Universidad Nacional de San Luis

Facultad de Ingeniería y Ciencias Agropecuarias

IMPLEMENTACIÓN DE UN CONTROLADOR NO LINEAL PARA UN LEVITADOR MAGNÉTICO

Gelatti, Lucas Daniel

Trabajo final de Ingeniería Electrónica

Director

Dr. Ing. Asensio, Maximiliano

Co Director

Ing. Peñaloza, Juan Pablo

Villa Mercedes, San Luis.

2024

DERECHO DE AUTOR

Dedicatoria

Quiero dedicarle este trabajo final a mis Padres, que su apoyo moral y económico fue lo que me impulsó a finalizar mis estudios. También a mis Abuelos que desde chiquito me influenciaron para que estudie ingeniería.

Resumen

En este trabajo se llevó a cabo la implementación y diseño de un controlador no lineal digital para un levitador magnético utilizando una técnica de linealización por realimentación de estados. Primero se estudiaron algunas de las aplicaciones de la levitación magnética y los levitadores, luego se realizó el modelado matemático del sistema y el diseño del controlador. Este controlador se validó realizando simulaciones en un software especializado. Luego se describe la manera en que se implementó en el prototipo, pasando de un modelo teórico abstracto a un prototipo en funcionamiento. Además, se diseñó e implementó una interfaz gráfica en Python para medir el desempeño del controlador y poder interactuar con el prototipo. Por último, se utilizó esta interfaz para realizar los ensayos correspondientes y verificar de forma cuantitativa que el desempeño del prototipo sea acorde a las simulaciones realizadas.

Palabras clave: Control no lineal, Levitador magnético, Python, STM32.

INDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPITULO 1: Propuesta

1. Introducción

La levitación magnética se basa en hacer uso de campos electromagnéticos para lograr la suspensión de un objeto en contra de la fuerza de gravedad. La particular ventaja de este método es que no requiere de un contacto físico para actuar sobre el objeto levitado lo que elimina por completo la fricción mecánica entre las partes del sistema.

Dadas sus ventajas, los sistemas de levitación magnética se presentan en diversas aplicaciones, que comprenden, desde trenes eléctricos de alta velocidad [1], túneles de viento, amortiguadores, frenos, actuadores, rodamientos sin contacto, biomedicina, entre otros.

Un ejemplo de aplicación en transporte de alta velocidad es el tren MagLev de Shanghái, que alcanza velocidades de hasta 430km/h [2]. Este sistema de transporte consiste en hacer levitar los vagones del tren sobre una vía diseñada especialmente. Lo que elimina el contacto entre las partes móviles, y, en consecuencia, el desgaste mecánico, reduciendo así los costes de mantenimiento [3].

En el área de la biomedicina se utilizan imanes permanentes para lograr separar las partículas en una solución de acorde con su densidad. Algunas aplicaciones que se le da a esta tecnología en este campo son en medicina forense, análisis de sangre y diagnóstico de enfermedades, identificación de sustancias, entre otros [4].

Un levitador magnético es un dispositivo eléctrico y electrónico que tiene como finalidad hacer levitar una pieza de material ferromagnético u otra bobina mediante la aplicación de un campo magnético.

El prototipo de levitador sobre el que se lleva a cabo este trabajo consiste de un electroimán que hace levitar una esfera imantada, o de un material ferromagnético, por efecto de atracción magnética, lo que constituye un sistema no lineal de tercer orden inestable. Para lograr la levitación del objeto es necesario hacer uso de un controlador automático.

Existen distintos esquemas de control lineal que logran la estabilidad y el desempeño dinámico requerido para un punto de operación [9], sin embargo, para ampliar el rango de trabajo del prototipo es necesario optar por técnicas de control más avanzadas. En este trabajo se lleva a cabo el diseño e implementación de un controlador no lineal, utilizando la técnica de linealización por realimentación de estados [10]. Además, se desarrolla una interfaz gráfica de usuario para poder visualizar en tiempo real las variables del sistema en una PC.

2. Objetivos

2.1. Objetivo general

El objetivo general de este trabajo es la implementación de un controlador automático no lineal que logre la estabilidad y tenga el desempeño dinámico requerido para el prototipo de levitador magnético disponible en el Laboratorio de Control Automático.

2.2. Objetivos específicos

A continuación, se listan los objetivos específicos abordados en este trabajo.

- Adquirir conocimientos sobre esquemas de control no lineales.
- Diseñar e implementar un controlador no lineal.
- Adquirir conocimientos sobre programación de microcontroladores.
- Adquirir conocimientos desarrollo de interfaces gráficas de usuario (GUI).
- Diseñar e implementar una interfaz gráfica de usuario para una buena presentación de datos.
- Realizar la comparación entre los resultados de simulación y el sistema real.
- Obtener una experiencia de trabajo profesional integrando un grupo de investigación y desarrollo.

3. Descripción del prototipo

El prototipo de levitador en el que se lleva a cabo este trabajo es el mostrado en la Figura Nº 1. Este modelo, que pertenece al Laboratorio de Control Automático, se trata de un levitador magnético por atracción. Cuenta con un electroimán que hace levitar un imán permanente esférico. Este electroimán es comandado por un convertidor CC-CC de topología puente completo, que permite controlar la cantidad de tensión aplicada en función del ancho de pulso de las señales de disparo de las llaves semiconductoras.

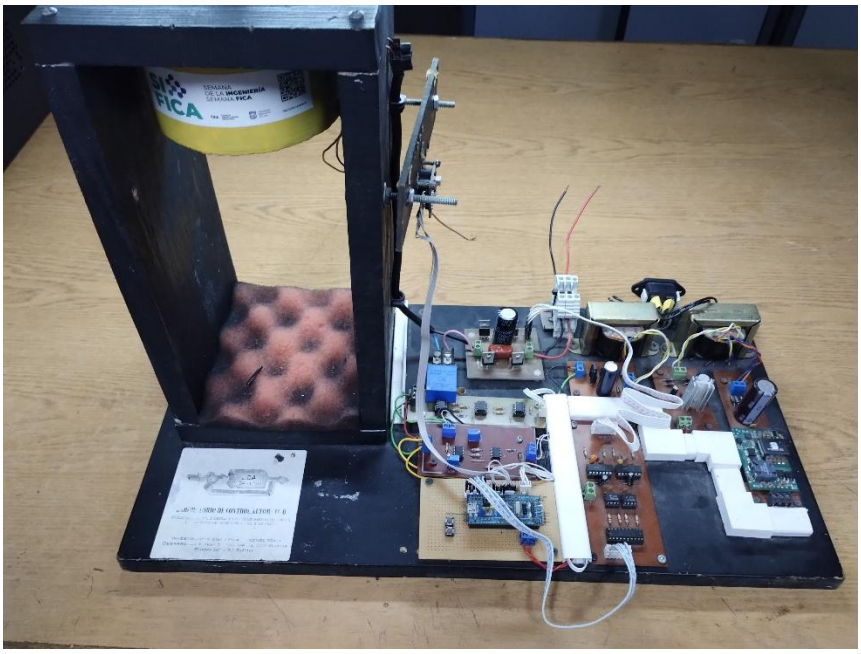

Figura Nº 1. Prototipo de levitador magnético

Además, cuenta con un sensor de posición que permite conocer la distancia relativa entre el objeto levitado y el inductor, un sensor de corriente, que mide esta magnitud en la bobina, y un microcontrolador programable, modelo STM32, que permite leer las señales de los sensores, procesarlas y emitir las señales PWM necesarias para comandar la etapa de potencia.

En la Figura Nº 2 se muestra el diagrama en bloques del prototipo de levitador, en donde se aprecian cada una de las partes que lo componen y cómo se comunican. La fuente de corriente continua proporciona una tensión fija al convertidor CC-CC, el cual alimenta al electroimán. Este convertidor permite obtener a su salida una tensión variable controlada por las señales de disparo de las llaves semiconductoras, estas son generadas por el microcontrolador y acondicionadas a los niveles requeridos por la etapa de potencia mediante un circuito adaptador de señales. Además, el prototipo cuenta con dos sensores, uno que mide la posición de la esfera, y el otro mide la corriente de la bobina. Cada uno de ellos cuenta con su respectivo circuito de adaptación para lograr señales compatibles con el microcontrolador, el cual se encarga de leer estas entradas, procesarlas y producir las señales de salida hacia la etapa de potencia.

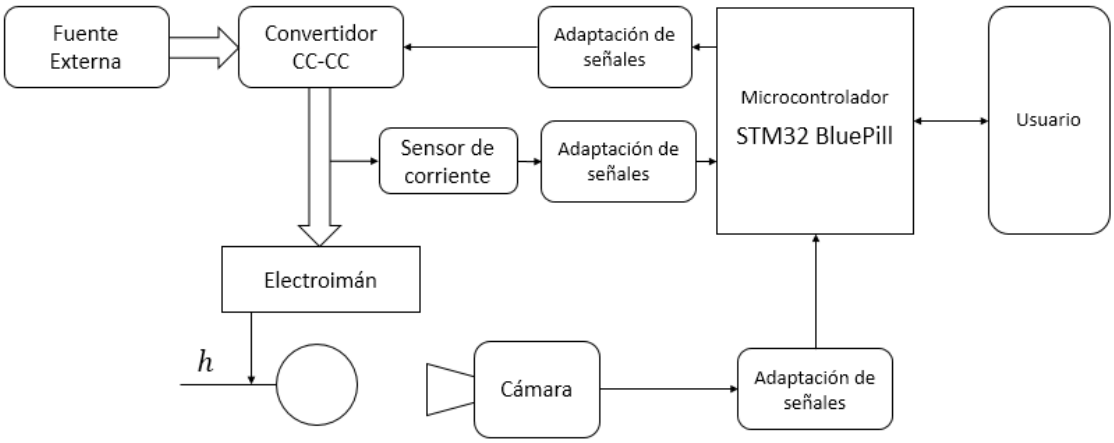

Figura Nº 2. Diagrama en bloques

CAPITULO 2: Diseño del controlador

En este capítulo se desarrolla el modelado matemático del levitador, y se diseña un controlador con la técnica de linealización por realimentación de estados. Las ganancias del controlador son obtenidas mediante el método de asignación de polos.

1. Modelado matemático

Para llevar a cabo el modelado matemático del levitador se analizó el esquema mostrado en la Figura Nº 3.

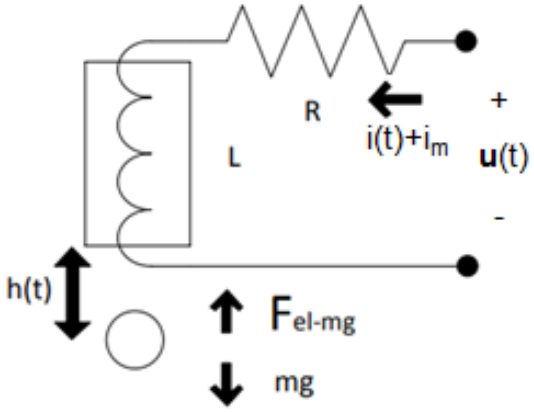

Figura Nº 3. Esquema del levitador

En donde $h(t)$ es la posición de la esfera, mg es el peso de la misma, F_{el-mg} es la fuerza electromagnética ejercida por el campo del electroimán sobre el objeto levitado, L es la inductancia de la bobina, R es su resistencia, $i(t)$ es la corriente que circula por el inductor, i_m es una corriente auxiliar "ficticia" que se utilizó para modelar el comportamiento producido por el imán permanente, finalmente, $u(t)$ es la tensión aplicada al sistema.

Este sistema se dividió en dos partes, una parte eléctrica, conformada por el circuito RL, y otra física, que consiste de la interacción del campo magnético con la esfera.

La Ecuación Nº 1 define el comportamiento dinámico entre la tensión de entrada u del circuito RL y la corriente $i(t)$ en la bobina.

$$\frac{di}{dt} = -\frac{R}{L}i + \frac{u}{L} \qquad (1),$$

donde R es la resistencia del bobinado, L es la inductancia total de la bobina, i es la corriente y u es la tensión aplicada.

El análisis del esquema físico se llevó a cabo teniendo en cuenta que la esfera es imantada, por lo tanto, existe una fuerza del electroimán sobre la esfera incluso en ausencia de corriente. Este fenómeno se modeló como una corriente adicional i_m, que circula de forma imaginaria sobre el electroimán, la cual produciría una fuerza equivalente en la esfera si esta no fuera imantada. Por último, la fuerza que ejerce la bobina sobre la misma es la derivada de la energía almacenada en el sistema, definida en la Ecuación Nº 2, respecto de la distancia.

$$W(h) = \frac{1}{2}(L_0 + L_h)(i + i_m)^2 \qquad (2).$$

Donde W es la energía del sistema, L_0 es la inductancia de la bobina con la esfera posicionada a una distancia infinita, L_h es el aporte de inductancia de la esfera cuando esta se encuentra en la proximidad de la bobina, i es la corriente del electroimán, y por último, i_m es la corriente equivalente del imán.

La Ecuación Nº 3 define a la inductancia L_h como el cociente entre una constante k y la posición h.

$$L_h = \frac{k}{h} \qquad (3),$$

donde k depende de los aspectos constructivos y los materiales que conforman tanto a la bobina como la esfera.

Por lo tanto, remplazando la Ecuación Nº 3 en la Nº 2 y realizando la derivada de la energía del sistema respecto a h, se tiene que

$$F_{el-mg} = \frac{dW}{dh} = \frac{k}{2}\frac{(i + i_m)^2}{h^2} \qquad (4).$$

Aplicando la segunda ley de Newton se obtiene la Ecuación Nº 5, y reemplazando 4 en 5 la aceleración de la esfera queda definida por la Ecuación Nº 6.

$$m\frac{d^2h}{dt} = mg - F_{el-mg} \qquad (5),$$

$$\frac{d^2h}{dt} = g - \frac{k}{2m}\frac{(i + i_m)^2}{h^2} \qquad (6).$$

Con la ecuación 1 y la 6 se expresó al sistema en sus variables de estado, como sigue.

$$\begin{cases} \dot{h} = \dfrac{dh}{dt} \\ \ddot{h} = g - \dfrac{k}{2m}\dfrac{(i+i_m)^2}{h^2} \\ \dot{i} = -\dfrac{R}{L}i + \dfrac{u}{L} \end{cases} \quad (7).$$

Este sistema de ecuaciones constituye un sistema no lineal, invariante en el tiempo, de tercer orden. En el cual, el origen de las no linealidades son los términos elevados al cuadrado.

2. Estrategia de control.

La estrategia de control propuesta para este sistema se muestra en la Figura Nº 4.

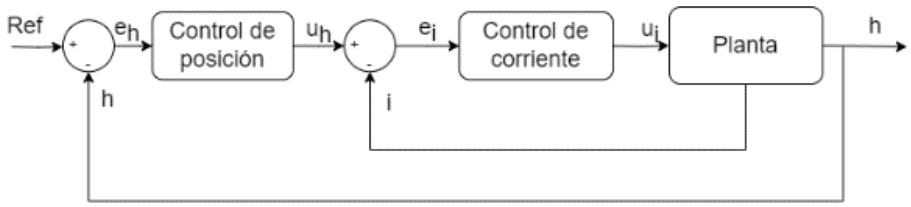

Figura Nº 4. Diagrama del controlador

La variable que se controla en el prototipo es la posición de la esfera (h), es decir, la salida de la planta. Esto se realiza mediante dos controladores automáticos anidados, uno de posición y uno interno de corriente. La entrada del sistema es una señal de referencia (Ref), o sea, la posición deseada de la esfera, a la cual se le resta la posición medida obteniendo una señal de error (e_h) que se utiliza como entrada al controlador de posición. La salida de este es una señal de referencia de corriente (u_h), a la que se le substrae el valor de la corriente medida en el electroimán y se obtiene la señal de error (e_i), la cual se utiliza como entrada al controlador de corriente. Este último, arroja a su salida una tensión variable (u_i) que es la que se aplica en el electroimán.

Debido a que el sistema de la Ecuación Nº 7 no es lineal, para poder aplicar una estrategia de control por realimentación de estados es necesario linealizar el sistema, ya que la técnica de asignación de polos solo funciona para sistemas lineales invariantes en el tiempo [4].

2.1. Linealización por realimentación de estados.

Uno de los métodos más populares para linealizar el sistema es obtener una aproximación lineal mediante expansión en series de Taylor [9] de la ecuación que contiene el termino no lineal, para un punto de operación seleccionado. La principal desventaja de este

método es que el error en la aproximación entre los modelos es pequeño solo para un intervalo relativamente pequeño alrededor del punto de operación. Fuera de este, el error se hace grande y el controlador tendrá un desempeño pobre o incluso podría dejar de funcionar.

Debido a las desventajas del enfoque tradicional, la técnica seleccionada para linealizar el sistema es la de linealización por realimentación de estados. Este método se basa en cancelar las no linealidades mediante una acción de control que depende de las variables de estado medidas.

La principal ventaja de los controladores basados en esta técnica es que se obtiene un sistema lineal para todo el rango de operación de la planta, además, las ganancias del controlador no dependen de los parámetros del sistema. No obstante, la ley de control linealizante es muy dependiente de estos, por lo que una variación en los mismos produce que el sistema se comporte como uno no lineal perturbado [10].

3. Diseño del lazo de posición

En esta sección se describe el procedimiento que se siguió para el diseño del controlador de posición. Primero se obtuvo una acción de control que linealiza el sistema, luego se discretizó el sistema lineal obtenido y por último se calcularon las ganancias que ubican los polos en el lugar deseado.

3.1. Linealización

Para aplicar esta estrategia se propuso como salida la posición h, luego se derivó hasta que se obtuvo la entrada en la derivada, es decir, la corriente.

$$\begin{aligned} y &= h(t) \\ \frac{dy}{dt} &= \frac{dh}{dt} \\ \frac{d^2y}{dt} &= g - \frac{k}{2m}\frac{u_h}{h^2} \end{aligned} \tag{8}$$

Donde

$$u_h = (i + i_m)^2 \tag{9}$$

Se buscó una acción de control linealizante $u_h(\omega_h)$ tal que cumpla con la condición de la Ecuación Nº 10.

$$\frac{d^2y}{dt} = \omega_h \rightarrow \omega_h = g - \frac{k}{2m}\frac{u_h}{h^2} \quad (10),$$

$$u_h = 2(g - \omega_h)\frac{m}{k}h^2 \quad (11),$$

donde ω_h es la nueva entrada al sistema linealizado definido en la Ecuación Nº 12.

$$\begin{bmatrix}\dot{h}\\ \ddot{h}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}h\\ \dot{h}\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}\omega_h \quad (12).$$

3.2. Ubicación de polos

Para obtener las ganancias de realimentación, se comenzó por discretizar el sistema a la frecuencia a la que se ejecutará el controlador en el microcontrolador del prototipo real. Esta, fue seleccionada en función a la máxima tasa de muestreo del sensor de posición, es decir, 10kHz. Por lo tanto, el periodo de muestreo T es de $100\mu s$.

3.2.1. Discretización

La discretización del sistema lineal de la Ecuación Nº 12 se llevó a cabo con el método Forward Euler.

$$\begin{aligned}\frac{h[k+1] - h[k]}{T} &= dh[k]\\ \frac{dh[k+1] - dh[k]}{T} &= \omega_h[k]\end{aligned} \quad (13),$$

$$\begin{bmatrix}h[k+1]\\ dh[k+1]\end{bmatrix} = \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix}\begin{bmatrix}h[k]\\ dh[k]\end{bmatrix} + \begin{bmatrix}0\\ T\end{bmatrix}\omega_h \quad (14).$$

Se expresó la Ecuación Nº 14 en forma matricial por razones de conveniencia.

$$\begin{aligned}\boldsymbol{x}[k+1] &= G_h\boldsymbol{x}[k] + H_h\omega_h[k]\\ y[k] &= C_h\boldsymbol{x}[k]\end{aligned} \quad (15),$$

donde:

- $\boldsymbol{x}[k] = \begin{bmatrix}h[k]\\ dh[k]\end{bmatrix}$ es el vector de estados.
- $G_h = \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix}$ es la matriz de estados.

- $H_h = \begin{bmatrix} 0 \\ T \end{bmatrix}$ es la matriz de entrada.
- $C_h = [1 \quad 0]$ es la matriz de salida.

3.2.2. Eliminación del error en estado estable

Para eliminar el error en estado estable del sistema, se decidió implementar un sistema de seguimiento con un integrador. Para ello, se modeló el sistema ampliado en función del error en estado estable definido por la Ecuación Nº 16

$$\boldsymbol{\zeta}_e[k+1] = \left(\begin{bmatrix} G_h & H_h \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{K} \right) \boldsymbol{\zeta}_e[k] \qquad (16),$$

donde:

- $\boldsymbol{\zeta}_e[k] = \begin{bmatrix} \boldsymbol{x}[k] - \boldsymbol{x}[\infty] \\ \boldsymbol{u}[k] - \boldsymbol{u}[\infty] \end{bmatrix}$ es el vector de error.
- $\boldsymbol{K} = [\boldsymbol{K_2} - \boldsymbol{K_2} G_h - \boldsymbol{K_1} C_h G_h \vdots \boldsymbol{I}_m - \boldsymbol{K_2} H_h - \boldsymbol{K_1} H_h G_h]$
- $\boldsymbol{K_1}$**:** Es la ganancia del error integrado.
- $\boldsymbol{K_2}$: Es la ganancia de las variables de estado del sistema linealizado.

