UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Diseñar y desarrollar un controlador embebido basado en PID aplicado a la regulación de fuerza de una mano robótica.

TRABAJO DE TITULACIÓN.

AUTORES: Banegas Rojas, Jandry Óscar

Sarmiento Sotomayor, Santiago René

DIRECTOR: Calderón Córdova, Carlos Alberto, Ing.

LOJA - ECUADOR

2017

2017

APROBACIÓN DE LA DIRECTORIA DEL TRABAJO DE TITULACIÓN

Ingeniero

Carlos Alberto Calderón Córdova

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: Diseñar y desarrollar un controlador embebido basado en PID aplicado a la regulación de fuerza de una mano robótica, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, diciembre de 2017

f) …….

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Nosotros Banegas Rojas Jandry Óscar, Sarmiento Sotomayor Santiago René declaramos ser autores del presente trabajo de titulación: Diseñar y desarrollar un controlador embebido basado en PID aplicado a la regulación de fuerza de una mano robótica, de la Titulación de Ingeniería en Electrónica y Telecomunicaciones, siendo el Ing. Carlos Alberto Calderón Córdova director del presente trabajo; y eximamos expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaramos conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad”

f. .....

Autor: Jandry Óscar Banegas Rojas

Cédula: 1105661886

f. .....

Autor: Santiago René Sarmiento Sotomayor

Cédula: 1104745862

DEDICATORIA

A Dios, por darme las posibilidades y fortalezas para llegar a este momento tan especial. A mis padres, que me han ayudado durante toda mi vida, con sus consejos, con su demostración de padres ejemplares que me enseñaron a no desfallecer en momentos difíciles. A toda mi familia y amigos que siempre me apoyaron en mi vida universitaria.

Santiago Sarmiento

A mis padres Olinda y Alfredo, por darme el apoyo económico y moral para afrontar los retos más difíciles y complejos que he afrontado a lo largo de mi vida, ya que nunca han perdido la fe en mí. A mi alter ego “Jandcan” que me ha mantenido presto ante toda situación. Y a toda la gente que confía y ha confiado en mí.

Jandry Banegas

AGRADECIMIENTO

A nuestros padres, por darnos todo el sustento, consejos y motivación para no desfallecer en este camino que es parte fundamental en nuestras vidas.

Al equipo de trabajo del proyecto Hand of Hope, un equipo con el cual la frase: “Si vas en grupo llegas lejos” se confirma, ya que nos ha llevado a lograr todos nuestros objetivos.

A nuestro director, Ing. Carlos Calderón por confiar en nosotros durante todos los trabajos que hemos ejecutado y en la realización de este proyecto.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo es parte del proyecto de investigación “Mano Esperanza”, el objetivo principal es diseñar e implementar un controlador PID embebido, aplicado a la regulación de la fuerza de agarre de una mano robótica. Se busca diseñar, seleccionar e implementar el controlador de mejor desempeño tomando como base las metodologías utilizadas en aplicaciones similares. Además, se busca analizar y comparar experimentalmente el desempeño del controlador de fuerza con retroalimentación basada en medición de corriente consumida por cada actuador, y medición de fuerza ejercida sobre el sensor FSR.

Como resultados del presente trabajo: se diseñaron seis controladores para cada uno de los dos tipos de retroalimentación de fuerza de agarre, y finalmente se implementó el controlador en el sistema embebido para evaluar el desempeño experimental con el prototipo de prótesis robótica. Para el control de fuerza de agarre basado en medición de corriente, el controlador Lambda ofreció los mejores resultados ($T_s<0.85s$ y P.O.<2%). En el caso del control de fuerza basado en FSR se implementaron los métodos Lambda y Cohen-Coon, pero no se logró la estabilización en la fase experimental.

PALABRAS CLAVES: Prótesis robótica, Control de fuerza de agarre, Controlador PID, FSR, resistor Shunt, actuador lineal.

ABSTRACT

The present work is part of the "Mano Esperanza" research project, the main objective is to design and implement an embedded PID controller, applied to the regulation of the grip force of a robotic hand. The aim is to design, select and implement the best performance controller based on the methodologies used in similar applications. In addition, it is sought to analyze and experimentally compare the performance of the force controller with feedback based on current measurement consumed by each actuator, and force measurement exerted on the FSR sensor.

As results of the present work: six controllers were designed for each of the two types of grip force feedback, and finally the controller was implemented in the embedded system to evaluate the experimental performance with the prototype of robotic prosthesis. For grip force control based on current measurement, the Lambda controller offered the best results (Ts <0.85s and P.O. <2%). In the case of force control based on FSR, Lambda and Cohen-Coon methods were implemented, but stabilization was not achieved in the experimental phase.

KEYWORDS: Robotic prosthesis, grip force control, PID controller, FSR, Shunt resistor, linear actuator.

INTRODUCCIÓN

Según Consejo Nacional para la Igualdad de Discapacidades (CONADIS), hasta noviembre de 2017, en el Ecuador existen aproximadamente 201.275 personas con discapacidad física-motriz, de manera específica en las provincias de Loja, El Oro y Zamora Chinchipe (provincias que conforman la Región 7) existen 5.605, 8.705 y 1.550 personas respectivamente, obteniendo un total de 15.860 personas con discapacidad física-motriz en la Región 7. La Región 7 es el campo de acción territorial de la presente investigación. Se debe considerar que las cifras mencionadas corresponden a personas con cualquier tipo y grado de discapacidad física-motriz [1].

En el Ecuador, según el CONADIS, la inserción laboral de las personas con discapacidad física-motriz, hasta agosto de 2017, es de 31.926 personas, lo que representa que solamente el 15.86% de personas con discapacidad motriz han tenido las oportunidades, condiciones y herramientas protésicas para poder desempeñar una actividad laboral [1]. Esta proporción nacional es baja y una de las causas principales es la falta de acceso a prótesis asequibles con capacidades funcionales óptimas para desempeñar actividades en el campo laboral. Además en la Región 7 la proporción es aún menor (12.98%) ya que son menores las oportunidades de acceso a prótesis asequibles con capacidades funcionales.

En base al problema anterior nace el proyecto “Mano de Esperanza”, cuya misión es desarrollar prótesis robóticas de bajo costo con la finalidad de aportar a la inclusión social y laboral de personas con discapacidad motriz en sus extremidades superiores, la misión del proyecto está sintonizada con las necesidades de los países en desarrollo [2].

El presente trabajo forma parte del proyecto “Mano de Esperanza”, el objetivo central abordado es el diseño y desarrollo de un controlador embebido, basado en PID, aplicado a la regulación de fuerza de la prótesis robótica desarrollada en [2].

En el entorno académico y científico este tipo de problemas es abordado desde algunas perspectivas. Por ejemplo en [3] para facilitar la manipulación de objetos de diferentes densidades y tamaños sin llegar a estropearlos, se retroalimenta la fuerza de agarre aplicada mediante el uso de sensores FSR (Force Sensitive Resistor). Entre otros métodos para el registro y regulación de fuerza de agarre, se realiza en base a la estimación de la fuerza aplicada por los actuadores en base a la corriente consumida por estos, obteniendo una relación posición-fuerza [4]. En [5] se implementa resistores Shunt embebidos en una tarjeta electrónica de control, lo que reduce el número de componentes externos para la estimación de fuerza de agarre en base a la corriente consumida por cada actuador, facilitando el ensamblaje de la prótesis robótica y la implementación del lazo de control. En

[6] se menciona que los sensores FSR son usados en aplicaciones biomecánicas para mediciones de fuerza, sin embargo estos sensores pueden variar su respuesta en determinadas condiciones como se ha estudiado en [7]. A pesar de lo comentado, los sensores FSR son una atractiva forma de medir la fuerza debido a su bajo costo en comparación con otros sensores especializados [7].

Existen diferentes metodologías de control aplicado a la regulación de fuerza en prótesis robóticas. En [8] se usa una metodología de optimización basado en el algoritmo CpG y una función de control PID paralelo de fuerza-posición, implementando en un computador personal y no en un sistema embebido. En [4] se propone una arquitectura de control adaptativo de fuerza-posición en base a una función de costo, sin embargo los sensores y el algoritmo de control no son parte de un sistema embebido. En [3] la metodología se basa en funciones PID en paralelo ejecutados en un computador personal y la adquisición de datos se implementa en un sistema embebido basado en Arduino UNO.

En base al análisis bibliográfico y a los objetivos del proyecto “Mano de Esperanza”, los cuales buscan el desarrollo de una prótesis robótica de bajo costo se considera que las mejores opciones para retroalimentar la variable fuerza de agarre son: sensor de fuerza basado en FSR y el sensor de corriente basado resistores Shunt, estas señales permitirán desarrollar un controlador de regulación de fuerza de agarre implementado en un sistema embebido. Además, mediante el análisis bibliográfico se considera que la arquitectura de control y la metodología a usar varían dependiendo de la prótesis que se utiliza, por ende, en el presente trabajo se propone una metodología para el diseño y desarrollo de lazo de control de fuerza de agarre aplicada al prototipo de prótesis robótica realizada en el proyecto “Mano de Esperanza” y documentada en [2].

El desarrollo del presente proyecto se encuentra estructurado de la siguiente forma. En el capítulo 1 se realiza la descripción del estado del arte tanto para el hardware como para el firmware a desarrollar, además se describen los métodos de sintonización que serán usados para diseñar el controlador de fuerza de la prótesis robótica. En el capítulo 2 se diseña e implementa el hardware y firmware que conforman el sistema de control de fuerza de la prótesis robótica, también se describe el diseño e implementación de los filtros digitales. En el capítulo 3 se diseñan y sintonizan los lazos de control aplicando 6 métodos diferentes, para los dos tipos de sensores (FSR y resistores shunt), de esta etapa se obtiene el mejor controlador para cada dedo que será llevado a la implementación. En el capítulo 4 se evalúa el controlador embebido de la prótesis robótica, mediante el agarre de dos tipos de objetos (esféricos y cilíndricos) con diferentes diámetros y pesos. Finalmente, en el capítulo 5 se mencionan las conclusiones más importantes obtenidas en el desarrollo del proyecto.

CAPÍTULO I

ESTADO DEL ARTE

1.1. Arquitectura del hardware.

En el diagrama de bloques del sistema embebido (ver figura 1.1), se muestra en el bloque 3, los elementos para el lazo de control de posición desarrollado en [2], en los bloques 4 y 5 se muestran los sensores que realizarán la conversión de la magnitud de fuerza a un nivel de voltaje que servirán como señales de retroalimentación para el lazo de control de fuerza de agarre.

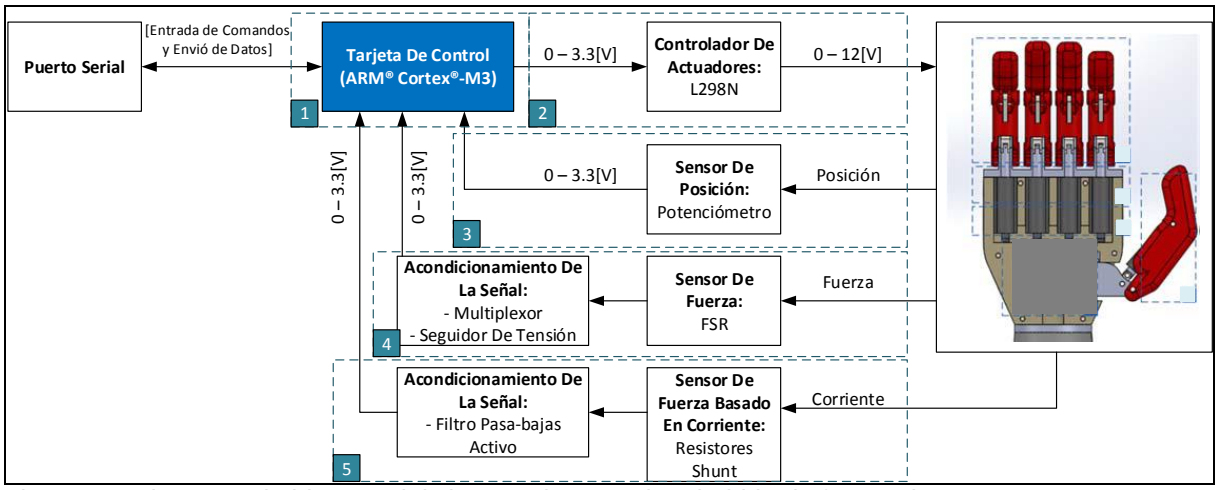


Figura 1.1 Diagrama de bloques del sistema de control embebido de fuerza de agarre
Fuente: Autores
Elaboración: Autores

En esta sección, se procede a describir los elementos que conforman el hardware del sistema embebido (ver figura 1.1), en el cual se implementa el algoritmo de control de fuerza de agarre basado en PID, y los filtros digitales para el correcto acondicionamiento de las señales de retroalimentación.

Además, se realiza una descripción del filtro activo analógico RC pasa-bajas, usado en el filtrado del voltaje en relación a la corriente obtenida a través del Resistor Shunt, y de los métodos de regresión usados para la obtención de las ecuaciones en el caso de los FSR.

1.1.1. Microcontrolador ARM® Cortex®-M3.

Como parte central del sistema embebido se escogió el microcontrolador ATSAM3X8E de Microchip®, el cual posee una arquitectura de 32 bits y trabaja a 84 [MHz], entre otras características que se describen en la tabla 1.1.

Además, ARM® brinda las librerías CMSIS que facilitan la configuración y uso de este microcontrolador, entre estas la más importante es CMSIS-DSP, la cual facilita la implementación de algoritmos de procesamiento digital de señales de forma eficiente en estos dispositivos [9].

Tabla 1.1 Parámetros del Microcontrolador ATSAM3X8E

Descripción	Estado
CPU	ARM® Cortex®-M3
Pin & Package	144LQFP
Flash [kB]	512
EEPROM [kB]	2
SRAM [kB]	96
Velocidad máxima [MHz]	84
Canales PWM	6
QEI	2
GPIO	103
Temperatura de Operación [°C]	-40 a 85
USB D, H/D, u OGT	H/D
SSI/SPI	1
I2C	2
UART	4
Canales ADC	15
Resolución ADC [Bits]	12
CAN MAC	2
SysTick	Si
SPI	6

Fuente: [10]
Elaboración: Autores

Las principales razones por las cuales se optó por este microcontrolador son: por la demanda de procesamiento requerido por el lazo de control de fuerza de agarre, los filtros digitales y por el hecho de contar con una arquitectura 32 bits. A demás, nos permite manejar variables punto flotante con un número inferior de ciclos de reloj, en comparación de un microcontrolador de arquitectura de 8 bits [11], como el ATmega328p usado en el trabajo previo [2].

1.1.2. L298N, controlador de motores DC.

Para el manejo de los Actuadores PQ12, se usó el driver L298N, el cual tiene un pin asignado para la colocación de un resistor Shunt (ver figura 1.2), a diferencia del controlador L293D usado en el trabajo previo que carecía de esto [2]. Este pin nos permite monitorear la corriente consumida por cada actuador, para ello debe conectarse a una entrada ADC del microcontrolador.

Este controlador soporta voltajes de operación de hasta 46 [V], puede suministrar una corriente de hasta 4 [A], y puede soportar una frecuencia de conmutación de hasta 40 [kHz]. Para el manejo de este controlador es necesario el uso de 2 pines digitales del microcontrolador y de un canal PWM, para cada motor (ver figura 1.3). El módulo del controlador se muestra en la figura 1.4.

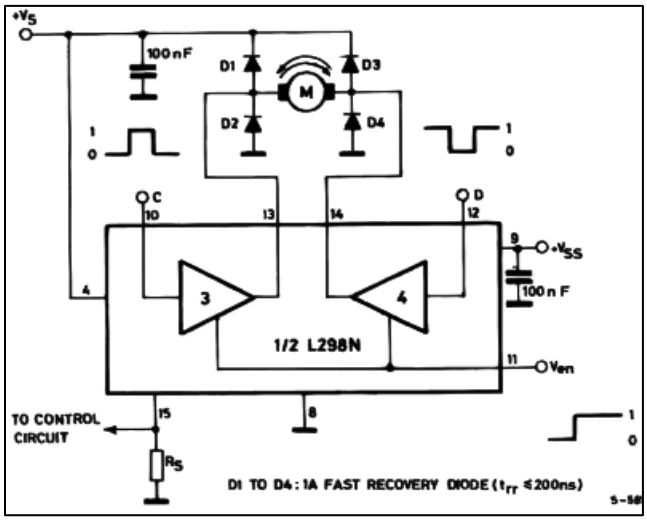


Figura 1.2 Diagrama del controlador L298N
Fuente: *[12]*
Elaboración: [12]

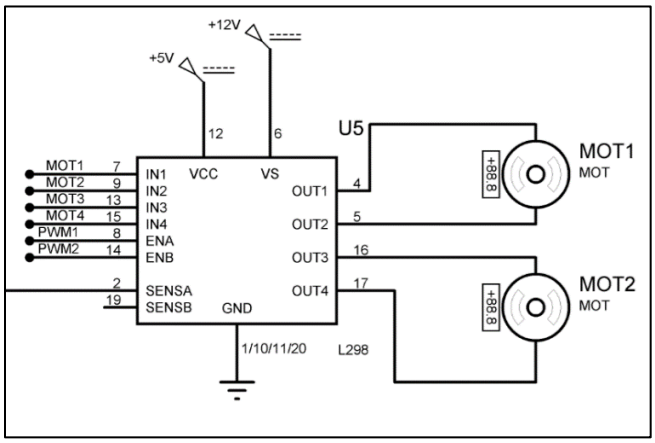


Figura 1.3 Esquemático de conexión del controlador L298N
Fuente: Autores
Elaboración: Autores

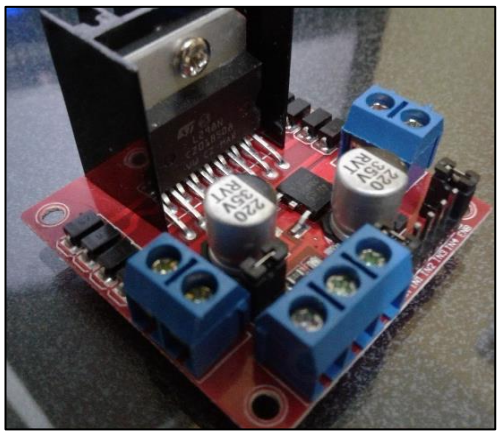


Figura 1.4 Módulo del controlador L298N
Fuente: Autores
Elaboración: Autores

1.1.3. Resistor Shunt.

Para la correcta medición de la corriente consumida por cada actuador, la hoja de datos del controlador L298N nos recomienda el uso de un resistor shunt (R_s) (ver figura 1.2), para encontrar el valor del Resistor Shunt, se hace uso de la siguiente fórmula [13]:

$$R_{shunt} = \frac{V_{shunt}}{I_{shunt}} \quad (1.1)$$

Dado que se desea que el voltaje que pase a través de este resistor sea igual a la corriente que consumen los motores, el valor de la Resistencia Shunt será de 1 [Ω].

Otro dato necesario para el correcto funcionamiento del Resistor Shunt es la potencia disipada por este, para este cálculo usaremos la fórmula siguiente [13]:

$$P = I^2 R \quad (1.2)$$

Ya que la corriente máxima de los actuadores es de 115 [mA] [2], la potencia disipada será de 13.23 [mW].

Además, para reducir el ruido se debe colocar un filtro activo luego del Resistor Shunt, la conexión se muestra a en la figura 1.5.

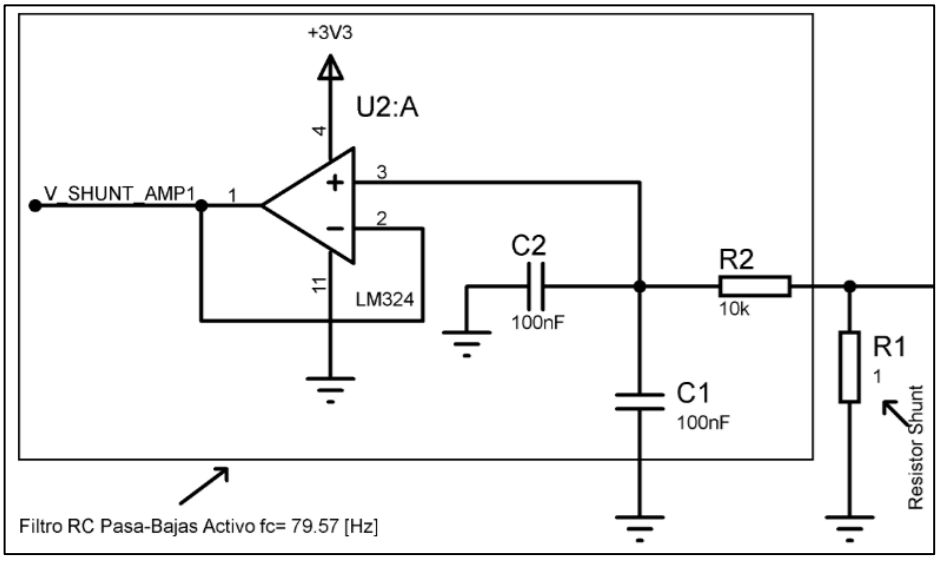


Figura 1.5 Conexión del resistor shunt y el filtro pasa-bajas
Fuente: Autores
Elaboración: Autores

1.1.3.1. Filtro activo

Una técnica de filtrado analógico usada en el acondicionamiento de la señal de retroalimentación obtenida por medio de los Resistores shunt, es la del filtro activo (ver figura 1.5). Este tipo de filtros tienen la capacidad de ofrecer una ganancia unitaria a la salida del filtro, independientemente de la carga colocada a la salida del mismo [14]. Esto permite aislar la impedancia interna de los pines del microcontrolador debido a que el amplificador operacional se comporta como un seguidor de tensión, y a su vez obtener una correcta medición[15].

La frecuencia de corte (f_c) de este filtro se puede obtener de la siguiente fórmula [16]:

$$f_c = \frac{1}{2\pi RC} \qquad (1.3)$$

Para la frecuencia de 79.6 [Hz]:

$$R = 10\ [k\Omega]$$

$$C = 200\ [nF]$$

La atenuación de este filtro en la frecuencia de corte es de -3 [dB], y el desfase es de 45° grados (ver figura 1.6). El uso de este tipo de filtros antes de muestrear la señal con el ADC es altamente recomendado, ya que funciona como un filtro de guardia que remueve todas las componentes de frecuencia fuera del rango de la frecuencia de corte, tal y como se describe en [17], [18].

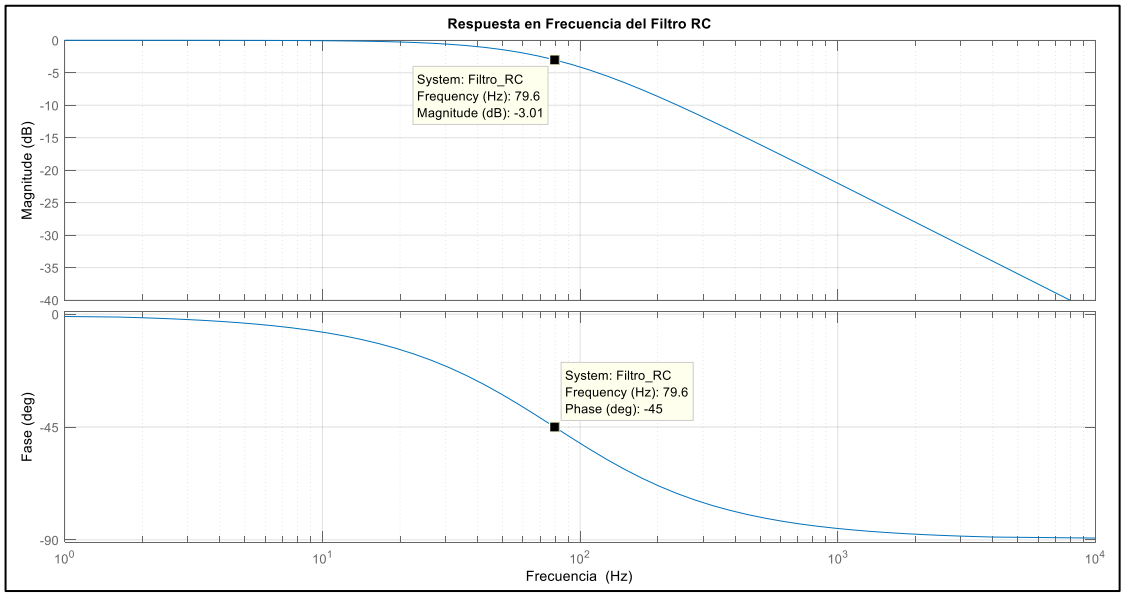


Figura 1.6 Respuesta en frecuencia del filtro RC
Fuente: Autores
Elaboración: Autores

1.1.4. Amplificador operacional LM324-N.

Como un elemento indispensable para la lectura de sensores es su acondicionamiento, el amplificador LM324-N se usa en un formato seguidor de tensión para evitar una lectura errónea del sensor. Una característica esencial del amplificador LM324-N es su capacidad de admitir fuentes de alimentación simples entre 3V a 32V [19].

Para un sistema embebido y portátil que requiere de un sistema de alimentación, este amplificador se convierte en una opción como componente en el acondicionamiento para la lectura de los sensores.

Características: [19]

- Alimentación simple 3V – 32V
- Alimentación dual +-1.5V - +-16V
- Drenaje de corriente 700μA
- Independencia de la salida con la alimentación
- Consumo de corriente 45nA

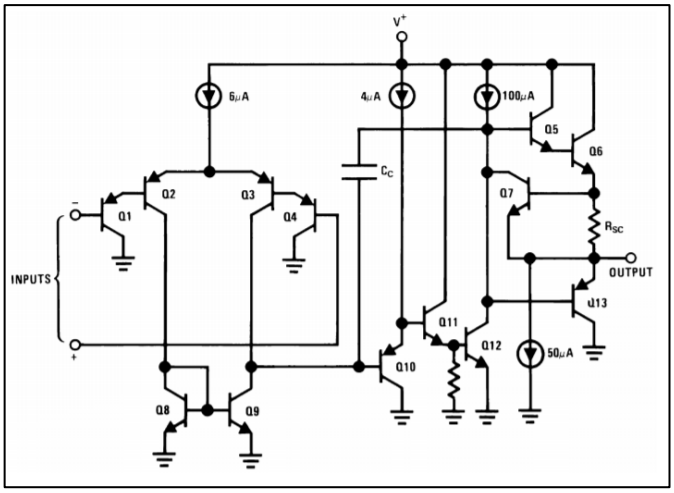


Figura 1.7 Diagrama del amplificador LM324-N
Fuente: [19]
Elaboración: [19]

1.1.5. FSR (Force-Sensitive Resistor).

Es un sensor que permite detectar presión y peso. El FSR está construido con diferentes filamentos [20], los mismos que al tener una presión cambian su resistencia óhmica facilitando detectar el peso aplicado mediante un divisor de voltaje.

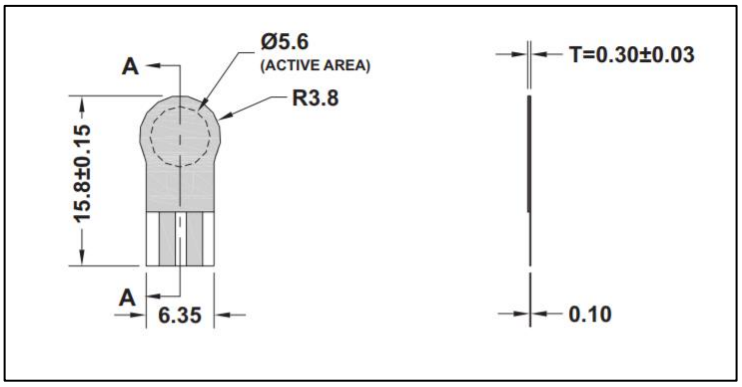


Figura 1.8 Medidas en milímetros del sensor FSR
Fuente: [20]
Elaboración: [20]

Para evitar errores en la lectura del sensor el fabricante (Interlink Electronics), [21] y [22] recomienda usar un amplificador configurado como seguidor de tensión, como se muestra en la figura 1.9, para impedir un desacople de impedancias entre el divisor de voltaje y el microcontrolador ARM® Cortex®-M3. El seguidor de tensión no es más que un acople de impedancias, que proporciona a la salida la misma tensión de entrada.

Ya que el controlador PID debe estar embebido en un microcontrolador ARM® Cortex®-M3, su ADC permite la medición de voltajes inferiores a 3.3V obligando a alimentar el sensor FSR con la máxima tensión permitida por el ADC. [9]

El divisor de tensión obedece a la siguiente ecuación:

$$V_{OUT} = \frac{R_M V}{(R_M + R_{FSR})} \qquad (1.4)$$

Donde:

R_M: Es la resistencia para implementar el divisor de voltaje.

V: Es el voltaje de alimentación del divisor de voltaje.

R_{FSR}: Sensor FSR 400 short tail Interlink Electronics.

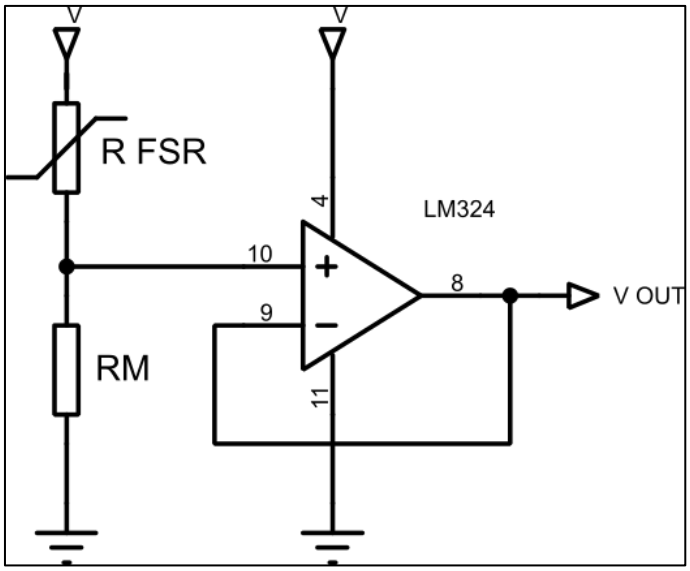


Figura 1.9 Acondicionamiento para reducir el efecto de carga mediante el acople de impedancias para el sensor FSR
Fuente: [20]
Elaboración: Autores

1.2. Arquitectura del firmware.

1.2.1. Resistor Shunt.

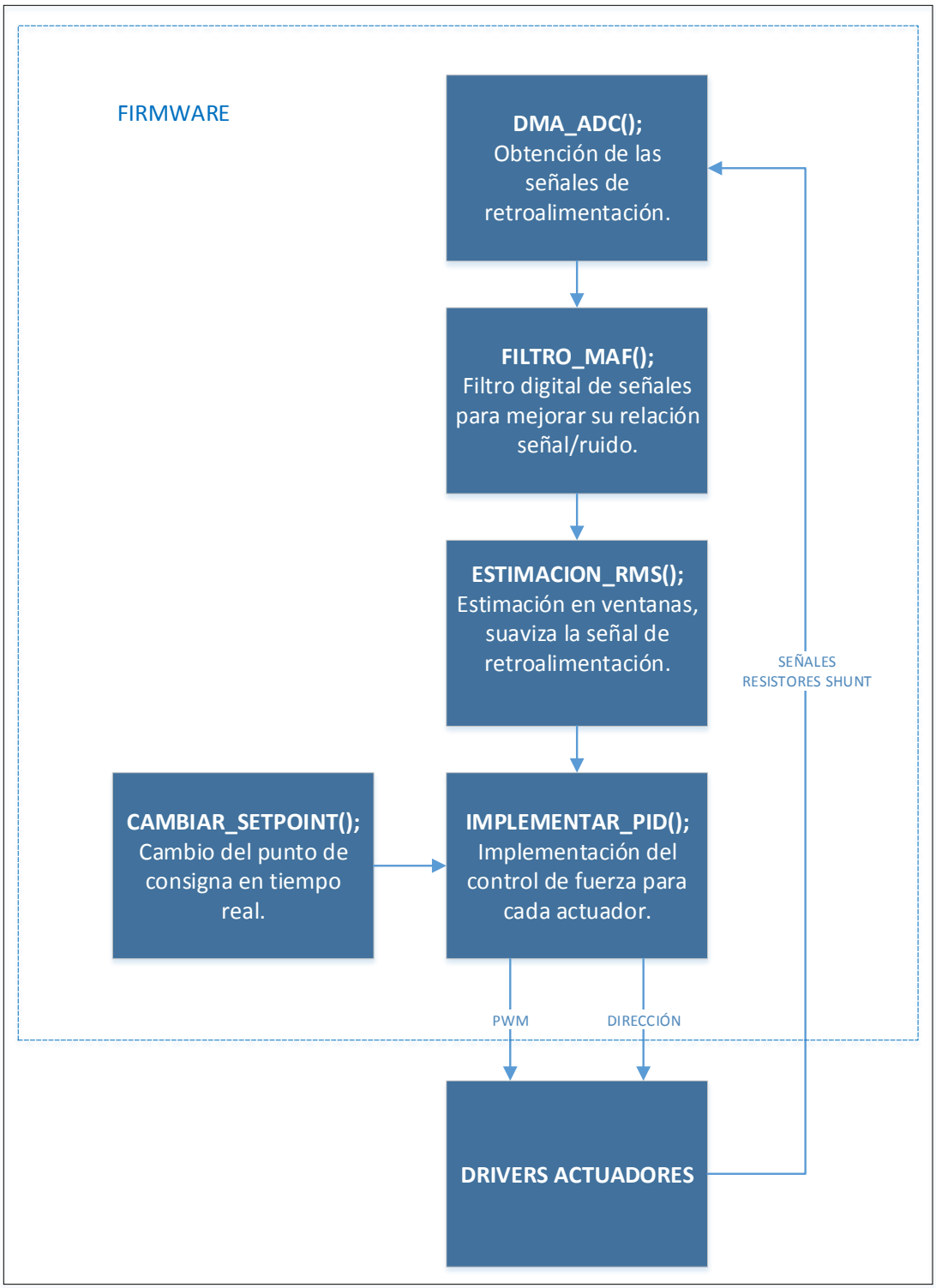


Figura 1.10 Diagrama de la arquitectura del firmware sensor de corriente
Fuente: Autores
Elaboración: Autores

En el caso del firmware, se ha iniciado con el diseño de su arquitectura (ver figura 1.10), en esta arquitectura se describen brevemente las funciones más importantes y se muestra el orden en que se ejecutaran cada una de estas. A continuación, se describen a detalle cada uno de estos bloques.

1.2.1.1. ADC mediante acceso directo a memoria (DMA).

En [10], se indica que la tasa máxima de muestreo del ADC del ATSAM3X8E es de $1x10^6$ muestras por segundo, para asegurar que las muestras sean obtenidas con el mínimo de latencia se debe obtener la data de los ADCs mediante el acceso directo a memoria (DMA), esto implica que los valores de conversión serán copiados directamente a un buffer de almacenamiento para su posterior uso, permitiendo que el ADC opere a su máxima capacidad posible reduciendo la latencia en la conversión de todos los canales ADC del microcontrolador tal y como se describe en [23],[24].

1.2.1.2. Filtro digital pasa-bajas MAF (Moving Average Filter).

En este bloque, se realiza el filtrado digital de la señal de voltaje en función de la corriente consumida por cada actuador, a través de los resistores shunt como se muestra en el bloque 5 de la arquitectura del hardware (ver figura 1.1). Este filtro es una variación del filtro FIR donde cada uno de sus coeficientes son iguales, dado que estima la señal de salida mediante el promedio de las señales de entrada anteriores [25]. La fórmula de este filtro se muestra a continuación [26]:

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k] \qquad (1.5)$$

Donde:

$y: es\ la\ se\tilde{n}al\ de\ salida\ estimada.$

$x: son\ las\ se\tilde{n}ales\ de\ entrada\ con\ ruido.$

$N: es\ el\ orden\ del\ filtro.$

Se escogió este tipo de filtro digital debido a que su coste computacional es reducido, convirtiéndolo en un algoritmo de ejecución rápida [27].

1.2.1.3. Estimación RMS.

Dado que el control de los actuadores es mediante señales PWM, se crea histéresis en la señal de retroalimentación debido a su inductancia. En la figura 1.11, se muestra este efecto.

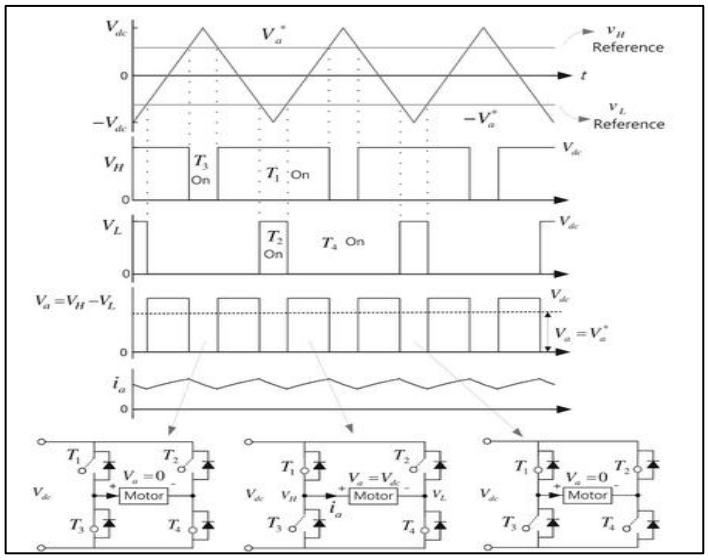


Figura 1.11 Efecto de histéresis en la señal de corriente
Fuente: [28]
Elaboración: [28]

Con la estimación RMS de la señal de corriente, obtenemos el valor medio cuadrático de la señal que es una aproximación más fiable a la corriente real consumida por cada uno de los actuadores. Además, para la implementación se necesita aplicar el método de ventana deslizante [29], que consiste en usar un buffer circular para el almacenamiento de la señales de entrada anteriores.

La fórmula en tiempo discreto de la estimación RMS es [29]:

$$y_{RMS}[n] = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x[n]^2} \qquad (1.6)$$

Donde:

y_{RMS}: $es\ la\ señal\ RMS\ de\ salida$

x: $señales\ de\ entrada\ con\ histeresis$

N: $orden\ de\ la\ ventana$

1.2.1.4. Implementación PID.

En este bloque se implementa el algoritmo de control PID, para ello se usará como señal de retroalimentación la estimación RMS de las señales, se describirá más a detalle el algoritmo en la siguiente sección. La salida de este bloque es el porcentaje del ciclo de trabajo de la señal PWM y la dirección que deberán tomar cada actuador según estime el controlador PID, estas señales son enviadas de forma directa al controlador físico de los actuadores.

1.2.1.5. Cambio de set-point.

Este bloque permite al usuario poder ingresar un valor de referencia, que es usado en el lazo de control. Para que el sistema embebido responda en tiempo real, se implementa esta función en una interrupción del UART.

1.2.2. FSR.

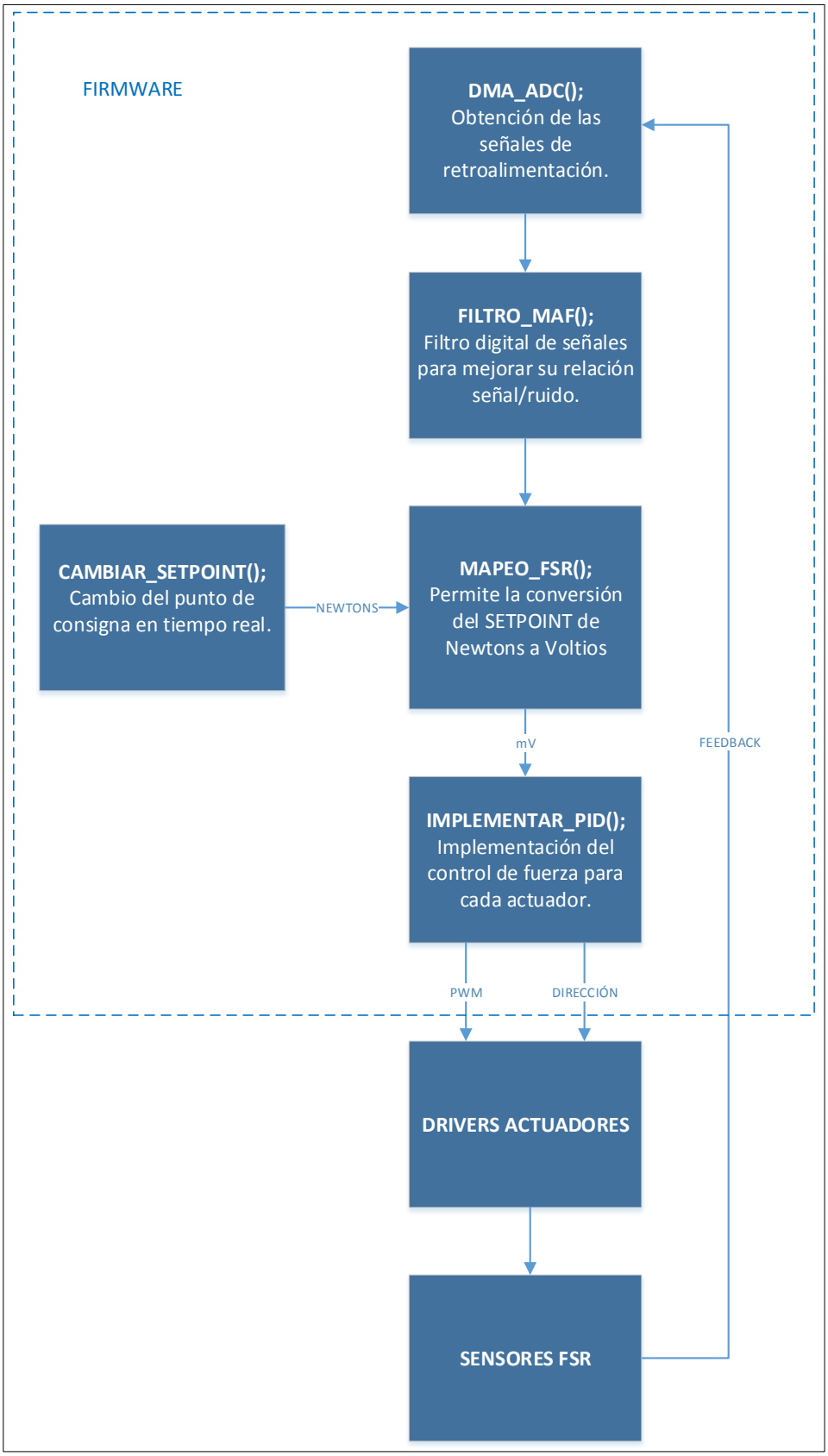


Figura 1.12 Diagrama de la arquitectura del firmware sensor fuerza FSR
Fuente: Autores
Elaboración: Autores

1.2.2.1. Mapeo FSR.

En este bloque se realiza una conversión de unidades, logrando establecer una relación entre Newtons y Voltios, permitiendo al bloque IMPREMENTAR_PID() mantener un control constante en unidades de mili voltios. Se realiza una regresión polinomial a partir de datos previos, tomados del sensor FSR y una galga extensiométrica, con unidades de voltios y Newtons del sensor respectivo en tiempo real.

