Taller de Automatización y Control (TA135)

Trabajo Práctico N°1

Introducción al control digital

1°C 2025

Integrantes del Grupo 4		
Nombre y apellido	Padrón	Correo
Manuel Lalía	107979	mlalia@fi.uba.ar
Felipe Perassi	107630	fperassi@fi.uba.ar

1. Introducción

El presente trabajo práctico tiene como objetivo caracterizar los componentes que se van a utilizar durante el desarrollo del proyecto cuatrimestral de la materia Taller de Control (TA135). Para esto, se realizan distintas pruebas en los sensores y actuadores a fin de conocer sus límites operativos y los retardos que pueden inducir en el sistema en general.

En primer lugar, se realizaron ensayos sobre tres tipos de sensores: un potenciómetro lineal de 10 $k\Omega$, un sensor ultrasónico HC-SR04 y una IMU MPU6050. En cada caso, se analizaron la resolución mínima, la precisión de la medición y el tiempo promedio requerido para adquirir una muestra, modelando el ruido de medición mediante un proceso estocástico gaussiano.

Asimismo, se evaluó el comportamiento del actuador del sistema, un servomotor MG996R encargado de modificar el ángulo de la barra. Se determinó la mínima resolución angular que puede comandarse en la práctica, teniendo en cuenta tanto las especificaciones del fabricante como pruebas empíricas de la *dead band width*. Además, se obtuvo el tiempo que tarda el servo en responder a un cambio de 30° y se analizó el caso en el que se realice el control a una frecuencia de 1 Hz.

Por último, se estudiaron las limitaciones temporales asociadas a la transmisión de datos por puerto serie en Arduino. Se compararon dos métodos de envío de datos tipo `float` y se analizaron sus respectivos tiempos de transmisión.

2. Instrumental utilizado

Los componentes y el equipamiento utilizado a la hora de realizar las mediciones que se efectuaron en el presente trabajo práctico fueron los que se listan a continuación:

- Una fuente de 5 $V - 3$ A,
- Un protoboard y cables para protoboard.
- Una placa Arduino UNO R3.
- Un potenciómetro 10 $k\Omega$ lineal.
- Un Sensor de distancia HC-SR04.
- Una IMU MPU6050 en placa de desarrollo.
- Un Servo MG996R 12 Kg.
- El sistema sugerido por la cátedra, implementado como se puede observar en la Fig. 1.

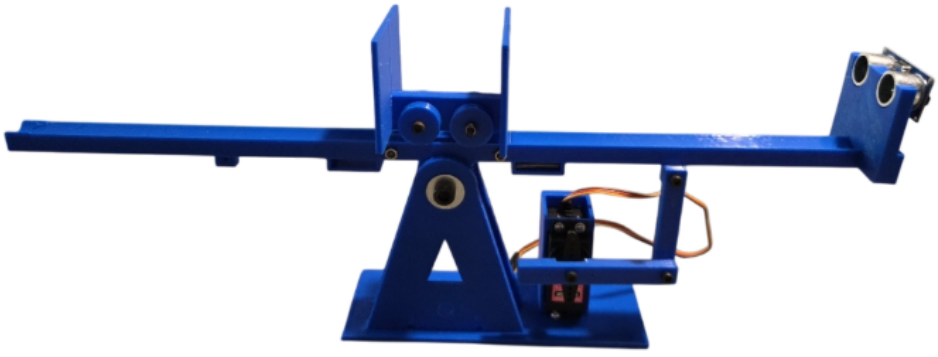

Fig. 1: Sistema implementado.

3. Limitaciones de los sensores

3.1. Potenciómetro

En primer lugar, a modo de obtener la resolución de lectura del potenciómetro, se debe tener en cuenta que esta está dada por la resolución del conversor analógico-digital del Arduino, el cual utiliza 10 bits. Esto permite representar la señal analógica con 1024 niveles.

