARA SENO 10HZ%%%

%MPU6050%
A = importdata('datos1.txt');
[i,~]=find(isnan(A));
if isempty(i)==0 %isempty(A) returns logical 1 (true) if A is empty, and
     logical 0 (false)              %otherwise.
    datos = A(1:(i(1)-1),:);
else
    datos = A;
end
[n,~]=size(datos);

%%%%DOS CONTADORES PARA ELIMINAR VALORES REPETIDOS %%%%
j=1;
Acel(1,:) = datos(1,:);
Acel_memory=Acel(1,:);
for i=2:n
    Acel_read (1,:) = datos(i,:);
    if Acel_read(2)~=Acel_memory(2)
        j=j+3;
    end
    if Acel_read(3)==1
        Acel(j,:)=Acel_read(1,:);
    elseif Acel_read(3)==2
        Acel(j+1,:)=Acel_read(1,:);
    else
        Acel(j+2,:)=Acel_read(1,:);
    end
    Acel_memory=Acel_read;
end

figure
hold on
[m,n]=size(Acel);
% acelx=[]; acely=[];
acelz=[];
t=[];
```

```
for k=1:3:(m-2)
    t1 = linspace(Acel(k,1),Acel(k,104),100);
    t = [t t1];
    %acelx = [acelx Acel(k,4:103)];
    %acely = [acely Acel(k+1,4:103)];
    acelz = [acelz Acel(k+2,4:103)];
end
t=t(1385:2400); t=t-t(1); acelz=acelz(1385:2400);
plot(t,detrend(acelz),'b');

% ACEL LABORATORIO %
load('DATOS ADQUIRIDOS CON EL ACELERÓMETRO PRO/
    resultados_seno_10Hz_inalambrico')
tiempo_lab = tiempo(998:end)-1.924;
aceleracion_lab = aceleracion(998:end);
plot(tiempo_lab,aceleracion_lab,'r')
legend('MPU6050')
title('Comparativa para excitación tipo seno de 10Hz')
xlabel('t(s)')
ylabel('a_z (m/s^2)')
hold off

%------------DOMINIO DE LA FRECUENCIA------------%
%MPU6050%
deltat=[t(end)-t(1)]/length(t);
fs=1/deltat; %frecuencia de muestreo
%se hace la transformada de Fourier, para pasar al dominio de la
    frecuencia
acelz_fft=fft(acelz);
%A partir de la frecuencia de muestreo y el tiempo en el que se han
%registrado datos, se obtiene su equivalente en el dominio de la
    frecuencia
L=length(acelz);
P2 = abs(acelz_fft/L); % se normaliza
P1 = P2(1:L/2+1); % únicamente la mitad (simetría)
P1(2:end-1) = 2*P1(2:end-1);% Con la transformada rápida obtenemos la
    mitad de la amplitud
                          % de la señal original
f = fs*(0:(L/2))/L;
% se dibujan los resultados en el dominio de la frecuencia
figure
hold on; plot(f(5:end),P1(5:end),'b')
title('Espectro de frecuencias, seno 10Hz')
xlabel('Frecuencia [Hz]')
ylabel('Aceleración [m/s^2]')
%-----------------------------------------------%
%-----------------------------------------------%
% FILTRADO DE LA SEAL %
fs_exp = fs; %fijar el valor de la frecuencia de muestreo
```

```
ChebyOrder1=6; % orden del filtro
ChebyRipple1=1; % rizado de banda de paso de pico a pico
ChebyFreq1=10.3; % frecuencia de corte del filtro
[Bf1,Af1]=cheby1(ChebyOrder1,ChebyRipple1,ChebyFreq1/(fs_exp/2),'low');
   %filtro pasa bajos
% % % % % % % % % % %
ft = tf(Bf1,Af1);
[h w] = freqz(Bf1,Af1); %respuesta en frecuencia del filtro
figure
plot(w,abs(h))
title('Respuesta en frecuencia del filtro pasa bajos')
ylabel('Magnitud')
xlabel('Frecuencia normalizada')
% % % % % % % % % % %

ChebyOrder2=6;
ChebyRipple2=1;
ChebyFreq2=9.7; %fija el valor por encima de 5Hz
[Bf2,Af2]=cheby1(ChebyOrder2,ChebyRipple2,ChebyFreq2/(fs_exp/2),'high');
    %filtro pasa altos
% % % % % % % % % % %
ft = tf(Bf2,Af2);
[h w] = freqz(Bf2,Af2); %respuesta en frecuencia del filtro
figure
plot(w,abs(h))
title('Respuesta en frecuencia del filtro pasa altos')
ylabel('Magnitud')
xlabel('Frecuencia normalizada')
% % % % % % % % % % %