Su importancia radica en dar la capacidad al usuario de ingresar una fuerza de agarre o nuevo set-point requerido en Newtons. La regresión polinomial se realiza con el software Matlab®, el cual aplica el método de matriz de Vandermonde descrita en [30] para resolver la regresión y devolver una ecuación polinomial.

1.3. Controladores PID.

Es un sistema que, a través de una retroalimentación dada por un sensor, mide la desviación o error que existe entre la señal medida con la señal de referencia. Un algoritmo PID se compone de tres parámetros K_p, T_i, T_d, que se ajustan mediante sintonizadores con la finalidad de obtener una respuesta rápida y robusta frente a un cambio de set-point. [31]

Un controlador PID está dado por: [32]

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right) \quad (1.7)$$

Donde:

$u(t)$: Es la señal de control.

$e(t)$: Es el error de control $(e = y_{sp} - y)$

La señal de control es la suma de tres términos: (P) proporcional al error, (I) proporcional a la integral del error, (D) proporcional a la derivada del error.

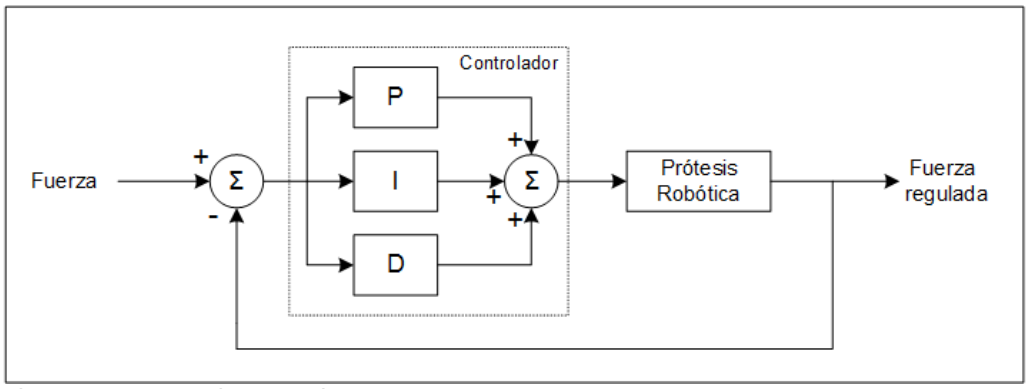


Figura 1.13 Lazo de control PID
Fuente: [32]
Elaboración: Autores

Para un control PID embebido, la formula (1.7) se transforma en su equivalente discretizada, para ello se usa la aproximación trapezoidal de la integral [33].

La fórmula queda de la siguiente manera [33]:

$$u(kT) = Kp\left[e(kT) + \frac{T}{T_i}\sum_{k=1}^{n} e(kT) + T_d\frac{e(kT) - e(kT-T)}{T}\right] + u_0 \quad (1.8)$$

De esta manera, el algoritmo PID se puede implementar en un sistema embebido.

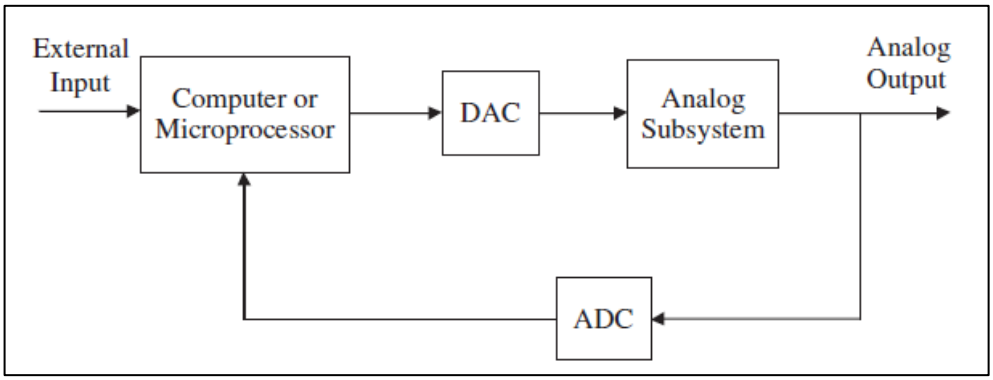


Figura 1.14 Diagrama de bloques de un sistema digital común
Fuente: [34]
Elaboración: [34]

Como se observa en la figura 1.14, el microcontrolador en el lazo de control digital es un bloque encargado de realizar los cálculos del controlador basado en el error de la señal de retroalimentación con respecto a un valor de consigna.

Para sintonizar el controlador PID se usan seis métodos descritos en el siguiente apartado.

1.3.1. Método de sintonización basado en Ziegler – Nichols.

Este método de sintonización asegura que la respuesta de un escalón como entrada del sistema no sobrepase del 25% de sobre-elongación, y que sus características de rapidez y estabilidad sean aceptables [35].

Existen dos procedimientos para usar esta sintonización: en lazo abierto, y en lazo cerrado [35]. En nuestro caso nos enfocaremos en la sintonización en lazo cerrado o bien llamado sintonización de respuesta en frecuencia, para esto se siguen los siguientes pasos [36]:

1) Incrementar la ganancia proporcional hasta que el sistema muestre oscilaciones mantenidas, el valor de ganancia que genere este efecto será la ganancia critica (K_c).

2) Obtener el periodo crítico de las oscilaciones mantenidas (T_u).

3) Mediante la siguiente tabla (ver tabla 1.2), se obtienen las ganancias del controlador.

Tabla 1.2 Tabla de obtención de ganancias mediante Ziegler-Nichols en lazo cerrado

Tipo de Controlador	K_p	T_i	T_d
Proporcional - P	$0.5 * K_c$	∞	0
Prop. Integral - PI	$0.45 * K_c$	$\frac{T_u}{1.2}$	0
Prop. Int. Derivativo - PID	$0.6 * K_c$	$\frac{T_u}{2}$	$0.125 * T_u$

Fuente: [28]
Elaboración: Autores

1.3.2. Método de sintonización basado en Tyreus – Luyben.

Este método es similar al de Ziegler-Nichols en lazo cerrado, la diferencia más notable es que antepone la robustez ante el rendimiento, dado que posee un alto coeficiente de amortiguamiento [37]. En la tabla 1.3, se encuentran las fórmulas para obtener las ganancias de los controladores basados en la ganancia crítica (K_c) y el periodo crítico (T_u).

Tabla 1.3 Tabla de obtención de ganancias mediante Tyreus-Luyben en lazo cerrado

Tipo de Controlador	K_p	T_i	T_d
Proporcional - P	–	–	–
Prop. Integral - PI	$\frac{K_c}{3.2}$	$2.2 * T_u$	–
Prop. Integral Derivativo - PID	$\frac{K_c}{2.2}$	$2.2 * T_u$	$\frac{T_u}{6.3}$

Fuente: [30]
Elaboración: Autores

1.3.3. Método de sintonización basado en Shinskey.

Este método posee una buena relación entre rendimiento y robustez ante el ruido y a las variaciones del punto de consigna o set-point [38]. Las fórmulas para la obtención de las ganancias de los controladores se muestran en la tabla 1.4.

Tabla 1.4 Tabla de obtención de ganancias mediante Shinskey en lazo cerrado

Tipo de Controlador	K_p	T_i	T_d
Proporcional - P	$\frac{K_c}{2}$	–	–
Prop. Integral - PI	$\frac{K_c}{2}$	$\frac{T_u}{2.2}$	–
Prop. Integral Derivativo - PID	$\frac{K_c}{4}$	$\frac{T_u}{2}$	$\frac{T_u}{8.3}$

Fuente: [31]
Elaboración: Autores

1.3.4. Método de sintonización basado en Shen – Yu.

Modificando el método de sintonización Ziegler – Nichols para evitar el sub-amortiguamiento que conlleva éste, Shen – Yu a través de un proceso de desintonización en sistemas de primer orden con tiempo muerto llegó a obtener nuevos coeficientes, el proceso completo de la modificación del método se describe en [39]. Las ganancias del controlador PI se muestra en la tabla 1.5.

Tabla 1.5 Tabla de obtención de ganancias mediante Shen-Yu en lazo cerrado

Tipo de Controlador	K_p	T_i	T_d
Prop. Integral - PI	$\frac{K_u}{3}$	$\frac{P_u}{0.5}$	–

Fuente: [32]
Elaboración: Autores

Donde K_u es la ganancia en la que el controlador mantiene oscilaciones permanentes, y P_u es el periodo de las oscilaciones.

1.3.5. Método de sintonización basado en Lambda.

Es una forma de control de modelo interno IMC descrito en [40] y [41], dota de un controlador de tipo proporcional integrativo (PI), el método lambda es capaz de producir un cambio suave y no oscilante al momento del cambio del set-point. Lambda (λ) es un parámetro que puede ser modificado por el usuario, contemplando con éste el tiempo deseado de la respuesta del proceso a un cambio del set-point.

Las reglas de sintonización se calculan a partir de las fórmulas desarrolladas por Chien, en la tabla 1.6 se muestran las ganancias del controlador PI.

Tabla 1.6 Tabla de obtención de ganancias mediante Lambda en lazo abierto

Tipo de Controlador	K_p	T_i	T_d
Prop. Integral - PI	$\frac{\tau}{G_p(\lambda+\theta)}$	τ	–

Fuente: [33]
Elaboración: Autores

Donde τ es el tiempo de proceso, se toma luego del tiempo muerto hasta el 63.2% de la curva del sistema en lazo abierto, G_p es la ganancia en porcentaje con respecto a la amplitud del escalón, θ es el tiempo muerto de la curva y λ es $\tau * \lambda_f$, siendo λ_f un factor que al ser multiplicado por τ no debe ser menor al tiempo muerto.

1.3.6. Método de sintonización basado en Cohen – Coon.

Usa el mismo método que aplica Ziegler - Nichols para obtener los parámetros a partir de una curva de un proceso dinámico. Esta técnica de sintonización se desarrolló asumiendo un proceso de primer orden con un tiempo muerto (θ) [40]. Cohen - Coon a diferencia de Ziegler-Nichols funciona correctamente en sistemas donde el tiempo muerto puede ser menor que dos veces el tiempo de proceso.

El diseño original del método tiene una respuesta con demasiada oscilación, cayendo en el desuso. Sin embargo, con unas modificaciones, las reglas de sintonización demostraron su efectividad minimizando las oscilaciones y respondiendo rápidamente [42].

Tabla 1.7 Tabla de obtención de ganancias mediante Cohen-Coon en lazo abierto

Tipo de Controlador	K_p	T_i	T_d
Prop. Integral - PI	$\frac{0.45}{G_p} * \left(\frac{\tau}{\theta} + 0.092\right)$	$3.33 t_d * \left(\frac{\tau + 0.092\,\theta}{\tau + 2.22\,\theta}\right)$	–
Prop. Integral Derivativo - PID	$\frac{0.67}{G_p} * \left(\frac{\tau}{\theta} + 0.185\right)$	$2.5 t_d * \left(\frac{\tau + 0.185\,\theta}{\tau + 0.611\,\theta}\right)$	$0.37\,\theta * \left(\frac{\tau}{\tau + 0.185\,\theta}\right)$

Fuente: [35]
Elaboración: Autores

CAPÍTULO II
DISEÑO E IMPLEMENTACIÓN DEL SISTEMA EMBEBIDO

2.1. Arquitectura Hardware.

El hardware del sistema embebido fue dividido en módulos para facilitar su diseño y testeo. Estos módulos se desarrollaron en la etapa de prototipado de lo que será la PCB final que ira embebida en la prótesis robótica, en reemplazo de la anterior que se desarrolló en [2].

Estos módulos fueron desarrollados usando el software CAD Proteus® 8. A continuación, se describirá la función de cada uno de estos y los elementos que los conforman.

2.1.1. Módulo de adquisición y acondicionamiento de señales.

Este módulo facilita el acople del Arduino Due, el cual es una placa de desarrollo basada en el ATSAM3X8E [43], ya que se diseñó en un formato tipo Shield. Además, contiene los circuitos de acondicionamiento de las señales de retroalimentación para los lazos de control de fuerza basado en la medición de corriente (ver figura 1.5) y basado en sensores FSR (ver figura 1.9).

La medición de las señales para el sensor FSR se realiza a través de un divisor de tensión con una resistencia RM de 3 [kΩ] (ver figura 1.9), el voltaje para alimentación del sensor y el amplificador de 3.3 [V]. El uso de el amplificador LM324-N en configuración de seguidor de tensión es un acoplador de impedancias que evita cualquier desacople debido a la placa de desarrollo Arduino Due.

En la figura 2.1 y 2.2, se observa el módulo diseñado en Proteus® 8. Los diagramas esquemáticos y la disposición del PCB se encuentran en el anexo A.

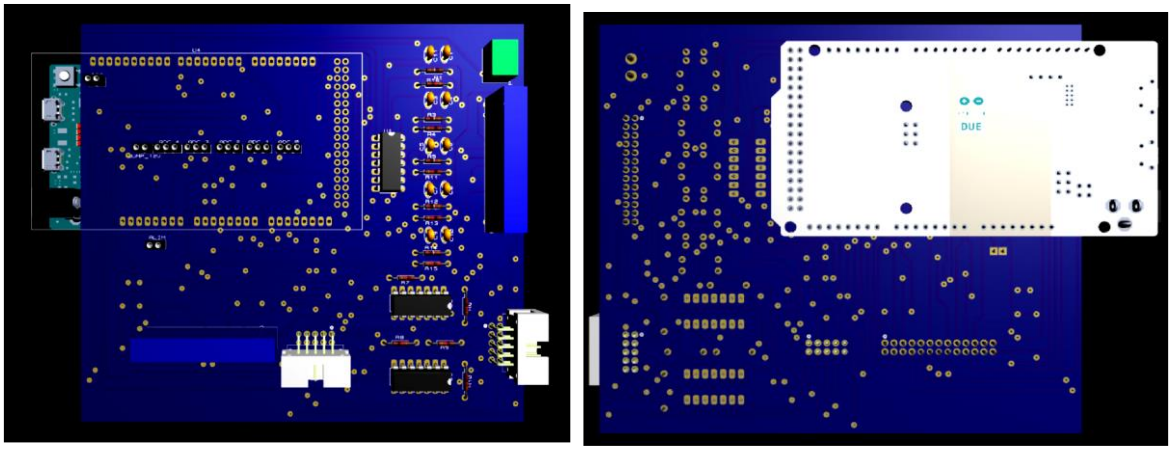


Figura 2.1 Modelo 3D del módulo de adquisición y acondicionamiento de las señales: a) Vista superior, y b) Vista inferior
Fuente: Autores
Elaboración: Autores

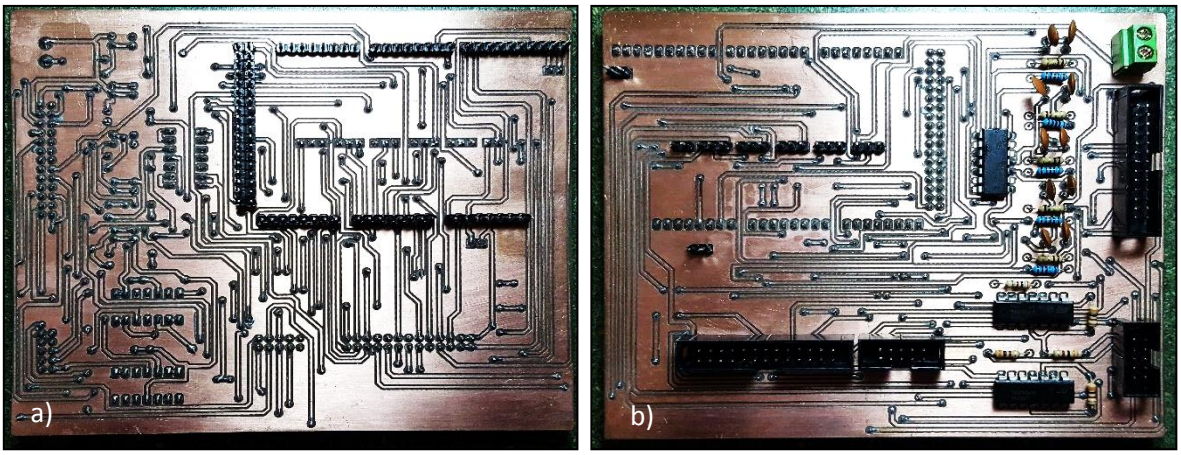


Figura 2.2 Módulo de adquisición y acondicionamiento de las señales:
a) Vista superior, y b) Vista inferior
Fuente: Autores
Elaboración: Autores

2.1.2. Módulo de conexión para los drivers L298N.

Para que cada driver L298N pueda controlar dos motores DC, necesita de trece conexiones hacia la tarjeta de control, alimentación y motores. Para controlar los cinco actuadores de la prótesis robótica se requiere de tres drivers de los cuales uno de ellos queda con cinco bornes libres, con el fin de facilitar la conexión de todos los pines a la tarjeta de control se ha desarrollado una PCB que convierte los tres drivers en un módulo para cinco actuadores.

Dos conectores para bus de datos J1 (26 pines) y J2 (10 pines) (ver figura 2.3) permiten la comunicación entre el presente módulo y el módulo de adquisición y acondicionamiento de señales descrito en la sección 2.1.1. En el anexo A se encuentra el esquema eléctrico de las conexiones.

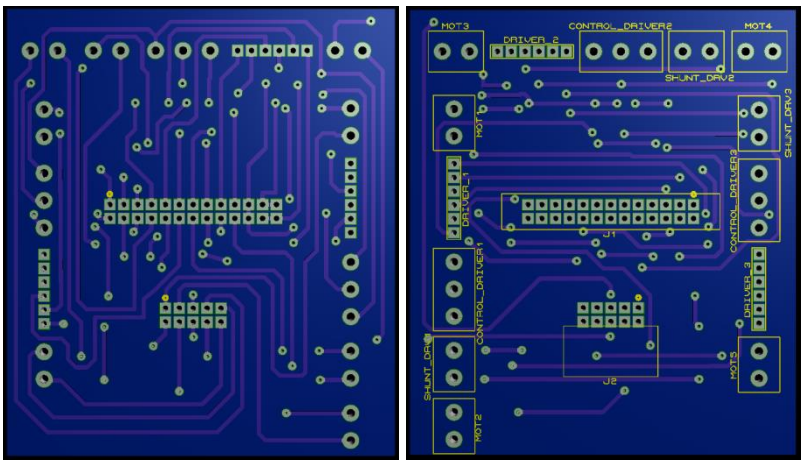


Figura 2.3 Modelo 3D del módulo para conexión de los drivers L298N, vista inferior y superior
Fuente: Autores
Elaboración: Autores

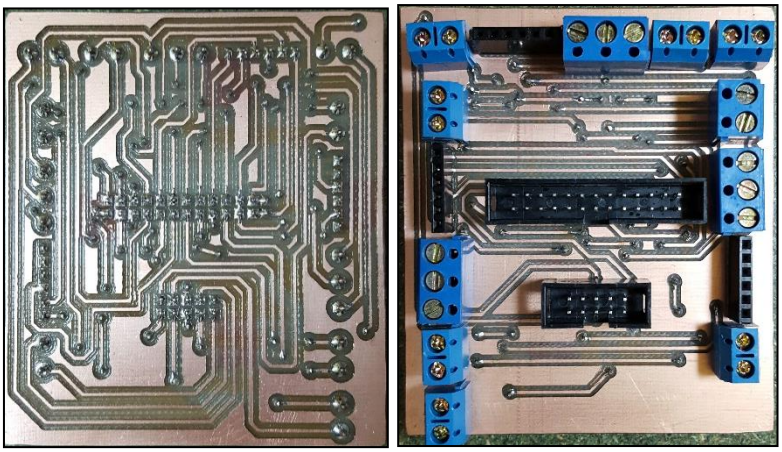


Figura 2.4 Módulo de conexión de los drivers L298N implementado, vista inferior y superior
Fuente: Autores
Elaboración: Autores

2.1.3. Módulo de conexiones para periféricos de la prótesis robótica.

Para las conexiones de los actuadores y los sensores FSR ubicados en la prótesis robótica se requiere de una PCB que permite la interconexión de todos los periféricos correspondientes. Esta PCB no comprende elementos pasivos ni activos, son pistas para la distribución de conexiones en sentido bilateral de la prótesis robótica, y hacia el módulo de adquisición de datos descrito en el apartado 2.1.1. En el anexo A se encuentra el esquema eléctrico de las conexiones.

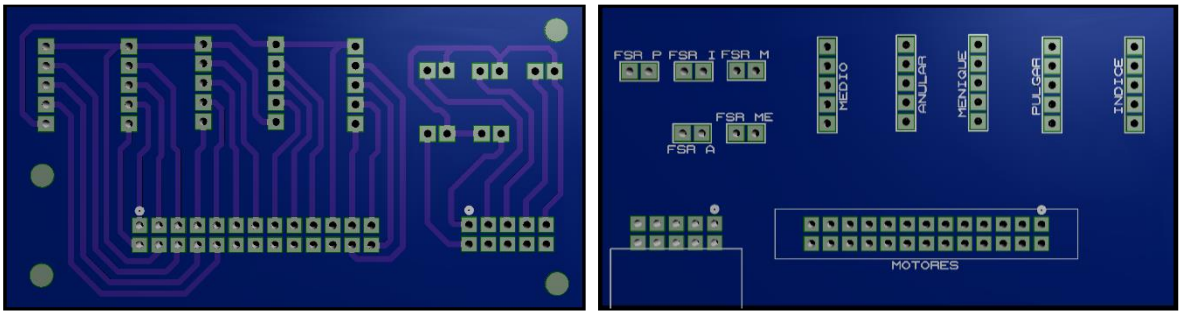


Figura 2.5 Modelo 3D del módulo de periféricos para la prótesis robótica
Fuente: Autores
Elaboración: Autores

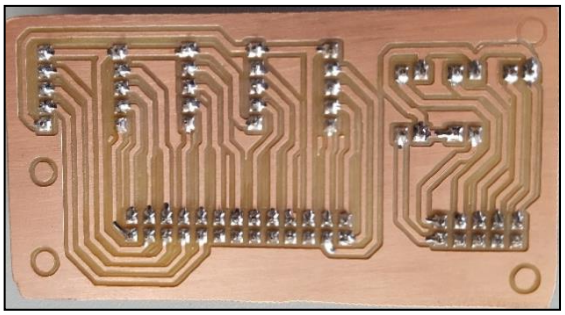


Figura 2.6 Módulo para conexión de periféricos de la prótesis implementado, vista inferior
Fuente: Autores
Elaboración: Autores

2.1.4. Modificación de los dedos y diseño de un domo para acople con el sensor FSR.

Un problema con el uso de los sensores FSR consiste en que no pueden ser flexionados, ya que este fenómeno influye en la respuesta del sensor [22]. Por lo tanto se requiere de una superficie plana en cada dedo a ubicarse el sensor. Con pruebas de agarre de la prótesis robótica construida en [2], se identificó el área donde las yemas de los dedos presionan a objetos cilíndricos, determinando el sitio donde debe ir ubicado el sensor FSR en los dedos de la prótesis.

El primer diseño empleado fue una extrusión con la forma del sensor FSR, la segunda versión se la realizó enfocándonos en ocultar los cables que necesita el sensor llevándolos a la parte posterior del dedo mediante un conducto, finalmente la tercera versión de la modificación de los dedos busca mantener la estética original, por lo que se recubrió parte de la zona rectangular del sensor dejando solamente la zona circular del mismo expuesta.

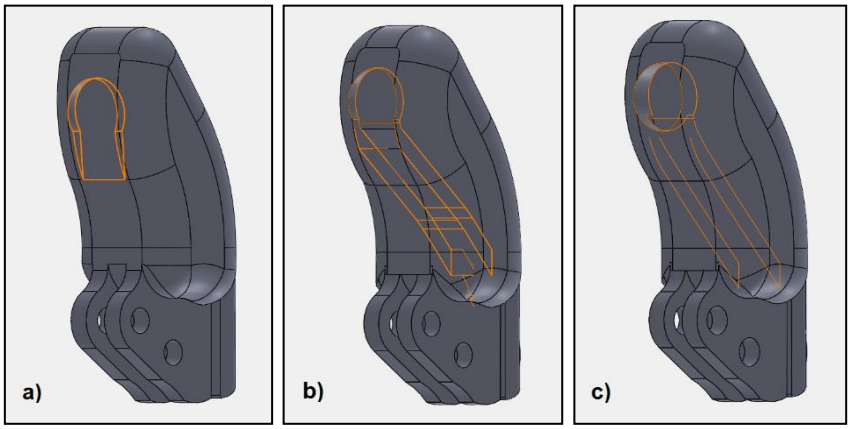


Figura 2.7 Modificación por extrusión del dedo índice para acople del sensor FSR a) versión 1, b) versión 2 , c) versión 3
Fuente: Autores
Elaboración: Autores

En la tercera versión c) de la figura 2.7, el sensor FSR puede acoplarse en una cavidad de 7.76 [mm] de diámetro con una profundidad de 2.47 [mm] desde el punto más elevado de la extrusión. El conducto para el paso de los cables de alimentación del sensor tiene de 6.85 [mm] de ancho, suficiente para el paso de los cables del sensor, en la parte posterior del cada dedo, donde culmina el conducto de los cables, se redujo el tamaño para tener una salida de 2.33 [mm] por 3 [mm].

A continuación, se muestran las modificaciones basadas en la versión tres de la figura 2.7, para acople del sensor FSR en los dedos pulgar, índice y medio.

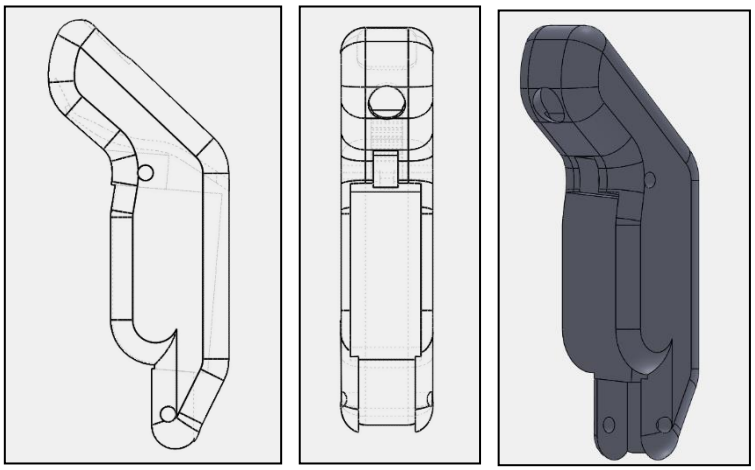


Figura 2.8 Vista lateral, frontal e isométrica del dedo pulgar
Fuente: Autores
Elaboración: Autores

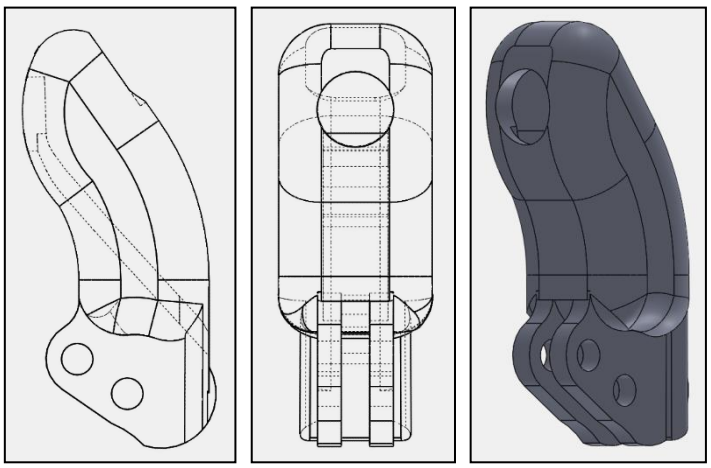


Figura 2.9 Vista lateral, frontal e isométrica del dedo índice
Fuente: Autores
Elaboración: Autores

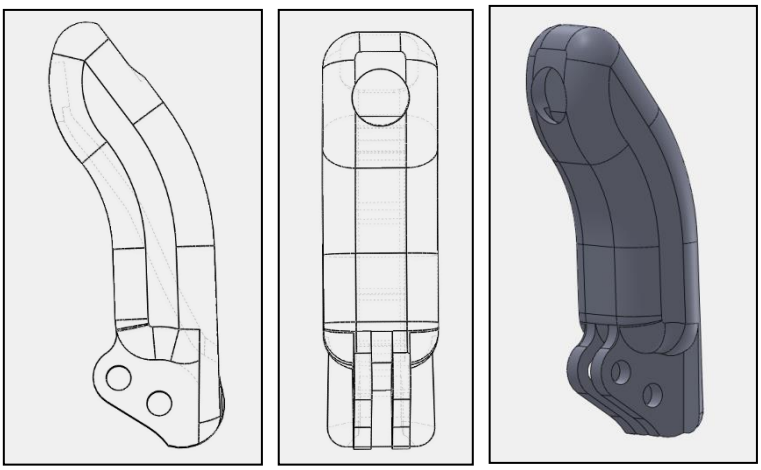


Figura 2.10 Vista lateral, frontal e isométrica del dedo Medio
Fuente: Autores
Elaboración: Autores

Un elemento que se emplea para sensar fuerza mediante el sensor FSR y no presión, es la incorporación de un objeto tipo domo de consistencia firme con un área circular que presione directamente en la zona activa del sensor FSR. Este proceso usado en [44] permite que se detecte una fuerza sin afectar a la sensibilidad o respuesta dinámica del sensor, en base a ello se decidió usarlo en el presente trabajo.

Se partió de un modelo presentado en [22], diseñando un domo con diámetro de 4.2 [mm] menor al del área activa del sensor y una altura de 2.6 [mm]. En ésta primera versión se añadió un segundo diámetro de 7.6 [mm], dando una forma de hongo para corresponder con la circunferencia exterior del sensor y mantener la prótesis estéticamente similar a la desarrollada en [2].

En la figura 2.11 se muestran las cinco versiones correspondientes a los diseños del domo, los cambios entre a), b), c) y d) buscan mejorar el acople del mismo con los dedos de las figuras 2.8 a 2.10. Con la cuarta versión del domo, d), se realizaron pruebas de agarre en las que se determinó una falla de diseño en la que el cilíndrico desplazaba e inclinaba el domo en el punto de colisión, esto llevó al aumento del diámetro inferior del domo a 5.4 [mm] para resolver el problema de desplazamiento del domo.

Para fijar cada sensor FSR a los dedos de la prótesis robótica se usó silicona para sensores marca ABRO [45] de uso profesional y no corrosiva, mientras que el domo se fijó al sensor FSR con cinta de doble cara de 100 micrómetros de espesor.

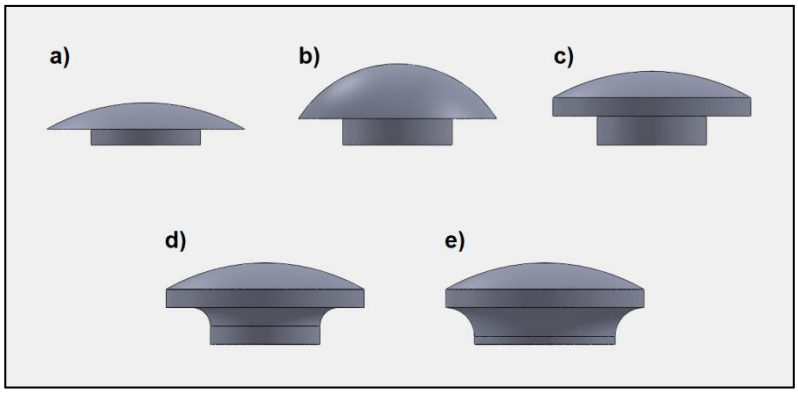


Figura 2.11 Versiones del domo, elemento de presión para el sensor FSR
Fuente: Autores
Elaboración: Autores

En el anexo C se muestra el efecto que tiene el cambio del domo entre la versión d) y e) de la figura anterior.

La fabricación de los dedos y el domo se realiza en una impresora 3D MakerBot modelo Replicator 2X con plástico ABS como materia prima [46].

2.2. Firmware.

En esta etapa se procedió a diseñar los filtros digitales y las funciones en lenguaje C que forman la arquitectura del firmware, tanto para el control basado en medición de corriente (ver figura 1.10), como en el control basado en medición de fuerza (ver figura 1.12).

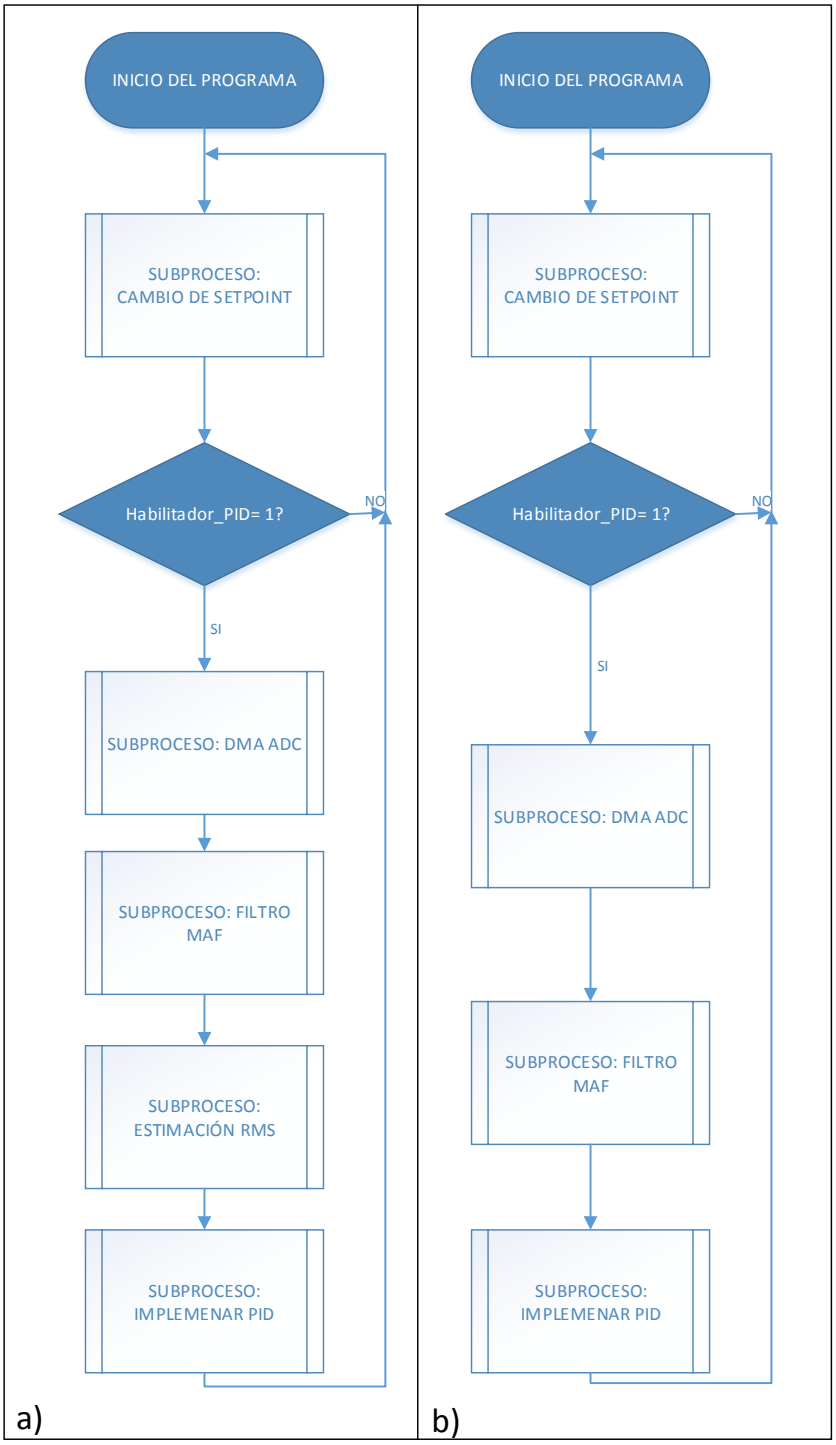


Figura 2.12 Diagrama de flujo general: a) Control basado en corriente, b) Control basado en fuerza
Fuente: Autores
Elaboración: Autores

En la figura 2.13, se muestran los diagramas de flujo del firmware de estas dos metodologías para el control de fuerza de agarre, la diferencia fundamental es la adición de una función para la estimación del valor eficaz o valor RMS de la señal de corriente que mejora la robustez del lazo de control basado en este parámetro. El código fuente desarrollado en lenguaje C se encuentra en el anexo B.
En los siguientes apartados se describe a profundidad cada una de los subprocesos.

2.2.1. DMA ADC.

Este subproceso está basado en una interrupción del conversor analógico digital del microcontrolador ATSAM3X8E, esta interrupción permite leer todos los canales de conversión analógica y los valores son almacenados en un buffer para su posterior procesamiento.

Como se muestra en el diagrama de flujo de este subproceso (ver figura 2.13), el microcontrolador espera en la interrupción hasta que todos los canales ADC sean leídos. Una vez culminado este paso, mediante acceso directo a memoria (DMA), se copia los valores de conversión en otro buffer para su procesamiento en los siguientes subprocesos. Al terminar el traspaso de los datos, el programa vuelve a su flujo principal.

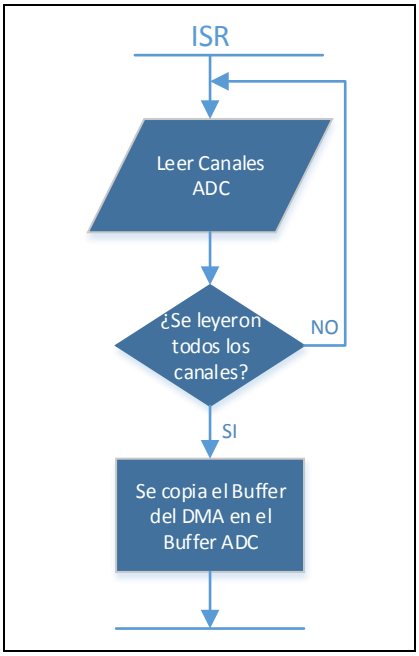


Figura 2.13 Diagrama de flujo del subproceso: DMA ADC
Fuente: Autores
Elaboración: Autores

El voltaje de referencia del conversor analógico digital es de 3.3 [V] y su resolución es de 12 bits, por ende, el mínimo de voltaje medido es de 0.8 [mV], a diferencia del trabajo previo en

donde el mínimo de voltaje medido era de 4.88 [mV] [2], lo cual asegura que la medición sea precisa y contenga menos ruido por la cuantización, como se describe en [27].

Este subproceso se aplica tanto para el método basado en medición de corriente como para el método basado en medición de fuerza sensada a través de los FSR.

2.2.2. Filtro MAF.

2.2.2.1. Diseño.

Como se muestra en la figura 2.14, la señal de salida del filtro pasa-bajas activo (ver figura 1.5) contiene ruido, esto debido a la conmutación producida por la señal de voltaje PWM que alimenta al actuador [28], dado que la corriente que pasa a través del actuador es directamente proporcional al voltaje al cual está sometido [25], la señal de corriente obtenida a través del resistor shunt muestra el mismo efecto.

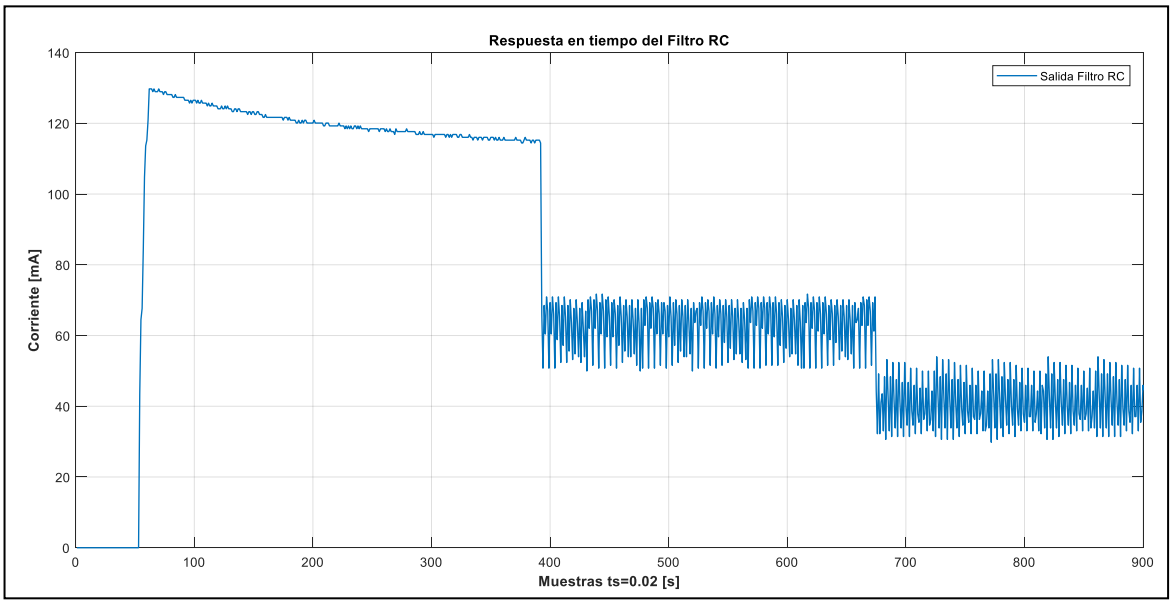


Figura 2.14 Salida del filtro RC del voltaje en función de la corriente consumida por el actuador
Fuente: Autores
Elaboración: Autores

En la figura anterior, la corriente obtenida al alimentar los actuadores con el 100% del ciclo de trabajo del PWM no muestra oscilaciones, sin embargo, al variar este parámetro se generan tales oscilaciones. Para el correcto diseño del filtro MAF que permita atenuar la magnitud de estos armónicos se debe obtener la FFT de la señal, este proceso muestra la frecuencia en la cual se encuentran (ver figura 2.15), para ello se usó el software Matlab®.

