Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado de Ingeniería en Electrónica Industrial y Automática

Instrumentación de una maqueta de un edificio de dos plantas para su uso como demostrador de sistemas de control de vibraciones

Autor:

Garzón Escudero, Jorge

Tutor:

Lorenzana Ibán, Antolín
Estructuras

Valladolid, Julio de 2018.

Resumen y palabras claves.

En esta memoria se confecciona el funcionamiento de una maqueta, la cual simula un edificio vibrando, es decir, un balanceo. En la estructura se incorporan distintos equipos los cuales permitirán controlar la generación de balanceo del edificio, eliminación del balanceo, amortiguamiento de la capacidad de eliminación de balanceo y obtención de medidas de aceleración que experimenta la maqueta. Para ello se utilizarán tres servomotores cuyo funcionamiento actuará en el balanceo de la estructura y dos acelerómetros que generarán las medidas de aceleración experimentadas.

Estos equipos serán controlados por Arduino mediante el software que se ha implementado. En concreto se utilizará un Arduino para realizar el control de los servomotores y otro distinto para el control de los acelerómetros.

Por último indicar, que mediante Raspberry se ha generado un "*interface*" para realizar el control de los Arduinos mediante software de una manera sencilla y cómoda.

Palabras claves

Balanceo, disipación, servomotor, Arduino y acelerómetro.

Índice

1. Introducción y objetivos.

Se va a realizar la construcción de una maqueta equivalente a un edificio. Se incluirá un sistema de disipación del balanceo del edificio, un generador de balanceo de estructura, un controlador de la capacidad de eliminación de balanceo y un medidor de aceleraciones.

Todo ello será generado mediante varios servomotores y acelerómetros controlados por el software implementado en la tecnología Arduino. Para facilitar un manejo intuitivo y correcto de la maqueta se creará un *interface* en Java soportado por una infraestructura Raspberry.

Se realizará la búsqueda de los equipos adecuados que cumplan con las expectativas requeridas. Una vez encontrados, se necesitará desarrollar el software necesario para su calibración individual.

A partir de la calibración de los equipos de manera individual, se realiza su inclusión en un mismo sistema, para ello será necesario modificar el software buscando un control general.

Aparecerán problemas a los cuales se irán encontrando solución.

Los objetivos marcados en el proceso de desarrollo del proyecto son:

- Obtención del equipo adecuado.
- Probar la validez del equipo elegido.
- Generar suficiente perturbación en la estructura para provocar un balanceo resonante.
- Obtener la calibración adecuada del servomotor que genera la perturbación para poder cambiar el balanceo de la estructura variando el periodo del ciclo.
- Obtener la calibración adecuada del servomotor que permite la actuación del péndulo de disipación de balanceo.

- Obtener la calibración adecuada del servomotor que interacciona con el péndulo de disipación de balanceo eliminando parte de su capacidad de disipación.

- Obtener la calibración adecuada de los acelerómetros, realizando el cambio de escala oportuno para obtener mediciones de aceleración en m/s^2.

- Generar el software necesario para realizar las calibraciones individuales de los equipos.

- Generar el software necesario para realizar el control del sistema conjunto.

- Determinar los parámetros correctos de funcionamiento de la maqueta.

2. Selección del equipo inmerso en la maqueta.

2.1 Introducción al desarrollo.

Comenzando con el desarrollo del proyecto se van a determinar los equipos electrónicos que entran en juego en la consecución del mismo. Como en todo proyecto aparecen problemas, los cuales se irán resolviendo a medida que se avanza en la investigación de los componentes utilizados.

Inicialmente se elegirá un equipo y se realizará un estudio completo para determinar si satisface las necesidades y cumple con los objetivos marcados. De esta manera habrá equipos que no son válidos y será necesaria la búsqueda de otros con distintas características que solucionen el problema encontrado.

Una vez determinada la manera de actuar, se procede a la presentación de los equipos electrónicos que van a ser utilizados:

- Servomotor.
- Arduino.
- Acelerómetro.
- Raspberry.

2.2 Arduino.

2.2.1 Introducción Arduino.

El equipo para llevar a cabo la carga computacional del proyecto con la que controlar los servomotores y la lectura de los sensores es el microcontrolador de Arduino.

Arduino es una plataforma de hardware de código abierto, en un entorno de desarrollo que está basado en el lenguaje de programación Processing.

Consta de una placa, con varios modos de alimentación, además de un conjunto de pines digitales y analógicos que funcionan como entradas y salidas.

Este dispositivo permite la conexión del plano analógico con el plano digital de una manera muy fácil y sencilla. Las ventajas que ofrece la utilización de este hardware son las siguientes:

- Accesible, las placas Arduino son relativamente baratas en comparación con otras plataformas basadas en un microcontrolador. La versión menos costosa del módulo Arduino puede ser montada a mano.

- Multiplataforma, el software de Arduino (IDE) se ejecuta en los sistemas operativos Windows, Macintosh OSX y Linux.

- Programación clara, el software de Arduino (IDE) es fácil de usar para los principiantes, pero lo suficientemente flexible para que los usuarios avanzados lo puedan aprovechar.

- Por su bajo costo y facilidad de uso es un instrumento adecuado para ser utilizado en los distintos proyectos llevados a cabo en los laboratorios de universidad.

- Existe amplia información en Internet.

2.2.2 Arduino Mega 2560.

El microcontrolador en concreto que se ha elegido es Arduino Mega 2560. Esta elección se ha realizado por cuestiones de conveniencia, ya que este equipo es el que posee el laboratorio donde se está llevando a cabo el proyecto. No obstante, se podría utilizar cualquier modelo de Arduino sin presentar problema alguno.

Las características más destacadas de Arduino Mega 2560 son:

- Voltaje operativo: 5V
- Voltaje de entrada: 5V
- Voltaje de entrada : 6-20V
- Pines digitales Entrada / Salida: 54 (15 de ellos proporcionan salida PWM)
- Pines analógicos de entrada: 16
- Corriente DC por cada pin Entrada / Salida: 40mA
- Corriente DC entregada en el pin 3.3V: 50mA
- Memoria Flash: 256 KB
- SRAM: 8KB
- EEPROM: 4KB
- *Clock Speed*: 16MHz

La disposición de sus elementos es apreciable en la ilustración 1.

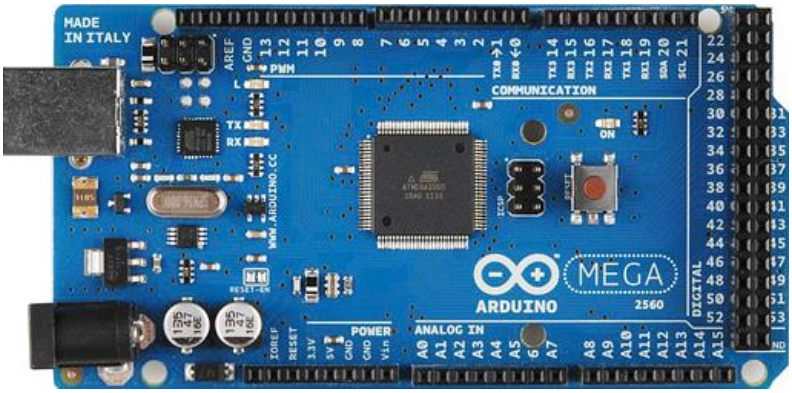

Ilustración 1 Arduino Mega 2560, muestra de los pines de la tarjeta (imagen obtenida de la web, ver Anexo)

Los pines más característicos del microcontrolador son los siguientes:

- VIN: La placa puede ser alimentada realizando la conexión en este pin.
- 3.3V: Este pin ofrece una salida de 3.3V y una corriente de 50 mA.
- 5V: Este pin ofrece una salida de 5V y una corriente de 40mA.
- GND: Este pin localiza la puesta a tierra de la placa (*ground* 0V).
- Pines analógicos: Son los pines señalados con una A (A0 - A15).
- Pines digitales: Son los pines señalados con número (0 - 53).

En el montaje que se va a llevar a cabo se utilizan algunos de los pines anteriormente mencionados. El funcionamiento o la función del resto de pines que se observan en la imagen pueden consultarse en el *datasheet* (incluido en el Anexo).

2.3 Servomotor.

2.3.1 Introducción servomotores.

Un servomotor es un dispositivo electromecánico, más comúnmente denominado motor paso a paso el cual tiene la propiedad característica de que gira un número determinado de grados alrededor de su eje y mantiene su posición.

Internamente está formado por un motor DC, un circuito integrado y un sistema de engranajes como se puede apreciar en la ilustración 2.

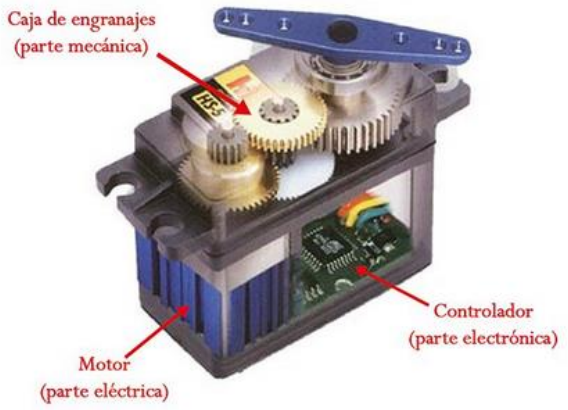

Ilustración 2 Partes de un servomotor (imagen obtenido de la web, ver Anexo).

Dentro de la familia de los servomotores, se distinguen dos tipos:

- Servomotores de ángulo de giro limitado: Generalmente permiten una rotación de 180º o menor. Nunca dan una vuelta completa.

- Servomotores de rotación continua: Son aquellos que permiten realizar una rotación completa (360º).

En ambos casos se puede controlar el ángulo girado alcanzando la posición requerida.

La conexión de los servomotores se realiza por medio de 3 cables, que encaminan la señal de alimentación (Voltaje positivo), tierra (*ground*) y la señal de control (señal PWM). Los colores utilizados en la mayoría de los casos son los indicados en la ilustración 3.

Ilustración 3 Gama de colores más utilizados para los distintos conductores.

Uno de los aspectos más importantes de un servomotor, es la denominada señal de control. Se trata de una señal PWM (formada por un impulso de onda cuadrada),

en la que variando el ancho del pulso cuando este se encuentra en su posición *HIGH* (*Cicle Duty*), hace que el motor gire un número de grados u otro.

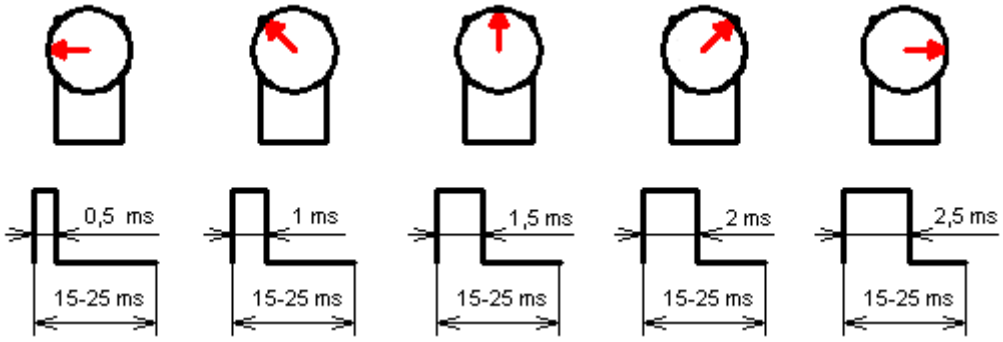

Ilustración 4 Grado de giro en función del ancho de pulso (imagen obtenido de la web, ver Anexo).

Las señales de PWM requeridas son similares para la mayoría de los modelos, no obstante conviene analizar el *datasheet* de cada equipo.

2.2.2 Servomotor "Tower Pro MG995".

En la ilustración 5, aparece el equipo mencionado, el cual se trató como una opción a tener en cuenta para el proyecto. Su *datasheet* se encuentra en el Anexo, lugar donde se pueden consultar y corroborar ciertos datos de interés que serán utilizados a continuación para estudiar este equipo.

Ilustración 5 Servomotor "Tower Pro MG995" (imagen obtenida de la web, ver anexo).

El motivo por el cual se descartó esta opción tiene que ver con el Arduino con el que se ha realizado el proyecto. Dicho equipo necesitaba ser alimentado de

manera externa (en vez de ser alimentado desde el USB), para proporcionar la suficiente potencia con la que obtener un correcto funcionamiento del servomotor.

En la hoja de características del servomotor se puede observar que la tensión de funcionamiento es de entre 4.8 V a 7.2V. Comprobando con el polímetro la tensión de salida del USB de nuestro equipo de trabajo obtenemos que su valor es de exactamente 4.8 V, valor que cumple las especificaciones.

La conexión se realiza como se observa en la imagen:

- Cable rojo del servo con el PIN de 5v de la placa de Arduino (alimentación).
- Cable marrón del servo con el PIN GND de la placa Arduino (Tierra).
- Cable naranja del servo con el PIN 8 PWM (señal de control).

Destacar que la elección del PIN 8, en cuanto a la señal de control se refiere, es fruto de como se ha desarrollado el programa en la plataforma de Arduino, cuyo código se incluye en el Anexo bajo el nombre de Servomotor_experimento. La conexión se realiza como en la ilustración 6.

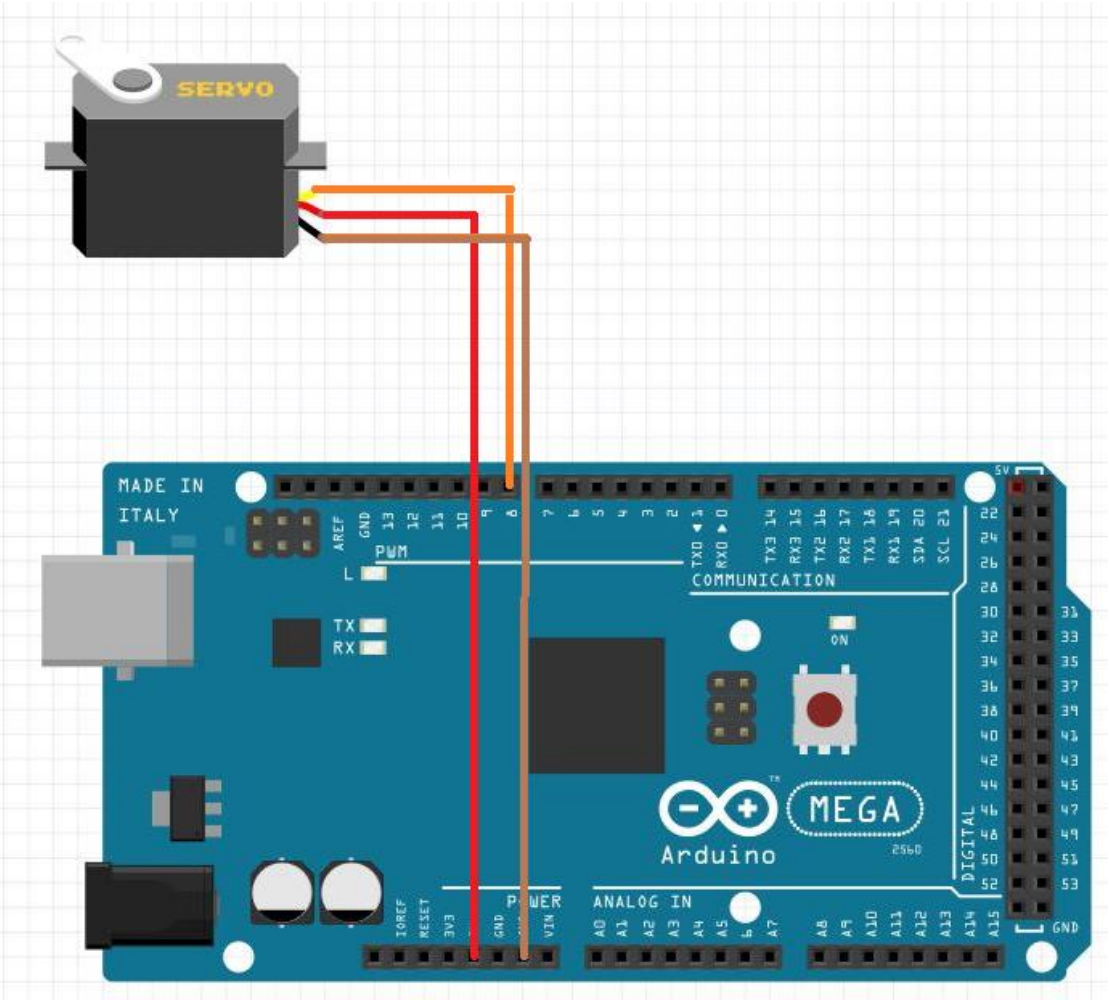

Ilustración 6 Conexión del servomotor con Arduino.

Una visión más clara del circuito eléctrico se observa en la ilustración 7:

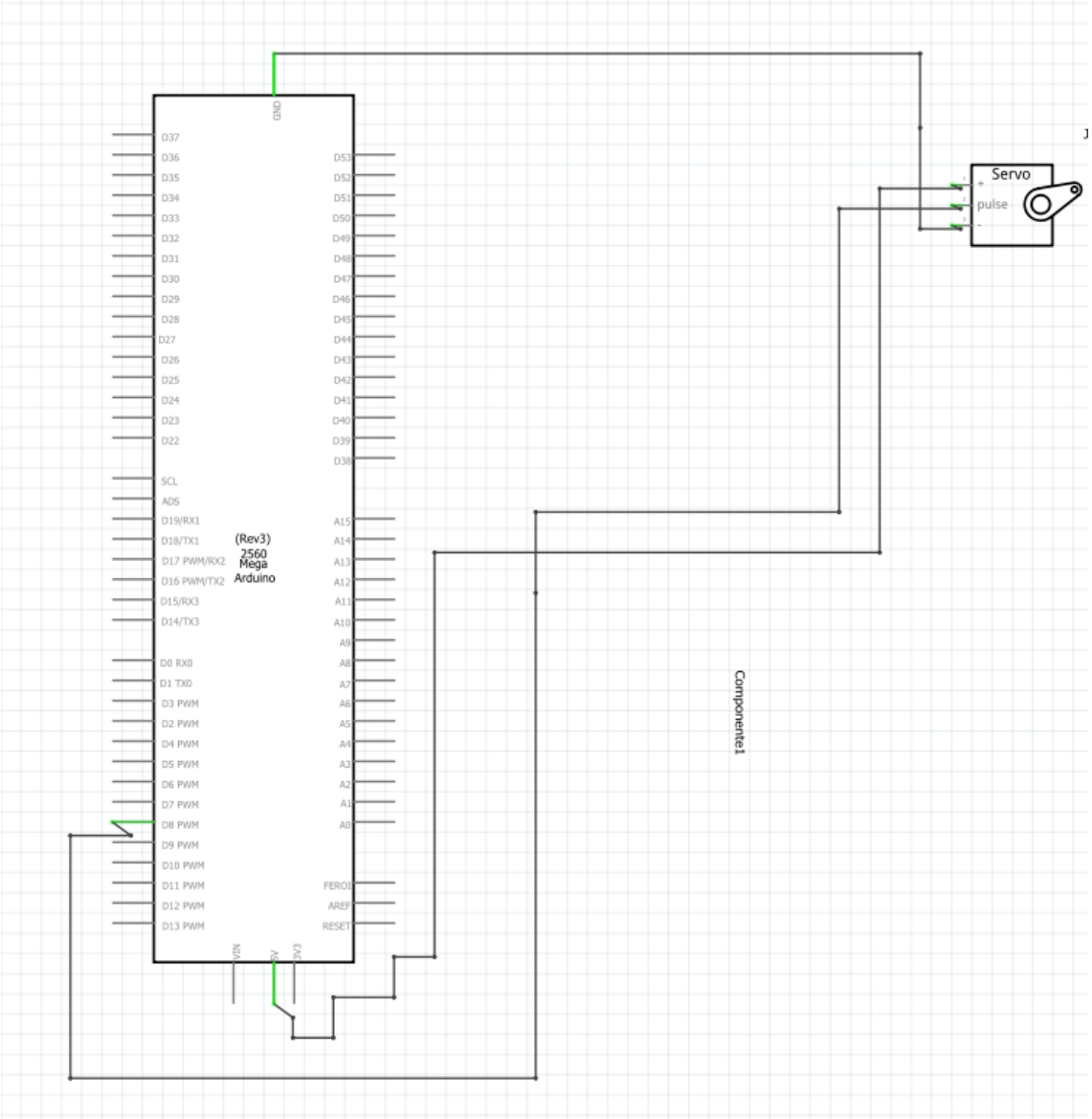

Ilustración 7 Esquema eléctrico de la conexión entre el servomotor y Arduino

Con el montaje realizado se carga el programa creado exclusivamente para hacer funcionar el servomotor, en la placa de Arduino y se ejecuta. El resultado de esta operación es nulo debido a que no se aprecia movimiento en el servomotor.

Una vez se revisa el correcto funcionamiento de las conexiones y el programa, se determina que la alimentación del servomotor no es la adecuada.

Consecuentemente con los resultados obtenidos, disponemos a alimentar Arduino de manera externa, con el objetivo de que los pines de salida de este equipo otorguen una tensión de al menos 5v y aumente la corriente suministrada al servomotor.

Para ello se realizan las mismas conexiones explicadas con anterioridad para el servomotor y añadimos una fuente de alimentación de 9 V y 0.5A.

Con alimentación externa y tras múltiples pruebas para calibrar en la mayor medida de lo posible el programa cargado en la placa de Arduino, el servomotor vibra de una forma un tanto "sospechosa", con lo que determinamos que el equipo se encuentra defectuoso.

Por consiguiente debido a las quejas en forma de comentario encontradas en la página de venta por otros clientes se decide la búsqueda de otro equipo el cual nos permita asegurar el correcto funcionamiento y obtener una mayor fiabilidad del sistema.

2.3.3 Servomotor "LOBOT LD-20MG".

En la ilustración 8 aparece el equipo mencionado, el cual se trató como una opción a tener en cuenta para el proyecto. Sus características se encuentran en el Anexo, lugar donde se pueden consultar y corroborar ciertos datos de interés que serán utilizados a continuación para estudiar este equipo.

Ilustración 8Ilustración 8 Servomotor "LOBOT LD- 20MG" (imagen obtenida de la web, ver Anexo).

.

El primer obstáculo que se plantea para su utilización es como en el caso anterior, la necesidad de alimentación externa. La tensión de alimentación necesaria es de 7.4V. Utilizando una fuente de alimentación adecuada el problema queda resuelto.

Realizamos el montaje, con las siguientes conexiones:

- Cable rojo del servo con el PIN de 5v de la placa de Arduino (alimentación).
- Cable negro del servo con el PIN GND de la placa Arduino (Tierra).
- Cable blanco del servo con el PIN 8 PWM (señal de control).

Destacar que la elección del PIN 8, en cuanto a la señal de control se refiere, es fruto de como se ha desarrollado el programa en la plataforma de Arduino, cuyo código se incluye en el Anexo bajo el nombre de Servomotor_experimento.