Además, la resolución en volts depende de la tensión de referencia que utiliza el ADC. Para el caso del Arduino, esta tensión de referencia es de 5 V. Por lo tanto, la mínima tensión distinguible es:

$$V_{min} = \frac{5\ V}{1024} \approx 4{,}88\ mV$$

Luego, con el propósito de representar el ángulo del potenciómetro, se deben mapear los 1024 niveles disponibles del ADC al rango de ángulos que barre el potenciómetro. Dado que este no da una vuelta completa, sino que recorre unos $300° \pm 5°$ según su hoja de datos, la resolución mínima para medir el ángulo del mismo es:

$$\theta_{min} = \frac{300°}{1024} \approx 0{,}29°$$

Por último, a fin de estimar el tiempo requerido para conseguir la medición del ADC, se desarrolla un algoritmo en el *IDE* de Arduino, donde se utiliza la función `micros()` y se promedia el resultado de 500 mediciones. Así, se obtiene un tiempo de medición de $t = 113{,}6\ \mu s$ en promedio.

3.2. Sensor de distancia

A fin de operar el sensor de distancia, se utiliza la librería `NewPing`, que se encarga de enviar una señal de ultrasonido y reportar el tiempo que tarda desde que sale hasta que se recibe un rebote. Para medir los tiempos, utiliza la función `micros()` que en la placa UNO R3 posee una resolución temporal de $\Delta t = 4\ \mu s$.

Como consecuencia de esto, se tiene un límite inferior en la resolución con la que se puede medir el tiempo de ida y vuelta del pulso ultrasónico en el sensor. Para ver cómo esta resolución se traslada a la medición de la distancia, se procede a calcular la resolución espacial. Para ello, se escala en un factor de $\frac{1}{2}$, pues solo interesa el tiempo de ida o vuelta, y se utiliza la velocidad del sonido (tarda 29,287 μs en viajar 1 cm). Así, la resolución espacial quedará dada por:

$$\Delta x = \frac{1\ cm \cdot \Delta t}{2 \cdot 29{,}287\ \mu s} \Longrightarrow \Delta x = \frac{1\ cm \cdot 4\ \mu s}{2 \cdot 29{,}287\ \mu s} \approx 0{,}68\ mm \tag{1}$$

De esta manera, corroborando estos cálculos, se realizan mediciones a partir de dos configuraciones de la planta según dos casos extremos para analizar la correspondencia entre tiempo medido por el sensor y resolución. Por un lado, se mide el caso en el que el carro se encuentra a una distancia simil a la mínima, que según el *datasheet* del sensor, es de 2 cm. Por el otro, se efectúan mediciones cuando el carro está en el extremo de la barra. Ambos casos se puede visualizar en la Fig. 3 y 2, respectivamente.

Fig. 2: Sistema configurado para las mediciones de distancia máxima

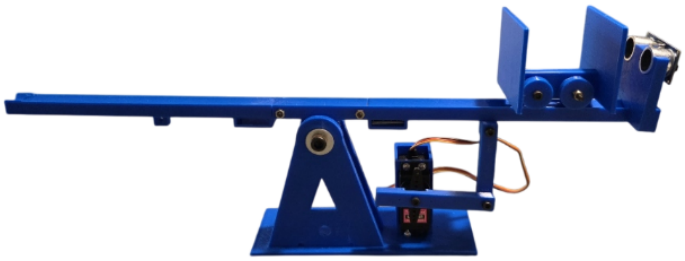

Fig. 3: Sistema configurado para las mediciones de distancia mínima

En base a estas configuraciones, en las Figs. 4 y 5, se pueden apreciar las mediciones obtenidas. En ambas figures se puede ver claramente el carácter digital de las mediciones mencionado anteriormente, ya que, estas no varían de forma continua, sino que cambian de niveles de a saltos de 0,68 mm. A pesar de que estas mediciones se tomaron frente a un objeto fijo que difiere en su distancia, se puede ver claramente que no se obtuvo un único valor constante, sino que ambas mediciones se ven afectada por un ruido.