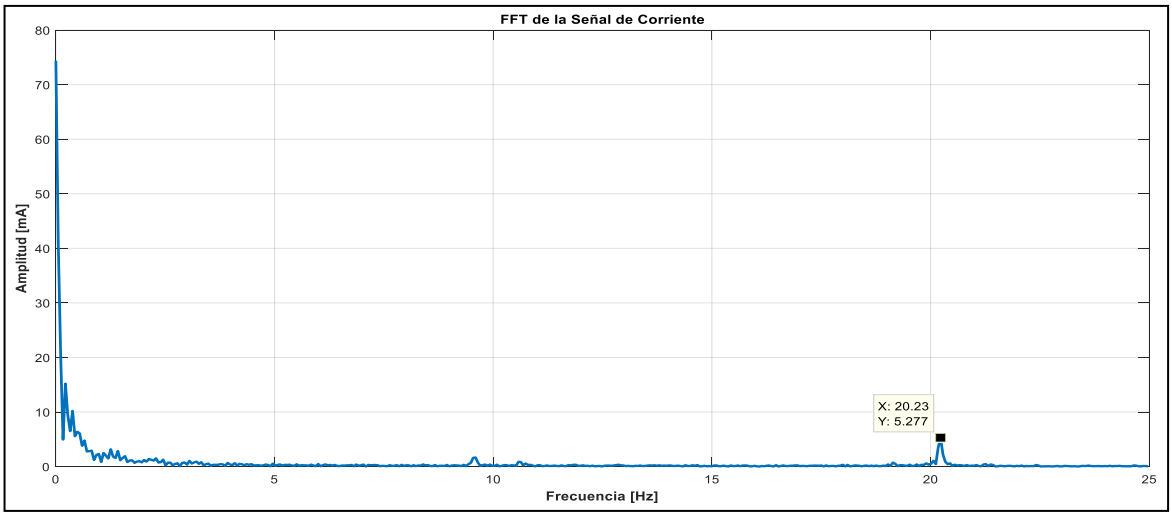


Figura 2.15 Transformada rápida de Fourier de la señal de corriente
Fuente: Autores
Elaboración: Autores

Como se observa (ver figura 2.15), los armónicos de la señal de corriente poseen una frecuencia de 20.23 [Hz], afirmando que se cumple con el criterio de Nyquist [47], ya que esta señal se ha muestreado con una frecuencia de 50 [Hz], cumple con los la siguiente relación [47]:

$$f_{muestreo} > 2\, f_{señal} \qquad (2.1)$$

Donde:

$f_{muestreo}$: Es la frecuencia de muestreo.

$f_{señal}$: Es la frecuencia de la señal a muestrear.

Con la frecuencia obtenida, se procede al diseño del filtro MAF con el objeto que la atenuación sea máxima en esa frecuencia. En la figura 2.16, se muestra que un filtro de 5to orden MAF cumple con esta tarea, obteniendo una atenuación de -59.63 [dB] de los armónicos.

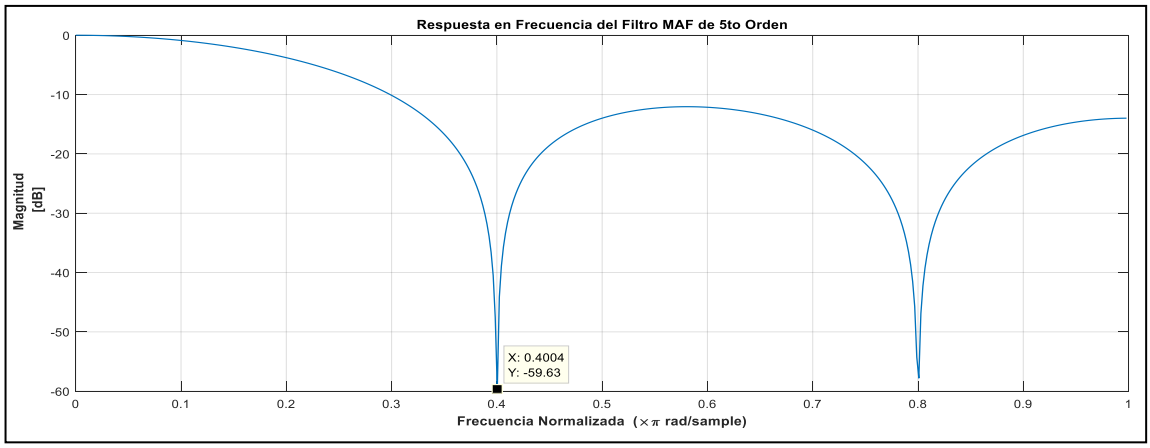


Figura 2.16 Respuesta en frecuencia del filtro MAF de 5to orden
Fuente: Autores
Elaboración: Autores

Al pasar la señal de salida del filtro RC activo por el filtro MAF de 5to orden, se muestra que los armónicos se han atenuado considerablemente (ver figura 2.17).

Con el filtro diseñado, se procede a implementarlo en el firmware del sistema embebido. Para el caso de los sensores FSR, se usa un filtro MAF de 2do orden como un filtro de suavizado.

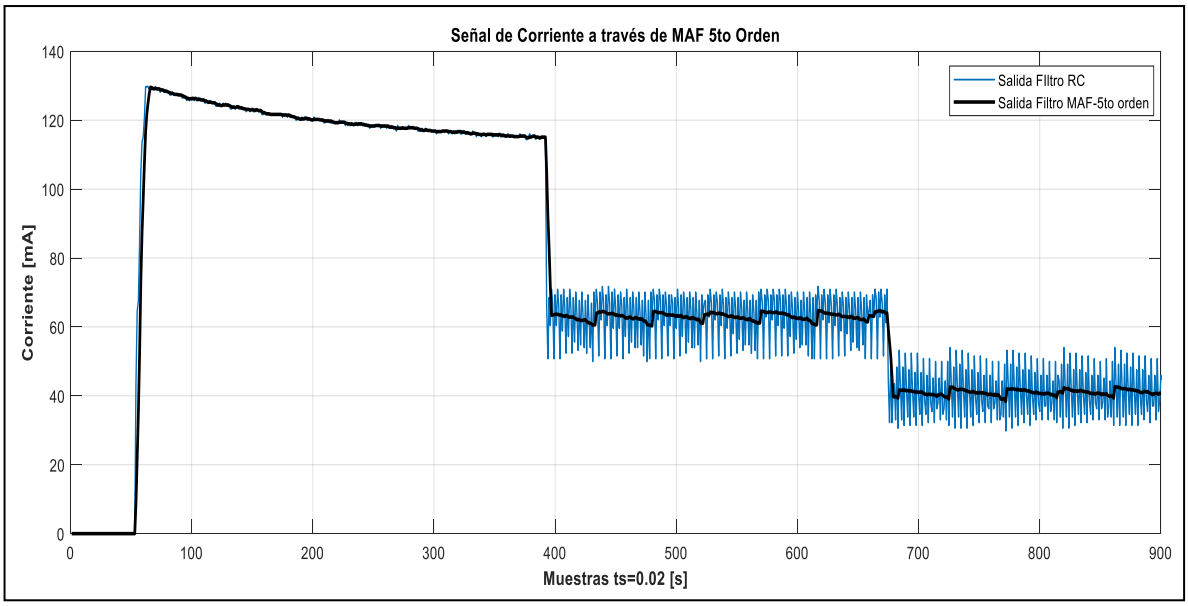


Figura 2.17 Señal de corriente filtrada usando MAF de 5to orden
Fuente: Autores
Elaboración: Autores

2.2.2.2. Implementación.

En el Firmware este filtro se implementa en un subproceso, que se ejecuta una vez obtenidas las señales mediante el ADC (ver figura 2.18).

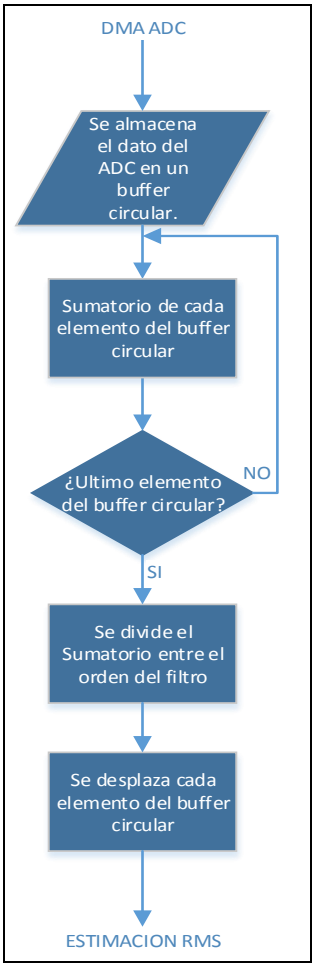


Figura 2.18 Diagrama de flujo del subproceso: filtro MAF
Fuente: Autores
Elaboración: Autores

El coste computacional de este filtro depende de su orden, ya que de ello realizará un número específico de sumas, como se muestra en la fórmula 1.5. En la figura 2.19 se muestra la diferencia del número de operaciones del filtro MAF de 5to y 2do Orden.

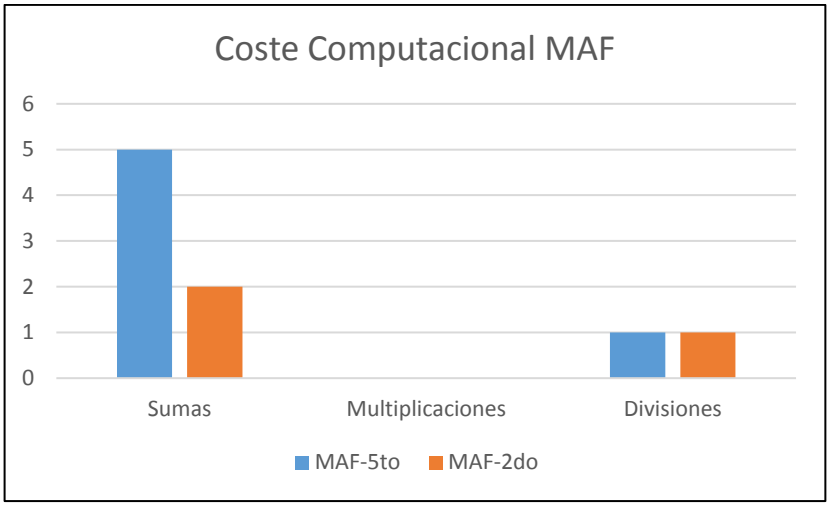


Figura 2.19 Coste computacional algoritmo MAF
Fuente: Autores
Elaboración: Autores

2.2.3. Estimación RMS.

2.2.3.1. Diseño.

Para obtener el valor eficaz de la señal de corriente es necesario aplicar una estimación RMS, y la razón de estimar este parámetro es el proveer al lazo de control basado en medición de corriente una señal estable en su retroalimentación.

En [29], se recomienda que se estime el RMS tomando en cuenta un mínimo de 12 ciclos para señales con frecuencia de 60 [Hz] y 10 para señales de 50[Hz]. Del diseño del filtro MAF encontramos que la señal de corriente tiene armónicos en la frecuencia de 20.23 [Hz], si asumimos una relación proporcional, determinamos que se necesita al menos cuatro ciclos para la correcta estimación RMS.

Con el número de ciclos podemos obtener el ancho de la ventana necesaria para una correcta estimación RMS, usando la siguiente formula [48]:

$$N = \frac{M f_{muestreo}}{f_{señal}} \qquad (2.2)$$

Donde:

N: Es el ancho de la ventana

$f_{señal}$: Frecuencia de la señal

$f_{muestreo}$: Es la frecuencia de muestreo

M: Es el número de ciclos de la señal

En nuestro caso, la ventana mínima requerida es de 10 muestras, pero en la práctica se usarán 3 veces el mínimo de ciclos es decir 12, por esto la ventana recomendada es de 30 muestras. Reemplazando esta ventana en la fórmula 1.6, se obtiene las siguientes señales filtradas (ver figura 2.20).

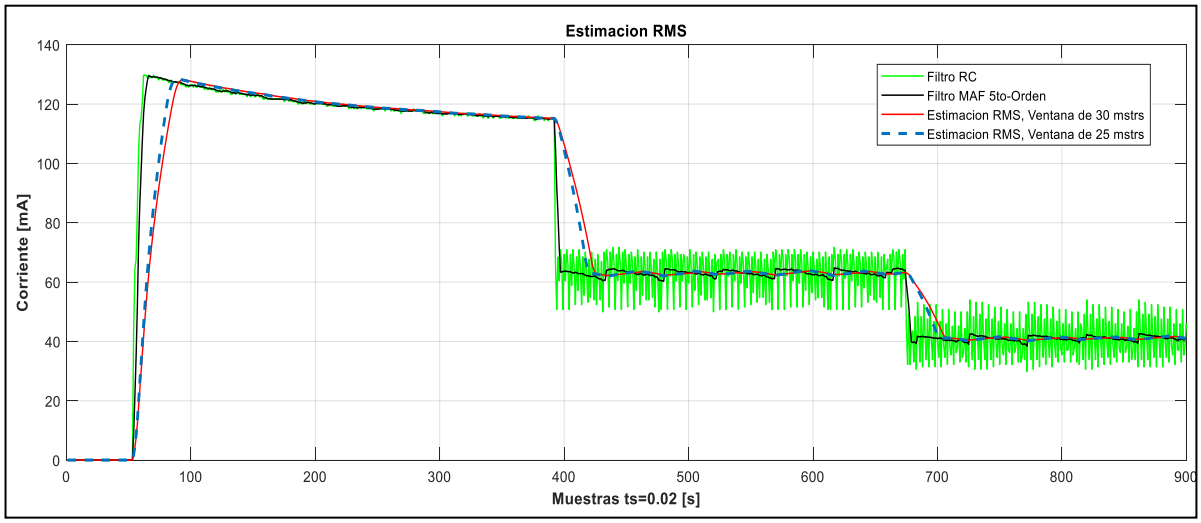


Figura 2.20 Estimación RMS, ventanas de 25 y 30 muestras
Fuente: Autores
Elaboración: Autores

Otra forma de estimación RMS se basa en EWMA (Exponentially Weighted Moving Average), un método recursivo cuya fórmula es [49]:

$$x_{rms_{N,\lambda}} = \sqrt{\left(1 - \frac{1}{w_{N,\lambda}}\right) x^2_{rms_{N-1,\lambda}} + \left(\frac{1}{w_{N,\lambda}}\right) x^2_N}\,, \tag{2.3}$$

$$w_{N,\lambda} = \lambda w_{N-1,\lambda} + 1 \tag{2.4}$$

Donde:

$x_{rms_{N,\lambda}}$: RMS de la muestra actual.

x^2_N: Muestra actual de entrada al cuadrado.

$x^2_{rms_{N-1,\lambda}}$: RMS de la muestra anterior al cuadrado.

λ: Factor de olvido.

$w_{N,\lambda}$: Factor de ponderación que se aplica a la muestra actual, igual a 1 en la primera iteración.

Esta fórmula (2.3), toma en consideración la estimación anterior y los datos nuevos de ingreso. El factor de olvido puede variar en el rango $0 \leq \lambda \leq 1$, el valor escogido es $\lambda = 0.925$, ya que se acerca al resultado de la ventana de 30 muestras (ver figura 2.21).

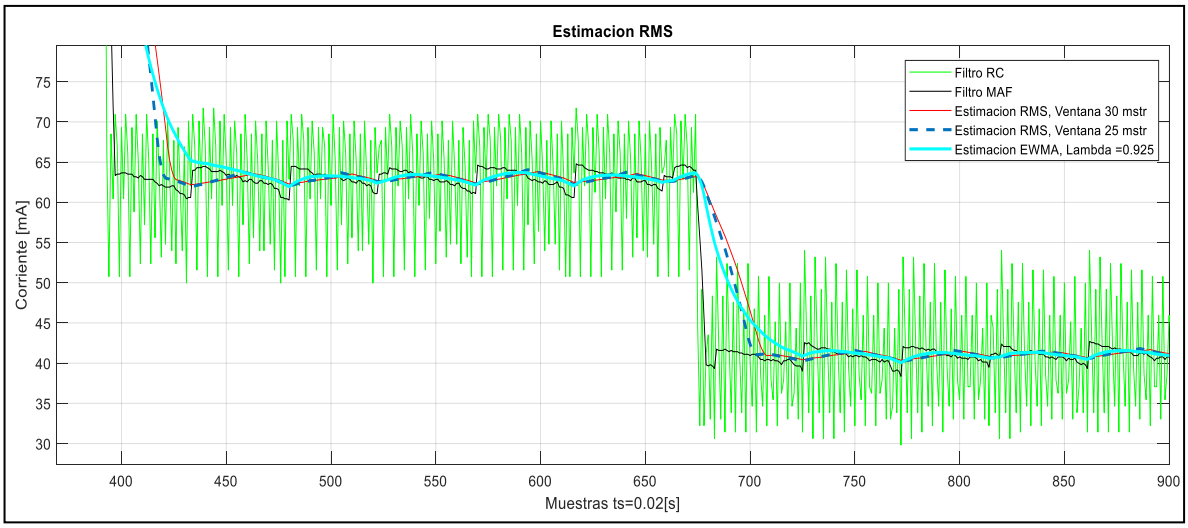


Figura 2.21 Estimación RMS usando EWMA $\lambda = 0.925$
Fuente: Autores
Elaboración: Autores

La elección del método RMS EWMA frente al RMS en ventanas, queda justificado por el coste computacional (ver figura 2.22). El método RMS EWMA se recomienda para aplicaciones en sistemas embebidos [49]. Además, se debe tomar en cuenta que este algoritmo debe ejecutarse para cada actuador de la prótesis robótica, lo que es una razón de peso para usar este método.

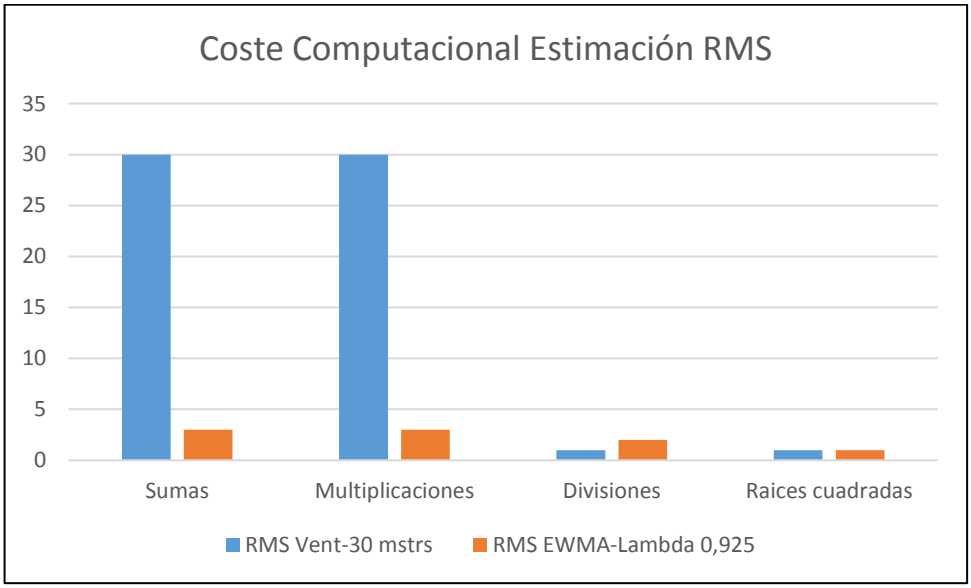


Figura 2.22 Coste computacional de los tipos de estimación RMS
Fuente: Autores
Elaboración: Autores

2.2.3.2. Implementación.

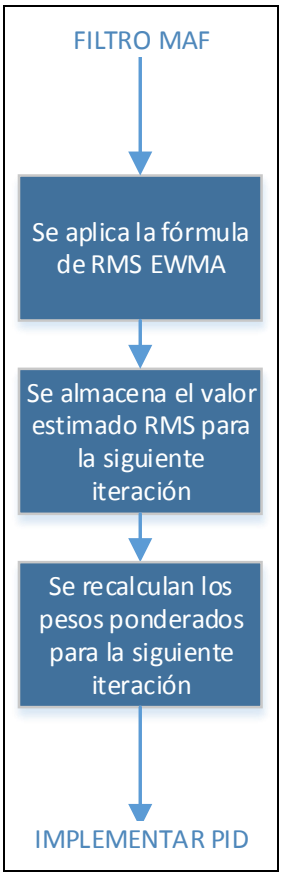


Figura 2.23 Diagrama de flujo del subproceso: Estimación RMS
Fuente: Autores
Elaboración: Autores

En firmware, la estimación del RMS se realiza como un subproceso. Las señales de entrada son las de salida del filtro MAF, y la salida sirve como señal de retroalimentación para el subproceso que se encarga de implementar el algoritmo PID (ver figura 2.23).

2.2.4. Mapeo FSR.

El bloque que implementa el controlador PID utiliza en todo momento variables en unidades de milivoltios para el caso de la retroalimentación con sensores FSR, por lo tanto, existen dos posibilidades para ingresar el set-point, en milivoltios o, si requiere que este dado en fuerza se debe realizar una conversión de unidades con un mapeo de Newtons a Voltios. El bloque descrito en este apartado no es más que una ecuación polinomial a la cual se le ingresa un dato en Newtons y devuelve su equivalente en Voltios, mismo que debe ser reescalado a milivoltios.

El proceso para determinar la ecuación que obedece a los sensores FSR se divide en dos:

- Obtención de los datos de calibración.
- Regresión polinomial.

Cada sensor FSR debe ser calibrado como se recomienda en [22], [44] y [50]; a pesar de las especificaciones dadas por el fabricante cada sensor varía su respuesta a la presión dependiendo de factores como: la precisión en la que fue pegado el domo en el área activa de los sensores, si el sensor ha sido flexionado, por el pegamento usado para adherir el sensor en los dedos.

Para la obtención de los datos de calibración es necesario un equipo de evaluación de modo que se construyó un dispositivo similar al utilizado en [22], dotado de un motor lineal Firgelli PQ12 mismo que se usa en la prótesis de [2], y una galga extensiométrica HT, modelo TAL220 [51] permitiendo la obtención de la fuerza aplicada por el motor en el sensor FSR. El mecanismo de compresión se realiza aumentando la fuerza del motor gradualmente regulando el PWM, el motor posee una base plana al final del vástago. Además se utilizó un Arduino MEGA [52] para adquirir la señal de fuerza y generar la señal PWM, y para el control de este equipo se diseñó un software basado en leguaje C (anexo B).

Con una balanza de precisión KERN EMS [53] se calibró la galga extensiométrica.

En la figura 2.24 se muestra el dispositivo de calibración implementado.

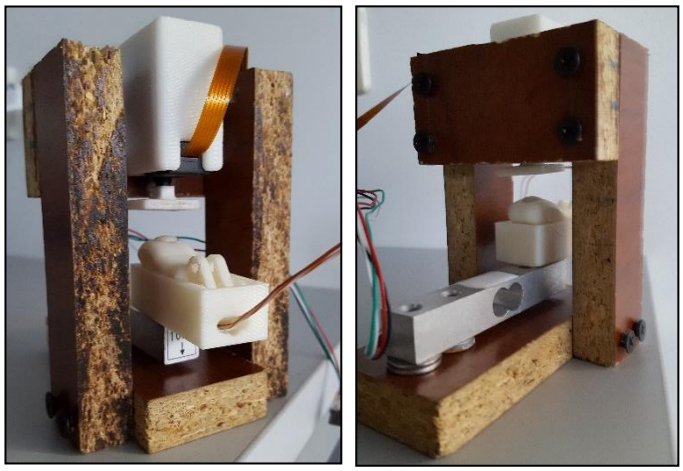


Figura 2.24 Implementación del dispositivo de calibración para los sensores FSR
Fuente: Autores
Elaboración: Autores

Los sensores tienen que ser calibrados en cada dedo puesto que, si son calibrados previamente, es probable que las flexiones que sufra el sensor por la manipulación afecten a la respuesta una vez fijado en los dedos. Para compensar este efecto es elemental un

molde negativo de cada dedo, situando la cara del sensor FSR transversal a la base plana del vástago del motor. En la figura 2.25 se muestran los moldes con cada uno de los dedos acoplados.

Con el dispositivo se logra calibrar un sensor FSR en aproximadamente 40 segundos, la fuerza máxima que aplica el motor lineal sobre el sensor es de 40 Newtons, sin embargo, por la dinámica de la prótesis robótica la fuerza que ejerce el actuador no corresponde a la fuerza de agarre.

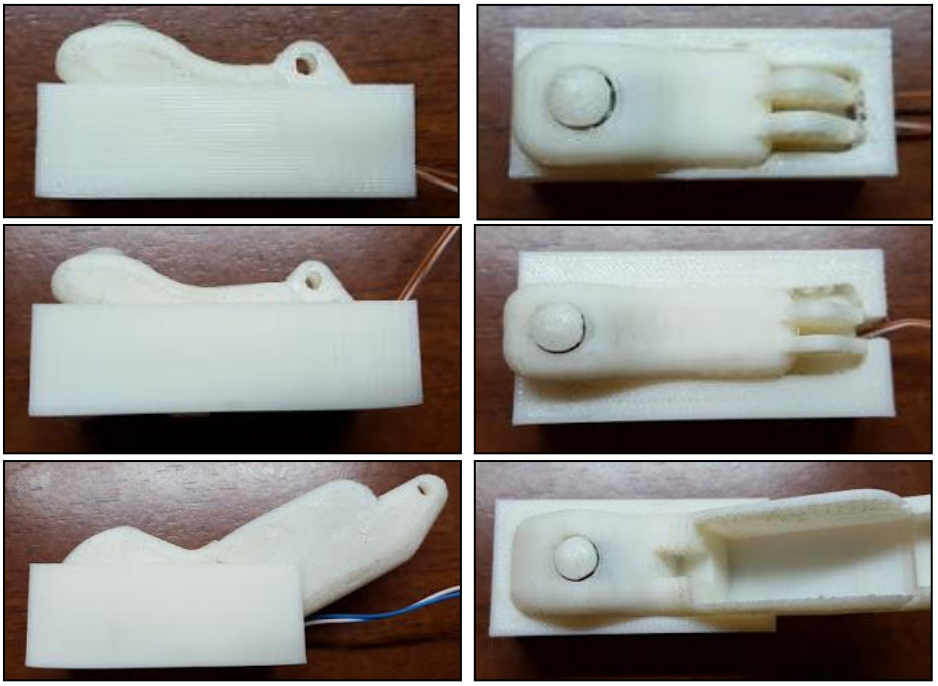


Figura 2.25 Moldes para acople transversal del sensor FSR con el vástago del dispositivo de calibración
Fuente: Autores
Elaboración: Autores

Posterior a la obtención de los datos de calibración, se realiza un modelamiento por regresión polinomial aplicando un software (anexo B) desarrollado en Matlab® que identifica, considerando cuatro decimales, el valor del coeficiente de Pearson al cuadrado (R^2) más cercano a la unidad evaluando desde el primer orden hasta el séptimo orden polinomial. Se limita los datos de calibración 2 Newtons sobre la fuerza máxima que ejerce cada dedo al agarrar un objeto cilíndrico.

Se descartaron las ecuaciones de primer, segundo y tercer orden por tener un valor R^2 inferior al obtenido con ecuaciones de orden superior, y las ecuaciones de quinto a séptimo orden porque su coeficiente R^2 coincide con el valor de la ecuación de cuarto orden. Por consiguiente, se estipuló utilizar las ecuaciones determinadas por regresión de cuarto orden.

Las siguientes ecuaciones se implementan para construir el bloque de mapeo en el ARM® Cortex®-M3,

$$V_{Indice} = 0.0013245 * N^4 - 0.0010114 * N^3 - 0.044065 * N^2 + 0.58453 * N - 0.032838 \quad (2.5)$$

$$V_{Medio} = 0.00019276 * N^4 - 0.0044178 * N^3 + 0.021655 * N^2 + 0.21473 * N - 0.35918 \quad (2.6)$$

$$V_{Pulgar} = 0.012399 * N^4 - 0.13361 * N^3 + 0.43799 * N^2 - 0.24423 * N + 0.055308 \quad (2.7)$$

Donde

V_{Indice}: Es el voltaje medido del sensor del dedo índice.

V_{Medio}: Es el voltaje medido del sensor del dedo medio.

V_{Pulgar}: Es el voltaje medido del sensor del dedo pulgar.

N: Es el set-point deseado dado en Newton.

Los valores del coeficiente de Pearson al cuadrado correspondientes son:

$$R^2_{Indice} = 0.9962\,;\ R^2_{Medio} = 0.9976\,;\ R^2_{Pulgar} = 0.9958$$

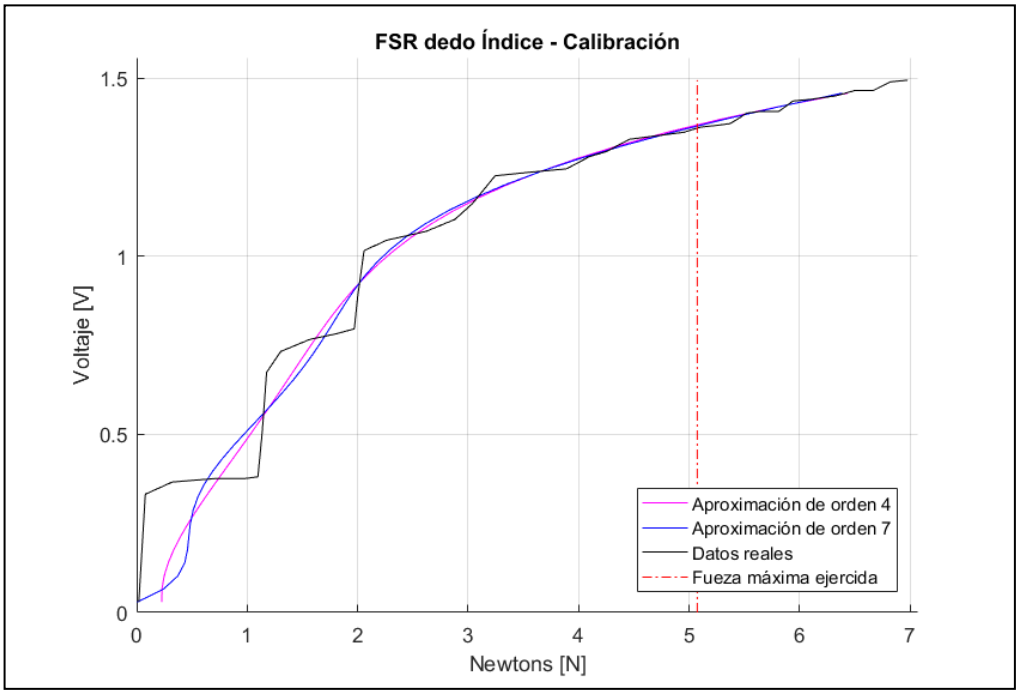


Figura 2.26 Curva de calibración del dedo índice
Fuente: Autores
Elaboración: Autores

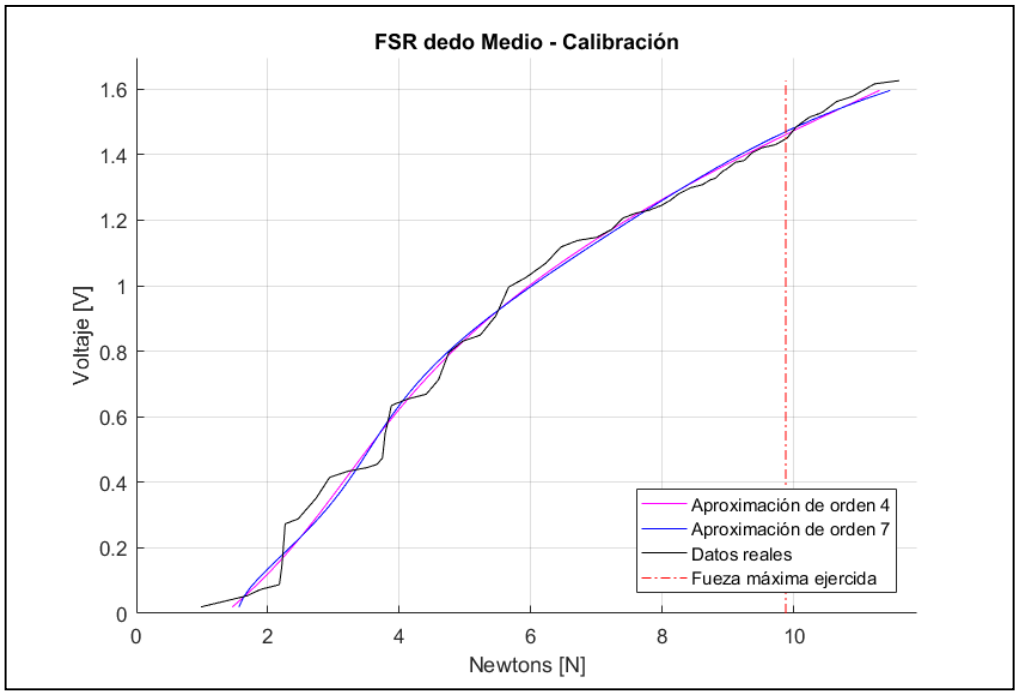


Figura 2.27 Curva de calibración del dedo medio
Fuente: Autores
Elaboración: Autores

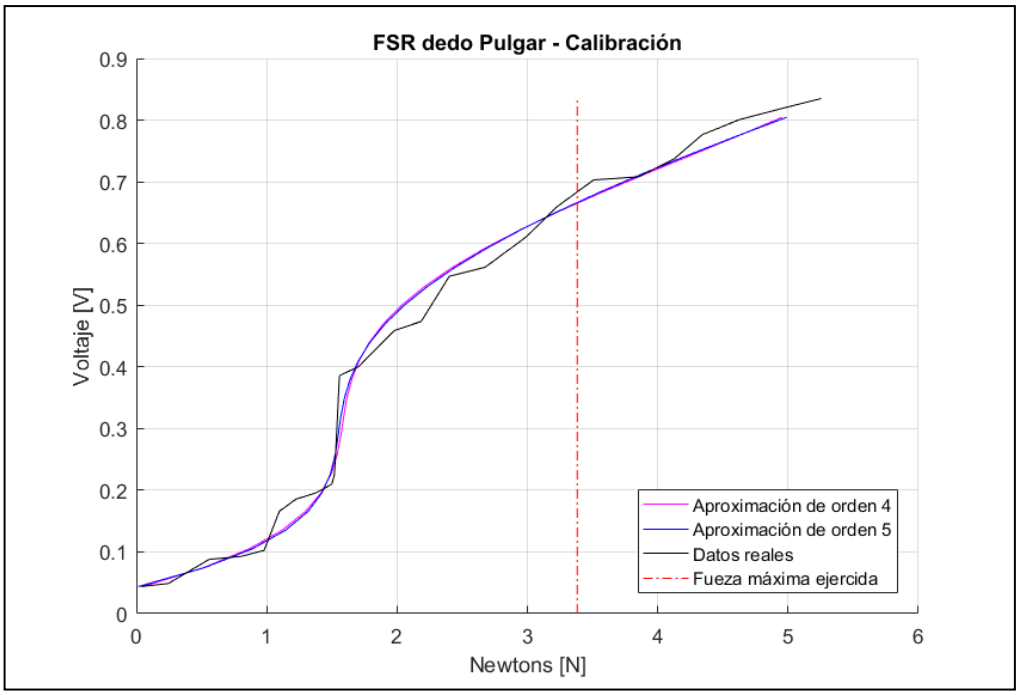


Figura 2.28 Curva de calibración del dedo pulgar
Fuente: Autores
Elaboración: Autores

Como se observa en las figuras anteriores, existe una aproximación de mayor orden para cada calibración, esta corresponde a la aproximación polinomial considerando los datos completos leídos por el dispositivo de calibración. En el anexo D se muestran las curvas de calibración sin limitaciones de fuerza.

2.2.5. Implementación del controlador PID en la tarjeta de control.

El controlador digital se ejecuta como un subproceso en el firmware del sistema embebido, en donde la señal de retroalimentación, en el caso del control basado en corriente es la estimación RMS, y en el caso del control basado en el sensor FSR, la señal de salida del filtro MAF de 2do orden. La salida de este controlador es un valor de PWM entre 0 a 255 y la dirección en el cual se moverá el actuador (ver figura 1.10 y 1.12).

Para el correcto diseño de este subproceso, se debe tomar en cuenta evitar overflow de las variables que son parte del cálculo de la salida del controlador. El diagrama de flujo se muestra a continuación (ver figura 2.29).

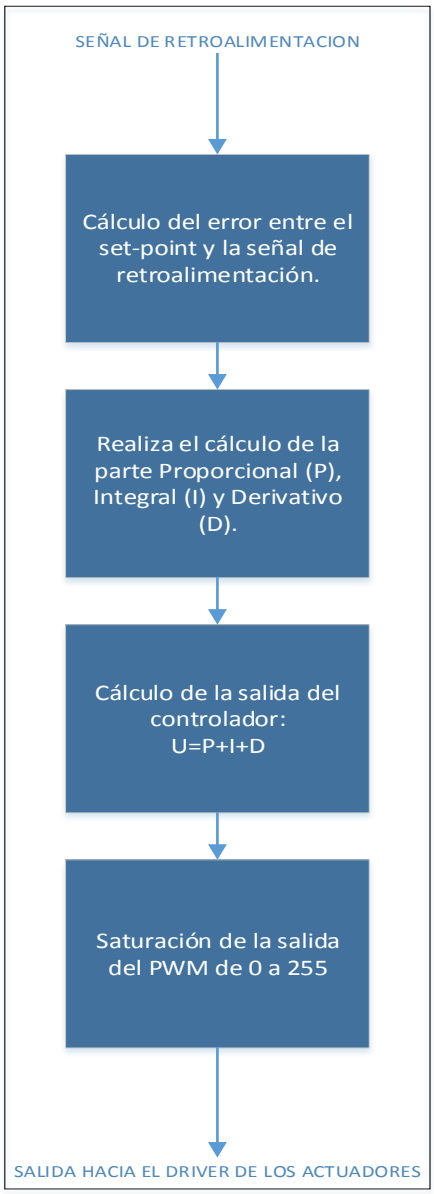


Figura 2.29 Diagrama de flujo del subproceso: Implementar PID
Fuente: Autores
Elaboración: Autores

2.2.6. Cambio de set-point.

Dado que el cambio de set-point debe poder realizarse en el momento que se desee, se lo implemento a este subproceso mediante una interrupción del UART. Además, para evitar cambios imprevistos mientras se realiza el subproceso, posee una bandera que sirve como habilitador para los restantes subprocesos, es decir mientras se esté gestionando el cambio de set-point no se podrá ejecutar ninguna otra acción. El diagrama de flujo de este subproceso se muestra a continuación (ver figura 2.30).

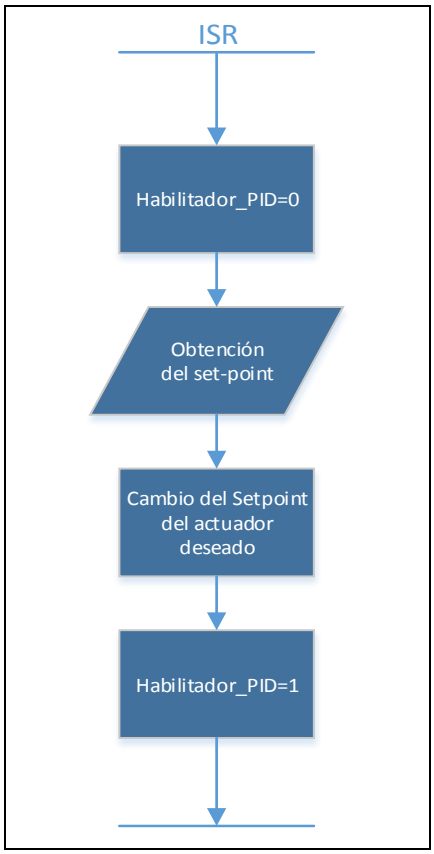


Figura 2.30 Diagrama de flujo del subproceso: Cambiar set-point
Fuente: Autores
Elaboración: Autores

Con la función descrita en la figura 2.30 concluye la parte del firmware del sistema embebido, todos los códigos tanto para el diseño, como el código implementado se encuentran en el anexo B.

CAPÍTULO III

DISEÑO Y SINTONIZACION DE LOS LAZOS DE CONTROL

3.1. Control de fuerza de agarre basado en medición de corriente.

El modelo matemático para los sistemas de control de fuerza de agarre, con señales de retroalimentación de corriente y fuerza se determinan mediante la función System Identification que forma parte de la aplicación PID Tuner de Matlab®. Para este estudio se denomina como sistema a cada dedo de la prótesis robótica con el sensor correspondiente.

System Identification estima los parámetros del sistema minimizando el error entre la salida del modelo y la respuesta medida del sistema. Este error, llamado función de pérdida es una función positiva de predicción de errores, el método para modelamiento del sistema se detalla en [54].

Los parámetros considerados para la evaluación del control PI son: porcentaje de overshoot no superior al 5% y el menor tiempo de establecimiento, tal que superen al conseguido en el trabajo anterior [2], estos parámetros se evaluarán entre los diferentes métodos de sintonización aplicados.

3.1.1. Identificación del sistema.

El proceso de identificación del sistema para el control de fuerza en base a la corriente, se realiza para cada dedo de la prótesis robótica ya que, por su dinámica, la respuesta ante un escalón de entrada varía en cada uno de estos [2].

Para el dedo pulgar, en la figura 3.1 se observa las respuestas a un escalón de entrada para el sistema real y para el modelo matemático. La similitud (Fit) entre estas dos señales se obtiene en porcentaje [54], en este caso es de 96.07% para el sistema perteneciente al dedo pulgar.