3.2.3. Obtención del vector de ganancias

Para la obtención de las ganancias del controlador se decidió calcular los polos de un sistema tipo de segundo orden y luego realizar la discretización a la frecuencia de muestreo de trabajo del microcontrolador, utilizando la relación entre el plano s y el plano z [7]. Este método arroja resultados correctos siempre y cuando las constantes de tiempo más significativas del sistema sean al menos diez veces más grandes que el periodo de muestreo [8].

La ecuación característica de un sistema de segundo orden es definida por la Ecuación Nº 17.

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \qquad (17),$$

donde ζ es el factor de amortiguamiento y ω_n es la frecuencia natural no amortiguada. Estos parámetros determinan la estabilidad relativa del sistema, el sobrepaso, el tiempo de levantamiento y el de asentamiento entre otros.

El tiempo de asentamiento para una tolerancia del 2% se calcula con

$$t_s = \frac{4}{\zeta \omega_n} \quad (18)$$

por lo que la frecuencia natural no amortiguada es

$$\omega_n = \frac{4}{\zeta t_s} \quad (19),$$

Los parámetros de diseño se seleccionaron en el dominio del tiempo, con un tiempo de asentamiento $t_s = 0.5s$ y un coeficiente de amortiguamiento $\zeta = 0.95$. La ubicación del tercer polo del sistema se seleccionó de manera tal que no afecte de manera desmedida la dinámica del sistema. Para ello, se multiplico por 30 el polo más negativo.

En la Ecuación Nº 20, se muestran los polos obtenidos para tiempo continuo, p, y en la Nº 21 el resultado discretizado, p_d.

$$p = [-8 \pm j2.63 \quad -240] \quad (20),$$

$$p_d = e^{Tp} = [0.9992 \pm j0.0003 \quad 0.9763] \quad (21).$$

Para calcular el vector de ganancias del controlador, primero se obtuvo la ecuación característica del sistema luego de aplicar la transformada Z a la Ecuación Nº 16.

$$z\boldsymbol{\zeta}_e = \left(\begin{bmatrix} G_h & H_h \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \boldsymbol{K}\right) \boldsymbol{\zeta}_e \quad (22),$$

$$\left(z\boldsymbol{I} - \begin{bmatrix} G_h & H_h \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \boldsymbol{K}\right) \boldsymbol{\zeta}_e = 0 \quad (23).$$

La ecuación característica está dada por el determinante de la matriz de la Ecuación Nº 23.

$$\det\left(z\boldsymbol{I} - \begin{bmatrix} G_h & H_h \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \boldsymbol{K}\right) = 0 \quad (24),$$

$$\det\begin{pmatrix} z-1 & -T & 0 \\ 0 & z-1 & -T \\ k_1 & k_2 & z+k_3 \end{pmatrix} = 0 \quad (25),$$

$$(z-1)^2(z+k_3) - (z-1)Tk_2 + T^2k_1 = 0 \quad (26),$$

$$z^3 + (k_3 - 2)z^2 + (1 - 2k_3 + Tk_2)z - Tk_2 + T^2k_1 - k_3 = 0 \quad (27).$$

Por lo tanto, igualando la Ecuación Nº 27 con la ecuación característica de los polos deseados se obtiene el siguiente sistema de ecuaciones lineales.

$$\begin{cases} k_3 - 2 = -(p_1 + p_2 + p_3) \\ 1 - 2k_3 + Tk_2 = -(p_1p_2 + p_1p_3 + p_2p_3) \\ Tk_2 - T^2k_1 + k_3 = -p_1p_2p_3 \end{cases} \quad (28).$$

Resolviendo el sistema de la Ecuación Nº 28 se obtuvieron los siguientes resultados de ganancias.

$$\boldsymbol{K} = \begin{cases} k_1 = 1.6803 \\ k_2 = 0.3864 \\ k_3 = -0.9747 \end{cases} \quad (29),$$

por último, se obtuvieron los valores de ganancias K_1 y $\boldsymbol{K_2}$ que serán los aplicados al controlador.

$$\boldsymbol{K_h} = [\boldsymbol{K_2}, K_1] = (\boldsymbol{K} + [0,0,1]) \begin{bmatrix} G - I & H \\ CG & CH \end{bmatrix}^{-1} \quad (30),$$

$$\begin{aligned} K_1 &= 1.6803 \\ \boldsymbol{K_2} &= [3861.2699 \quad 252.9440] \end{aligned} \quad (31),$$

$$\boldsymbol{K_h} = [3861.2699 \quad 252.9440 \quad 1.6803] \quad (32).$$

Finalmente, se escribe la ecuación de la acción de control completa, que se utiliza como entrada al controlador de corriente.

$$u_h = \left(g - \boldsymbol{K_h} \begin{bmatrix} e_h \\ -dh \\ \sum_0^n e_h \end{bmatrix} \right) 2\frac{m}{k}h^2 \quad (33),$$

con

$$e_h = Ref - h \quad (34),$$

donde Ref es la referencia de posición.

4. Diseño del lazo de corriente

En esta sección se describe el procedimiento que se siguió para el diseño del controlador de corriente. Primero se obtuvo una acción de control que elimina las no linealidades del sistema, luego se discretizó el sistema lineal equivalente obtenido y, por último, se calcularon las ganancias que ubican los polos en el lugar deseado.

4.1. Linealización

Recordando que anteriormente se decidió dividir al sistema en dos subsistemas, uno comprendido por la interacción de fuerzas entre el electroimán y la esfera, y el otro constituido por el circuito RL, y que la entrada del primero es la salida del segundo. Debido a que esta entrada/salida, definida en la Ecuación Nº 9, es un término no lineal, es necesario realizar la linealización de esta con respecto a la entrada (del circuito RL) para obtener un sistema lineal equivalente. Este se obtuvo mediante la misma técnica de linealización por realimentación de estados utilizada con anterioridad.

Se comenzó por definir a la salida como sigue.

$$y = (i + i_m)^2 \qquad (35),$$

luego se obtuvieron las derivadas de la salida respecto al tiempo hasta que aparece el término de la entrada.

$$\frac{dy}{dt} = 2(i + i_m)\frac{di}{dt} \qquad (36).$$

Reemplazando la Ecuación Nº 1 en la Nº 36 se tiene que

$$\frac{dy}{dt} = 2(i + i_m)\left(-\frac{R}{L}i + \frac{u}{L}\right) \qquad (37).$$

Se buscó una acción de control linealizante $u(\omega_i)$ tal que cumpla con la condición de la Ecuación Nº 38.

$$\frac{dy}{dt} = \omega_i \qquad (38),$$

por lo que se igualaron la Ecuación Nº 37 y Nº 38, y se despejo el valor de u.

$$\omega_i = 2(i + i_m)\left(-\frac{R}{L}i + \frac{u}{L}\right) \qquad (39),$$

$$u = \frac{\omega_i}{2(i + i_m)} L + Ri \quad (40).$$

Donde u es la acción de control que linealiza el sistema y ω_i es la entrada del nuevo sistema lineal mostrado en la Ecuación Nº 41.

$$\frac{d((i + i_m)^2)}{dt} = \omega_i \quad (41).$$

4.2. Ubicación de polos

Para obtener las ganancias de realimentación, se comenzó por discretizar el sistema a la frecuencia a la que se ejecutará el controlador en el microcontrolador del prototipo real, es decir, 10kHz. Por lo tanto, el periodo de muestreo T es de $100\mu s$, el mismo que el del lazo de posición.

4.2.1. Discretización

Se comenzó con la discretización del sistema lineal de la Ecuación Nº 41 con el método Forward Euler.

$$\frac{x[k+1] - x[k]}{T} = \omega_i[k] \quad (42),$$

$$x[k+1] = x[k] + T\omega_i[k] \quad (43),$$

donde:

- $x[k] = (i[k] + i_m)^2$ es la variable de estado del sistema.
- $T = \frac{1}{f_s} = 100\mu s$ es el periodo de muestreo.

Se expresó en forma matricial por razones de conveniencia.

$$\begin{aligned} \boldsymbol{x}[k+1] &= G_i \boldsymbol{x}[k] + H_i \omega_i[k] \\ y[k] &= C_i \boldsymbol{x}[k] \end{aligned} \quad (44),$$

donde:

- $G_i = [1]$ es la matriz de estados.
- $H_i = [T]$ es la matriz de entrada.
- $C_i = [1]$ es la matriz de salida.

4.2.2. Eliminación del error en estado estable

Para eliminar el error en estado estable del sistema, se decidió implementar un sistema de seguimiento con un integrador. Entonces se modeló el sistema ampliado en función del error en estado estable definido en la Ecuación Nº 45.

$$\boldsymbol{\zeta}_e[k+1] = \left(\begin{bmatrix} G_i & H_i \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{K}\right) \boldsymbol{\zeta}_e[k] \quad (45),$$

donde:

- $\boldsymbol{\zeta_e}[k] = \begin{bmatrix} \boldsymbol{x}[k] - \boldsymbol{x}[\infty] \\ \boldsymbol{u}[k] - \boldsymbol{u}[\infty] \end{bmatrix}$ es el vector de error.
- $\boldsymbol{K} = [\boldsymbol{K_2} - \boldsymbol{K_2} G_i - \boldsymbol{K_1} C_i G_i \vdots \boldsymbol{I}_m - \boldsymbol{K_2} H_i - \boldsymbol{K_1} H_i G_i]$ es la ganancia del sistema ampliado.
- $\boldsymbol{K_1}$**:** Es la ganancia del error integrado.
- $\boldsymbol{K_2}$: Es la ganancia de las variables de estado del sistema linealizado.

4.2.3. Obtención del vector de ganancias

Los polos del sistema se calcularon para tiempo continuo, del mismo modo que en la sección 3.2.3, a partir de la ecuación característica de un sistema de segundo orden definida en la Ecuación Nº 17 y luego se discretizaron a la frecuencia de muestreo seleccionada.

Los parámetros de diseño se seleccionaron en el dominio del tiempo, con un tiempo de asentamiento $t_s = 0.05s$ y un coeficiente de amortiguamiento $c_a = 0.95$. Estos parámetros dinámicos se seleccionaron para que la corriente se estabilice 10 veces más rápido que la posición, con el objetivo de poder desacoplar los lazos de realimentación.

$$\omega_n = \frac{4}{t_s c_a} = 84.211 \quad (46)$$

$$\zeta = c_a = 0.95 \quad (47)$$

$$p_{1,2} = -80.00 \pm j26.295 \quad (48)$$

$$p_{d1,d2} = e^{Tp_{1,2}} = 0.99202 \pm j0.0026 \quad (49)$$

Una vez obtenidos los polos en el plano z, se procedió a obtener las ganancias K_1 y $\boldsymbol{K_2}$ con el método de asignación de polos.

$$\begin{matrix} K_1 = 0.7838 \\ K_2 = 158.7268 \end{matrix} \quad (50)$$

$$\boldsymbol{K_i} = [K_1 \;\; K_2] \tag{51}$$

Por último, en la Ecuación Nº 52, se muestra la acción de control completa, que se utiliza para actuar sobre la planta.

$$u = \left(\boldsymbol{K_i} \left[\sum_{0}^{n} e_i \right] \right) \frac{L}{2(i + i_m)} + Ri \tag{52}$$

Con

$$e_i = u_h - (i + i_m)^2 \tag{53}$$

5. Diseño de un observador de estados

Para poder aplicar el control por realimentación de estados se deben conocer todas las variables de estado del sistema. Estas variables son la posición, la velocidad y la corriente. Para la primera y la última, la planta cuenta con sensores que permiten obtener una medición de sus valores, sin embargo, no se cuenta con información sobre la velocidad. Para obtener el valor de esta variable de manera sencilla, es posible aproximar la derivada de la señal de posición mediante algún método numérico. No obstante, este método tiene la desventaja de amplificar los ruidos de la medición. Este ruido amplificado podría afectar el desempeño del sistema e incluso introducir oscilaciones que lo vuelvan inestable.

Para evitar estos problemas se decidió implementar un observador de estados que utiliza el modelo matemático para producir una estimación de las variables de estado en función de la salida y la entrada del sistema.

5.1. Ubicación de polos

Para el diseño se utilizó el sistema discreto de la Ecuación Nº 12 y se calcularon los polos para un tiempo de asentamiento $t_S = 0.05$ y un factor de amortiguamiento de $c_a = 0.95$. Esta selección se realizó teniendo en cuenta tanto el ruido como el tiempo de asentamiento del control de posición. Esto se debe a que, si el observador es más veloz, menos tiempo tardará en reducir el error entre la estimación y el valor real, logrando un mejor desempeño en el controlador. Sin embargo, esto se logra con ganancias de mayor magnitud, lo que produce la amplificación del ruido que puede estar presente en la medición, y, en consecuencia, comprometiendo la estabilidad del sistema. De modo contrario, si las

ganancias del observador tuvieran un valor demasiado pequeño y el tiempo de asentamiento fuera lento, el desempeño dinámico en las respuestas transitorias del controlador se verá afectado de manera negativa.

Los polos del observador se muestran en la Ecuación Nº 54 tanto para tiempo discreto como para tiempo continuo.

$$p = -80 \pm j26.295$$
$$p_d = e^{Tp} = 0.992 \pm j0.0026 \quad (54).$$

El vector de ganancia $\boldsymbol{K}_o$ se calculó tal que se cumpla la condición de la Ecuación Nº 55.

$$det(G_h' - C_h'\boldsymbol{K_o} - \lambda I) = 0 \quad (55),$$

$$\begin{vmatrix} \lambda - 1 + K_1 & K_2 \\ -T & \lambda - 1 \end{vmatrix} = 0 \quad (56),$$

$$\lambda^2 + (K_1 - 2)\lambda + (K_2 T + 1 - K_1) = z^2 + 2 * 0.992z + 0.9841 \quad (57),$$

$$K_1 = 2 - 2 * 0.992$$
$$K_2 = (0.9841 + K_1 - 1)/T \quad (58),$$

$$\boldsymbol{K_o} = [0.0159 \quad 0.7035] \quad (59).$$

Por último, en la Ecuación Nº 60, se muestra el observador calculado

$$\hat{\boldsymbol{x}}_{k+1} = G_h\hat{\boldsymbol{x}}_k + H_h\omega_h - \boldsymbol{K_o}C_h(\boldsymbol{x_k} - \hat{\boldsymbol{x}}_k) \quad (60),$$

donde $\hat{\boldsymbol{x}}_k$ representa las variables de estado estimadas.

6. Diagrama en bloques

En la Figura Nº 5 se muestra el diagrama en bloques de los controladores implementados, en conjunto con el observador de estados. En este diagrama se observa que la entrada es la señal de referencia Ref, es decir, la posición deseada de la esfera. Con este valor y la posición medida h se calcula el error de posición. Esta señal se hace pasar por un integrador discreto, que acumula el error para poder eliminar el error en estado estable. Luego se multiplica el error de la muestra actual, la velocidad observada y el error integrado por el vector de ganancias K_h, obteniendo la acción de control intermedia ω_h.

La velocidad observada es calculada por el observador de estados, el cual toma como entradas a ω_h y la posición medida para producir a la salida el vector de estados

estimados de la siguiente muestra $\hat{\boldsymbol{x}}_{k+1}$. Mediante el retraso z^{-1} aplicado a este se obtiene el vector de estados estimados de la muestra actual $\hat{\boldsymbol{x}}_k$, el cual contiene la posición y velocidad estimadas. La primera es la que se resta a h para calcular el error y multiplicarlo por la ganancia del observador K_o, esta señal es la que permite cerrar el lazo del observador y asegurar que la diferencia entre las variables estimadas y las reales tiendan a 0. Luego, las matrices G_h, H_h y C_h son las que contienen la información del sistema lineal.

Después de obtener ω_h, se utiliza la ley de control linealizante para calcular la referencia de corriente u_h. Está en conjunto con la corriente medida, se usa para calcular el error de corriente, el cual es integrado y multiplicado por el vector de ganancias K_i, obteniendo así la acción de control intermedia ω_i. Finalmente esta se utiliza para calcular la acción de control u mediante la ley de control linealizante. Esta señal es la tensión variable que es aplicada a la planta.

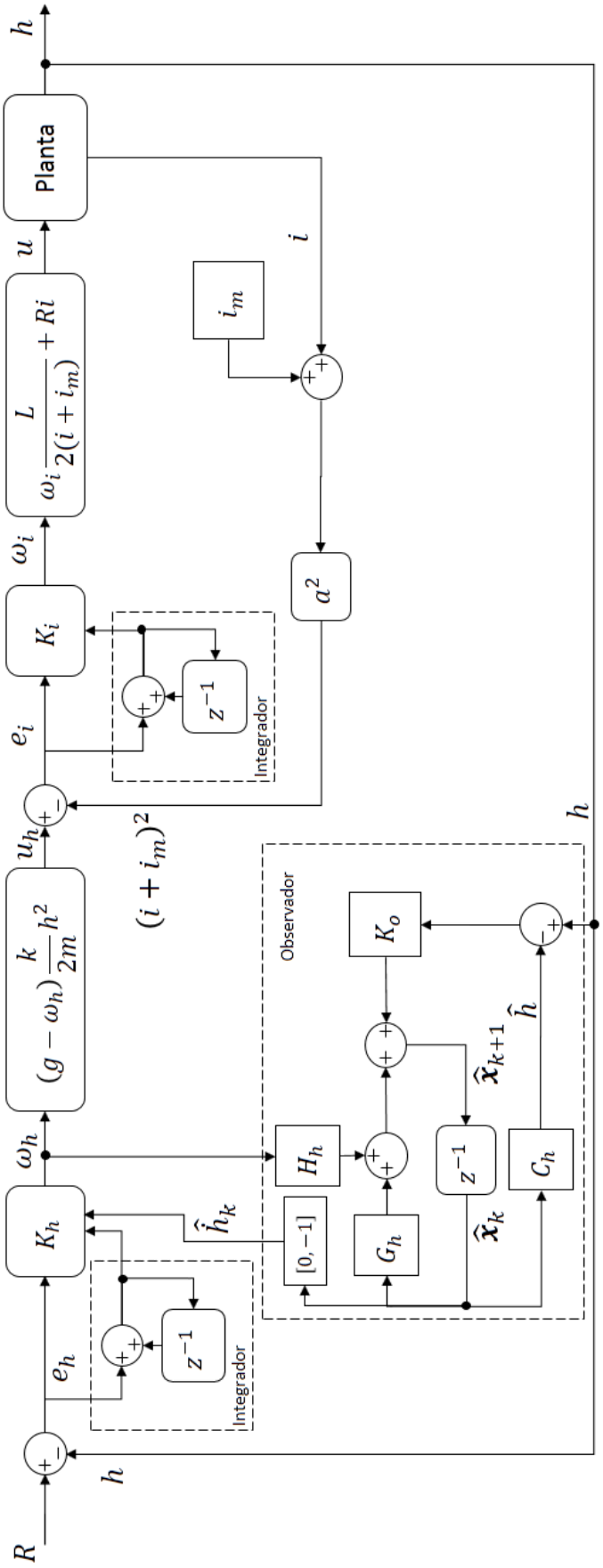

Figura Nº 5. Diagrama completo del controlador

CAPITULO 3: Simulaciones

En este capítulo se obtienen y analizan los resultados de simulación que validan el funcionamiento del controlador. El entorno utilizado para realizar las simulaciones es un software dedicado a esta tarea.

1. Construcción de la planta

Se comenzó por construir una versión de tiempo continuo de la planta, tal como se muestra en la Figura Nº 6. Este diagrama representa las ecuaciones del sistema definido en la Ecuación Nº 7.

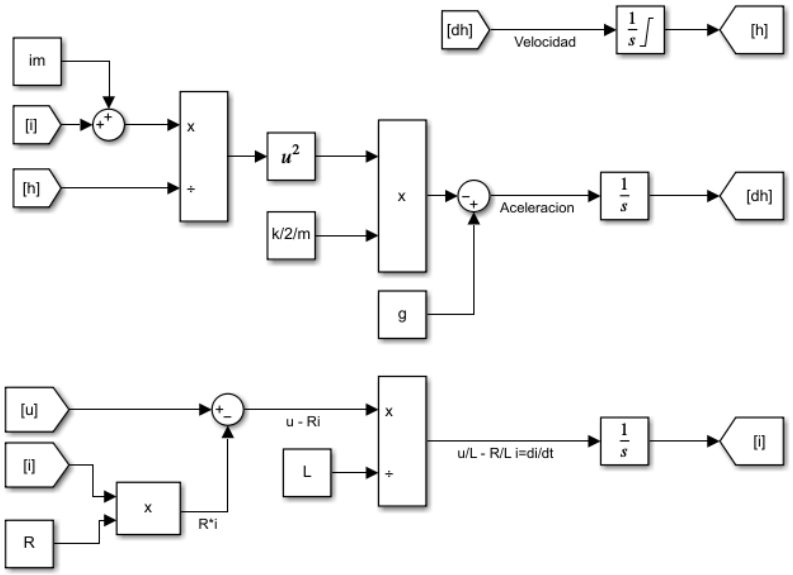

Figura Nº 6. Planta simulada

Los parámetros con los que se implementó la simulación son iguales a los valores medidos del prototipo real y se muestran en la Tabla Nº 1.