%filtro
respt = acelz;
respt = filter(Bf2,Af2,respt);
respt = filter(Bf1,Af1,respt);
%representamos señal sin filtrar en el dominio del tiempo%
figure
plot(t,acelz,'b')
title('Señal en el dominio del tiempo, seno 10Hz')
xlabel('tiempo [s]')
ylabel('Aceleración [m/s^2]')
%representamos señal filtrada en el dominio del tiempo%
figure
plot(t,respt,'b')
title('Señal filtrada en el dominio del tiempo, seno 10Hz')
xlabel('tiempo [s]')
ylabel('Aceleración [m/s^2]')
%hacemos la transformada a la señal filtrada%
acelzfiltrada_fft=fft(respt);
L = length(acelz);
P2_f = abs(acelzfiltrada_fft/L);
```

```
P1_f = P2_f(1:L/2+1);
P1_f(2:end-1) = 2*P1_f(2:end-1);
f = fs*(0:(L/2))/L;
figure
plot(f(5:end),P1_f(5:end),'b')
title('Espectro de frecuencias con señal filtrada, seno 10Hz')
xlabel('Frecuencia [Hz]')
ylabel('Aceleración [m/s^2]')
% plot(f_lab,P1_lab,'r')
%-----------------------------------------------%
%-----------------------------------------------%