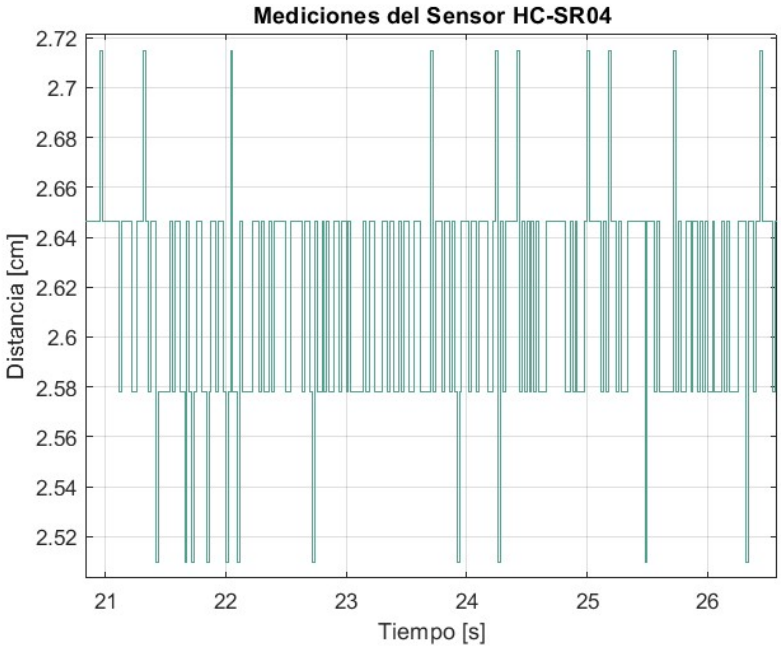

Fig. 4: Mediciones del sensor digital a una distancia mínima

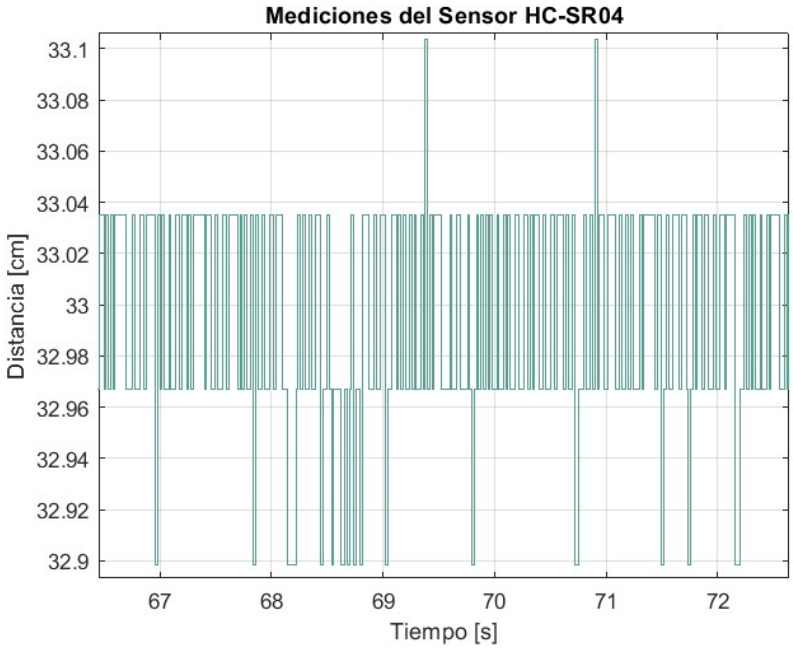

Fig. 5: Mediciones del sensor digital a una distancia máxima

A fin de estimar el impacto de este ruido, se lo modela como gaussiano, verificando el ajuste del modelo mediante los histogramas de las Figs. 6 y 7. Debido a la limitación de resolución dada por la implementación de la librería, el ajuste gaussiano no es el ideal en ninguno de los casos. Sin embargo, será de utilidad para obtener una aproximación de la influencia del ruido en el valor medido.