La conexión se realiza como se puede apreciar en la ilustración 9, con la correspondiente alimentación externa:

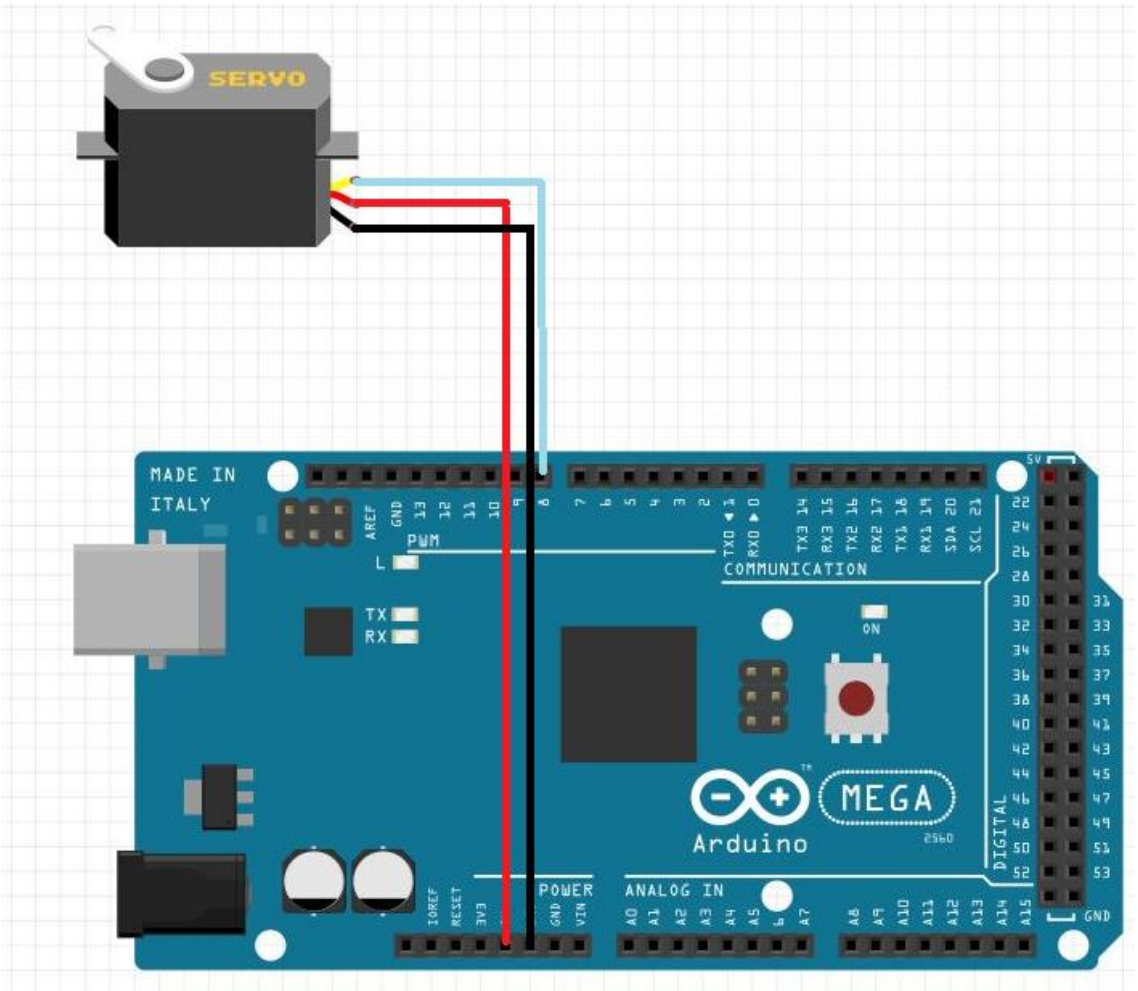

Ilustración 9 Conexión Servomotor y Arduino.

Una visión más clara del esquema eléctrico se puede apreciar en la ilustración 10:

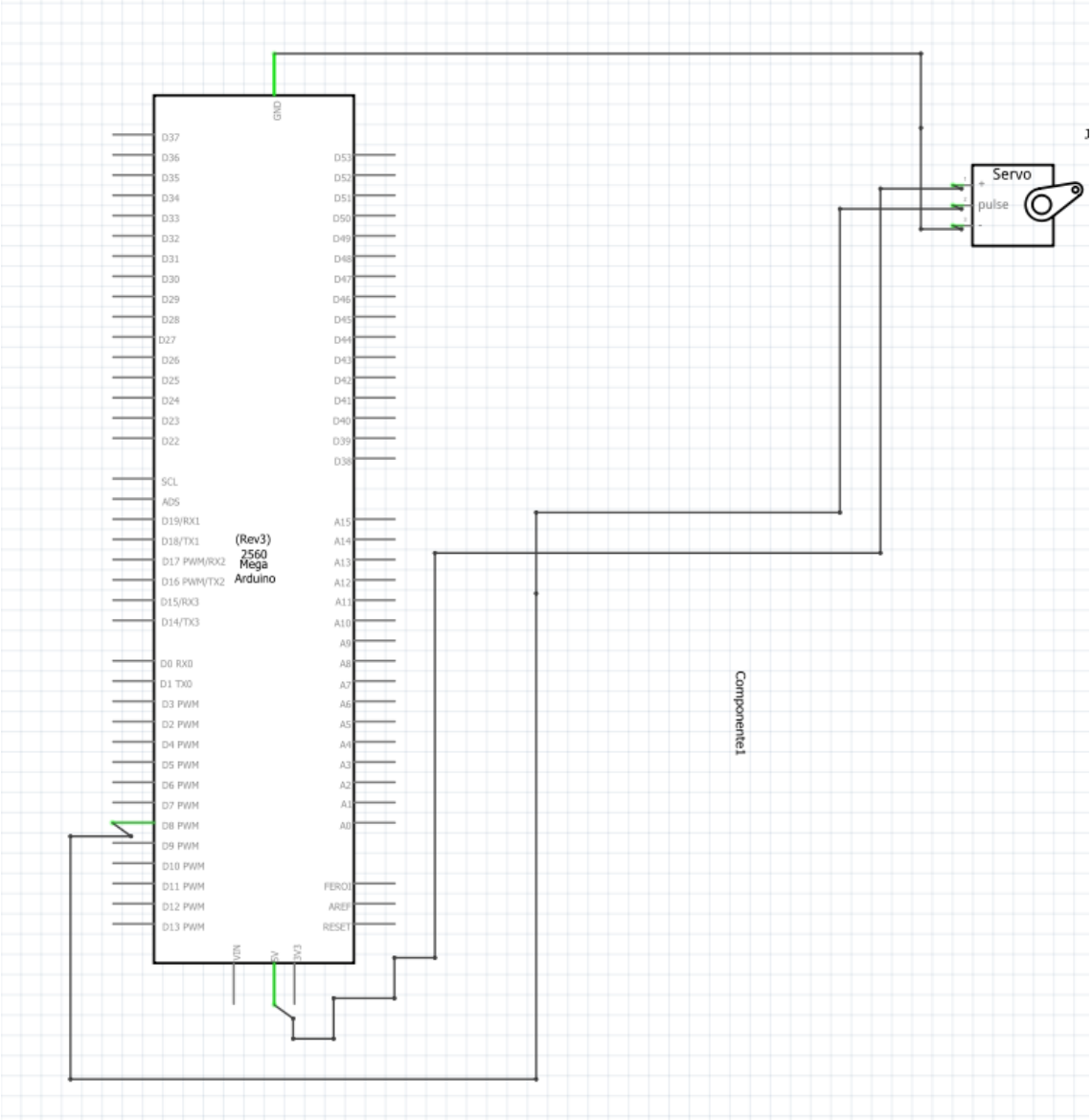

Ilustración 10 Esquema eléctrico de la conexión entre el servomotor y el Arduino.

Con el montaje realizado, se carga en la placa de Arduino el programa creado exclusivamente para hacer funcionar el servomotor y se ejecuta.

Posteriormente, a base de probar y modificar los parámetros del programa se obtiene la configuración más adecuada del servomotor. Este funciona de manera correcta.

El inconveniente que se observa es la velocidad angular máxima con la que es capaz de oscilar el servomotor. Uno de los objetivos que debe cumplir el servomotor es crear un balanceo en la estructura que simule la vibración del edificio, para que

este entre en resonancia. Por lo tanto determinamos que el equipo no cumple con lo requerido.

Consecuentemente se opta por buscar otro servomotor que no tenga tanta fuerza pero pueda realizar giros lo suficientemente rápidos como para hacer vibrar la estructura.

2.3.4 Servomotor “SG90 micro servo”.

En la ilustración 11 aparece el equipo mencionado, el cual se trata de la opción elegida para el proyecto. Su *datasheet* se encuentra en el Anexo, lugar donde se pueden consultar y corroborar ciertos datos de interés que serán utilizados a continuación para estudiar este equipo.

Inicialmente se observa que la tensión de alimentación necesaria para hacer funcionar este dispositivo es de 4.8v, recordando que es justamente la que aparece en la salida del puerto USB del equipo que utilizamos para alimentar el Arduino.

Ilustración 11 Servomotor "SG90 micro servo" (Imagen obtenida de la web, ver Anexo).

Por lo tanto se elimina la necesidad de utilizar alimentación externa para la placa de Arduino. Realizamos el montaje oportuno.

La conexión se realiza como se observa en la imagen:

- Cable rojo del servo con el PIN de 5v de la placa de Arduino (alimentación).
- Cable marrón del servo con el PIN GND de la placa Arduino (Tierra).
- Cable naranja del servo con el PIN 8 PWM (señal de control).

Destacar que la elección del PIN 8, en cuanto a la señal de control se refiere, es fruto de como se ha desarrollado el programa en la plataforma de Arduino, cuyo código se incluye en el Anexo bajo el nombre de ServomotorExperimento. La conexión se realiza como se indica en la ilustración 13.

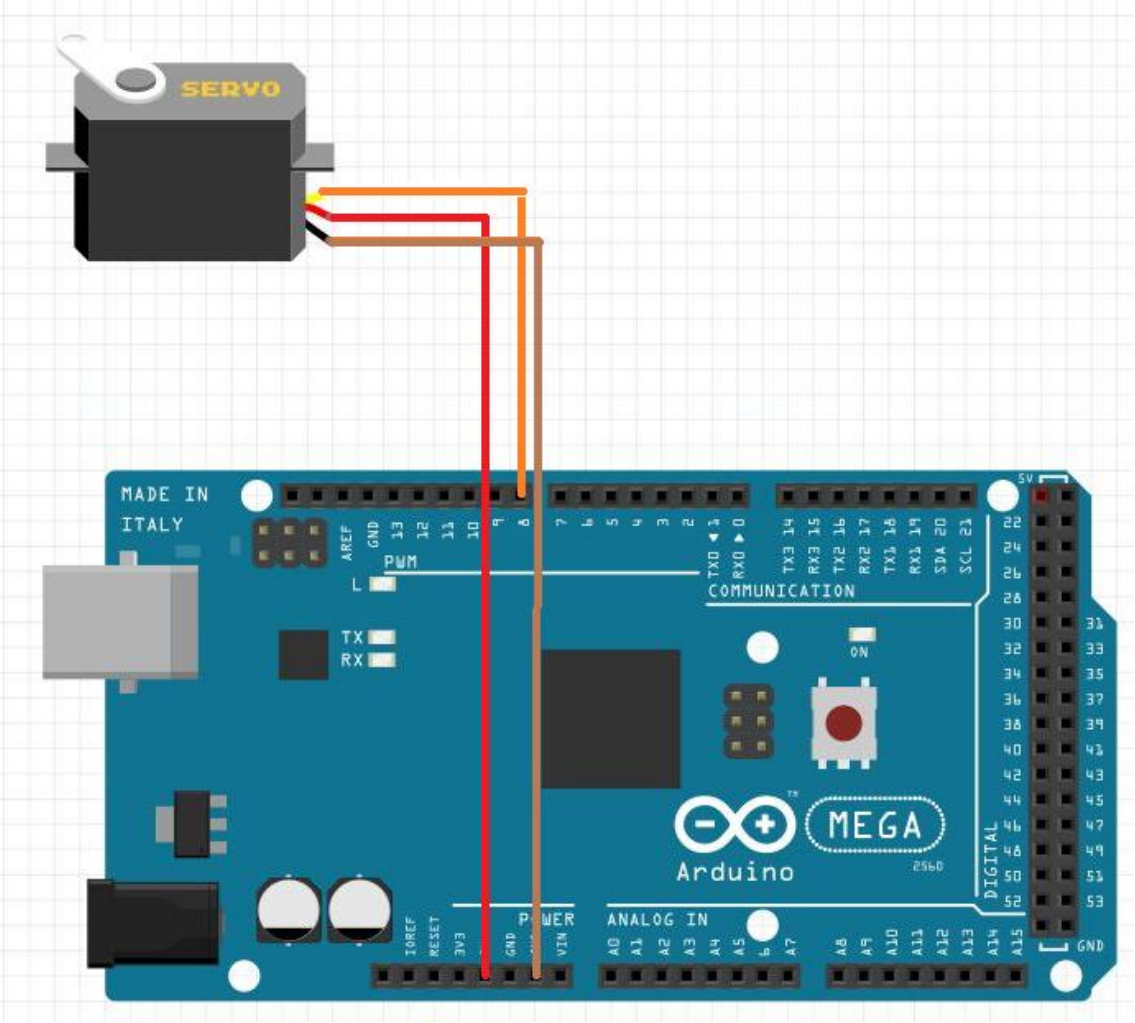

Ilustración 12 Conexión entre Servomotor y Arduino.

Una visión más clara del esquema eléctrico se aprecia en la ilustración 14:

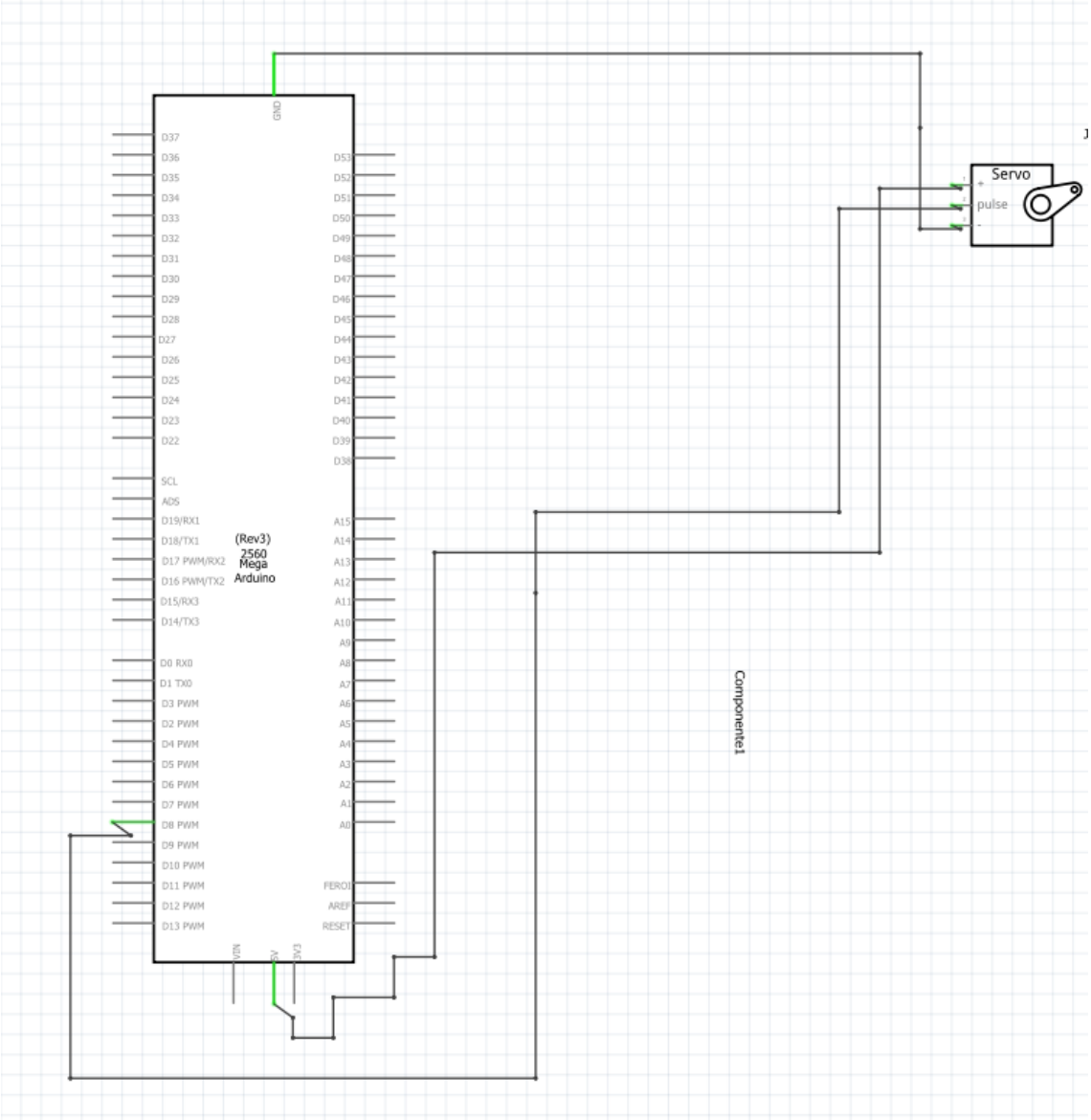

Ilustración 13 Esquema eléctrico de la conexión entre el servomotor y Arduino.

Una vez que se realizan las conexiones de manera correcta se carga en la placa de Arduino el programa creado exclusivamente para hacer funcionar el servomotor y se ejecuta.

El servomotor funciona de manera correcta, por lo que realizando los ajustes oportunos en el programa tanto del ángulo de giro, así como de los tiempos de espera, se consigue determinar la frecuencia máxima y la frecuencia mínima aproximado de trabajo que posee este equipo.

Con el objetivo de corroborar que cumple con las exigencias necesarias se realiza el estudio experimental basado en un ensayo de prueba y error para determinar los puntos anteriormente mencionados.

Modificamos dos variables en el programa de Arduino:

- Ángulo girado: Determina los grados que el servomotor va a girar desde su posición inicial.
- Tiempo de espera: Viene determinado por los comandos *Delay* del programa, donde se especifica el tiempo de espera en microsegundos, que permiten que el servomotor alcance el ángulo que se introduce con el parámetro anterior.

Después de realizar el experimento, obtenemos los siguientes resultados que se ofrecen en la tabla 1:

Amplitud	Frecuencia máxima (Hz)	Ángulo de giro (º)	Tiempo de espera (µs)	Periodo (s)
Mínima	**45,45**	**7**	**22**	**0,022**
Máxima	**1,25**	**180**	**800**	**0.800**

Tabla 1 Resultados obtenidos variando ángulo girado y tiempo de espera.

Si analizamos los datos expuestos se observa que ancho de banda de frecuencia es de 45,45 Hz -1,25 HZ aproximadamente, por lo que sería suficiente, a priori, para hacer vibrar la maqueta del edificio y que está entre en resonancia.

Una vez superadas las pruebas y problemas que se han ido presentando, se elige este equipo para su inclusión en la maqueta descrita en este proyecto.

2.4 Acelerómetros.

2.4.1 Acelerómetros. Introducción.

En el proyecto que se tiene entre manos se van a utilizar acelerómetros con el objetivo de medir las aceleraciones que soporta la estructura que ejemplifica el edificio.

Los acelerómetros o sensores de aceleración, proporcionan una señal eléctrica según varíe una magnitud física. La magnitud física en este caso puede ser una aceleración o una vibración.

Estos dispositivos electromecánicos permiten detectar fuerzas estáticas y dinámicas de aceleración. Con fuerza estática hacemos referencia a la gravedad de la tierra y con fuerza dinámica se hace referencia al movimiento y a las vibraciones.

Los acelerómetros pueden medir la aceleración en uno, dos o tres ejes. En cuanto a su funcionamiento, se va a comentar de manera muy básica los que contienen internamente placas capacitivas para producir la señal eléctrica, debido a que es el que usaremos en el proyecto. No obstante, existen otros modelos como los basados en materiales piezoeléctricos los cuales generan una pequeña carga eléctrica cuando se deforma el material fruto de una tensión mecánica (por ejemplo aceleración).

Las placas pueden ser fijas o desplazarse según un pequeño resorte que varía su longitud en función de la aceleración o vibración que experimenta el equipo. Las placas se mueven unas respecto a las otras lo que hace variar la capacidad, permitiendo general una señal eléctrica en función de la capacitancia mencionada. De esta manera se consigue cuantificar la aceleración del equipo.

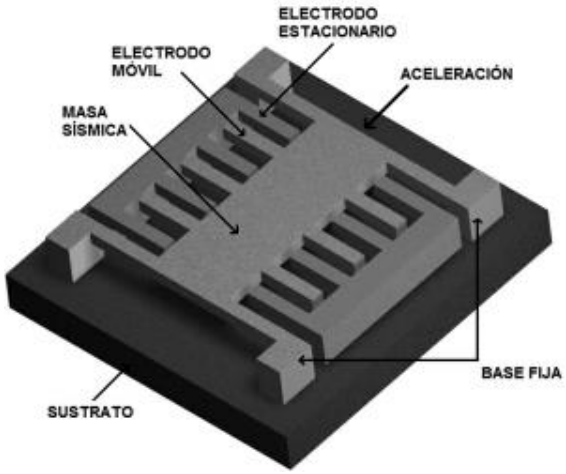

Ilustración 14 Disposición de la placas en un acelerómetro capacitivo (imagen obtenida en la web, ver Anexo).

En la ilustración 15 se representa una forma de situar las placas internamente en un acelerómetro, lo que arroja luz a la explicación.

2.4.2 Acelerómetro MMA7361LC. Pololu 0j7193.

Antes de hablar sobre las características de este acelerómetro, cabe destacar que dicho equipo se monta sobre un circuito impreso para mejorar la accesibilidad de las conexiones así como de las funciones o capacidades que este dispositivo es capaz de ofrecer.

Por consiguiente, el nombre del equipo que forma el conjunto antes mencionado es “Pololu 0j7193” el cual se puede observar en la ilustración 11.

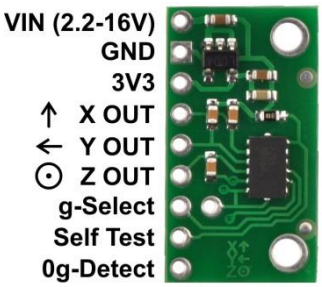

Ilustración 15 Acelerómetro " MMA7361LC" (Imagen obtenida de la web, ver Anexo).

Junto con el circuito impreso se incluyen una serie de clavijas de ángulo recto, las cuales son soldadas en los *pads* que se aprecian en la imagen, con el objetivo de facilitar la conexión y obtener una mayor accesibilidad a las prestaciones del dispositivo.

Los pines que se van a utilizar en el funcionamiento de la maqueta son los siguientes:

- VIN: PIN de alimentación del equipo (2.2 – 16 V).
- GND: PIN puesta a tierra del dispositivo.
- Eje Y: Aceleración que se experimenta en el eje Y.
- Eje Z: Aceleración que se experimenta en el eje Z.