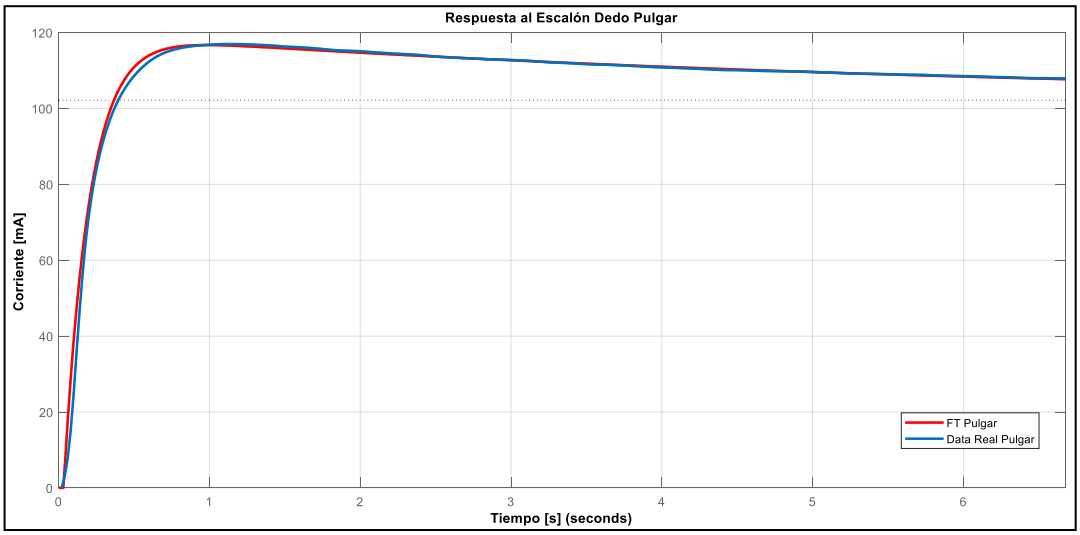
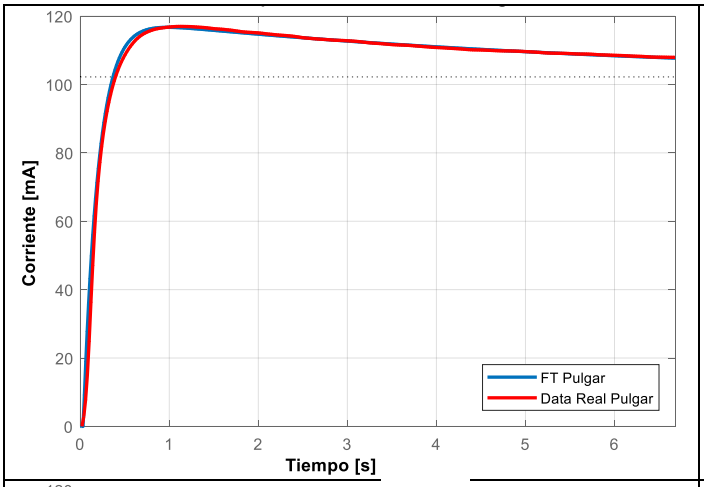
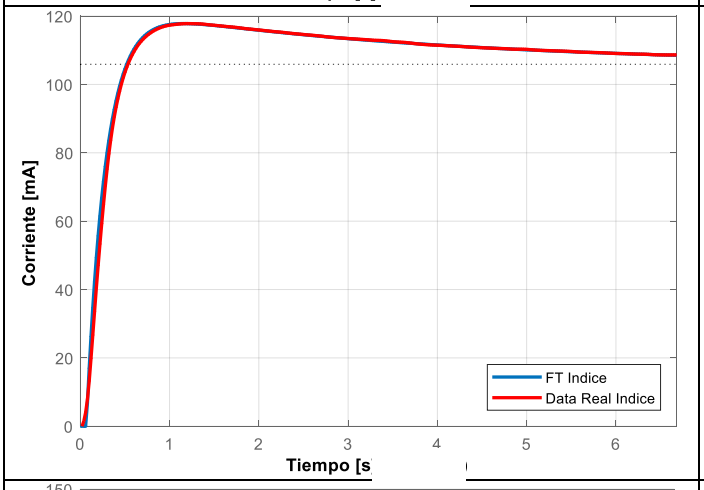
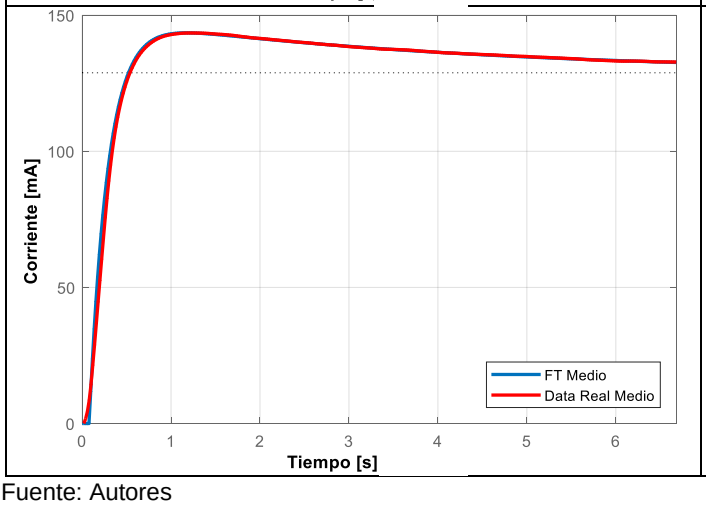


Figura 3.1 Respuesta al escalón de entrada del dedo pulgar, real vs modelo matemático
Fuente: Autores
Elaboración: Autores

En la tabla 3.1, se resume los modelos matemáticos obtenidos en esta etapa para cada dedo, junto a la respuesta al escalón de estas y del sistema real.

Tabla 3.1 Modelos matemáticos y su respuesta a un escalón de entrada de los sistemas basados en corriente

	Modelo matemático en lazo abierto del dedo pulgar. $G(s)_{pulgar} = \frac{2.669s + 0.4008}{0.9656s^2 + 5.873s + 1} e^{-0.0334s}$ $Fit \rightarrow 96.07\%$
	Modelo matemático en lazo abierto del dedo índice. $G(s)_{índice} = \frac{1.671s + 0.4153}{0.7558s^2 + 3.705s + 1} e^{-0.0675s}$ $Fit \rightarrow 96.04\%$
	Modelo matemático en lazo abierto del dedo medio. $G(s)_{medio} = \frac{2.278s + 0.5052}{0.8256s^2 + 4.13s + 1} e^{-0.0799s}$ $Fit \rightarrow 94.78\%$

Fuente: Autores
Elaboración: Autores

3.1.2. Sintonización de los controladores PID.

Obtenido el modelo matemático de cada sistema, se diseña el controlador de tipo PI usando los métodos descritos en el capítulo 2.

3.1.2.1. Dedo pulgar.

Para el dedo pulgar, se procedió a buscar la ganancia critica (K_{cr}) que fuerce al sistema a entrar en oscilaciones mantenidas en lazo cerrado (ver figura 3.2), de lo cual se obtiene también el periodo crítico en segundos (T_{cr}).

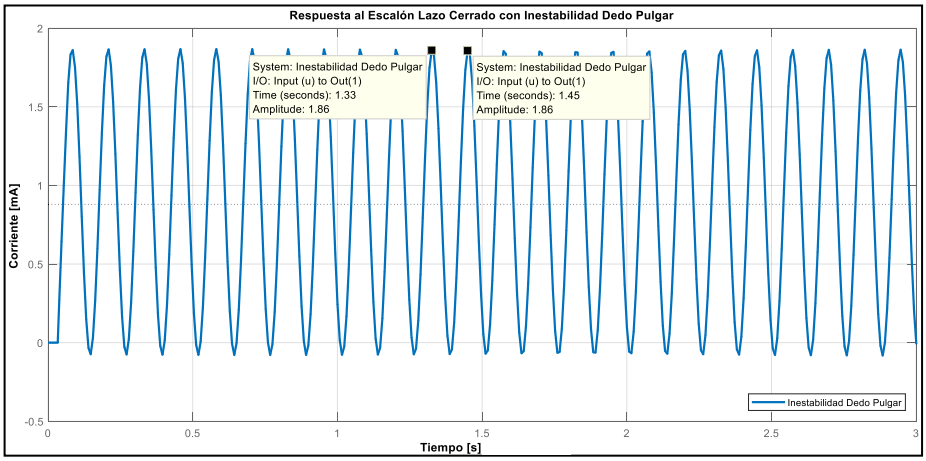


Figura 3.2 Inestabilidad en lazo cerrado del dedo pulgar
Fuente: Autores
Elaboración: Autores

El valor de cada variable para la sintonización en lazo abierto y lazo cerrado del sistema del dedo pulgar, se resumen a continuación:

Lazo cerrado:

$$K_{cr} = 18.392 \quad (3.1)$$

$$T_{cr} = 0.13 \quad (3.2)$$

Lazo abierto:

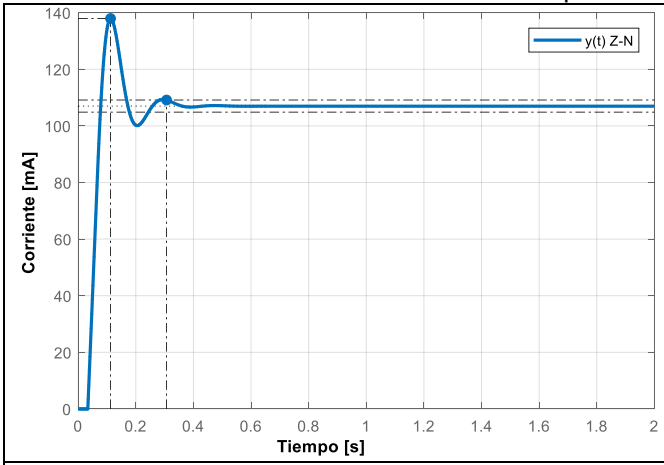
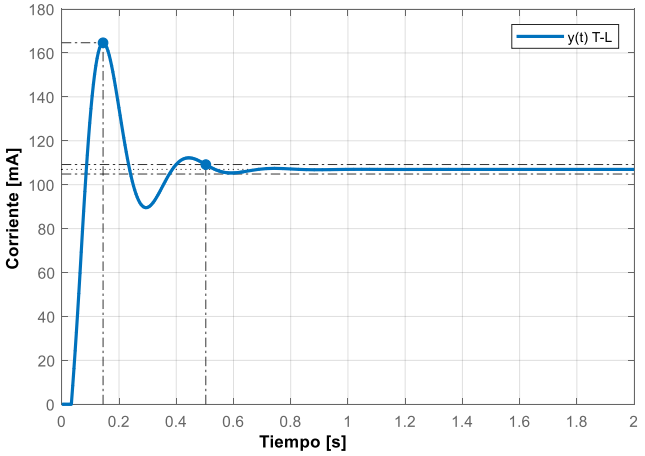
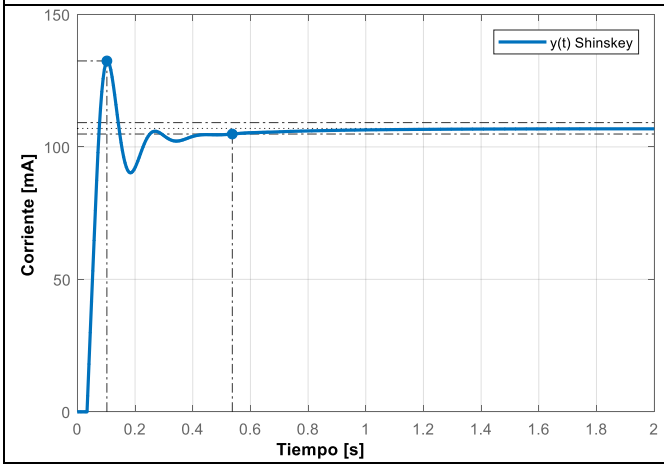
$$G_p = 0.4008 \quad (3.3)$$

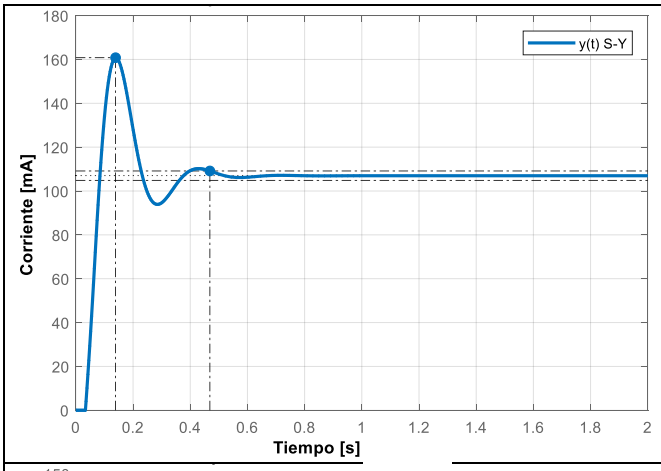
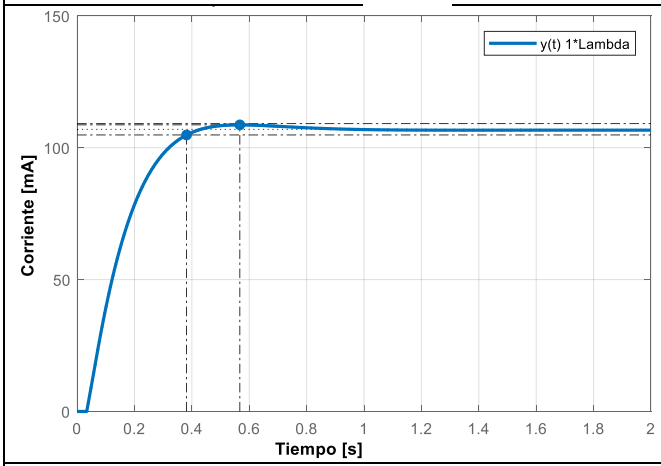
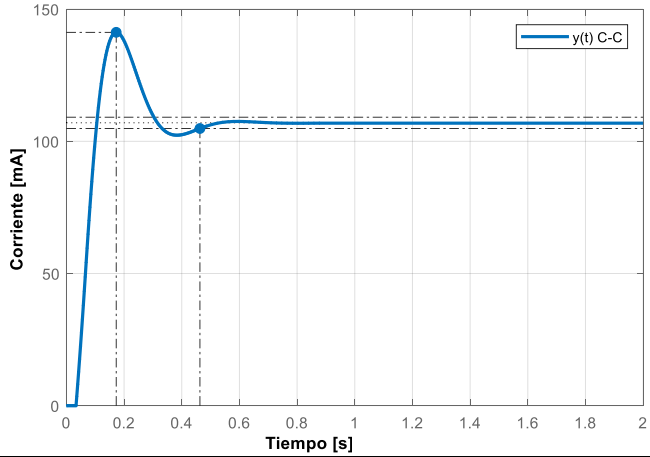
$$\tau = 0.135 \quad (3.4)$$

$$\theta = 0.03342 \quad (3.5)$$

En la tabla 3.2, se resume las ganancias obtenidas y la respuesta al escalón de 107 [mA] de cada método de sintonización.

Tabla 3.2 Sintonización de los controladores PI para el sistema del dedo pulgar

	Controlador PI sintonizado con Ziegler-Nichols Ganancias - Kp = 8.276 - Ki = 53.05 Controlador $$C(s) = \frac{8.276s + 53.05}{s}$$
	Controlador PI sintonizado con Tyreus-Luyben Ganancias - Kp = 5.747 - Ki = 98.25 Controlador $$C(s) = \frac{5.747s + 98.25}{s}$$
	Controlador PI sintonizado con Shinskey Ganancias - Kp = 9.196 - Ki = 32.15 Controlador $$C(s) = \frac{9.196s + 32.15}{s}$$

	Controlador PI sintonizado con Shen-Yu Ganancias - Kp = 6.131 - Ki = 94.32 Controlador $$C(s) = \frac{6.131s + 94.32}{s}$$
	Controlador PI sintonizado con Lambda Ganancias - Kp = 2 - Ki = 14.81 Controlador $$C(s) = \frac{2s + 14.81}{s}$$
	Controlador PI sintonizado con Cohen-Coon Ganancias - Kp = 4.639 - Ki = 63.15 Controlador $$C(s) = \frac{4.639s + 63.15}{s}$$

Fuente: Autores
Elaboración: Autores

En la tabla 3.3, se encuentran el valor de los parámetros a evaluar de cada método de sintonización aplicado al dedo pulgar.

Tabla 3.3 Comparación de los métodos de sintonización para un controlador PI, aplicado al sistema del dedo pulgar

Método	Overshoot (%)	Tiempo de establecimiento (segundo)
Ziegler-Nichols	28.9	0.307
Tyreus-Luyben	53.9	0.503
Shinskey	23.8	0.537
Lambda	1.57	0.382
Cohen-Coon	32	0.463
Shen-Yu	50.3	0.469

Fuente: Autores
Elaboración: Autores

De la tabla anterior, la mejor respuesta al escalón en base al criterio planteado de menor porcentaje de overshoot y de tiempo de establecimiento es el del controlador Lambda.

3.1.2.2. Dedo índice.

El sistema del dedo índice presenta oscilaciones mantenidas ante una ganancia crítica en lazo cerrado (ver figura 3.3). Los valores de las variables para la sintonización en lazo abierto y cerrado se muestran a continuación:

Lazo cerrado:

$$K_{cr} = 11.89375 \quad (3.6)$$

$$T_{cr} = 0.24 \quad (3.7)$$

Lazo abierto:

$$G_p = 0.41534 \quad (3.8)$$

$$\tau = 0.18 \quad (3.9)$$

$$\theta = 0.06754 \quad (3.10)$$

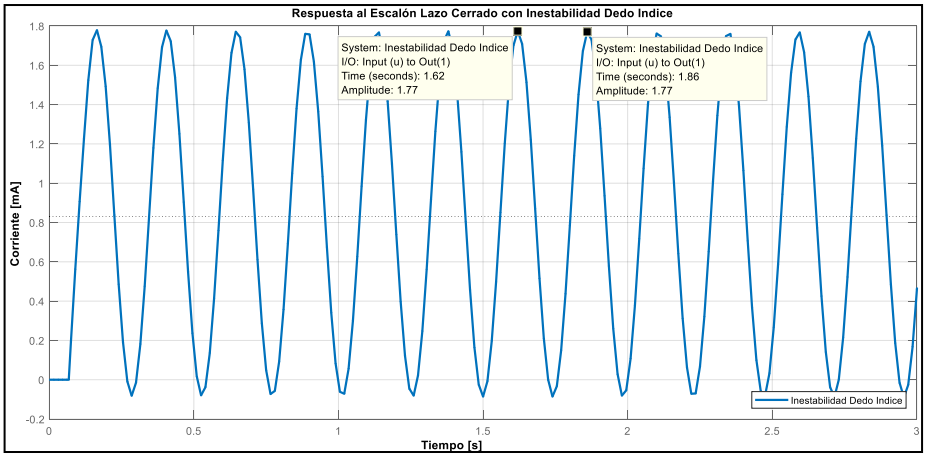
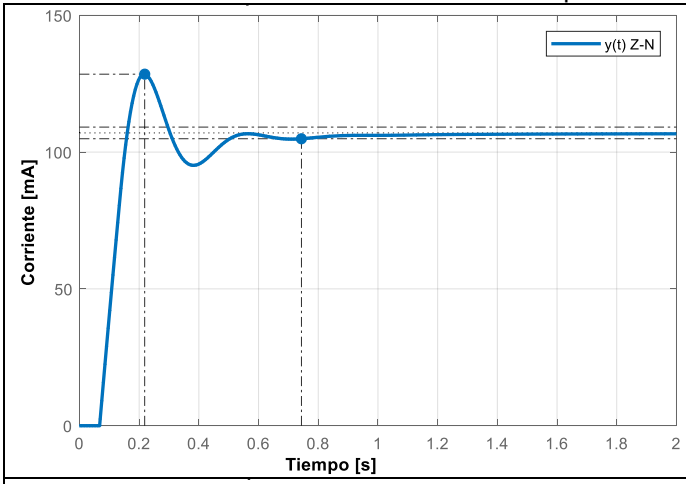
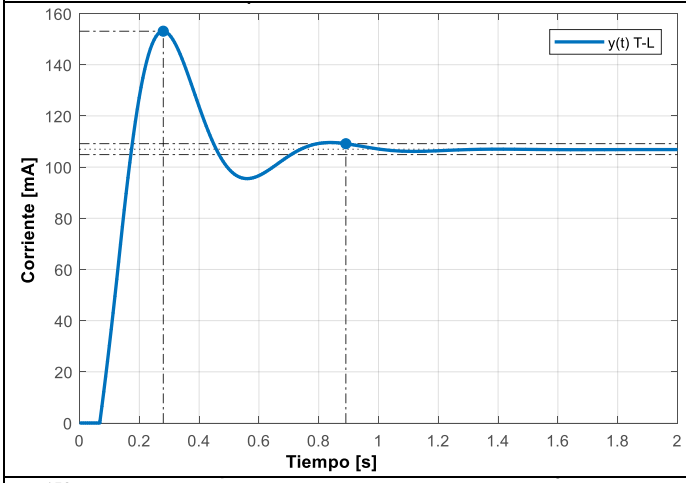
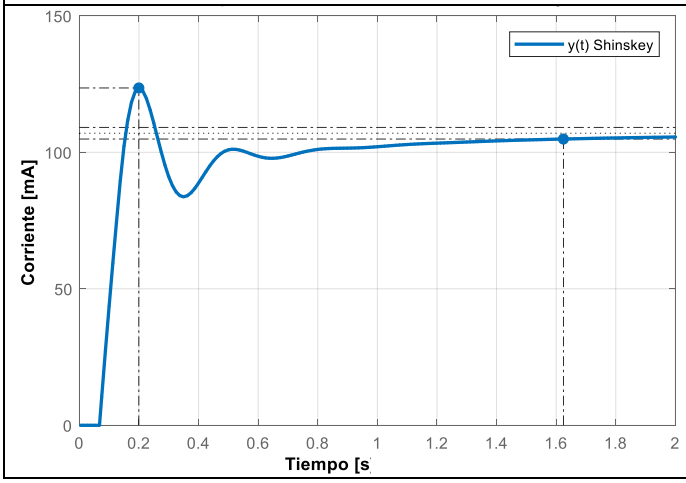
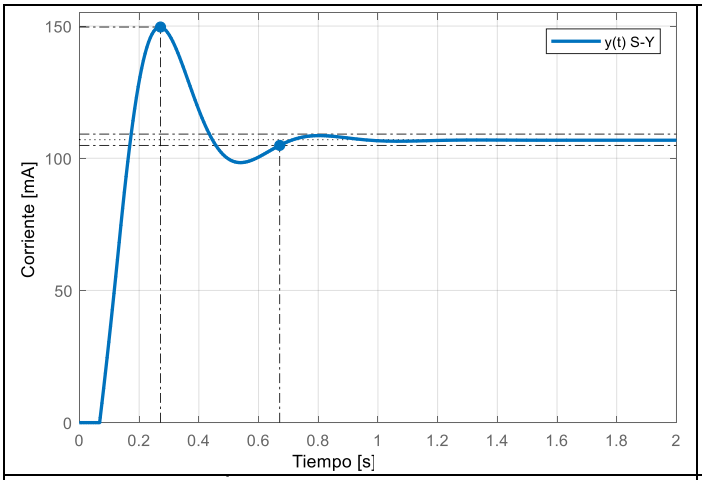
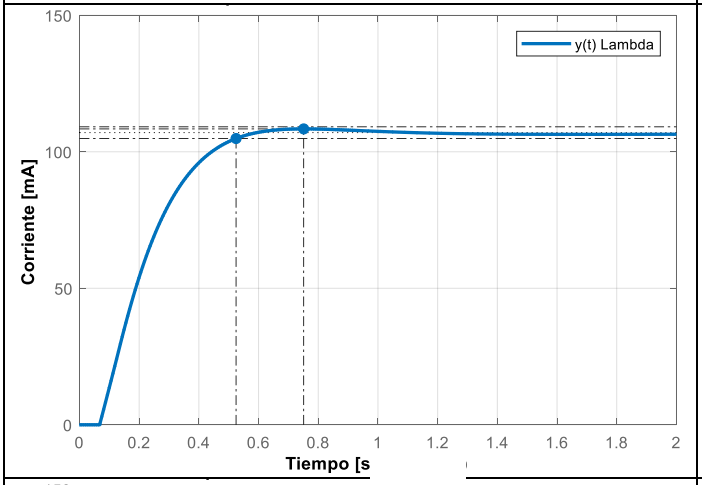
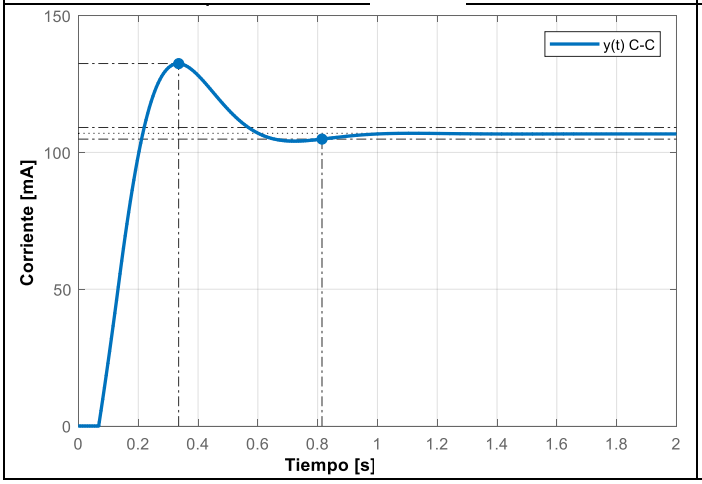


Figura 3.3 Inestabilidad en lazo cerrado del dedo índice
Fuente: Autores
Elaboración: Autores

En la tabla 3.4, se resume las ganancias obtenidas y la respuesta al escalón de 107 [mA] de cada método de sintonización para el dedo índice.

Tabla 3.4 Sintonización de los controladores PI para el sistema del dedo índice

	Controlador PI sintonizado con Ziegler-Nichols Ganancias - Kp = 5.352 - Ki = 18.58 Controlador $$C(s) = \frac{5.352s + 18.58}{s}$$
	Controlador PI sintonizado con Tyreus-Luyben Ganancias - Kp = 3.717 - Ki = 34.41 Controlador $$C(s) = \frac{3.717s + 34.41}{s}$$
	Controlador PI sintonizado con Shinskey Ganancias - Kp = 5.947 - Ki = 11.26 Controlador $$C(s) = \frac{5.947s + 11.26}{s}$$

	Controlador PI sintonizado con Shen-Yu Ganancias - Kp = 3.965 - Ki = 33.04 Controlador $$C(s) = \frac{3.965s + 33.04}{s}$$
	Controlador PI sintonizado con Lambda Ganancias - Kp = 1.751 - Ki = 9.726 Controlador $$C(s) = \frac{1.751s + 9.726}{s}$$
	Controlador PI sintonizado con Cohen-Coon Ganancias - Kp = 2.987 - Ki = 23.53 Controlador $$C(s) = \frac{2.987s + 23.53}{s}$$

Fuente: Autores
Elaboración: Autores

En la tabla 3.5, se encuentran los valores de los parámetros a evaluar de cada método de sintonización aplicado al dedo índice. En esta, se observa que el mejor rendimiento se obtiene con el método Lambda.

Tabla 3.5 Comparación de los métodos de sintonización para un controlador PI, aplicado al sistema del dedo índice

Método	Overshoot (%)	Tiempo de establecimiento (segundo)
Ziegler-Nichols	20.1	0.743
Tyreus-Luyben	43.1	0.891
Shinskey	15.5	1.62
Lambda	1.29	0.525
Cohen-Coon	23.8	0.815
Shen-Yu	39.8	0.671

Fuente: Autores
Elaboración: Autores

3.1.2.3. Dedo medio.

El sistema del dedo medio, presenta oscilaciones mantenidas ante una ganancia critica en lazo cerrado (ver figura 3.4). Los valores de las variables para la sintonización en lazo abierto y cerrado se muestran a continuación:

Lazo cerrado:

$$K_{cr} = 8.2655 \quad (3.11)$$

$$T_{cr} = 0.29 \quad (3.12)$$

Lazo abierto:

$$G_p = 0.50516 \quad (3.13)$$

$$\tau = 0.171 \quad (3.14)$$

$$\theta = 0.0799 \quad (3.15)$$

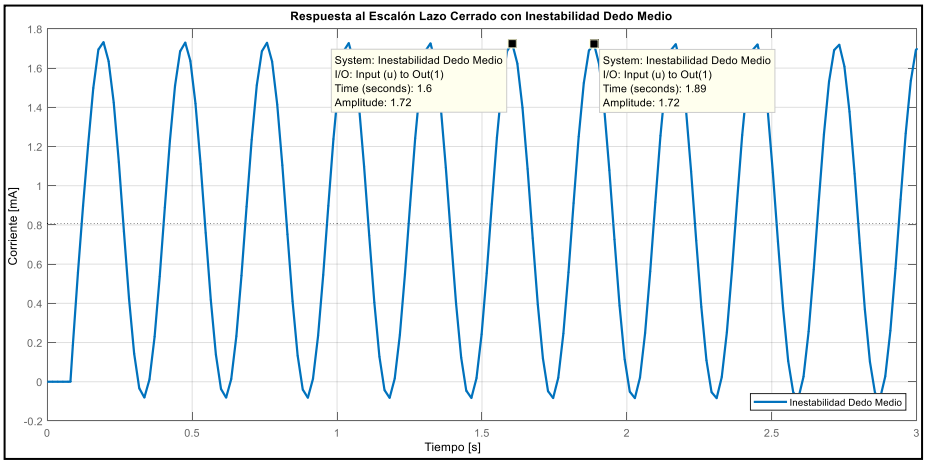
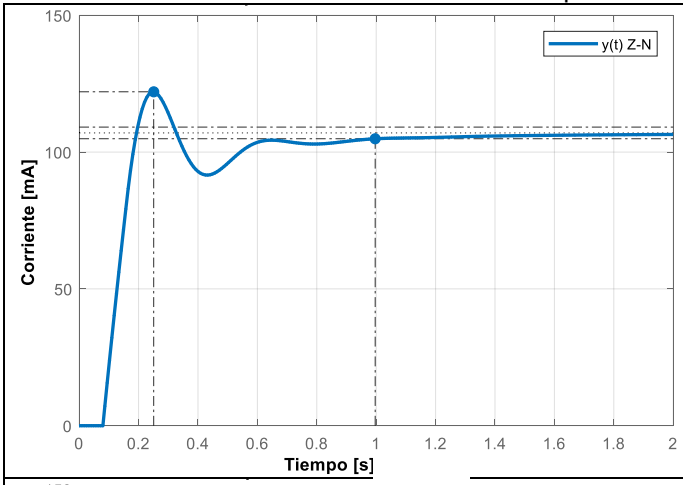
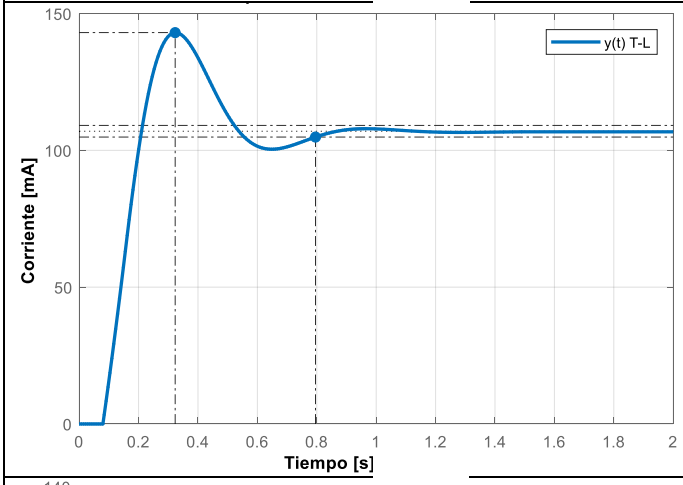
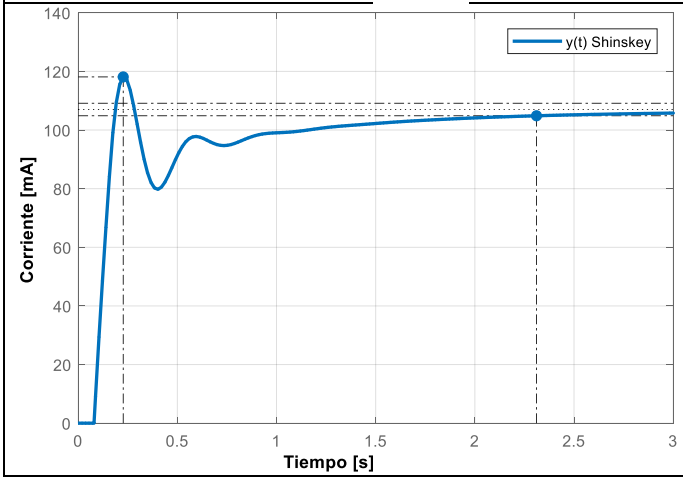
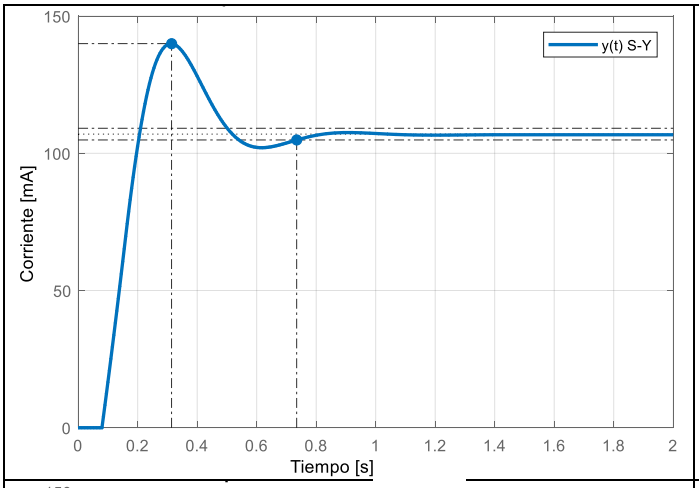
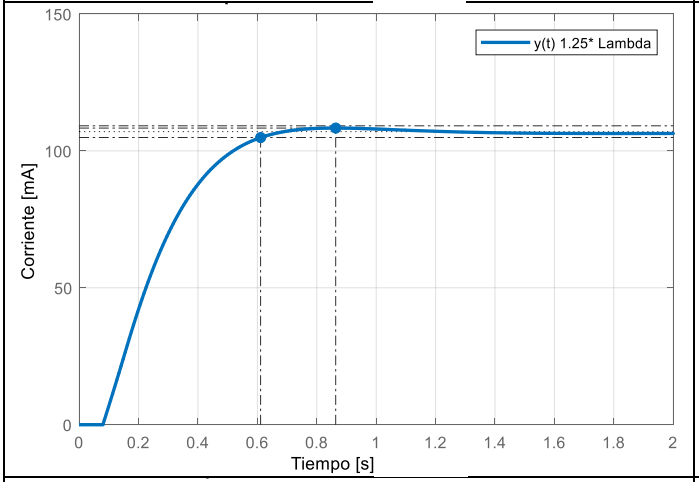
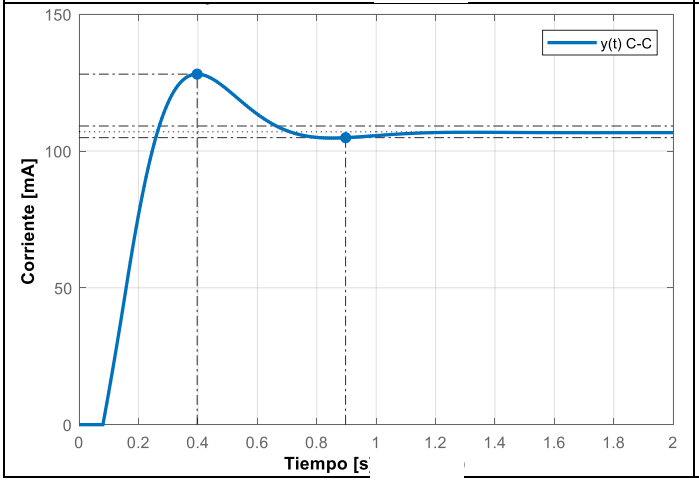


Figura 3.4 Inestabilidad en lazo cerrado del dedo medio
Fuente: Autores
Elaboración: Autores

En la tabla 3.6, se resume las ganancias obtenidas y la respuesta al escalón de 107 [mA] de cada método de sintonización para el dedo medio.

Tabla 3.6 Sintonización de los controladores PI para el sistema del dedo medio

	Controlador PI sintonizado con Ziegler-Nichols Ganancias - Kp = 3.719 - Ki = 10.69 Controlador $$C(s) = \frac{3.719s + 10.69}{s}$$
	Controlador PI sintonizado con Tyreus-Luyben Ganancias - Kp = 2.583 - Ki = 19.79 Controlador $$C(s) = \frac{2.583s + 19.79}{s}$$
	Controlador PI sintonizado con Shinskey Ganancias - Kp = 4.133 - Ki = 6.478 Controlador $$C(s) = \frac{4.133s + 6.478}{s}$$

	Controlador PI sintonizado con Shen-Yu Ganancias - Kp = 2.755 - Ki = 19 Controlador $$C(s) = \frac{2.755s + 19}{s}$$
	Controlador PI sintonizado con 1.25*Lambda Ganancias - Kp = 1.153 - Ki = 6.741 Controlador $$C(s) = \frac{1.153s + 6.741}{s}$$
	Controlador PI sintonizado con Cohen-Coon Ganancias - Kp = 1.988 - Ki = 14.6 Controlador $$C(s) = \frac{1.988s + 14.6}{s}$$

Fuente: Autores
Elaboración: Autores

En la tabla 3.7, se encuentran los valores de los parámetros a evaluar de cada método de sintonización aplicado al dedo medio. En esta, se observa que el mejor rendimiento se obtiene con el método Lambda.

Tabla 3.7 Comparación de los métodos de sintonización para un controlador PI, aplicado al sistema del dedo medio

Método	Overshoot (%)	Tiempo de establecimiento (segundo)
Ziegler-Nichols	14	0.997
Tyreus-Luyben	33.7	0.796
Shinskey	10.4	2.31
Lambda	1.24	0.611
Cohen-Coon	19.7	0.897
Shen-Yu	30.8	0.735

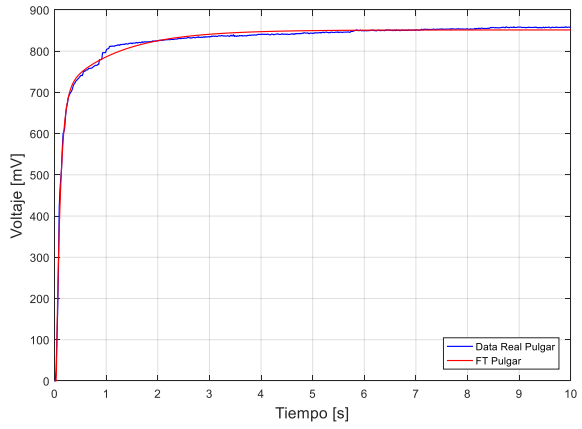
Fuente: Autores
Elaboración: Autores

3.2. Control de fuerza de agarre basado en medición de fuerza.

3.2.1. Identificación del sistema.

Ya que la dinámica en los dedos de la prótesis robótica de [2] es diferente entre los dedos pulgar, índice y medio, es necesario determinar una función de transferencia para cada sistema. A continuación, en la tabla 3.8 se muestran las funciones de transferencia de cada sistema y el parámetro Fit [54] que muestra Matlab® como la similitud en porcentaje entre los datos reales y la función estimada. La amplitud del escalón de entrada se considera 255, al requerir que el controlador entregue una salida con un máximo de PWM en 255.

Tabla 3.8 Modelos matemáticos y su respuesta a un escalón de entrada de los sistemas basados en medición de fuerza

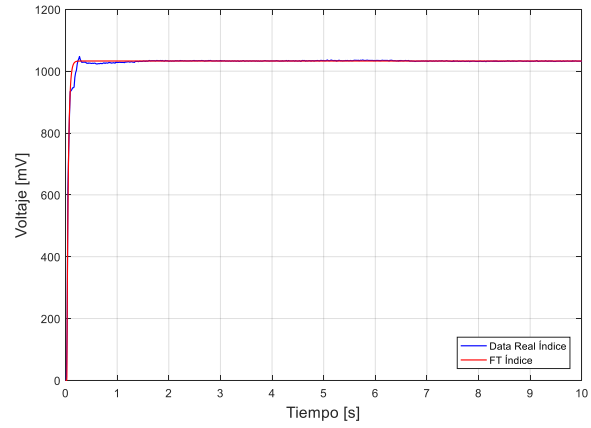
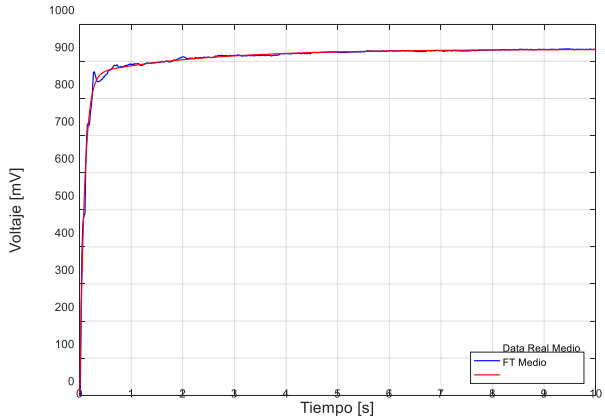


Modelo matemático en lazo abierto del dedo pulgar.

Función de transferencia.

$$G(s)_{pulgar} = \frac{3.022s + 3.339}{0.08625s^2 + 1.172s + 1}\exp^{(-0.0334s)}$$

$Fit \rightarrow 91.38\%$

	Modelo matemático en lazo abierto del dedo índice. Función de transferencia $G(s)_{índice} = \frac{4.051}{0.02765s + 1}\exp^{(-0.0251s)}$ $Fit \rightarrow 86.65\%$
	Modelo matemático en lazo abierto del dedo medio. Función de transferencia $G(s)_{medio} = \frac{7.344s + 3.656}{0.1917s^2 + 2.254s + 1}\exp^{(-0.00658s)}$ $Fit \rightarrow 89.54\%$

Fuente: Autores
Elaboración: Autores

3.2.2. Sintonización de los controladores PID.

Para los métodos de sintonización en lazo cerrado como: Ziegler-Nichols, Tyreus-Luyben, Shinskey, Shen-Yu, se requiere de una ganancia crítica K_{cr}, que lleve a cada sistema hacia una oscilación permanente como se muestra en la figura 3.5 y consecuentemente se conoce un periodo crítico T_{cr}, mientras, los métodos Lambda y Cohen-Conn requieren de parámetros determinados a partir de la curva de cada sistema de la tabla 3.8.