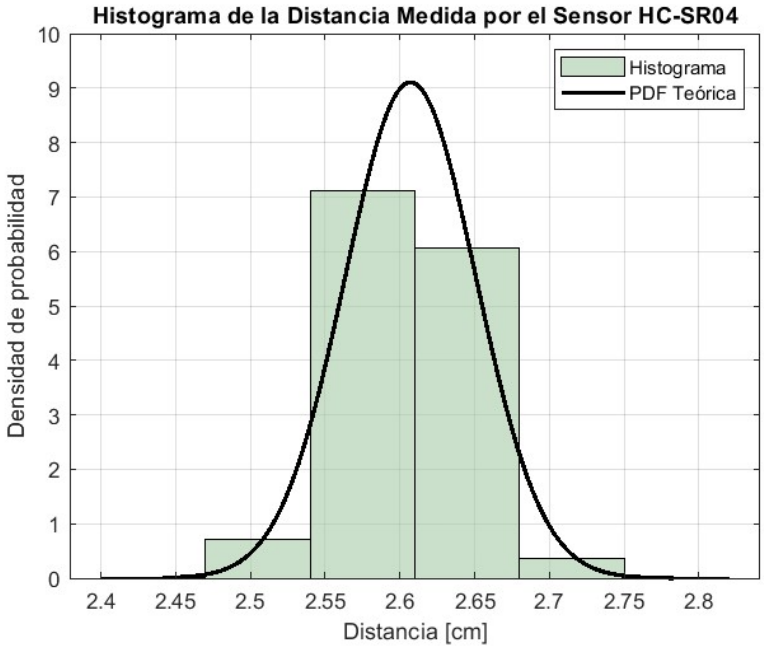

Fig. 6: Histograma de las mediciones del sensor a distancia mínima

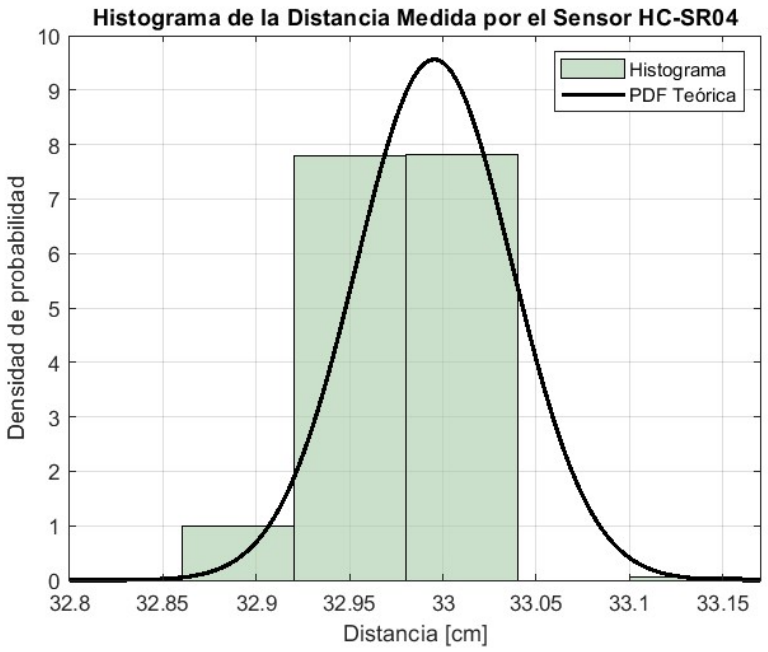

Fig. 7: Histograma de las mediciones del sensor a distancia máxima

Siguiendo esta lógica, mediante el modelado gaussiano del ruido en las mediciones, se calculó el desvío estándar de las mediciones, obteniendo los siguientes valores:

- En el caso de distancia mínima, $\sigma \approx 0{,}44\ mm$.
- En el caso de distancia máxima, $\sigma \approx 0{,}42\ mm$

Considerando que son prácticamente iguales y que en un intervalo de 3σ alrededor de la media se encuentra aproximadamente el 99,9 % de las muestras, se puede interpretar que la máxima variación típica de las mediciones es del orden de 5σ. Por lo tanto, se propone utilizar este valor como una estimación representativa de la mínima resolución efectiva con la que es posible distinguir cambios en la distancia medida. De esta manera, se sugiere que en ambos casos:

$$\Delta x_{min} = 5\sigma \approx 0{,}22\ cm$$

Esta estimación tiene sentido si se lo compara con la resolución dada en la hoja de datos del sensor, la cual es de 0,3 cm. Además, como se puede visualizar en las Figs. 4 y 5, la máxima variación en distancia se corresponde con este valor.

De esta manera, las mediciones de tiempo solo difieren de las mediciones de distancia por una constante dada por la expresión 1 vista anteriormente, por lo que, se obtiene que la resolución temporal es:

$$\Delta t_{min} = \Delta x_{min} \cdot 2 \cdot 29{,}287 \frac{\mu s}{cm} \approx 12{,}9\ \mu s$$

Se puede visualizar que este valor es mayor a la resolución mínima obtenida al principio, por ende, la influencia del ruido es mayor que el efecto que pueden causar las limitaciones en la implementación de la librería.

Por otro lado, se realizó un programa en el *IDE* de Arduino cuyo propósito es promediar el tiempo que se tarda en tomar 500 muestras de la medición de distancia. Para ello, se mide el tiempo que

tarda el sonido en efectuar la medición a través del sensor y se calculan, indirectamente, las distancias a partir de la relación 1 formulada anteriormente.