La conexión con Arduino se realizaría como en la ilustración 17:

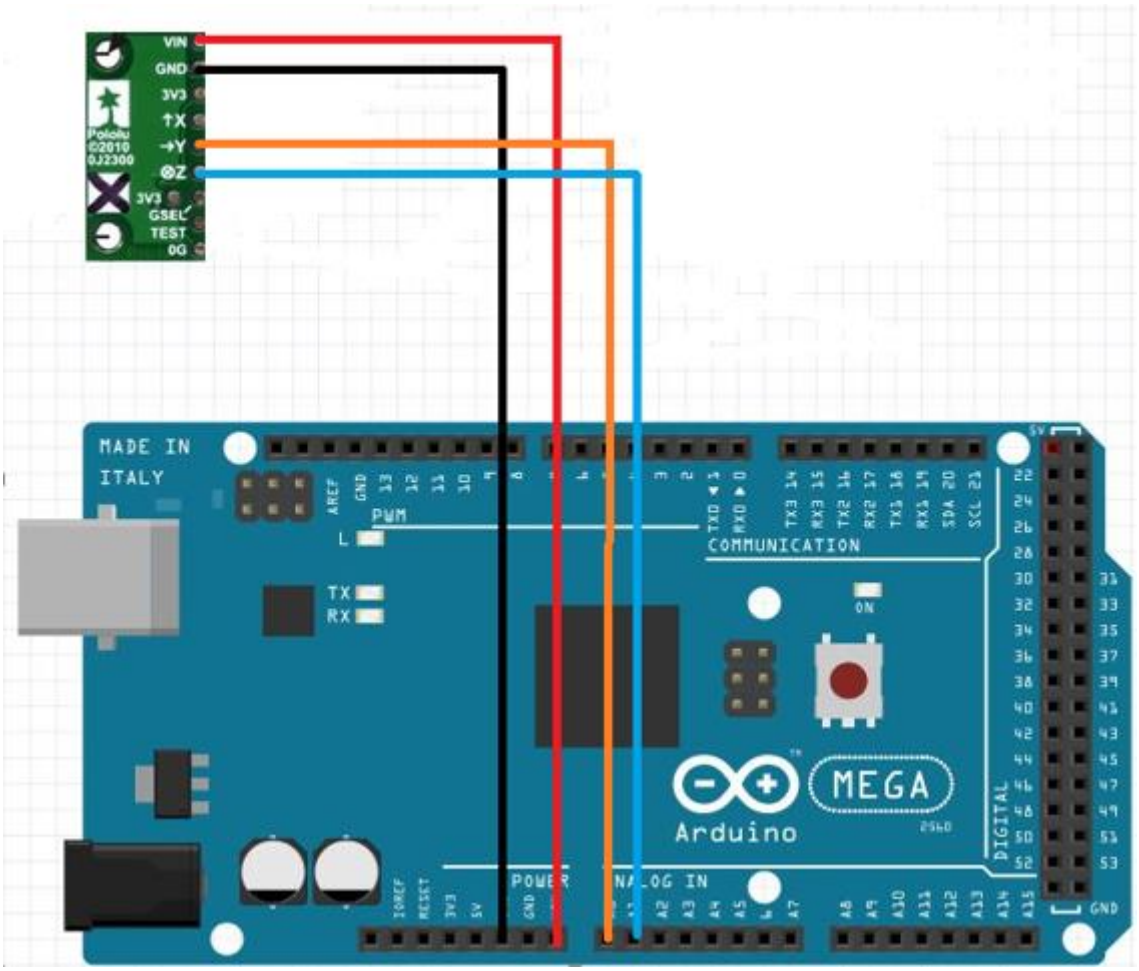

Ilustración 16 Conexión Acelerómetro con Arduino.

La tensión de salida de los pines de eje Z y eje Y van a ser valores de entre 0 y 3,3V. Las aceleraciones positivas, en el sentido que marca la flecha, viene indicado por valores de tensión superiores al punto medio de tensión del pin de 3,3V. Las aceleraciones negativas, en el sentido contrario que marcan las flechas, viene indicado por los valores de tensión inferior al punto medio de tensión del pin de 3.3V. No obstante, el valor de tensión que entra por los distintos pines de entrada de Arduino referente a los ejes, son codificados en una escala de 0 a 1023 valores proporcionalmente al valor de la tensión registrada.

Los demás pines con sus respectivas funciones no serán necesarios para alcanzar el objetivo que se espera de este equipo. No obstante se va a indicar de manera muy básica que ofrecen los pines que quedan sin conectar:

- 3V3: Indica la tensión real que tenemos para calcular el punto medio de aceleraciones positivas / negativas.
- Eje X: Aceleración que se experimenta en el eje X.
- *g-Setect*: Se utiliza para cambiar la sensibilidad del dispositivo.
- *Self Test*: La función de autoprueba para verificar funcionamiento correcto.
- *0g-Detect*: Indica caída libre cuando los tres ejes marcan gravedad 0.

La corriente que ofrecen los pines de salida de este dispositivo es de 50mA. Indicar que se puede ampliar la información que se ofrece entrando en los lugares web del Anexo, donde se incluyen los *datasheet.*

2.5 Raspberry.

Raspberry es un pequeño ordenador adjuntado en una placa de circuito impreso con reducidas dimensiones, el cual abre un campo de posibilidades. Raspberry destaca entre otras cosas por la utilidad de sus servicios a un bajo coste de adquisición. Posee conector HDMI, un conector LAN, conector USB, lector de tarjetas miniSD que funciona como memoria interna etc.

Este mini ordenador ofrece la posibilidad de funcionar con un software basado en lenguaje Python, Scratch, Sonic Pi, Java etc.

El modelo que se ha elegido se denomina Raspberry Pi 3 Model B, adquirido en conjunto con una pantalla LCD de 5 pulgadas. El equipo mencionado es el que aparece en la ilustración 18.

Ilustración 17 Raspberry Pi Model 3 (Imagen obtenida de la web, ver Anexo).

El microcontrolador es incorporado a una pantalla LCD de 5 pulgadas, opción muy recomendable para cumplir el objetivo de utilizar este equipo como interface (ilustración 18).

Ilustración 18 Raspberry Pi incluida en pantalla LCD de 5 pulgadas (imagen obtenida de la web, ver Anexo).

Para entrar más en concreto en las características de Raspberry se puede consultar el *datasheet* incluido en el Anexo.

3. Calibración de los componentes que forman el sistema.

3.1 Calibración. Introducción.

Una vez conocido el funcionamiento de los distintos elementos que conforman el proyecto, toca resolver los problemas de calibración que van apareciendo en la maqueta.

La calibración se va a realizar de manera específica tanto para los servomotores como para los acelerómetros.

En cuanto a los servomotores, se podrá controlar el ángulo girado y en el caso del servomotor 1 se podrá controlar la frecuencia a la que realiza el ciclo de movimiento. Por otro lado se tendrá que poner especial interés en el posicionamiento de los servomotores para que los elementos móviles que accionen no choquen con la estructura y provoquen daños en el propio servomotor o en la estructura.

En el caso de los acelerómetros, la calibración que se va a realizar consiste en la conversión de la tensión que envían los acelerómetros a Arduino, para obtener valores reales de aceleración en la salida por pantalla de los datos.

3.2 Calibración. Servomotores.

En primer lugar se tratará de manera completa la calibración de los tres servomotores utilizados en la maqueta, mostrando especial interés en los puntos que se han descrito en la introducción.

3.2.1 Calibración servomotor de excitación (servomotor 1).

Este servomotor tiene una calibración distinta a los otros dos utilizados en la maqueta, controlando dos aspectos como son:

- Ángulo girado y posición.
- Periodo de oscilación del ciclo de giro.

De manera muy básica, se va a estudiar la manera de posicionar el servomotor adecuadamente, con el objetivo de no dañar el equipo mencionado por choques de los elementos móviles que hace girar el servomotor y las paredes fijas. Por otro

lado, se comentará como se controla el periodo de oscilación de giro y como se optimiza su espacio de trabajo en función de las características del equipo elegido.

3.2.1.2 Calibración servomotor de excitación. Periodo.

En el caso del servomotor de excitación, debido a que será posible controlar el periodo en el que realiza el ciclo de movimiento que hace vibrar la estructura, se ha realizado un cálculo aproximado de la frecuencia de excitación óptima que hace entrar a la estructura en resonancia.

Para ello se ha realizado un experimento que consiste en hacer vibrar la estructura de manera brusca. Cuando la estructura está vibrando se recogen los datos mediante dos programas:

- Lecturas recogidas por Arduino con origen en el acelerómetro de la maqueta.

- Lecturas recogidas por "SIRIUS" con origen en acelerómetros externos.

Se recogen los datos de ambos sistemas en función del tiempo de muestreo, con el objetivo de graficarlos buscando una aproximación real del acelerómetro implicado en la maqueta a partir de los acelerómetros externos, los cuales son de carácter profesional (más precisos).

Destacaremos que en este apartado los datos y los tiempos de lectura han sido acondicionados como se explicará en el apartado de calibración de acelerómetros, utilizando el resultado de ese trabajo.

Para el cálculo de la frecuencia de resonancia de la estructura, solo es necesario tener en cuenta los datos acondicionados obtenidos por el acelerómetro de la maqueta.

A continuación se expone una pequeña parte de la muestra de los datos recogidos a partir de los cuales se generará la gráfica (tabla 2):

Tiempo	Medida Arduino
1,3748	**409**
1,3800	**415**
1,3852	**414**
1,3904	**416**
1,3956	**419**

1,4008	422
1,4060	423
1,4112	419
1,4164	422
1,4216	420
1,4268	419
1,4320	420
1,4372	416
1,4424	426
1,4476	417
1,4528	410
1,4580	409
1,4632	408
1,4684	401
1,4736	398
1,4788	401
1,4840	407
1,4892	392
1,4944	397
1,4996	406
1,5048	398
1,5100	395
1,5152	398
1,5204	399
1,5256	392
1,5308	387
1,5360	388
1,5412	387
1,5464	380
1,5516	373
1,5568	378
1,5620	371
1,5672	358
1,5724	350
1,5776	347
1,5828	341
1,5880	331
1,5932	324
1,5984	323
1,6036	314
1,6088	306
1,6140	306
1,6192	309
1,6244	304
1,6296	292
1,6348	294
1,6400	301
1,6452	302
1,6504	297
1,6556	303
1,6608	311
1,6660	290
1,6712	289
1,6764	291
1,6816	298
1,6868	285
1,6920	278
1,6972	279

1,7024	**278**
1,7076	**275**
1,7128	**275**
1,7180	**276**
1,7232	**280**
1,7284	**276**
1,7336	**274**

Tabla 2 Muestras de aceleración recogidas por Arduino en función del tiempo.

Una vez obtenida una muestra de datos lo suficientemente grande se representa (ilustración 19):

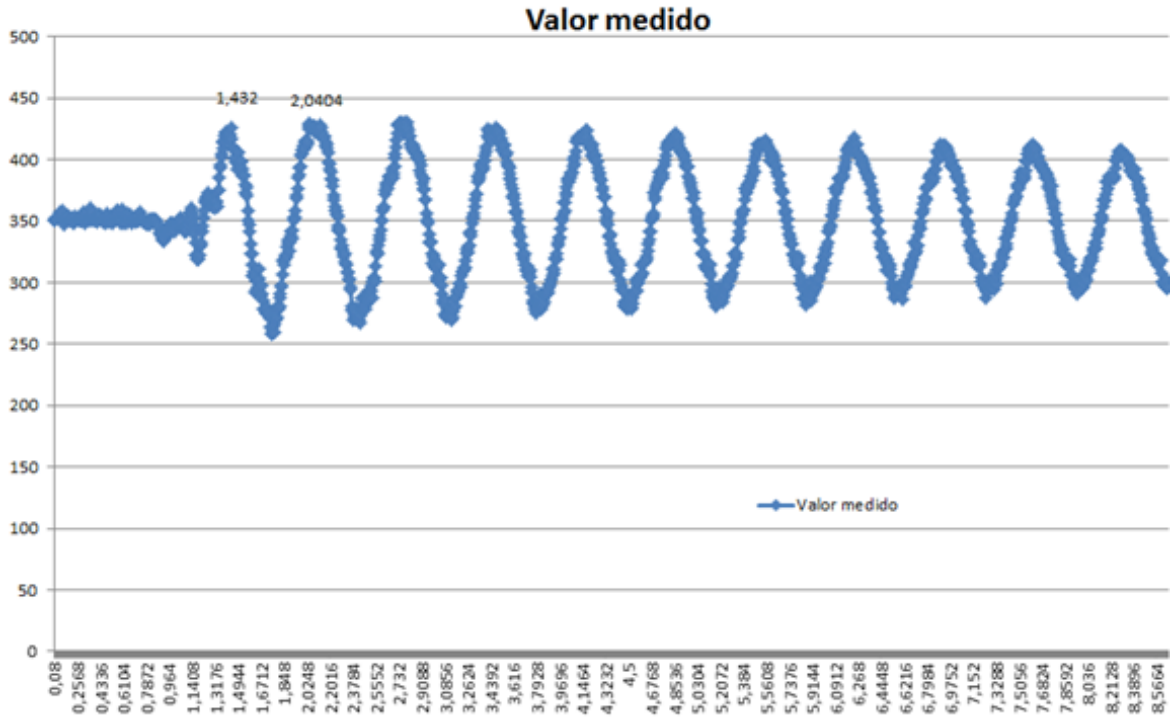

Ilustración 19 Gráfica del conjunto de medidas de aceleración recogidas por Arduino en función del tiempo.

En el anterior gráfico se muestra la medida acondicionada leída por Arduino que genera el acelerómetro que se encuentra en la estructura en movimiento, representada en función del tiempo en el que se recogen las muestras.

No es necesario obtener un conjunto de datos muestreados tan amplios, no obstante, dicho conjunto es válido para obtener el periodo de excitación "T" buscado.

Como se observa en la imagen se ha indicado el tiempo en el que se obtiene el punto que forma el máximo de las crestas (*t0* y *t1*) de la señal de vibración de la

estructura. Los dos puntos a los que hacen referencia los tiempos indicados forman un periodo de oscilación de la estructura, por lo que calculando ese intervalo se obtendrá una aproximación al periodo con el que la estructura entra en resonancia (ecuación 1).

$$T = t1 - t0 = 2{,}0404 - 1{,}432 = 0{,}6084s \qquad (1)$$

El dato que se ha calculado será utilizado como periodo inicial en el programa de control de servomotores que se ha creado para ser utilizado por Arduino, introduciéndose el valor en el programa en milisegundos. El nombre del programa es Servomotores (ver Anexo).

Por otro lado, se ha incluido la opción en la maqueta de variar el tiempo en el que se realiza el ciclo de giro del péndulo (su periodo). Se tiene que tener en cuenta que para que un ciclo de giro sea correcto, visiblemente, el péndulo no debe detenerse en ningún punto.

Puesto que el servomotor tiene una velocidad fija en cuestión ha recorrido de ángulos, aumentando demasiado el periodo, el servomotor tiene tiempo suficiente de alcanzar la posición programada y esperar en ese punto a la siguiente instrucción. Con ello el ciclo de giro del péndulo quedaría interrumpido durante ese tiempo de espera y se mantendría inmóvil, produciéndose un ciclo de giro incorrecto.

En el caso de que el periodo sea pequeño, el péndulo en el desarrollo del ciclo de movimiento no alcanzara las posiciones programadas, pudiendo aparecer el problema de centrado que se expondrá en el próximo apartado (ver 3.2.1.2). El equipo que utilizamos funciona de manera correcta, no apreciándose el problema mencionado en el desarrollo de su funcionamiento normal, razón por la cual no se tendrá en cuenta.

El resultado producido por la variación del periodo consiste en cambiar la vibración de la estructura. El periodo óptimo que hace entrar en resonancia a la maqueta es el calculado con anterioridad.

Por tanto y debido a las razones aportadas en los párrafos anteriores inferimos un rango de valores válidos para que el funcionamiento del equipo sea adecuado y correcto.

En cuanto al cálculo de este intervalo, la manera de proceder, fue completamente experimental. Inicialmente trabajamos con el siguiente ciclo de giro de péndulo:

$$90^{o} \rightarrow\ 120^{o} \rightarrow 60^{o} \rightarrow 120^{o}$$

Inicialmente, cuando se conecta el Arduino, se autocalibra el servomotor a una posición de 90º automáticamente, realizándose posteriormente el ciclo repetitivo 120º - 60º - 120º - 60º.

Aumentando el valor del periodo se alcanza un punto en el que el péndulo visiblemente se queda parado en los extremos. Una vez detectado el periodo para el cual se da esta situación, se deberá ir disminuyendo el valor del periodo hasta que visiblemente el ciclo se realice de manera continua. En el caso del periodo mínimo, se puede obtener disminuyendo el valor del periodo hasta que apenas se aprecie movimiento. En nuestro caso se ha optado por un valor de 50 ms (puede calibrarse a valores inferiores).

En el caso del intervalo antes citado se obtienen los siguientes resultados (tabla3):

Periodo máximo (ms)	Periodo de resonancia(ms)	Periodo mínimo (ms)
690	**608**	**50**

Tabla 3 Tabla con el intervalo de periodos de funcionamiento válidos para el servomotor 1.

Con el objetivo de optimizar el control sobre el periodo del ciclo de giro del péndulo se buscó la manera de aumentar el intervalo de valores válidos. La solución obtenida consistió en aumentar el número de grados que formaban el ciclo de giro. El ciclo de giro final que se utiliza en la maqueta es el siguiente:

$$90^{o} \rightarrow\ 180^{o} \rightarrow 0^{o} \rightarrow 180^{o} \rightarrow 0^{o}$$

Inicialmente, cuando se conecta el Arduino, se autocalibra el servomotor a una posición de 90º automáticamente, realizándose posteriormente el ciclo repetitivo 180º- 0º-180º -0º. El ciclo de giro final utilizado en el programa aumenta el intervalo de periodos válidos (tabla 4):

Periodo máximo (ms)	Periodo de resonancia(ms)	Periodo mínimo (ms)
800	**608**	**50**

Tabla 4 Tabla con el intervalo final de periodos de funcionamiento válidos para el servomotor 1.

Por limitaciones del equipo, el intervalo válido de periodos no puede ser más amplio ya que el servomotor posee un ciclo de giro máximo 0º - 180º (ver hoja de características Anexo), y es con el aumento de amplitud del ciclo de giro con el que se consigue un mayor intervalo de valores.

Por último, destacar que por debajo de 800 ms, el ciclo de giro que realiza el servomotor se va reduciendo, es decir, va disminuyendo el número de grados recorridos durante su ejecución por falta de tiempo.

3.2.1.2 Calibración servomotor de excitación. Ángulo girado y posición.

Uno de los aspectos más importantes en cuanto a la calibración del servomotor consiste en adecuar el ángulo girado al periodo del ciclo de giro. Si el periodo es demasiado pequeño, el servomotor no conseguirá alcanzar los puntos programados (no conseguirá realizar el giro ordenado de manera completa).

Una de las posibles consecuencias surgidas sino se tiene en cuenta este aspecto, consiste en el desajuste de la posición del péndulo que mueve el servomotor. El ciclo de giro que puede realizar el servomotor de excitación por ejemplo en nuestra maqueta es el siguiente:

$$90º \rightarrow 120º \rightarrow 60º \rightarrow 120º$$

Inicialmente, cuando se conecta el Arduino, se autocalibra el servomotor a una posición de 90º automáticamente, realizándose posteriormente el ciclo repetitivo 120º- 60º-120º -60º.

Una vez expuesto el funcionamiento, en el caso de que haya una pequeña diferencia en el ángulo girado cuando el periodo de oscilación no es el adecuado (no se produce de manera completa el giro programado por falta de tiempo), puede provocar que el péndulo se descentre. Si existiese una mínima diferencia del número de grados recorridos por el péndulo en el semiperiodo 120º- 60º con respecto del número de grados recorridos en el semiperiodo 60º-120º, con sucesivas iteraciones el péndulo se iría descentrando acercándose cada vez más a uno de los extremos del ciclo de giro (60º o 120º), presentando un funcionamiento erróneo.

En nuestro caso no se ha incorporado una línea de acción que centre el ciclo de giro en cada iteración debido a que el funcionamiento de la maqueta no va a ser prolongado, por lo tanto no se va a apreciar el descentrado. Por otro lado se ha comprobado que los servomotores poseen un funcionamiento correcto (no se da el caso explicado). Debido a estas razones se ha decidido apostar por que el ciclo de movimiento del péndulo sea más suave evitando alcanzar posiciones intermedias

que provoquen tirones en su movimiento, en vez de asegurar la posición de centrado (se necesitaría asegurar que el péndulo siempre llegase a 90º en el desarrollo del ciclo normal). Esta es una de las líneas futuras de investigación para mejorar el funcionamiento y las características de la maqueta que se incluirán en el apartado de línea de mejoras futuras.

Para el periodo de resonancia de la estructura (T=0.6084s), mediante los datos obtenidos del "*dataheet*" del servomotor SG90, se puede comprobar que el ciclo de giro no se realiza de manera completa. La velocidad del servomotor es de 0,1 s /60º. El periodo de resonancia no es lo suficientemente grande para que el servomotor alcance la posición programada (ecuación 2).

$$\text{Grados} * 0{,}1s / 60º \qquad (2)$$

Como se explicaba en el apartado anterior, el ciclo de giro que está programado es:

$$90º \rightarrow 180º \rightarrow 0º \rightarrow 180º$$

Inicialmente, cuando se conecta el Arduino, se autocalibra el servomotor a una posición de 90º automáticamente, realizándose posteriormente el ciclo repetitivo 180º - 0º - 180º - 0º.

El semiperiodo está formado por 180º:

$$180º * 0{,}1s / 60º = 0{,}3s$$

El periodo está formado por 360º:

$$360º * 0{,}1s / 60º = 0{,}6s < 0{,}608s$$

Teóricamente el servomotor podría realizar el ciclo de giro de manera completa como se ha demostrado con los anteriores cálculos. Realizando la comprobación de manera experimental se observa que para 0.608s el ciclo de giro no se completa.

Esta situación se produce porque hay que añadir un tiempo extra producido por la computación de las instrucciones del programa de Arduino. Como se exponía en el apartado anterior (3.2.1.1), para el ciclo de giro 90º → 180º → 0º → 180º, obtenemos experimentalmente los siguientes valores de periodo de ejecución (tabla 5):

Periodo máximo (ms)	Periodo de resonancia(ms)	Periodo mínimo (ms)
800	608	50

Tabla 5 Tabla con el intervalo final de periodos de funcionamiento válidos para el servomotor 1.

Donde para 800 ms se completa el ciclo de giro, mientras que para un valor de periodo menor el número de grados recorridos es menor proporcionalmente a la disminución de periodo.

Otro de los aspectos claves en la calibración de este servomotor consiste en el posicionamiento adecuado en su montaje en la estructura. Inicialmente el servomotor "SG90" al ser conectado a la corriente (sin haberle dado ninguna orden programada) busca su posición central, es decir, 90º con el objetivo de centrarse y disponer 90º de giro en ambos sentidos.

El método para situar adecuadamente el servomotor de manera física sin el aspa de plástico, consiste en posicionar sobre la estructura el dispositivo y dejarlo fijo.

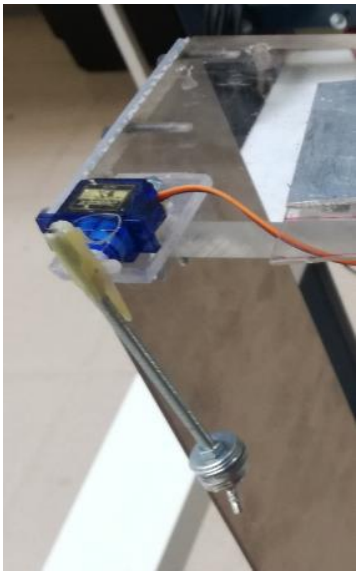

Ilustración 20 Posicionamiento del servomotor excitador en la estructura (Servomotor 1).

Una vez realizado este paso, conectar el servomotor a la corriente, con el objetivo de que alcance su posición de 90º (se autocentre). El aspa de plástico se puede colocar en el engranaje de salida del servomotor en distintas posiciones. En nuestro caso, cuando el servomotor se encuentra en 90º, dispusimos el aspa de plástico (en su lado largo) en la misma dirección y sentido que el alto de la estructura (ilustración 20). Una vez unido el péndulo al aspa de plástico, este quedó posicionado perpendicular al suelo. Con ello se consigue que el péndulo pueda recorrer 90º en cada sentido de giro. El montaje es el que se observa en la imagen anterior.

3.2.2 Servomotor de amortiguamiento del péndulo (servomotor 2).

El método para situar adecuadamente el servomotor de amortiguamiento del péndulo de manera física sin el aspa de plástico, consiste en posicionar sobre la estructura el dispositivo y dejarlo fijo.

Ilustración 21 Servomotor de amortiguamiento del péndulo (servomotor 2).

Una vez realizado este paso se conecta el servomotor a la corriente, con el objetivo de que alcance su posición de 90º (se autocentre). El aspa de plástico se puede colocar en el engranaje de salida del servomotor en distintas posiciones. Una vez que el servomotor se encuentra en la posición de 90º (debido al autocentrado), se coloca la estructura formada por la chapa y el brazo de plástico en el aspa, en la

posición alejada donde no influencia al péndulo de disipación (ilustración 21). Es decir, se coloca la chapa en la posición fija que va a tener cuando no interviene en el funcionamiento del sistema. Posteriormente y en función del sentido de giro que se necesite para alcanzar la posición donde la chapa interviene en el funcionamiento del sistema, se programa en el equipo el giro correspondiente. En nuestro caso la posición de actuación de la chapa se produce cuando el servomotor alcanza la posición de 0º.