3.2.2.1. Dedo pulgar.

A continuación, se muestra la ganancia crítica y el periodo crítico en segundos que lleva al sistema del dedo pulgar a oscilaciones permanentes (ver figura 3.5).

$$K_{cr} = 1.5786 \quad (3.16)$$

$$T_{cr} = 0.115 \quad (3.17)$$

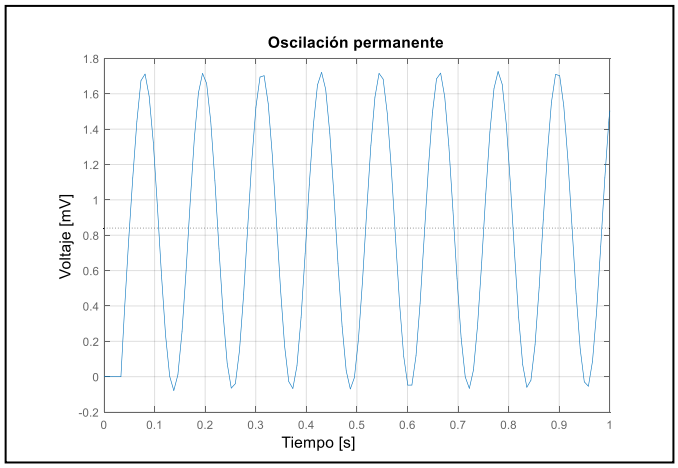


Figura 3.5 Inestabilidad en lazo cerrado del dedo pulgar
Fuente: Autores
Elaboración: Autores

Variables G_p, τ, θ de los métodos de sintonización en lazo abierto:

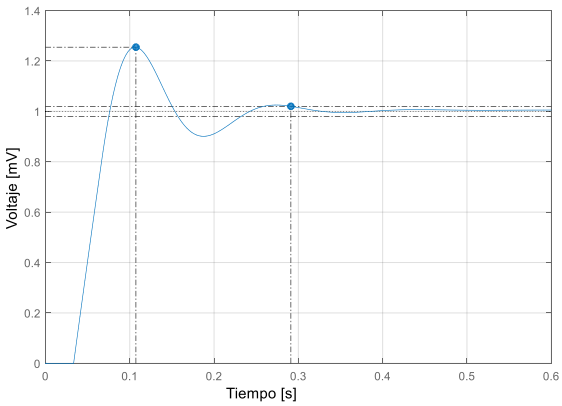
$$G_p = 3.3389 \quad (3.18)$$

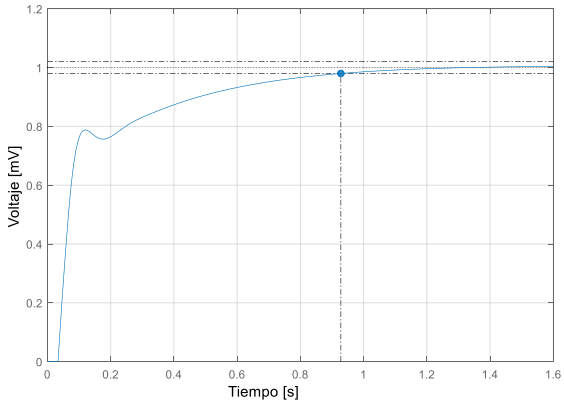
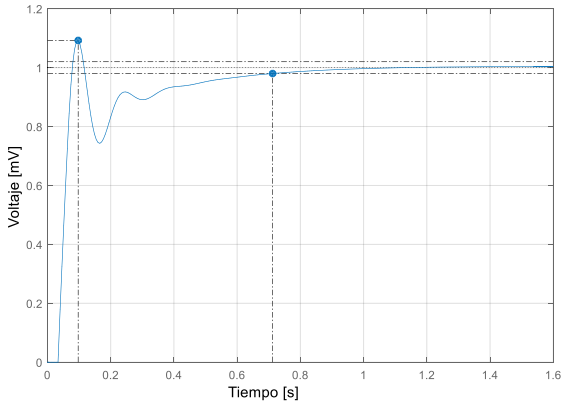
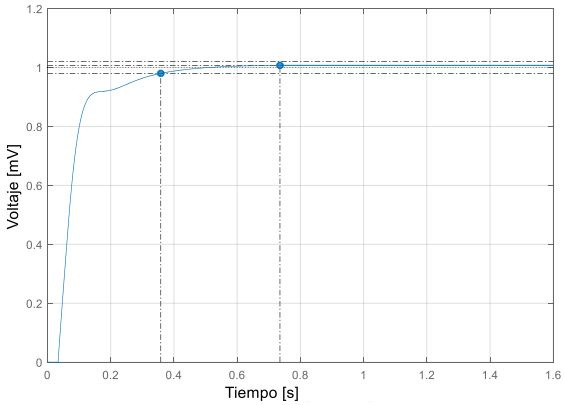
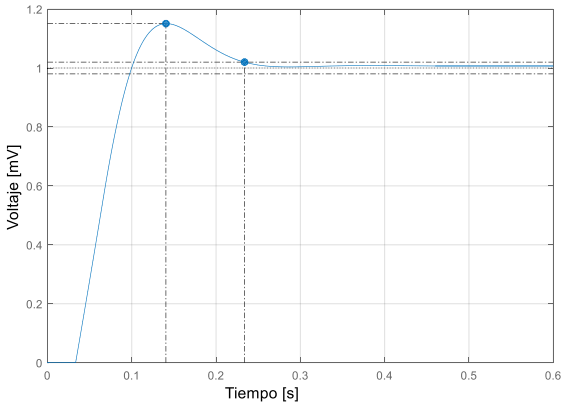
$$\tau = 0.114 \quad (3.19)$$

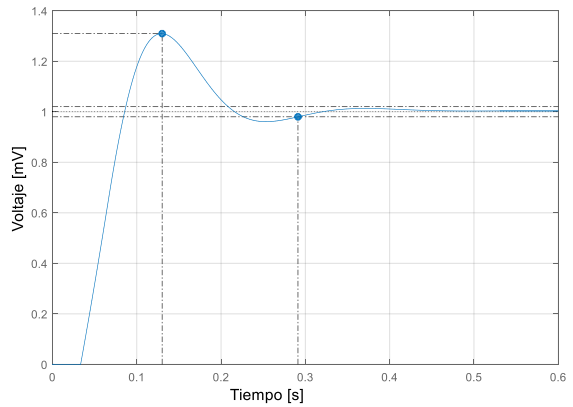
$$\theta = 0.033435 \quad (3.20)$$

En la tabla 3.9, se resume las ganancias obtenidas y la respuesta al escalón unitario en unidades de [mV] de cada método de sintonización para el dedo pulgar.

Tabla 3.9 Sintonización de los controladores PI para el sistema del dedo pulgar de controlador basado en medición de fuerza

	Controlador PI sintonizado con Ziegler-Nichols Ganancias - Kp = 0.71037 - Ki = 7.4126 Controlador $C(s) = \frac{0.71037 * 7.4126}{s}$

	Controlador PI sintonizado con Tyreus-Luyben Ganancias - Kp = 0.49331 - Ki = 1.9499 Controlador $$C(s) = \frac{0.49331s * 1.9499}{s}$$
	Controlador PI sintonizado con Shinskey Ganancias - Kp = 0.7893 - Ki = 3.1198 Controlador $$C = \frac{0.7893s * 3.1198}{s}$$
	Controlador PI sintonizado con Lambda a 0.4 Ganancias - Kp = 0.432 - Ki = 3.7895 Controlador $$C = \frac{0.432s * 3.7895}{s}$$
	Controlador PI sintonizado con Cohen-Coon Ganancias - Kp = 0.47193 - Ki = 6.8146 Controlador $$C = \frac{0.47193s * 6.8146}{s}$$

	Controlador PI sintonizado con Shen-Yu Ganancias - Kp = 0.5262 - Ki = 9.1513 Controlador $C = \frac{0.5262s * 9.1513}{s}$

Fuente: Autores
Elaboración: Autores

A partir de las simulaciones de sintonización para el sistema del dedo pulgar mostradas en la tabla 3.9, se realiza una tabla comparativa (ver tabla 3.10) de los parámetros de evaluación entre los seis métodos de sintonización de un controlador PI.

Tabla 3.10 Comparación de los métodos de sintonización de un controlador PI, aplicado al sistema del dedo pulgar

Método	Overshoot (%)	Tiempo de establecimiento (segundo)
Ziegler-Nichols	25.5	0.291
Tyreus-Luyben	0	0.928
Shinskey	9.24	0.712
Lambda	0.725	0.358
Cohen-Coon	15.1	0.234
Shen-Yu	31	0.291

Fuente: Autores
Elaboración: Autores

El método que cumple con las condiciones mencionadas en 3.1 es Lambda. Entre todos los métodos de sintonización éste se destaca por ser robusto ante variaciones del set-point y, tener el menor tiempo de establecimiento con 0.358 segundos.

3.2.2.2. Dedo índice.

A continuación, se muestra la ganancia crítica y el periodo crítico en segundos que lleva al sistema del dedo índice a oscilaciones permanentes (ver figura 3.6), datos utilizados para la simulación de sintonizadores de lazo cerrado. En la tabla 3.11 se encuentran todos los métodos de sintonización mencionados, aplicados al sistema del dedo índice.

$$K_{cr} = 0.59679 \quad (3.21)$$

$$T_{cr} = 0.078 \quad (3.22)$$

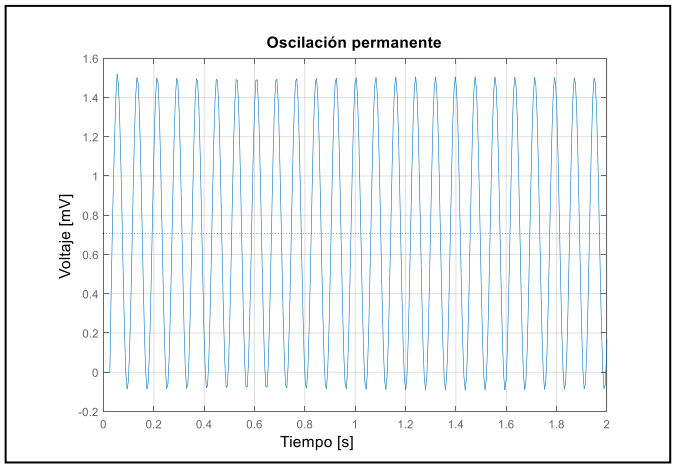


Figura 3.6 Sistema del dedo índice con ganancia crítica
Fuente: Autores
Elaboración: Autores

Variables G_p, τ, θ de los métodos de sintonización en lazo abierto:

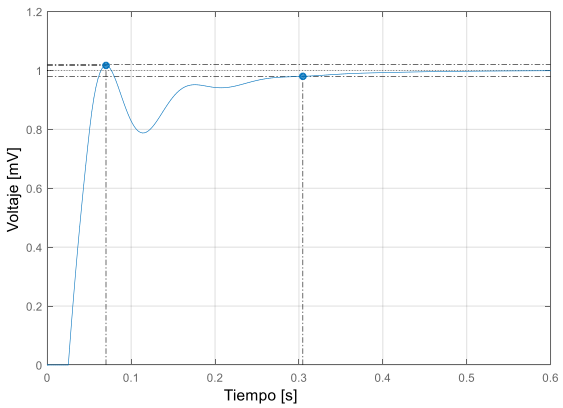
$$G_p = 4.0506 \quad (3.23)$$

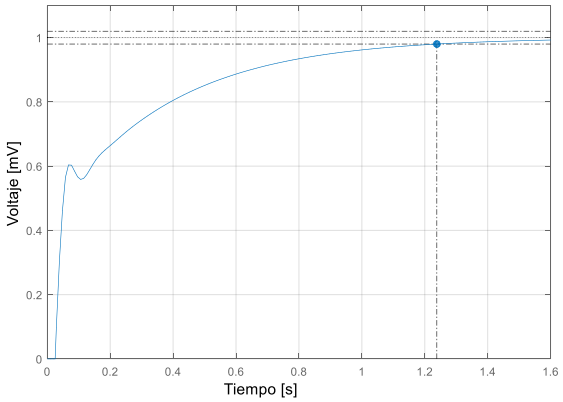
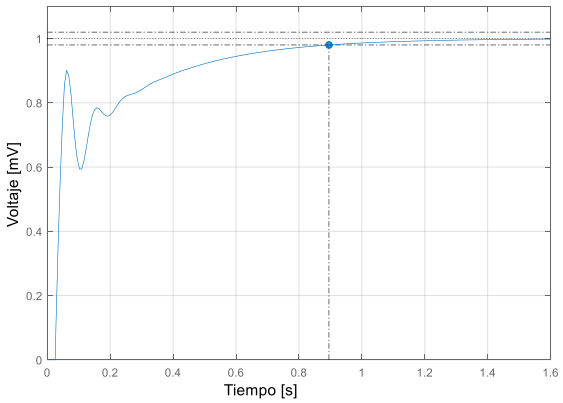
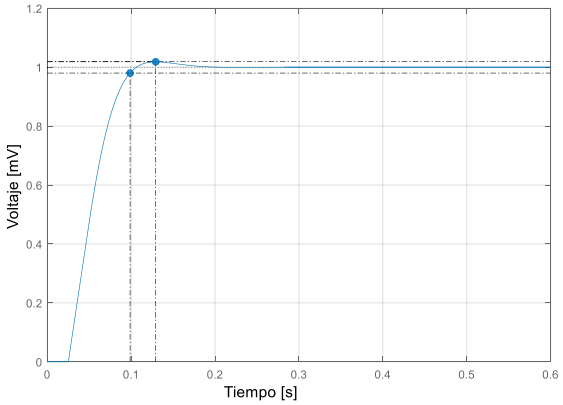
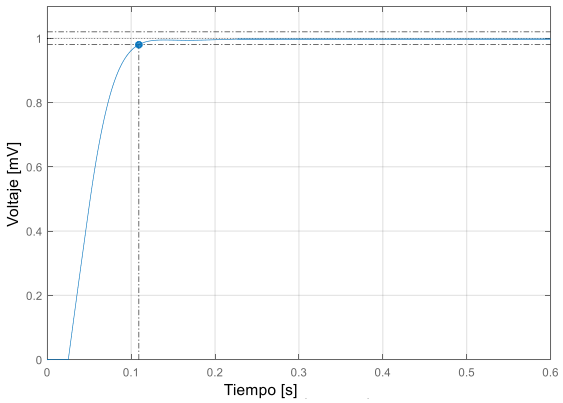
$$\tau = 0.0283 \quad (3.24)$$

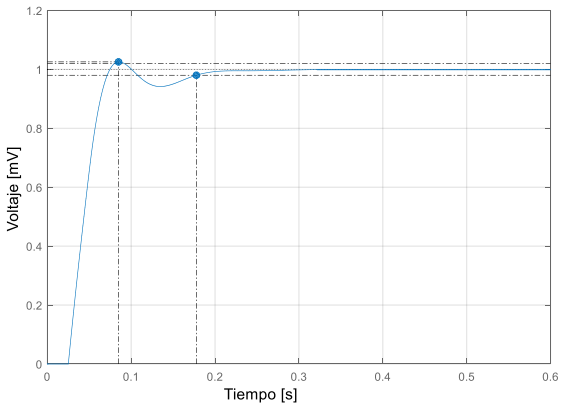
$$\theta = 0.02514 \quad (3.25)$$

En la tabla 3.11, se resume las ganancias obtenidas y la respuesta al escalón unitario en unidades de [mV] de cada método de sintonización para el dedo índice.

Tabla 3.11 Sintonización de los controladores PI para el sistema del dedo índice

	Controlador PI sintonizado con Ziegler-Nichols Ganancias - Kp = 0.26856 - Ki = 4.1316 Controlador $C(s) = \frac{0.26856s * 4.1316}{s}$

	Controlador PI sintonizado con Tyreus-Luyben Ganancias - Kp = 0.1865 - Ki = 1.0868 Controlador $$C(s) = \frac{0.1865s * 1.0868}{s}$$
	Controlador PI sintonizado con Shinskey Ganancias - Kp = 0.31711 - Ki = 2.8829 Controlador $$C = \frac{0.31711s * 2.8829}{s}$$
	Controlador PI sintonizado con Lambda a 0.5 Ganancias - Kp = 0.13074 - Ki = 4.6197 Controlador $$C = \frac{0.13074s * 4.6197}{s}$$
	Controlador PI sintonizado con Cohen-Coon Ganancias - Kp = 0.13528 - Ki = 4.4399 Controlador $$C = \frac{0.13528s * 4.4399}{s}$$



Controlador PI sintonizado con Shen-Yu

Ganancias
- Kp = 0.19893
- Ki = 5.1008

Controlador

$$C = \frac{0.19893s * 5.1008}{s}$$

Fuente: Autores
Elaboración: Autores

Tabla 3.12 Comparación de los métodos de sintonización para un controlador PI, aplicado al sistema del dedo índice

Método	Overshoot (%)	Tiempo de establecimiento (segundo)
Ziegler-Nichols	1.71	0.305
Tyreus-Luyben	0	1.24
Shinskey	0	0.895
Lambda	1.82	0.098
Cohen-Coon	0	0.109
Shen-Yu	2.54	0.178

Fuente: Autores
Elaboración: Autores

En la tabla 3.12 se muestran los métodos Tyreus-Luyben y Shinskey a pesar de no tener overshoot su tiempo de establecimiento frente al resto de métodos es mayor, por ende, los mejores métodos por debajo del 5% de overshoot son Ziegler-Nichols, Lambda, Cohen-Coon y Shen-Yu. De estos cuatro métodos Lambda y Cohen-Coon presentan los menores tiempos de estabilización por lo tanto se considera que el mejor método para aplicar al sistema del dedo índice es Cohen-Coon por presentar mayor robustez frente a cambios de set-point que el método Lambda.

3.2.2.3. Dedo medio.

A continuación, se muestra la ganancia crítica y el periodo crítico en segundos que lleva al sistema del dedo medio a oscilaciones permanentes (ver figura 3.7), datos utilizados para la simulación de sintonizadores de lazo cerrado.

$$K_{cr} = 6.41838 \quad (3.26)$$

$$T_{cr} = 0.024 \quad (3.27)$$

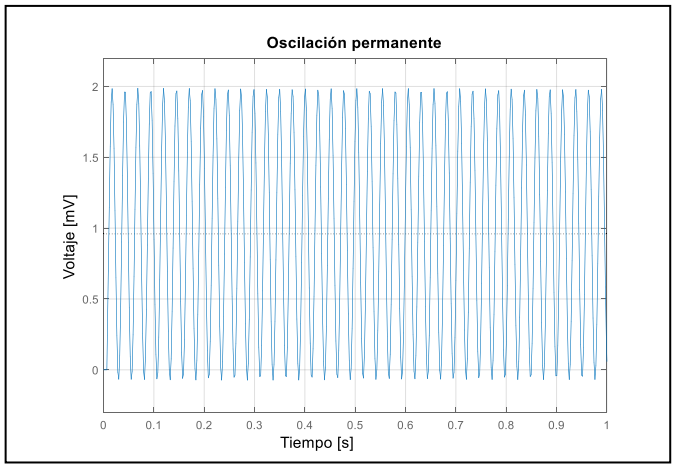


Figura 3.7 Sistema del dedo medio con ganancia crítica
Fuente: Autores
Elaboración: Autores

Variables G_p, τ, θ de los métodos de sintonización en lazo abierto:

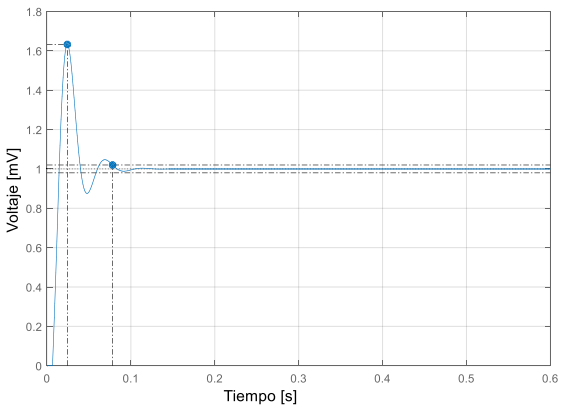
$$G_p = 3.6565 \quad (3.28)$$

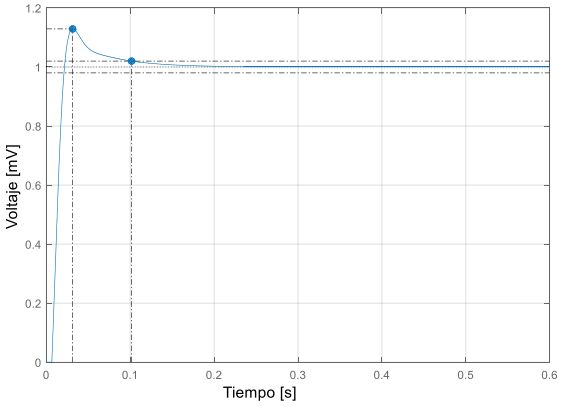
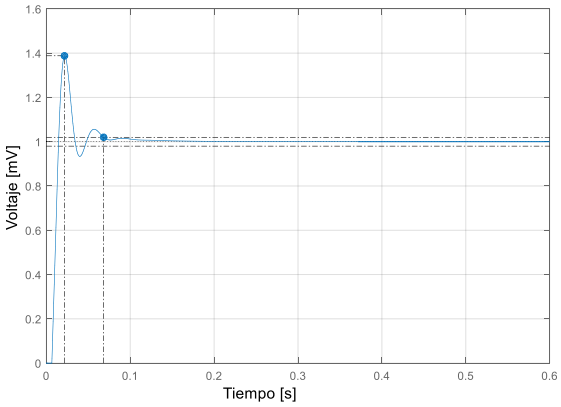
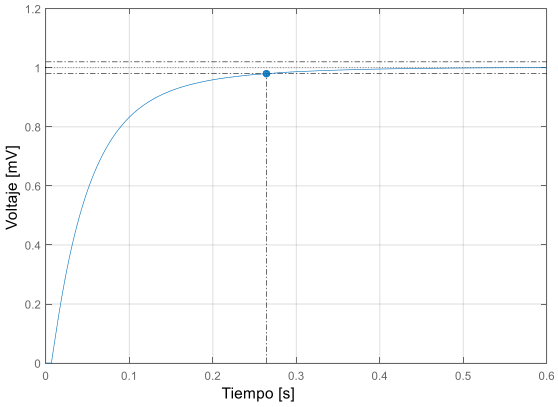
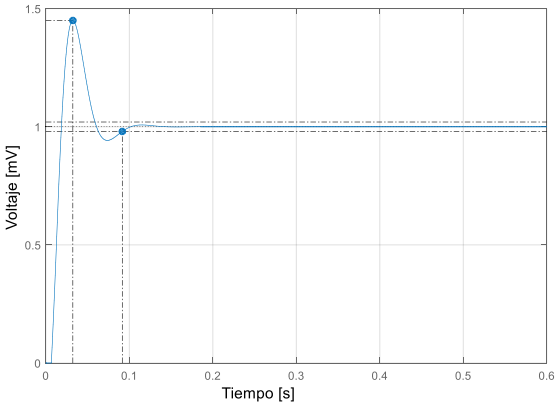
$$\tau = 0.102 \quad (3.29)$$

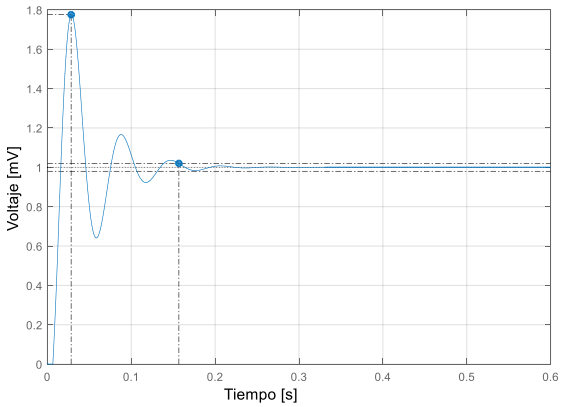
$$\theta = 0.006585 \quad (3.30)$$

En la tabla 3.13, se resume las ganancias obtenidas y la respuesta al escalón unitario en unidades de [mV] de cada método de sintonización para el dedo medio.

Tabla 3.13 Sintonización de los controladores PI para el sistema del dedo medio

	Controlador PI sintonizado con Ziegler-Nichols Ganancias - Kp = 2.8883 - Ki = 144.4135 Controlador $C(s) = \frac{2.8883s * 144.4135}{s}$

	Controlador PI sintonizado con Tyreus-Luyben Ganancias - Kp = 2.0057 - Ki = 37.9876 Controlador $$C(s) = \frac{2.0057 * 37.9876}{s}$$
	Controlador PI sintonizado con Shinskey Ganancias - Kp = 3.2092 - Ki = 60.7801 Controlador $$C = \frac{3.2092s * 60.7801}{s}$$
	Controlador PI sintonizado con Lambda a 0.5 Ganancias - Kp = 0.48442 - Ki = 4.7493 Controlador $$C = \frac{0.48442s * 4.7493}{s}$$
	Controlador PI sintonizado con Cohen-Coon Ganancias - Kp = 1.9176 - Ki = 99.3938 Controlador $$C = \frac{1.9176s * 99.3938}{s}$$



Controlador PI sintonizado con Shen-Yu

Ganancias
- Kp = 2.1395
- Ki = 178.2883

Controlador

$$C = \frac{2.1395s * 178.2883}{s}$$

Fuente: Autores
Elaboración: Autores

El método de sintonización Lambda es el único que cumple con el criterio de tener un overshoot menor a 5%, a pesar de que el resto de métodos tienen un tiempo de establecimiento menor al de Lambda, todos tienen un overshoot sobre el 12.9% como se muestra en la tabla 3.14.

Tabla 3.14 Comparación de los métodos de sintonización para un controlador PI, aplicado al sistema del dedo medio

Método	Overshoot (%)	Tiempo de establecimiento (segundo)
Ziegler-Nichols	63.3	0.0782
Tyreus-Luyben	12.9	0.102
Shinskey	38.8	0.0684
Lambda	0	0.264
Cohen-Coon	45	0.0915
Shen-Yu	77.6	0.157

Fuente: Autores
Elaboración: Autores

En este caso al ser Lambda el método con overshoot de 0% y 0.264 segundos en el tiempo de establecimiento se considera para ser aplicado al sistema del dedo medio.

CAPITULO IV

IMPLEMENTACIÓN DEL CONTROLADOR Y PRUEBAS EN LA PRÓTESIS ROBÓTICA

La evaluación del agarre de la prótesis robótica se ejecutará con dos tipos de objetos: objetos cilíndricos con cuatro diferentes diámetros (45, 55, 65 y 75 [mm]) y una altura de 110 [mm], y objetos esféricos con los mismos diámetros antes mencionados. Ambos tipos de objetos con una cavidad interna para agregar peso. Los objetos se recubrieron con una silicona para agregar adherencia en el agarre con la prótesis.

Se ha denominado con un código para diferenciar a los objetos, especificando el tipo de objeto con su letra inicial y el diámetro con números del 1 a 4 de menor a mayor, en la tabla 4.1 se distingue la abreviación y pesos de cada uno de los objetos de prueba.

Tabla 4.1 Renombre de los objetos de prueba y pesos correspondientes de 0%, 50% y 100% de capacidad

Objeto	Diámetro ϕ	Abreviación	Peso a 0%	Peso a 50%	Peso a 100%
Cilindro	45 [mm]	C1	84 g	204 g	289 g
Cilindro	55 [mm]	C2	93 g	255 g	394 g
Cilindro	65 [mm]	C3	102 g	365 g	548 g
Cilindro	75 [mm]	C4	116 g	467 g	763 g
Esfera	45 [mm]	E1	16 g	50 g	76 g
Esfera	55 [mm]	E2	22 g	83 g	144 g
Esfera	65 [mm]	E3	30 g	122 g	192 g
Esfera	75 [mm]	E4	43 g	200 g	348 g

Fuente: Autores
Elaboración: Autores

Se realizarán tres pruebas por objeto, variando en cada uno el peso desde vació o 0%, 50% y a 100% de la capacidad en cada uno. Una rutina programada para pruebas cambia cada 3 segundos el set-point, variándolo cada 25% hasta llegar a la máxima fuerza que cada dedo puede aplicar. Para la retroalimentación con medición de corriente se considera un rango entre 0-100[mA] para el set-point, equivalente a 0-30 [N] de torque ejercido por los actuadores PQ12 [55], por el contrario, en la retroalimentación con medición de fuerza se debe mantener el set-point bajo el límite superior del sistema del dedo pulgar (ver tabla 3.8).

4.1. Controlador basado en medición de corriente.

El controlador que se sometió a pruebas fue el obtenido mediante el método de sintonización Lambda, dado que tiene los mejores parámetros con respecto al tiempo de establecimiento y sobre-elongación (ver tablas 3.5, 3.7, 3.10).

En la implementación de este controlador se agrega el bloque que evita el integral-windup, efecto que produce la parte integrativa [28].El método para eliminar este efecto es el de back-calculation, el cuál es ampliamente usado en sistemas [28].En la figura 4.1 se muestra el diagrama de bloques del controlador aplicado al dedo pulgar con back-calculation desarrollado en Simulink®.

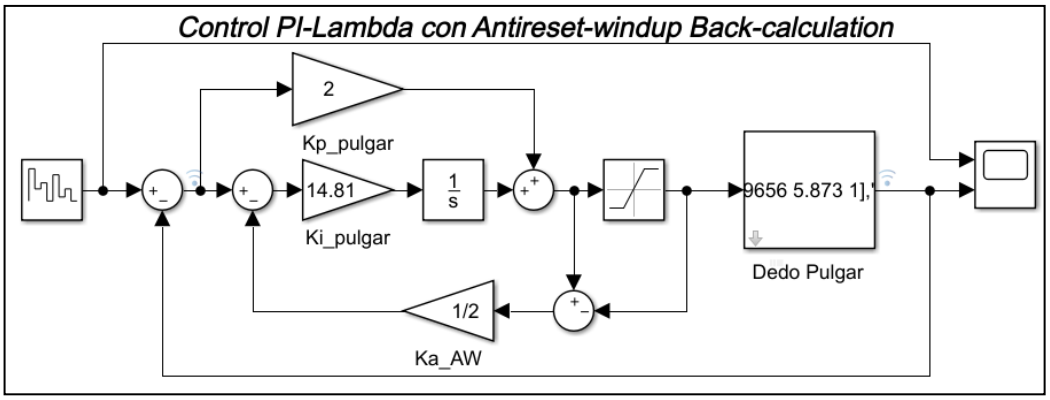


Figura 4.1 Diagrama de bloques del sistema del dedo pulgar con Back-calculation
Fuente: Autores
Elaboración: Autores

4.1.1. Pruebas de agarre con objetos.

Las pruebas se ejecutaron ante escalones 25 [mA] de amplitud con respecto al anterior, hasta el límite máximo de 100 [mA] de corriente consumida por cada actuador. Además, se busca el set-point en el cuál los objetos dejen de desplazarse por efecto de la gravedad, con lo que encontraremos los límites de control de fuerza de agarre para objetos cilíndricos y esféricos de la prótesis robótica desarrollada en [1].

4.1.1.1. Agarre de objetos cilíndricos.

El agarre del objeto cilíndrico de mayor diámetro (C4) se muestra en la figura 4.2. En la imagen se aprecia que el cilindro está recubierto por una silicona mejorando el agarre al aumentar la fricción entre el objeto y la prótesis robótica.

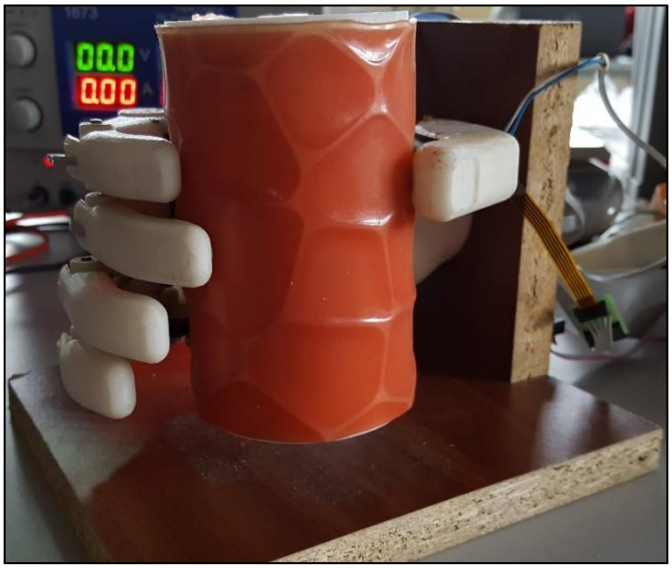


Figura 4.2 Objeto C4 en agarre de 3 dedos
Fuente: Autores
Elaboración: Autores

En la figura 4.3 se muestra la repuesta del lazo de control de fuerza de agarre basado en corriente implementado ante el objeto C4 a 0% de su capacidad. En la figura 4.4 se muestra la respuesta ante el mismo objeto con el 100% de peso. Los controladores manifiestan un comportamiento robusto a los escalones de entrada.

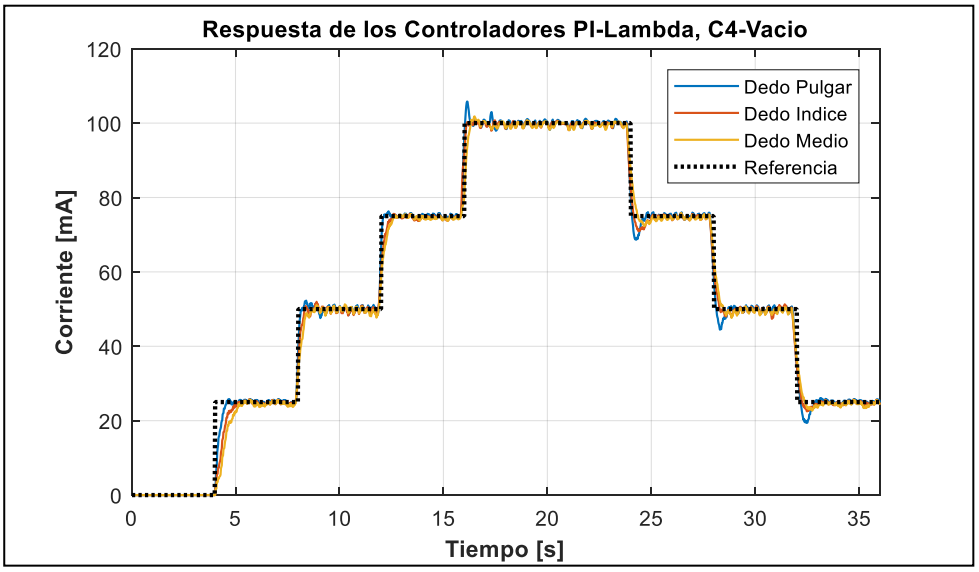


Figura 4.3 Respuesta de los controladores implementados, en el agarre del objeto vacío C4
Fuente: Autores
Elaboración: Autores

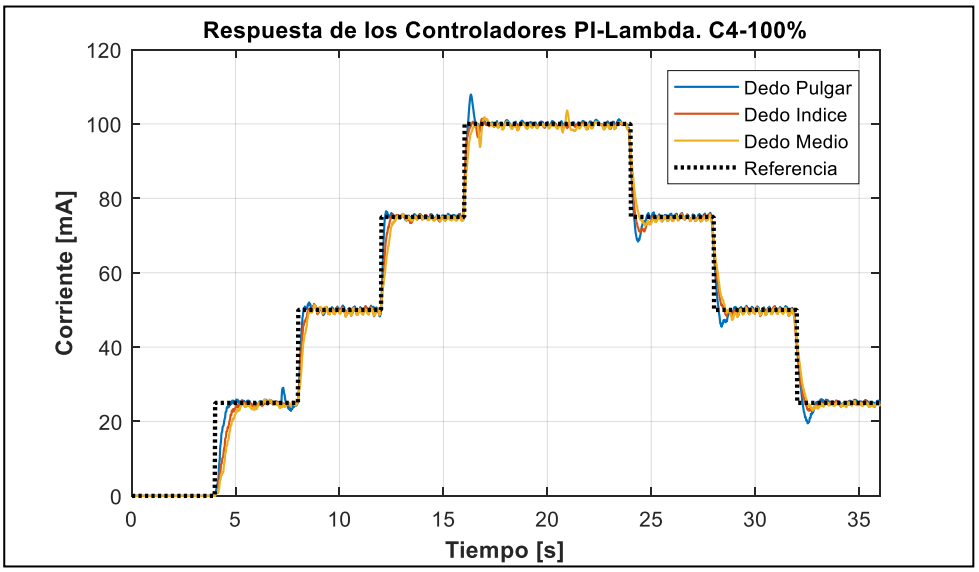


Figura 4.4 Respuesta de los controladores implementados, en el agarre del objeto C4-100% lleno
Fuente: Autores
Elaboración: Autores

En la figura anterior, se observa que el dedo pulgar se establece en 0.6 [s], el índice y medio en aproximadamente 0.85 [s] ante el set-point de 25 [mA]. Además, el dedo pulgar experimenta más perturbaciones que el resto, debido a que está sometido a la fuerza del dedo índice y medio.

El cilindro más pesado deja de desplazarse a partir del escalón de 50 [mA] de corriente, lo que equivale a 11.25 [N] de torque en cada actuador [55]. En la tabla 4.2, se resume las observaciones realizadas con respecto al desplazamiento de los objetos cilíndricos, la D significa desplazamiento y el no desplazamiento está representado por SD.

Tabla 4.2 Observación de deslizamiento de los objetos cilíndricos aplicado el control de fuerza basado en corriente a la prótesis robótica

Objeto	S-P 1	S-P 2	S-P 3	S-P 4	Peso
C1	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%
C2	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%
C3	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%
C4	SD	SD	SD	SD	0%
	D	SD	SD	SD	50%
	D	D	SD	SD	100%

Fuente: Autores
Elaboración: Autores

En el anexo E, se muestran las respuestas del controlador ante las pruebas con los objetos restantes.

4.1.1.2. Agarre de objetos esféricos.

El objeto esférico de mayor radio, E4, se acopla perfectamente a un agarre de 3 dedos como se muestra en la figura 4.5. Se observa, que se instaló membranas de silicona para mejorar el agarre del objeto esférico en la yema de cada dedo y en la palma de la prótesis robótica.

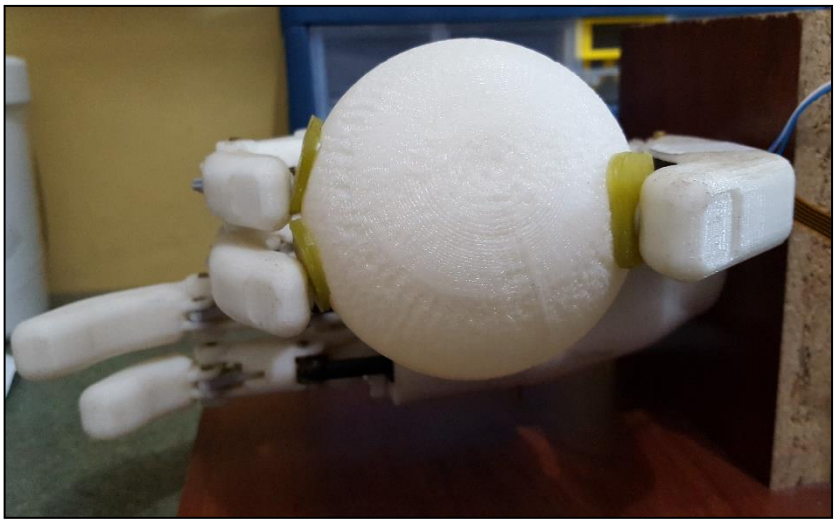


Figura 4.5 Objeto E4 en agarre de 3 dedos
Fuente: Autores
Elaboración: Autores

La respuesta ante diferentes escalones de entrada al agarrar este objeto se muestra en la figura 4.6 en el caso de la esfera vacía, y en el caso de la esfera a su máximo peso la respuesta se muestra en la figura 4.7.

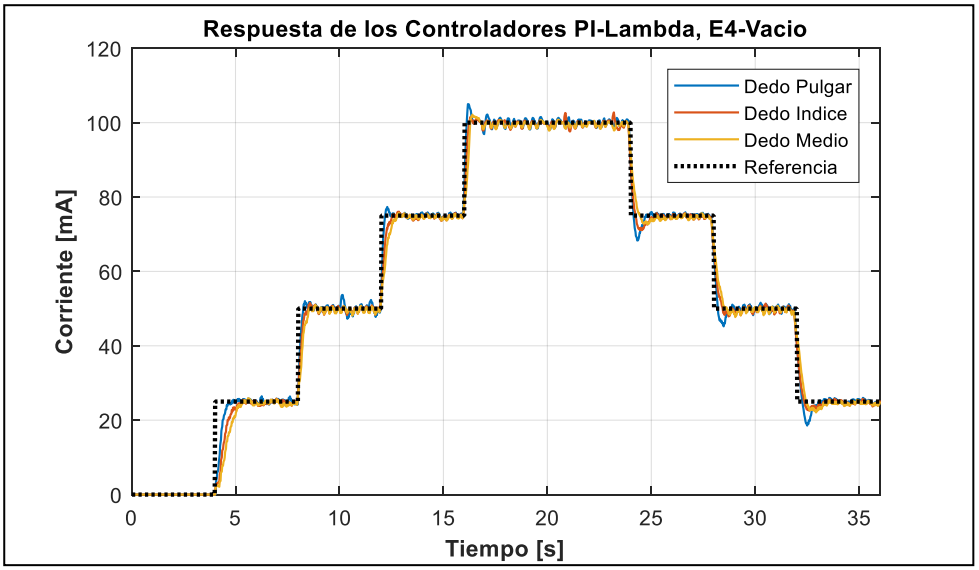


Figura 4.6 Respuesta de los controladores implementados, en el agarre del objeto E4-vacío
Fuente: Autores
Elaboración: Autores

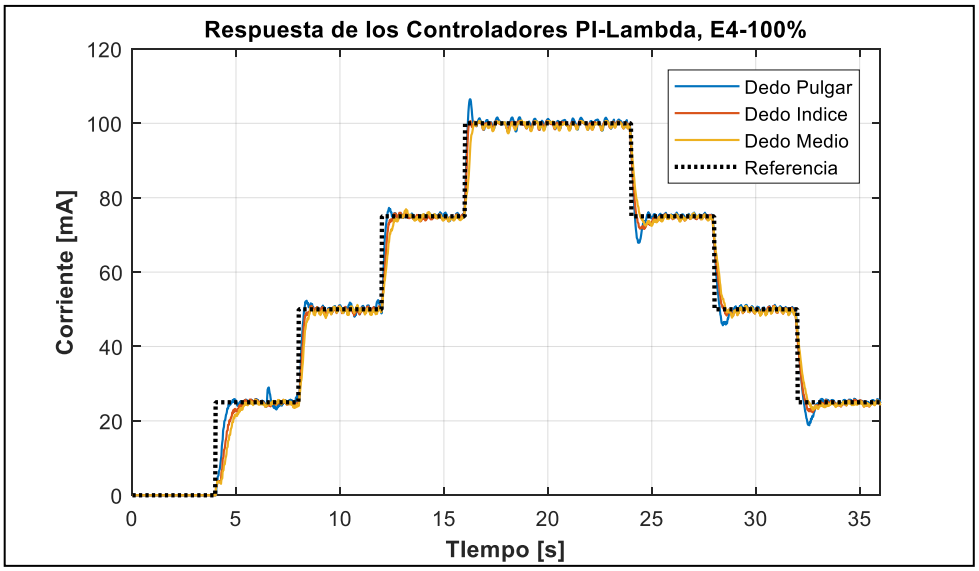


Figura 4.7 Respuesta de los controladores implementados, en el agarre del objeto E4-100% lleno
Fuente: Autores
Elaboración: Autores

En la implementación, se observa que en el agarre de esferas no existe desplazamiento del objeto en el peso máximo de la esfera E4. El efecto de las perturbaciones se evidencia en el dedo pulgar, de la misma manera que en el caso de los cilindros, ya que la fuerza ejercida por los dedos índice y medio provocan esto. Los tiempos de establecimiento se mantienen iguales que en el agarre de objetos cilíndricos.

Las observaciones realizadas, sobre el desplazamiento de los objetos ante diferentes set-points, se resumen en la tabla 4.3.

Tabla 4.3 Observación de deslizamiento de los objetos esféricos aplicado el control de fuerza basado en corriente a la prótesis robótica

Objeto	S-P 1	S-P 2	S-P 3	S-P 4	Peso
E1	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%
E2	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%
E3	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%
E4	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	SD	SD	SD	SD	100%

Fuente: Autores
Elaboración: Autores

La respuesta de los controladores ante los objetos esféricos restantes se muestran en el anexo E.

4.2. Controlador basado en medición de fuerza.

El controlador que se sometera a pruebas es el obtenido mediante el método de sintonización Lambda para los dedos pulgar y medio y Cohen-Coon para el dedo índice, dado que tienen los mejores parámetros con respecto al tiempo de establecimiento y sobre-elongación (ver tablas 3.10, 3.12 y 3.14).

El método back-calculation descrito en 4.1 al ser aplicado para el control mediante la medición de fuerza con los sensores FSR, presentó fallas al no disminuir la fuerza ejercida por los actuadores y llegar al set-point ingresado. Para solucionar este inconveniente se implementó el método incremental [34]. En la figura 4.8 se muestra el diagrama de bloques del controlador aplicado al dedo pulgar con el método incremental desarrollado en Simulink®.