Tabla Nº 1. Parámetros de simulación

Parámetro	Valor
$\boldsymbol{Vcc}$	$12V$
$\boldsymbol{R}$	3.8Ω
$\boldsymbol{L}$	$32mH$
$\boldsymbol{m}$	$39g$
$\boldsymbol{i_m}$	$0.903A$
$\boldsymbol{k}$	$0.004H.m$
$\boldsymbol{i(0)}$	$0.618A$
$\boldsymbol{h(0)}$	$35mm$

El controlador y el observador se implementaron en un bloque “function” que se asemeja a un microcontrolador, tanto en el funcionamiento como en la programación. Este bloque se configuró a una frecuencia de 10Khz, la misma que el prototipo.

2. Resultados de simulación

Se realizaron distintas simulaciones para determinar la estabilidad y el desempeño dinámico del controlador.

2.1. Respuesta a una entrada escalón

Se realizó la simulación de la planta para una entrada escalón con valor de 5mm, las condiciones iniciales y los parámetros son los descritos en la Tabla Nº 1. El resultado obtenido se muestra en la Figura Nº 7.

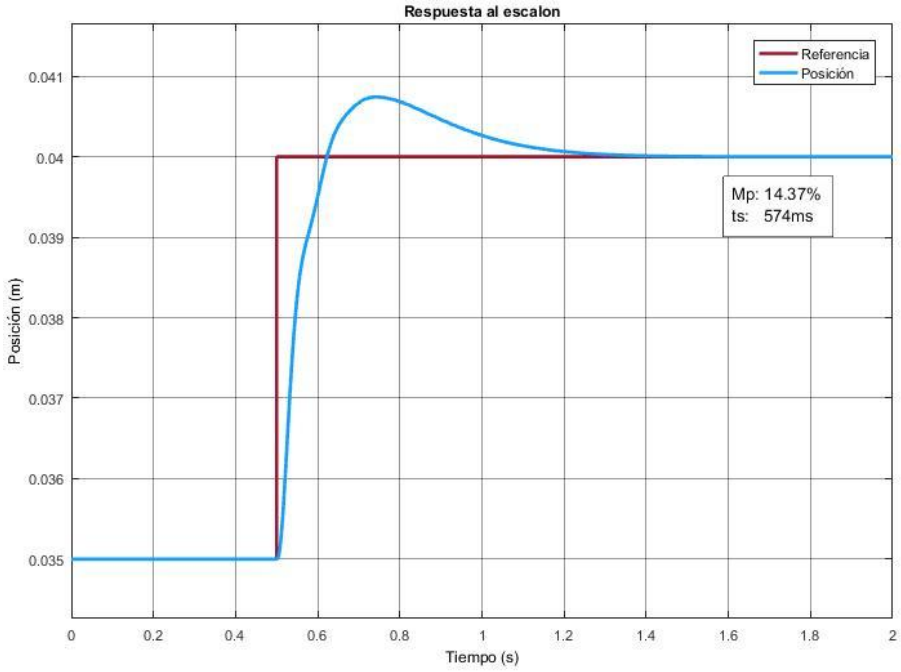

Figura Nº 7. Respuesta al escalón

En la imagen se aprecia tanto la señal de referencia como la posición de la esfera, el sobrepaso máximo obtenido fue de 14.37% y el tiempo de asentamiento de 574ms. Estos valores se encuentran un poco por encima de los calculados, sin embargo, son aceptables.

2.1.1. Acción de control

En la Figura Nº 8 se muestra el resultado de la acción de control para la simulación de la entrada escalón. Es importante que la acción de control no supere $\pm Vcc$, de lo contrario se producirá una saturación en la planta resultando en la inestabilidad del sistema. Del resultado obtenido se observa que efectivamente la acción de control no superó la tensión de alimentación del prototipo.

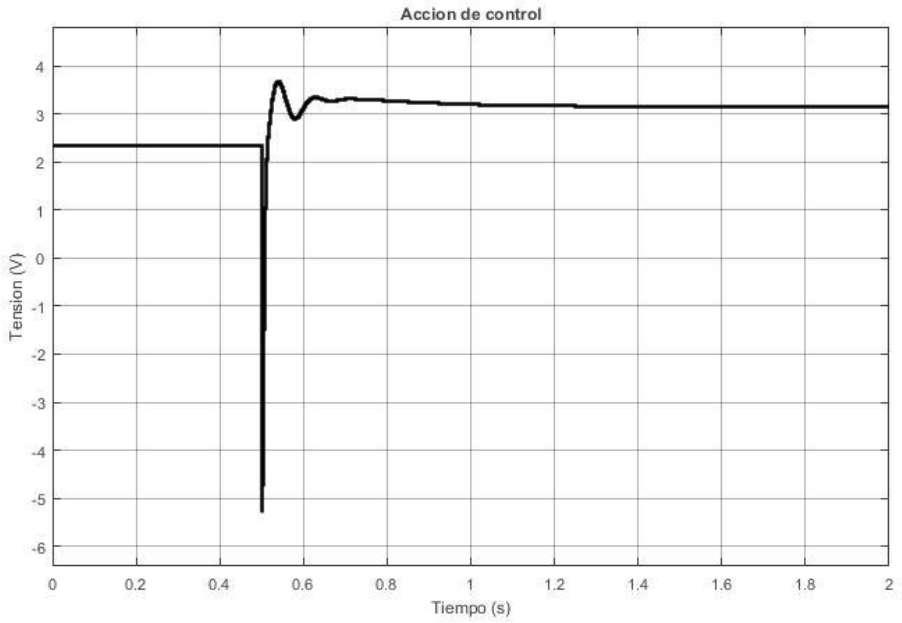

Figura Nº 8. Acción de control

2.2. Respuesta ante variaciones paramétricas

Se realizó la simulación para una variación paramétrica que consiste en la duplicación de la masa de la esfera en $t = 0.5s$. Los resultados obtenidos se muestran en la Figura Nº 9.

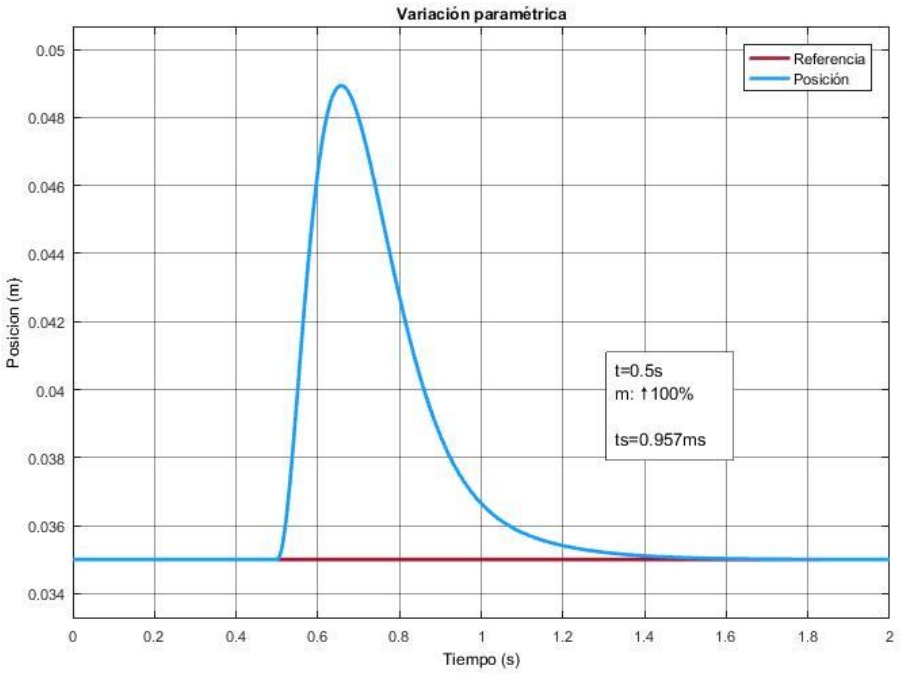

Figura Nº 9. Respuesta ante una perturbación

En la Figura Nº 9 se muestra una referencia constante en 35mm y la señal de posición de la esfera. Ante una variación paramétrica del 100% se produce un desvío de aproximadamente 14mm, sin embargo, el controlador es capaz de corregir esta desviación,

llevando a la esfera de nuevamente al valor de referencia. Esta simulación se realizó para comprobar la estabilidad del sistema ante variaciones paramétricas, y se concluyó que el sistema continúa siendo estable aun ante grandes perturbaciones.

2.3. Señal de posición contaminada con ruido

Se realizó la simulación del controlador introduciendo ruido blanco en la medición de la señal de posición. Este tiene una varianza de 10^{-6} y una frecuencia de 10Khz. En la Figura Nº 10 se muestran los resultados obtenidos.

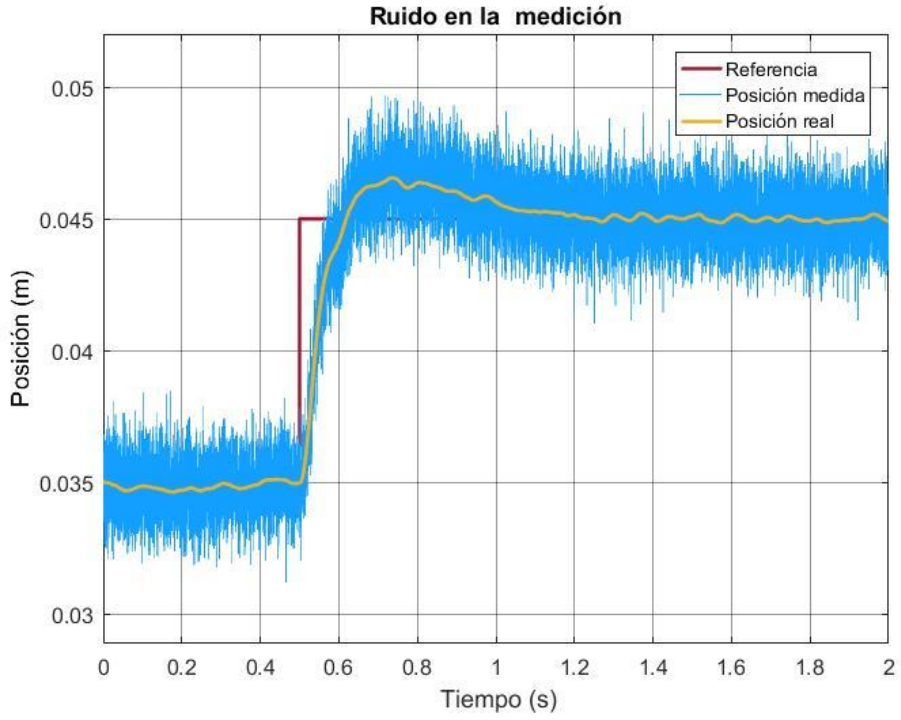

Figura Nº 10. Respuesta ante ruido

El principal objetivo de esta simulación fue analizar la estabilidad del sistema ante ruidos introducidos en el proceso de medición. En la gráfica anterior se observa que el desempeño dinámico sigue siendo satisfactorio incluso ante la presencia de ruido.

En la Figura Nº 11, se muestra la velocidad real y la observada, se ve que el observador no amplifica de manera desmedida el ruido, y la señal producida se aproxima a la velocidad real.

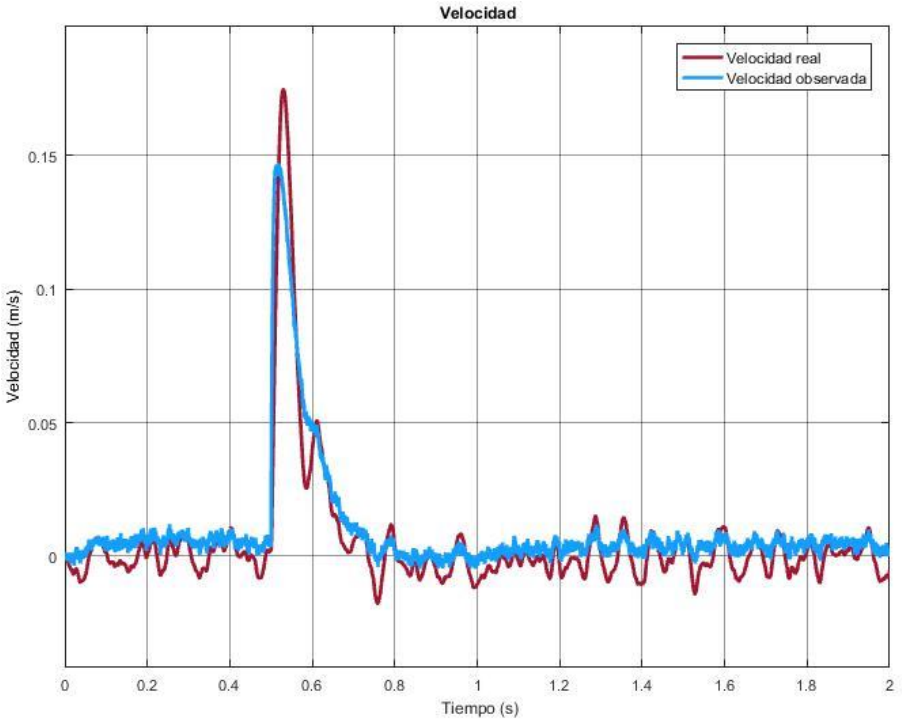

Figura Nº 11. Velocidad y estimación del observador ante ruido

CAPITULO 4: Implementación del controlador

En este capítulo se describe como se implementó el controlador calculado en el capítulo 2 y validado en el 3, en el levitador prototipo. El microcontrolador presente en la planta es un STM32F103C8T6, el cual se programó haciendo uso de la interfaz de desarrollo provista por el fabricante, “STM32CubeIDE v1.10.1”.

1. Configuración de pines

Para realizar la configuración de las entradas y salidas del microcontrolador se utilizó la herramienta proporcionada en el programa CubeIDE y siguiendo el esquemático del prototipo mostrado en la Figura Nº 12.

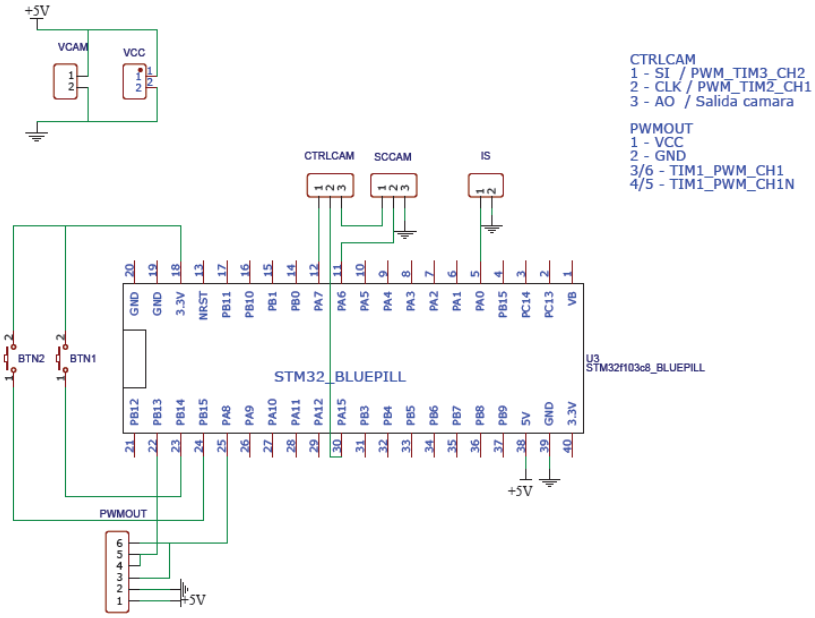

Figura Nº 12. Diagrama esquemático del microcontrolador

En la Figura Nº 12 se observan las conexiones de los pines del microcontrolador con cada una de los conectores a las etapas de adaptación de señal del levitador. Los parámetros con los que se configuró cada pin se muestra en la Tabla Nº 2.

Tabla Nº 2: Configuración de pines

PIN	Conexión	Periférico	Descripción
PA8	PWM	TIM1-PWM	Señal PWM que comanda el puente H.
PB13	PWM	TIM1-PWMN	Señal PWM complementaria.
PB14	Pulsador	GPIO	Pulsador de propósito general.
PB15	Pulsador	GPIO	Pulsador de propósito general.
PA6	Cámara	TIM3-IC	Detector de pulso.
PA7	Cámara	TIM3-PWM	Generador de pulso SI.
PA15	Cámara	TIM2-PWM	Señal de reloj.

PA0	Sensor de corriente	ADC1-IN0	Conversor analógico digital.

En las siguientes secciones se explican en detalle la función de cada entrada/salida del microcontrolador.

2. Convertidor CC-CC

La salida u del controlador es un valor de tensión, la forma en la que esta tensión es aplicada al electroimán del prototipo es a través de un convertidor de continua a continua puente H. Este convertidor hace variar la tensión de salida en función del ancho de pulso aplicado.

Para obtener la relación entre el ancho de pulso y la tensión de salida se analizó la gráfica de la Figura Nº 13. En esta se encuentra representada la forma de la tensión aplicada sobre el electroimán.

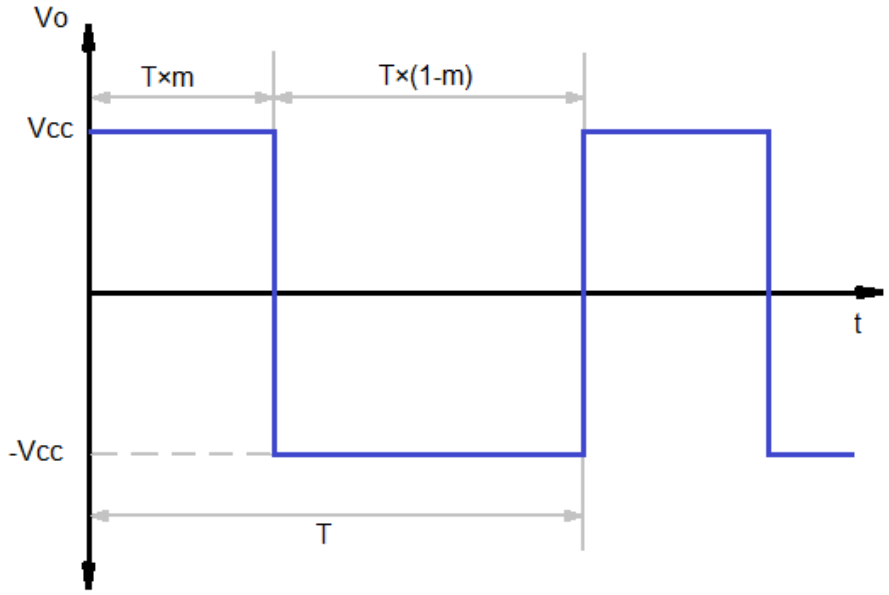

Figura Nº 13. Tensión de salida de un convertidor puente H

Se decidió trabajar con la tensión de salida promedio para obtener la relación entre el índice de modulación m y la tensión de salida promedio $\hat{v}_o$.

$$\hat{v}_o = \frac{V_{cc}(Tm) + (-V_{cc}(1-m)T)}{T} \quad (61),$$

$$\hat{v}_o = V_{cc}(2m-1) \quad (62).$$

Por lo tanto, el índice de modulación en función de la tensión de salida u requerida es

$$m = \frac{\left(\frac{u}{V_{cc}} + 1\right)}{2} \quad (63),$$

con $0 \leq m \leq 1$.

Para implementar la Ecuación Nº 63 en el microcontrolador se decidió dejar que el índice de modulación varíe de 0 a 2 y a este valor multiplicarlo por la mitad del periodo del temporizador 1. Tal como se muestra en la Ecuación Nº 64, en donde $CCR1$ es el registro que contiene el ancho de pulso de la señal PWM de salida.

$$TIM1 \rightarrow CCR1 = \left(\frac{u}{V_{cc}} + 1\right) 3600 \quad (64).$$

El temporizador 1 fue configurado con un periodo de 7200 ciclos de reloj, dado que la frecuencia del reloj interno del microcontrolador es de 72MHz, se obtiene como resultado una señal PWM con una frecuencia de 10KHz.

3. Adquisición de datos

Una de las partes más importantes al implementar un controlador digital es la etapa de adquisición de datos de los sensores. Como se mencionó anteriormente, este levitador cuenta con un sensor de posición y uno de corriente, y cada uno cuenta con sus respectivas etapas de adaptación de señales.

3.1. Sensor de posición

Este sensor es el que se encarga de medir la distancia relativa entre la esfera y el electroimán. Consiste en una cámara basada en el integrado TSL1401 [11], el cual es un sensor optoelectrónico que contiene una matriz de 128x1 fotodiodos dispuestos de manera lineal.

En la Figura Nº 14 se muestra una imagen de este dispositivo, el cual cuenta con 6 pines para su conexión, detallados en la Tabla Nº 3.

Figura Nº 14. Módulo de desarrollo basado en el sensor TSL1401

Tabla Nº 3: Pines de salida del TSL1401

PIN	Descripción
Vcc	Tensión de alimentación 3,3V a 5V
Gnd	Tierra.
AO	Señal de salida.
OUT	Señal de salida amplificada.
CLK	Señal de reloj.
SI	Entrada serial.

3.1.1. Funcionamiento

El diagrama en bloques del sensor se encuentra en la Figura Nº 15. En este se aprecia que cada fotodiodo contiene un circuito integrador que acumula un valor de tensión proporcional a la intensidad de luz recibida durante el periodo de integración.