Cabe destacar la importancia de tener especial cuidado en las primeras pruebas de funcionamiento del sistema de la placa, con los choques que se pueden producir entre los elementos móviles y las paredes fijas de la maqueta. Para ello se recomienda programar puntos de posicionamiento del servomotor anteriores a la posición final deseada, e ir programando posiciones cada vez más cercana a la deseada. El montaje es el que se observa en la imagen anterior.

3.2.3 Servomotor de activación del péndulo de disipación (servomotor 3).

El método para situar adecuadamente el servomotor de activación del péndulo de disipación de manera física sin el aspa de plástico, consiste en posicionar sobre la estructura el dispositivo y dejarlo fijo.

Ilustración 22 Servomotor de activación del péndulo de disipación (servomotor 3).

Una vez realizado este paso se conecta el servomotor a la corriente, con el objetivo de que alcance su posición de 90º (se autocentre). El aspa de plástico se puede colocar en el engranaje de salida del servomotor en distintas posiciones. Una vez que el servomotor se encuentra en la posición de 90º (debido al autocentrado), se coloca el aspa de plástico con las sujeciones del péndulo de manera paralela al péndulo (ilustración 22). En la posición inicial el péndulo no debe actuar en el sistema, por lo que a 90º se encuentra encerrado por la estructura. Una vez determinado este punto inicial hay que tener especial cuidado con el sentido de giro del servomotor. En nuestro caso y según la posición del servomotor se necesita que este se situé en 180º, posición de apertura con la que se permite la intervención del péndulo en el funcionamiento de la maqueta.

Cabe destacar la importancia de tener especial cuidado en las primeras pruebas de funcionamiento del sistema de la placa, con los choques que se pueden producir entre los elementos móviles y las paredes fijas de la maqueta. Para ello se recomienda programar puntos de posicionamiento del servomotor anteriores a la posición final deseada, e ir programando posiciones cada vez más cercana a la deseada. El montaje es el que se observa en la imagen anterior.

3.2.4 Disposición de los servomotores en la maqueta.

Una vez determinada la manera de posicionar los servomotores en la maqueta, junto con la calibración de los mismos, es decir, se ha comprobado que funcionan de manera correcta individualmente, se necesita que funcionen como una unidad conjunta. Para ello se delega en una placa de Arduino el control de los tres servomotores bajo la tutela de un mismo programa, al cual denominaremos Servomotores (Ver Anexo).

Este programa basa su funcionamiento en los programas individuales, utilizados para comprobar el funcionamiento de cada servomotor por separado, implementando un control conjunto que permite activar o desactivar las tareas programadas para cada servomotor.

Por último indicar que en el Anexo se añade el programa mencionado, donde se expone su explicación, para buscar una fácil interpretación y comprensión.

3.3 Calibración de acelerómetros.

3.3.1 Introducción. Calibración de acelerómetros. Posicionamiento.

En este apartado se desarrollará la manera de proceder para obtener un calibrado de los acelerómetros adecuado para la maqueta. Las lecturas que realiza Arduino de los acelerómetros corresponden a variaciones de tensión. Estas variaciones de tensión son convertidas por el microcontrolador a una escala que comprende desde 0 hasta 1023, de una manera proporcional a la variación de tensión escuchada.

Por tanto, los datos que se pueden obtener por parte de Arduino de manera directa serán valores de entre 0 y 1023. Estos datos serán transformados a una escala más comprensible como son los m/s^2. Una vez que se consigue el modelo de transformación de escalas de manera teórica, este deberá ser comprobado en el equipo real para verificar su utilidad.

Los pasos para determinar la calibración de los acelerómetros vienen determinados por:

- Cálculo teórico del modelo de cambio de escala.
- Comprobación en el equipo real del modelo teórico

En cuanto al posicionamiento de los acelerómetros, estos se superponen de manera fija en las dos "baldas" que incluye la maqueta. Los acelerómetros implicados registrarán las aceleraciones que se produzcan en los ejes Z e Y respectivamente. El eje Y en la maqueta representará el desplazamiento paralelo del edificio respecto al suelo (aceleración en el balanceo lateral producida por los desplazamientos laterales de la maqueta). Por su parte el eje Z representará la aceleración que experimenta en sentido vertical con respecto del suelo (la variación de aceleración producida por la variación de la altura de la maqueta durante el balanceo). El posicionamiento de los acelerómetros en la maqueta se puede observar en las ilustraciones 23 y 24.

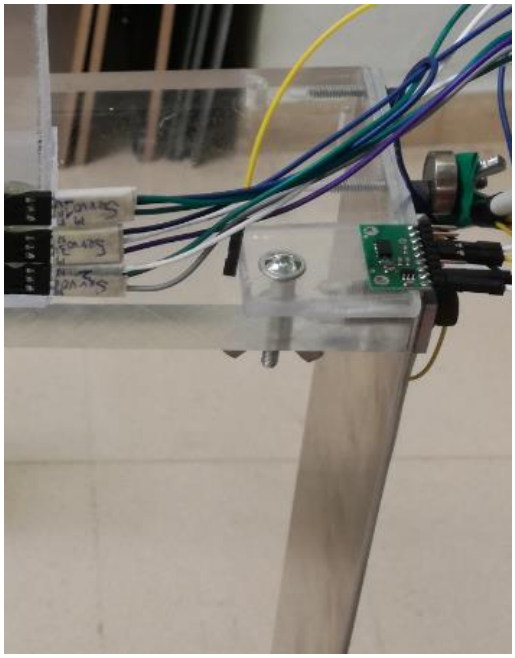

Ilustración 23 Posicionamiento del acelerómetro 1.

Ilustración 24 Posicionamiento del acelerómetro 2.

3.3.2 Cálculo teórico del modelo de cambio de escala.

Se va a proceder a explicar el conjunto de pasos que se han llevado a cabo para obtener el modelo teórico que permite realizar el cambio de escala de la aceleración medida por Arduino a m/s^2.

3.3.2.1 Obtención y representación de datos.

Para calcular el modelo teórico que se utilizará para realizar el cambio de escala de los valores aportados por Arduino (escala de 0 a 1023) a una escala útil y que permita una mejor comprensión, como son los m/s^2, se necesitarán dos equipos de toma de datos que trabajen a la par. Estos equipos realizarán la adquisición de datos, es decir, medirán las aceleraciones experimentadas por la maqueta a la vez y compartiendo una línea de tiempo común. Cabe destacar que la adquisición de datos se producirá bajo un movimiento de la estructura de la maqueta producido externamente.

Los equipos utilizados para realizar la adquisición de datos son:

- Lecturas recogidas por Arduino con origen en el acelerómetro inmerso en la maqueta.

- Lecturas recogidas por "SIRIUS" con origen en acelerómetros externos.

Como se mencionó con anterioridad, se recogen los datos de ambos sistemas en función del tiempo de muestreo, con el objetivo de graficarlos buscando una aproximación real del acelerómetro implicado en la maqueta a partir de los acelerómetros externos, los cuales son de carácter profesional (más precisos).

En cuanto al tipo de acelerómetros utilizados, los que se encuentran conectados a Arduino, internamente están formados por placas capacitivas para producir la señal eléctrica. Los acelerómetros denominados externos y controlados por "SIRIUS", son del tipo piezoeléctrico. El funcionamiento de ambos equipos ha sido explicado en el apartado de acelerómetros (2.4.1).

Cabe destacar que "SIRIUS" es una herramienta software especializada en medición de la empresa DEWEsoft. El desarrollo de este tipo de herramientas ha permitido que la adquisición de datos en la medición de cualquier magnitud, se realice de manera más precisa. Con SIRIUS se ha conseguido plantear un escenario

de fácil utilización, el cual posee gran utilidad debido a su versatilidad y diversidad de equipos con los que puede trabajar.

El experimento será llevado a cabo sobre el Eje Y del acelerómetro de la maqueta. Los datos obtenidos se consideran válidos para calcular la aproximación al modelo acondicionador del eje Z y del eje Y. Es decir, suponemos que el modelo será el mismo y válido para los tres ejes de medida, aunque para el eje Z se tenga que acondicionar al valor de la gravedad.

Antes de proceder a la toma de datos surge el problema de cómo determinar el tiempo de muestreo que posee Arduino en el registro de las aceleraciones. Problema que se soluciona añadiendo el comando *"millis()"*. Este comando mide el tiempo desde el inicio del lazo de ejecución del programa de Arduino, hasta que se da por finalizado. El siguiente paso consiste en visualizar por pantalla el valor del tiempo almacenado en el comando, de manera normal.

Cabe destacar que el comando se añadió en el programa Acele_calibra (ver Anexo) con la consiguiente instrucción para que el tiempo apareciera en pantalla y se pudiese registrar, y sin la intromisión de esperas programadas (*"Delay"*). No se expone el código del programa puesto que la inclusión de dicho comando es muy sencilla. En el Anexo se introduce un enlace donde se pueden consultar ejemplos de utilización del comando mencionado.

De esta manera se realizó el experimento de toma de tiempos de muestra en cuantiosas ocasiones, dando lugar a un resultado exacto en cada una de ellas. El resultado del tiempo de muestreo de Arduino es de 0,01664s (sin la intromisión de comandos "*Delay"* en el programa). Por tanto cada medida que sea tomada en Arduino se producirá en un tiempo múltiplo de ese valor proporcional a la iteración que se este ejecutando.

Una vez solucionado el problema, se procede a la toma de datos. Como se ha mencionado con anterioridad, se utilizan los dos equipos de acelerómetros sobre la estructura. Se pone en movimiento la estructura y en el mismo instante se activa Arduino y Sirius para el registro de las aceleraciones. Aproximadamente el experimento dura 8 segundos, se obtienen 1050 muestras (todo ello referenciado a los datos obtenidos por parte de Sirius y los acelerómetros piezoeléctricos debido a que es el equipo profesional).

Los datos obtenidos se guardan y se extraen a una hoja de cálculos donde llevar a cabo su representación.

En la siguiente tabla se da a conocer una pequeña muestra del conjunto de datos que se utilizarán para formalizar la gráfica de la ilustración 25. El conjunto de datos con el que se trabaja se introducirá en los anejos, bajo el nombre de Calibración T servo 1. Las medidas iniciales se corresponden con las cuatro primeras columnas del archivo.

Las muestras que se dan a conocer en las siguientes páginas, son fruto del conjunto de datos iniciales, aplicando las modificaciones que se indican en cada apartado de manera acumulativa. Es decir, los pasos se van aplicando en la muestra del apartado anterior sucesivamente hasta alcanzar el acondicionamiento de datos adecuado. En la tabla 6 se da a conocer una muestra de los datos brutos.

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	-0,03221531	0	351
0,00833333	-0,05569101	0,01664	351
0,01666667	-0,06259563	0,03328	352
0,025	-0,04740547	0,04992	353
0,03333333	-0,03635808	0,06656	353
0,04166667	-0,01840607	0,0832	353
0,05	-0,02669161	0,09984	353
0,05833333	-0,02945346	0,11648	354
0,06666667	-0,01702515	0,13312	354
0,075	-0,037739	0,14976	357
0,08333333	-0,02254884	0,1664	355
0,09166667	-0,05983378	0,18304	356
0,1	-0,04464362	0,19968	358
0,10833333	-0,02807254	0,21632	353
0,11666667	-0,07088117	0,23296	347
0,125	-0,0142633	0,2496	352
0,13333333	-0,02945346	0,26624	351
0,14166667	-0,04464362	0,28288	352
0,15	-0,07226209	0,29952	350
0,15833333	-0,04464362	0,31616	352
0,16666667	-0,05845286	0,3328	354
0,175	-0,04050085	0,34944	354
0,18333333	-0,06811933	0,36608	352
0,19166667	-0,03635808	0,38272	350
0,2	-0,06121471	0,39936	350
0,20833333	0,00921241	0,416	352
0,21666667	-0,00735868	0,43264	349
0,225	-0,05983378	0,44928	353
0,23333333	-0,037739	0,46592	351
0,24166667	-0,04050085	0,48256	348
0,25	-0,00735868	0,4992	351
0,25833333	-0,02254884	0,51584	354
0,26666667	-0,03635808	0,53248	351
0,275	-0,06259563	0,54912	350
0,28333333	-0,06259563	0,56576	352
0,29166667	-0,0142633	0,5824	352

0,3	**-0,05154824**	**0,59904**	**352**
0,30833333	**-0,037739**	**0,61568**	**350**
0,31666667	**-0,00459683**	**0,63232**	**351**
0,325	**-0,01840607**	**0,64896**	**354**
0,33333333	**-0,01012053**	**0,6656**	**351**
0,34166667	**-0,03359623**	**0,68224**	**350**
0,35	**0,00230779**	**0,69888**	**353**
0,35833333	**-0,01012053**	**0,71552**	**357**
0,36666667	**-0,07226209**	**0,73216**	**353**
0,375	**-0,0667384**	**0,7488**	**348**
0,38333333	**0,02440257**	**0,76544**	**353**
0,39166667	**-0,06535748**	**0,78208**	**351**
0,4	**-0,02945346**	**0,79872**	**352**
0,40833333	**-0,08192857**	**0,81536**	**354**
0,41666667	**-0,09711873**	**0,832**	**357**
0,425	**0,61129516**	**0,84864**	**353**
0,43333333	**0,35996705**	**0,86528**	**350**
0,44166667	**0,8211956**	**0,88192**	**353**
0,45	**1,0255723**	**0,89856**	**361**

Tabla 6 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS.

Representando el conjunto de medidas de aceleración con respecto del tiempo de los distintos equipos, se obtiene la gráfica de la ilustración 25:

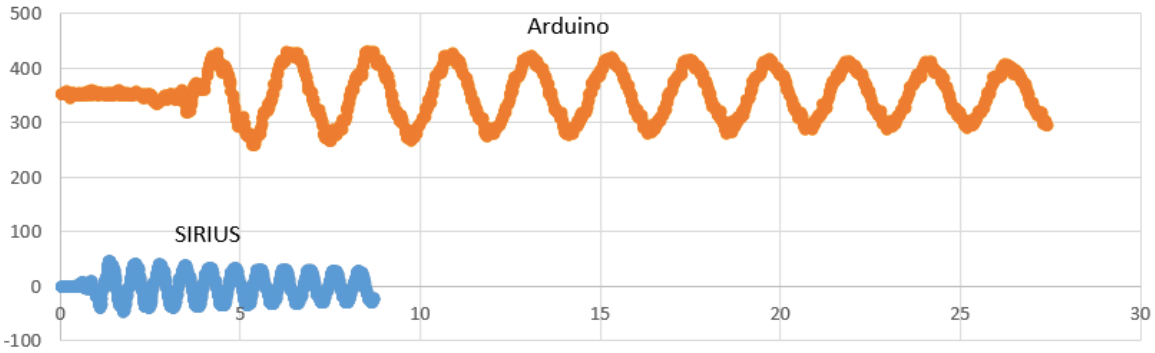

Ilustración 25 Gráfica de las muestras de medidas de aceleración obtenidas por Arduino y SIRIUS.

Como se observa en la gráfica las medidas obtenidas por SIRIUS distan bastante de las obtenidas mediante Arduino, debido a que los datos deben de ser acondicionados para que las gráficas sean comparables. La gráfica obtenida por

los acelerómetros piezoeléctricos se encuentra en m /s^2, como se observa es bastante precisa y centrada en 0 m/s^2 , lo que es lógico ya que el balanceo de la estructura en movimiento debe producir aceleraciones positivas y negativas.

3.3.2.2 Acondicionamiento del tiempo.

El problema que aparece a simple vista entre las gráficas, más allá de que para compararlas haya que acondicionar los datos obtenidos por Arduino, es la disposición del tiempo. Recordar que la captura de datos se inició en los dos sistemas en el mismo instante y se finalizó en el mismo instante. La gráfica de Arduino tiene una línea temporal de datos aproximadamente tres veces mayor que SIRIUS, lo que es ilógico puesto que como se indicó con anterioridad el inicio y el fin de adquisición de datos se realizó en el mismo instante.

Esto hace suponer que el tiempo de muestreo de Arduino calculado con el comando *"millis()"* dista en gran medida de la realidad, no obstante, aporta una perspectiva de tiempos desde la cual iniciar el acondicionamiento de la escala temporal.

Por tanto los datos de la gráfica de Arduino necesitarán ser acondicionados con respecto de los datos obtenidos con SIRIUS para poder calcular el modelo.

Inicialmente acondicionaremos el tiempo de adquisición de datos, ya que debería ser el mismo. Para ello consultamos el tiempo final determinado por los datos recogidos:

Tiempo final SIRIUS: 8,7s Tiempo final Arduino: 27,4s

Relación que existe entre los tiempos de las gráficas (ecuación 3):

$$\frac{\text{Tiempo final Arduino}}{\text{Tiempo final SIRIUS}} = 3{,}15 \qquad (3)$$

Por tanto hay que reducir el tiempo de adquisición de datos de Arduino 3,15 veces para hacer coincidir el tiempo de adquisición de datos.

A continuación se muestran en la tabla 7, 55 valores que forman parte de la población de datos utilizados, con la correspondiente actualización en el tiempo de Arduino:

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	-0,03221531	0	351
0,00833333	-0,05569101	0,0052	351
0,01666667	-0,06259563	0,0104	352
0,025	-0,04740547	0,0156	353
0,03333333	-0,03635808	0,0208	353
0,04166667	-0,01840607	0,026	353
0,05	-0,02669161	0,0312	353
0,05833333	-0,02945346	0,0364	354
0,06666667	-0,01702515	0,0416	354
0,075	-0,037739	0,0468	357
0,08333333	-0,02254884	0,052	355
0,09166667	-0,05983378	0,0572	356
0,1	-0,04464362	0,0624	358
0,10833333	-0,02807254	0,0676	353
0,11666667	-0,07088117	0,0728	347
0,125	-0,0142633	0,078	352
0,13333333	-0,02945346	0,0832	351
0,14166667	-0,04464362	0,0884	352
0,15	-0,07226209	0,0936	350
0,15833333	-0,04464362	0,0988	352
0,16666667	-0,05845286	0,104	354
0,175	-0,04050085	0,1092	354
0,18333333	-0,06811933	0,1144	352
0,19166667	-0,03635808	0,1196	350
0,2	-0,06121471	0,1248	350
0,20833333	0,00921241	0,13	352
0,21666667	-0,00735868	0,1352	349
0,225	-0,05983378	0,1404	353
0,23333333	-0,037739	0,1456	351
0,24166667	-0,04050085	0,1508	348
0,25	-0,00735868	0,156	351
0,25833333	-0,02254884	0,1612	354
0,26666667	-0,03635808	0,1664	351
0,275	-0,06259563	0,1716	350
0,28333333	-0,06259563	0,1768	352
0,29166667	-0,0142633	0,182	352
0,3	-0,05154824	0,1872	352
0,30833333	-0,037739	0,1924	350
0,31666667	-0,00459683	0,1976	351
0,325	-0,01840607	0,2028	354
0,33333333	-0,01012053	0,208	351
0,34166667	-0,03359623	0,2132	350
0,35	0,00230779	0,2184	353
0,35833333	-0,01012053	0,2236	357
0,36666667	-0,07226209	0,2288	353
0,375	-0,0667384	0,234	348
0,38333333	0,02440257	0,2392	353
0,39166667	-0,06535748	0,2444	351
0,4	-0,02945346	0,2496	352
0,40833333	-0,08192857	0,2548	354
0,41666667	-0,09711873	0,26	357
0,425	0,61129516	0,2652	353

0,43333333	**0,35996705**	**0,2704**	**350**
0,44166667	**0,8211956**	**0,2756**	**353**
0	**-0,03221531**	**0**	**351**

Tabla 7 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS modificada 1.

Se representan los datos con el tiempo de muestreo de Arduino modificado, junto con los datos recogidos por SIRIUS (ilustración 26):

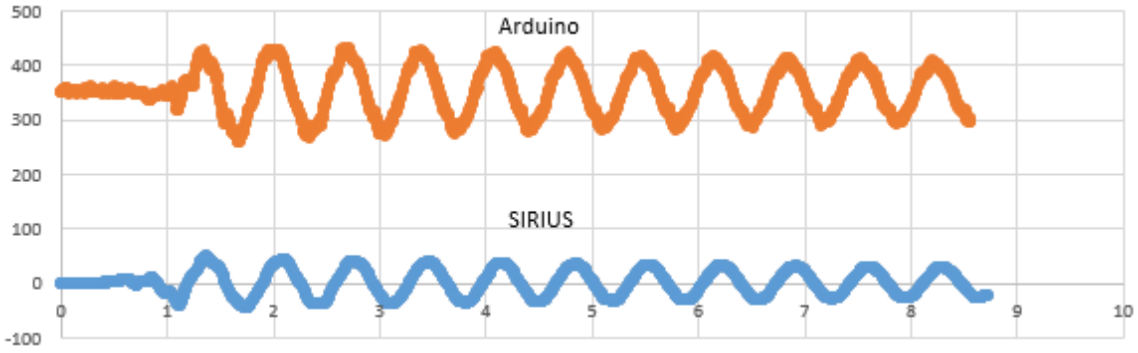

Ilustración 26 Gráfica de la muestra de medidas de aceleración obtenidas por Arduino y SIRIUS modificada 1.

Como se puede observar en la gráfica anterior (ilustración 26), la línea temporal de adquisición de datos ha sido acondicionados de manera adecuada, no obstante, se puede apreciar que hay un pequeño retraso entre las dos gráficas.

Tiempo final SIRIUS: 8,7s Tiempo final Arduino: 8,62s

Como se observa en el tiempo final, hay un pequeño retraso en el tiempo de muestras de Arduino con respecto de SIRIUS, lo que penaliza la precisión del modelo que se quiere construir.

Este retraso se extiende a todos los puntos que forman la gráfica, aunque no es acumulable, por lo que se soluciona de manera fácil añadiendo un factor de corrección al tiempo de muestreo de Arduino (ecuación 4).

F. corrección = Tiempo final SIRIUS - Tiempo final Arduino = 8,7 – 8,62 = 0,08s (4)

Sumamos el factor de corrección al tiempo de muestreo de Arduino obteniendo los siguientes datos (una pequeña muestra de la población), conjuntamente con los obtenidos de SIRIUS (tabla 8):

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	-0,03221531	0,08	351
0,00833333	-0,05569101	0,0852	351
0,01666667	-0,06259563	0,0904	352
0,025	-0,04740547	0,0956	353
0,03333333	-0,03635808	0,1008	353
0,04166667	-0,01840607	0,106	353
0,05	-0,02669161	0,1112	353
0,05833333	-0,02945346	0,1164	354
0,06666667	-0,01702515	0,1216	354
0,075	-0,037739	0,1268	357
0,08333333	-0,02254884	0,132	355
0,09166667	-0,05983378	0,1372	356
0,1	-0,04464362	0,1424	358
0,10833333	-0,02807254	0,1476	353
0,11666667	-0,07088117	0,1528	347
0,125	-0,0142633	0,158	352
0,13333333	-0,02945346	0,1632	351
0,14166667	-0,04464362	0,1684	352
0,15	-0,07226209	0,1736	350
0,15833333	-0,04464362	0,1788	352
0,16666667	-0,05845286	0,184	354
0,175	-0,04050085	0,1892	354
0,18333333	-0,06811933	0,1944	352
0,19166667	-0,03635808	0,1996	350
0,2	-0,06121471	0,2048	350
0,20833333	0,00921241	0,21	352
0,21666667	-0,00735868	0,2152	349
0,225	-0,05983378	0,2204	353
0,23333333	-0,037739	0,2256	351
0,24166667	-0,04050085	0,2308	348
0,25	-0,00735868	0,236	351
0,25833333	-0,02254884	0,2412	354
0,26666667	-0,03635808	0,2464	351
0,275	-0,06259563	0,2516	350
0,28333333	-0,06259563	0,2568	352
0,29166667	-0,0142633	0,262	352
0,3	-0,05154824	0,2672	352
0,30833333	-0,037739	0,2724	350
0,31666667	-0,00459683	0,2776	351
0,325	-0,01840607	0,2828	354
0,33333333	-0,01012053	0,288	351
0,34166667	-0,03359623	0,2932	350
0,35	0,00230779	0,2984	353
0,35833333	-0,01012053	0,3036	357
0,36666667	-0,07226209	0,3088	353
0,375	-0,0667384	0,314	348
0,38333333	0,02440257	0,3192	353
0,39166667	-0,06535748	0,3244	351

0,4	**-0,02945346**	**0,3296**	**352**
0,40833333	**-0,08192857**	**0,3348**	**354**
0,41666667	**-0,09711873**	**0,34**	**357**
0,425	**0,61129516**	**0,3452**	**353**
0,43333333	**0,35996705**	**0,3504**	**350**
0,44166667	**0,8211956**	**0,3556**	**353**
0,45	**1,0255723**	**0,3608**	**361**

Tabla 8 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS modificada 2.