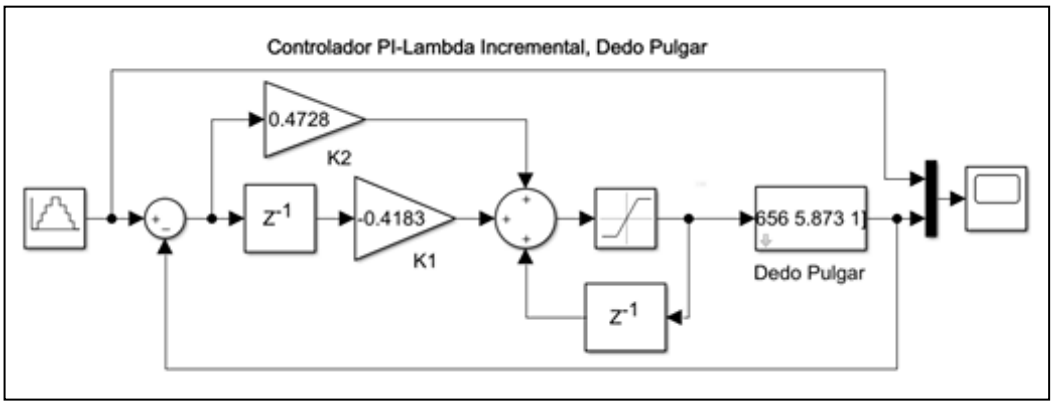


Figura 4.8 Diagrama de bloques del sistema del dedo pulgar con método incremental
Fuente: Autores
Elaboración: Autores

4.2.1. Pruebas de agarre con objetos.

Para la evaluación del controlador se realizó unas variaciones ante lo estipulado en tercer párrafo al inicio de éste capítulo, debido a que al agarrar los objetos desde el 50% de peso se generó perturbación entre los dedos índice y medio hacia el dedo pulgar, en otras palabras la fuerza ejercida por los dos dedos opuestos al dedo pulgar hacían que éste retrocediera considerablemente dejando caer al objeto por efecto de la gravedad.

Por lo tanto, se planteó empezar con un set-point de 200 [mV] para los dedos índice y medio, mientras que para el dedo pulgar se empieza desde 400 [mV], ambos con cuatro incrementos de 100 [mV].

4.2.1.1. Agarre de objetos cilíndricos.

En la figura 4.9 se observa la respuesta del controlador PI, basado en medición de fuerza, para el objeto C1 con 0% de peso adicional. A diferencia de lo simulado se observa que los sensores FSR se ven afectados por las perturbaciones generadas por cada actuador del dedo opuesto al sensor medido. En la figura 4.10 se distingue que el domo del dedo medio no ejerce presión contra el cilindro de manera transversal, de modo que no se va a sensar correctamente la fuerza aplicada por el actuador, este efecto se muestra en la figura 4.9 con la curva perteneciente al dedo medio donde se observa que no se registra el cambio al set-point de 600 [mV].

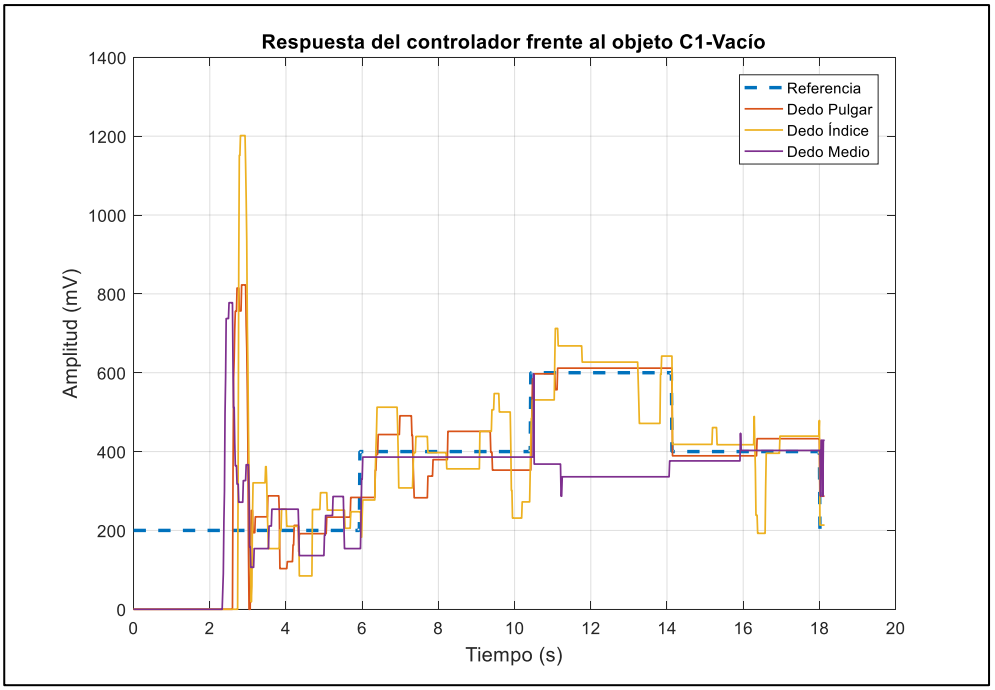


Figura 4.9 Respuesta de los controladores implementados, en el agarre del objeto C1-vacío
Fuente: Autores
Elaboración: Autores

Figura 4.10 Ajuste de los sensores FSR frente al objeto C1
Fuente: Autores
Elaboración: Autores

En la figura 4.11 se muestra la respuesta del controlador para el objeto C4 con 0% de peso adicional. Con este objeto los sensores FSR de los dedos pulgar, índice y medio presionan transversalmente, permitiendo al microcontrolador obtener una medida correcta de la fuerza aplicada por los actuadores.

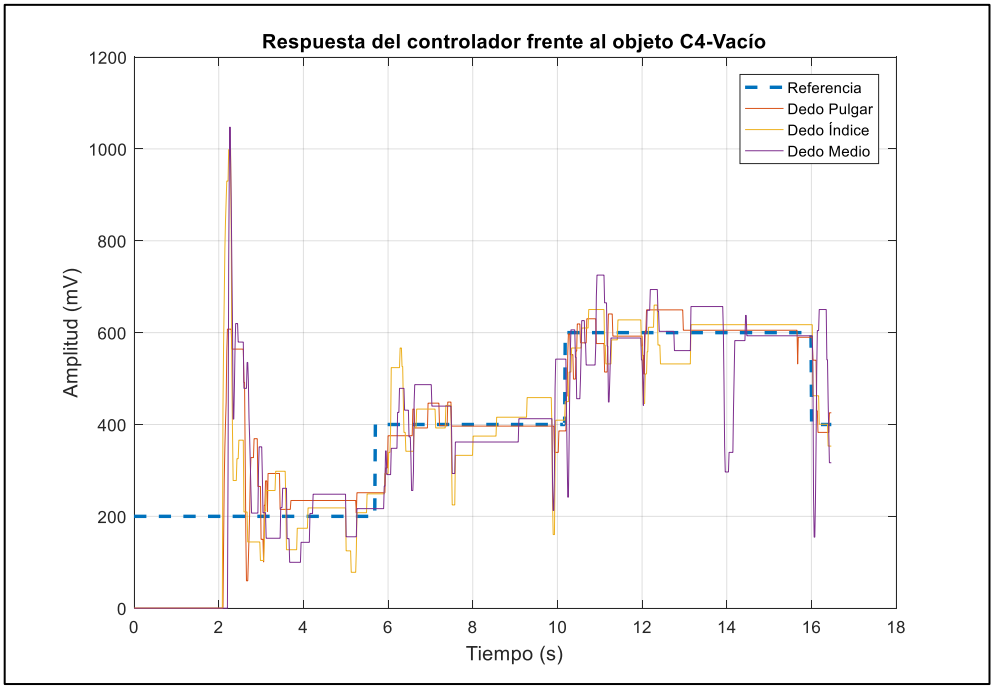


Figura 4.11 Respuesta de los controladores implementados, en el agarre del objeto C4-vacío
Fuente: Autores
Elaboración: Autores

La respuesta del controlador frente a los objetos C2, C3, como las fotografías de la prótesis efectuando el agarre a todos los objetos se encuentran en el anexo E.

Para las pruebas del controlador se aplicó los set-points especificados en el segundo párrafo del apartado 4.2.1 (ver tabla 4.4).

En la tabla 4.5 se describe si los objetos de prueba cilíndricos sufrieron un deslizamiento vertical marcando con la letra D (Deslizamiento), y para el caso contrario con las letras SD (Sin Deslizamiento).

Tabla 4.4 Set-points aplicados a los dedos de la prótesis robótica

Sistema o Dedo	S-P 1	S-P 2	S-P 3	S-P 4
Pulgar	400 [mV]	500 [mV]	600 [mV]	700 [mV]
Índice	200 [mV]	300 [mV]	400 [mV]	500 [mV]
Medio	200 [mV]	300 [mV]	400 [mV]	500 [mV]

Fuente: Autores
Elaboración: Autores

Tabla 4.5 Observación de deslizamiento de los objetos cilíndricos aplicado el control de fuerza a la prótesis robótica

Objeto	S-P 1	S-P 2	S-P 3	S-P 4	Peso
C2	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	D	SD	SD	SD	100%
C3	SD	SD	SD	SD	0%
	SD	SD	SD	SD	50%
	D	D	SD	SD	100%
C4	SD	SD	SD	SD	0%
	D	D	SD	SD	50%
	D	D	D	D	100%

Fuente: Autores
Elaboración: Autores

En la tabla 4.5 se ha exento el objeto C1 ya que no es posible aplicar el control de fuerza debido a que el domo del dedo medio no presiona el objeto transversalmente, mientras que, en los objetos C2, C3 y C4 es aplicable el control de fuerza con la retroalimentación mediante sensores FSR, ya que al efectuar el agarre todos los sensores se acoplan ejerciendo una fuerza transversal al objeto. Sin embargo, el método de sintonización, aceptado para el controlador basado en la medición de fuerza en el capítulo 3, no ha dado resultados positivos, no se han estabilizado los sistemas de los dedos en ningún objeto cilíndrico.

4.2.1.2. Agarre de objetos esféricos.

En las figuras 4.12 y 4.13 se muestra el agarre de la prótesis con los objetos E2 y E4, se aprecia que los dedos índice y medio no ejercen la fuerza desde el centro del domo,

comprometiendo el funcionamiento del controlador al obtener lecturas erróneas de la fuerza ejercida por el actuador de estos dedos. Ya que este efecto lo sufren todos los objetos esféricos no es posible aplicar el controlador de fuerza en este tipo de objetos.

En el anexo E se muestra el agarre de los objetos E1 y E3.

Figura 4.12 Ajuste de los sensores FSR frente al objeto E2
Fuente: Autores
Elaboración: Autores

Figura 4.13 Ajuste de los sensores FSR frente al objeto E4
Fuente: Autores
Elaboración: Autores

CONCLUSIONES

- Tras las simulaciones de los controladores estudiados para el desarrollo del controlador PID de fuerza de agarre en la prótesis robótica, se determinó que el método de sintonización Lambda, entre Ziegler-Nichols, Tyreus-Luyben, Shinskey, Shen-Yu y Cohen-Coon presenta mayor robustez frente a cambios de set-point así como disminuye el tiempo de estabilización para los controladores basados en retroalimentación mediante sensores FSR y de medición de corriente diseñados en el presente trabajo.

- Los controladores basados en medición de corriente, lograron la estabilidad de cada sistema al interactuar en conjunto, el tiempo de establecimiento es menor al segundo y el porcentaje de sobre-elongación es menor al 2% en cada uno de los sistemas.

- El controlador basado en la medición de fuerza de agarre mediante sensores FSR no logra una estabilización del sistema a pesar de que las simulaciones muestren lo contrario, se ha concluido que este efecto se genera porque los actuadores realizan movimientos lineales con una resolución baja al tener que iniciar con un 20% de la potencia total admisible para romper la inercia del mecanismo.

- El pre-procesado de la señal de retroalimentación para el sistema de control en el caso de la medición de corriente, agrega retardo al sistema. Esto a medida que se requiera más exactitud en la medición.

- Entre los dedos pulgar, índice y medio de la prótesis robótica se presenta distinta cinemática en cuanto a su diseño, por lo tanto, para considerar a los sensores FSR como la retroalimentación del controlador fue necesario determinar la respuesta que presentan cada uno frente a un escalón de 12V aplicados a los actuadores de los dedos. La información leída por los sensores es afectada por factores mecánicos, de construcción y principalmente porque cada sensor sigue una curva característica distinta a pesar de mantener una tendencia.

- Para la estimación RMS de la corriente, debe usarse un número de muestras tal que el error sea menor al 2%, para que el sistema sea estable.

- La primera versión del domo tipo hongo presionaba un área circular de 4.2 [mm] de diámetro del área activa del sensor FSR, sin embargo, la diferencia entre el diámetro inferior y el superior de éste al momento de agarrar un objeto cilíndrico presentaba fallas de inclinación del domo debido a la fuerza ejercida en el punto de presión y consecuentemente un desfase del punto de presión del domo con la zona activa del sensor FSR. El problema fue corregido en una nueva versión del domo agrandando el diámetro inferior a 5.4 [mm].

- El uso de algoritmos de estimación RMS como el EWMA, permite obtener una medición de la corriente eficaz de forma rápida, mejorando la calidad del controlador embebido.

- Puesto que cada sensor FSR ha presentado una curva característica distinta, a pesar de obtener sus datos de calibración en las mismas condiciones, se concluye que se debe calibrar cada sensor sin excepción.

BIBLIOGRAFÍA

[1] C. N. para la I. de Discapacidades, “Estadísticas de Discapacidad”, *Consejo Nacional para la Igualdad de Discapacidades*.

[2] C. A. Calderon, C. Ramirez, V. Barros, y G. Punin, “Design and Deployment of Grasp Control System applied to robotic hand prosthesis”, *IEEE Lat. Am. Trans.*, vol. 15, núm. 2, pp. 181–188, feb. 2017.

[3] A. S. Sadun, J. Jalani, J. Abdul Sukor, y F. Jamil, “Force control for a 3-Finger Adaptive Robot Gripper by using PID controller”, en *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, 2016, pp. 1–6.

[4] Ricardo Andrecioli y Erik D. Engeberg, “Adaptive sliding manifold slope via grasped object stiffness detection with a prosthetic hand”, *Mechatronics*, vol. 23, núm. 8, pp. 1171–1179, dic. 2013.

[5] S. D. Lee, K. H. Ahn, y J. B. Song, “Torque control based sensorless hand guiding for direct robot teaching”, en *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 745–750.

[6] J. S. Schofield, K. R. Evans, J. S. Hebert, P. D. Marasco, y J. P. Carey, “The effect of biomechanical variables on force sensitive resistor error: Implications for calibration and improved accuracy”, *J. Biomech.*, vol. 49, núm. 5, pp. 786–792, mar. 2016.

[7] J. G. Dabling, A. Filatov, y J. W. Wheeler, “Static and cyclic performance evaluation of sensors for human interface pressure measurement”, en *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012, pp. 162–165.

[8] R. Barone *et al.*, “Multilevel control of an anthropomorphic prosthetic hand for grasp and slip prevention”, *Adv. Mech. Eng.*, vol. 8, núm. 9, p. 1687814016665082, sep. 2016.

[9] Y. Bai, *Practical Microcontroller Engineering with ARM? Technology*. John Wiley & Sons, 2015.

[10] “ATSAM3X8E - 32-bit SAM Microcontrollers”. [En línea]. Disponible en: http://www.microchip.com/wwwproducts/en/ATsam3x8e. [Consultado: 16-oct-2017].

[11] J. Yiu, *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*. Newnes, 2013.

[12] M. Gutcher, C. Gimenez, S. Higgins, y D. Dougherty, “Cloud CubeSat Frictionless Testing Apparatus”, 2009.

[13] S. A. Dyer, *Wiley Survey of Instrumentation and Measurement*. John Wiley & Sons, 2004.

[14] B. Carter y R. Mancini, *Op Amps for Everyone*. Newnes, 2017.

[15] C. W. de Silva, *Sensor Systems: Fundamentals and Applications*. CRC Press, 2016.

[16] T. L. Floyd y D. Buchla, *Electronics Fundamentals: Circuits, Devices & Applications*. Pearson Education, Limited, 2013.

[17] D. G. Manolakis y V. K. Ingle, *Applied Digital Signal Processing: Theory and Practice*. Cambridge University Press, 2011.

[18] R. J. Schilling y S. L. Harris, *Digital Signal Processing using MATLAB*. Cengage Learning, 2016.

[19] “LM324-N Low Power Quad Operational Amplifier | TI.com”. [En línea]. Disponible en: http://www.ti.com/product/LM324-N?keyMatch=lm324&tisearch=Search-EN-Everything. [Consultado: 20-oct-2017].

[20] “Interlink Electronics”. [En línea]. Disponible en: http://interlinkelectronics.com/FSR400short.php. [Consultado: 29-jul-2017].

[21] A. Saeedi, F. Almasganj, y M. Pourebrahim, “Plantar pressure monitoring by developing a real-time wireless system”, en *2014 21th Iranian Conference on Biomedical Engineering (ICBME)*, 2014, pp. 211–214.

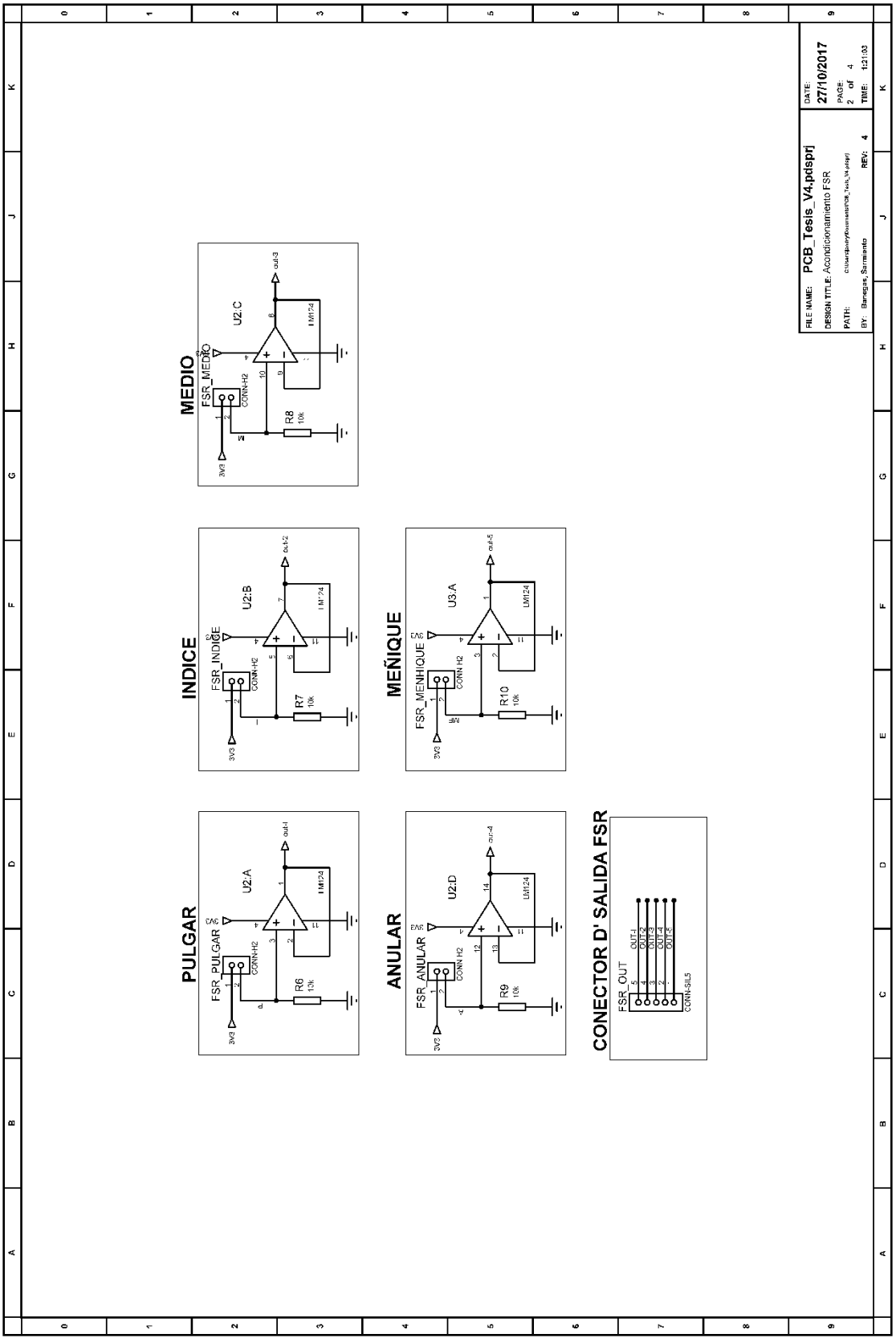
[22] J. A. Flórez y A. Velásquez, “Calibration of force sensing resistors (fsr) for static and dynamic applications”, en *2010 IEEE ANDESCON*, 2010, pp. 1–6.

[23] C. Georgopoulos, *Interface Fundamentals in Microprocessor-Controlled Systems*. Springer Science & Business Media, 2012.
[24] H. F. Rashvand y A. Abedi, *Wireless Sensor Systems for Extreme Environments: Space, Underwater, Underground, and Industrial*. John Wiley & Sons, 2017.
[25] K. Lynch, N. Marchuk, y M. Elwin, *Embedded Computing and Mechatronics with the PIC32 Microcontroller*. Newnes, 2015.
[26] D. Ashby *et al.*, *Circuit Design: Know It All*. Newnes, 2011.
[27] S. Smith, *Digital Signal Processing: A Practical Guide for Engineers and Scientists*. Newnes, 2013.
[28] S.-H. Kim, *Electric Motor Control: DC, AC, and BLDC Motors*. Elsevier, 2017.
[29] P. F. Ribeiro, C. A. Duque, P. M. Ribeiro, y A. S. Cerqueira, *Power Systems Signal Processing for Smart Grids*. John Wiley & Sons, 2013.
[30] X.-D. Zhang, *Matrix Analysis and Applications*. Cambridge University Press, 2017.
[31] V. Alfaro, *MÉTODOS DE SINTONIZACIÓN DE CONTROLADORES PID QUE OPERAN COMO SERVOMECANISMOS*, vol. 13. 2003.
[32] K. J. Åström y T. Hägglund, *Control PID avanzado*. Pearson Educación, 2009.
[33] D. Ibrahim, *Microcontroller Based Applied Digital Control*. Wiley, 2006.
[34] M. S. Fadali y A. Visioli, *Digital Control Engineering: Analysis and Design*. Academic Press, 2012.
[35] Ó. R. García, A. P. Vidal, y A. G. Aparicio, *Prácticas de Sistemas de Control - Continuos y Discretos*. Universidad Miguel Hernández, 2017.
[36] J. Jantzen, *Foundations of Fuzzy Control: A Practical Approach*. John Wiley & Sons, 2013.
[37] K. K. Tan, Q.-G. Wang, y C. C. Hang, *Advances in PID Control*. Springer Science & Business Media, 2012.
[38] A. Yufka y A. Yazici, "An Intelligent PID Tuning Method for an Autonomous Mobile Robot", 2010.
[39] S.-H. Shen y C.-C. Yu, "Use of relay-feedback test for automatic tuning of multivariable systems", *AIChE J.*, vol. 40, núm. 4, pp. 627–646, abr. 1994.
[40] M. King, *Process Control: A Practical Approach*. Wiley, 2016.
[41] "Fundamentals of lambda tuning | Control Engineering". [En línea]. Disponible en: https://www.controleng.com/single-article/fundamentals-of-lambda-tuning/aaea12b67526bff287b7bed3626df907.html. [Consultado: 19-oct-2017].
[42] "Tuning PID control loops for fast response | Control Engineering". [En línea]. Disponible en: https://www.controleng.com/single-article/tuning-pid-control-loops-for-fast-response/cbd4ca5011f3899a6173b6eb97bb8dd3.html. [Consultado: 19-oct-2017].
[43] "Arduino Due". [En línea]. Disponible en: https://store.arduino.cc/usa/arduino-due. [Consultado: 24-oct-2017].
[44] R. S. Hall, G. T. Desmoulin, y T. E. Milner, "A technique for conditioning and calibrating force-sensing resistors for repeatable and reliable measurement of compressive force", *J. Biomech.*, vol. 41, núm. 16, pp. 3492–3495, dic. 2008.
[45] "9-AB". [En línea]. Disponible en: http://www.abro.com/9-ab.html. [Consultado: 25-oct-2017].
[46] "Connected 3D Printing Solutions | MakerBot". [En línea]. Disponible en: https://www.makerbot.com/. [Consultado: 24-oct-2017].
[47] M. Pelgrom, *Analog-to-Digital Conversion*. Springer, 2016.
[48] U. B. Mujumdar y J. S. Joshi, "Microcontroller based true RMS current measurement under harmonic conditions", en *2010 IEEE International Conference on Sustainable Energy Technologies (ICSET)*, 2010, pp. 1–5.
[49] "Moving Root Mean Square - MATLAB". [En línea]. Disponible en: https://www.mathworks.com/help/dsp/ref/dsp.movingrms-system-object.html. [Consultado: 26-oct-2017].
[50] D. Krklješ, L. Nagy, y K. Babkovič, "Force-dependent contact area excitation of FSR force sensor utilizing dome-shaped rubber element", en *2012 28th International Conference on Microelectronics Proceedings*, 2012, pp. 181–184.

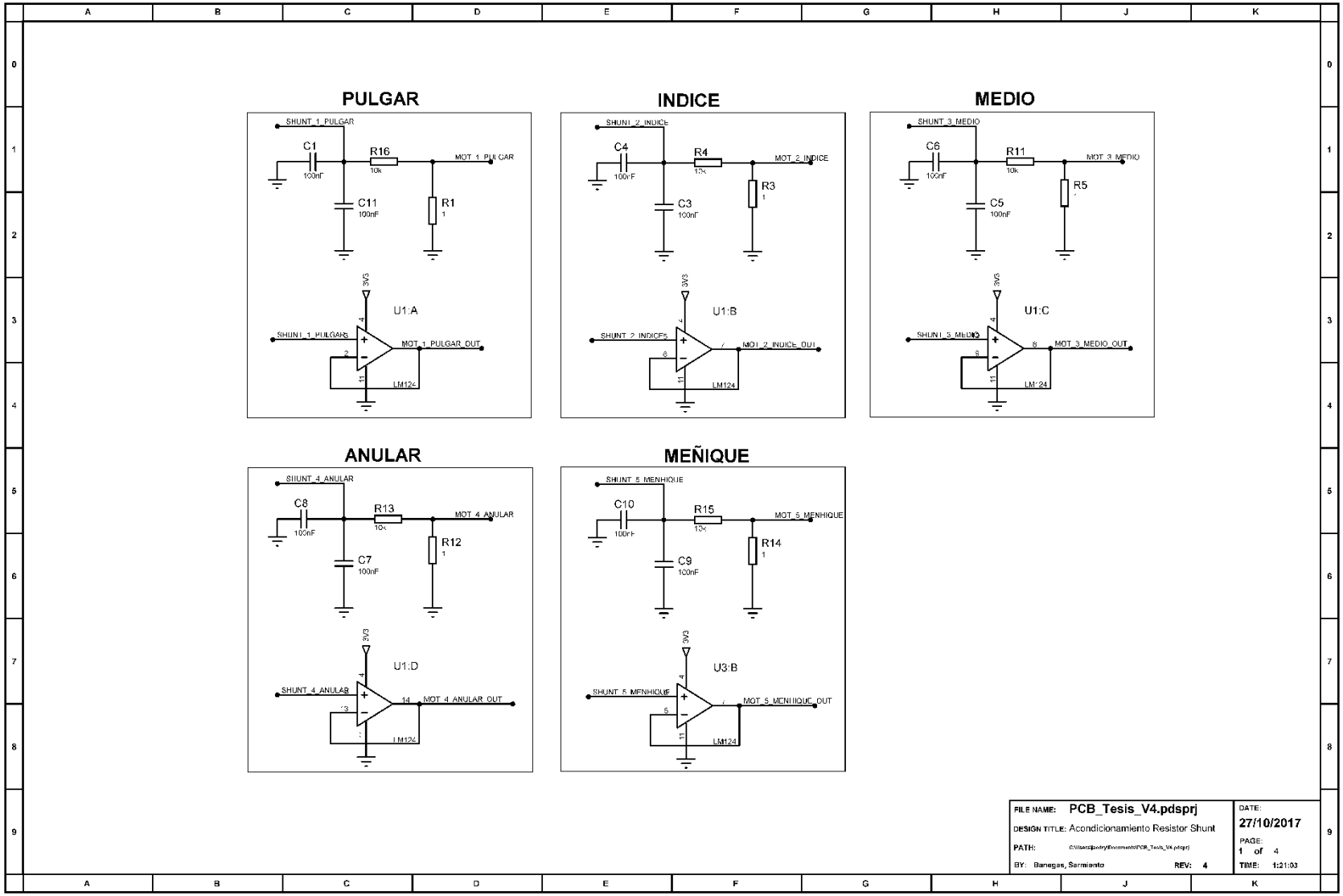
[51]“Load Cell - 10kg, Straight Bar (TAL220) - SEN-13329 - SparkFun Electronics”. [En línea]. Disponible en: https://www.sparkfun.com/products/13329. [Consultado: 26-oct-2017].
[52]“Arduino Mega 2560 R3”, *Arduino.cl*, 24-oct-2014. .
[53]“Balanza de precisión EMS - KERN & SOHN GmbH”. [En línea]. Disponible en: https://www.kern-sohn.com/shop/es/balanzas-de-laboratorio/balanzas-de-precision/EMS/. [Consultado: 26-oct-2017].
[54]“Loss Function and Model Quality Metrics - MATLAB & Simulink - MathWorks España”. [En línea]. Disponible en: https://es.mathworks.com/help/ident/ug/model-quality-metrics.html?requestedDomain=www.mathworks.com. [Consultado: 30-oct-2017].
[55]“Smallest Linear Actuator PQ12-P”, *www.actuonix.com*. [En línea]. Disponible en: https://www.actuonix.com//ProductDetails.asp?ProductCode=PQ12%2DP. [Consultado: 21-nov-2017].

ANEXOS

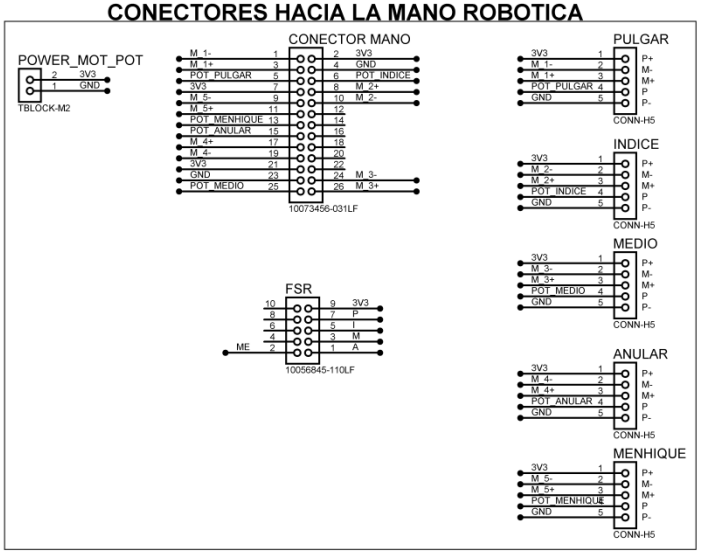
Anexo A



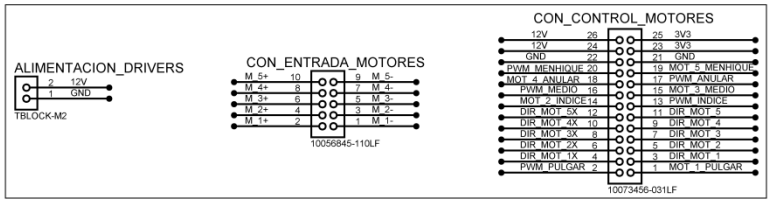
Esquemáticos del módulo de adquisición de datos.


PULGAR
SHUNT_1_PULGAR
C1
100nF
R16
10k
MOT_1_PULGAR
C11
100nF
R1
1
3V3
U1:A
SHUNT_1_PULGAR
MOT_1_PULGAR_OUT
LM124
INDICE
SHUNT_2_INDICE
C4
100nF
R4
10k
MOT_2_INDICE
C3
100nF
R3
1
3V3
U1:B
SHUNT_2_INDICE
MOT_2_INDICE_OUT
LM124
MEDIO
SHUNT_3_MEDIO
C6
100nF
R11
10k
MOT_3_MEDIO
C5
100nF
R5
1
3V3
U1:C
SHUNT_3_MEDIO
MOT_3_MEDIO_OUT
LM124
ANULAR
SHUNT_4_ANULAR
C8
100nF
R13
10k
MOT_4_ANULAR
C7
100nF
R12
1
3V3
U1:D
SHUNT_4_ANULAR
MOT_4_ANULAR_OUT
LM124
MEÑIQUE
SHUNT_5_MENHIQUE
C10
100nF
R15
10k
MOT_5_MENHIQUE
C9
100nF
R14
1
3V3
U3:B
SHUNT_5_MENHIQUE
MOT_5_MENHIQUE_OUT
LM124
FILE NAME: PCB_Tesis_V4.pdsprj
DESIGN TITLE: Acondicionamiento Resistor Shunt
PATH:
BY: Barregas, Sarmiento
REV: 4
DATE: 27/10/2017
PAGE: 1 of 4
TIME: 1:21:03

CONECTORES HACIA LA MANO ROBOTICA

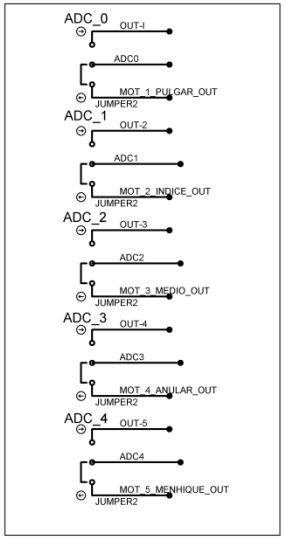


CONECTORES HACIA MODULO DE DRIVERS

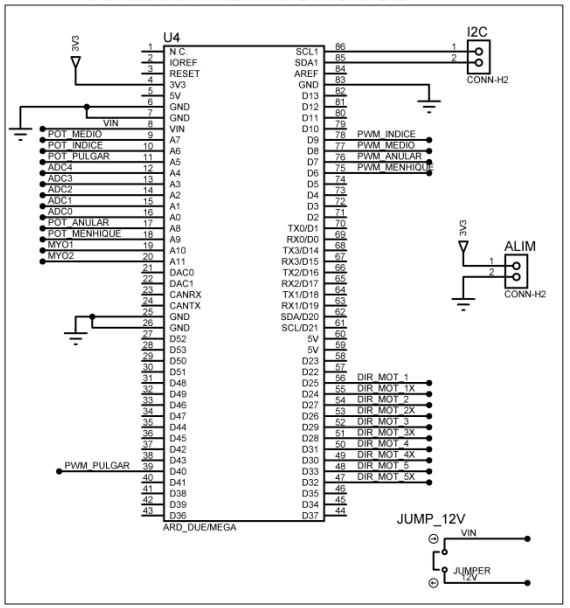


FILE NAME:	PCB_Tesis_V4.pdsprj
DESIGN TITLE:	Conectores Modula Adq. Datos
PATH:	C:\Users\jandry\Documents\PCB_Tesis_V4.pdsprj
BY:	Banegas, Sarmiento
REV:	4
DATE:	27/10/2017
PAGE:	3 of 4
TIME:	1:21:03

JUMPERS SELECCION DE SENSOR

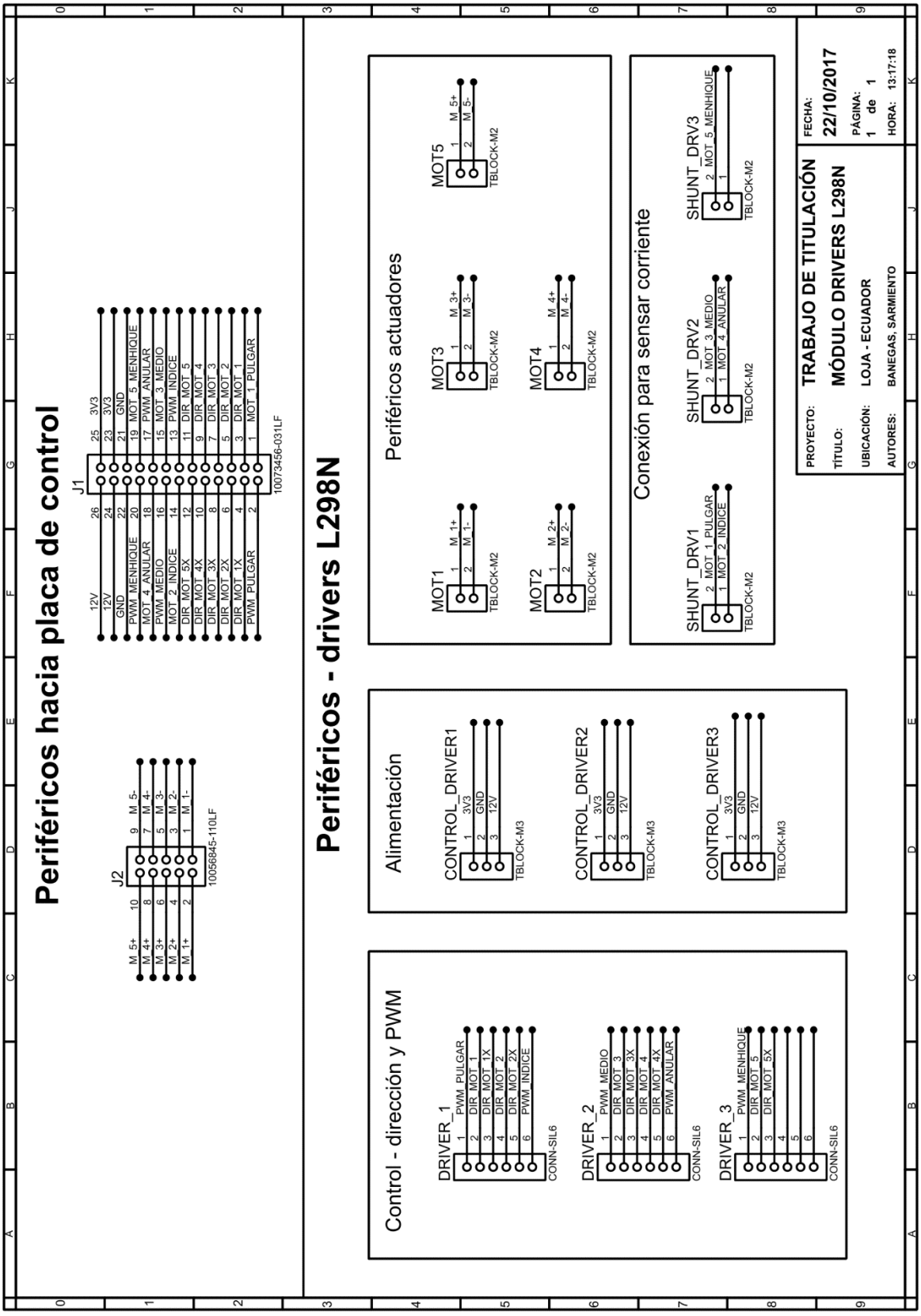


CONEXION AL ARDUINO DUE

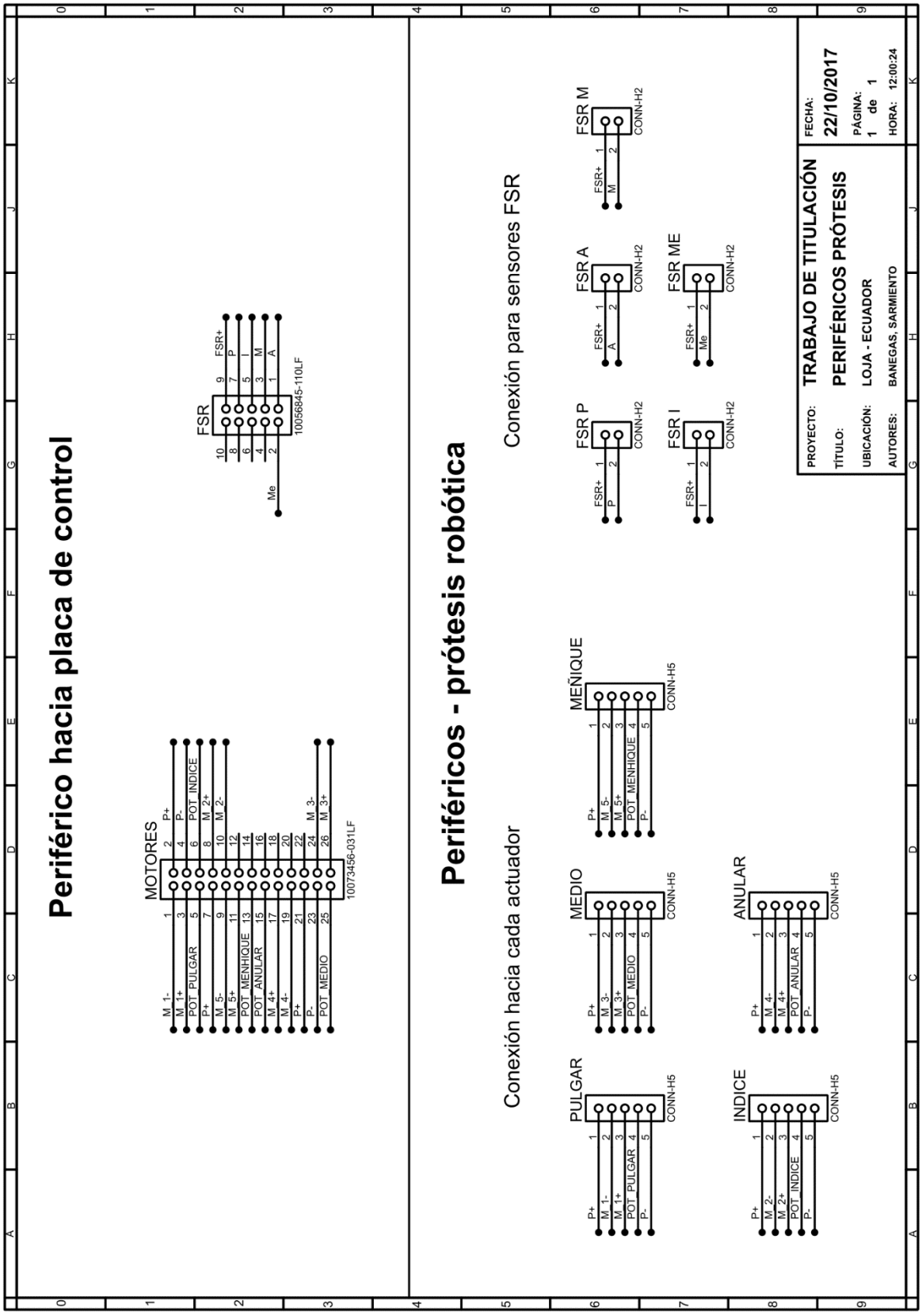


FILE NAME:	PCB_Tesis_V4.pdsprj
DESIGN TITLE:	Conexion del Arduino Due
PATH:	C:\Users\jandry\Documents\PCB_Tesis_V4.pdsprj
BY:	Banegas, Sarmiento
REV:	4
DATE:	27/10/2017
PAGE:	4 of 4
TIME:	1:21:03

Esquemático del módulo de conexión de los drivers L298N.