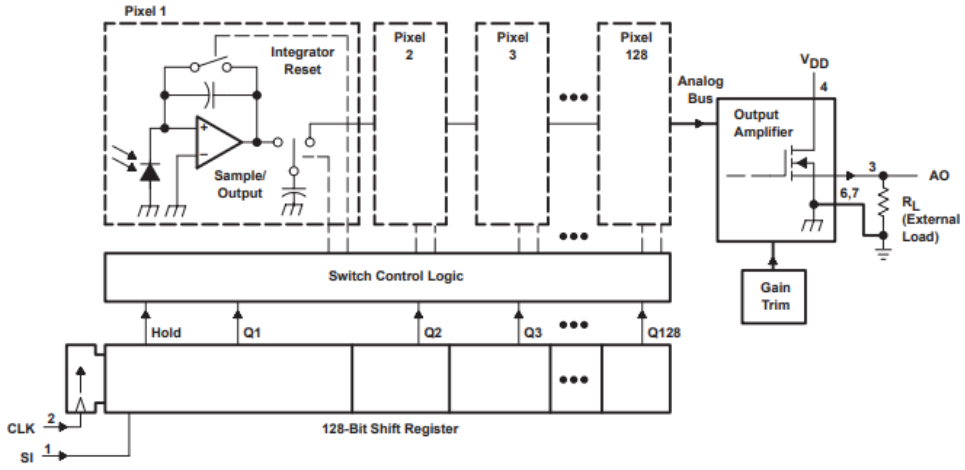

Figura Nº 15. Diagrama en bloques funcional del integrado TSL1401

El periodo de integración está dado por el periodo de la señal SI, la cual es un pulso corto que se aplica para reiniciar los integradores y activar el circuito de salida del integrado. El diagrama de tiempos de las señales se muestra en la Figura Nº 16, en esta figura se puede observar que el valor analógico de tensión de cada pixel se obtiene, tanto en AO como OUT,

uno detrás del otro arrancando por el primero y terminando por el 128, y el parámetro que dicta la velocidad a la que se los obtiene es la frecuencia de la señal CLK.

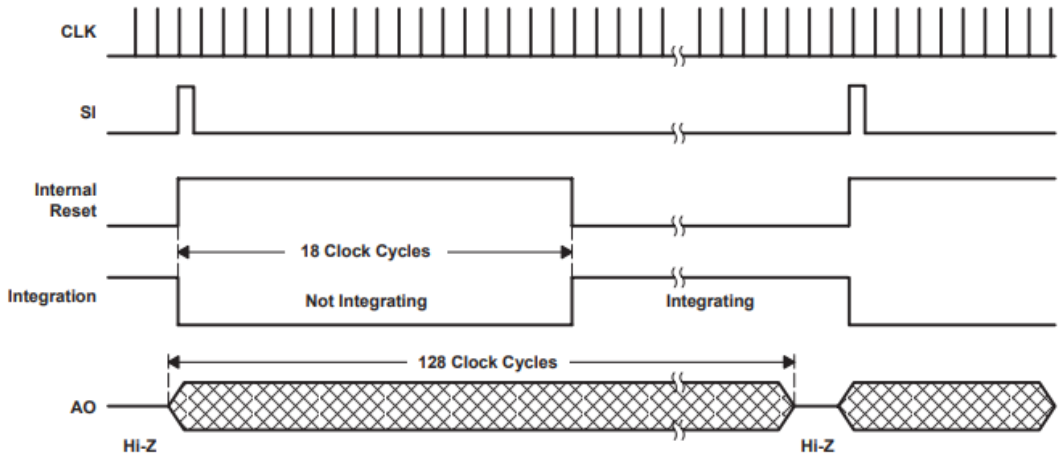

Figura Nº 16. Diagrama de temporización de las señales

Por todo lo dicho anteriormente el tiempo medido entre el inicio de SI y el flanco ascendente de la señal OUT es proporcional a la posición de la esfera tal como se muestra en la Figura Nº 17.

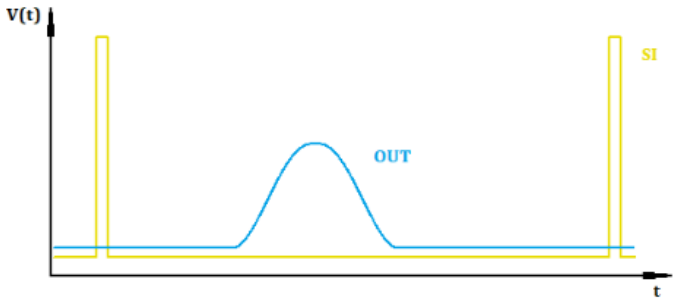

Figura Nº 17. Salida del sensor y señal SI

En la Figura Nº 18 se muestra las distintas señales de salida para distintas posiciones de la esfera con el objetivo de esclarecer el funcionamiento.

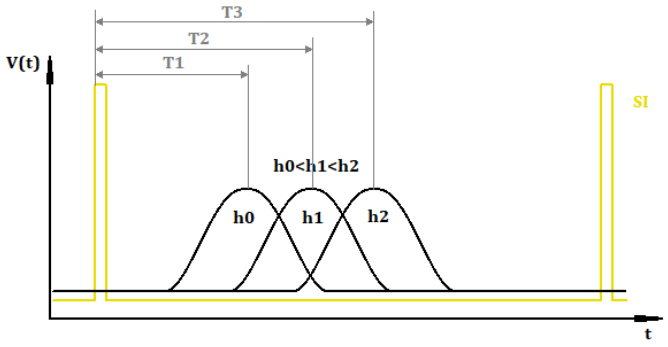

Figura Nº 18. Distintas salidas en función de la posición de la esfera

3.1.2. Etapa de adaptación de señales

Para mejorar la detección del flanco y hacer al sensor más confiable se decidió diseñar una etapa de adaptación de señales con el fin de acondicionar la magnitud de la señal OUT a 3,3V. El circuito implementado se muestra en la Figura Nº 19.

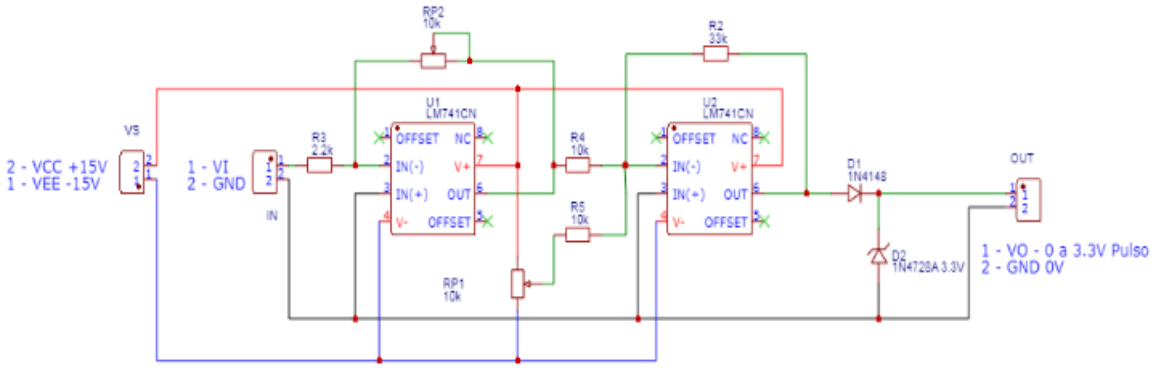

Figura Nº 19: Etapa de adaptación de señales.

El circuito utiliza dos etapas de amplificación construidas con amplificadores operacionales LM741. La primera etapa es un amplificador inversor con ganancia ajustable, y la segunda es un amplificador sumador inversor, que suma a la señal de salida de la etapa anterior una componente de tensión continua ajustable.

Este circuito funciona amplificando la señal de entrada, de manera que se necesite un menor nivel de iluminación para la detección, y luego recortar a valores de tensión más bajos, para fijar un umbral de detección. La ganancia y la tensión de umbral se ajustan mediante RP2 y RP1 respectivamente.

3.1.3. Medición de la posición

Para lograr utilizar el sensor de posición, se configuró el microcontrolador para generar las señales de CLK y SI. Para la primera se utilizó el temporizador 2 (TIM2) con un periodo de 72 ciclos de reloj, es decir, una frecuencia de 1MHz. El ancho de pulso se configuró en 35 ciclos de reloj.

Para generar la señal SI y medir el retardo entre esta y la señal OUT, se utilizaron 2 canales del temporizador 3 (TIM3). Este se configuró con un periodo de 7200 ciclos de reloj, es decir, una frecuencia de 10kHz. El canal 1 se utilizó para realizar la medición, el temporizador detecta el flanco ascendente de la señal y guarda el valor del contador en el registro CCR1, además se genera una interrupción que se utiliza para almacenar el valor de este registro en una variable.

El canal 2 del temporizador se utiliza para generar la señal SI, este fue configurado como generador de ancho de pulso modulado, el valor del ancho de pulso es de 73 ciclos de reloj.

Para convertir el valor del temporizador dado en cantidad de ciclos de reloj a un valor en metros, correspondiente a la posición de la esfera, se utilizó una recta de calibración con los valores mostrados en la Ecuación Nº 65.

$$h = (0.02706 \times T - 66.24) \times 1000 \ [m] \tag{65}$$

Esta fue obtenida realizando la medición de la posición de la esfera con un calibre y comparándola con el valor medido en el temporizador, en un rango de 12.5mm a 92.5mm con un intervalo de 10mm. Luego se obtuvo la recta que mejor aproxima los puntos con el método de mínimos cuadrados. La gráfica de las mediciones y la recta mencionada se muestra en la Figura Nº 20.

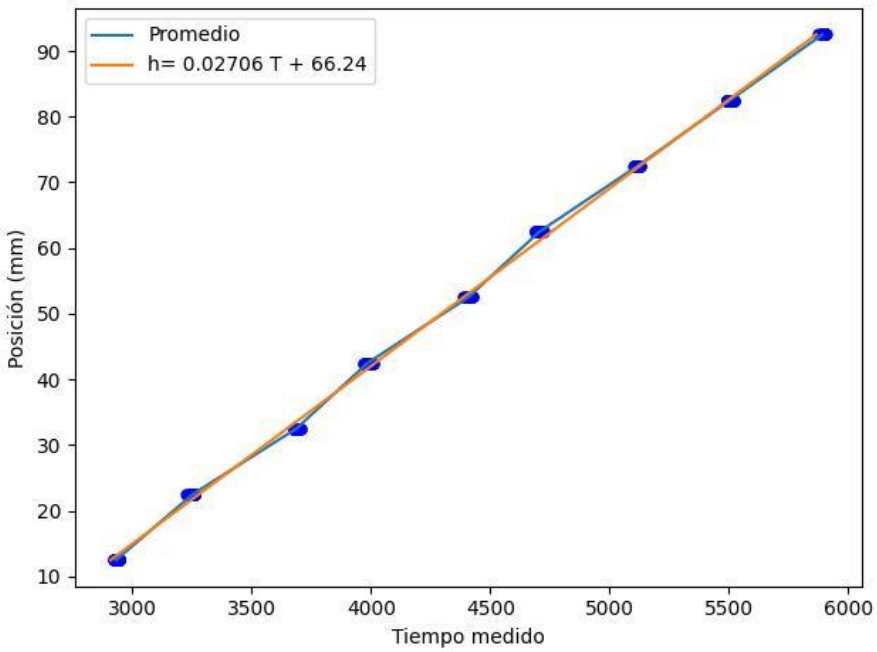

Figura Nº 20. Recta de calibración del sensor

3.2. Sensor de corriente

El sensor de corriente en conjunto con su etapa de adaptación de señales genera un valor de tensión proporcional a la cantidad de corriente que pasa por el mismo. Por lo que para realizar la medición se utiliza un conversor analógico digital (ADC).

El microcontrolador cuenta con 2 ADCs de 12 bits con 10 canales cada uno. El que se utilizó para realizar la conversión es el canal 1 del ADC1. El inicio de la conversión se

realiza cada vez que se reinicia el contador del temporizador 3, lo que establece la frecuencia de muestreo en 10kHz. Una vez que este periférico termina de realizar la conversión, genera una interrupción en donde se convierte el valor leído del ADC en un valor de corriente mediante una recta de calibración, definida en la Ecuación Nº 66.

Para obtenerla se realizaron 3 mediciones con un osciloscopio y una pinza de corriente. La primera se realizó con un ciclo de trabajo del 50% en el convertidor de CC-CC por lo que la corriente promedio en la bobina tomo un valor de 0A, la segunda y tercera medición se realizaron con un ciclo de trabajo de 0% y 100% obteniendo los valores máximos y mínimos de la corriente. Los resultados se muestran en la Tabla Nº 4.

Tabla Nº 4: Resultados de las mediciones para calibración

Ciclo de trabajo	Corriente	ADC
0%	-4.785A	0
50%	0A	2048
100%	4.700A	4096

El procedimiento utilizado para obtener los parámetros de la Ecuación Nº 66 se muestra a continuación.

$$i = V_{ADC} * m + b \quad (66),$$

$$m = \frac{i_{100\%} - i_{0\%}}{4096 - 0} = 0.0023157 \quad (67),$$

$$b = i_{0\%} = -4.785A \quad (68).$$

Por lo tanto, la corriente

$$i = V_{ADC} * 0.0023157 - 4.785 \quad (69),$$

donde:

- V_{ADC}: Es el valor generado por el conversor.
- i: Es el valor de corriente.

4. Controlador

Las ecuaciones tanto del controlador como del observador se implementaron en la interrupción generada por el ADC. En un principio se implementó el algoritmo del controlador con variables representadas en punto flotante, sin embargo, el tiempo de procesamiento era mayor que el tiempo entre interrupciones por lo que funcionaba a 5Khz. Por lo tanto, se decidió

implementar un algoritmo de punto fijo, lo que redujo de manera notable el tiempo de procesamiento. La comparación se puede ver en el Capítulo 6: Resultados experimentales.

Para lograr una buena precisión y evitar problemas de desbordamiento (“Overflow”) en los resultados intermedios, se decidió utilizar todas variables de 64 bits. Se utilizaron 24 bits para representar la parte fraccional, 39 para la parte entera y un bit de signo. Por lo tanto el numero decimal más pequeño que se puede representar es $\pm 2^{-24} \cong 5.96 \times 10^{-8}$ y el máximo número representable es $\pm 2^{39} + (1 - 2^{-24}) \cong 5.498 \times 10^{11}$. Con esta selección realizada se obtuvo un buen rango de números representables.

Independientemente del modo de representación de las variables, a continuación, se explica el funcionamiento de este algoritmo.

Primero se realiza la conversión de los valores medidos con las rectas de calibración, y se filtran con un filtro pasa bajas digital de respuesta al impulso infinita de polo simple (Single-pole IIR). Estos filtros se utilizaron para reducir el ruido en las mediciones.

Luego se realiza el cálculo de la referencia explicado en la siguiente sección. Este valor se utiliza para calcular el error de posición y su integral. Esta última se calcula acumulando el error en cada muestra.

Con los valores obtenidos se calcula la entrada “wh” y se aplica la ley de control de la Ecuación Nº 33, dando como resultado la referencia del controlador de corriente.

Se calcula el error de corriente y su integral, luego se aplica la ley de control de la Ecuación Nº 52 y se divide por la tensión de alimentación del prototipo (12V) para obtener un valor entre -1 y 1, es decir el índice de modulación o ciclo de trabajo.

Luego a este índice se le suma 1, se multiplica por la mitad del periodo de muestreo y se escribe en el registro CCR1 del temporizador 1. Por último, se calcula la velocidad con la Ecuación Nº 60.

En la Figura Nº 21 se muestra parte del código implementado, la versión completa se visualiza en el ANEXO 1: Código fuente STM32.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){
      // Adquisición de datos
      i_f=(HAL_ADC_GetValue(&hadc1)*IA)-IB;
      h = ((h_raw*HA) - HB)/1000;

      capturas=0; //Control de sobrecaptura

      // Filtrado
      i=fpm(IIR_FIL , ( i_f - i )) + i;
      hf=fpm(IIR_FIL , ( h - hf )) + hf;

      // Calcular la referencia
      REF_pointer[RefCtl.ref_sel]();
      n_muestra= (n_muestra+1)%(RefCtl.periodo*2);

      // CONTROL DE POSICION //
      eh=(ref-hf);
      eh_int+=eh;
      Wh=fpm(KH1,eh)-fpm(KH2,dh_est)+fpm(KH3,eh_int);
      uhc=fpm(fpm(G-Wh,hf),fpm(hf,ACH0));

      // CONTROL DE CORRIENTE //
      i_io=i-IM;
      ei=uhc-fpm(i_io,i_io);
      ei_int+=ei;
      Wi=(fpm(KI1,ei)+fpm(KI2,ei_int));
      acC=(fpd(fpm(Wi,ACI0),i_io)+fpm(ACI1,i));

      TIM1->CCR1 =((fpd(acC,VI)+FACTOR_ESC)*3600)>>FBITS;