Se representan el conjunto de los datos con la modificación observada en la tabla 8, en la ilustración 27:

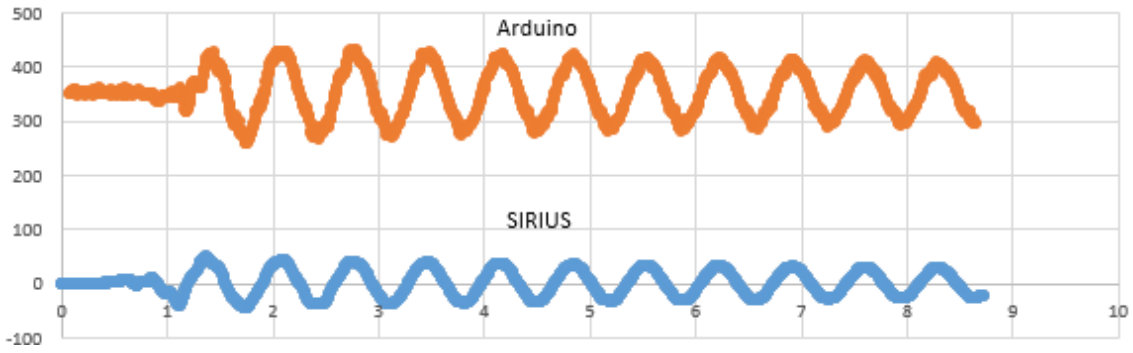

Ilustración 27 Gráfica de la muestra de medidas de aceleración obtenidas por Arduino y SIRIUS modificada 2.

Como se puede observar en la ilustración 27, los datos quedan acondicionados en la misma escala de tiempo, por lo que el problema de la línea temporal en cuestión queda solucionado.

3.3.2.3 Modelo aproximado.

El siguiente paso consiste en realizar el acondicionamiento de medidas recogidas por ambos equipos. Para ello se buscará la formalización de un modelo que transforme las medidas obtenidas con Arduino, en las medidas reales obtenidas con SIRIUS. Se utilizará un modelo lineal.

El objetivo es conseguir el siguiente modelo (ecuación 5):

$$A_R = K * A_A + B \qquad (5)$$

A_R = Aceleración calculada con Sirius (Medida SIRIUS).

A_A= Aceleración calculada con Arduino (Medida Arduino).

K= Factor de multiplicación (Pendiente de la recta).

B= Termino de ajuste a 0 (Término independiente).

La consecución del modelo, debe conseguir ajustar la gráfica de valores acondicionados recogidos por Arduino, con la gráfica de valores obtenidos con SIRIUS. De esta manera se conseguirá un modelo que convierta la medida de la aceleración en la escala de Arduino recogida del acelerómetro de la maqueta, en una aceleración que se muestre por pantalla en la escala de m/s^2 equivalente a la señal real leída. Para ello se van a contemplar dos casos:

- Pendiente, K=0,5.
- Pendiente, K=0,6.

3.3.2.3.1 Modelo aproximado. Pendiente K=0,5.

Para el caso de que la pendiente sea K=0,5, las medidas de Arduino se modificarían de la siguiente manera (tabla 9), aplicando la fórmula (ecuación 6):

.

$$A_R = K * A_A = 0{,}5 * A_A \quad (6)$$

Datos modificados:

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	**-0,032215308**	**0,08**	**175,5**
0,008333333	**-0,055691011**	**0,0852**	**175,5**
0,016666667	**-0,062595628**	**0,0904**	**176**
0,025	**-0,04740547**	**0,0956**	**176,5**
0,033333333	**-0,036358077**	**0,1008**	**176,5**
0,041666667	**-0,018406069**	**0,106**	**176,5**
0,05	**-0,026691612**	**0,1112**	**176,5**
0,058333333	**-0,02945346**	**0,1164**	**177**

0,066666667	-0,017025145	0,1216	177
0,075	-0,037739001	0,1268	178,5
0,083333333	-0,022548841	0,132	177,5
0,091666667	-0,059833784	0,1372	178
0,1	-0,044643622	0,1424	179
0,108333333	-0,028072536	0,1476	176,5
0,116666667	-0,070881173	0,1528	173,5
0,125	-0,014263298	0,158	176
0,133333333	-0,02945346	0,1632	175,5
0,141666667	-0,044643622	0,1684	176
0,15	-0,072262093	0,1736	175
0,158333333	-0,044643622	0,1788	176
0,166666667	-0,05845286	0,184	177
0,175	-0,040500849	0,1892	177
0,183333333	-0,068119325	0,1944	176
0,191666667	-0,036358077	0,1996	175
0,2	-0,061214708	0,2048	175
0,208333333	0,009212406	0,21	176
0,216666667	-0,007358679	0,2152	174,5
0,225	-0,059833784	0,2204	176,5
0,233333333	-0,037739001	0,2256	175,5
0,241666667	-0,040500849	0,2308	174
0,25	-0,007358679	0,236	175,5
0,258333333	-0,022548841	0,2412	177
0,266666667	-0,036358077	0,2464	175,5
0,275	-0,062595628	0,2516	175
0,283333333	-0,062595628	0,2568	176
0,291666667	-0,014263298	0,262	176
0,3	-0,051548239	0,2672	176
0,308333333	-0,037739001	0,2724	175
0,316666667	-0,004596832	0,2776	175,5
0,325	-0,018406069	0,2828	177
0,333333333	-0,010120527	0,288	175,5
0,341666667	-0,033596233	0,2932	175
0,35	0,002307787	0,2984	176,5
0,358333333	-0,010120527	0,3036	178,5
0,366666667	-0,072262093	0,3088	176,5
0,375	-0,066738404	0,314	174
0,383333333	0,024402568	0,3192	176,5
0,391666667	-0,065357476	0,3244	175,5
0,4	-0,02945346	0,3296	176
0,408333333	-0,081928566	0,3348	177
0,416666667	-0,097118728	0,34	178,5
0,425	0,61129516	0,3452	176,5
0,433333333	0,35996705	0,3504	175
0,441666667	0,8211956	0,3556	176,5
0,45	1,0255723	0,3608	180,5

Tabla 9 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS modificada 3.

Como se puede apreciar en la gráfica de la ilustración 28 la modificación expuesta en la tabla 9 aplicada al conjunto de datos obtendría el siguiente resultado:

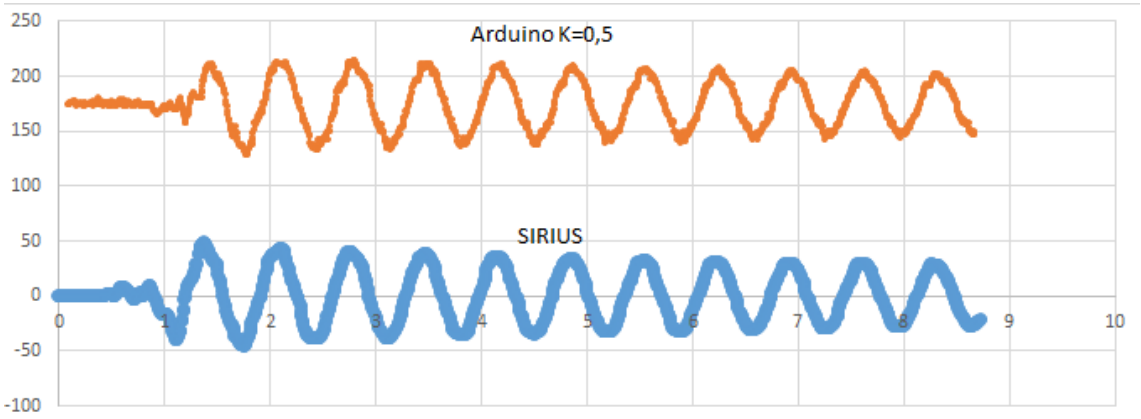

Ilustración 28 Gráfica de la muestra de medidas de aceleración obtenidas por Arduino y SIRIUS modificada 3.

El valor medio de las aceleraciones de Arduino se ha reducido a la mitad, quedando aproximadamente su valor medio en 175 unidades. El resto del ajuste del modelo para conseguir la coincidencia de gráficas se llevará a cabo con el término independiente B.

Como se puede apreciar en la gráfica anterior, para que los conjuntos de datos coincidan, se ha de hacer coincidir el valor medio de ambos. En este caso, como se busca acondicionar los datos medidos con Arduino, el valor medio de dichos datos se acondicionará a un valor de 0 (valor medio de SIRIUS).

Aplicaremos ahora el modelo conjunto, siendo su fórmula la vista con anterioridad (ecuación 5):

$$A_R = K * A_A + B \qquad (5)$$

Para una pendiente de K=0,5, el valor de B para hacer coincidir los valores medios será de -175 unidades, valor que se obtiene de la gráfica anterior de manera visual o realizando la media del conjunto de datos que forman la población. En este caso:

Valor Medio de medidas Arduino = 175 unidades

Como queremos que el conjunto de medidas tenga un valor medio de 0, el término independiente, B, tendrá signo negativo.

El modelo aproximado calculado sería el siguiente:

$$A_R = 0{,}5 * A_A - 175$$

Por tanto, realizando esta modificación a los datos (tabla 10), la muestra con la que se viene ejemplificando los resultados de las modificaciones aplicadas, tendrá los siguientes valores:

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	-0,032215308	0,08	0,5
0,008333333	-0,055691011	0,0852	0,5
0,016666667	-0,062595628	0,0904	1
0,025	-0,04740547	0,0956	1,5
0,033333333	-0,036358077	0,1008	1,5
0,041666667	-0,018406069	0,106	1,5
0,05	-0,026691612	0,1112	1,5
0,058333333	-0,02945346	0,1164	2
0,066666667	-0,017025145	0,1216	2
0,075	-0,037739001	0,1268	3,5
0,083333333	-0,022548841	0,132	2,5
0,091666667	-0,059833784	0,1372	3
0,1	-0,044643622	0,1424	4
0,108333333	-0,028072536	0,1476	1,5
0,116666667	-0,070881173	0,1528	-1,5
0,125	-0,014263298	0,158	1
0,133333333	-0,02945346	0,1632	0,5
0,141666667	-0,044643622	0,1684	1
0,15	-0,072262093	0,1736	0
0,158333333	-0,044643622	0,1788	1
0,166666667	-0,05845286	0,184	2
0,175	-0,040500849	0,1892	2
0,183333333	-0,068119325	0,1944	1
0,191666667	-0,036358077	0,1996	0
0,2	-0,061214708	0,2048	0
0,208333333	0,009212406	0,21	1
0,216666667	-0,007358679	0,2152	-0,5
0,225	-0,059833784	0,2204	1,5
0,233333333	-0,037739001	0,2256	0,5
0,241666667	-0,040500849	0,2308	-1
0,25	-0,007358679	0,236	0,5
0,258333333	-0,022548841	0,2412	2
0,266666667	-0,036358077	0,2464	0,5
0,275	-0,062595628	0,2516	0
0,283333333	-0,062595628	0,2568	1
0,291666667	-0,014263298	0,262	1
0,3	-0,051548239	0,2672	1
0,308333333	-0,037739001	0,2724	0

0,316666667	-0,004596832	0,2776	0,5
0,325	-0,018406069	0,2828	2
0,333333333	-0,010120527	0,288	0,5
0,341666667	-0,033596233	0,2932	0
0,35	0,002307787	0,2984	1,5
0,358333333	-0,010120527	0,3036	3,5
0,366666667	-0,072262093	0,3088	1,5
0,375	-0,066738404	0,314	-1
0,383333333	0,024402568	0,3192	1,5
0,391666667	-0,065357476	0,3244	0,5
0,4	-0,02945346	0,3296	1
0,408333333	-0,081928566	0,3348	2
0,416666667	-0,097118728	0,34	3,5
0,425	0,61129516	0,3452	1,5
0,433333333	0,35996705	0,3504	0
0,441666667	0,8211956	0,3556	1,5
0,45	1,0255723	0,3608	5,5

Tabla 10 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS modificada 4.

Representando gráficamente el conjunto de datos con la modificación expuesta en la tabla 10 se obtiene la ilustración 29:

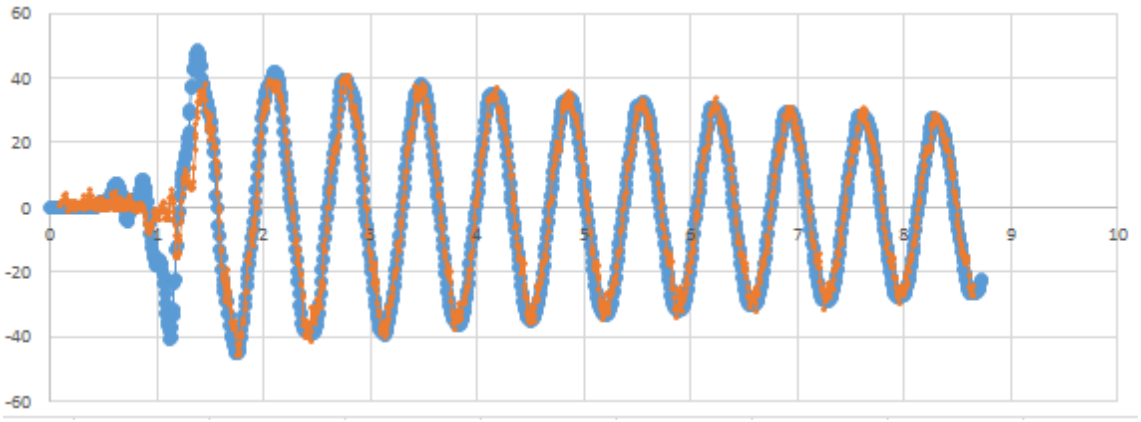

Ilustración 29 Gráfica de la muestra de medidas de aceleración obtenidas por Arduino y SIRIUS modificada 4.

Como se puede observar en la ilustración 29, el conjunto de datos coincide de manera correcta, concluyendo que el modelo aplicado es adecuado para realizar la conversión de escala que se estaba buscando.

El modelo aproximado final es:

$$A_R = 0{,}5 * A_A - 175$$

Incluyendo esta conversión en el programa de Arduino, los valores que se observen por pantalla estarán en una escala de m/s^2.

3.3.2.3.2 Modelo aproximado. Pendiente K=0,6

Para el caso de que la pendiente fuese K=0,6, las medidas de Arduino se modificarían de la siguiente manera (una pequeña muestra de la población), aplicando la siguiente fórmula (ecuación):

.

$$A_R = K * A_A \quad (6)$$

Muestra de datos modificados (tabla 11):

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	-0,032215308	0,08	210,6
0,008333333	-0,055691011	0,0852	210,6
0,016666667	-0,062595628	0,0904	211,2
0,025	-0,04740547	0,0956	211,8
0,033333333	-0,036358077	0,1008	211,8
0,041666667	-0,018406069	0,106	211,8
0,05	-0,026691612	0,1112	211,8
0,058333333	-0,02945346	0,1164	212,4
0,066666667	-0,017025145	0,1216	212,4
0,075	-0,037739001	0,1268	214,2
0,083333333	-0,022548841	0,132	213
0,091666667	-0,059833784	0,1372	213,6
0,1	-0,044643622	0,1424	214,8
0,108333333	-0,028072536	0,1476	211,8
0,116666667	-0,070881173	0,1528	208,2
0,125	-0,014263298	0,158	211,2
0,133333333	-0,02945346	0,1632	210,6
0,141666667	-0,044643622	0,1684	211,2
0,15	-0,072262093	0,1736	210
0,158333333	-0,044643622	0,1788	211,2
0,166666667	-0,05845286	0,184	212,4
0,175	-0,040500849	0,1892	212,4
0,183333333	-0,068119325	0,1944	211,2
0,191666667	-0,036358077	0,1996	210
0,2	-0,061214708	0,2048	210
0,208333333	0,009212406	0,21	211,2

0,216666667	-0,007358679	0,2152	209,4
0,225	-0,059833784	0,2204	211,8
0,233333333	-0,037739001	0,2256	210,6
0,241666667	-0,040500849	0,2308	208,8
0,25	-0,007358679	0,236	210,6
0,258333333	-0,022548841	0,2412	212,4
0,266666667	-0,036358077	0,2464	210,6
0,275	-0,062595628	0,2516	210
0,283333333	-0,062595628	0,2568	211,2
0,291666667	-0,014263298	0,262	211,2
0,3	-0,051548239	0,2672	211,2
0,308333333	-0,037739001	0,2724	210
0,316666667	-0,004596832	0,2776	210,6
0,325	-0,018406069	0,2828	212,4
0,333333333	-0,010120527	0,288	210,6
0,341666667	-0,033596233	0,2932	210
0,35	0,002307787	0,2984	211,8
0,358333333	-0,010120527	0,3036	214,2
0,366666667	-0,072262093	0,3088	211,8
0,375	-0,066738404	0,314	208,8
0,383333333	0,024402568	0,3192	211,8
0,391666667	-0,065357476	0,3244	210,6
0,4	-0,02945346	0,3296	211,2
0,408333333	-0,081928566	0,3348	212,4
0,416666667	-0,097118728	0,34	214,2
0,425	0,61129516	0,3452	211,8
0,433333333	0,35996705	0,3504	210
0,441666667	0,8211956	0,3556	211,8
0,45	1,0255723	0,3608	216,6

Tabla 11 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS modificada 5.

Como se puede apreciar en la ilustración 30 las medidas de la aceleración de Arduino queda reducida cuantiosamente. Representando el conjunto de datos recogidos con la modificación de la tabla 11 se obtiene la ilustración 30:

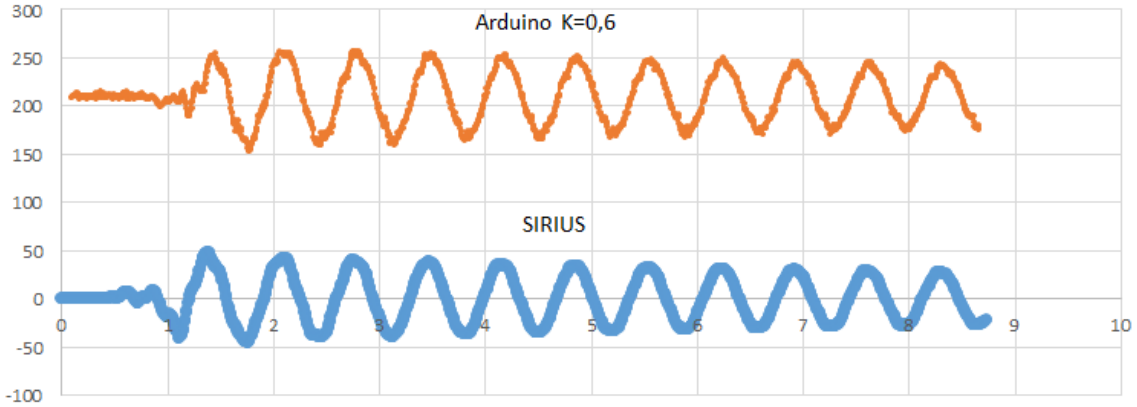

Ilustración 30 Gráfica de la muestra de medidas de aceleración obtenidas por Arduino y SIRIUS modificada 5.

El valor medio de las aceleraciones de Arduino se ha reducido de manera cuantiosa, quedando aproximadamente su valor medio en 211 unidades. El resto del ajuste del modelo para conseguir la coincidencia de gráficas se llevará a cabo con el término independiente B.

Como se puede apreciar en la gráfica anterior, para que los conjuntos de datos coincidan, se ha de hacer coincidir el valor medio de ambos. En este caso, como se busca acondicionar los datos medidos con Arduino, el valor medio de dichos datos se acondicionará a un valor de 0 (valor medio de SIRIUS).

Se aplicará el modelo al conjunto de datos, siendo su fórmula la vista con anterioridad (ecuación 5):

$$A_R = K * A_A + B \quad (5)$$

Para una pendiente de K=0,6, el valor de B para hacer coincidir los valores medios será de -211 unidades, valor que se obtiene de la gráfica anterior de manera visual o realizando la media del conjunto de datos que forman la población. En este caso:

Valor Medio de medidas Arduino = 211 unidades

Como queremos que el conjunto de medidas tenga un valor medio de 0, el término independiente, B, tendrá signo negativo.

El modelo aproximado calculado sería el siguiente:

$$A_R = 0{,}6 * A_A - 211$$

Por tanto, realizando esta modificación a los datos, la muestra con la que se viene ejemplificando los resultados de las modificaciones aplicadas, tendrá los siguientes valores tabla 12:

Tiempo Sirius	Medida Sirius	Tiempo Arduino	Medida Arduino
0	-0,032215308	0,08	210,6
0,008333333	-0,055691011	0,0852	210,6
0,016666667	-0,062595628	0,0904	211,2
0,025	-0,04740547	0,0956	211,8
0,033333333	-0,036358077	0,1008	211,8
0,041666667	-0,018406069	0,106	211,8
0,05	-0,026691612	0,1112	211,8
0,058333333	-0,02945346	0,1164	212,4
0,066666667	-0,017025145	0,1216	212,4
0,075	-0,037739001	0,1268	214,2
0,083333333	-0,022548841	0,132	213
0,091666667	-0,059833784	0,1372	213,6
0,1	-0,044643622	0,1424	214,8
0,108333333	-0,028072536	0,1476	211,8
0,116666667	-0,070881173	0,1528	208,2
0,125	-0,014263298	0,158	211,2
0,133333333	-0,02945346	0,1632	210,6
0,141666667	-0,044643622	0,1684	211,2
0,15	-0,072262093	0,1736	210
0,158333333	-0,044643622	0,1788	211,2
0,166666667	-0,05845286	0,184	212,4
0,175	-0,040500849	0,1892	212,4
0,183333333	-0,068119325	0,1944	211,2
0,191666667	-0,036358077	0,1996	210
0,2	-0,061214708	0,2048	210
0,208333333	0,009212406	0,21	211,2
0,216666667	-0,007358679	0,2152	209,4
0,225	-0,059833784	0,2204	211,8
0,233333333	-0,037739001	0,2256	210,6
0,241666667	-0,040500849	0,2308	208,8
0,25	-0,007358679	0,236	210,6
0,258333333	-0,022548841	0,2412	212,4
0,266666667	-0,036358077	0,2464	210,6
0,275	-0,062595628	0,2516	210
0,283333333	-0,062595628	0,2568	211,2
0,291666667	-0,014263298	0,262	211,2
0,3	-0,051548239	0,2672	211,2
0,308333333	-0,037739001	0,2724	210
0,316666667	-0,004596832	0,2776	210,6
0,325	-0,018406069	0,2828	212,4
0,333333333	-0,010120527	0,288	210,6
0,341666667	-0,033596233	0,2932	210
0,35	0,002307787	0,2984	211,8
0,358333333	-0,010120527	0,3036	214,2
0,366666667	-0,072262093	0,3088	211,8
0,375	-0,066738404	0,314	208,8
0,383333333	0,024402568	0,3192	211,8
0,391666667	-0,065357476	0,3244	210,6
0,4	-0,02945346	0,3296	211,2
0,408333333	-0,081928566	0,3348	212,4
0,416666667	-0,097118728	0,34	214,2
0,425	0,61129516	0,3452	211,8

0,433333333	**0,35996705**	**0,3504**	**210**
0,441666667	**0,8211956**	**0,3556**	**211,8**
0,45	**1,0255723**	**0,3608**	**216,6**

Tabla 12 Tabla de muestras de medidas de aceleración recogidas por Arduino y SIRIUS modificada 6.