%%%ACELERÓMETRO LABORATORIO%%%
deltat_lab=[tiempo_lab(end)-tiempo_lab(1)]/length(tiempo_lab);
fs_lab=1/deltat_lab;
aceleracion_lab_fft=fft(aceleracion_lab);
L=length(aceleracion_lab);
P2_lab = abs(aceleracion_lab_fft/L);
P1_lab = P2_lab(1:L/2+1);
P1_lab(2:end-1) = 2*P1_lab(2:end-1);
f_lab = fs_lab*(0:(L/2))/L;
%%%COMPARATIVA CON MPU6050 SIN FILTRAR%%%
figure
hold on
plot(f(5:end),P1(5:end),'b',f_lab,P1_lab,'r')
title('Comparativa en el dominio de la frecuencia, seno 10Hz')
legend('MPU6050','352C33')
plot([10 10],[0 2],'--')
xlabel('Frecuencia [Hz]')
ylabel('Aceleración [m/s^2]')
hold off
%%%COMPARATIVA CON MPU6050 FILTRADAR%%%
figure
hold on
plot(f(5:end),P1_f(5:end),'b',f_lab,P1_lab,'r')
title('Comparativa en el dominio de la frecuencia (señal filtrada), seno
    10Hz')
legend('MPU6050','352C33')
plot([10 10],[0 2],'--')
xlabel('Frecuencia [Hz]')
ylabel('Aceleración [m/s^2]')
hold off
```

5 Conclusiones y desarrollos futuros

En este último capítulo se resumirá el desarrollo del proyecto y los objetivos logrados. Además, se expondrá una serie de tareas de interés de cara a la realización de trabajos futuros.

A lo largo de la memoria se han ido analizando un conjunto variado de configuraciones y métodos con el objetivo de lograr unos requisitos de partida: adquisición de datos inalámbrica con una frecuencia superior a unos 200 valores por segundo. La primera configuración arrojó unos resultados bastante lejos con respecto a la frecuencia de adquisición. Y no fue hasta la segunda configuración con el método de adquisición basado en la aplicación de Roger Meier [2] cuando se superaron los requisitos y se estableció la configuración definitiva, compuesta por dos módulos ESP8266 comunicados inalámbricamente sin necesidad de disponer de una red local. Los señales comparadas con el acelerómetro PCB PIEZOTRONICS 352C33 resultaron bastante aceptables para frecuencias de excitación inferiores a 30 Hz, teniéndose en cuenta el coste de los equipos y la transmisión inalámbrica.

Como se ha comentado en la introducción, el objetivo principal del proyecto en el que se ha trabajado con este TFG fue, desde un principio, la monitorización de una estructura inalámbricamente. Aunque en el presente trabajo no se ha alcanzado esa fase final, se comentan a continuación algunos pasos a seguir para su realización en trabajos futuros. Una vez se dispone de los desarrollos logrados en este trabajo, el principal paso a completar sería el de modificar el código de Matlab con el que se hace la correcta lectura del .txt, archivo en el que se capturan las cadenas de aceleraciones simultáneamente. El objetivo es leer y procesar (dominio de la frecuencia) una cierta cantidad de cadenas de manera iterativa, de esta forma se lograría un análisis a tiempo real de los valores de las aceleraciones de la estructura. Además, se podrían introducir condiciones en el código del módulo servidor para capturar las cadenas en el .txt únicamente cuando las aceleraciones superen un cierto valor, logrando así no almacenar valores de aceleraciones que no aporten información a la determinación de estados críticos de la estructura en cuestión. Por último, debido a la sensibilidad relativamente baja y la alta densidad de ruido de salida de los acelerómetros de bajo coste, se podría trabajar en lograr un aumento efectivo en la sensibilidad del acelerómetro utilizando el fenómeno de resonancia estocástica (SR) [17]. SR es un enfoque en el que, de forma contraria a la intuición, las señales débiles se amplifican en lugar de abrumarlas por la adición de ruido.

Finalmente, incorporando a los desarrollos alcanzados en este proyecto los trabajos futuros anteriormente comentados, se dispondría de un sistema de bajo coste capaz de monitorizar una estructura inalámbricamente.

Apéndice A
Puesta en funcionamiento del sistema

Este apéndice A se centrará en la descripción detallada, paso a paso, de la puesta en funcionamiento del sistema. Con ello se pretende lograr que cualquier usuario con unos conocimientos básicos sea capaz de utilizar los desarrollos logrados en este proyecto.

A.1 Equipos

Se tendrá que disponer de los siguientes equipos:

- Un ordenador.
- Dos módulos ESP8266.
- Un sensor MPU6050.
- Dos conectores USB 2.0 micro-b 5 pin y cables para el conexionado del sensor al módulo.

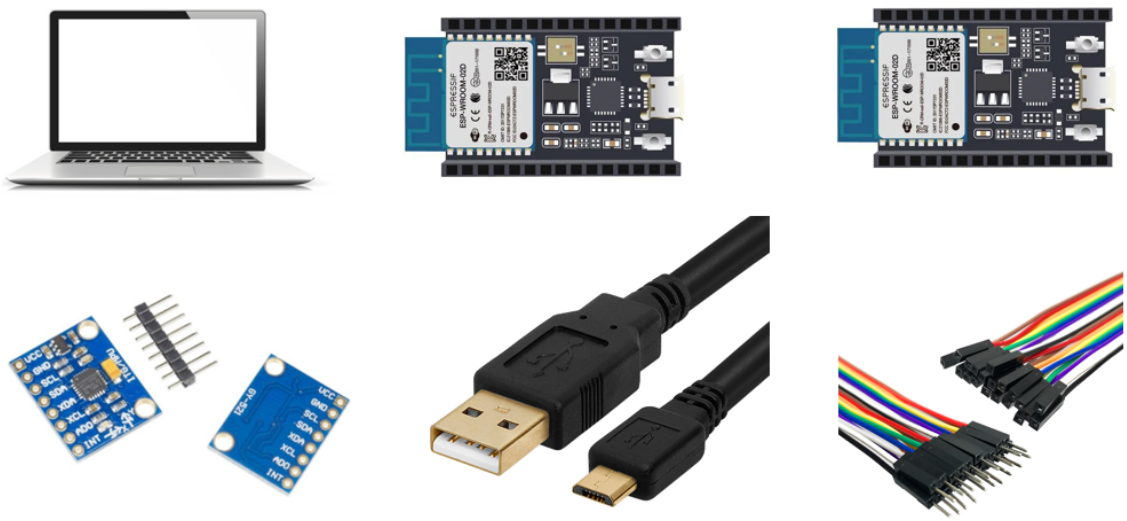


Figura A.1 Equipos necesarios.

Pinchando en los siguientes enlaces se accede a las páginas de compra: [**Comprar ESP8266**], [**Comprar MPU6050**].

A.2 Programas

En el equipo en el que se vaya a conectar el sistema, será necesario contar con los siguientes programas:

- Matlab.
- El IDE de Arduino.
- La aplicación CoolTerm.

Los dos últimos son de uso gratuito y se pueden descargar pinchando en los siguientes enlaces:[**Descargar IDE de Arduino**] , [**Descargar CoolTerm**].

A.3 Librerías e instalaciones en IDE de Arduino

Por un lado, se deberá realizar la instalación de la placa ESP8266 en el IDE de Arduino. Con ello, se podrá programar el ESP8266 en el IDE utilizándose su lenguaje de programación. Para realizar dicha instalación será necesario completar los siguientes pasos:

- 1. Una vez se está en el IDE de Arduino, se despliega la pestaña de Archivo y se hace clic en preferencia.
- 2. Se pega el siguiente enlace *hhttp://arduino.esp8266.com/stable/package_esp8266com_index.jso* en el campo "Gestor de URLs Adicionales de Tarjetas" y se acepta.
- 3. Se despliega la pestaña de Herramientas y se hace clic en Placa > Gestor de Tarjetas. Se busca ESP8266 y se pulsa instalar para la" ESP8266 por ESP8266 Comunidad".
- Después de unos segundos debe aparecer como instalado, dándose por terminada la instalación.

Por otro lado, se deberán instalar las siguientes librerías, pinchando en los respectivos enlaces se podrán descargar directamente:

- Librería Open Source "ESPAsyncWebServer", [**Descargar librería**].
- Librería Open Source "ESPAsync TCP", [**Descargar librería**].

Una vez estén descargadas, se descomprimirán las librerías y se moverán a la carpeta de bibliotecas de instalación de Arduino IDE.

A.4 Puesta a punto del Servidor

Antes que nada se procederá a conectar el MPU6050 al módulo ESP8266 como se indica en el siguiente esquema.

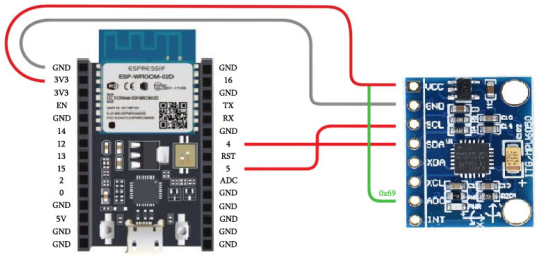


Figura A.2 Esquema conexión MPU6050 al módulo ESP8266.

Seguidamente se abrirá el IDE de Arduino y se conectará el módulo ESP8266 a un puerto USB del ordenador. Una vez conectado, se desplegará el menú herramientas y se seleccionará: Placa:"Generic ESP8266 Module" y Puerto:"al que se haya conectado".
Tras esto, se copiará el siguiente código y se le dará a "subir" para cargarlo en la placa.

Código A.1 Código cargado en el servidor.