Debido a que el sonido tarda 29,287 μs en viajar 1 cm, el tiempo que se tarda en tomar las mediciones dependerá pura y exclusivamente de la distancia que se esté midiendo. A fin de realizar un análisis de mejor y peor caso, se toman los dos casos anteriormente mencionados: cuando el carro está en el extremo de la barra como se ve en la Fig. 2 y cuando se encuentra cerca de la distancia mínima que puede medir el sensor, lo cual se puede visualizar en la Fig. 3.

De esta forma, se obtienen los siguientes resultados.

- Distancia mínima de 2,61 cm con un tiempo promedio de medición 2,38 ms.
- Distancia máxima de aproximadamente 33 cm, con un tiempo promedio de medición de 4,3 ms.

Cabe destacar que, al igual que en el inciso anterior, se hizo uso de la función `micros()` para sensar los tiempos.

3.3. IMU

Para estimar la resolución de la IMU se dejó la barra fija en 0° y se guardó la estimación del ángulo obtenida a través del filtro complementario durante 100 segundos, tomando con una frecuencia de 50 Hz, 5000 muestras. A partir de los valores medidos se observa, en la Fig. 8 un valor medio constante de 0° para la medición del ángulo que se ve afectado por un ruido blanco estocástico que se modela como un proceso gaussiano.

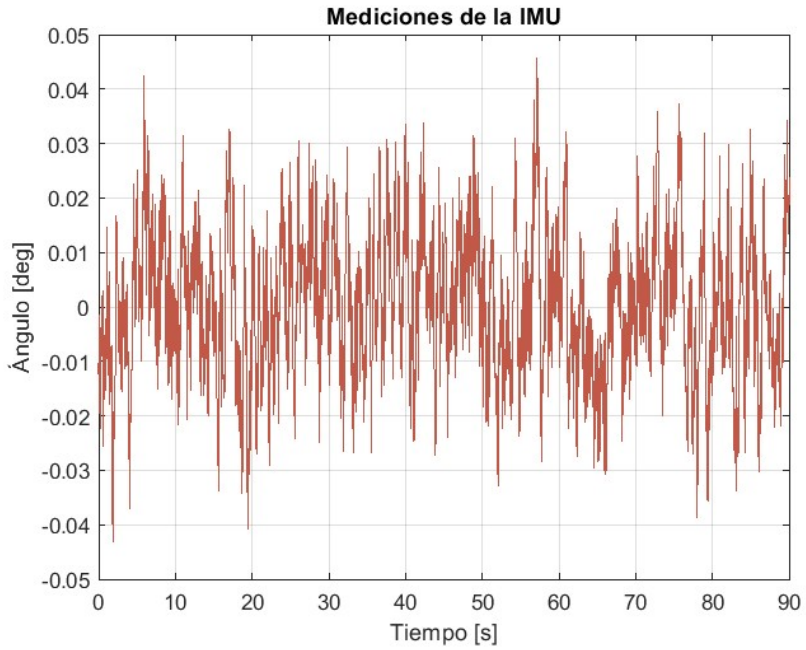

Fig. 8: Mediciones de la IMU

Para verificar que se puede modelar el ruido como gaussiano, se grafica un histograma con las muestras en la Fig. 9. De esta manera, se observa claramente que las mediciones se ajustan correctamente a la distribución teórica propuesta.