      // OBSERVADOR
      eo=hf-h_est;
      long long dh_tmp=dh_est+fpm(B1,Wh)+fpm(KO2,eo);
      h_est = h_est+fpm(G12,dh_est)+fpm(Wh,B0)+fpm(KO1,eo);
      dh_est=dh_tmp;
            ....
```

Figura Nº 21. Algoritmo de control en punto fijo implementado en la interrupción del ADC

5. Generación de referencia

Se decidió calcular la señal de referencia en cada interrupción del ADC debido a que resulta más rápido que enviarla desde la interfaz gráfica, además, se logra una mejor resolución para señales que varían con el tiempo debido a que es muy complicado lograr la sincronización perfecta entre esta última y el microcontrolador.

La referencia es generada por un array que contiene 4 punteros a funciones. Cada una de estas funciones se utiliza para generar un tipo de referencia distinto, por lo que variando el índice del puntero de 0 a 3, se generan una señal constante, cuadrada, triangular y sinusoidal respectivamente.

Para implementar la señal sinusoidal se decidió incluir una tabla con valores pre calculados, ya que calcular la función seno es costoso computacionalmente. Esta tabla

contiene 2500 puntos, correspondiente al primer cuadrante, es decir, el intervalo semiabierto $\left[0;\frac{\pi}{2}\right)$ de una señal seno. Para obtener el segundo se invierte la dirección en la que se recorre la tabla, en el tercero se invierte el valor, por último, para el cuarto se invierten los dos parámetros. De esta manera se reduce el espacio requerido de 40Kb si se almacenase la tabla completa, a tan solo 10Kb, una reducción significativa teniendo en cuenta que el microcontrolador cuenta con solo 64Kb de memoria para almacenar el programa.

Cada una de las funciones utilizan 3 variables para generar la señal, el offset, la amplitud y el periodo, estas variables fueron creadas de forma global para que pudieran ser modificadas por cualquier otra función en el programa, Lo cual facilito enormemente la comunicación con la interfaz.

En la Figura Nº 22 se muestra el significado de cada una de estas magnitudes con el ejemplo de una señal triangular.

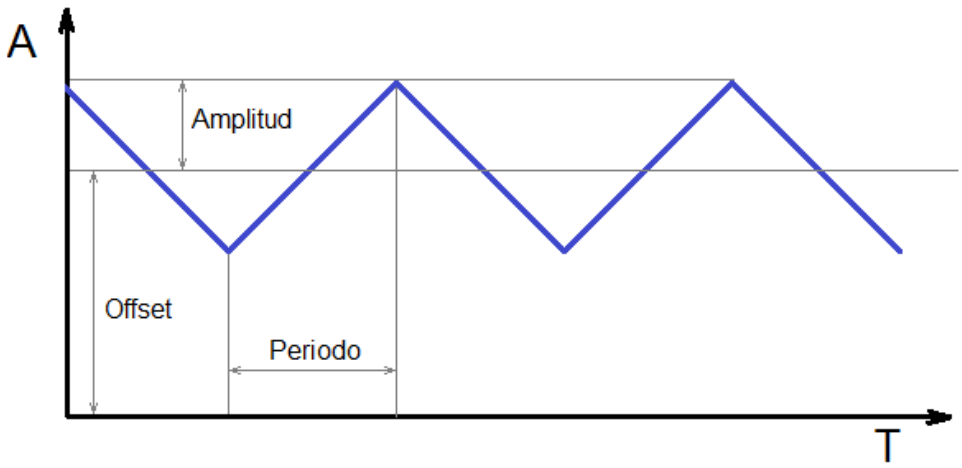

Figura Nº 22. Parámetros de control de la referencia

Por motivos de simplicidad en la programación se decidió que la variable ‘periodo’ sea la mitad del periodo real de la señal.

6. Comunicación con interfaz

Para realizar la comunicación con la interfaz gráfica, se dispuso de una conexión a través de un cable USB. El microcontrolador STM32 posee la capacidad de generar una conexión a través de un puerto serial virtual, que corre sobre el protocolo de comunicación USB 2.0 con una velocidad de 12Mbits/s.

6.1. Recepción de datos

El microcontrolador se programó para que reciba datos en un buffer de 16bytes, este buffer recibe en total 4 números que son los correspondientes a la selección de la señal de referencia, el offset, la amplitud y el periodo. En la Figura Nº 23, se muestra como está constituido.

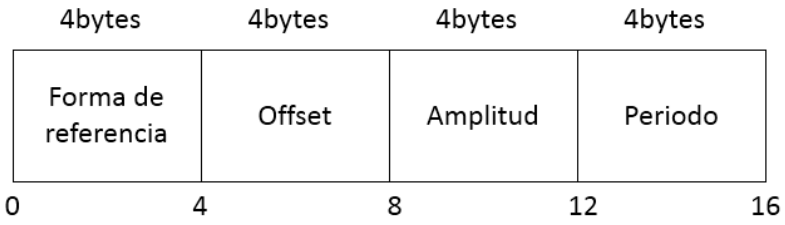

Figura Nº 23. Buffer de recepción de datos

Una vez que se reciben los datos, se activa una interrupción en donde estos se decodifican y se almacenan en las variables que controlan la generación de referencia.

6.2. Transmisión de datos

Se programó el microcontrolador para que envíe los datos de las señales de posición, referencia, corriente, referencia de corriente y velocidad. Al final de 1 de cada 4 interrupciones generadas por el ADC se almacenan los valores de las muestras de las señales mencionadas en un buffer de 1001 bytes para su posterior envío. Una vez que este buffer está repleto, se procede a iniciar la transmisión hacia la interfaz. Debido a esto, se obtiene una tasa de muestreo de 2.5Khz, lo que en la práctica resultó suficiente para representar las señales.

Como cada muestra de cada señal es un numero de punto fijo de 64 bits, se descartan los 32 bits más significativos de cada una de las muestras y se usan 4 bytes del buffer para almacenar cada una de ellas. Esto se realizó de esta manera dado que los valores que toman estas señales nunca superan magnitudes de ±128, durante el funcionamiento normal del prototipo.

En total tiene espacio para almacenar 250 muestras, 50 por cada una de las 5 señales. Se estructuró tal y como se muestra en la Figura Nº 24.

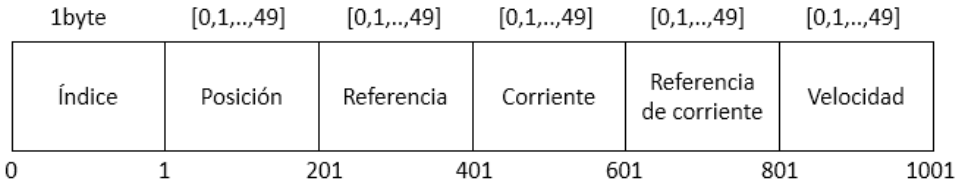

Figura Nº 24. Buffer de envío de datos hacia la interfaz

Debido a que cada señal se almacena en un buffer circular, el índice donde se encuentre la muestra más antigua es desconocido por la interfaz, por lo que el primer byte que se envía es justamente este índice. Con esto se evitan problemas de desincronización y se asegura una representación correcta de la señal en la gráfica de la interfaz, ya que puede reordenar los datos para luego graficarlos.

7. Configuración de los botones

Este prototipo cuenta con dos botones (superior e inferior) que se pueden programar para realizar la función que se desee al ser pulsados. El botón inferior se configuró para que reinicie el controlador, estableciendo los acumuladores de error y las variables estimadas a 0, además de establecer el valor de referencia igual al valor de posición medido en ese momento.

El botón superior se configuró para variar la señal de referencia en ±5mm por lo que simula una entrada escalón. Si este botón se deja pulsado por más de 2 segundos, se detienen las interrupciones del ADC y se coloca el ciclo de trabajo del PWM en 50%, es decir, se lleva la corriente promedio a 0. Entrando de esta manera en un modo de reposo, el cual se indica con el parpadeo intermitente del led indicador cada 1 segundo.

Para salir de este modo se puede presionar el botón inferior, o reiniciar el microcontrolador. Si se vuelve a presionar el botón superior, el levitador entra en modo demostración, lo cual implica que levanta la esfera de la base y la lleva a 35mm. Luego se aplica una referencia cuadrada con 3mm de amplitud y una frecuencia de 2hz durante 10 periodos, seguido a una onda triangular con los mismos parámetros, una sinusoidal y por último se aleja la esfera del electroimán lentamente hasta depositarla en la base, entrando nuevamente al modo de reposo.

En la Figura Nº 25 se muestra una imagen de la esfera levitando y el detalle de la base removible mencionada anteriormente.

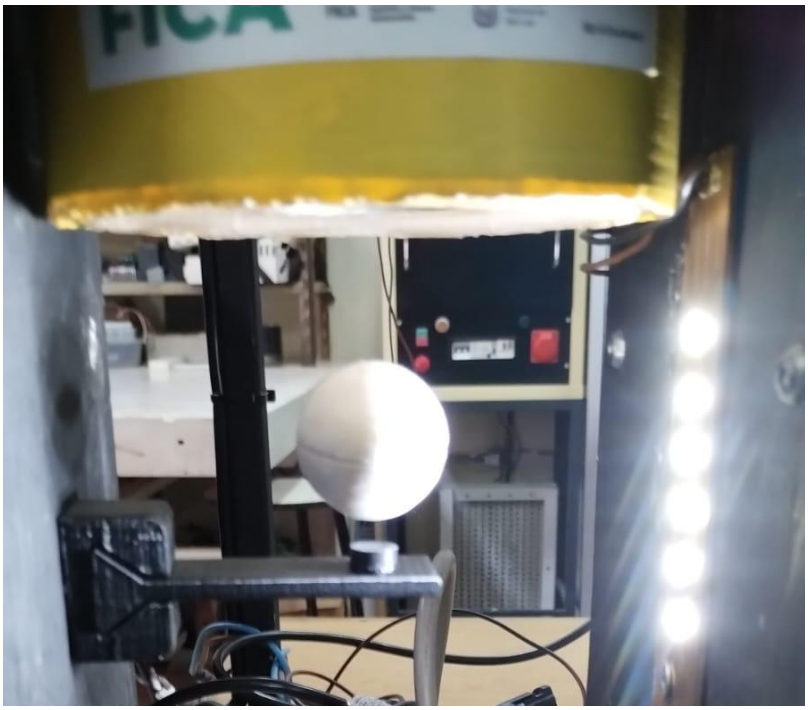

Figura Nº 25: Esfera levitando

CAPITULO 5: Interfaz gráfica

En este capítulo se trata el diseño e implementación de una interfaz gráfica de usuario, la cual permite interactuar con el levitador de una manera rápida y sin la necesidad de utilizar instrumentos de medición ajenos al mismo.

1. Objetivos

Los objetivos que debe cumplir esta interfaz se listan a continuación.

- Presentar tanto los datos de los sensores como la referencia en una gráfica que se actualice en tiempo real.
- Permitir la variación de la señal de referencia.
- Mostrar el valor numérico de las variables medidas principales.
- Se deben poder realizar ensayos y obtener las gráficas correspondientes.
- Es deseable que no tenga problemas compatibilidad con otros sistemas operativos.
- Debe tener licencia de código abierto.

2. Selección del lenguaje de programación

Los lenguajes de programación que se analizaron para el diseño de esta interfaz fueron C++ y Python.

En cuanto a C++ se tiene la ventaja de que se puede lograr un mejor rendimiento ya que es un lenguaje compilado. La principal desventaja son los largos tiempos de desarrollo que requiere, debido a que cada vez que se desea probar algún cambio, es necesario compilar el programa nuevamente.

En cuanto a Python, es un lenguaje fácil de entender que tiene un tiempo de desarrollo menor que C++. Además, el autor de este trabajo ya contaba con experiencia previa en el desarrollo de interfaces gráficas sobre esta plataforma. La principal desventaja es que el rendimiento que se puede lograr es menor en comparación a la otra opción analizada.

En cuanto a las librerías y paquetes que se utilizaron en el desarrollo de este programa, por lo general se encontraban disponibles, las mismas o equivalentes, en ambos lenguajes. Con la desventaja de que en C++ se deben encontrar los códigos fuente de cada librería a utilizar y compilarlas directamente en el programa, o bien, incluir los archivos binarios de la librería pre compilada en los archivos de distribución.

El lenguaje de programación seleccionado para la implementación de esta interfaz gráfica de usuario es Python en su versión 3.8. Se eligió este lenguaje debido a que es posible lograr el desempeño requerido con un programa bien estructurado y optimizado. Y esto último fue más sencillo de lograr debido a la rápida introducción de cambios durante el diseño del programa.

2.1. Selección del módulo de desarrollo de GUI

Para desarrollar esta interfaz, Python cuenta con varias opciones de módulos de entre las cuales se destacan PyQt5, PySide2 y Tkinter. Esta última, es la opción de desarrollo que viene instalada de manera nativa en Python, su ejecución suele ser pesada y un tanto tediosa su programación. En cambio, PyQt5 y PySide2, son módulos basados en Qt5.

Qt es un marco de desarrollo (“Framework”) completo con herramientas diseñadas para agilizar la creación de aplicaciones e interfaces de usuario para plataformas de escritorio, integradas y móviles [12]. Este marco está construido en C, y cuenta con distintas herramientas gráficas que permiten un rápido diseño de la interfaz. Además, se encuentra disponible con licencia de código abierto.

La librería que se decidió utilizar es PySide2, debido a que trae incluidas todas las herramientas adicionales de depuración y desarrollo de Qt, por defecto en el paquete de instalación.

3. Diseño de la interfaz

Se realizó el diseño estético de la ventana principal en el programa QtDesigner, el cual permite la rápida disposición de elementos y widgets que la componen.

Esta disposición se muestra en la Figura Nº 26. Está dividida en dos secciones, una izquierda y una derecha, en la primera se encuentran todos los botones, casillas de texto y numéricas, que permiten la interacción del usuario con el levitador. En la parte derecha se dispuso de un contenedor donde se colocó la gráfica de los datos, la cual se trata más adelante.

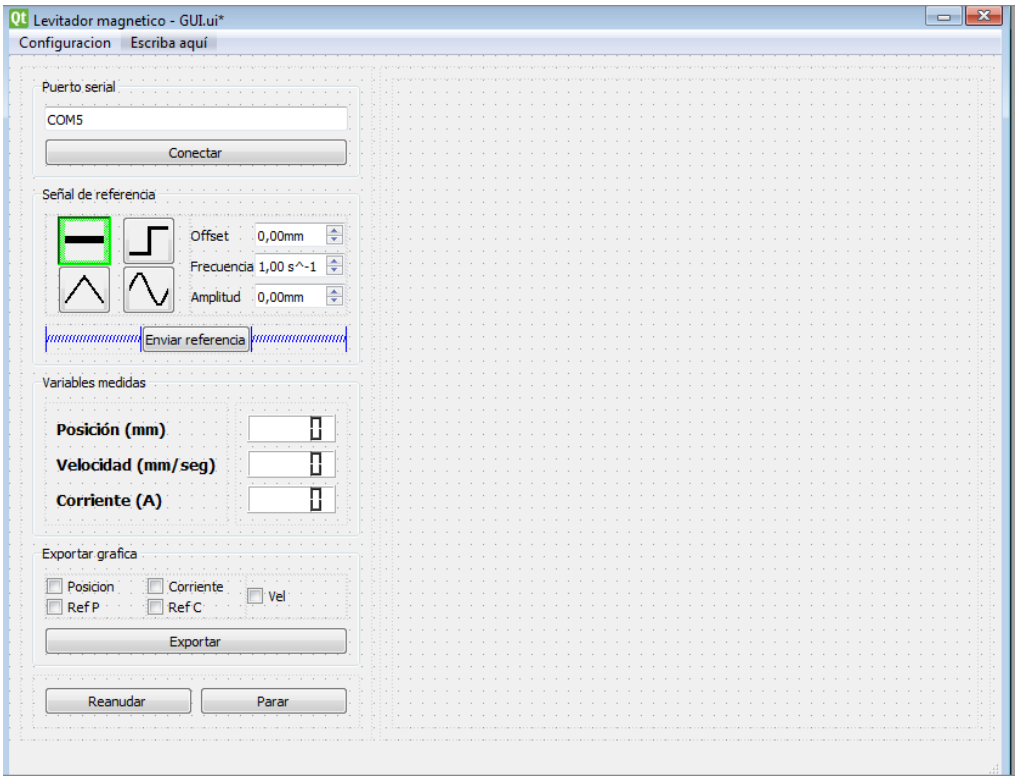

Figura Nº 26. Diseño de la interfaz en QtDesigner

4. Estructura del programa

En esta sección se describe como se estructuró el programa y cada una de las partes que lo componen. El diagrama general se muestra en la Figura Nº 27.

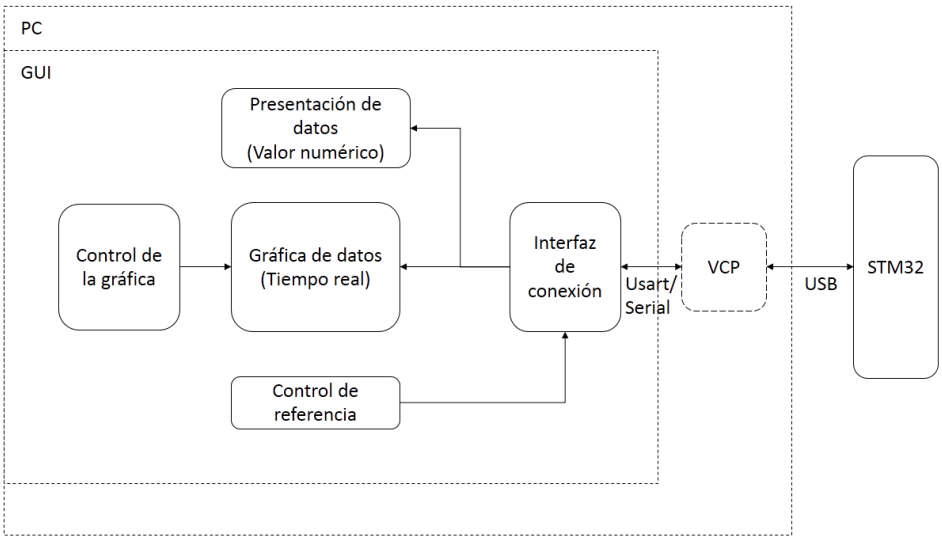

Figura Nº 27. Estructura del programa

Esta GUI consiste de una ventana que se ejecuta en una PC y se conecta con el microcontrolador del prototipo de levitador a través de un cable USB. En la PC se encuentra

instalado un driver que convierte esta conexión en una conexión serial virtual, por la cual se pueden enviar y recibir datos.

4.1. Interfaz de conexión

La conexión y desconexión al puerto serie deseado se realiza mediante la primera caja de grupo, mostrada en la Figura Nº 28. Esta caja contiene una línea de edición de texto que permite colocar el nombre del puerto, y un botón que realiza la conexión.

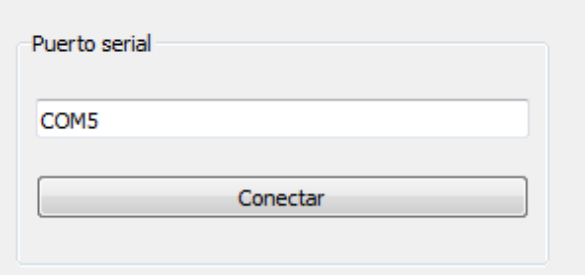

Figura Nº 28. Comunicación con el microcontrolador

Este botón genera una instancia de una clase “Datareader”. Este objeto es el que utiliza el programa para interactuar con el microcontrolador, es decir, enviar y recibir datos. La ventaja de separar esta funcionalidad de la ventana principal es que es ejecutada en un hilo de procesamiento paralelo, lo cual mejoró la latencia del programa.