```
#include "Wire.h"
#include <ESP8266WiFi.h>
#include "ESPAsyncWebServer.h"

//Direccion I2C de la IMU
#define MPU1 0x68

const char* ssid = "ESP8266-Access-Point";
const char* password = "bienvenido";

AsyncWebServer server(80);

int16_t AcX1, AcY1, AcZ1;
float axprint1, ayprint1, azprint1;

String vecaxprint1_1, vecayprint1_1, vecazprint1_1, myaxprint1,
    myayprint1, myazprint1;
String memoryax1, memoryay1, memoryaz1;

String vecaxprint1_2, vecayprint1_2, vecazprint1_2;

int ITER;

unsigned long cont, t1, t2;
float t1s, t2s;
String cadcont, cadt1, cadt2;
```

```
String readAX1_2() {
  return String(vecaxprint1_2);
}

String readAY1_2() {
  return String(vecayprint1_2);
}

String readAZ1_2() {
  return String(vecazprint1_2);
}

void setup(){

  Serial.begin(115200);

  Serial.print("Setting AP (Access Point)");
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.println(IP);

  server.on("/acx1_2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readAX1_2().c_str());
  });
  server.on("/acy1_2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readAY1_2().c_str());
  });
  server.on("/acz1_2", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readAZ1_2().c_str());
  });

  Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL
  Wire.beginTransmission(MPU1);
  Wire.write(0x6B);// PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);

  server.begin();
}

void loop(){

    if(WiFi.softAPgetStationNum() == 1){

    t1=millis();
      for(ITER=0; ITER<100; ITER++){
    Wire.beginTransmission(MPU1);
    Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
    Wire.endTransmission(false);
```

```
    Wire.requestFrom(MPU1,6,true); //A partir del 0x3B, se piden 6
       registros
    AcX1=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
    AcY1=Wire.read()<<8|Wire.read();
    AcZ1=Wire.read()<<8|Wire.read();

    //Escalado de las aceleraciones
    //MPU6050_1
    axprint1=AcX1*(9.81/16384)-2.7;
    ayprint1=AcY1*(9.81/16384);
    azprint1=AcZ1*(9.81/16384);