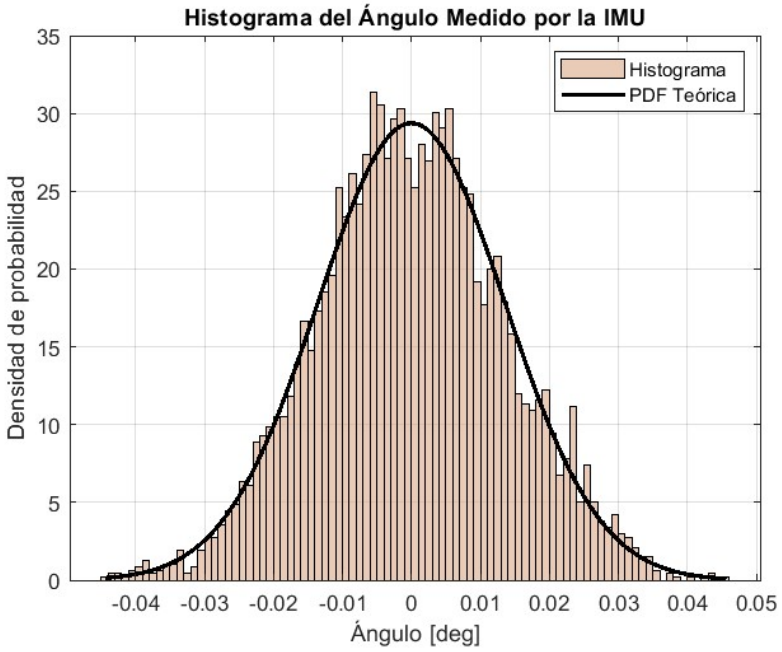

Fig. 9: Histograma de las mediciones de la IMU

Igual que para el caso anterior, se calcula el desvío estándar $\sigma \approx 0{,}014°$ de la distribución y se toma la resolución del sensor como 5 veces este valor. Por lo tanto, la mínima resolución del ángulo medido por la IMU es:

$$\theta_{min} \approx 0{,}07°;$$

que, al igual que en el caso del sensor, es un valor que sigue las características de lo visto en la Fig 8, ya que, es aproximadamente el valor de la variación en ángulos de las mediciones.

Luego, en base al algoritmo ya implementado, se mide el tiempo promedio que se tarda en obtener cada estimación de la IMU, dando así el siguiente resultado:

$$t_{\text{IMU}} \approx 4{,}05\ ms$$

4. Limitaciones de actuadores

4.1. Servomotor

El ángulo del servomotor se controla mediante una señal *PWM* (modulada por ancho de pulso) generada a través de la librería `Servo.h` de Arduino, la cual utiliza el *Timer1* de la placa, que posee una resolución de 16 bits.

La frecuencia de operación de la señal *PWM* es de 50 Hz, lo que implica un período de 20 ms. Dentro de este período, el ancho del pulso varía típicamente entre 1 ms y 2 ms, donde 1 ms representa una posición angular de 0° y 2 ms corresponde a 180°. La función `writeMicroseconds()` permite ajustar el ancho del pulso con una resolución de hasta 1 μs.

No obstante, según el *datasheet* de un servomotor MG996R, existe una *dead band width* de aproximadamente 5 μs. Esto significa que el servomotor no responde a variaciones en el ancho del pulso menores a este valor. Por lo tanto, aunque Arduino pueda generar pulsos con una resolución temporal de 1 μs, la mínima variación efectiva que puede comandar una diferencia en la posición del servo es de 5 μs.

Con esto, se puede calcular la cantidad de posiciones distinguibles que pueden comandarse entre 1 ms y 2 ms con pasos de a 5 μs:

$$n = \frac{2000\mu s - 1000\mu s}{5us} = 200$$

Dado que el rango de movimiento del servo es de 180°, la mínima resolución angular que puede comandarse es:

$$\theta_{min} = \frac{180^\circ}{200} = 0{,}9^\circ$$

Por lo tanto, la resolución angular mínima que puede alcanzarse en la práctica al comandar el servomotor es de $0,9$ por paso de 5 μs.

Al caracterizar la planta, se obtuvo la transferencia entre la señal de control (ángulo que se envía por `Servo.write()`) y el ángulo α de la barra medido con la IMU. De esta caracterización, se obtiene que la relación entre el ángulo de la barra y el de el servo está dada por una constante del siguiente valor:

$$\frac{\alpha}{\theta} = 0{,}363$$

En función de esto y en base a lo desarrollado anteriormente, se puede obtener la mínima resolución con la que se podrá comandar la barra a través del servo:

$$\alpha_{min} = \theta_{min} \cdot 0{,}363 \approx 0{,}33^\circ$$

Siguiendo con esta lógica, se buscó contrastar estos resultados de manera práctica mediante una experiencia en el laboratorio. Para ello, se realizó un barrido de diferencias de tiempo con la función `writeMicroseconds()`, con el objetivo de analizar experimentalmente el valor del *dead band width* del servomotor.

Los resultados obtenidos indicaron que el servomotor no respondía a variaciones menores a 15 μs entre señales PWM consecutivas. Es decir, sólo comenzaba a moverse cuando la diferencia entre comandos superaba dicho valor.