Esta clase funciona creando una conexión al puerto serie pasado como parámetro, y estableciendo un temporizador cada 19ms. Estas muestras son desempaquetadas, es decir, convertidas de una cadena de bytes a un array de números enteros de 32 bits, luego se dividen por la precisión de punto fijo, es decir, 2.0^{24} para obtener la representación de punto flotante y son guardadas en la variable que contiene los últimos 3 segundos de información de la señal.

4.2. Control de la referencia

El control de la referencia se realiza con los botones y números de la caja “Señal de referencia” mostrada en la Figura Nº 29.

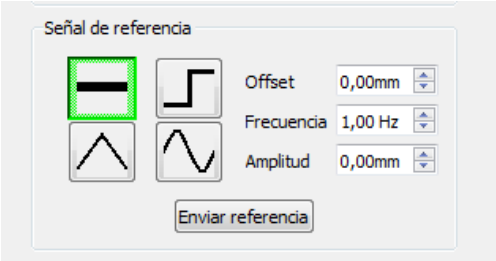

Figura Nº 29. Control de la referencia

Cada uno de los botones permiten seleccionar el tipo de señal, si es constante, una onda cuadrada, triangular o sinusoidal. Cada una de estas tienen tres parámetros de ajuste, offset, amplitud y frecuencia, los cuales se describen en la Figura Nº 30.

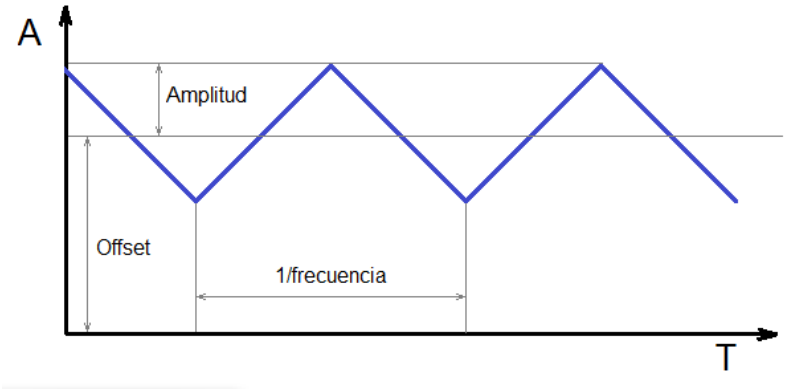

Figura Nº 30. Parámetros de control aplicados a una referencia triangular

Para enviar la nueva referencia al microcontrolador se utiliza el botón de “Enviar referencia”. Este botón funciona escribiendo los parámetros introducidos por el usuario en la variable que contiene el vector de referencia en la interfaz de conexión, y enviándolos hacia el microcontrolador.

4.3. Presentación de datos

La presentación de los datos se realiza con los widgets “LCD Display”, contenidos en la caja “Variables medidas” que se muestra en la Figura Nº 31. Estos permiten conocer al usuario el valor numérico de la última muestra recibida para cada variable de estado del sistema. Este valor se actualiza al mismo tiempo que la gráfica mediante un temporizador de 50ms.

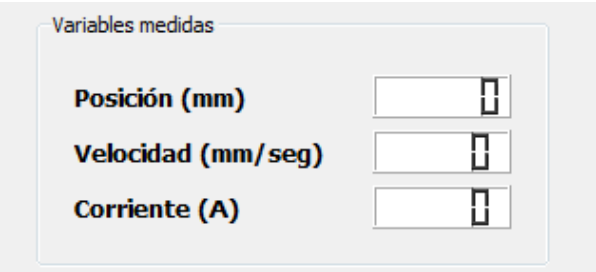

Figura Nº 31. Presentación de los valores numéricos de las variables del sistema

4.4. Gráfica

La gráfica de la/s señal/es recibida es la parte central del programa. Para lograr graficar los datos, en un principio, se comenzó utilizando la librería “MatPlotLib”, que es la más popular de Python en esta función debido a su similitud con MatLab.

Las gráficas que se obtienen con esta librería tienen un aspecto estético profesional, pero su rendimiento no es el óptimo para una aplicación en tiempo real, incluso luego de realizar las optimizaciones pertinentes. El mejor tiempo de procesamiento que se utilizaba para actualizar la gráfica cada vez que se recibía un paquete de muestras, era mayor a 50ms antes de las optimizaciones. Este tiempo se logró reducir a 20ms realizando optimizaciones y reduciendo la cantidad de muestras graficadas, sin embargo, se requiere que el tiempo de procesamiento de la imagen sea menor que el tiempo en que se reciben nuevos datos, es decir, 19ms. Con esto se garantiza cumplir con el objetivo de visualización en tiempo real sin pérdida de datos entre medio. Además, tener poco tiempo de procesamiento libre significó que la interacción con el programa presentaba retrasos detectables, lo cual degradaba la experiencia de usuario y resultaba en bloqueos ocasionales.

Debido a esta gran desventaja, se decidió reemplazarla por el modulo “pyqtgraph”. El enfoque sobre el cual se encuentra desarrollado este último es el rendimiento y la velocidad de actualización en detrimento del aspecto.

Con la adopción de esta nueva librería se lograron reducir los tiempos de procesamiento en la actualización a menos de 3.2ms. Lo cual permitió la incluir la opción de realizar tres gráficas en tiempo real, una para cada variable de estado del sistema.

Además, tiene la ventaja de que es más sencillo para el usuario acceder a las configuraciones de la gráfica. Con un simple clic derecho sobre la misma, se abre toda una lista de opciones que permite ajustar desde la escala de los ejes, hasta poder realizar la visualización de la transformada rápida de Fourier de la señal.

Los controles para comenzar y detener tanto la actualización de la gráfica como la adquisición de datos son los que se muestran en la Figura Nº 32.

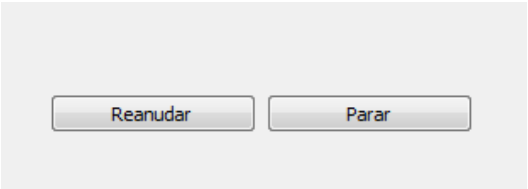

Figura Nº 32. Controles de la gráfica en tiempo real

Cabe destacar, que la señal que se grafica en tiempo real es una representación sub muestreada de la señal original, tomando 1 de cada 10 muestras.

4.4.1. Exportar gráfica

Debido a que la gráfica principal de esta interfaz, es una representación sub muestreada de la señal original, no es fiable para realizar ensayos y mediciones sobre la misma.

Para obtener la gráfica completa, con todos los datos correspondientes a un intervalo de tiempo de 3 segundos, se utiliza el botón “Exportar” contenido en la caja “Exportar gráfica” mostrado en la Figura Nº 33.

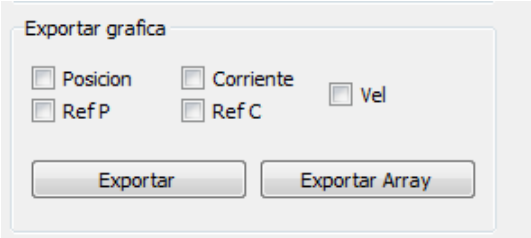

Figura Nº 33. Opciones de exportación de datos

Cada una de los botones seleccionables permiten elegir cada una de las señales a graficar. Una vez que se presiona el botón “Exportar”, el programa detiene la adquisición de datos y la actualización de la gráfica principal, para abrir otra ventana que contiene la gráfica completa de las señales seleccionadas.

Debido a que el rendimiento no es un factor limitante, se utiliza la librería “MatPlotLib” para obtenerla. Lo que permite obtener un resultado más profesional además de ofrecer distintos formatos para guardar la imagen generada.

El botón llamado “Exportar Array”, genera un archivo de Python que contiene los valores de las señales en forma numérica, de modo tal que se puedan procesar y analizar detalladamente por un programa externo.

4.5. Configuración

En la barra de herramientas de la ventana se encuentra una pestaña llamada configuración, que al hacer clic abre una ventana aparte, la cual es mostrada en la Figura Nº 34.

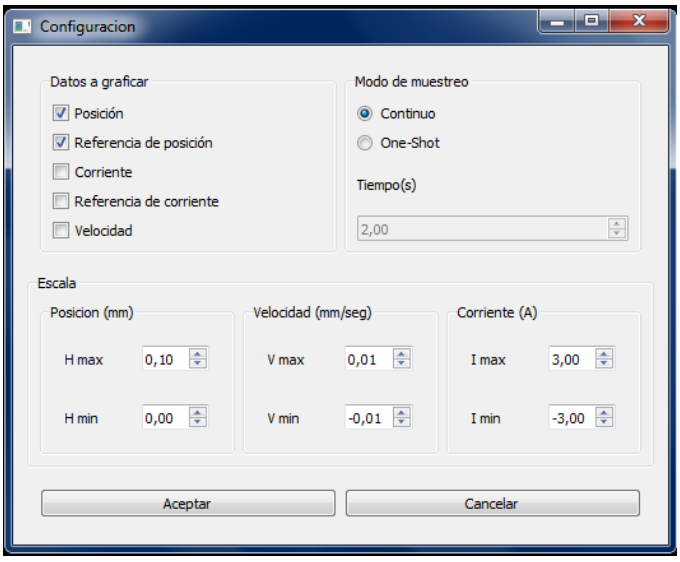

Figura Nº 34. Ventana de configuración

Esta ventana permite al usuario seleccionar cada una de las señales a graficar y ajustar la escala de los ejes “y” de cada una. Además, permite la selección del modo de funcionamiento de la interfaz gráfica. Esta cuenta con dos modos, el continuo, que es el descrito hasta el momento, y el modo de disparo simple (One-Shot).

Este último, fue pensado para realizar ensayos de la respuesta al escalón del levitador y se describe en la sección siguiente.

4.6. Modo One-Shot

El modo One-shot fue pensado para realizar ensayos de la respuesta a una entrada escalón en el levitador, de modo tal que se asegure una buena presentación de datos en la gráfica. Para poder utilizarlo, se debe entrar en la pestaña de configuración y seleccionarlo. Una vez seleccionado se puede ajustar la duración total de cada ensayo, hasta un máximo de 3 segundos.

Una vez guardados los cambios en la configuración se realizan las siguientes modificaciones en la ventana principal, mostrada en la Figura Nº 35.

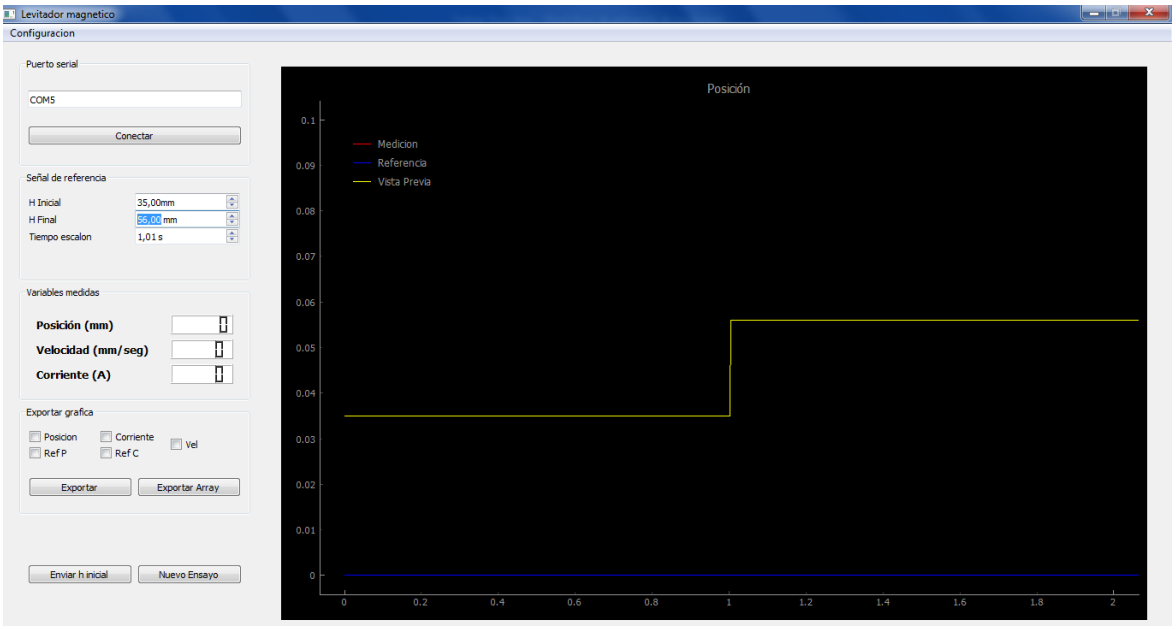

Figura Nº 35. Interfaz con el modo One-Shot configurado

En el recuadro de control de referencia, desaparecen los botones para seleccionar el tipo de señal, y cambian los nombres de las cajas numéricas. Permitiendo ahora, ajustar el valor inicial del escalón, el valor final y el tiempo del mismo.

En la gráfica de la posición, aparece una nueva señal, en amarillo, llamada “Vista previa”, que permite al usuario visualizar en tiempo real como es la entrada que se aplicara en el ensayo.

Para enviar la referencia del valor inicial del escalón, el usuario debe hacer clic en el botón “Enviar h inicial”. Este botón tiene la función de establecer el valor inicial en el microcontrolador, bloquear el ingreso de parámetros de la señal y reanudar tanto la adquisición de datos como la actualización de la gráfica. Una vez finalizado este proceso, el mismo botón cambia de nombre a “Iniciar ensayo”. El objetivo principal de este proceso es poder lograr la estabilización de la esfera en la referencia inicial.

Una vez que el usuario hace clic en “Iniciar ensayo” se da comienzo al experimento con dos temporizadores. El primero, una vez transcurrido el tiempo de escalón, envía el valor final del mismo al microcontrolador. El otro temporizador es el que detiene la actualización y adquisición de datos una vez se alcanza el tiempo de ensayo configurado.

CAPITULO 6: Resultados experimentales

En este capítulo se evalúa el desempeño del controlador y de la interfaz gráfica mediante ensayos experimentales.

1. Parámetros del ensayo

Los parámetros utilizados para realizar los ensayos subsecuentes se encuentran en la Tabla Nº 5.

Tabla Nº 5. Parámetros de los ensayos

Parámetro	Valor
$\boldsymbol{Vcc}$	$12V$
$\boldsymbol{R}$	3.8Ω
$\boldsymbol{L}$	$32mH$
$\boldsymbol{m}$	$39g$
$\boldsymbol{i_m}$	$0.903A$
$\boldsymbol{k}$	$0.004H.m$
$\boldsymbol{i(0)}$	$0.618A$
$\boldsymbol{h(0)}$	$35mm$

La esfera seleccionada de entre las disponibles, es un imán esférico con los valores de características i_m, k y m de la anterior tabla.

La disposición general de los equipos utilizados para realizar los ensayos se muestra en la Figura Nº 36.

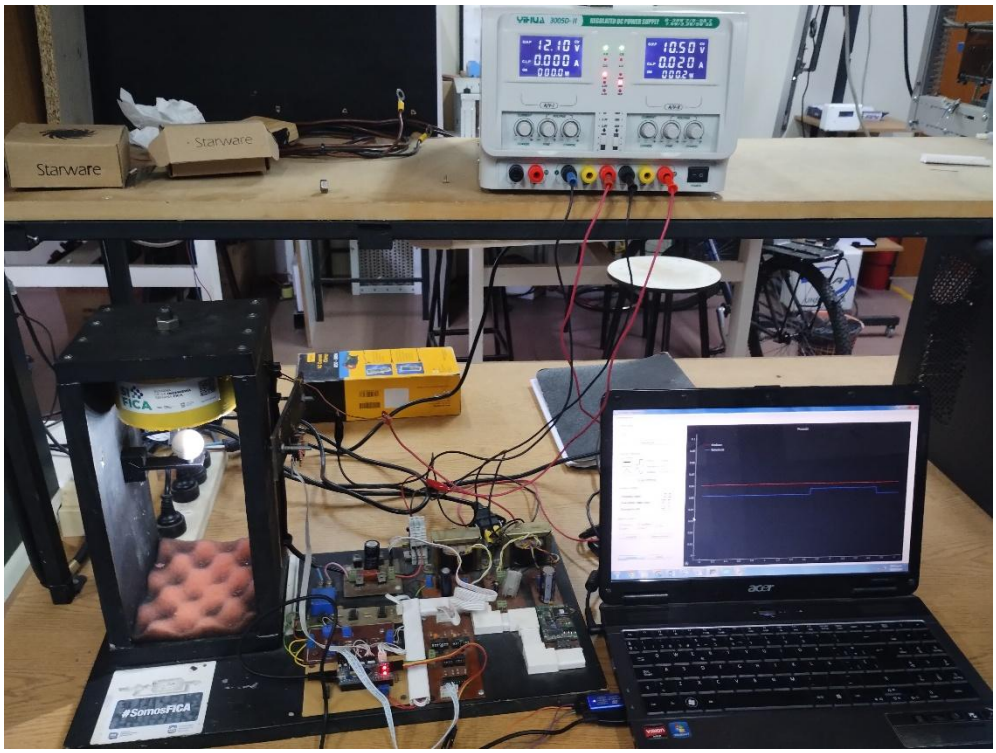

Figura Nº 36: Disposición general de los equipos necesarios para los ensayos

Todos los resultados de las pruebas realizadas fueron obtenidos mediante la interfaz gráfica de usuario diseñada anteriormente.

2. Estabilidad

En este ensayo se comprobó la estabilidad del sistema ante una referencia constante, para esto se colocó de forma manual a la esfera en el eje de la bobina, a una distancia de 30mm de la misma y se presionó el botón que reinicia el controlador. Luego de un corto tiempo se soltó la misma comprobando de manera visual el funcionamiento del controlador.

Para obtener el rango de operación del controlador, se decidió ir variando la señal de referencia, con pasos de 1mm, mediante la interfaz gráfica hasta que se indujera la inestabilidad en el sistema.

2.1. Límite inferior

Para obtener el límite inferior del rango de operación del levitador magnético se colocó la esfera a 30mm del electroimán y se reinició el controlador, luego se fue cambiando la referencia, en pasos de -1mm, con el recaudo de dejar el tiempo necesario para que el controlador estabilice el sistema entre cada cambio de referencia.

De este modo, el sistema alcanzó la inestabilidad con una referencia de 23mm. El momento en el que esto se produjo se muestra en la Figura Nº 37.

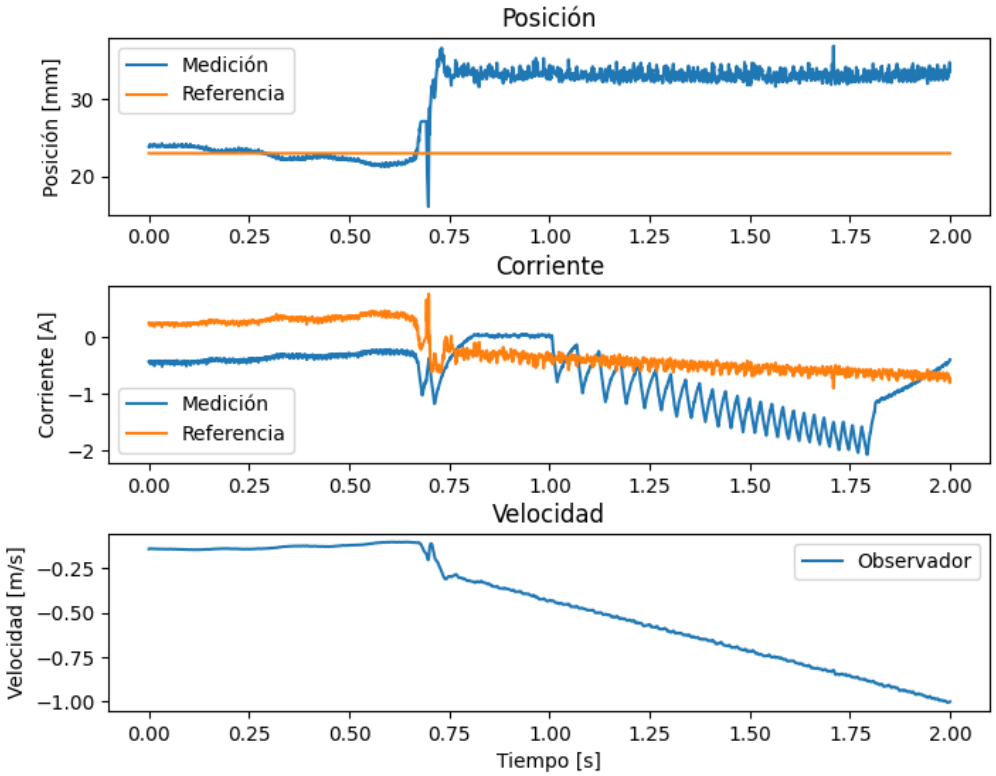

Figura Nº 37: Limite de estabilidad inferior

Este resultado se atribuye a que la forma cóncava del núcleo de la bobina induce oscilaciones perpendiculares al eje de la misma en la esfera. Esto provoca que esta se salga del campo de visión del sensor de posición y que este produzca valores erróneos, por ende, se induce la inestabilidad del sistema.

2.2. Límite superior

Para obtener el límite superior del rango de operación del levitador magnético se colocó la esfera a 45mm del electroimán y se reinició el controlador, luego de que esta alcanzara el estado estable, se fue cambiando la referencia, en pasos de 1mm, con el recaudo de dejar el tiempo necesario para que el controlador estabilice el sistema entre cada cambio de referencia.

De este modo, el sistema alcanzó la inestabilidad con una referencia de 51mm. El momento en el que esto se produjo se muestra en la Figura Nº 38.

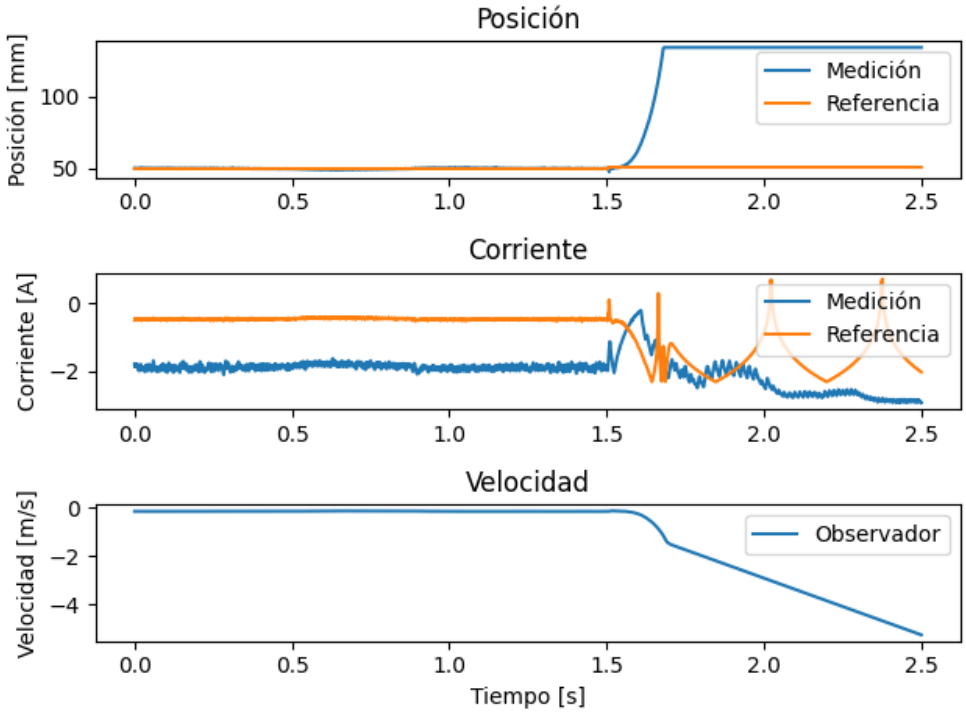

Figura Nº 38: Limite de estabilidad superior

Este resultado se atribuye a que el nivel de tensión requerido por el levitador para operar a esta distancia es incapaz de ser proporcionado por la fuente. Para solucionar esto se podría incrementar la tensión de la misma, sin embargo, esto no es recomendable, ya que el diseño de la etapa de potencia, y del levitador en general, se realizó para una tensión de alimentación de 12V [9]. Además de esto, la sobre exigencia del prototipo produce un rápido calentamiento en el inductor, provocando una variación paramétrica de la inductancia y resistencia que afectan de manera negativa el desempeño del controlador.