    myaxprint1 = String(axprint1);
    vecaxprint1_1= memoryax1 + " " + myaxprint1;
    memoryax1=vecaxprint1_1;
    myayprint1 = String(ayprint1);
    vecayprint1_1= memoryay1 + " " + myayprint1;
    memoryay1=vecayprint1_1;
    myazprint1 = String(azprint1);
    vecazprint1_1= memoryaz1 + " " + myazprint1;
    memoryaz1=vecazprint1_1;
    delay(2.3);
      }
      memoryax1="";
      memoryay1="";
      memoryaz1="";
      t2=millis();
      t1s = t1/1000.0;
      t2s = t2/1000.0;
      cadt1=String(t1s);
      cadt2=String(t2s);
      cadcont=String(cont);
      vecaxprint1_2 = cadt1 + " " + cadcont + " " + "1" + vecaxprint1_1 +
          " " + cadt2;
      vecayprint1_2 = cadt1 + " " + cadcont + " " + "2" + vecayprint1_1 +
          " " + cadt2;
      vecazprint1_2 = cadt1 + " " + cadcont + " " + "3" + vecazprint1_1 +
          " " + cadt2;
      cont++; //en convertir estas cadenas se invierte en torno a 1ms
    }
}
```

Por último, una vez compilado, se desconecta el módulo del puerto serie y se cierra la ventana con el código del servidor.

A.5 Puesta a punto del Cliente

En este caso, únicamente será necesario conectar el otro módulo ESP8266 a un puerto USB del ordenador. Una vez conectado, al igual que para el servidor, se desplegará el menú herramientas y se seleccionará: Placa:"Generic ESP8266 Module" y Puerto:"al que se haya conectado".
Tras esto, se copiará el siguiente código y se le dará a "subir" para cargarlo en la placa.

Código A.2 Código cargado en el cliente.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti WiFiMulti;

const char* ssid = "ESP8266-Access-Point";
const char* password = "bienvenido";

const char* serverNameax1_2 = "http://192.168.4.1/acx1_2";
const char* serverNameay1_2 = "http://192.168.4.1/acy1_2";
const char* serverNameaz1_2 = "http://192.168.4.1/acz1_2";

String axprint1_2, ayprint1_2, azprint1_2;

void setup() {
  Serial.begin(115200);
  Serial.println();

  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP(ssid, password);
  while((WiFiMulti.run() == WL_CONNECTED)) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Connected to WiFi");
}

void loop() {
    if ((WiFiMulti.run() == WL_CONNECTED)) {
      axprint1_2 = httpGETRequest(serverNameax1_2);
      ayprint1_2 = httpGETRequest(serverNameay1_2);
      azprint1_2 = httpGETRequest(serverNameaz1_2);

      Serial.println(axprint1_2);
      Serial.println(ayprint1_2);
      Serial.println(azprint1_2);
```

```
      delay(2.3);
    }
}

String httpGETRequest(const char* serverName) {
  WiFiClient client;
  HTTPClient http;

  http.begin(client, serverName);

  int httpResponseCode = http.GET();

  String payload = "--";

  http.end();

  return payload;
}
```

Del mismo modo que antes, una vez compilado, se desconecta el módulo del puerto serie y se cierra la ventana con el código del cliente.

A.6 Adquisición de los datos con CoolTerm

Una vez tenemos cargados los códigos correspondientes en cada módulo se está en condiciones de realizar la adquisición de datos con CoolTerm. Para ello, el módulo servidor deberá estar **desconectado** y el cliente **conectado**. Tras esto, se conecta con la aplicación el módulo cliente.

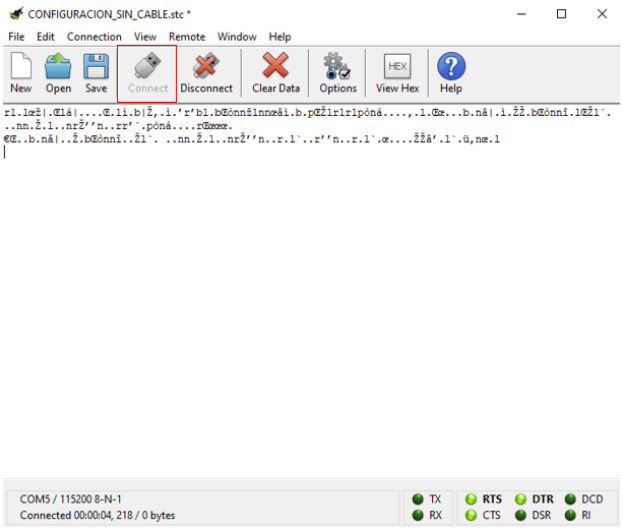


Figura A.3 Conexión del módulo cliente con Coolterm.

A continuación, se configura la aplicación Coolterm. Para ello, como se muestra en la siguiente captura, se indica en las configuraciones cuál es el puerto por el que se transferirán los datos y los baudios (115200) de dicha transmisión por el puerto serie.