A partir de esta observación, se repitió el cálculo de la resolución angular considerando un *dead band width* de 15 μs:

$$n = \frac{2000\ \mu s - 1000\ \mu s}{15\ us} = \frac{200}{3} \Longrightarrow \theta_{min} = \frac{180^\circ}{\frac{200}{3}} = 2{,}7^\circ \Longrightarrow \alpha_{min} = \theta_{min} \cdot 0{,}363 \approx 0{,}98^\circ$$

Este resultado concuerda con lo observado empíricamente durante la experiencia, ya que, a simple vista el servo y la barra realizaban desplazamientos angulares notorios.

Por otra parte, se quiere obtener el tiempo requerido para mover el servo un diferencial de 30°. Según la hoja de datos, para una alimentación de 4,8 V el servo tarda 170 ms en moverse 60°, lo que daría un tiempo de respuesta teórico de 85 ms para nuestro caso.

Sin embargo, este valor se da para el servo sin carga y supone una respuesta lineal, sin tener en cuenta la aceleración y desaceleración del movimiento. Puesto que en la práctica se utiliza el servo para mover la barra, se envian varios diferenciales consecutivos de 30° al servo y se observa cuánto tarda la medición del ángulo de la IMU en alcanzar el valor final. Esto se puede visualizar en la Fig. 10

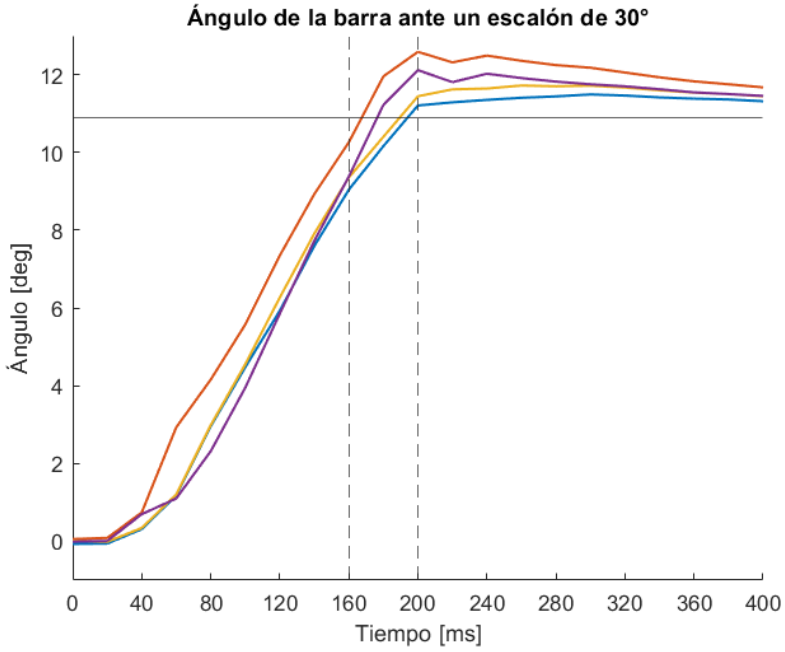

Fig. 10: Respuesta a un diferencial de 30°

Se puede analizar que, en las cuatro curvas mostradas, el tiempo de establecimiento (despreciando el sobrepico que es menor de 2°) se encuentra entre los 160 y 200 ms. Este valor es más del doble que el teórico calculado en base a los datos del datasheet, pero al medir de esta manera se está teniendo en cuenta la carga y el comportamiento real del servo. Además, se está trabajando de una forma más realista al integrar la planta utilizada en este proyecto.

Si se quisiera realizar este mismo control, pero a una frecuencia de 1 Hz, la resolución angular del sistema no se vería afectada. Esto se debe a que esta depende únicamente del ancho del pulso, que varía entre 1 y 2 ms, y no del período total de la señal PWM. Del mismo modo, el tiempo de respuesta ante un diferencial de 30° tampoco se vería modificado, ya que, si el comando se actualiza directamente desde una posición inicial a una final separadas por 30°, el servo ejecutará el mismo movimiento que con una señal de 50 Hz.

Sin embargo, el impacto del cambio de frecuencia se hace evidente cuando se desea actualizar periódicamente la posición del servo. A 50 Hz, se enviaba un nuevo pulso cada 20 ms, permitiendo realizar correcciones rápidas y generar movimientos suaves. Al reducir la frecuencia a 1 Hz, sólo se envía un comando por segundo, lo que introduce un retardo significativo en la capacidad de reacción del sistema, dando lugar a retardos en el control y movimientos más bruscos.