El rango de operación del electroimán entonces es de 28mm partiendo desde los 23mm hasta los 51mm, el punto medio de este se encuentra en 37mm.

2.3. Otras formas de inestabilidad

En los numerosos ensayos y pruebas realizados sobre el prototipo se detectaron varias formas adicionales de inducir la inestabilidad en el sistema. La primera fue que en posiciones cercanas al extremo inferior del rango de operación no es posible realizar escalones invertidos mayores a 3mm, por ejemplo, un cambio de referencia de 50mm a 46mm, dado que la tensión requerida excede a la proporcionada por la fuente. Otra forma en la que el sistema se vuelve inestable es ante movimientos en dirección normal al eje del electroimán, esto induce oscilaciones en la esfera que se ven amplificados en el tiempo. Esto se debe a que el sensor de posición arroja valores erróneos de medición cuando la esfera no se encuentra centrada sobre la línea de visión del mismo.

Por último, y la más desconcertante, es que la esfera, levitando en estado estable con una referencia constante, luego de un tiempo de algunos minutos comienza a “vibrar”. Esta vibración es un movimiento de rotación oscilatorio que se produce sobre un eje normal al eje del electroimán centrado sobre la esfera. Esta oscilación continua hasta que se invierte la orientación de la misma, produciendo un efecto de repulsión, debido a que el campo del electroimán es de la misma polaridad que el polo del imán, lo cual provoca que este sea proyectado hacia uno de los lados del levitador. Al momento de finalizar este trabajo no se encontró explicación alguna para este fenómeno.

3. Respuesta a una entrada escalón

Se ensayó el prototipo con un cambio de referencia en forma de escalón con una amplitud de 6mm centrado en el punto de operación de 35mm, es decir, la referencia cambia de 32mm a 38mm. Este punto fue seleccionado debido a que se encuentra lo suficientemente alejado de las zonas sensibles del prototipo, de esta manera se evitan los factores ajenos al diseño que pudieran incurrir en un detrimento del desempeño del controlador. Este punto de operación se mantiene para todos los ensayos realizados a no ser que se indique lo contrario. El resultado obtenido se muestra en la Figura Nº 39.

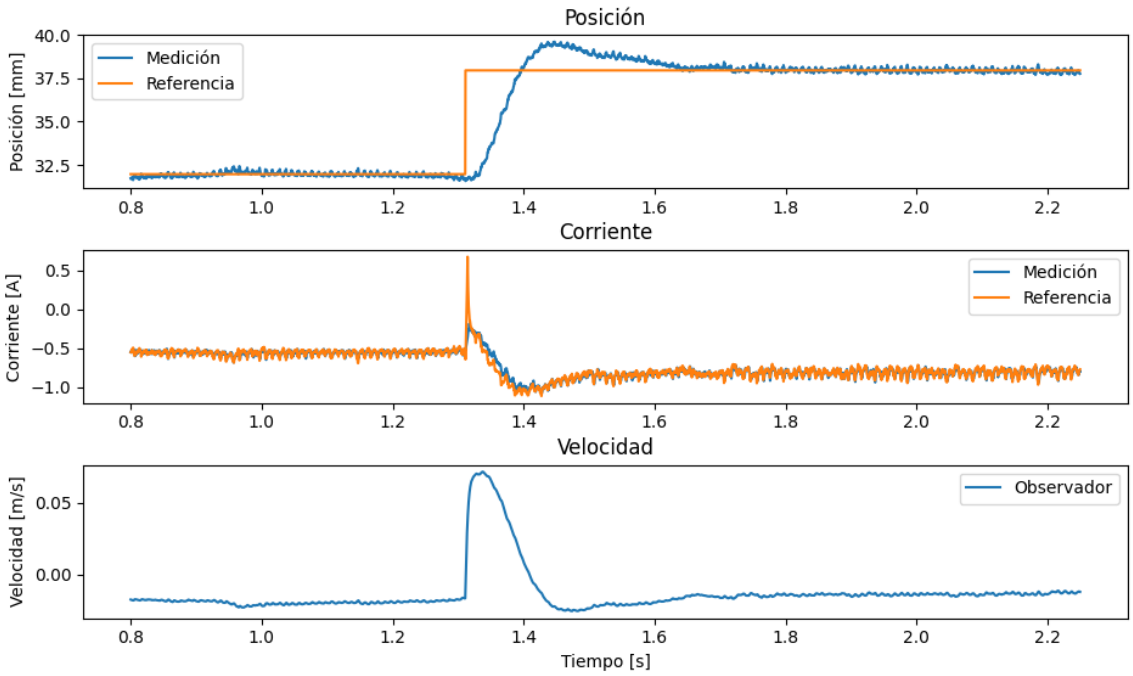

Figura Nº 39. Señales del prototipo ante un cambio de referencia escalón

En la primera gráfica se aprecian la posición y la referencia. Se observa que el sistema presenta un tiempo de asentamiento de aproximadamente 361ms con un sobrepaso máximo del 27%. Además, la esfera alcanza la referencia eliminando el error en estado estable. En la segunda gráfica se observa la corriente en la bobina y su referencia. En esta se observa que

el sistema sigue la referencia de manera casi perfecta, exceptuando el pico pronunciado en $t = 1.314s$.

En la última gráfica se observa la velocidad obtenida con el observador. Se aprecia que la señal no presenta grandes componentes de ruido, menos incluso que el ruido presente en la señal de posición.

3.1. Respuesta ante una entrada escalón negativo

Se ensayó el prototipo con un cambio de referencia en forma de escalón con una amplitud de -6mm. El resultado se muestra en la Figura Nº 40.

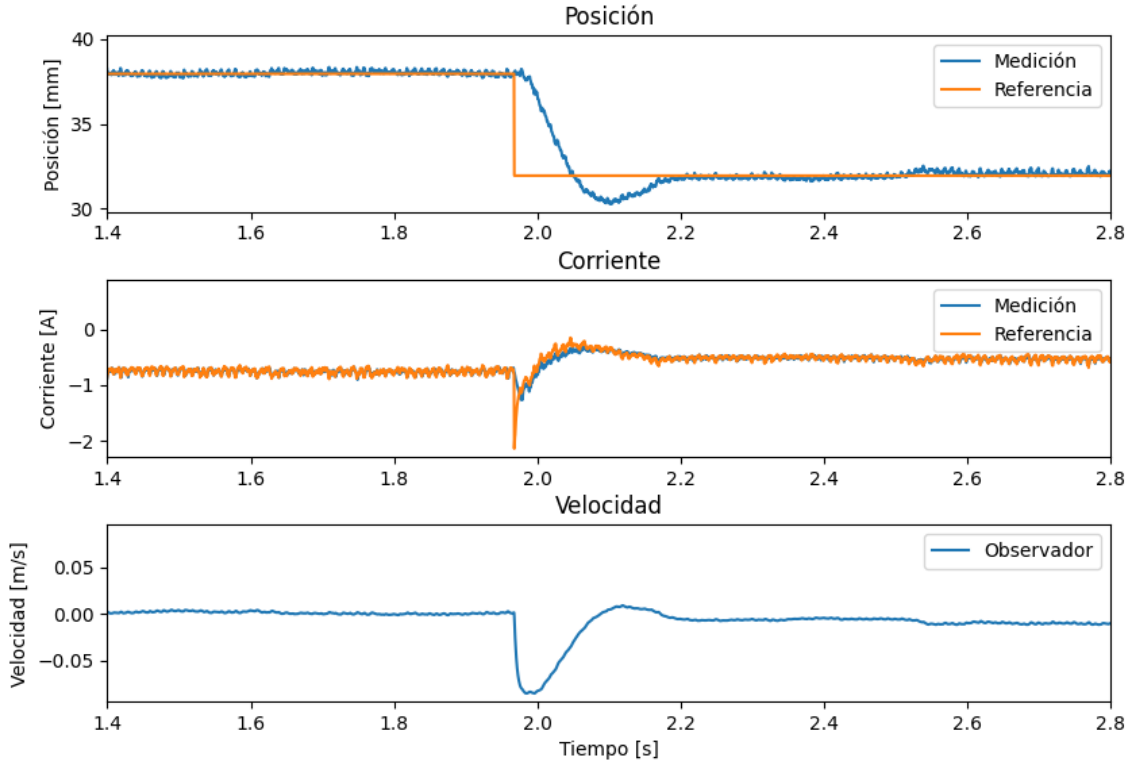

Figura Nº 40. Señales de salida del prototipo ante un escalón descendente.

En la primera gráfica se observa que el sistema presenta un tiempo de asentamiento de 268ms y un sobrepaso de aproximadamente 28%. No presenta error en estado estable.

De los ensayos se concluye que la respuesta tanto a un escalón positivo como negativo es simétrica en la forma. Sin embargo, la segunda presenta un menor tiempo de asentamiento.

4. Respuesta ante una onda cuadrada

Se decidió ensayar el prototipo con una entrada de referencia cuadrada con una amplitud de 3mm y una frecuencia de 1Hz. El resultado se muestra en la Figura Nº 41.

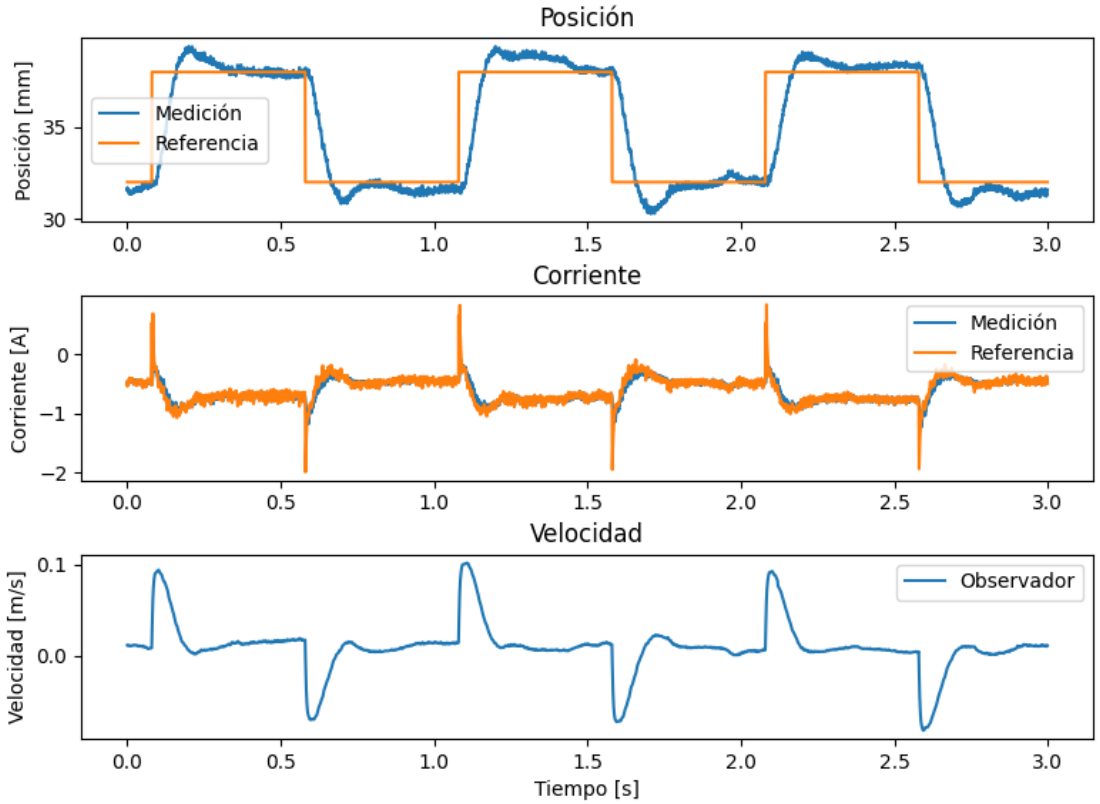

Figura Nº 41. Respuesta del sistema ante una referencia cuadrada

En la gráfica se observa que el sistema es capaz de seguir la señal de referencia sin volverse inestable.

5. Respuesta ante una señal triangular

Se decidió ensayar el prototipo con una entrada de referencia triangular con una amplitud de 3mm y una frecuencia de 1Hz. Los resultados del ensayo se muestran en la Figura Nº 42.

En la gráfica se observa que el sistema presenta sobrepaso cuando se produce el cambio de la pendiente en la referencia. También se observa que las señales de corriente presentan más ruido que en los ensayos anteriores, sin embargo, el sistema es capaz de seguir la señal de referencia sin volverse inestable.

La velocidad calculada con el observador debería asemejarse a una onda cuadrada, sin embargo, esto no sucede, lo cual se atribuye a la distorsión de la forma de onda de la posición.

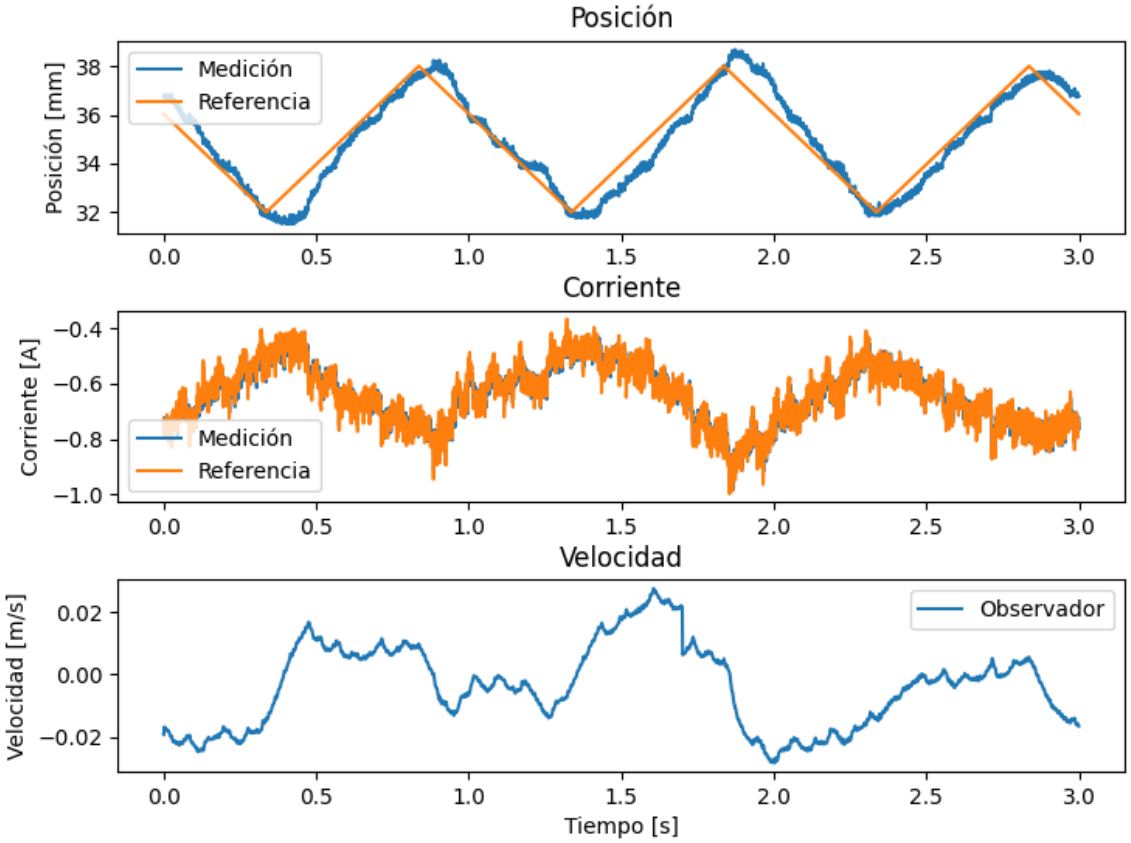

Figura Nº 42. Respuesta del sistema ante una entrada de referencia triangular

6. Respuesta ante una señal sinusoidal

Se realizó el ensayo del prototipo con una referencia sinusoidal de 3mm de amplitud, un offset de 35mm y una frecuencia de 1Hz. El resultado obtenido se muestra en la Figura Nº 43.

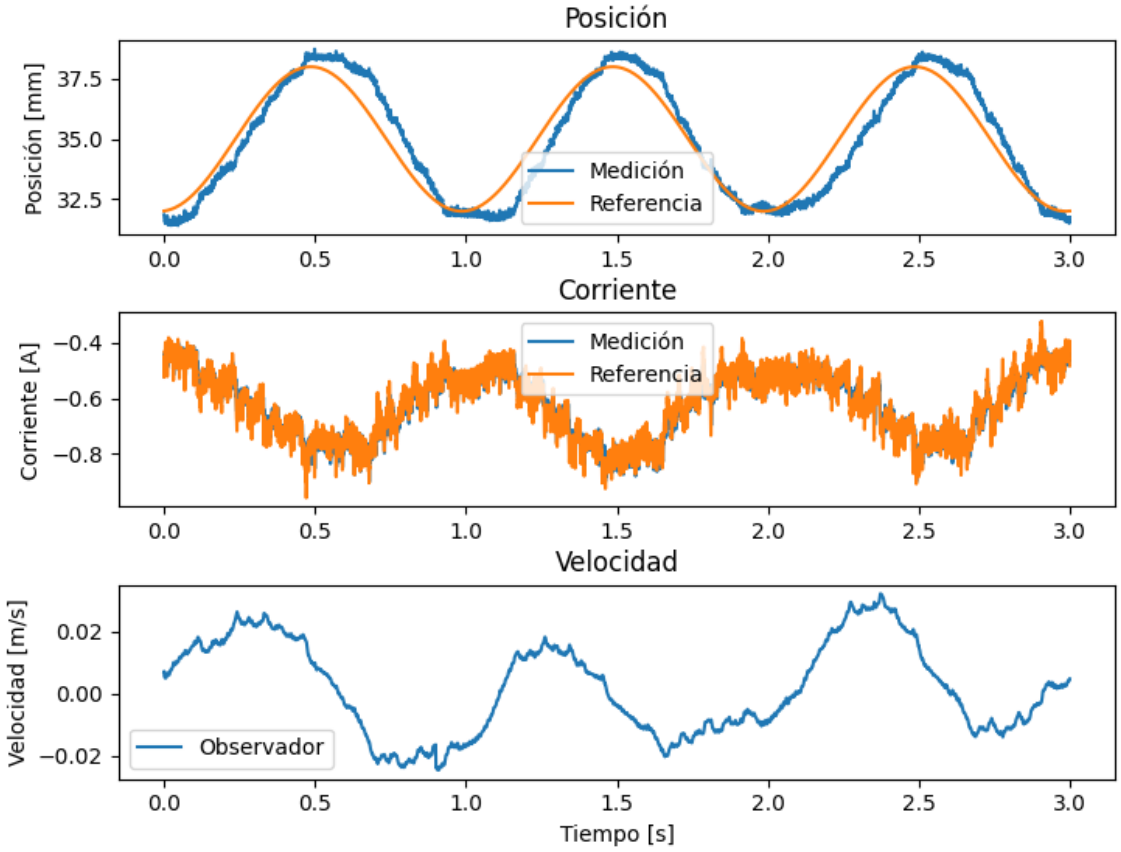

Figura Nº 43. Respuesta del sistema ante una señal de entrada sinusoidal

En la gráfica se observa que la señal de posición sigue a la de referencia con un leve desfasaje, también se ve un ligero sobrepaso. La gráfica de velocidad se asemeja a una onda sinusoidal desfasada 90º con respecto a la posición, resultado el cual es el esperado. La gráfica de la corriente es una onda sinusoidal distorsionada ya que el valor que se controla es el cuadrado de la misma.

7. Desempeño de la interfaz

Para analizar el desempeño de la interfaz y verificar que la actualización de los datos se realice en tiempo real, se midió el tiempo de barrido de las señales utilizando un cronometro centesimal. En total se dejaron pasar 10 periodos de barrido, lo cual arrojó un resultado de 30.24 segundos. Lo cual significa que el tiempo de actualización es de 3.02s.

También se realizaron mediciones de tiempo de ejecución en las funciones de actualización de la gráfica y recepción de datos. Estas fueron obtenidas con el modulo “time” nativo de Python, promediando 100 ejecuciones.

Los resultados de tiempo de ejecución para la actualización de la gráfica en tiempo real fueron de 3.2 ms graficando las 5 señales al mismo tiempo (peor caso). Para el tiempo de muestreo el resultado fue de 19.99 ms.

8. Comparación punto fijo vs flotante

Como se mencionó en el Capítulo 4, se implementó el algoritmo de control tanto en punto flotante como en punto fijo. En este ensayo se midió el tiempo de ejecución de cada uno.

Para realizar esto, se modificó el código implementado en el microcontrolador para medir el tiempo de ejecución con el temporizador 4, y enviar los resultados a la interfaz gráfica en lugar de la posición. Estos resultados fueron exportados con el fin de poder graficar las dos señales juntas, tal como se muestra en la Figura Nº 44

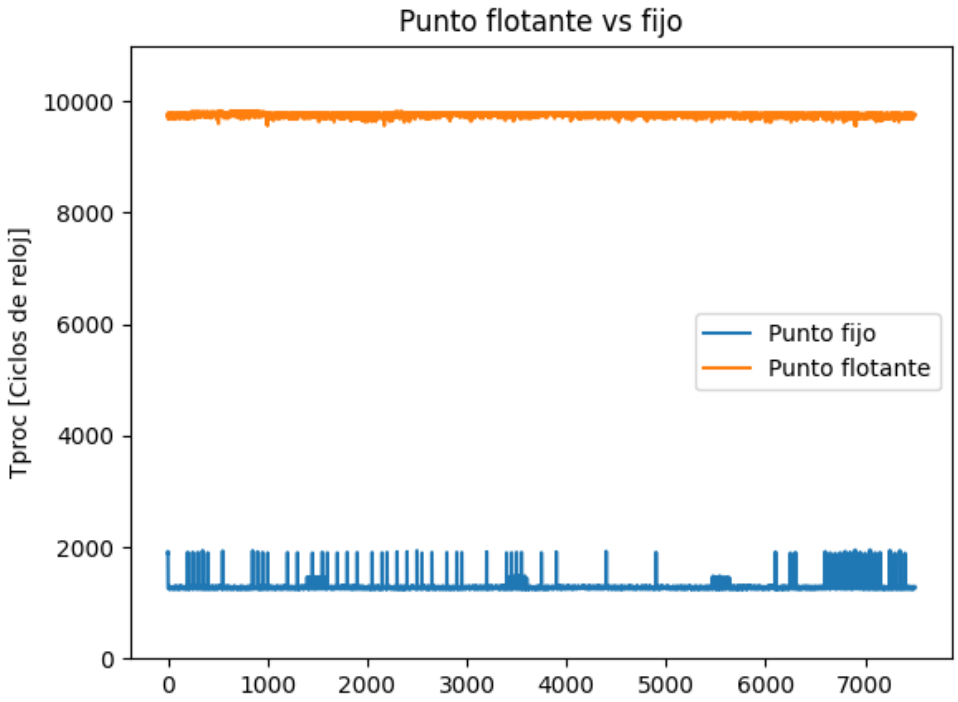

Figura Nº 44. Comparación del tiempo de procesamiento entre el algoritmo de punto fijo y el de punto flotante

En esta se observa la gran diferencia en los tiempos de procesamiento entre un algoritmo y otro. Se calculó el tiempo promedio con los datos de la gráfica, los resultados se muestran en la Tabla Nº 6.

Tabla Nº 6: Tiempos de ejecución punto fijo vs punto flotante

Algoritmo	Promedio [Ciclos]	Promedio [μs]
Punto fijo	1271	17.65
Punto flotante	9766	135.64

De los resultados obtenidos se infiere que el algoritmo de punto fijo requiere el 13% del tiempo de procesamiento, una reducción del 87% con respecto al de punto flotante.

9. Levitador en funcionamiento

En la Figura Nº 45 se muestra dos esferas levitando a una distancia de 35mm con respecto al electroimán.

Figura Nº 45. Levitador en funcionamiento

Por último, en la Figura Nº 46 se muestra una captura de pantalla de la gráfica en tiempo real con las esferas levitando. La señal de referencia es una onda cuadrada de 3mm de amplitud y 0.5Hz de frecuencia, con 35mm de offset.

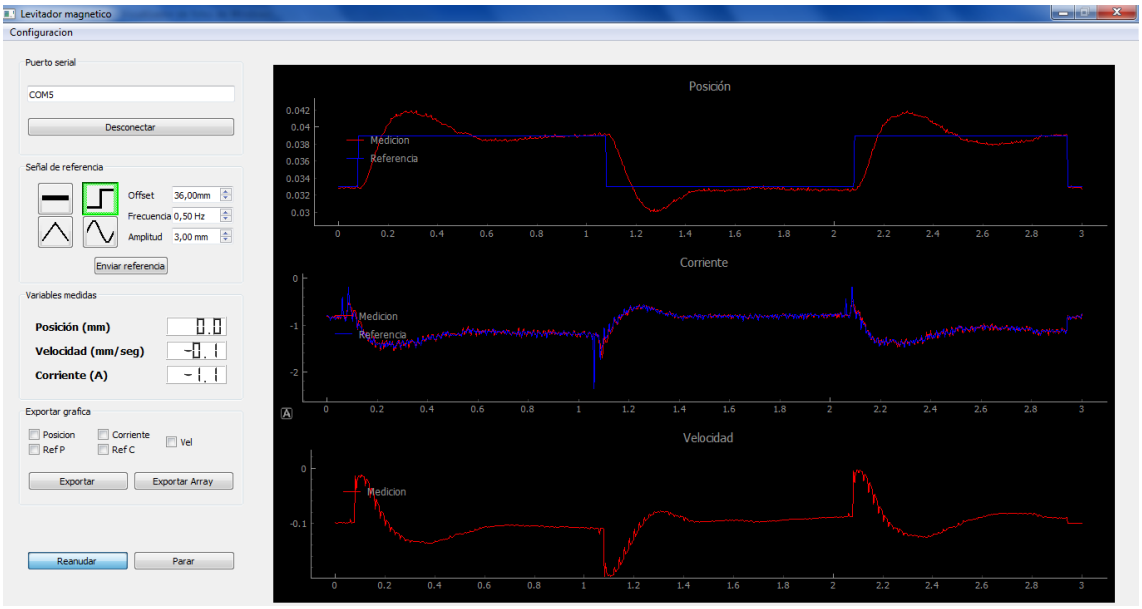

Figura Nº 46. Interfaz gráfica en funcionamiento

CAPITULO 7: Conclusiones

En este trabajo se llevó a cabo el diseño y la implementación de un controlador de posición no lineal para un levitador magnético por la técnica de linealización por realimentación de estados. Esta técnica resultó en un sistema estable y robusto ante cambios de referencia. Además, se obtuvo un rango de operación de 28 mm, desde los 23mm hasta los 51mm. El límite inferior de este rango viene dado por la forma cóncava del núcleo del electroimán, que, ante cualquier perturbación en dirección perpendicular al eje del mismo, introduce oscilaciones perpendiculares que ocasionan la inestabilidad del sistema. El límite superior se encuentra dado por la cantidad de corriente que puede entregar la fuente, que es cercana a la máxima para la que el electroimán fue diseñado.

La interfaz gráfica resultó de mucha utilidad para obtener los resultados experimentales e interactuar de forma sencilla con el prototipo de levitador magnético. Tiene el problema de que la gráfica, tanto de corriente como de velocidad, presentan ruido en las primeras dos muestras que se reciben del microcontrolador, este defecto se produce en la comunicación y no en el controlador, se desconoce su causa.

Por último, la diferencia entre el tiempo de ejecución de un algoritmo de punto fijo y uno de punto flotante es mayor a la esperada. Por lo que los algoritmos de punto fijo deben ser la opción principal para implementar controladores en microcontroladores sin unidad de punto flotante, para mejorar el rendimiento del mismo.

Como trabajo futuro queda propuesto investigar más a fondo los fenómenos que causan la inestabilidad en el sistema y realizar la comparación de los rangos de operación de este controlador y uno clásico lineal.

BIBLIOGRAFÍA

[1] Hyung-Woo Lee, Ki-Chan Kim and Ju Lee, "Review of maglev train technologies," in IEEE Transactions on Magnetics, vol. 42, no. 7, pp. 1917-1925, July 2006, doi: 10.1109/TMAG.2006.875842.

[2] Shanghai MagLev Transportation Development Co., Ltd. “Shanghai Maglev Official Website”. Acceso: Nov. 6, 2024. [Online.] Disponible: http://www.smtdc.com/en/gycf3.html

[3] J.C. Moreno. “Control Lineal y No Lineal de un Levitador Magnético”. Universidad Politécnica de Catalunya, Barcelona, España. 2010.

[4] Karakuzu, B., Anil İnevi, M., Tarim, E.A. *et al.* Magnetic levitation-based miniaturized technologies for advanced diagnostics. *emergent mater.* (2024). https://doi.org/10.1007/s42247-024-00762-6

[5] Dabbagh, S.R., Alseed, M.M., Saadat, M., Sitti, M. and Tasoglu, S. (2022), Biomedical Applications of Magnetic Levitation. Adv. NanoBiomed Res., 2: 2100103. https://doi.org/10.1002/anbr.202100103

[6] K. Ogata, “Sistemas de control digital en tiempo discreto” (2da Edición, cap. 6). Prentice Hall Hispanoamericana, México, 1996.

[7] B. C. Kuo, “Sistemas de control Digital” (1ra Edición, cap. 3). Compañía Editorial Continental S.A. México, 1997.

[8] K. Ogata, “Sistemas de control digital en tiempo discreto” (2da Edición, cap. 4). Prentice Hall Hispanoamericana, México, 1996.

[9] L.E. Venghi, Implementación y Control de un Levitador Magnético. Laboratorio de Control Automático (LCA). Facultad de Ingeniería y Ciencias Agropecuarias, Universidad Nacional de San Luis. Villa Mercedes, San Luis, Argentina. 2015.

[10] H. K. Khalil, “Nonlinear Systems” (3rd Edition, pp 505-540), Prentice Hall, Upper Saddle River, 2002.

[11] TSL1401 Datasheet, Texas Instruments. 1996.

[12] QT5.15 Documentation. QtGroup. 2024. https://doc.qt.io/qt-5/

ANEXO 1: Código fuente STM32