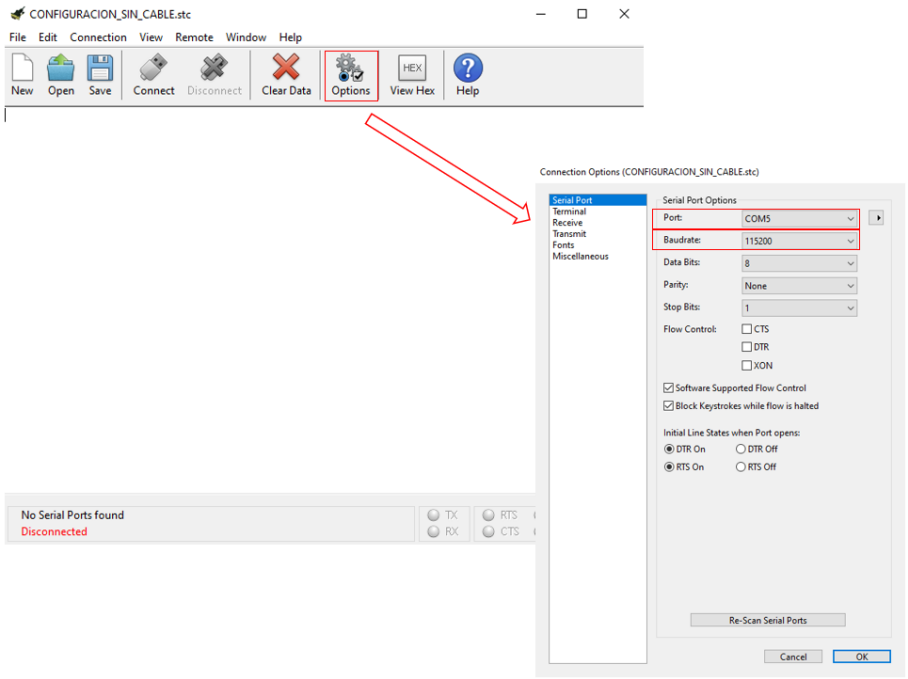


Figura A.4 Configuraciones del Coolterm.

Se define ahora cuál va a ser el archivo .txt donde se capturarán las cadenas de las aceleraciones arrojadas por el servidor.

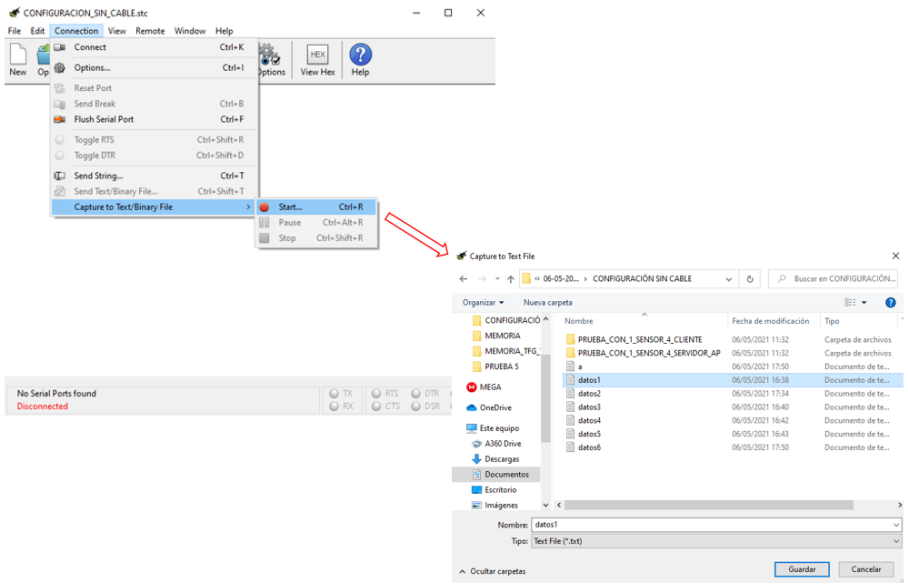


Figura A.5 Selección del .txt donde queremos capturar los datos.

Seguidamente, se conecta el servidor a la alimentación. Una vez se ha establecido la conexión inalámbrica entre ambos, se empezarán a mostrar las cadenas con los valores de las aceleraciones. Cuando se desee terminar la transmisión, se desconectarán ambos módulos y se parará la captura que activamos en el .txt.