5. Limitaciones de tareas adicionales

Con el objetivo de analizar y medir el tiempo necesario para enviar 100 datos de tipo `float` a través del puerto serie de un Arduino, se desarrolló un programa en su *IDE*. Este algoritmo mide el tiempo que toma completar una iteración de 100 envíos dentro del *loop*, utilizando distintas funciones de transmisión. De esta manera, se evalúan dos funciones: el uso de la función `Serial.print()`, que transmite los datos como texto ASCII legible, y la función `matlabsend()`, implementada en el curso, que encapsula la lógica de `Serial.write()` para enviar datos binarios precedidos por un encabezado que señala el inicio de la transmisión, destinado a su posterior lectura en Matlab.

Los resultados obtenidos son los siguientes:

- Tiempo para enviar 100 `float`s utilizando `Serial.print()`: 28,586 ms, lo que implica un tiempo de aproximadamente 286 μs por `float`.
- Tiempo para enviar 100 `float`s utilizando `matlabsend()`: 62,504 ms, equivalente a unos 625 μs por `float`.

La diferencia de tiempos es evidente, ya que, la función `matlabsend()` no solo envía los datos, sino que también transmite un título y realiza operaciones adicionales, como la asignación de un puntero en memoria, lo que incrementa el tiempo total respecto al simple envío de texto mediante `Serial.print()`.

6. Conclusiones

A modo de conclusión, se muestran los resultados obtenidos en cada uno de los incisos del trabajo práctico. En la primer tabla, se pueden observar los distintos valores de resolución tanto de los sensores, como del actuador. Mientras que, en la segunda, se ven los tiempos calculados a la hora de realizar ciertas instrucciones mediante la placa Arduino UNO R3.

Componente	Resolución	Valor
Potenciómetro	V_{min}	4,88 mV
	θ_{min}	0,29°
HC-SR04	Δx_{min}	0,22 cm
	Δt_{min}	12,9 μs
IMU MPU6050	θ_{min}	0,07°
Servo MG996R	θ_{min}	2,70°
	α_{min}	0,98°

Tabla 1: Resultados de las resoluciones de los instrumentos utilizados

Componente	Acción	Tiempo
Potenciómetro	Realizar una medición con el ADC.	113,6 μs
HC-SR04	Realizar medición de distancia mínima (2,61 cm).	2,38 ms
	Realizar medición de distancia máxima (33 cm).	4,30 ms
IMU MPU6050	Realizar medición del ángulo.	4,05 ms
Servo MG996R	Mover el servo un diferencial de 30°.	160 a 200 ms
Puerto Serie	Enviar 100 `float`s usando `Serial.print()`.	268 μs por dato
	Enviar 100 `float`s usando `matlabsend()`.	625 μs por dato

Tabla 2: Resultados de los tiempos en realizar cada acción

A lo largo del informe, se observó que, si bien algunos componentes tienen una buena resolución teórica, su desempeño práctico puede verse condicionado por factores como el ruido o las restricciones

de las librerías utilizadas. Por ejemplo, el sensor de ultrasonido mostró una resolución limitada por la forma en que implementa la medición de tiempos y el ruido generado por las mediciones.

Para la IMU, por su parte, se pudo estimar el impacto que tiene el ruido sobre las mediciones obtenidas de una forma más precisa que la del sensor digital, puesto que toma valores continuos y el ajuste gaussiano lo modela mejor. Dando así, una resolución más adecuada.

En el caso del servomotor, el análisis práctico de su comportamiento evidenció una *dead band* más amplia de la especificada en el *datasheet*, así como un tiempo de respuesta significativamente mayor cuando se encuentra en el armado de la planta. Estas diferencias entre las especificaciones teóricas y el comportamiento real, muestran la necesidad de caracterizar mediante mediciones los componentes del sistema.

Además, el estudio de la transmisión de datos a través del puerto Serial, resaltó el impacto temporal que puede tener el envío de una cantidad dada de tipos de dato `float`s y su comparación con referenciar un puntero y enviar cuatro *bytes* más por iteración, resultando en el doble de tiempo.

En resumen, el trabajo fue una buena oportunidad para poner en práctica conceptos vistos en las clases de la materia y entender cómo las limitaciones físicas de sensores y actuadores impactan en el desempeño de la planta. Además, el uso de herramientas como Arduino y Matlab permitieron realizar las mediciones y analizar los resultados obtenidos.