Representando gráficamente el conjunto de datos, con la modificación expuesta en la muestra de la tabla 12, se obtiene la ilustración 31:

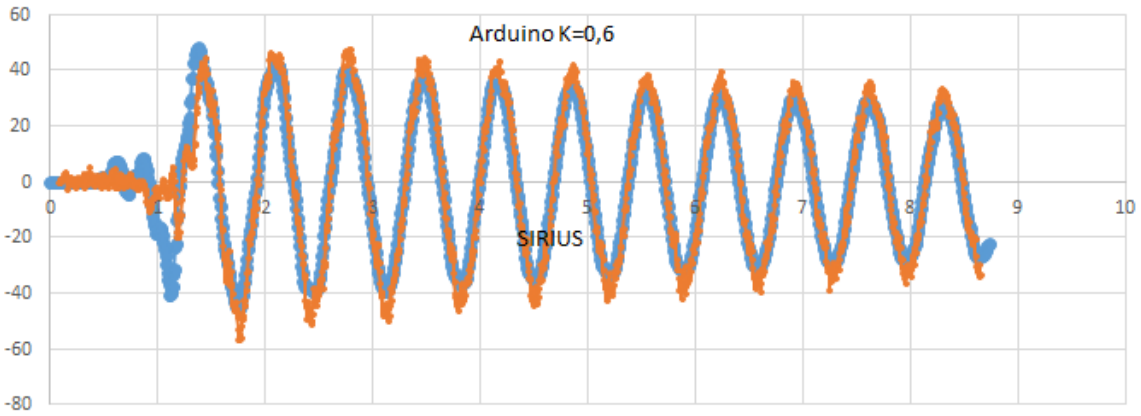

Ilustración 31 Gráfica de la muestra de medidas de aceleración obtenidas por Arduino y SIRIUS modificada 6.

Como se puede observar, el conjunto de datos coincide de manera menos precisa, concluyendo que el modelo aplicado no es adecuado para realizar la conversión de escala que se estaba buscando. El modelo aproximado final es:

$$A_R = 0{,}6 * A_A - 210$$

Incluyendo esta conversión en el programa de Arduino, los valores que se saquen por pantalla estarán en una escala de m/s^2 obteniéndose un valor de aceleración más impreciso que en el caso que se ha manejado en el apartado anterior. Por este motivo no se utilizará este modelo en el programa que realiza lecturas en la maqueta.

El modelo final teórico válido final es:

$$A_R = 0{,}5 * A_A - 175$$

3.3.3 Inclusión de modelo teórico en la maqueta real.

3.3.3.1 Introducción

Una vez que se ha conseguido calcular el modelo teórico, es necesario probar su validez en la realidad. Dicho modelo será utilizado para transformar el valor de aceleración medido en el eje Z y en el eje Y. Como se indicó con anterioridad se ha supuesto que los ejes inicialmente son idénticos, por lo que en las mismas condiciones de aceleración se obtendrán las mismas medidas. Destaquemos que la diferencia de calibración real se basa en que el eje Y será ajustado a 0 m/s^2 y el eje Z será ajustado a 9,8 m/s^2.

3.3.3.2 Calibración del eje Y del acelerómetro.

Para realizar la calibración del eje Y se necesita incorporar el modelo teórico calculado en el programa que se utiliza en Arduino, con el objetivo de realizar el cambio de escala. El problema que surge consiste en como determinar que la aceleración de salida en m/s^2 es correcta sin la necesidad de utilizar otro equipo de medición.

La solución que se propone consiste en situar el acelerómetro, en este caso, con el eje Y paralelo al suelo (en la posición en la que se va a situar en la maqueta). De esta manera, y en ausencia de balanceo de la maqueta, el conjunto del equipo destinado a realizar la medida de la aceleración deberá mostrar por pantalla un valor de aproximadamente 0 m/s^2, es decir, deberá mostrar que el sistema se encuentra en reposo.

Una vez que la maqueta se encuentre en estado de reposo, se iniciará el programa de recogida de medidas, visualizando cuan coherente es el modelo teórico y si es necesario tomar acción para realizar un ajuste mayor de la medida de salida de aceleración.

Este estudio experimental se llevará a cabo cogiendo 55 valores mostrados por pantalla una vez que el sistema haya funcionado durante 10 segundos. La espera de 10 segundos es debida a que el sistema al arrancar muestra valores erróneos hasta que se estabiliza.

El primer modelo teórico incorporado en el programa de control de Arduino denominado Acele_calibra, será:

$$A_R = 0{,}5 * A_A - 175$$

Los 55 valores recogidos se muestran en la siguiente tabla 13:

Medida del Eje Y
1
3,5
3
3
2,5
4
2,5
2
2,5
3
4,5
2,5
3
4
3
2,5
3
3

2,5
3
3
3,5
4
3
3,5
4
3
3,5
3,5
3
3
3,5
2,5
4
3,5
4
3

4
3,5
3
2,5
4
2,5
3
3
3
4
3,5
3
3
3,5
3,5
4
3
1,5
Media = 3,14

Tabla 13 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala.

Como se puede apreciar en la tabla 13, el modelo teórico consigue aproximar de manera correcta el valor esperado por pantalla con un pequeño error. Para solucionar este problema se realizarán las modificaciones pertinentes del término independiente, B.

Puesto que la media es 3, habrá que buscar un valor de B que lleve el valor ofrecido por pantalla a un valor aproximado de 0.

$$A_R = 0{,}5 * A_A - 175,\ B_0 = -175$$

Por tanto B modificará su valor:

$$3 + X = 0;\quad X = -3$$

$$B = B_0 - 3 = -175 - 3 = -178$$

El nuevo modelo será el siguiente:

$$A_R = 0{,}5 * A_A - 178$$

Se inicia de nuevo la recogida de datos, incluyendo este modelo en el programa Acele_calibra. Los 55 valores recogidos se muestran en la siguiente tabla 14:

Medida del Eje Y		
0,5	1	0,5
1,5	2	1
0	2	1
1	2	1,5
1	1	1
0,5	0,5	1,5
2	1	1,5
1,5	1	2
1,5	0	1
2	1,5	0,5
1	0,5	1
2	1	1,5
0,5	1	0,5
2,5	1	1,5
1	1,5	2
2	2	1,5
0,5	1	2
1	0	1,5
	1,5	Media = 1,209091

Tabla 14 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala modificación 1.

Como se puede observar el valor medio de las aceleraciones expuestas se ha aproximado bastante al valor deseado, sin embargo aún se puede ajustar en mayor medida.

Esta vez partimos del siguiente modelo:

$$A_R = 0{,}5 * A_A - 178, B_0 = -178$$

Por tanto B modificará su valor:

$$1 + X = 0 \quad X = -1$$

$$B = B_0 + 3 = -178 - 1 = -179$$

El nuevo modelo será el siguiente:

$$A_R = 0{,}5 * A_A - 179$$

Se inicia de nuevo la recogida de datos incluyendo este modelo en el programa Acele_calibra. Los 55 valores recogidos se muestran en la siguiente tabla 15:

Medida del Eje Y		
0	0	0
0	1	1
-0,5	1	0
0	1,5	-1,5
0,5	0,5	1,5
0,5	0,5	0,5
0	0,5	0,5
1	1,5	-0,5
0,5	0	1
1	0,5	0,5
-0,5	-0,5	-0,5
1	0	0
0,5	2	-0,5
0	1	0
0,5	0	0,5
-0,5	0	-0,5
0	1	1
0	0	Media = 0,3

Tabla 15 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala modificación 2.

Como se puede observar el valor medio de las aceleraciones expuestas en la tabla 15 se ha aproximado bastante al valor deseado, siendo este resultado válido. Por otro lado, se va a intentar ajustar un poco más el modelo:

Esta vez partimos del siguiente modelo:

$$A_R = 0{,}5 * A_A - 179, B_0 = -179$$

Por tanto B modificará su valor:

$$1 + X = 0 \quad X = -1$$

$$B = B_0 - 1 = -179 - 1 = -180$$

El nuevo modelo será el siguiente:

$$A_R = 0{,}5 * A_A - 180$$

Se inicia de nuevo la recogida de datos incluyendo este modelo en el programa Acele_calibra. Los 55 valores recogidos se muestran en la siguiente tabla 16:

Medida del Eje Y
-1
-0,5
1
0,5
-0,5
0
0
-0,5
-1
0
-0,5
0
-0,5
-1
-1
0
-2
-0,5

-0,5
0,5
-0,5
-0,5
0,5
-0,5
-1
0
-1
0,5
-0,5
-0,5
0,5
-0,5
-1
0
0,5
-0,5
-1

0
-0,5
-1
0,5
-1
0
1
0
-1
0
0
-1
-0,5
-1
0
-0,5
-1
-0,5
Media = - 0,35

Tabla 16 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala modificación 3.

Como se puede apreciar el valor medio de las medidas mostradas por pantalla posee un valor de -0,35. Este valor se considerará aceptable, no obstante, se comete aproximadamente el mismo error que con el modelo anterior. El modelo final incluido en el programa que controla los acelerómetros denominado Acelerometros (ver Anexo), será el calculado en el paso anterior (por conveniencia):

$$A_R = 0{,}5 * A_A - 179$$

Las medidas obtenidas por pantalla por tanto, tienen su escala de m/s^2. Cabe destacar que las medidas obtenidas presentan una pequeña oscilación y varían en cada iteración. En el apartado de medidas futuras, se incorporará la idea de la utilización de otros acelerómetros más precisos (acelerómetros piezoeléctricos).

3.3.3.3 Calibración del eje Z del acelerómetro.

Para realizar la calibración del eje Z se necesita incorporar el modelo teórico calculado en el programa que se utiliza en Arduino, con el objetivo de realizar el cambio de escala.

El problema que surge consiste en como determinar que la aceleración de salida en m/s^2 es correcta sin la necesidad de utilizar otro equipo de medición.

La solución que se propone consiste en situar el acelerómetro, en este caso, con el eje Z perpendicular al suelo y sentido hacia el suelo.

De esta manera, y en ausencia de balanceo de la maqueta, el conjunto del equipo destinado a realizar la medida de la aceleración deberá mostrar por pantalla un valor de aproximadamente 9,81 m/s^2, es decir, deberá mostrar el valor de la gravedad y con sentido positivo.

Una vez que la maqueta se encuentre en estado de reposo, se iniciará el programa de recogida de medidas, visualizando cuan coherente es el modelo teórico y si es necesario tomar acción para realizar un ajuste mayor de la medida de salida de aceleración.

Este estudio experimental se llevará a cabo cogiendo 55 valores mostrados por pantalla una vez que el sistema haya funcionado durante 10 segundos. La espera de 10 segundos es debida a que el sistema al arrancar muestra valores erróneos hasta que se estabiliza.

El primer modelo teórico incorporado en el programa de control de Arduino denominado Acele_calibra, será:

$$A_R = 0{,}5 * A_A - 175$$

Los 55 valores recogidos se muestran en la siguiente tabla 17:

Medida del Eje Y
53,5
53
53
53
53,5
53
53,5
53,5
54,5
53,5
53
53,5
52,5

53,5
53
53,5
53
54
55,5
54
52,5
53
53
53,5
53,5
53,5

52,5
54
54
53
54
53,5
54,5
53,5
53
54
52,5
53
56

53,5	**53,5**	**53,5**
53	**54**	**53**
54	**53**	**54,5**
55	**53**	**52**
53	**53**	**54**
53,5	**54,5**	**Media = 53,5**

Tabla 17 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala.

Como se puede apreciar en la tabla 17, el modelo teórico no consigue aproximar de manera correcta el valor esperado por pantalla.

Para solucionar este problema se realizarán las modificaciones pertinentes del término independiente, B.

Puesto que la media es 53, habrá que buscar un valor de B que lleve el valor ofrecido por pantalla a un valor aproximado de 9 0 10.

$$A_R = 0{,}5 * A_A - 175, B_0 = -175$$

Por tanto B modificará su valor:

$$53 + X = 10 \quad X = -44$$

$$B = B_0 - 90 = -175 - 44 = -219$$

El nuevo modelo será el siguiente:

$$A_R = 0{,}5 * A_A - 219$$

Se inicia de nuevo la recogida de datos incluyendo este modelo en el programa Acele_calibra. Los 55 valores recogidos se muestran en la siguiente tabla 18:

Medida del Eje Y		
9,5	7,5	8
9,5	8	8
9	8	10,5
8,5	7	6,5
8	11	9
9,5	9	8,5
9	9	9
9	9	8,5
8	8,5	9
9	8,5	8,5
9,5	10	9,5
10	9,5	7,5
9	9,5	9,5
9,5	10	8,5
10	9	9
9	9,5	8,5
8	9	9
8	9,5	9,5
	8,5	Media = 8,8

Tabla 18 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala modificación 1.

Como se puede observar en la tabla 18 el valor medio de las aceleraciones expuestas se ha aproximado bastante al valor deseado, sin embargo aún se puede ajustar en mayor medida.

Esta vez partimos del siguiente modelo:

$$A_R = 0{,}5 * A_A - 219,\ B_0 = -219$$

Por tanto B modificará su valor:

$$9 + X = 10 \quad X = +1$$

$$B = B_0 + 3 = -219 + 1 = -218$$

El nuevo modelo será el siguiente:

$$A_R = 0{,}5 * A_A - 218$$

Se inicia de nuevo la recogida de datos incluyendo este modelo en el programa Acele_calibra. Los 55 valores recogidos se muestran en la siguiente tabla 19:

Medida del Eje Y	9	11
9,5	10,5	10,5
10	11,5	7
10	9,5	10
10,5	8,5	14
10	11	11
10	11,5	11
10	11	10
10	11	9,5
10	10	9,5
10,5	11	9,5
9	9,5	10,5
12.50	10,5	11
10	10	10
11	9,5	10,5
9	10	10,5
11	11	10,5
13	10,5	9,5
9,5	9	Media = 10,24

Tabla 19 Muestra de 55 valores de aceleración de salida, una vez aplicado el modelo de cambio de escala modificación 2.

Como se puede apreciar en la tabla 19 el valor medio de las medidas mostradas por pantalla posee un valor de 10. Este valor se considerará aceptable, por lo que el modelo final incluido en el programa que controla los acelerómetros, denominado Acelerómetros (ver Anexo), es el siguiente:

$$A_R = 0{,}5 * A_A - 218$$

Por último indicar que el software que se ha incorporado realiza la media de 50 medidas para obtener el dato final y mostrarlo por pantalla.

Las medidas obtenidas por pantalla por tanto, tienen su escala de m/s^2. Cabe destacar que las medidas obtenidas presentan una pequeña oscilación y varían en cada iteración. En el apartado de medidas futuras, se incorporará la idea de la utilización de otros acelerómetros más precisos (acelerómetros piezoeléctricos).

3.3.3.3 Calibración del acelerómetro 2.

Para el acelerómetro 2, debido a que son diferentes, llevará su propia calibración. Por consiguiente se deberá realizar dicha calibración siguiendo los pasos, con el que se han calculado los modelos teóricos para los ejes Y y Z del acelerómetro 1.

Posteriormente comprobar en la maqueta real los resultados obtenidos.

El modelo incorporado en el *software* para el acelerómetro 2 son los indicados a continuación.

Modelo del eje Y2:

$$A_R = 0{,}5 * A_A - 469$$

Modelo del eje Z2:

$$A_R = 0{,}5 * A_A - 448$$

4. Conexión de servomotores y acelerómetros con Arduino.

4.1 Introducción.

Una vez explicada la calibración de los distintos equipos que se utilizan en la maqueta, queda determinar cómo se conectan a Arduino. Como se mencionó con anterioridad, para evitar una alta carga computacional, se va a utilizar un Arduino AT Mega 2560 para realizar el control de los servomotores y otro Arduino AT Mega 2560 para realizar una lectura constante de aceleraciones en el eje Z y en el eje Y.

4.2 Conexión de Arduino con servomotores.

La conexión de los servomotores se realizará con el denominado Arduino 1 con el objetivo de facilitar la explicación.

Como se mencionó durante la selección del servomotor adecuado, cada uno de ellos va a realizar tres conexiones con la placa de Arduino siendo:

- Cable rojo del servo con el pin de 5v de la placa de Arduino (alimentación).
- Cable marrón del servo con el pin GND de la placa Arduino (Tierra).
- Cable naranja del servo con el pin PWM (señal de control).

Para facilitar la conexión con el Arduino 1 de los servomotores, se ha llevado a cabo una unión de los tres terminales de masa de los servos bajo un mismo conductor que será el punto de unión en el pin de Arduino 1 denominado GND.

El mismo proceso se ha realizado con los tres terminales de alimentación de los servos y el pin de 5v de la placa de Arduino 1.

En cuanto a los terminales de señal PWM (señal de control), se ha determinado para cada servomotor un pin PWM específico, los cuales no deben de ser intercambiados, ya que el software incorporado está diseñado para las conexiones que se van a describir a continuación:

Pin PWM Nº 8 → Señal de control Servomotor de excitación.

Pin PWM Nº 9 → Señal de control Servomotor de amortiguamiento del péndulo.

Pin PWM Nº 10 → Señal de control Servomotor de activación del péndulo.

De una manera gráfica, la conexión se realizaría de la siguiente manera (ilustración 32):

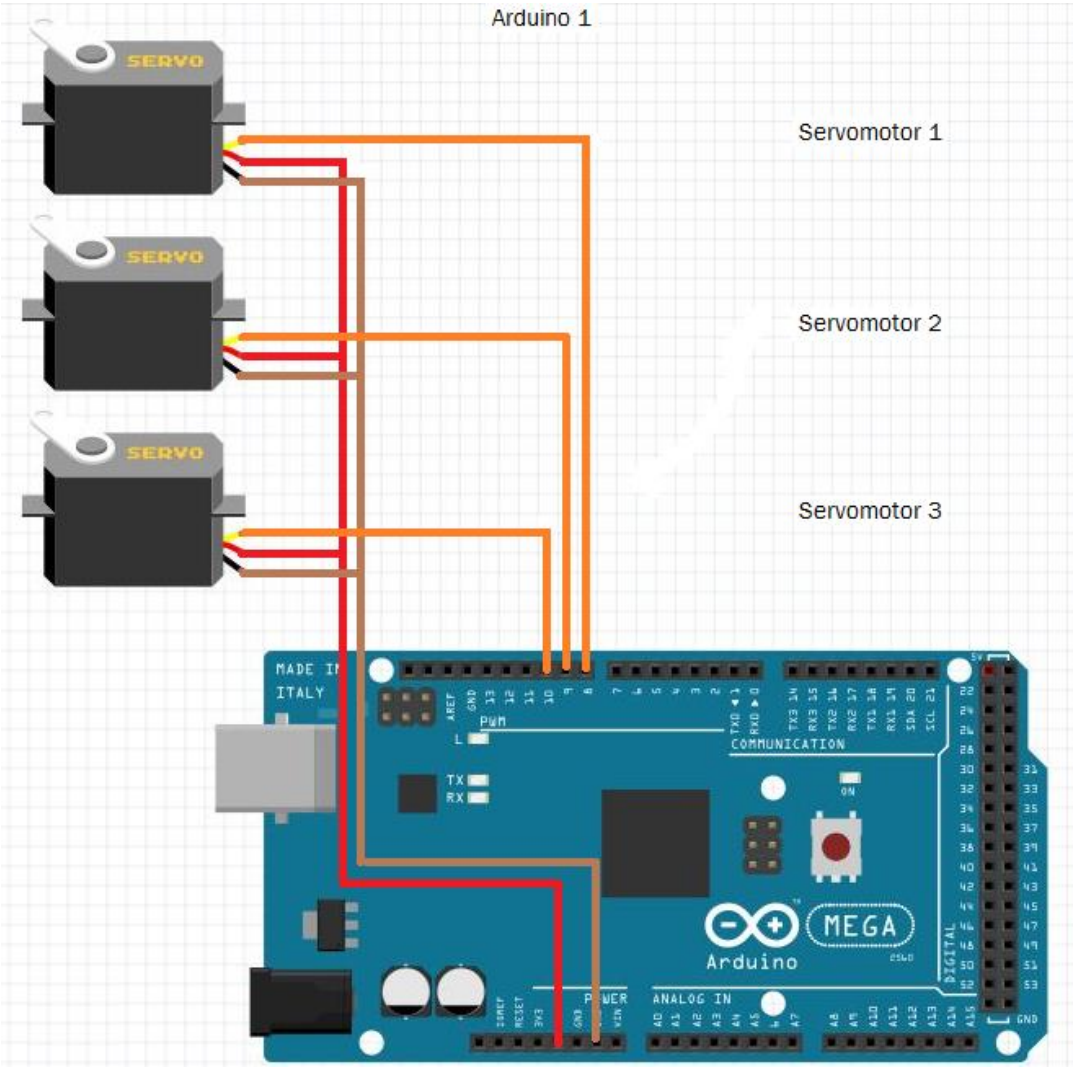

Ilustración 32 Conexión de los tres servomotores con Arduino 1.

El esquema eléctrico del circuito se muestra en la ilustración 33:

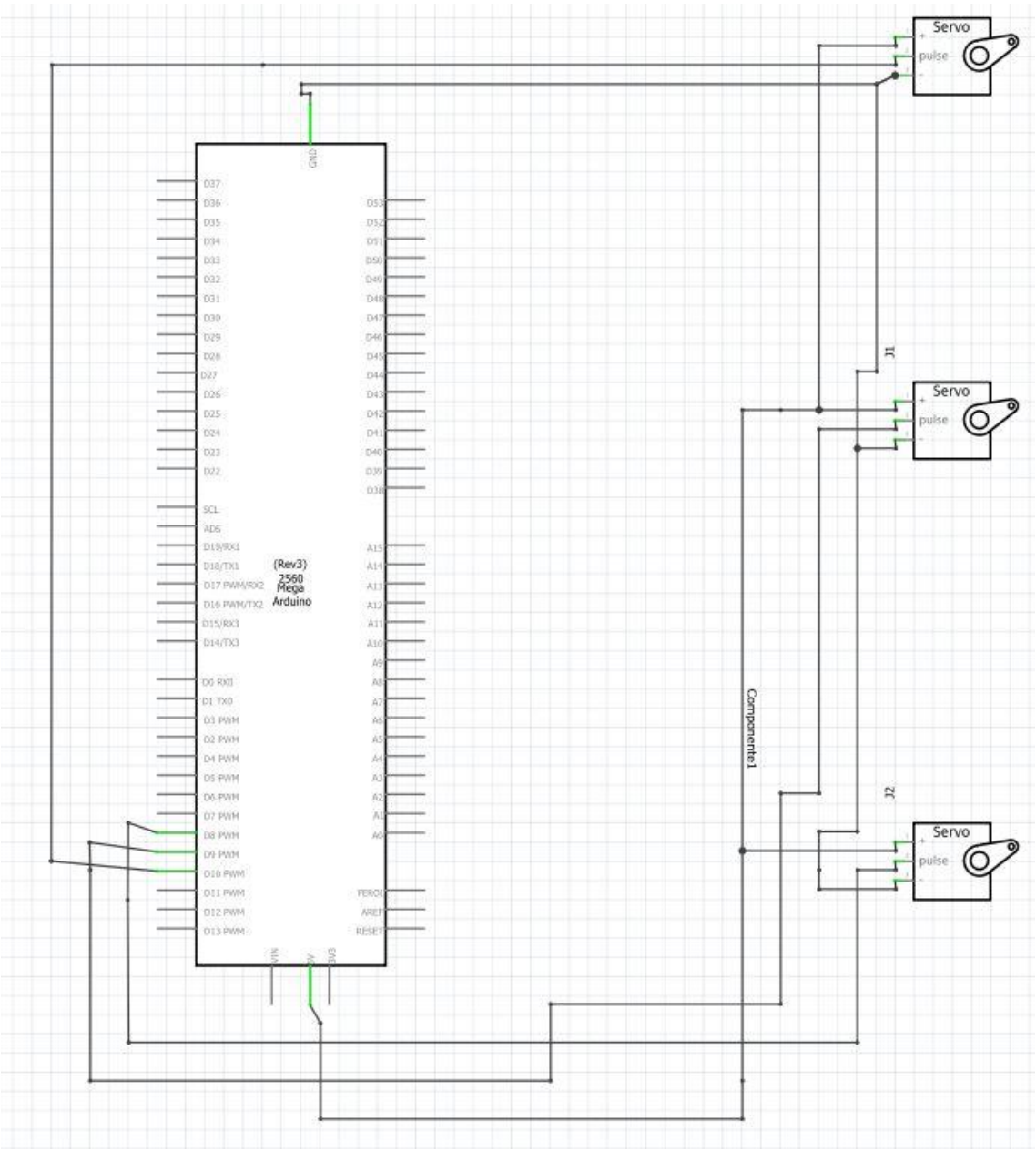

Ilustración 33 Esquema eléctrico de la conexión de los tres servomotores con Arduino 1.