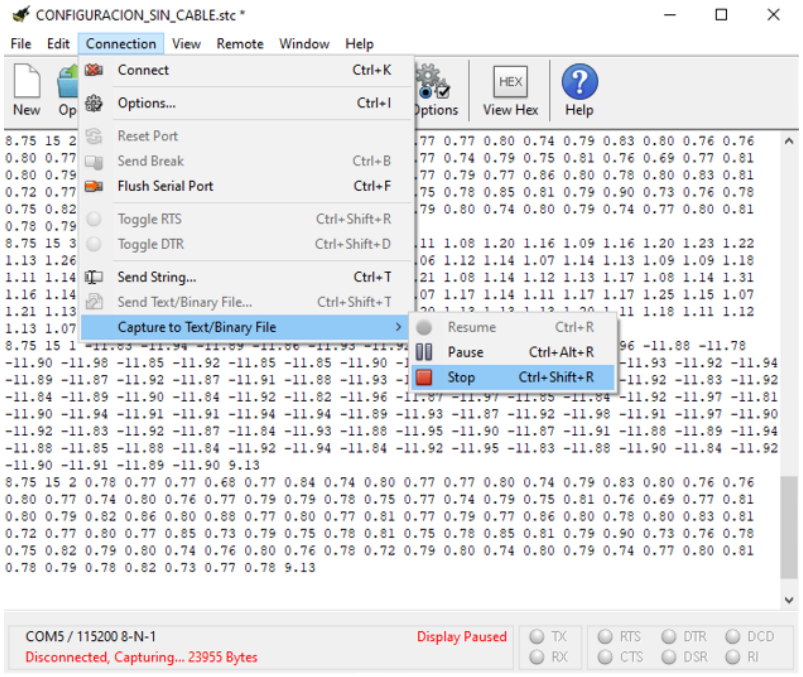


Figura A.6 Fin de la captura en el .txt.

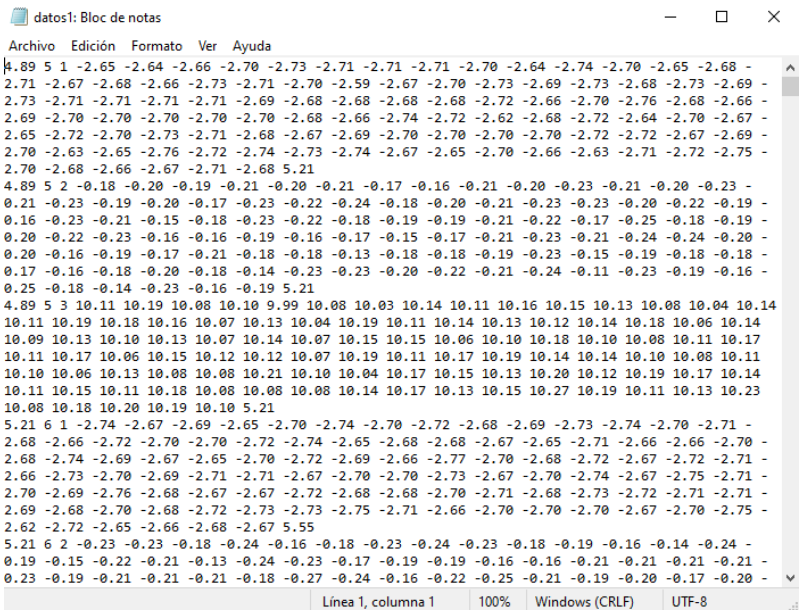


Figura A.7 Valores guardados en el .txt.

Obteniéndose finalmente el .txt con las cadenas almacenadas.

A.7 Lectura y representación de las aceleraciones con Matlab

Una vez se dispone del .txt con los valores almacenados (este deberá estar guardado en la misma carpeta que el archivo de Matlab), se procede ha realizar una correcta lectura de las cadenas con Matlab. Para ello, se copiará el siguiente código en un nuevo Script.

Código A.3 Código en Matlab.

```
clear all; close all; clc

A = importdata('datos1.txt');
[i,~]=find(isnan(A));
if isempty(i)==0 %returns logical 1 if A is empty, and logical 0
    otherwise.
    datos = A(1:(i(1)-1),:);
else
    datos = A;
end
[n,~]=size(datos);

%%%% DOS CONTADORES PARA ELIMINAR VALORES REPETIDOS %%%%
j=1;
Acel(1,:) = datos(1,:);
Acel_memory=Acel(1,:);
for i=2:n
    Acel_read (1,:) = datos(i,:);
    if Acel_read(2)~=Acel_memory(2)
        j=j+3;
    end
    if Acel_read(3)==1
        Acel(j,:)=Acel_read(1,:);
    elseif Acel_read(3)==2
        Acel(j+1,:)=Acel_read(1,:);
    else
        Acel(j+2,:)=Acel_read(1,:);
    end
    Acel_memory=Acel_read;
end

% REPRESENTACIÓN DE LOS TRES EJES %
figure
[m,n]=size(Acel);
acelx=[];
acely=[];
acelz=[];
t=[];
for k=1:3:(m-2)
    t1 = linspace(Acel(k,1),Acel(k,104),100);
    t = [t t1];
    acelx = [acelx Acel(k,4:103)];
```

```
    acely = [acely Acel(k+1,4:103)];
    acelz = [acelz Acel(k+2,4:103)];
end
plot(t,acelx,'r',t,acely,'g',t,acelz,'c')
xlabel('t(s)')
ylabel('a (m/s^2)')
title('Aceleraciones frente al tiempo')
legend('Acelx','Acely','Acelz')

% NOS CENTRAMOS EN EL EJE Z %
figure
plot(t,detrend(acelz),'c')
xlabel('t(s)')
ylabel('a (m/s^2)')
title('Aceleración en eje z frente al tiempo [detrend()]')
```

Las gráficas resultantes deberán ser similares a las mostradas a continuación.