Esquemático del módulo de conexiones para periféricos de la prótesis robótica.



Anexo B

Código completo de implementación para el microcontrolador

```
1   /**
2   * Mano Robotica TT
3   * Autor: Jandry Oscar Banegas Rojas
4   * Coautor: Santiago Sarmiento
5   * Date: 25/08/17
6   * Para más información: jandcan@gmail.com
7   */
8   #define BAUD 115200
9   //#define FSR_MODO
10  #include <asf.h>
11  #include <string.h>
12  #include <arm_math.h>
13  #include "Mi_UART.h"
14  #include "Mi_Delay.h"
15  #include "Mano_Robotica.h"
16
17  #define BUFFER 6
18
19  #define PULGAR 0 /*CANAL A7*/
20  #define INDICE 1//CANAL A6
21  #define MEDIO  2 //CANAL A5
22  #define ANULAR 3 //CANAL A4
23  #define MENHIQUE 4 //CANAL A3
24
25  #define PWM_PULGAR PWM->PWM_CH_NUM[3].PWM_CDTY //PINC8, Arduino Pin 40
26  #define PWM_INDICE PWM->PWM_CH_NUM[4].PWM_CDTY //PINC21, Arduino Pin 9
27  #define PWM_MEDIO PWM->PWM_CH_NUM[5].PWM_CDTY //PINC22, Arduino Pin
28  #define PWM_ANULAR  PWM->PWM_CH_NUM[6].PWM_CDTY //PINC23, Arduino Pin 7
29  #define PWM_MENHIQUE PWM->PWM_CH_NUM[7].PWM_CDTY //PINC24, Arduino Pin
6
30
31  #define SETPOINT_FUERZA_INICIAL 400 //80 mA de corriente
32
33  volatile char cadena_PWM[BUFFER]={0};
34  volatile unsigned char buff=0;
35  volatile unsigned char PID_auto=0;
36  volatile unsigned int contador=1;
37  volatile unsigned int bajada=0;
38
39  volatile unsigned char habilitador=0;
40  volatile uint16_t buff_ADC_temp[5];
41  volatile unsigned char hab_nuevo_setpoint=0;
42  volatile unsigned char identificador_n_setpoint;
43  volatile unsigned int setpoint_val;
44
45  volatile uint32_t Contador=1;
46
47  typedef struct PID{
48    /*************************/
49    float32_t _k1;
50    float32_t _k2;
51    float32_t _k0;
52    float32_t _u_ant;
53    float32_t _u;
54    float32_t _error[3];
55    /************************/
56    float32_t _kp;
```

```
57    float32_t _ki;
58    float32_t _kd;
59    float32_t _P;
60    float32_t _I;
61    float32_t _D;
62    float32_t _Medida;
63    float32_t _Setpoint;
64    float32_t _Error;
65    float32_t _Error_ant;
66    float32_t _U;
67    unsigned char _PWM_MAX;
68    unsigned char _PWM_MIN;
69    float32_t _Delta_t;
70    /*
71    Variables Back-calculation Anti-reset wind up
72    */
73    float32_t _Anti_windup;
74    float32_t _ka;
75    float32_t _Error_sat;
76    float32_t _U_sat;
77
////////////////////////////////////////////////////////////////////////////
78    float32_t _Limite_inferior;
79    float32_t _Limite_superior;
80    unsigned char _dir_sal;
81    unsigned char _ident;
82  }PID;
83
84  typedef struct PID_INC{
85    float32_t _k1;
86    float32_t _k2;
87    float32_t _k3;
88    float32_t _u_ant;
89    float32_t _u;
90    float32_t _error[3];
91    float32_t _Setpoint;
92    unsigned char _PWM_MAX;
93    unsigned char _PWM_MIN;
94  }PID_INC;
95
96  typedef struct FSR_REG{
97    float32_t w[6];
98    float32_t y;
99  }FSR_REG;
100
101 typedef struct FSR_DERIV{
102   float32_t _x[2];
103   float32_t _y;
104   float32_t _limite;
105 }FSR_DERIV;
106
107 /*_PID Sensor FSR*/
108 PID PI_pulgar_FSR={
109   ._k1=-0.4183,
110   ._k2=0.4728,
111   ._k0=0,
112   ._u_ant=0,
113   ._error={0,0,0},
114
115   ._kp=0.18579,
116   ._ki=1.412,
```

```
117    ._kd=0,
118    ._ka=0,//ka=1/kp
119    ._Setpoint=400,//[newtons] 0.6296 voltios
120    ._P=0,
121    ._I=0,
122    ._D=0,
123    ._Anti_windup=0,
124    ._Error=0,
125    ._Error_ant=0,
126    ._dir_sal=0,
127    ._PWM_MAX=255,
128    ._PWM_MIN=55,
129    ._Delta_t=0.015,
130    ._ident= PULGAR
131  };
132  PID PI_indice_FSR={
133    ._k1=-0.1576,
134    ._k2=0.1799,
135    ._k0=0,
136    ._u_ant=0,
137    ._error={0,0,0},
138
139    ._kp=0.11786,
140    ._ki=1.8789,
141    ._kd=0,
142    ._ka=0,//ka=1/kp
143    ._Setpoint=200,//newtons, 1.268
144    ._P=0,
145    ._I=0,
146    ._D=0,
147    ._Anti_windup=0,
148    ._Error=0,
149    ._Error_ant=0,
150    ._dir_sal=0,
151    ._PWM_MAX=255,
152    ._PWM_MIN=55,
153    ._Delta_t=0.015,
154    ._ident= INDICE
155  };
156  PID PI_medio_FSR={
157    ._k1=-0.2053,
158    ._k2=0.2344,
159    ._k0=0,
160    ._u_ant=0,
161    ._error={0,0,0},
162
163    ._kp=0.1695,
164    ._ki=1.314,
165    ._kd=0,
166    ._ka=1/0.17437,//ka=1/kp
167    ._Setpoint=200,//newtons, 1.268
168    ._P=0,
169    ._I=0,
170    ._D=0,
171    ._Anti_windup=0,
172    ._Error=0,
173    ._Error_ant=0,
174    ._dir_sal=0,
175    ._PWM_MAX=255,
176    ._PWM_MIN=55,
177    ._Delta_t=0.015,
```

```
178    ._ident= MEDIO
179 };
180
181 void derivada(FSR_DERIV *,float32_t RAW);
182 void regresion(FSR_REG *,float32_t RAW);
183 float32_t pow_f32(float32_t base, unsigned char expo);
184
185 void Implementar_PID(PID *,float32_t);
186 static void ADC_config(void);
187 void lectura_dummy_ADC(void);
188
189 static void PWM_config(void);
190 static void Motores_Pines_config(void);
191
192 void Subir_mot(PID *, unsigned char);
193 void Bajar_mot(PID *, unsigned char);
194
195 int main (void)
196 {
197   /* Deshabilita el watchdog */
198   WDT->WDT_MR = WDT_MR_WDDIS;
199   Mano Mano_Robotica={
200
/*************************************************************************/
201     /* Configuracion dedo Pulgar
*/
202
/*************************************************************************/
203     ._Pulgar._Identificador=PULGAR, ._Pulgar._Nombre_str={"Pulgar"},
._Pulgar._IMOT=0,
204     ._Pulgar._Fir.y=0,
205     ._Pulgar._EWMA._wn_lambda=1,
206
/*************************************************************************/
207     /* Configuracion dedo Indice
*/
208
/*************************************************************************/
209     ._Indice._Identificador=INDICE, ._Indice._Nombre_str={"Indice"},
._Indice._IMOT=0,
210     ._Indice._Fir.y=0,
211     ._Indice._EWMA._wn_lambda=1,
212
/*************************************************************************/
213     /* Configuracion dedo Medio
*/
214
/*************************************************************************/
215     ._Medio._Identificador=MEDIO, ._Medio._Nombre_str={"Medio"},
._Medio._IMOT=0,
216     ._Medio._Fir.y=0,
217     ._Medio._EWMA._wn_lambda=1,
218
/*************************************************************************/
219     /* Configuracion dedo Anular
*/
220
/*************************************************************************/
221     ._Anular._Identificador=ANULAR, ._Anular._Nombre_str={"Anular"},
._Anular._IMOT=0,
222     ._Anular._Fir.y=0,
```

```
223      ._Anular._EWMA._wn_lambda=1,
224
/*************************************************************************/
225      /* Configuracion ded Meñique
*/
226
/*************************************************************************/
227      ._Menhique._Identificador=MENHIQUE,
._Menhique._Nombre_str={"Menhique"}, ._Menhique._IMOT=0,
228      //._Menhique._Fir.y=0,
229      //._Menhique._Nlms.c=C,
._Menhique._Nlms.mu=MU,._Menhique._Nlms.orden=LONG_FILT,
._Menhique._Temp={0,0,0},
230      //._Menhique._Rms.ventana=N_RMS,
231      ._Menhique._Fir.y=0,
232      ._Menhique._EWMA._wn_lambda=1
233
234    };
235    encerar_FIR(&Mano_Robotica._Pulgar._Fir);
236    encerar_FIR(&Mano_Robotica._Indice._Fir);
237    encerar_FIR(&Mano_Robotica._Medio._Fir);
238    encerar_FIR(&Mano_Robotica._Anular._Fir);
239    encerar_FIR(&Mano_Robotica._Menhique._Fir);
240
////////////////////////////////////////////////////////////////////////
241    //encerar_RMS(&Mano_Robotica._Pulgar._Rms);
242    /*_PID Corriente Shunt */
243    #ifndef FSR_MODO
244      PID PI_pulgar={
245        ._k1=-2,
246        ._k2=2.0054,
247        ._k0=0,
248        ._u_ant=0,
249        ._error={0,0,0},
250
251        ._kp=2,
252        ._ki=14.81,
253        ._kd=0,
254        ._ka=1/2,//ka=1/kp
255        ._Setpoint=0,
256        ._P=0,
257        ._I=0,
258        ._D=0,
259        ._Anti_windup=0,
260        ._Error=0,
261        ._Error_ant=0,
262        ._dir_sal=0,
263        ._PWM_MAX=255,
264        ._PWM_MIN=55,
265        ._Delta_t=0.02,
266        ._ident= PULGAR
267      };
268      PID PI_indice={
269        ._k1= -1.7510,
270        ._k2=1.7573,
271        ._k0=0,
272        ._u_ant=0,
273        ._error={0,0,0},
274
275        ._kp=1.751,
276        ._ki=9.726,
```

```
277         ._kd=0,
278         ._ka=1/1.751,//ka=1/kp
279         ._Setpoint=0,
280         ._P=0,
281         ._I=0,
282         ._D=0,
283         ._Anti_windup=0,
284         ._Error=0,
285         ._Error_ant=0,
286         ._dir_sal=0,
287         ._PWM_MAX=255,
288         ._PWM_MIN=55,
289         ._Delta_t=0.02,
290         ._ident= INDICE
291       };
292       PID PI_medio={
293         ._k1=-1.1530,
294         ._k2=1.1569,
295         ._k0=0,
296         ._u_ant=0,
297         ._error={0,0,0},
298 
299         ._kp=1.153,
300         ._ki=6.6741,
301         ._kd=0,
302         ._ka=1/1.153,//ka=1/kp
303         ._Setpoint=0,
304         ._P=0,
305         ._I=0,
306         ._D=0,
307         ._Anti_windup=0,
308         ._Error=0,
309         ._Error_ant=0,
310         ._dir_sal=0,
311         ._PWM_MAX=255,
312         ._PWM_MIN=55,
313         ._Delta_t=0.02,
314         ._ident= MEDIO
315       };
316       PID PI_anular={
317         ._kp=1.852,
318         ._ki=10.83,
319         ._kd=0,
320         ._ka=1/1.852,//ka=1/kp
321         ._Setpoint=0,
322         ._P=0,
323         ._I=0,
324         ._D=0,
325         ._Anti_windup=0,
326         ._Error=0,
327         ._Error_ant=0,
328         ._dir_sal=0,
329         ._PWM_MAX=255,
330         ._PWM_MIN=55,
331         ._Delta_t=0.02,
332         ._ident= ANULAR
333       };
334       PID PI_meniq={
335         ._kp=2.354,
336         ._ki=12.13,
337         ._kd=0,
```

```
338         ._ka=1/2.354,//ka=1/kp
339         ._Setpoint=0,
340         ._P=0,
341         ._I=0,
342         ._D=0,
343         ._Anti_windup=0,
344         ._Error=0,
345         ._Error_ant=0,
346         ._dir_sal=0,
347         ._PWM_MAX=255,
348         ._PWM_MIN=55,
349         ._Delta_t=0.02,
350         ._ident= MENHIQUE
351       };
352     #endif
353     #ifdef FSR_MODO
354
355       /*Estructuras de mapeo FSR*/
356       FSR_REG Pulgar_Reg={
357         .w={0.055308,-0.24423,0.43799,-0.13361,0.012399}
358       };
359       FSR_REG Indice_Reg={
360         .w={-0.032838,0.58453,-0.044065,-0.010114,0.0013245}
361       };
362       FSR_REG Medio_Reg={
363         .w={-0.35918,0.21473,0.021655,-0.0044178,0.00019276}
364       };
365
366       FSR_DERIV Pulgar_dev={
367         ._x={0,0},
368         ._y=0,
369         ._limite=64
370       };
371       FSR_DERIV Indice_dev={
372         ._x={0,0},
373         ._y=0,
374         ._limite=64
375       };
376       FSR_DERIV Medio_dev={
377         ._x={0,0},
378         ._y=0,
379         ._limite=64
380       };
381     #endif
382
383     sysclk_init();
384     SysTick_Config(SystemCoreClock/1000); //se colocara en milisegundos
el delay
385
386     Motores_Pines_config();
387     UART_config();
388     PWM_config();
389     ADC_config();
390     //adc_start(ADC);
391     char _cadena[100];
392     lectura_dummy_ADC();
393     /*
394     PIOC->PIO_PER|=PIO_PC13;
395     PIOC->PIO_OER|=PIO_PC13;
396     PIOC->PIO_PUDR|=PIO_PC13;
397     */
```

```
398
399    #ifdef PLOTEAR_DATA
400    uint32_t Contador=0;
401    uint8_t PWM_Auto=0;
402    #endif
403
404    while(1)
405    {
406      adc_start(ADC);
407
408      if (hab_nuevo_setpoint)
409      {
410        hab_nuevo_setpoint=0;//encera la bandera
411        /*sprintf(_cadena,"nuevo setpoint Indet %d, setpoint
%d\n",identificador_n_setpoint,setpoint_val);
412        Enviar_UART(_cadena);*/
413        switch(identificador_n_setpoint){//identifica a que dedo debe
cambiar el setpoint
414        #ifndef FSR_MODO
415          case  PULGAR:
416          PI_pulgar._Setpoint=setpoint_val;
417          break;
418          case INDICE:
419          PI_indice._Setpoint=setpoint_val;
420          break;
421          case MEDIO:
422          PI_medio._Setpoint=setpoint_val;
423          break;
424          case ANULAR:
425          PI_anular._Setpoint=setpoint_val;
426          break;
427          case MENHIQUE:
428          PI_meniq._Setpoint=setpoint_val;
429          break;
430        #else
431          case  PULGAR:
432          PI_pulgar_FSR._Setpoint=setpoint_val;
433          break;
434          case INDICE:
435          PI_indice_FSR._Setpoint=setpoint_val;
436          break;
437          case MEDIO:
438          PI_medio_FSR._Setpoint=setpoint_val;
439          break;
440        #endif
441        }
442        }else if(habilitador){
443
444        habilitador=0;
445        Mano_Robotica._Buffer_ADC=&buff_ADC_temp[0];// hace una copia del
buffer ADC usado en el PDC
446        //
447
Obtener_Corriente_Actuador(&Mano_Robotica,&Mano_Robotica._Pulgar);
448
Obtener_Corriente_Actuador(&Mano_Robotica,&Mano_Robotica._Indice);
449        Obtener_Corriente_Actuador(&Mano_Robotica,&Mano_Robotica._Medio);
450
Obtener_Corriente_Actuador(&Mano_Robotica,&Mano_Robotica._Anular);
451
Obtener_Corriente_Actuador(&Mano_Robotica,&Mano_Robotica._Menhique);
```

```
452         #ifdef FSR_MODO
453         derivada(&Pulgar_dev,Mano_Robotica._Pulgar._IMOT);
454         derivada(&Indice_dev,Mano_Robotica._Indice._IMOT);
455         derivada(&Medio_dev,Mano_Robotica._Medio._IMOT);
456         #endif
457         #ifndef FSR_MODO
458         //Enviar_Data_Mano(&Mano_Robotica);
459
sprintf(_cadena,"d%.2f,,#d%.2f#,,d%.2f#,,d%.2f#\n",Mano_Robotica._Pulgar._E
WMA._y[1],Mano_Robotica._Indice._EWMA._y[1],
Mano_Robotica._Medio._EWMA._y[1],PI_pulgar._Setpoint);
460         Enviar_UART(_cadena);
461         #endif
462
463         if (PID_auto)
464         {
465
466           /**Rutina Cambio de Setpoint*/
467         #ifndef FSR_MODO
468           if (contador==200 && bajada==0)
469           {
470             //Enviar_UART("Incrementando...");
471             //_Delay_ms(100);
472             contador=1;
473             if (PI_pulgar._Setpoint==100)
474             {
475               bajada=1;
476               }else{
477               PI_pulgar._Setpoint+=25;
478               PI_indice._Setpoint+=25;
479               PI_medio._Setpoint+=25;
480               PI_anular._Setpoint+=25;
481               PI_meniq._Setpoint+=25;
482             }
483           }
484           if (contador==200 && bajada==1)
485           {
486             //Enviar_UART("Incrementando...");
487             //_Delay_ms(100);
488             contador=1;
489             if (PI_pulgar._Setpoint==25)
490             {
491               bajada=0;
492               }else{
493               PI_pulgar._Setpoint-=25;
494               PI_indice._Setpoint-=25;
495               PI_medio._Setpoint-=25;
496               PI_anular._Setpoint-=25;
497               PI_meniq._Setpoint-=25;
498             }
499           }
500         #else
501
502           if (contador==600 && bajada==0)
503           {
504             //Enviar_UART("Incrementando...");
505             //_Delay_ms(100);
506             /*if (PI_indice_FSR._Setpoint==200)
507             {
508               PI_indice_FSR._PWM_MAX=100;
509             }else{
```

```
510             PI_indice_FSR._PWM_MAX=200;
511           }*/
512           contador=1;
513           if (PI_pulgar_FSR._Setpoint==800)
514           {
515             bajada=1;
516             }else{
517             PI_pulgar_FSR._Setpoint+=100;
518             PI_indice_FSR._Setpoint+=100;
519             PI_medio_FSR._Setpoint+=100;
520           }
521         }
522         if (contador==600 && bajada==1)
523         {
524           //Enviar_UART("Incrementando...");
525           //_Delay_ms(100);
526           contador=1;
527           if (PI_pulgar_FSR._Setpoint==400)
528           {
529             bajada=0;
530           }else{
531             PI_pulgar_FSR._Setpoint-=100;
532             PI_indice_FSR._Setpoint-=100;
533             PI_medio_FSR._Setpoint-=100;
534           }
535         }
536         #endif
537         contador++;
538
539         #ifndef FSR_MODO
540         Implementar_PID(&PI_pulgar,Mano_Robotica._Pulgar._EWMA._y[1]);
541         Implementar_PID(&PI_indice,Mano_Robotica._Indice._EWMA._y[1]);
542         Implementar_PID(&PI_medio,Mano_Robotica._Medio._EWMA._y[1]);
543
//Implementar_PID(&PI_anular,Mano_Robotica._Anular._EWMA._y[1]);
544
//Implementar_PID(&PI_meniq,Mano_Robotica._Menhique._EWMA._y[1]);
545         #endif
546         #ifdef FSR_MODO
547         regresion(&Pulgar_Reg,3);
548         regresion(&Indice_Reg,3);
549         /*PI_pulgar_FSR._Setpoint=500;//Pulgar_Reg.y*1000;
550         PI_indice_FSR._Setpoint=500;//Indice_Reg.y*1000;
551         PI_medio_FSR._Setpoint=500;//Indice_Reg.y*1000;*/
552         //if (PI_pulgar_FSR._error[0]<=40)
553         //{
554
//Implementar_PID(&PI_pulgar_FSR,Pulgar_dev._y);//Mano_Robotica._Pulgar._IM
OT);
555           //
556         //}
557         //else
558         //{
559
//Implementar_PID(&PI_pulgar_FSR,Mano_Robotica._Pulgar._IMOT);//Mano_Roboti
ca._Pulgar._IMOT);
560           //
561         //}
562         if (PI_indice_FSR._error[0]<=40 || PI_indice_FSR._error[0]>=-
40)
563         {
```

```
564             Implementar_PID(&PI_indice_FSR,Indice_dev._y);
565           }
566           else
567           {
568             Implementar_PID(&PI_indice_FSR,Mano_Robotica._Indice._IMOT);
569           }
570           if (PI_medio_FSR._error[0]<=40 || PI_indice_FSR._error[0]>=-40)
571           {
572
573
Implementar_PID(&PI_medio_FSR,Medio_dev._y);//Mano_Robotica._Medio._IMOT);/
574           }
575           else
576           {
577
Implementar_PID(&PI_medio_FSR,Mano_Robotica._Medio._IMOT);//Mano_Robotica._
Medio._IMOT);/
578           }
579
Implementar_PID(&PI_pulgar_FSR,Mano_Robotica._Pulgar._IMOT);//Mano_Robotica
._Pulgar._IMOT);
580           //Implementar_PID(&PI_indice_FSR,Mano_Robotica._Indice._IMOT);
581
//Implementar_PID(&PI_medio_FSR,Mano_Robotica._Medio._IMOT);//Mano_Robotica
. Medio. IMOT);/*/
582           #endif
583
584         }
585         #ifndef FSR_MODO
586           _Delay_ms(20);
587         #endif
588         #ifdef FSR_MODO
589
sprintf(_cadena,"d%.2f#d%.2f#",PI_indice_FSR._Setpoint,Mano_Robotica._Pulga
r._IMOT);
590           Enviar_UART(_cadena);
591           if (PI_indice_FSR._error[0]<=40 || PI_indice_FSR._error[0]>=-40
)
592           {
593             sprintf(_cadena,"d%.2f#",Indice_dev._y);
594             Enviar_UART(_cadena);
595           }
596           else
597           {
598             sprintf(_cadena,"d%.2f#",Mano_Robotica._Indice._IMOT);
599             Enviar_UART(_cadena);
600           }
601           if (PI_medio_FSR._error[0]<=40 || PI_medio_FSR._error[0]>=-40 )
602           {
603             sprintf(_cadena,"d%.2f#\n",Medio_dev._y);
604             Enviar_UART(_cadena);
605           }
606           else
607           {
608             sprintf(_cadena,"d%.2f#\n",Mano_Robotica._Medio._IMOT);
609             Enviar_UART(_cadena);
610           }
611
612
//sprintf(_cadena,"d%.2f#d%.2f#d%.2f#d%.2f#\n",Pulgar_dev._y,Indice_dev._y,
Medio_dev._y,PI_pulgar_FSR._Setpoint);
```

```
613           //Enviar_UART(_cadena);
614           _Delay_ms(15);
615         #endif
616       }
617     }
618 }
619
/*****************************************************************************/
620 /* Interrupciones
*/
621
/*****************************************************************************/
622 void UART_Handler(){
623   //volatile uint32_t ul_status;
624   char dato=UART->UART_RHR;
625   if (dato=='S')
626   {
627     PID_auto=(PID_auto==0)?1:0;
628   }
629   if (dato=='L')
630   {
631     PI_pulgar_FSR._Setpoint+=100;
632     PI_indice_FSR._Setpoint+=100;
633     PI_medio_FSR._Setpoint+=100;
634   }
635   if (dato=='K')
636   {
637     PI_pulgar_FSR._Setpoint-=100;
638     PI_indice_FSR._Setpoint-=100;
639     PI_medio_FSR._Setpoint-=100;
640   }
641   if (dato=='C')
642   {
643     PWM_PULGAR=255;
644     PWM_INDICE=255;
645     PWM_MEDIO=255;
646     PWM_ANULAR=255;
647     PWM_MENHIQUE=255;
648     /*Direcciones*/
649     PIOD->PIO_SODR|=PIO_PD0;
650     PIOA->PIO_CODR|=PIO_PA15;
651
652     PIOD->PIO_SODR|=PIO_PD2;
653     PIOD->PIO_CODR|=PIO_PD1;
654
655     PIOD->PIO_SODR|=PIO_PD6;
656     PIOD->PIO_CODR|=PIO_PD3;
657     #ifndef FSR_MODO
658       PIOA->PIO_CODR|=PIO_PA7;
659       PIOD->PIO_SODR|=PIO_PD9;
660       PIOC->PIO_CODR|=PIO_PC1;
661       PIOD->PIO_SODR|=PIO_PD10;
662     #endif
663   }
664   if (dato=='O')
665   {
666     PWM_PULGAR=255;
667     PWM_INDICE=255;
668     PWM_MEDIO=255;
669     PWM_ANULAR=255;
670     PWM_MENHIQUE=255;
```

```
671     /*Direcciones*/
672     PIOD->PIO_CODR|=PIO_PD0;
673     PIOA->PIO_SODR|=PIO_PA15;
674
675     PIOD->PIO_CODR|=PIO_PD2;
676     PIOD->PIO_SODR|=PIO_PD1;
677
678     PIOD->PIO_CODR|=PIO_PD6;
679     PIOD->PIO_SODR|=PIO_PD3;
680     #ifndef FSR_MODO
681       PIOA->PIO_SODR|=PIO_PA7;
682       PIOD->PIO_CODR|=PIO_PD9;
683
684       PIOC->PIO_SODR|=PIO_PC1;
685       PIOD->PIO_CODR|=PIO_PD10;
686     #endif
687   }
688   if (dato=='P')
689   {
690     buff=0;
691     //dato=leerCaracter_Usart();
692     while(buff<BUFFER){
693       dato=LeerCaracter_UART();
694       if (dato != '?')
695       {
696         cadena_PWM[buff++]=dato;
697         }else{
698         cadena_PWM[buff]='\0';
699         if (!PID_auto)
700         {
701           PWM_PULGAR=atoi((char *)cadena_PWM);//setea el PWM
702         }
703         else
704         {
705           setpoint_val=atoi((char *)cadena_PWM);//setea el setpoint
706           identificador_n_setpoint=PULGAR;
707           hab_nuevo_setpoint=1;
708         }
709         break;
710       }
711     }
712     //Enviar_UART("\n\r PWM MOT1: "),Enviar_UART(cadena_PWM);
713   }
714   if (dato == 'I')
715   {
716     buff=0;
717     //dato=leerCaracter_Usart();
718     while(buff<BUFFER){
719       dato=LeerCaracter_UART();
720       if (dato != '?')
721       {
722         cadena_PWM[buff++]=dato;
723         }else{
724         cadena_PWM[buff]='\0';
725         if (!PID_auto)
726         {
727           PWM_INDICE=atoi((char *)cadena_PWM);//setea el PWM
728         }
729         else
730         {
731           setpoint_val=atoi((char *)cadena_PWM);//setea el setpoint
```

```
732             identificador_n_setpoint=INDICE;
733             hab_nuevo_setpoint=1;
734           }
735           break;
736         }
737       }
738       //Enviar_UART("\n\r PWM MOT2: "),Enviar_UART(cadena_PWM);
739     }
740     if (dato == 'M')
741     {
742       buff=0;
743       //dato=leerCaracter_Usart();
744       while(buff<BUFFER){
745         dato=LeerCaracter_UART();
746         if (dato != '?')
747         {
748           cadena_PWM[buff++]=dato;
749           }else{
750           cadena_PWM[buff]='\0';
751           if (!PID_auto)
752           {
753             PWM_MEDIO=atoi((char *)cadena_PWM);//setea el PWM
754           }
755           else
756           {
757             setpoint_val=atoi((char *)cadena_PWM);//setea el setpoint
758             identificador_n_setpoint=MEDIO;
759             hab_nuevo_setpoint=1;
760           }
761           break;
762         }
763       }
764       //Enviar_UART("\n\r PWM MOT2: "),Enviar_UART(cadena_PWM);
765     }
766     if (dato == 'A')
767     {
768       buff=0;
769       //dato=leerCaracter_Usart();
770       while(buff<BUFFER){
771         dato=LeerCaracter_UART();
772         if (dato != '?')
773         {
774           cadena_PWM[buff++]=dato;
775           }else{
776           cadena_PWM[buff]='\0';
777           if (!PID_auto)
778           {
779             PWM_ANULAR=atoi((char *)cadena_PWM);//setea el PWM
780           }
781           else
782           {
783             setpoint_val=atoi((char *)cadena_PWM);//setea el setpoint
784             identificador_n_setpoint=ANULAR;
785             hab_nuevo_setpoint=1;
786           }
787           break;
788         }
789       }
790       //Enviar_UART("\n\r PWM MOT2: "),Enviar_UART(cadena_PWM);
791     }
792     if (dato == 'N')
```

```
793   {
794     buff=0;
795     //dato=leerCaracter_Usart();
796     while(buff<BUFFER){
797       dato=LeerCaracter_UART();
798       if (dato != '?')
799       {
800         cadena_PWM[buff++]=dato;
801         }else{
802         cadena_PWM[buff]='\0';
803         if (!PID_auto)
804         {
805           PWM_MENHIQUE=atoi((char *)cadena_PWM);//setea el PWM
806         }
807         else
808         {
809           setpoint_val=atoi((char *)cadena_PWM);//setea el setpoint
810           identificador_n_setpoint=MENHIQUE;
811           hab_nuevo_setpoint=1;
812         }
813         break;
814       }
815     }
816     //Enviar_UART("\n\r PWM MOT2: "),Enviar_UART(cadena_PWM);
817   }
818   /*
819   MOTOR 1 -> PINES D0 y A15 (25,24)
820   MOTOR 2-> PINES D2 y D1 (27,26)
821   MOTOR 3 -> PINES D6 y D3 (29,28)
822   MOTOR 4 -> PINES A7 y D9 (31,30)
823   MOTOR 5 -> PINES C1 y D10 (33,32)
824   */
825
826   if (dato=='a'){
827     PIOD->PIO_SODR|=PIO_PD0;
828     PIOA->PIO_CODR|=PIO_PA15;
829   }
830   if (dato == 's'){
831     PIOD->PIO_CODR|=PIO_PD0;
832     PIOA->PIO_SODR|=PIO_PA15;
833   }
834   if (dato == 'q')
835   {
836     PIOD->PIO_SODR|=PIO_PD2;
837     PIOD->PIO_CODR|=PIO_PD1;
838   }
839   if (dato == 'w')
840   {
841     PIOD->PIO_CODR|=PIO_PD2;
842     PIOD->PIO_SODR|=PIO_PD1;
843   }
844   if (dato == 'e')
845   {
846     PIOD->PIO_SODR|=PIO_PD6;
847     PIOD->PIO_CODR|=PIO_PD3;
848   }
849   if (dato == 'r')
850   {
851     PIOD->PIO_CODR|=PIO_PD6;
852     PIOD->PIO_SODR|=PIO_PD3;
853   }
```

```
854   if (dato == 'd')
855   {
856     PIOA->PIO_SODR|=PIO_PA7;
857     PIOD->PIO_CODR|=PIO_PD9;
858   }
859   if (dato == 'f')
860   {
861     PIOA->PIO_CODR|=PIO_PA7;
862     PIOD->PIO_SODR|=PIO_PD9;
863   }
864   if (dato == 't')
865   {
866     PIOC->PIO_SODR|=PIO_PC1;
867     PIOD->PIO_CODR|=PIO_PD10;
868   }
869   if (dato == 'y')
870   {
871     PIOC->PIO_CODR|=PIO_PC1;
872     PIOD->PIO_SODR|=PIO_PD10;
873   }
874 }
875 void ADC_Handler(){
876   if ((adc_get_status(ADC)& ADC_ISR_RXBUFF) == ADC_ISR_RXBUFF)
877   {
878     habilitador=1;
879     PDC_ADC->PERIPH_RPR = (uint32_t) buff_ADC_temp; // RPR almacena la
direccion del buffer para el DMA
880     PDC_ADC->PERIPH_RCR = sizeof(buff_ADC_temp);
881     PDC_ADC->PERIPH_PTCR = PERIPH_PTCR_RXTEN; // Hablita la recepcion
del controlador de perifericos DMA,  PDM en ingles
882   }
883 }
884 void SysTick_Handler(){
885   msTics++;
886 }
887
888
/*********************************************************************/
889 /* FUNCIONES
*/
890
/*********************************************************************/
891
892 void ADC_config(){
893   //Arduino due tiene los puertos A0-A7 conectados de forma inversa,
HsDP no saben rutear un culo
894   pmc_enable_periph_clk(ID_ADC);
895   adc_init(ADC, SystemCoreClock, ADC_FREQ_MAX, ADC_STARTUP_FAST);
896   adc_disable_interrupt(ADC, 0xFFFFFFFF);//deshabilita todas las
interrupciones
897   adc_configure_timing(ADC, 0, ADC_SETTLING_TIME_3, 1);
898   adc_set_resolution(ADC, ADC_MR_LOWRES_BITS_12);
899   ADC->ADC_MR|=ADC_MR_FWUP_OFF|ADC_MR_SLEEP_NORMAL;
900   //adc_configure_power_save(ADC, ADC_MR_SLEEP_NORMAL,
ADC_MR_FWUP_OFF);
901   adc_set_bias_current(ADC, 1); // Bias current - maximum performance
over current consumption
902   adc_stop_sequencer(ADC); // not using it
903   adc_disable_all_channel(ADC);
904   adc_enable_channel(ADC,PULGAR);
905   adc_enable_channel(ADC,INDICE);
```

```
906   adc_enable_channel(ADC,MEDIO);
907   adc_enable_channel(ADC,ANULAR);
908   adc_enable_channel(ADC,MENHIQUE);
909   //adc_enable_channel(ADC,5);
910   //adc_enable_interrupt(ADC, ADC_IER_DRDY);
911   //NVIC_EnableIRQ((IRQn_Type) ID_ADC);
912   adc_enable_interrupt(ADC,ADC_IER_RXBUFF);
913   NVIC_EnableIRQ((IRQn_Type) ID_ADC);
914   adc_configure_trigger(ADC, ADC_TRIG_SW, 0);
915   //adc_start(ADC);
916   //lectura_dummy_ADC();// funcion para evitar errores en el ADC segun
errata del datasheet
917 }
918 /*
919
920 */
921 void lectura_dummy_ADC(){
922   for (unsigned char i=0;i<15;i++)
923   {
924     adc_start(ADC);
925     while (habilitador){
926       habilitador=0;
927       _Delay_ms(10);
928     }
929   }
930 }
931 /*
932
933 */
934 void PWM_config(){
935
936   pmc_enable_periph_clk(ID_PWM);
937   PIOC->PIO_PER=0;//deshabilita todo
938   PIOC-
>PIO_PDR=PIO_PC8B_PWML3|PIO_PC21B_PWML4|PIO_PC22B_PWML5|PIO_PC23B_PWML6|PIO
_PC24B_PWML7; //Habilita el puerto PWM
939   PIOC->PIO_IER=0;//deshabilita todas las interrupcione ns
940   PIOC->PIO_IDR=0xFFFFFFFF;
941   PIOC-
>PIO_ABSR|=PIO_PC8B_PWML3|PIO_PC21B_PWML4|PIO_PC22B_PWML5|PIO_PC23B_PWML6|P
IO_PC24B_PWML7; //Asignar el pin al  periferico B///
942   //**********************************Config Canal 3L PINC8, Arduino
Pin 40
943   PWM->PWM_CH_NUM[3].PWM_CMR= PWM_CMR_CPRE_MCK_DIV_128;
944   PWM->PWM_CH_NUM[3].PWM_CPRD=255;
945   PWM->PWM_CH_NUM[3].PWM_CDTY=0;
946   PWM->PWM_CH_NUM[3].PWM_CMR|=PWM_CMR_CALG;
947   PWM->PWM_ENA = PWM_ENA_CHID3;
948   //**********************************Config Canal 4L PINC21, Arduino
Pin 9
949   PWM->PWM_CH_NUM[4].PWM_CMR= PWM_CMR_CPRE_MCK_DIV_128;
950   PWM->PWM_CH_NUM[4].PWM_CPRD=255;
951   PWM->PWM_CH_NUM[4].PWM_CDTY=0;
952   PWM->PWM_CH_NUM[4].PWM_CMR|=PWM_CMR_CALG;
953   PWM->PWM_ENA |= PWM_ENA_CHID4;
954   //**********************************Config Canal 5L PINC22, Arduino
Pin 8
955   PWM->PWM_CH_NUM[5].PWM_CMR= PWM_CMR_CPRE_MCK_DIV_128;
956   PWM->PWM_CH_NUM[5].PWM_CPRD=255;
957   PWM->PWM_CH_NUM[5].PWM_CDTY=0;
958   PWM->PWM_CH_NUM[5].PWM_CMR|=PWM_CMR_CALG;
```

```
959    PWM->PWM_ENA |= PWM_ENA_CHID5;
960    //*********************************Config Canal 6L PINC23, Arduino
Pin 7
961    PWM->PWM_CH_NUM[6].PWM_CMR= PWM_CMR_CPRE_MCK_DIV_128;
962    PWM->PWM_CH_NUM[6].PWM_CPRD=255;
963    PWM->PWM_CH_NUM[6].PWM_CDTY=0;
964    PWM->PWM_CH_NUM[6].PWM_CMR|=PWM_CMR_CALG;
965    PWM->PWM_ENA |= PWM_ENA_CHID6;
966    //*********************************Config Canal 7L PINC24, Arduino
Pin 6
967    PWM->PWM_CH_NUM[7].PWM_CMR= PWM_CMR_CPRE_MCK_DIV_128;
968    PWM->PWM_CH_NUM[7].PWM_CPRD=255;
969    PWM->PWM_CH_NUM[7].PWM_CDTY=0;
970    PWM->PWM_CH_NUM[7].PWM_CMR|=PWM_CMR_CALG;
971    PWM->PWM_ENA |= PWM_ENA_CHID7;
972
973 }
974 /*
975
976 */
977 void Motores_Pines_config(){
978    //Habilita los PInes como GPIO
979    /*
980    MOTOR 1 -> PINES D0 y A15 (25,24)
981    MOTOR 2-> PINES D2 y D1 (27,26)
982    MOTOR 3 -> PINES D6 y D3 (29,28)
983    MOTOR 4 -> PINES A7 y D9 (31,30)
984    MOTOR 5 -> PINES C1 y D10 (33,32)
985    */
986    PIOA->PIO_PER|=PIO_PA7|PIO_PA15;
987    PIOC->PIO_PER|=PIO_PC1;
988    PIOD-
>PIO_PER|=PIO_PD0|PIO_PD1|PIO_PD2|PIO_PD3|PIO_PD6|PIO_PD9|PIO_PD10; //PINES
989    //se declara como salidas
990    PIOA->PIO_OER|=PIO_PA7|PIO_PA15;
991    PIOC->PIO_OER|=PIO_PC1;
992    PIOD-
>PIO_OER|=PIO_PD0|PIO_PD1|PIO_PD2|PIO_PD3|PIO_PD6|PIO_PD9|PIO_PD10; //PINES
993    //Se deshabilita las resistencias Pull-up internas
994    PIOA->PIO_PUDR|=PIO_PA7|PIO_PA15;
995    PIOC->PIO_PUDR|=PIO_PC1;
996    PIOD-
>PIO_PUDR|=PIO_PD0|PIO_PD1|PIO_PD2|PIO_PD3|PIO_PD6|PIO_PD9|PIO_PD10;
997 }
998 /*
999
1000*/
1001void Implementar_PID(PID *Pntr, float32_t medida){
1002   int _PWM;
1003   Pntr->_Medida=medida;
1004   #ifdef FSR_MODO
1005   Pntr->_error[0]=Pntr->_Setpoint-Pntr->_Medida;//error
1006   Pntr->_U=Pntr->_u_ant+Pntr->_k2*Pntr->_error[0]+Pntr->_k1*Pntr-
>_error[1]+Pntr->_k0*Pntr->_error[2];
1007   Pntr->_u_ant=Pntr->_U;
1008   Pntr->_error[2]=Pntr->_error[1];
1009   Pntr->_error[1]=Pntr->_error[0];
1010   #else
1011   Pntr->_Error=Pntr->_Setpoint-Pntr->_Medida;
1012   Pntr->_P=Pntr->_kp*Pntr->_Error;// parte proporcional
1013   Pntr->_I+=Pntr->_ki*(Pntr->_Error-Pntr->_Anti_windup)*Pntr-
```

```
>_Delta_t;//parte integral - bc Anti windup
1014  Pntr->_D=Pntr->_kd*(Pntr->_Error - Pntr->_Error_ant)/Pntr-
>_Delta_t;//parte derivatica
1015  Pntr->_Error_ant=Pntr->_Error;
1016
1017  /*salida_CONTROL*/
1018  Pntr->_U=Pntr->_P + Pntr->_I + Pntr->_D;
1019  Pntr->_U_sat=Pntr->_U;
1020
1021  #endif
1022  if (Pntr->_U<0) {
1023    _PWM=Pntr->_U*(-1);
1024    if (_PWM>Pntr->_PWM_MAX) {
1025      _PWM=Pntr->_PWM_MAX;
1026      } else if (_PWM<Pntr->_PWM_MIN) {
1027      _PWM=Pntr->_PWM_MIN;
1028    }
1029    Bajar_mot(Pntr,_PWM);
1030    Pntr->_U_sat=(float)_PWM*(-1);
1031    } else if (Pntr->_U>0) {
1032    _PWM=Pntr->_U;
1033    if (_PWM>Pntr->_PWM_MAX) {
1034      _PWM=Pntr->_PWM_MAX;
1035      } else if(_PWM<Pntr->_PWM_MIN) {
1036      _PWM=Pntr->_PWM_MIN;
1037    }
1038    Subir_mot(Pntr,_PWM);
1039    Pntr->_U_sat=(float)_PWM;
1040  }
1041  /*
1042  Antireset Windup back calculation
1043  */
1044  Pntr->_Error_sat=Pntr->_U-Pntr->_U_sat;
1045  Pntr->_Anti_windup=Pntr->_Error_sat*Pntr->_ka;
1046}
1047/*
1048
1049*/
1050void Subir_mot(PID *Pntr, unsigned char Valor){
1051  switch (Pntr->_ident){
1052    case PULGAR:
1053    PIOD->PIO_SODR|=PIO_PD0;
1054    PIOA->PIO_CODR|=PIO_PA15;
1055    PWM_PULGAR=Valor;
1056    break;
1057
1058    case INDICE:
1059    PIOD->PIO_SODR|=PIO_PD2;
1060    PIOD->PIO_CODR|=PIO_PD1;
1061    PWM_INDICE=Valor;
1062    break;
1063
1064    case MEDIO:
1065    PIOD->PIO_SODR|=PIO_PD6;
1066    PIOD->PIO_CODR|=PIO_PD3;
1067    PWM_MEDIO=Valor;
1068    break;
1069    case ANULAR:
1070    PIOA->PIO_CODR|=PIO_PA7;
1071    PIOD->PIO_SODR|=PIO_PD9;
1072    PWM_ANULAR=Valor;
```

```
1073      break;
1074      case MENHIQUE:
1075      PIOC->PIO_SODR|=PIO_PC1;
1076      PIOD->PIO_CODR|=PIO_PD10;
1077
1078      PWM_MENHIQUE=Valor;
1079      break;
1080    }
1081}
1082/*
1083
1084*/
1085void Bajar_mot(PID *Pntr, unsigned char Valor){
1086    switch (Pntr->_ident){
1087      case PULGAR:
1088        PIOD->PIO_CODR|=PIO_PD0;
1089        PIOA->PIO_SODR|=PIO_PA15;
1090        PWM_PULGAR=Valor;
1091      break;
1092      case INDICE:
1093        PIOD->PIO_CODR|=PIO_PD2;
1094        PIOD->PIO_SODR|=PIO_PD1;
1095        PWM_INDICE=Valor;
1096      break;
1097      case MEDIO:
1098        PIOD->PIO_CODR|=PIO_PD6;
1099        PIOD->PIO_SODR|=PIO_PD3;
1100        PWM_MEDIO=Valor;
1101      break;
1102      case ANULAR:
1103        PIOA->PIO_SODR|=PIO_PA7;
1104        PIOD->PIO_CODR|=PIO_PD9;
1105        PWM_ANULAR=Valor;
1106      break;
1107      case MENHIQUE:
1108        PIOC->PIO_CODR|=PIO_PC1;
1109        PIOD->PIO_SODR|=PIO_PD10;
1110        PWM_MENHIQUE=Valor;
1111      break;
1112    }
1113}
1114
1115void regresion(FSR_REG *Pntr,float32_t RAW){
1116    float32_t *w;
1117    w=&Pntr->w[0];
1118    Pntr-
>y=w[0]+w[1]*RAW+w[2]*pow_f32(RAW,2)+w[3]*pow_f32(RAW,3)+w[4]*pow_f32(RAW,4
);
1119}
1120
1121float32_t pow_f32(float32_t base, unsigned char expo){
1122    float32_t y=1;
1123    for (unsigned char i=0;i<expo;i++)
1124    {
1125      y=y*base;
1126    }
1127    return y;
1128}
1129
1130void derivada(FSR_DERIV *Pntr,float32_t RAW){
1131    float32_t deriv;
```

```
1132   Pntr->_x[0]=RAW;
1133   deriv=Pntr->_x[0]-Pntr->_x[1];
1134   if (deriv < 40 && deriv > (-40) && RAW>0)
1135   {
1136     Pntr->_y=RAW-deriv;
1137     Pntr->_x[1]=Pntr->_y;
1138   }else{
1139     Pntr->_y=RAW;
1140     Pntr->_x[1]=Pntr->_y;
1141   }
1142
1143}
```