4.3 Conexión de Arduino con acelerómetros.

La conexión de los acelerómetros se llevará a cabo con el denominado Arduino 2, con el objetivo de facilitar la explicación. Como se mencionó en el apartado de selección de acelerómetros, se van a realizar cuatro conexiones:

Los pines que se van a utilizar en el funcionamiento por parte del acelerómetro son los siguientes:

- VIN: Pin de alimentación del equipo (2.2 – 16 V).
- GND: Pin puesta a tierra del dispositivo.
- Eje Y: Aceleración que se experimenta en el eje Y.
- Eje Z: Aceleración que se experimenta en el eje Z.

Para realizar la conexión con Arduino se han unido los 2 terminales de los acelerómetros denominados Vin bajo un mismo conductor para realizar una fácil conexión con el pin Vin de la placa de Arduino 2. El mismo proceso se ha realizado con los dos terminales GND, conectando el conductor único formado, con el pin GND de la placa Arduino 2.

En cuanto a los ejes de medida de los acelerómetros, puesto que el Arduino 2 va a recibir señales, se han de configurar los puntos de conexión de la placa como entradas analógicas. Debido al software cargado en la placa de Arduino 2, la conexión debe ser realizada tal y como se mostrará a continuación:

Acelerómetro 1

Eje Y1 → pin analógico Nº A0

Eje Z1 → pin analógico Nº A1

Acelerómetro 2

Eje Y2 → pin analógico Nº A3

Eje Z2 → pin analógico Nº A4

Cabe destacar que no importa cuál de los dos acelerómetros es denominado como acelerómetro 1 o acelerómetro 2, solo hay que mantener la correlación eje – pin

indicada. Esta correlación es obligatoria en el modelo de cambio de escala incorporado en el software. Un conexionado distinto, provocaría medidas de aceleración erróneas.

La conexión se realiza de la siguiente manera (ilustración 34):

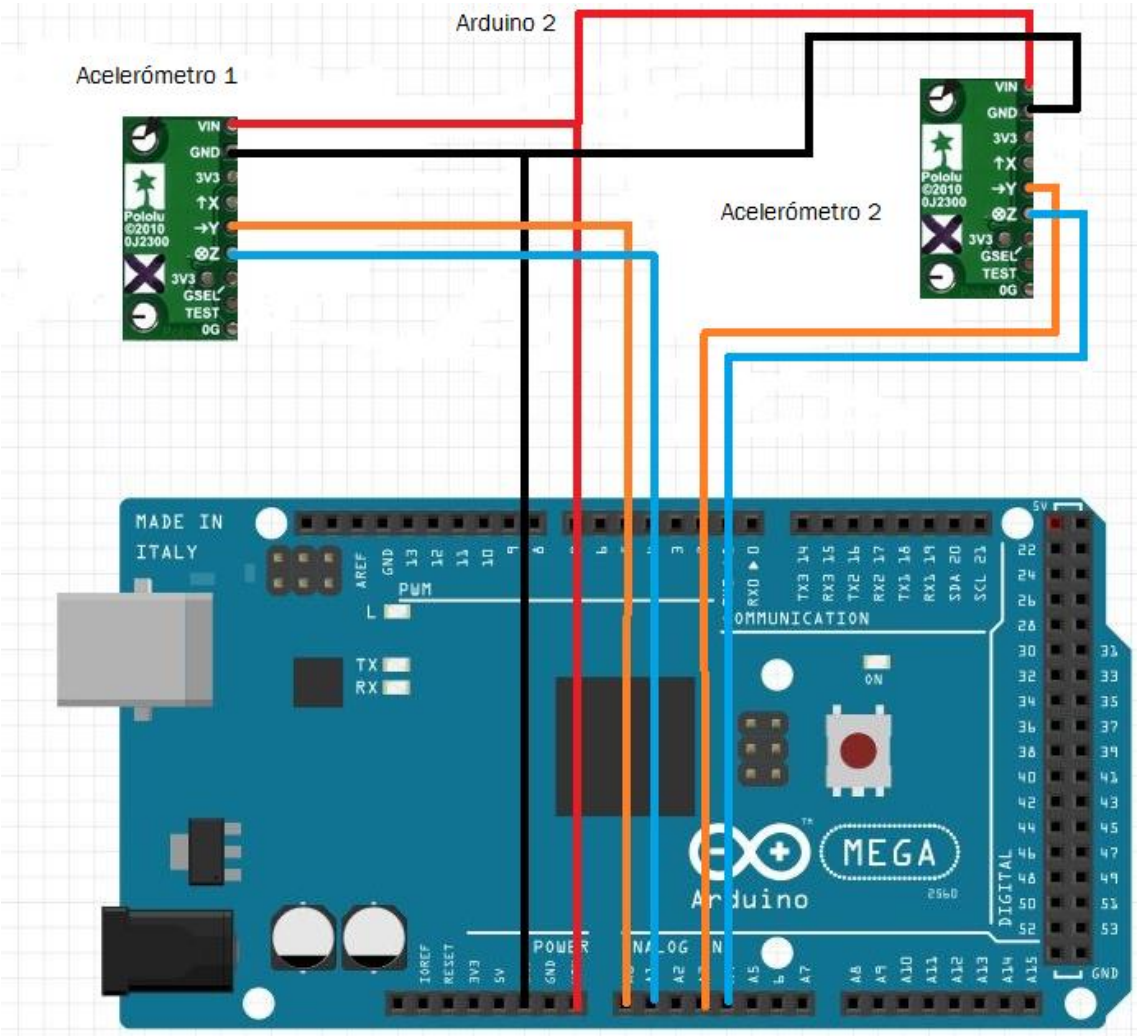

Ilustración 34 Conexión de los dos acelerómetros con Arduino 2.

4.4 Conexión de Arduino con PC. Leyenda de control de maqueta.

Una vez realizada la conexión de los distintos equipos con Arduino, se necesita un *interface* con el cual visualizar las posibilidades que ofrece la maqueta. Una de las opciones consiste en realizar la conexión de los Arduinos con un PC vía USB. Esta conexión permite la alimentación de la placa así como el envío y recibo de datos entre PC y la placa de Arduino.

Para realizar la interconexión de los elementos mencionados se necesita tener incorporado en el Pc un software denominado "Arduino" (adquirible en la página oficial de Arduino de manera gratuita).

Este software aparte de realizar la interconexión por puerto USB, permite programar el software interno de la placa de Arduino, además de un compilador, que asegura una correcta caligrafía del programa.

Inicialmente se selecciona el tipo de placa Arduino que se está utilizando y el puerto de conexión de la misma. Estas especificaciones se señalan en la ilustración 35.

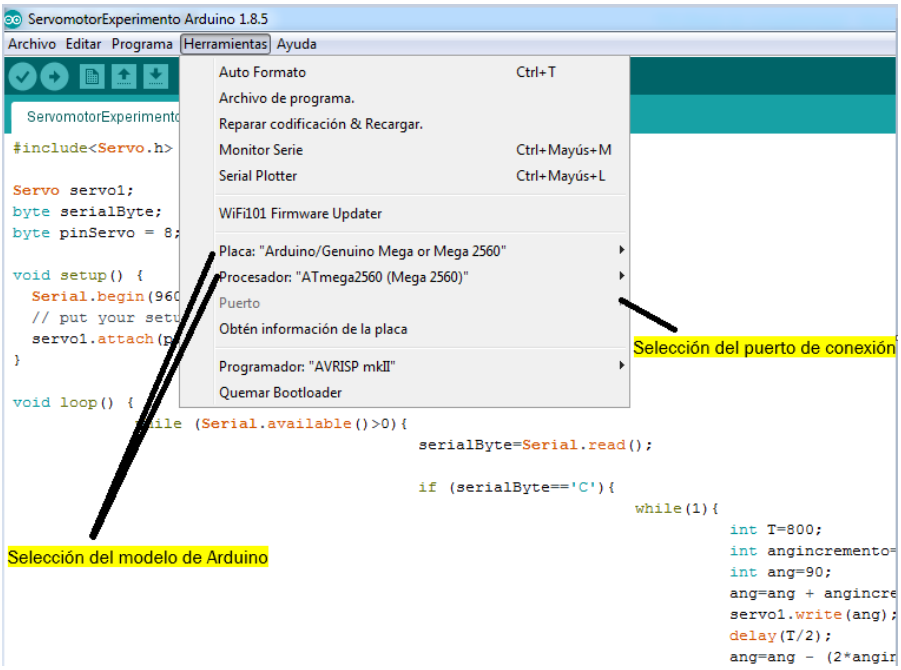

Ilustración 35 Guía para utilizar el programa Arduino.

Una vez se comprueba que la caligrafía es correcta (compilador), el programa se puede cargar en la placa (botón para subir programa). Los programas que se

deberían de cargar son los creados para los acelerómetros y servomotores indicados con anterioridad. El software denominado “Arduino”, incorpora una pestaña denominada “Monitor Serie”, lugar a través del cual se permite enviar caracteres o números, y recibir datos. Estas especificaciones se señalan en la ilustración 36.

Ilustración 36 Guía de funcionamiento del interface Arduino.

Los comandos que permiten la utilización de la maqueta a partir de este programa denominado “Arduino” (instalado en un Pc) son caracteres (letras) y números (números enteros) introducidos por la pestaña “Monitor Serie”:

A → Pongo en marcha el servomotor de excitación (Marcha servo 1).

B → Paro del servomotor de excitación (Paro servo 1).

C → Pongo en marcha el servomotor de amortiguamiento del péndulo (Marcha servo 2).

D → Paro del servomotor de amortiguamiento del péndulo (Paro servo 2).

E → Dejo libre el péndulo de disipación de balanceo (Marcha servo 3).

F → Encierro el péndulo de disipación de balanceo (Paro servo 3).

Introduzco número → Cambio el periodo del ciclo de giro del servomotor 1.

Cabe destacar que el número introducido debe ser entero y en milisegundos. A continuación se indica el intervalo de funcionamiento adecuado y para el cual se ha calibrado el sistema. Un periodo de ciclo de giro fuera del siguiente intervalo

indicado se considerará como erróneo y estará fuera de las garantías de trabajo del sistema.

Periodo máximo (ms)	Periodo de resonancia(ms)	Periodo mínimo (ms)
800	608	50

Tabla 20 Intervalo de periodos válidos de funcionamiento.

4.4 Conexión de Arduino con Raspberry.

4.4.1 Arquitectura del sistema.

Como se comentó en el apartado 2.5 Raspberry, la utilidad de este mini ordenador es enorme debido a las posibilidades de uso que se le pueden dar. En el caso de este proyecto esa utilidad será aprovechada para funcionar como un intermediario que permita la formalización de un *interface* desde el cual realizar un control más factible de las opciones que posee la maqueta en cuestión. Para ello se conectarán los dos Arduinos por medio del cable USB (cada uno a un conector USB) con la placa de Raspberry incluida en la pantalla LCD de 5 pulgadas.

Cabe destacar que este apartado, es decir, la inclusión del sistema Raspberry junto con el software necesario, no es objeto de estudio de este proyecto, no obstante, se buscará realizar una explicación básica de la arquitectura empleada en la comunicación de los distintos datos generados. El esquema de la arquitectura en el que se apoyará la explicación es el que aparece en la ilustración 34.

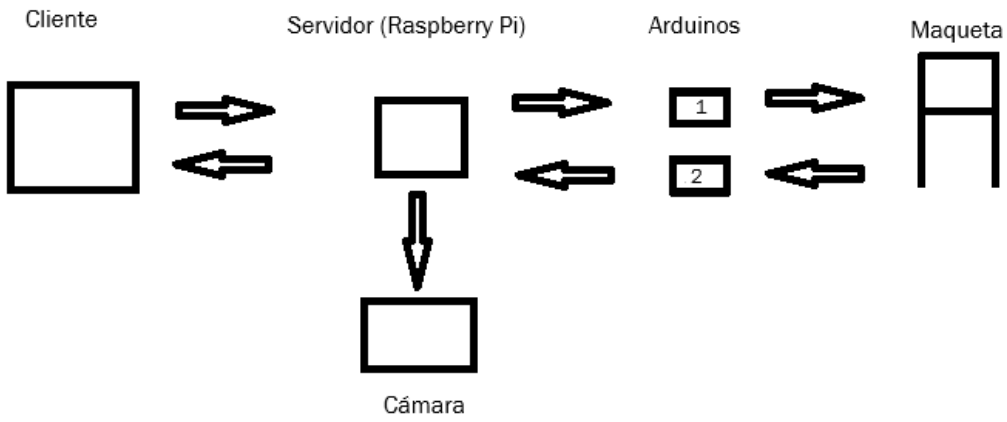

Ilustración 37 Arquitectura del sistema con la inclusión de Raspberry.

El desarrollo y objeto de estudio de este proyecto incluye hasta la programación software de los Arduinos. Por consiguiente Se va a utilizar la infraestructura de Raspberry como un servidor que contiene el software denominado cliente. Arduino y Raspberry se conectara vía USB (conexión por puerto serie).

El software que se ha denominado como cliente, funcionará como un emisor de órdenes (buscando el control de los servomotores o de la cámara), y como un receptor (recibe los datos de las medidas incluidas en los acelerómetros y las muestras en una gráfica). Como se ha indicado antes este software o programa será ejecutado por parte de un contenedor cuyo objetivo es hacer funcionar los programas. El contenedor se denomina Payara, el cual está incluido dentro de la Raspberry.

Por otro lado, Arduino, es un sistema autónomo que se encuentra ejecutando sus lazos de funcionamiento programados de manera continua siempre y cuando reciban energía. Esto quiere decir que en el caso del Arduino 1 se encuentra escuchando de manera continua el puerto al que está conectado, esperando la orden que cambie el estado de alguno de los servomotores. En el caso de Arduino 2, la transmisión de datos por el puerto serie recogidos de los acelerómetros se realiza de manera continua.

Por tanto el software cliente se encarga de activar él envìo o la escucha de los Arduinos y la activación de la cámara en función de las órdenes que se envíen externamente (por el usuario).

4.4.2 Integración en una plataforma Java. Interface.

Una vez entendida a groso modo la arquitectura del sistema cabe destacar que se ha creado un entorno Java con el cual optimizar y facilitar la utilización de la maqueta. Se ha abandonado la idea de utilizar la pantalla LCD en la que está incluida Raspberry debido a problemas en el desplazamiento de la maqueta por su voluminosidad y peso. Motivo por el que se ha generado un programa que ejecutándolo en el pc, directamente se conecta a la red a la cual se encuentra conectado el servidor Raspberry. Por lo tanto su accesibilidad es mundial. Este programa despliega un conjunto de opciones con las que controlar los servomotores, graficar los datos de los acelerómetros, controlar la cámara e introducir valores enteros en milisegundos para controlar el ciclo de giro del péndulo excitador. A mayores se incorpora un botón con el cual permitir o no la introducción de eventos externos.

Debido a que el programa se encuentra en desarrollo no se puede incorporar una guía de funcionamiento.

Cabe destacar que el *software* cargado en el Arduino 1 es el mismo que se carga en el caso de conexión de Arduino con Pc. Por el contrario, para el caso de las lecturas de los acelerómetros, ha sido necesario generar un software distinto para el Arduino 2, denominado “Acelerometros_Raspbery_Media50”.

El software que controla Raspberry, realiza la lectura solamente del eje Y1 y del eje Y2. Para realizar la gráfica de las medidas de aceleración se necesita recibir por el puerto serie de Raspberry desde Arduino 2, las medidas de aceleración en un formato determinado (ver Anexo).

Una vez se realiza la prueba para comprobar que la formalización de la gráfica a partir de las medidas de aceleración es correcta, se observa que no lo es. El error es provocado por un cúmulo de ruido.

Para solucionar este problema, en el *software* cargado en Arduino 2, se incorpora un bucle en el que se realiza la media de 50 medidas, para generar el dato que posteriormente será graficado. Solución utilizada para las gráficas generadas a partir de las medidas del eje Y1 y el eje Y2.

5. Medidas futuras de mejora de la maqueta.

A continuación se van a exponer un conjunto de ideas sobre la maqueta en cuestión. Estas ideas se encaminan a una futura mejora de los sistemas de medición, los elementos mecánicos, el software empleado, la calibración de los equipos utilizados etc.

Las ideas que en este apartado se indican, tienen como objetivo un desarrollo más especializado de este primer prototipo, con lo que afianzar su utilidad, mejorar sus características, aportando datos de manera más precisa. Por otro lado, se abre la posibilidad de que el proyecto sea continuado con visión de mejora por otro alumno en relación al desarrollo de su TFG.

- Como se explicó en el apartado de calibración de servomotores, uno de los puntos críticos del funcionamiento correcto de la maqueta consiste en que el servomotor de excitación de la estructura no se descentre. Por tanto, una de las más importantes medidas de mejora a tener en cuenta consiste en asegurar el centrado del ciclo de giro en el péndulo de excitación. Como sugerencia, una posibilidad de llevar a la realidad esta idea consistiría en dividir el ciclo de giro en varios pasos y a su vez generar distintos intervalos de tiempo que en su conjunto formarían el tiempo de ciclo de giro. Apostando por intervalos de tiempo distinto, siendo de mayor duración los desplazamientos hacia la posición de centrado, e intervalos de menor duración en el desplazamiento del péndulo a sus extremos. Profundizando en el tema, se comprobaría la validez de esta idea.

- Otro de los problemas encontrados en la maqueta consiste en la limitación de los servomotores utilizados. Se sugiere la búsqueda de un remplazo para ellos, el cual posea una capacidad de giro mayor y el cual sea más robusto con el fin de evitar la aparición de holguras (se aumentaría la vida útil de la maqueta). En cuanto a que la capacidad de giro sea mayor, se incide en este punto con el objetivo de aumentar el rango de tiempos de ciclo de giro válidos para hacer funcionar la maqueta de manera correcta.

- Realizar la calibración del eje Z y el eje X a partir de una población de datos obtenida de dos equipos de medición distintos, es decir, realizar el proceso de manera individual para la obtención del modelo de cambio de escala como se realiza con el eje Y, para los ejes Z y X.

- Búsqueda de equipos de medida de aceleración más precisos. Los acelerómetros utilizados en la maqueta pierden bastante precisión debido a que su funcionamiento se basa en la emisión de una señal de salida en función de la variación de capacidad producida por el desplazamiento de placas mediante unos muelles. Estos resortes introducen un error en la medida debido a su balanceo fruto de la fuerza elástica. Se sugiere de la utilización de acelerómetros piezoeléctricos, los cuales solucionarían el problema aportando mayor precisión a los datos recogidos.

- Se sugiere investigar una modificación del software, con el objetivo de que se pueda controlar el ángulo girado por el servomotor de excitación como se hace con el periodo del ciclo de giro.

- Otra de las posibilidades de la maqueta, consistiría en general un control de lazo cerrado, el cual manipulase de manera automática la amortiguación del péndulo mediante la chapa (control del servomotor de amortiguamiento del péndulo) en función de los valores de medida de aceleración recogidos de los acelerómetros. Para ello se necesitaría una modificación de software.

6. Costes.

A continuación se va exponer el conjunto de precios de los distintos equipos utilizados en el sistema elaborado. Cabe destacar que el cálculo de costes se va a realizar con el montaje del sistema en el que se utiliza la infraestructura Raspberry, incluyendo el número de horas de trabajo utilizadas en el montaje y desarrollo del mismo.

Descripción del artículo	Cantidad	Precio (€/ unidad)	Importe (+ 21% IVA) (€)
Cámara Raspberry 5MP	1	5,62	6,8
Raspberry Pi 3 modelo B	1	29,47	35,66
Arduino Mega 2560	2	34,17	84,7
Servomotores SG90	3	2,99	10,85
MMA7361LC. Pololu 0j7193	1	6,25	7,56
PL. Aluminio 15 x 3 1 metro	2	2	4,84
Placa policarbonato 15 x 80 x 160	2	1,975	5
Cableado y accesorios varios (aprox)			80
Horas montaje y desarrollo	330	12	3960
TOTAL		3313,23	4193,97

Tabla 21 Despiece de costes del sistema montado

7. Conclusiones.

En el apartado de conclusiones se va a tener en cuenta el conjunto de objetivos que se han cumplido de los mencionados en el apartado uno (1.Introducción). Por otro lado se darán a conocer el conjunto de competencias que ha aportado al alumno la consecución de este TFG a nivel personal, lo que ha significado el desarrollo del aprendizaje. Consecución de objetivos:

- Obtención del equipo adecuado.

En cuanto a la obtención del equipo adecuado cabe destacar que se ha realizado una selección de componentes para la consecución de la maqueta buscando el funcionamiento y no la durabilidad ni la precisión. La máxima que se buscaba es obtener un funcionamiento correcto de la maqueta. Una vez alcanzado ese punto, buscar una buena calibración de la misma y una mejora de su fiabilidad, durabilidad y precisión. De esta manera en el apartado cinco (5. Medidas futuras de mejora de la maqueta) se insistió en la búsqueda de acelerómetros más precisos y servomotores de mayor durabilidad con el fin de evitar holguras y rápido deterioro.

- Probar la validez del equipo elegido.

Como se ha demostrado en los apartados, 2.3 Servomotores y 2.4 Acelerómetros, los equipos elegidos permiten el desarrollo funcional de la maqueta de manera correcta.

- Generar suficiente perturbación en la estructura para provocar un balanceo resonante.

En el apartado 3.2.1.2 Calibración servomotor de excitación. Periodo, se expone el experimento realizado para la obtención del periodo de resonancia de la estructura. El intervalo de funcionamiento descrito y comprobado permite que se produzca este fenómeno.

- Obtener la calibración adecuada del servomotor que genera la perturbación para poder cambiar el balanceo de la estructura variando el periodo del ciclo.

En el apartado 3.2.1.2 Calibración servomotor de excitación. Periodo se calcula el intervalo de periodos válidos para la excitación de la maqueta, es decir, periodos válidos para generar el balanceo. Como se ha explicado, la implementación *software* realizada permite variar el periodo de giro del servomotor 1.

- Obtener la calibración adecuada del servomotor que permite la actuación del péndulo de disipación de balanceo.