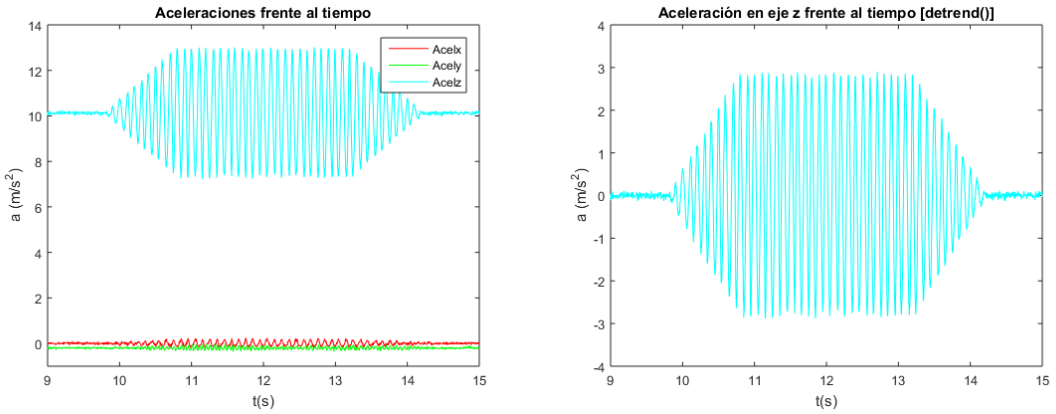


Figura A.8 Representación de las aceleraciones en los tres ejes (Izq.). Representación de las amplitudes de las aceleraciones en el eje z (dcha.).

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] ERBESSD INSTRUMENTS. [Consulta: año 2021]. [**Consultar fuente**].

[2] Roger Meier's Freeware «CoolTerm». [Consulta: año 2021]. [**Consultar fuente**].

[3] Luis Llamas «Determinar la orientación con Arduino y el IMU MPU-6050». [Consulta: año 2021]. [**Consultar fuente**].

[4] Luis Llammas «El bus I2C en Arduino». [Consulta: año 2021]. [**Consultar fuente**].

[5] ArduProject «MPU6050 y su programación en Arduino». [Consulta: año 2021]. [**Consultar fuente**].

[6] IvenSense «Datasheet MPU6050». [Consulta: año 2021]. [**Consultar fuente**].

[7] Random Nerd Tutorials «Installing ESP8266 Board in Arduino IDE». [Consulta: año 2021]. [**Consultar fuente**].

[8] Espressif Systems «Datasheet ESP8266-DevKitC». [Consulta: año 2021]. [**Consultar fuente**].

[9] Random Nerd Tutorials «ESP32 DHT11/DHT22 Web Server – Temperature and Humidity using Arduino IDE». [Consulta: año 2021]. [**Consultar fuente**].

[10] Excel vba datos «68 excel vba 2019: extraer texto de pagina web con etiqueta html con id, tag y class (scraping)». [Consulta: año 2021]. [**Consultar fuente**].

[11] Excel vba datos «09 curso excel vba 2016: scraper o extraer un dato (etiqueta html con ID) de pagina web». [Consulta: año 2021]. [**Consultar fuente**].

[12] Automatetheweb «Learn to write Excel VBA code to automate your web browser.». [Consulta: año 2021]. [**Consultar fuente**].

[13] MathWorks «webread». [Consulta: año 2021]. [**Consultar fuente**].

[14] Random Nerd Tutorials «ESP8266 Client-Server Wi-Fi Communication Between Two Boards (NodeMCU)». [Consulta: año 2021]. [**Consultar fuente**].

[15] PCB PIEZOTRONICS «Model: 352C33». [Consulta: año 2021]. [**Consultar fuente**].

[16] Brüel Kjaer «Modelo 3160, Módulo generador». [Consulta: año 2021]. [**Consultar fuente**].

[17] APS 400 ELECTRO-SEIS. [Consulta: año 2021]. [**Consultar fuente**].

[18] Europe PMC. «Extraction of Bridge Fundamental Frequencies Utilizing a Smartphone MEMS Accelerometer». [Consulta: año 2021]. [**Consultar fuente**].

[19] TFG. «Análisis vibratorio de un dispositivo mecánico. Aplicación a un sistema ferroviario». [Consulta: año 2021]. [**Consultar fuente**].