```
        /* USER CODE BEGIN Header */
        /**

******************************************************************************
***
          * @file           : main.c
          * @brief          : Main program body

******************************************************************************
***
          * @attention
          *
          * Copyright (c) 2024 STMicroelectronics.
          * All rights reserved.
          *
          * This software is licensed under terms that can be found in the
LICENSE file
          * in the root directory of this software component.
          * If no LICENSE file comes with this software, it is provided AS-
IS.
          *

******************************************************************************
***
          */
        /* USER CODE END Header */
        /* Includes ------------------------------------------------------
----------*/
        #include "main.h"
        #include "usb_device.h"

        /* Private includes ----------------------------------------------
----------*/
        /* USER CODE BEGIN Includes */
        #include "sinlut.h"
        #include "usbd_cdc_if.h"
        /* USER CODE END Includes */

        /* Private typedef -----------------------------------------------
----------*/
        /* USER CODE BEGIN PTD */

        /* USER CODE END PTD */

        /* Private define ------------------------------------------------
----------*/
        /* USER CODE BEGIN PD */
        /* Constantes utiles representadas en punto fijo*/
        #define FBITS  24
        #define FACTOR_ESC  16777216
        #define B_ENT  40
        #define  VI ((long long) 201326592 )
        #define  K ((long long) 6711 )
        #define  MASA ((long long) 645923 )
        #define  L ((long long) 536871 )
        #define  R ((long long) 63753421 )
        #define  IM ((long long) 15149826 )
        #define  G ((long long) 164584489 )
```

```
#define  HA ((long long) 454075 )
#define  HB ((long long) 1060921464 )
#define  IA ((long long) 38851 )
#define  IB ((long long) 80278979 )
#define  KH1 ((long long) 64781359147 )
#define  KH2 ((long long) 4243696124 )
#define  KH3 ((long long) 28190756 )
#define  KI1 ((long long) 2662993809 )
#define  KI2 ((long long) 11802771 )
#define  KO1 ((long long) 267479 )
#define  KO2 ((long long) 11802671 )
#define  G11 ((long long) 16777216 )
#define  G12 ((long long) 1678 )
#define  G21 ((long long) 0 )
#define  G22 ((long long) 16777216 )
#define  B0 ((long long) 0 )
#define  B1 ((long long) 1678 )
#define  IIR_FIL ((long long) 1677722 )
#define  ACH0 ((long long) 3271557120 )
#define  ACI0 ((long long) 251658 )
#define  ACI1 ((long long) 63753421 )

/* Variables globales del controlador */
long long  h;
long long  h_raw;
long long  hf;
long long  i;
long long  i_f;
long long  ref;
long long  n_muestra=0;
long long  eh_int =0;
long long  ei_int=0;
long long  uhc=0;
long long  acC=0;
long long  h_est=0;
long long  dh_est=0;
long long  Wh=0;
long long  Wi=0;
long long  eh=0;
long long  ei=0;
long long  i_io;
long long  eo=0;

/* Comunicacion con la interfaz */
#define N_SAMPLES 50
#define L_SAMPLES 4
#define OFF_BUF N_SAMPLES*L_SAMPLES
#define BUF_LEN (OFF_BUF*5+1)
#define SS_FACTOR 4
/* USER CODE END PD */

/* Private macro -------------------------------------------------
----------*/
/* USER CODE BEGIN PM */
#define fpm(a,b) (((((long long)a)*(b))>>FBITS))
#define fpd(a,b) ((((long long)a<<FBITS)/(b)))
/* USER CODE END PM */

/* Private variables ---------------------------------------------
----------*/
ADC_HandleTypeDef hadc1;
```

```
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;

/* USER CODE BEGIN PV */
/* Variables de conexion a la interfaz*/
uint8_t send_buffer[BUF_LEN];
RTR_t rec_buffer;

typedef union{
     long long*sts[5];
     uint8_t* sts_s[5];
}STS_t;
STS_t Sen_Int;
uint8_t ss_counter=0;
uint8_t buf_ind=0;
uint8_t capturas=0;
uint8_t demo_mode=0;
long long debug_t=0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------
----------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------
----------*/
/* USER CODE BEGIN 0 */
/* Copiar las variables al buffer descartando valores grandes*/
void copy(int index){
     for (int k=0; k<4;k++){
          send_buffer[k+index]=*(Sen_Int.sts_s[0]+k);  //src:  ff  ff
ff ff ff .ff ff ff -> dst: [ff .ff ff ff]
          send_buffer[OFF_BUF+k+index]=*(Sen_Int.sts_s[1]+k);
          send_buffer[OFF_BUF*2+k+index]=*(Sen_Int.sts_s[2]+k);
          send_buffer[OFF_BUF*3+k+index]=*(Sen_Int.sts_s[3]+k);
          send_buffer[OFF_BUF*4+k+index]=*(Sen_Int.sts_s[4]+k);
     }
}


struct REF_status{
     int ref_sel;
     long long  offset;
     long long  amp;
     int  periodo;
}typedef REF_status;
static REF_status RefCtl;
void (*REF_pointer[4])(void);
```

```
void data_received(void){
	if(demo_mode==0){
		RefCtl.ref_sel = rec_buffer.data[0];
		RefCtl.offset = (long long)rec_buffer.data[1];
		RefCtl.amp = (long long)rec_buffer.data[2];
		RefCtl.periodo=rec_buffer.data[3];
	}
}
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
	if(TIM3->CCR1>2900 && capturas==0){
		h_raw=TIM3->CCR1;
		capturas=1;
	}
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){
	// Adquisición de datos
	i_f=(HAL_ADC_GetValue(&hadc1)*IA)-IB;
	h = ((h_raw*HA) - HB)/1000;

	capturas=0; //Control de sobrecaptura

	// Filtrado
	i=fpm(IIR_FIL , ( i_f - i )) + i;
	hf=fpm(IIR_FIL , ( h - hf )) + hf;

	// Calcular la referencia
	REF_pointer[RefCtl.ref_sel]();
	n_muestra= (n_muestra+1)%(RefCtl.periodo*2);

	// CONTROL DE POSICION //
	eh=(ref-hf);
	eh_int+=eh;
	Wh=fpm(KH1,eh)-fpm(KH2,dh_est)+fpm(KH3,eh_int);
	uhc=fpm(fpm(G-Wh,hf),fpm(hf,ACH0));

	// CONTROL DE CORRIENTE //
	i_io=i-IM;
	ei=uhc-fpm(i_io,i_io);
	ei_int+=ei;
	Wi=(fpm(KI1,ei)+fpm(KI2,ei_int));
	acC=(fpd(fpm(Wi,ACI0),i_io)+fpm(ACI1,i));

	TIM1->CCR1 =((fpd(acC,VI)+FACTOR_ESC)*3600)>>FBITS;

	// OBSERVADOR
	eo=hf-h_est;
	long long dh_tmp=dh_est+fpm(B1,Wh)+fpm(KO2,eo);
	h_est = h_est+fpm(G12,dh_est)+fpm(Wh,B0)+fpm(KO1,eo);
	dh_est=dh_tmp;

	// Guardar los datos para enviar en el buffer
	if(ss_counter==SS_FACTOR-1){
		uint16_t index=(buf_ind*4+1);
		send_buffer[0]=buf_ind;
		copy(index);
		buf_ind=(buf_ind+1)%N_SAMPLES;
		ss_counter=0;
	}else{
		ss_counter++;
	}
```

```
        if(buf_ind==0){
              CDC_Transmit_FS(&send_buffer[0], (uint16_t)BUF_LEN);
        }
        //HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13,0);
  }

  /* Funcion para reiniciar el controlador */
  void reset_ctrl(void){
        RefCtl.ref_sel=0;
        RefCtl.offset=h;
        hf=h;
        h_est=hf; dh_est=0;
        ei_int=0; eh_int=0;
  }

  /* Modo demostracion */
  void demo_mode_fun(void){
        // Desactiva la recepcion de referencias desde la GUI
        demo_mode=1;
        HAL_Delay(100);
        reset_ctrl();

        // Periodo de 1 seg
        RefCtl.periodo=10000;
        // Amplitud de 3mm. Total 6mm
        RefCtl.amp=3*(FACTOR_ESC/1000);
        long long h_orig=hf;
        HAL_Delay(2000);

        // Levanta la esfera
        while (RefCtl.offset>(FACTOR_ESC/1000)*35){
              RefCtl.offset-=FACTOR_ESC/10000;
              HAL_Delay(100);
        }

        HAL_Delay(1000);
        // Pasa por todas las referencias
        for(int rs=1;rs<4;rs++){
              RefCtl.ref_sel=rs;
              int periodos_transcurridos=0;
              n_muestra=0;
              while(periodos_transcurridos<10){

if(n_muestra==19999){periodos_transcurridos++;HAL_Delay(5);}
                    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
              }
              RefCtl.ref_sel=0;
              HAL_Delay(500);
        }
        // Devuelve la esfera a su posicion original
        RefCtl.ref_sel=0;
        while (RefCtl.offset<h_orig){
              RefCtl.offset+=FACTOR_ESC/10000;
              HAL_Delay(100);
        }
        demo_mode=0;
  }

  // Generacion de referencia
  void REF_cte(void){ref=RefCtl.offset;}
  void REF_cuad(void){
```

```
            ref= (n_muestra > RefCtl.periodo) ? RefCtl.offset+RefCtl.amp :
RefCtl.offset-RefCtl.amp;
        }
        void REF_tri(void){
            ref= (n_muestra > RefCtl.periodo) ?
                        (RefCtl.offset-
fpm(RefCtl.amp,(2*fpd(n_muestra,RefCtl.periodo)-((long long)3<<FBITS)))) :

      RefCtl.offset+fpm(RefCtl.amp,(2*fpd(n_muestra,RefCtl.periodo)-((long
long)1<<FBITS)));
        }
        void
REF_sen(void){ref=RefCtl.offset+fpm(RefCtl.amp,calc_sin((int)((5000*n_muest
ra)/RefCtl.periodo)));}
        void (*REF_pointer[4])(void)={REF_cte,REF_cuad,REF_tri,REF_sen};

        /* USER CODE END 0 */

        /**
          * @brief  The application entry point.
          * @retval int
          */
        int main(void)
        {
          /* USER CODE BEGIN 1 */

          /* USER CODE END 1 */

          /* MCU Configuration--------------------------------------------
----------*/

          /* Reset of all peripherals, Initializes the Flash interface and
the Systick. */
          HAL_Init();

          /* USER CODE BEGIN Init */

          /* USER CODE END Init */

          /* Configure the system clock */
          SystemClock_Config();

          /* USER CODE BEGIN SysInit */
          /* Señales a mandar */
          Sen_Int.sts[0]=&hf;
          Sen_Int.sts[1]=&ref;
          Sen_Int.sts[2]=&i;
          Sen_Int.sts[3]=&uhc;
          Sen_Int.sts[4]=&dh_est;
          /* USER CODE END SysInit */

          /* Initialize all configured peripherals */
          MX_GPIO_Init();
          MX_ADC1_Init();
          MX_TIM1_Init();
          MX_TIM2_Init();
          MX_TIM3_Init();
          MX_USB_DEVICE_Init();
          MX_TIM4_Init();
          /* USER CODE BEGIN 2 */
          HAL_ADC_Start_IT(&hadc1);
```

```
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
  HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);
  HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  RefCtl.periodo=5000;
  RefCtl.offset=35<<FBITS;
  RefCtl.ref_sel=0;
  while (1)
  {
       if(HAL_GPIO_ReadPin(BTN_INF_GPIO_Port, BTN_INF_Pin)){
             reset_ctrl();
             HAL_Delay(100);
       }
       if(HAL_GPIO_ReadPin(BTN_SUP_GPIO_Port, BTN_SUP_Pin)){
             RefCtl.offset+=(FACTOR_ESC/1000)*5;
             HAL_Delay(500);
             RefCtl.offset-=(FACTOR_ESC/1000)*5;
             HAL_Delay(500);
             if(HAL_GPIO_ReadPin(BTN_SUP_GPIO_Port, BTN_SUP_Pin)){
                   HAL_ADC_Stop_IT(&hadc1);
                   TIM1->CCR1=3600;
                   for(int k=0; k<10;k++){
                         HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
                         HAL_Delay(100);
                   }
                   // MODO SUSPENDER
                   while(!HAL_GPIO_ReadPin(BTN_INF_GPIO_Port,
BTN_INF_Pin)){
                         HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
                         HAL_Delay(250);
                         // Modo DEMO
                         if(HAL_GPIO_ReadPin(BTN_SUP_GPIO_Port,
BTN_SUP_Pin)){
                               HAL_ADC_Start_IT(&hadc1);
                               demo_mode_fun();
                               HAL_ADC_Stop_IT(&hadc1);
                               TIM1->CCR1=3600;
                         }
                   }
                   HAL_ADC_Start_IT(&hadc1);
             }
       }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
```

```
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType                                    =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if  (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,  FLASH_LATENCY_2)  !=
HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection                             =
RCC_PERIPHCLK_ADC|RCC_PERIPHCLK_USB;
  PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
  PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL_DIV1_5;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};
```

```
  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */

  /** Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T3_TRGO;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 7199;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
```

```
          {
            Error_Handler();
          }
          sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
          if   (HAL_TIM_ConfigClockSource(&htim1,   &sClockSourceConfig)   !=
HAL_OK)
          {
            Error_Handler();
          }
          if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
          {
            Error_Handler();
          }
          sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
          sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
          if  (HAL_TIMEx_MasterConfigSynchronization(&htim1,  &sMasterConfig)
!= HAL_OK)
          {
            Error_Handler();
          }
          sConfigOC.OCMode = TIM_OCMODE_PWM1;
          sConfigOC.Pulse = 3600;
          sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
          sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
          sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
          sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
          sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
          if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
          {
            Error_Handler();
          }
          sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
          sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
          sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
          sBreakDeadTimeConfig.DeadTime = 36;
          sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
          sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
          sBreakDeadTimeConfig.AutomaticOutput                                =
TIM_AUTOMATICOUTPUT_DISABLE;
          if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) !=
HAL_OK)
          {
            Error_Handler();
          }
          /* USER CODE BEGIN TIM1_Init 2 */

          /* USER CODE END TIM1_Init 2 */
          HAL_TIM_MspPostInit(&htim1);

        }

        /**
          * @brief TIM2 Initialization Function
          * @param None
          * @retval None
          */
        static void MX_TIM2_Init(void)
        {

          /* USER CODE BEGIN TIM2_Init 0 */
```

```
  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 71;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if   (HAL_TIM_ConfigClockSource(&htim2,   &sClockSourceConfig)   !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if  (HAL_TIMEx_MasterConfigSynchronization(&htim2,  &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 35;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim2, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */
  HAL_TIM_MspPostInit(&htim2);

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{
```

```
  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_IC_InitTypeDef sConfigIC = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 0;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 7199;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_IC_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if  (HAL_TIMEx_MasterConfigSynchronization(&htim3,  &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 15;
  if (HAL_TIM_IC_ConfigChannel(&htim3, &sConfigIC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 72;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  HAL_TIM_MspPostInit(&htim3);

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
```

```
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 0;
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 65535;
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if   (HAL_TIM_ConfigClockSource(&htim4,   &sClockSourceConfig)   !=
HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if  (HAL_TIMEx_MasterConfigSynchronization(&htim4,  &sMasterConfig)
!= HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
```

```
  /*Configure GPIO pin : PC13 */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : BTN SUP Pin BTN INF Pin */
  GPIO_InitStruct.Pin = BTN_SUP_Pin|BTN_INF_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error
return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line
number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and
line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n",
file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```