En el apartado 3.2.3 Servomotor de activación del péndulo de disipación (servomotor 3), se incide la manera de proceder para realizar la calibración adecuada del servomotor, gestionando su funcionamiento y destacando la importancia de poner atención para no dañar el equipo utilizado por choques entre elementos móviles y paredes fijas.

- Obtener la calibración adecuada del servomotor que interacciona con el péndulo de disipación de balanceo eliminando parte de su capacidad de disipación.

En el apartado 3.2.2 Servomotor de amortiguamiento del péndulo (servomotor 2), se incide la manera de proceder para realizar la calibración adecuada del servomotor, gestionando su funcionamiento y destacando la importancia de poner atención para no dañar el equipo utilizado por choques entre elementos móviles y paredes fijas.

- Obtener la calibración adecuada de los acelerómetros, realizando el cambio de escala oportuno para obtener mediciones de aceleración en m/s^2.

En el apartado 3.3 Calibración de acelerómetros, se exponen los experimentos llevados a cabo para realizar el cálculo del modelo que permita el cambio de escala a partir de las medidas generadas en Arduino por las lecturas de los acelerómetros. Cabe destacar la importancia del tratamiento de los datos, lo cual otorga la precisión del modelo generado.

- Generar el software necesario para realizar las calibraciones individuales de los equipos.

Como se incluye en el apartado Programas del Anexo, se ha generado un *software* lo más liviano posible, con el objetivo de obtener una fácil comprensión. Los conocimientos han sido obtenidos de innumerables lugares web (ver Anexo), así como en el diccionario de comandos de Arduino, donde se ha realizado el estudio de los procesos a tener en cuenta, utilizados y que han sido comentados en el código del programa.

- Generar el software necesario para realizar el control del sistema conjunto.

Como se incluye en el apartado Programas del Anexo se ha generado un software a partir de los programas utilizados para la calibración de equipos de manera individual. Los conocimientos han sido obtenidos de innumerables lugares web, así como en el diccionario de comandos de Arduino, donde se ha realizado el estudio de los procesos a tener en cuenta, utilizados y que han sido comentados en el código del programa.

- Determinar los parámetros correctos de funcionamiento de la maqueta, es decir realizar una guía de instrucciones que permita su fácil utilización.

Con el interface que se ha desarrollado para Raspberry, el funcionamiento de la maqueta se ha facilitado lo más posible. Las instrucciones pertinentes para su manejo se recogen en el apartado 4.4.2 Integración en una plataforma Java. Interface.

En el caso de Arduino se ha generado una guía explicada en el apartado 4.4 Conexión de Arduino con PC. Leyenda de control de maqueta.

Por otro lado, una vez cubierto el completo de los objetivos en mayor o menor medida, se va a proceder a explicar el conjunto de competencias personales adquiridas con el desarrollo del TFG.

- He aprendido a elaborar un proyecto desde los cimientos, mediante la búsqueda de componentes en la web que permitieran desarrollar el conjunto de necesidades que suponía la maqueta.

- He aprendido a gestionar los problemas que iban apareciendo durante el desarrollo de los distintos apartados, observando que de vez en cuando hay que dar un giro a lo planeado que permita la obtención de los resultados requeridos.

- He aumentado en gran medida mis conocimientos sobre la plataforma Arduino, llegando a dominar la programación software relevante al proyecto y he conseguido expandir mi visión de la utilidad de este microcontrolador.

- Me ha permitido conocer de manera básica el entorno Raspberry.

- He aprendido a calibrar los distintos equipos inmersos en la maqueta de manera real, observando los problemas más comunes que pueden surgir. Mantener especial cuidado cuando se trabaja con elementos móviles, modelos aproximados pero no exactos, cálculos teóricos pero no exactos en la realidad etc.

- He aprendido a trabajar de manera individual y de manera colectiva, desarrollando avances propios con mi trabajo, que posteriormente he explicado a mis supervisores, optimizando a partir de sus sugerencias los métodos utilizados y los resultados obtenidos.

- He aprendido que la consecución de un trabajo un tanto mayor a los que acostumbro a realizar, necesita un esfuerzo constante sin perder la paciencia frente a resultados no deseados y la aparición de problemas.

8. Webgrafía.

A continuación se van a incluir el conjunto de enlaces web de los que se ha obtenido información.

En los enlaces siguientes se indica una guía básica para dar los primeros pasos en programación de servomotores:

https://programarfacil.com/tutoriales/fragmentos/servomotor-con-arduino/

http://www.electroensaimada.com/servomotor.html

Información básica para el control de servomotores:

https://www.luisllamas.es/controlar-un-servo-con-arduino/

Información básica de cómo funciona un acelerómetro:

https://www.luisllamas.es/como-usar-un-acelerometro-arduino/

Información básica sobre acelerómetros capacitivos:

http://panamahitek.com/acelerometros-de-3-ejes-lo-que-necesitas-saber/

En esta página he consultado el funcionamiento completo de los distintos comandos utilizados en el programa:

https://www.arduino.cc/

Información básica sobre Arduino AT Mega:

http://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

En esta página conseguí una idea general sobre Raspberry PI:

http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/

Ejemplos del comando *“milis()”*:

https://hetpro-store.com/TUTORIALES/arduino-millis/

Tutorial del programa “Fritzing” con el que se han construido los esquemas eléctricos y los dibujos de conexión:

https://www.youtube.com/watch?v=AA1HVkbDxJQ

Anexo

CARACTERÍSTICAS DE LOS EQUIPOS EMPLEADOS.

Datasheet servomotor “Tower Pro MG995”:

http://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

Datasheet microcontrolador “Raspberry Pi 3 model B”:

https://docs-emea.rs-online.com/webdocs/14ba/0900766b814ba5fd.pdf

Características servomotor “LOBOT LD-20MG”:

https://www.banggood.com/es/LOBOT-LD-20MG-20KG-Large-Torque-Metal-Gear-Digital-Servo-for-RC-Models-p-1147714.html?cur_warehouse=CN

Datasheet servomotor “SG90 Micro Servo”:

http://akizukidenshi.com/download/ds/towerpro/SG90.pdf

Datasheet de la placa de Arduino “Mega 2560”:

http://html.alldatasheet.com/html-pdf/897466/ATMEL/MEGA2560/151/1/MEGA2560.html

Datasheet acelerómetro “MMA7361LC”:

https://www.nxp.com/docs/en/data-sheet/MMA7361LC.pdf

Especificaciones acelerómetro montado en circuito impreso “Pololu 0j7193”

https://www.pololu.com/product/1251

DIRECCIÓN WEB DE IMÁGENES.

En este apartado se incluye la dirección web de las imágenes que han sido utilizadas en esta memoria a modo explicativo y que no han sido creadas por mí. .

Imagen de Arduino Mega 2560 (ilustración 1):

https://www.google.com/search?q=arduino+Mega2560&client=firefox-b-ab&source=lnms&tbm=isch&sa=X&ved=0ahUKEwirgbX8t4PcAhWEwxQKHUC0ApEQ_AUICigB&biw=1280&bih=913#imgrc=ZJZzmYvcIBcJxM:

Imagen de servomotor (ilustración 2):

https://www.google.com/search?q=servomotores&client=firefox-b-ab&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj70YKauYPcAhWF0RQKHag2AgQQ_AUICigB&biw=1280&bih=913#imgrc=hEZHJzgkoPBatM:

Imagen del control PWM de un servomotor, *cicle duty (ilustración 4):*

https://www.google.com/search?q=cycle+duty+servomotores&client=firefox-b-ab&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjWgZqxuoPcAhWLWhQKHS1zB7QQ_AUICigB&biw=1280&bih=913#imgrc=fag3hirBv63eQM:

Imagen del servomotor “Tower Pro MG955” (ilustración 5):

https://www.google.com/search?q=tower+pro+MG995&client=firefox-b-ab&source=lnms&tbm=isch&sa=X&ved=0ahUKEwixvL7mu4PcAhULPRQKHRrEADoQ_AUICigB&biw=1280&bih=913#imgrc=kmiVIB3fXfJGZM:

Imagen servomotor “SG90 Micro Servo” (ilustración 11):

https://www.google.com/search?q=SG90+micro+servo&client=firefox-b-ab&source=lnms&tbm=isch&sa=X&ved=0ahUKEwic-cDZv4PcAhWJWxQKHbovDgIQ_AUICigB&biw=1280&bih=913#imgrc=VLAhg0OJ0TvmgM:

Acelerómetro capacitivo (ilustración 15):

https://www.google.com/search?q=acelerometro+capacitivo+disposicion+de+las+placas&client=firefox-b-ab&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjUusDEwoPcAhUCthQKHYQOAeQQ_AUICigB&biw=1280&bih=913#imgrc=xQ8tEMSo84rePM:

Acelerómetro “Pololu 0j7193” (ilustración 16):

https://www.google.es/search?q=pololu+MMA7361LC&source=lnms&tbm=isch&sa=X&ved=0ahUKEwir3KXgyoPcAhWKOxQKHadeC2cQ_AUICygC&biw=1304&bih=702#imgrc=Tw0R7HPrOtwW1M:

Raspberry Pi Model 3 (ilustración 18):

https://es.rs-online.com/web/p/products/8968660/?grossPrice=Y&cm_mmc=ES-PLA-DS3A-_-google-_-PLA_ES_ES_Semiconductores-_-Kits_De_Desarrollo_Para_Semiconductor%7CKits_De_Desarrollo_De_Procesador_Y_Microcontrolador-_-PRODUCT+GROUP&matchtype=&gclid=EAIaIQobChMIxde2uc6D3AIVxp0bCh33MQxjEAQYASABEgLLVPD_BwE&gclsrc=aw.ds.

Raspberry Pi y pantalla LCD (ilustración 19):

https://es.aliexpress.com/store/product/Raspberry-Pi-3-Model-B-Board-Kit-5-Inch-LCD-HDMI-USB-Touch-Screen-5V-2/1708600_32800660886.html?spm=a219c.search0104.3.114.22552fabFSp5ay&ws_ab_test=searchweb0_0,searchweb201602_2_10065_10344_10068_10547_10342_10343_10340_10548_10341_10084_10083_10618_10307_10301_10303_10313_10059_10184_10534_100031_10103_441_10624_442_10623_10622_10621_10620_10142,searchweb201603_37,ppcSwitch_4&algo_expid=cebef6f0-e525-40be-afbd-8adcf0d20c8e-16&algo_pvid=cebef6f0-e525-40be-afbd-8adcf0d20c8e&priceBeautifyAB=0

PROGRAMAS

A continuación se van a exponer los programas utilizados que se incluyen en los Arduinos para realizar el control de la maqueta, desde un PC con el programa Arduino.

Acele_calibra()

byte pinAcel = A0; // Referenciamos a una variable el pin A0 analógico.

float K=0.5; //Constante que multiplica el valor de salida del acelerómetro para obtener la aproximación real de aceleración.

int b1= -180; //Constante que se suma al valor de salida del acelerómetro para obtener la aparoximación real de la aceleración en el eje.

int eje = 0; //Variable que guarda la lectura del acelerómetro 1 (valor entre 0 y 1023) del eje.

float acelReal=0; // Variable que almacena el valor fruto de la conversión de la lectura del eje en m/s^2.

byte serialByte;

```
String readString; //Variable en la que guardamos la cadena de texto que se introduce por el puerto serie.

char c;// Guarda el valor de readString.

boolean lectura;//Variable que guarda el estado para leer o no los acelerómetros (activa/desactiva)

void setup() {


 Serial.begin(9600);

 Serial.println(" Introduce G para recibir datos y H para detener la lectura de datos");

 pinMode(A0, INPUT); //Definimos el PIN A0 como entrada.

}

void loop() {

 while (Serial.available()) {

                    c = Serial.read();  //gets one byte from serial buffer

                    readString += c; //makes the string readString

                    delay(2);

  }


   if (readString.length() > 0) {

      Serial.println(readString);


                        if (readString == "G") { //Se activa el pulsador del servo 1.

                                    lectura = 1;

                        }

                        if(readString=="H")  { //Se pone en stop el pulsador del servo 1.

                                    lectura = 0;

                        }
```

```
                    Serial.print("lectura: ");
                    Serial.println(lectura);
                    readString="";
  }

   if (lectura){

          eje = analogRead(pinAcel);
          acelReal= K*eje+b1;
          Serial.println(acelReal);
          delay(150);
     }
}
```

Servomotores()

```
byte pinAcelY1 = A0; //Referenciamos a una variable el pin A0 analógico.
byte pinAcelZ1= A1; //Referenciamos a una variable el pin A1 analógico.
byte pinAcelY2=A3; //Referenciamos a una variable el pin A3 analógico.
byte pinAcelZ2=A4; //Referenciamos a una variable el pin A4 analógico.
float K=0.5; //Constante que multiplica el valor de salida del acelerómetro para obtener la aproximación real de aceleración.
int b1= -179; //Constante que se suma al valor de salida del acelerómetro para obtener la aproximación real de la aceleración en y1.
int b2= -469;//Constante que se suma al valor de salida del acelerómetro para obtener la aproximación real de la aceleración en y2.
int c1=-223;//Constante que se suma al valor de salida del acelerómetro para obtener la aproximación real de la aceleración en z1.
int c2=-448;//Constante que se suma al valor de salida del acelerómetro para obtener la aproximación real de la aceleración en z2.
```

int ejeY1 = 0; //Variable que guarda la lectura del acelerómetro 1 (valor entre 0 y 1023) del eje Y.

float acelRealY1=0; // Variable que almacena el valor fruto de la conversión de la lectura del eje Y1 en m/s^2.

float acelRealZ1=0;// Variable que almacena el valor fruto de la conversión de la lectura del eje z1 en m/s^2.

int ejeZ1 = 0;//Variable que guarda la lectura del acelerómetro 1 (valor entre 0 y 1023) del eje Z.

int ejeY2=0;//Variable que guarda la lectura del acelerómetro 2 (valor entre 0 y 1023) del eje Y.

float acelRealY2=0;// Variable que almacena el valor fruto de la conversión de la lectura del eje Y2 en m/s^2.

float acelRealZ2=0;// Variable que almacena el valor fruto de la conversión de la lectura del eje Z2 en m/s^2.

int ejeZ2=0;//Variable que guarda la lectura del acelerómetro 2 (valor entre 0 y 1023) del eje Z.

byte serialByte;

String readString; //Variable en la que guardamos la cadena de texto que se introduce por el puerto serie.

char c;// Guarda el valor de readString.

boolean lectura;//Variable que guarda el estado para leer o no los acelerómetros (activa/desactiva)

void setup() {

  Serial.begin(9600);

  Serial.println("Empezamos a recibir datos...");

  pinMode(A0, INPUT); //Definimos el PIN A0 como entrada.

  pinMode(A1,INPUT); //Definimos el PIN A1 como entrada.

  pinMode(A3, INPUT); //Definimos el PIN A3 como entrada.

  pinMode(A4, INPUT); //Definimos el PIN A4 como entrada.

}

```
void loop() {


 // A continuació se abre la escucha del puerto a la espera de los caracteres G y H con los que inciar la lectura de

 // los acelerómetros.

while (Serial.available()) {

    c = Serial.read();  //gets one byte from serial buffer

    readString += c; //makes the string readString

    delay(2);

  }


   if (readString.length() > 0) {

      Serial.println(readString);


                        if (readString == "G") { //Se activa el pulsador del servo 1.

                        lectura = 1;

                        }

                        if(readString=="H")  { //Se pone en stop el pulsador del servo 1.

                        lectura = 0;

                        }

                        Serial.print("lectura: ");

                        Serial.println(lectura);

                        readString="";

   }

 //A continuación se hace la lectura de los pines conectados a los acelerómetros, se realiza el cambio de escala y
```

```
// se muestra por pantalla (se envian por el puerto serie)
    if (lectura){
      float media1=0;
      float media2=0;
      float media3=0;
      float media4=0;

      int i=0;
  // Se hace la media de 50 interaciones y se imprime el dato por pantalla del dato con cambio de escala
       for(i=0;i<50; i++){
            ejeZ1 = analogRead(pinAcelZ1); // Lectura del PIN A0
            acelRealZ1= K*ejeZ1+c1; // Conversión a m/s^2
            media1= media1 + acelRealZ1;

            ejeY1 = analogRead(pinAcelY1);
            acelRealY1= K*ejeY1+b1;
            media2= media2 + acelRealY1;

            ejeZ2 = analogRead(pinAcelZ2);
            acelRealZ2= K*ejeZ2+c2;
            media3= media3 + acelRealZ2;

            ejeY2 = analogRead(pinAcelY2);
            acelRealY2=K*ejeY2+b2;
            media4= media4 + acelRealY2;
            delay(10);
```

```
        }

        media1= media1/50;
        media2= media2/50;
        media3= media3/50;
        media4= media4/50;

           Serial.println("Eje Z acelerómetro 1: ");
           Serial.println(media1); // Imprimimos el valor real.
           Serial.println("EJE Y acelerometro1 en m/s^2: ");
           Serial.println(media2);
           Serial.println("Eje Z acelerómetro 2: ");
           Serial.println(media3);
           Serial.println("EJE Y acelerómetro 2 en m/s^2: ");
           Serial.println(media4);
      }

}
```

Servomotores()

```
#include<Servo.h>
```

String readString; //Variable en la que guardamos la cadena de texto que se introduce por el puerto serie.

volatile int T=608; // Periodo de oscilación del servo1 en milisegundos.

int Periodo=0;// variable en la que guardamos el numero procedente de la conversión de la cadena de texto.

boolean pulsadorServo1;//Activar/desactivar servo1 durante el tiempo que se mantenga en funcionamiento el programa.

char c;// Guarda el valor de readString.

//Creamos los tres servomotores e indicamos los pines a partir de los cuales se realizará el control PWM.

```
Servo servo1;

byte pinServo1 = 8;


Servo servo2;

byte pinServo2=9;


Servo servo3;

byte pinServo3=10;


void setup() {


        Serial.begin(9600);

        servo1.attach(pinServo1);

        servo2.attach(pinServo2);

        servo3.attach(pinServo3);



 }


void loop() {
```

// Si se activa el puerto, leemos y vemos si se trata de una letra o de un número. Consecuentemente se ejecuta la acción correspondiente.

```
  while (Serial.available()) {
                        c = Serial.read();  //gets one byte from serial buffer
                        readString += c; //makes the string readString
                        delay(2);
  }
```

// A continuación se programan los movimientos de los servomotores en función de los caracteres o el intervalo de periodos válidos

//que se exponen en la leyenda

```
   if (readString.length() > 0) {
      Serial.println(readString);

                        if (readString == "A") { //Se activa el pulsador del servo 1.
                                    pulsadorServo1 = 1;
                        }else if (readString=="B")  { //Se pone en stop el pulsador del servo 1.
                                    pulsadorServo1= 0;
                        } else if (readString=="C") { //Se pone en marcha el servo 2 realizando el giro descrito.

                                    Serial.print("Servo 2 ABRIR ");
                                    servo2.write(0);
                                    delay(300);
```

```
} else if (readString=="D") {//Se pone en marcha el servo 2 realizando el giro descrito.

                    Serial.print("Servo 2 CERRAR ");
                    servo2.write(90);
                    delay(300);

} else if (readString=="E"){ //Se pone en marcha el servo 3 realizando el giro descrito.

                    Serial.print("Servo 3 ABRIR ");
                    servo3.write(180);
                    delay(300);

} else if (readString=="F") { //Se pone en marcha el servo 3 realizando el giro descrito.

                    Serial.print("Servo 3 CERRAR ");
                    servo3.write(90);
                    delay(300);

} else {

          Periodo = readString.toInt();  //convert readString into a number

          if (Periodo > 0 ) {

                    T=Periodo;
```

```
                                        Serial.print("Periodo T: ");
                                        Serial.println(T);
                                        pulsadorServo1 = 1;


                                }
    }
     Serial.print("Servo1: ");
     Serial.println(pulsadorServo1);
     readString="";
  }
```

//Si el pulsador del servo 1 se encuentra activo se realiza el siguiente ciclo de giros en función del periodo de oscilación T.

```
      if (pulsadorServo1){


                int angincremento=90;
                int ang=90;
                ang=ang + angincremento;
                servo1.write(ang);
                delay(T/2);
                ang=ang - (2*angincremento);
                servo1.write(ang);
                delay(T/2);



     }
}
```

ServomotorExperimento()

```
#include<Servo.h> // biblioteca de los servomotores.


Servo servo1; // Se define la clase servo

byte serialByte;

byte pinServo = 8;


void setup() {

  Serial.begin(9600);

  servo1.attach(pinServo); // Se asigna la clase servo 1 al pin 8 (control PWM)

}


void loop() {


  // Se inicia la escucha de caracteres del puerto serie

          while (Serial.available()>0){

                                  serialByte=Serial.read();

  // Introduciendo el caracter C se inicia el movimento del servo, introduciendo el caracter F se para el servo.

  // Modificando la valiable T y angincremento se varía el periodo (T) y el angulo girado (angincremento)

                                  if (serialByte=='C'){

                                                  while(1){

                                                          int T=800;

                                                          int angincremento=90;

                                                          int ang=90;
```

```
                                        ang=ang + angincremento;
                                        servo1.write(ang);
                                        delay(T/2);
                                        ang=ang - (2*angincremento);
                                        servo1.write(ang);
                                        delay(T/2);

                                        if (Serial.available()>0){

serialByte=Serial.read();
                                                    if (serialByte=='F')
break;
                                                    }
                                    }
                                }
                            }
    }
```

Acelerometros_Raspberry_Media50

```
byte pinAcelY1 = A0; //Referenciamos a una variable el pin A0 analógico.
byte pinAcelY2=A3; //Referenciamos a una variable el pin A3 analógico.
float K=0.5; //Constante que multiplica el valor de salida del acelerómetro para obtener la aproximación real de aceleración.
int b1= -179; //Constante que se suma al valor de salida del acelerómetro para obtener la aproximación real de la aceleración en y1.
int b2= -455;//Constante que se suma al valor de salida del acelerómetro para obtener la aproximación real de la aceleración en y2.
int ejeY1 = 0; //Variable que guarda la lectura del acelerómetro 1 (valor entre 0 y 1023) del eje Y.
```

float acelRealY1=0; // Variable que almacena el valor fruto de la conversión de la lectura del eje Y1 en m/s^2.

int ejeY2=0;//Variable que guarda la lectura del acelerómetro 2 (valor entre 0 y 1023) del eje Y.

float acelRealY2=0;// Variable que almacena el valor fruto de la conversión de la lectura del eje Y2 en m/s^2.

byte serialByte;

String readString; //Variable en la que guardamos la cadena de texto que se introduce por el puerto serie.

char c;// Guarda el valor de readString.

boolean lectura;//Variable que guarda el estado para leer o no los acelerómetros (activa/desactiva)

```
void setup() {


  Serial.begin(9600);

  Serial.println("Empezamos a recibir datos...");

  pinMode(A0, INPUT); //Definimos el PIN A0 como entrada.

  pinMode(A3, INPUT); //Definimos el PIN A3 como entrada.


}




void loop() {




while (Serial.available()) {

                    c = Serial.read();  //gets one byte from serial buffer

                    readString += c; //makes the string readString

                    delay(2);
```

```
  }
// A continuación se declaran las variables utilizadas para calcular la media de los siguientes 50 valores leídos.
//Las 50 lecturas y acumulaciones de medida se realizan dentro del bucle for.
   float media1=0;
   float media2=0;
   int i=0;

      for(i=0;i<50; i++){

            ejeY1 = analogRead(pinAcelY1);
            acelRealY1= K*ejeY1+b1;
            media1= media1 + acelRealY1;
            ejeY2 = analogRead(pinAcelY2);
            acelRealY2=K*ejeY2+b2;
            media2= media2 + acelRealY2;
            delay(1);
      }

      media1= media1/50;
      media2= media2/50;

// Formato de salida para la graficación de medidas en Raspberry
      Serial.print("{");
      Serial.print(media1);
      Serial.print(",");
      Serial.print(media2);
```

```
        Serial.println("}");


         delay(10);


}
```