Código para la obtención de FFT de la señal de corriente, creación del filtro MAF y de la estimación RMS.

```
%Programa para La obtencion de la FFT de la señal sensada, creacion del
%filtro MAF y de la Estimacion RMS
%
%Por: Jandry O. Banegas
%Fecha: 1/11/2017
clc
clear all
close all
%% Leer Data_RAW
Snal=load('DATA_NOFIR_MENIQ.mat');
figure()
plot(Snal.seal_meniq);
grid on
T=0.020;
Fs=1/T;
%% FT
FFT=fft(Snal.seal_meniq);
L=length(Snal.seal_meniq);
P2=abs(FFT./L);
P1=P2(1:L/2+1);
P1(2:end-1)=2*P1(2:end-1);
f=Fs*(0:L/2)/L;
figure()
plot(f,P1)
grid on
%% Creacion del filtro FIR MAF y RMS
maf=ones(5,1)/5;
figure()
freqz(maf)
figure()
Filtrada_Senal=conv(maf,Snal.seal_meniq);
plot(Snal.seal_meniq,'-g');
hold on
plot(Filtrada_Senal,'-k');
hold on
M=30;
for N=M:length(Filtrada_Senal)
    x1=Filtrada_Senal(N:-1:N-M+1);
    y_vent1(N)=rms_mio(x1);
end
plot(y_vent1,'-r')
hold on
M=25;
```

```
for N=M:length(Filtrada_Senal)
    x1=Filtrada_Senal(N:-1:N-M+1);
    y_vent2(N)=rms_mio(x1);
end
plot( y_vent2,'--','LineWidth',2)
[y_EWMA,w]=RMS_expo(Filtrada_Senal,0.98);
hold on
plot(y_EWMA,'-c','LineWidth',2)
grid on
%% FFT de la señal Filtrada y Estimada su RMS
FFT=fft(y_EWMA);
L=length(y_EWMA);
P2=abs(FFT/L);
P1=P2(1:L/2+1);
f=Fs*(0:L/2)/L;
P1(2:end-1)=2*P1(2:end-1);
figure()
plot(f,P1)
grid on;
```

Código desarrollado para la calibración de los sensores FSR.

```
//Autor: Santiago Sarmiento
//Fecha: 14-09-2017
//Lectura de sensores de presión para calibración


#include "HX711.h"

//Variables Balanza
#define DOUT  24
#define CLK   26
#define Vout  28
HX711 balanza(DOUT, CLK);

int Filtro1Balanza[2]= {0,0};
int Balanza = 0;

//Variables FSR
#define FSR_Pin A0
float FSR = 0;

long Rref = 10000;//Resistor de referencia
int Vref = 5;//Voltaje de referencia


int Filtro1FSR[2]= {0,0};
float Fout3 = 0;
float Fout2 = 0;
float Voltaje = 0;

//Variables Motor lineal
#define Motor_IN1 2
#define Motor_IN2 3
#define SensorMotor A3
#define VoltajeRefPOT 50

int aux = 0;
int potMotor = 0;
String dato = "";
```

```
int PWMM = 22;

//Funciones
float FiltroBalanza(int sensor);
float FiltroFSR(int sensor);
void Inicializacion();
float Leer_ADC(int val_ADC, long Rref);
void MotorBajar(int PWMM1);
void MotorSubir(int PWMM1);
void MotorStop();
String lectura_array();

String  teclado = "";

void setup() {
  Serial.begin(9600);
  Serial.print("Lectura del valor del ADC:  ");
  Inicializacion();

  //Serial.print("Lectura del valor del ADC:  ");
  Serial.println(balanza.read());
  Serial.println("No ponga ningun  objeto sobre la balanza");
  Serial.println("Destarando...");
  Serial.println("...");
  balanza.set_scale(197.30); // Establecemos la escala
  balanza.tare();  //El peso actual es considerado Tara.

  Serial.println("Empezar ------> I");
  //Serial.println("Bajar ------> B");
  //Serial.println("Subir ------> S");
//  while(1){
//    dato = lectura_array();
//    if(lectura_array()=="I")
//      break;
//    else if(lectura_array()=="B")
//      MotorBajar(100);
//    else if(lectura_array()=="S")
//      MotorSubir(100);
//  }

  MotorBajar(100);
  delay(30);

}

void loop() {

  //FSR  = Leer_ADC(FSR_Pin,Rref);
  Balanza = FiltroBalanza(balanza.get_units());
  FSR  = Leer_ADC(FSR_Pin,Rref);
  Serial.print(FSR,0);
  Serial.print("  ");
  Serial.println(Balanza);

  //if((FSR>748)&&(FSR<754)&&(aux==0)){
  //if((FSR>=398)&&(FSR<403)&&(aux==0)||(Balanza>4000)){
  if(Balanza > 4000){
    MotorStop();
    Serial.println("Datos Completos");
    delay(1000);
    while(analogRead(SensorMotor) < 900){
```

```
      MotorSubir(255);
    }
    MotorStop();
    aux = 1;
  }
  else if(aux==0){
    MotorBajar(PWMM);
    if(analogRead(SensorMotor)<5){
      aux = 1;
      MotorStop();
    }
    if((Leer_ADC(FSR_Pin,Rref) <= FSR)&&(Balanza > 50)){
      PWMM = PWMM + 1;
      if(PWMM >= 255)
        PWMM = 255;
    }
  }


}

//Inicio de funciones
void Inicializacion(){
  pinMode(FSR_Pin, INPUT);
  pinMode(Vout, OUTPUT);  digitalWrite(Vout, HIGH);
  pinMode(Motor_IN1, OUTPUT);
  pinMode(Motor_IN2, OUTPUT);
  pinMode(VoltajeRefPOT,OUTPUT);digitalWrite(VoltajeRefPOT, HIGH);

  while(analogRead(SensorMotor) < 900){
    MotorSubir(255);
  }
  MotorStop();
}

float FiltroBalanza(int sensor){
  float y1;
  Filtro1Balanza[0]= sensor;
  y1 = (Filtro1Balanza[0]+Filtro1Balanza[1])>>1;
  Filtro1Balanza[1] = y1;
  return y1;
}
float FiltroFSR(int sensor){
  float y1;
  Filtro1FSR[0]= sensor;
  y1 = (Filtro1FSR[0]+Filtro1FSR[1])>>1;
  Filtro1FSR[1] = y1;
  return y1;
}
float Leer_ADC(int val_ADC, long Rref){
  //Retorna un valor de 0 a 1023 en relacion al voltaje
  //Rref es el valor de la resistencia fija.
  float FSR = 0; int Vref = 5;//Voltaje de referencia
  FSR = analogRead(val_ADC);
  //FSR = Filtro1(FSR);
  if(FSR != 0){
    //-------------Lectura de voltaje Vin
    //Descomentar para dar un valor en voltaje de 0 a 5
    //FSR = 0.004882812 * FSR;

    //-------------Lectura de resistencia Rin
```

```
    //Descomentar para dar un valor en OHM del sensor
    //FSR = ((Rref*Vref)/FSR)-Rref;

    //-------------Salida identica que Vin con f?rmula de datasheet
    //Descomentar para salida de voltaje del circuito divisor.
    //FSR = (Rref*Vref)/(Rref+FSR);

    return FSR;
  }
}
//Funciones motor
void MotorBajar(int PWMM){
  analogWrite(Motor_IN1,PWMM);
  //digitalWrite (Motor_IN1, HIGH);
  digitalWrite (Motor_IN2, LOW);
}
void MotorSubir(int PWMM){
  digitalWrite (Motor_IN1, LOW);
  //digitalWrite (Motor_IN2, HIGH);
  analogWrite(Motor_IN2,PWMM);
}

void MotorStop(){
  digitalWrite (Motor_IN1, LOW);
  digitalWrite (Motor_IN2, LOW);
}

String lectura_array(){
  char inChar = 0;
  String datoin = "";
  while (Serial.available() > 0)
    delay(10);
  inChar = Serial.read();
  datoin += inChar;

  if (datoin.length() > 0)
  {
    return datoin;
  }
}
```

Código para el sistema FSR, desarrollado para optimizar la regresión polinomial sin limitaciones en la data.

```
%PROYECTO DE FIN DE TITULACIÓN
%Autor: Santiago Sarmiento
clear all
close all
clc
limite = 2; %x Newtons sobre la fuerza maxima medida
activador = 0;%1 para recortar la curva de calibracion
              %0 para mantener todos los valores
delta = 0.015;
%% Pulgar
%Lectura de la FUNCION DE TRANSFERENCIA medida
[TF_volt, NA] = textread('TF_ARM_Pulgar2.txt', '%f %u');
TF_tiempo = 0:delta:(length(TF_volt)-1)*delta;
TF_volt = TF_volt*(3.3/4096);%Conversion a voltaje
```

```
%Regresion desde puntos medidos para Ec empirica
%Lectura de datos de calibracion
[voltajeP, pesoP] = textread('2PulgarFSR_DomoAreaPequena.txt', '%f %u');
[voltajeP2, pesoP2] = textread('3PulgarFSR_DomoAreaGrande.txt', '%f %u');

pesoP    = pesoP*0.0098;%Newtons
voltajeP  = voltajeP*(5/1024);%Voltaje
pesoP2    = pesoP2*0.0098;%Newtons
voltajeP2 = voltajeP2*(5/1024);%Voltaje
if (activador==1)
    voltajeP2 = voltajeP2([1:length(find(voltajeP2<=max(TF_volt)+0.2))]);
    pesoP2 = pesoP2([1:length(voltajeP2)]);
    voltajeP = voltajeP([1:length(find(voltajeP<=max(TF_volt)+0.2))]);
    pesoP = pesoP([1:length(voltajeP)]);
end
%Limites

V = min(voltajeP):((max(voltajeP)-min(voltajeP))/(length(voltajeP)-
1)):max(voltajeP); V = V';
V2 = min(voltajeP2):((max(voltajeP2)-min(voltajeP2))/(length(voltajeP2)-
1)):max(voltajeP2); V2 = V2';

%Mejor criterio de regresion
for n=1:1:8
    EcP = polyfit(voltajeP2, pesoP2, n);
    F_Pulgar = polyval(EcP, voltajeP2);%Resultados de la funcion por
regresion
    R2_Pulgar(n) = round((corr(pesoP2,F_Pulgar))^2,4);%Redondeo 3
decimales
    MSE_Pulgar(n)=round(immse(pesoP2,F_Pulgar),4);%Redondeo 3 decimales
end
[~, MSEindex] = find(MSE_Pulgar == min(MSE_Pulgar),1,'first');
[~, R2index] = find(R2_Pulgar == max(R2_Pulgar),1,'first');%Se escoje
menor orden con R^2 max
EcP = polyfit(voltajeP2, pesoP2, R2index);%Construccion de la ecuacion
F_Pulgar = polyval(EcP, V2);%Resultados de la funcion por regresion

% %Lectura de la FUNCION DE TRANSFERENCIA medida
% [TF_volt, TF_tiempo] = textread('TF_ARM_Pulgar2.txt', '%f %u');

% TF_tiempo = 0:delta:(length(TF_volt)-1)*delta;
% TF_volt = TF_volt*(3.3/4096);%Conversion a voltaje

%TF de voltaje a Newtons
TF_Newton_Pulgar = polyval(EcP, TF_volt);
for i=1:1:length(TF_volt)
    if(TF_volt(i)<min(voltajeP2))
        TF_Newton_Pulgar(i)=0;
    end
end
%Salida Pantalla
disp('Criterio de calibración de la curva del FSR, R2 (Pearson^2)')
disp(' ')
disp('Parámetros dedo PULGAR')
disp(['Grado de regresión = ',num2str(R2index)])
disp(['R2 = ',num2str(R2_Pulgar(R2index))])
disp([['Ecuación parámetros: '],num2str(EcP)])
disp('------------------------------------------------------------')

%Graficar
```

```
figure(1)
%subplot(1,2,1)
hold on
title('FSR dedo Pulgar - Calibración');
xlabel('Voltaje [V]'); ylabel('Newtons [N]');
plot(V2,F_Pulgar,'m'); grid on
%plot(voltajeP,pesoP,'g');
plot(voltajeP2,pesoP2,'b');
%text(voltajeP2(60),max(TF_Newton_Pulgar),'\leftarrow Fueza máxima
aplicada');
plot([0 voltajeP(end)],[max(TF_Newton_Pulgar) max(TF_Newton_Pulgar)],'-
.r');
legend('Aproximación polinomial','Datos reales domo 5.4mm','Fuerza máxima
aplicada por el actuador','Location','northwest');
%legend('Aproximación polinomial','Datos reales domo 4.2mm','Datos reales
domo 5.4mm','Fuerza máxima aplicada por el actuador');
hold off

% subplot(1,2,2)
% hold on
% title('Función de Transferencia dedo Pulgar');
% xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
% plot(TF_tiempo,TF_Newton_Pulgar,'m'); grid on
% %xlim([0 2])
% %legend('Respuesta al impulso del dedo Pulgar');
hold off

%%
%% Índice
%Regresion desde puntos medidos para Ec empirica
%Lectura de datos
[voltajeI, pesoI] = textread('2IndiceFSR_DomoAreaPequena.txt', '%f %u');
[voltajeI2, pesoI2] = textread('3IndiceFSR_DomoAreaGrande.txt', '%f %u');
pesoI2 = pesoI2*0.0098;%Newtons
voltajeI=voltajeI*(5/1024);%Voltaje
voltajeI2=voltajeI2*(5/1024);%Voltaje
pesoI=pesoI*0.0098;%Newtons
%EcI = polyfit(voltajeI, pesoI, 6);

%Eliminacion de la saturacion
for i=1:2:length(voltajeI2)
    if((voltajeI2(i)==max(voltajeI2))&(voltajeI2(i+2)==max(voltajeI2)))
        voltajeI2 = voltajeI2(1:i-2);
        pesoI2 = pesoI2(1:i-2);
        break;
    end
end

V = min(voltajeI):((max(voltajeI)-min(voltajeI))/(length(voltajeI)-
1)):max(voltajeI);
V = V';
V2 = min(voltajeI2):((max(voltajeI2)-min(voltajeI2))/(length(voltajeI2)-
1)):max(voltajeI2);
V2 = V2';
%Mejor criterio de regresion
for n=1:1:7
    EcI = polyfit(voltajeI2, pesoI2, n);
    F = polyval(EcI, voltajeI2);%Resultados de la funcion por regresion
    R2_Indice(n) = round((corr(pesoI2,F))^2,4);%Redondeo 3 decimales
    MSE_Indice(n)=round(immse(pesoI2,F),4);%Redondeo 3 decimales
end
```

```
%R2index=0
[~, MSEindex] = find(MSE_Indice == min(MSE_Indice),1,'first');
[~, R2index] = find(R2_Indice == max(R2_Indice),1,'first');

EcI = polyfit(voltajeI2, pesoI2, R2index);
F = polyval(EcI, V2);%Resultados de la funcion por regresion

%Lectura de la FUNCION DE TRANSFERENCIA medida
[TF_volt, NA] = textread('TF_ARM_Indice.txt', '%f %u');
TF_tiempoI = 0:delta:(length(TF_volt)-1)*delta;
TF_volt = TF_volt*(3.3/4096);
%TF de voltaje a Newtons
TF_Newton_Indice = polyval(EcI, TF_volt);
for i=1:1:length(TF_volt)
    if(TF_volt(i)<min(voltajeI2))
        TF_Newton_Indice(i)=0;
    end
end
%Salida Pantalla
disp('Parámetros dedo INDICE')
disp(['Grado de regresión = ',num2str(R2index)])
disp(['R2 = ',num2str(R2_Indice(R2index))])
disp([['Ecuación parámetros: '],num2str(EcI)])
disp('------------------------------------------------------------')

%Graficar
figure(2)
%subplot(1,2,1)
hold on
title('FSR dedo Índice - Calibración');
xlabel('Voltaje [V]'); ylabel('Newtons [N]');
plot(V2,F,'m'); grid on
%plot(voltajeI,pesoI,'g');
plot(voltajeI2,pesoI2,'b');
plot([0 voltajeI2(end)],[max(TF_Newton_Indice) max(TF_Newton_Indice)],'-
.r');
legend('Aproximación polinomial','Datos reales domo 5.4mm','Fuerza máxima
aplicada por el actuador','Location','northwest');
%legend('Aproximación polinomial','Datos reales domo 4.2mm','Datos reales
domo 5.4mm','Fuerza máxima aplicada por el actuador');
hold off

% subplot(1,2,2)
% hold on
% title('Función de Transferencia dedo Índice');
% xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
% plot(TF_tiempoI,TF_Newton_Indice,'m'); grid on
% %xlim([0 2])
% %legend('Respuesta al impulso del dedo Pulgar');
hold off
%% Medio
%Regresion desde puntos medidos para Ec empirica
%Lectura de datos
[voltajeM, pesoM] = textread('2MedioFSR_DomoAreaPequena.txt', '%f %u');
[voltajeM2, pesoM2] = textread('3MedioFSR_DomoAreaGrande.txt', '%f %u');
pesoM2=pesoM2*0.0098;%Newtons
voltajeM=voltajeM*(5/1024);%Voltaje
voltajeM2=voltajeM2*(5/1024);%Voltaje
pesoM=pesoM*0.0098;%Newtons
%Eliminacion de la saturacion
for a=101:1:length(voltajeM)
```

```
    if((voltajeM(a)==max(voltajeM))&&(voltajeM(a+1)==max(voltajeM)))
        voltajeM = voltajeM(2:a);
        pesoM = pesoM(2:a);
        break;
    end
end
%Eliminacion de la saturacion
for i=1:1:length(voltajeM2)
    if((voltajeM2(i)==max(voltajeM2))&&(voltajeM2(i+1)==max(voltajeM2)))
        voltajeM2 = voltajeM2(1:i);
        pesoM2 = pesoM2(1:i);
        break;
    end
end

V = min(voltajeM):((max(voltajeM)-min(voltajeM))/(length(voltajeM)-
1)):max(voltajeM); V = V';
V2 = min(voltajeM2):((max(voltajeM2)-min(voltajeM2))/(length(voltajeM2)-
1)):max(voltajeM2); V2 = V2';

%Mejor criterio de regresion
for n=1:1:7
    EcM = polyfit(voltajeM2, pesoM2, n);
    F = polyval(EcM, voltajeM2);%Resultados de la funcion por regresion
    R2_Medio(n) = round((corr(pesoM2,F))^2,4);%Redondeo 3 decimales
    MSE_Medio(n)=round(immse(pesoM2,F),4);%Redondeo 3 decimales
end
%R2index=0
[~, MSEindex] = find(MSE_Medio == min(MSE_Medio),1,'first');
[~, R2index] = find(R2_Medio == max(R2_Medio),1,'first');

EcM = polyfit(voltajeM2, pesoM2, R2index);
F = polyval(EcM, V2);

%Lectura de la FUNCION DE TRANSFERENCIA medida
[TF_volt, NA] = textread('TF_ARM_Medio.txt', '%f %u');
TF_tiempoM = 0:delta:(length(TF_volt)-1)*delta;
TF_volt = TF_volt*(3.3/4096);
%TF de voltaje a Newtons
TF_Newton_Medio = polyval(EcM, TF_volt);
for i=1:1:length(TF_volt)
    if(TF_volt(i)<min(voltajeM2))
        TF_Newton_Medio(i)=0;
    end
end

%Salida Pantalla
disp('Parámetros dedo MEDIO')
disp(['Grado de regresión = ',num2str(R2index)])
disp(['R2 = ',num2str(R2_Medio(R2index))])
disp([['Ecuación parámetros: '],num2str(EcM)])
disp('------------------------------------------------------------')

%Graficar
figure(3)
%subplot(1,2,1)
hold on
title('FSR dedo Medio - Calibración');
xlabel('Voltaje [V]'); ylabel('Newtons [N]');
plot(V2,F,'m'); grid on
%plot(voltajeM,pesoM,'g');
```

```
%ylim([0 40]);
plot(voltajeM2,pesoM2,'b');
plot([0 voltajeM2(end)],[max(TF_Newton_Medio) max(TF_Newton_Medio)],'-
.r');
legend('Aproximación polinomial','Datos reales domo 5.4mm','Fueza máxima
aplicada por el actuador','Location','northwest');
%legend('Aproximación polinomial','Datos reales domo 4.2mm','Datos reales
domo 5.4mm','Fueza máxima aplicada por el actuador');
hold off

% subplot(1,2,2)
% hold on
% title('Función de Transferencia dedo Medio');
% xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
% plot(TF_tiempoM,TF_Newton_Medio,'m'); grid on
% %xlim([0 2])
% %legend('Respuesta al impulso del dedo Pulgar');
hold off

%% CONCLUSION
% Cada sensor FSR tiene una funcion de transferencia diferente, la
informacion medida de cada sensor se ve
% afectada por factores mecanicos, construccion, y principalmente porque
cada sensor sigue una curva
% caracteristica distinta.
```

Código para determinar las ecuaciones (incluidas en el código principal) por regresión polinomial limitando la data de ingreso.

```
%PROYECTO DE FIN DE TITULACIÓN
%Autor: Santiago Sarmiento
%Regresion de la curva de calibracon hasta 2 Newtons mas del limite
%superior medido de cada FSR accionado ya en la protesis
%% SE NECESITA EJECUTAR ANTES Regresion_EC_FSR.m
close all
clc
limite = 2;
delta = 0.015;
aux = 1;
%% Pulgar
V2 = min(voltajeP2):((max(voltajeP2)-min(voltajeP2))/(length(voltajeP2)-
1)):max(voltajeP2); V2 = V2';

%Lectura de la FUNCION DE TRANSFERENCIA medida
[TF_volt, NA] = textread('TF_ARM_Pulgar2.txt', '%f %u');

TF_volt = TF_volt*(3.3/4096);%Conversion a voltaje

F2 = polyval(EcP,V2);%Funcion del script anterior

%TF_Newton_Pulgar = polyval(EcP, TF_volt);%Valor maximo alcanzado por el
FSR pulgar
%Nuevos limites para regresion
pesoP2    =
pesoP2([1:length(find(pesoP2<max(TF_Newton_Pulgar)+limite))]);
voltajeP2 = voltajeP2([1:length(pesoP2)]);
V2        = V2([1:length(find(V2<max(voltajeP2)))]);
F2        = F2([1:length(V2)]);
```

```
%Mejor criterio de regresion
for n=1:1:4
    EcP_Newton = polyfit(V2, F2, n);
    F_Pulgar = polyval(EcP_Newton, V2);%Resultados de la funcion por
regresion
    R2(n) = round((corr(F2,F_Pulgar))^2,3);%Redondeo 3 decimales
    MSE(n)=round(immse(F2,F_Pulgar),3);%Redondeo 3 decimales
end
[~, MSEindex] = find(MSE == min(MSE),1,'first');
[~, R2index] = find(R2 == max(R2),1,'first');%Se escoje menor orden con
R^2 max

EcP_Newton = polyfit(V2, F2, R2index);%Construccion de la ecuacion Volt a
Newton
EcP_Volt = polyfit(F2, V2, R2index);%Construccion de la ecuacion Newton a
Volt
F_Pulgar = polyval(EcP_Newton, V2);%Resultados de la funcion por
regresion
V_Pulgar = polyval(EcP_Volt,F2);
% V3 = 0:0.001:1.5;
% F_Pulgar = polyval(EcP2, V3);

%TF de voltaje a Newtons
TF_Newton_Pulgar2 = polyval(EcP_Newton, TF_volt);
for i=1:1:length(TF_volt)
    if(TF_volt(i)<min(voltajeP2))
        TF_Newton_Pulgar2(i)=0;
    end
end

%Graficar
figure(1)
%subplot(1,2,1)
hold on; grid on
title('FSR dedo Pulgar - Calibración');

if(aux==0)
    xlabel('Voltaje [V]'); ylabel('Newtons [N]');
    plot(V2,F_Pulgar,'m')%Nueva regresion de menor orden
    plot(V2, F2,'b');% Datos previos
    plot(voltajeP2,pesoP2,'k')% Datos medidos
    Maxima fuerza
    plot([0 voltajeP2(end)],[max(TF_Newton_Pulgar2)
max(TF_Newton_Pulgar2)],'-.r');
else
    %Grafica invertido ejes
    ylabel('Voltaje [V]'); xlabel('Newtons [N]');
    plot(F_Pulgar,V2,'m')%Nueva regresion de menor orden
    plot(F2,V2,'b');% Datos previos
    plot(pesoP2,voltajeP2,'k')% Datos medidos
    plot([max(TF_Newton_Pulgar2) max(TF_Newton_Pulgar2)],[0
voltajeP2(end)],'-.r');
    %ylim([0 max(TF_Newton_Pulgar2)+limite]); xlim([0 max(V2)+0.1])
end

Curva_ARM = [['Aproximación de orden '],num2str(length(EcP_Newton)-1)];
Curva_Orig = [['Aproximación de orden '],num2str(length(EcP)-1)];
legend(Curva_ARM,Curva_Orig,'Datos reales','Fueza máxima
ejercida','Location','southeast');
```

```
% % subplot(1,2,2)
% % hold on
% % title('Función de Transferencia dedo Pulgar');
% % xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
% % plot(TF_tiempo,TF_Newton_Pulgar2,'m'); grid on
% % xlim([0 3])

%Salida Pantalla
X=[['Ecuación limitada a la fuerza máxima ejercida por el actuador más
'],num2str(limite),[' Newtons']];
disp('Criterio de calibración de la curva del FSR, R2 (Pearson^2)')
disp(X);disp(' ')
disp('Parámetros dedo PULGAR')
disp(['Grado de regresión = ',num2str(R2index)])
disp(['R2 = ',num2str(R2_Pulgar(R2index))])
disp([['Newton =  '],num2str(EcP_Newton)])
disp(['Volt = ',num2str(EcP_Volt)])
MSE_P = immse(TF_Newton_Pulgar,TF_Newton_Pulgar2);
disp(['MSE ',num2str(MSE_P)])
disp('-----------------------------------------------------------------')


%% Indice
V2 = min(voltajeI2):((max(voltajeI2)-min(voltajeI2))/(length(voltajeI2)-
1)):max(voltajeI2); V2 = V2';

%Lectura de la FUNCION DE TRANSFERENCIA medida
[TF_volt, NA] = textread('TF_ARM_Indice.txt', '%f %u');

TF_volt = TF_volt*(3.3/4096);%Conversion a voltaje

F2 = polyval(EcI,V2);%Funcion del script anterior

%TF_Newton_Indice = polyval(EcI, TF_volt);%Valor maximo alcanzado por el
FSR pulgar
%Nuevos limites para regresion
pesoI2    =
pesoI2([1:length(find(pesoI2<max(TF_Newton_Indice)+limite))]);
voltajeI2 = voltajeI2([1:length(pesoI2)]);
V2        = V2([1:length(find(V2<max(voltajeI2)))]);
F2        = F2([1:length(V2)]);

%Mejor criterio de regresion
for n=1:1:4
    EcI_Newton = polyfit(V2, F2, n);
    F_Indice = polyval(EcI_Newton, V2);%Resultados de la funcion por
regresion
    R2(n) = round((corr(F2,F_Indice))^2,3);%Redondeo 3 decimales
    MSE(n)=round(immse(F2,F_Indice),3);%Redondeo 3 decimales
end
[~, MSEindex] = find(MSE == min(MSE),1,'first');
[~, R2index] = find(R2 == max(R2),1,'first');%Se escoje menor orden con
R^2 max

EcI_Newton = polyfit(V2, F2, R2index);%Construccion de la ecuacion
F_Indice = polyval(EcI_Newton, V2);%Resultados de la funcion por
regresion
EcI_Volt = polyfit(F2,V2,R2index);
V_Indice = polyval(EcI_Volt,F2);
```

```
%TF de voltaje a Newtons
TF_Newton_Indice2 = polyval(EcI_Newton, TF_volt);
for i=1:1:length(TF_volt)
    if(TF_volt(i)<min(voltajeI2))
        TF_Newton_Indice2(i)=0;
    end
end

%Graficar
figure(2)
% % subplot(1,2,1)
hold on; grid on
title('FSR dedo Índice - Calibración');
if(aux==0)
    xlabel('Voltaje [V]'); ylabel('Newtons [N]');
    plot(V2,F_Indice,'m')%Nueva regresion de menor orden
    plot(V2, F2,'b');% Datos previos
    plot(voltajeI2,pesoI2,'k')% Datos medidos
    ylim([0 max(TF_Newton_Indice2)+limite]); xlim([0 max(V2)+0.1])
    %Maxima fuerza
    plot([0 voltajeI2(end)],[max(TF_Newton_Indice2)
max(TF_Newton_Indice2)],'-.r');
else
    ylabel('Voltaje [V]'); xlabel('Newtons [N]');
    plot(F_Indice,V2,'m')%Nueva regresion de menor orden
    plot( F2,V2,'b');% Datos previos
    plot(pesoI2,voltajeI2,'k')% Datos medidos
    xlim([0 max(TF_Newton_Indice2)+limite]); ylim([0 max(V2)+0.1])
    %Maxima fuerza
    plot([max(TF_Newton_Indice2) max(TF_Newton_Indice2)],[0
voltajeI2(end)],'-.r');
end
Curva_ARM = [['Aproximación de orden '],num2str(length(EcI_Newton)-1)];
Curva_Orig = [['Aproximación de orden '],num2str(length(EcI)-1)];
legend(Curva_ARM,Curva_Orig,'Datos reales','Fueza máxima
ejercida','Location','southeast');

% % subplot(1,2,2)
% % title('Función de Transferencia dedo Índice');
% % xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
% % plot(TF_tiempoI,TF_Newton_Indice2,'m'); grid on
% % xlim([0 3])
% % %legend('Respuesta al impulso del dedo Pulgar');
% % hold off

%Salida Pantalla
disp('Parámetros dedo INDICE')
disp(['Grado de regresión = ',num2str(R2index)])
disp(['R2 = ',num2str(R2_Indice(R2index))])
disp([['Newton = '],num2str(EcI_Newton)])
disp(['Volt = ',num2str(EcI_Volt)])
MSE_I = immse(TF_Newton_Indice,TF_Newton_Indice2);
disp(['MSE ',num2str(MSE_I)])
disp('------------------------------------------------------------')


%% Medio
V2 = min(voltajeM2):((max(voltajeM2)-min(voltajeM2))/(length(voltajeM2)-
1)):max(voltajeM2); V2 = V2';

%Lectura de la FUNCION DE TRANSFERENCIA medida
```

```
[TF_volt, NA] = textread('TF_ARM_Medio.txt', '%f %u');

TF_volt = TF_volt*(3.3/4096);%Conversion a voltaje

F2 = polyval(EcM,V2);%Funcion del script anterior

%TF_Newton_Medio = polyval(EcM, TF_volt);%Valor maximo alcanzado por el
FSR pulgar
%Nuevos limites para regresion
pesoM2    = pesoM2([1:length(find(pesoM2<max(TF_Newton_Medio)+limite))]);
voltajeM2 = voltajeM2([1:length(pesoM2)]);
V2        = V2([1:length(find(V2<max(voltajeM2)))]);
F2        = F2([1:length(V2)]);

%Mejor criterio de regresion
for n=1:1:4
    EcM_Newton = polyfit(V2, F2, n);
    F_Medio = polyval(EcM_Newton, V2);%Resultados de la funcion por
regresion
    R2(n) = round((corr(F2,F_Medio))^2,3);%Redondeo 3 dEcMmales
    MSE(n)=round(immse(F2,F_Medio),3);%Redondeo 3 dEcMmales
end
[~, MSEindex] = find(MSE == min(MSE),1,'first');
[~, R2index] = find(R2 == max(R2),1,'first');%Se escoje menor orden con
R^2 max

EcM_Newton = polyfit(V2, F2, R2index);%Construccion de la ecuacion
F_Medio = polyval(EcM_Newton, V2);%Resultados de la funcion por regresion

EcM_Volt = polyfit(F2,V2,R2index);
V_Medio = polyval(EcM_Volt,F2);

%TF de voltaje a Newtons
TF_Newton_Medio2 = polyval(EcM_Newton, TF_volt);
for i=1:1:length(TF_volt)
    if(TF_volt(i)<min(voltajeM2))
        TF_Newton_Medio2(i)=0;
    end
end

%Graficar
figure(3)
% % subplot(1,2,1)
hold on; grid on
title('FSR dedo Medio - Calibración');
if(aux==0)
    xlabel('Voltaje [V]'); ylabel('Newtons [N]');
    plot(V2,F_Medio,'m')%Nueva regresion de menor orden
    plot(V2, F2,'b');% Datos previos
    plot(voltajeM2,pesoM2,'k')% Datos medidos
    ylim([0 max(TF_Newton_Medio2)+limite]); xlim([0 max(V2)+0.1])
    %Maxima fuerza
    plot([0 voltajeM2(end)],[max(TF_Newton_Medio2)
max(TF_Newton_Medio2)],'-.r');
else
    ylabel('Voltaje [V]'); xlabel('Newtons [N]');
    plot(F_Medio,V2,'m')%Nueva regresion de menor orden
    plot(F2,V2,'b');% Datos previos
    plot(pesoM2,voltajeM2,'k')% Datos medidos
    xlim([0 max(TF_Newton_Medio2)+limite]); ylim([0 max(V2)+0.1])
    %Maxima fuerza
```

```
    plot([max(TF_Newton_Medio2) max(TF_Newton_Medio2)],[0
voltajeM2(end)],'-.r');
end
Curva_ARM = [['Aproximación de orden '],num2str(length(EcM_Newton)-1)];
Curva_Orig = [['Aproximación de orden '],num2str(length(EcM)-1)];
legend(Curva_ARM,Curva_Orig,'Datos reales','Fueza máxima
ejercida','Location','southeast');

% % subplot(1,2,2)
% % hold on
% % title('Función de Transferencia dedo Medio');
% % xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
% % plot(TF_tiempoM,TF_Newton_Medio2,'m'); grid on
% % xlim([0 3])
%legend('Respuesta al impulso del dedo Pulgar');
hold off

%Salida Pantalla
disp('Parámetros dedo MEDIO')
disp(['Grado de regresión = ',num2str(R2index)])
disp(['R2 = ',num2str(R2_Medio(R2index))])
disp([['Newton ='],num2str(EcM_Newton)])
disp(['Volt = ',num2str(EcM_Volt)])
MSE_M = immse(TF_Newton_Medio,TF_Newton_Medio2);
disp(['MSE ',num2str(MSE_M)])
disp('------------------------------------------------------------')
%% Comparacion de funcion de transferencia

TF_plot_mayor = ['Conversión con ecuación de orden ',num2str(length(EcP)-
1)];
TF_plot_mayor2 = ['Conversión con ecuación de orden
',num2str(length(EcI)-1)];
TF_plot_mayor3 = ['Conversión con ecuación de orden
',num2str(length(EcM)-1)];
TF_plot_menor = ['Conversión con ecuación de orden
',num2str(length(EcM_Newton)-1)];
figure(4)
subplot(1,3,1)
hold on; grid on
title('FSR dedo Pulgar')
plot(TF_tiempo,TF_Newton_Pulgar,'b')
plot(TF_tiempo,TF_Newton_Pulgar2,'m')
xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
%ylim([0 11])
hold off
legend(TF_plot_mayor, TF_plot_menor,'Location','southeast')
subplot(1,3,2)
hold on; grid on
title('FSR dedo Indice')
plot(TF_tiempoI,TF_Newton_Indice,'b')
plot(TF_tiempoI,TF_Newton_Indice2,'m')
xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
%ylim([0 11])
hold off
legend(TF_plot_mayor2, TF_plot_menor,'Location','southeast')
subplot(1,3,3)
hold on; grid on
title('FSR dedo Medio')
plot(TF_tiempoM,TF_Newton_Medio,'b')
plot(TF_tiempoM,TF_Newton_Medio2,'m')
xlabel('Tiempo [segundos]'); ylabel('Newtons [N]');
```

```
%ylim([0 11])
hold off
legend(TF_plot_mayor3, TF_plot_menor,'Location','southeast')

t_plot = 0:0.015:3;
opt.StepAmplitude = 255;

figure(5)
FT_Pulgar=load('TF_dedo_Pulgar');
FT_Pulgar = FT_Pulgar.TF_Pulgar; %Planta Pulgar
hold on;
step(FT_Pulgar*255,3);
plot(TF_tiempo,TF_Newton_Pulgar2,'m');
title('Respuesta al Impulso')
legend('Aproximación', 'Datos reales','Location','southeast')
xlabel('Tiempo'); ylabel('Amplitud [Newtons]');
grid on

figure(6)
FT_Indice=load('TF_dedo_Indice');
FT_Indice = FT_Indice.TF_Indice; %Planta Pulgar
hold on;
step(FT_Indice*255,3);
plot(TF_tiempoI,TF_Newton_Indice2,'m');
title('Respuesta al Impulso')
legend('Aproximación', 'Datos reales','Location','southeast')
xlabel('Tiempo'); ylabel('Amplitud [Newtons]');
ylim([0 max(TF_Newton_Indice2)+0.1])
grid on

figure(7)
FT_Medio=load('TF_dedo_Medio');
FT_Medio = FT_Medio.TF_Medio; %Planta Pulgar
hold on;
step(FT_Medio*255,3);
plot(TF_tiempoM,TF_Newton_Medio2,'m');
title('Respuesta al Impulso')
legend('Aproximación', 'Datos reales','Location','southeast')
xlabel('Tiempo'); ylabel('Amplitud [Newtons]');
grid on
```

Código para la simulación y sintonización de los controladores basados en medición de corriente.

```
% Programa para la sintonizacion de los controladores basados en medicion
% de corriente, 6 metodos para cada uno.
%
% Por: Jandry O. Banegas
% Fecha: 01/11/2017
clc
clear all
close all
%% DATA_PULGAR
opt_step= stepDataOptions('StepAmplitude',255);
Data=load('OUT_Pulgar.mat');

Tmues=(1:334)*0.02;
ft=load('FT_planta_pulgar.mat');
disp('/*Funcion de Transferencia Dedo Pulgar*/')
```

```
P_FT=tf(ft.TF_Pulgar)
step(ft.TF_Pulgar,max(Tmues),opt_step)
hold on
plot(Tmues,Data.y,'-r')
hold off
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
legend('FT Pulgar','Data Real Pulgar')
grid on
%%
Ku=18.392;
Pulgar_kcritico=feedback(Ku*ft.TF_Pulgar,1);
figure()
step(Pulgar_kcritico,3)
grid on;
%% PI_PULGAR
% U=K(e(t)+1/Ti*Int(e(t)))
% Zigler-Nichols Close-loop
s=tf('s');
Ku=18.392;
Tu=2.95-2.82;
Kp=0.45*Ku;
Ki=(Kp/Tu/1.2)*1/s;
disp('/*Controlador PI-ZN Pulgar*/');
PI=Kp+Ki
Ys=feedback(PI*ft.TF_Pulgar,1);
figure()
step(Ys,stepDataOptions('StepAmplitude',107),'-b')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
% Tyreus-Luyben
Kp_TL=Ku/3.2;
Ki_TL=(Kp_TL/Tu/0.45)*1/s;
disp('/*Controlador PI-Tyreus-Luyben Pulgar*/');
PI_TL=Kp_TL+Ki_TL
Ys_TL=feedback(PI_TL*ft.TF_Pulgar,1);
figure()
step(Ys_TL,stepDataOptions('StepAmplitude',107),'-y')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
% Shinskey
Kp_SH=Ku/2;
Ki_SH=(Kp_SH/Tu/2.2)*1/s;
disp('/*Controlador PI-Shinskey Pulgar*/');
PI_SH=Kp_SH+Ki_SH
Ys_SH=feedback(PI_SH*ft.TF_Pulgar,1);
figure()
step(Ys_SH,stepDataOptions('StepAmplitude',107),'--g')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
% Shen-YU
Kp_SY=Ku/3;
Ki_SY=(Kp_SY/Tu/0.5)*1/s;
disp('/*Controlador PI-Shen-YU Pulgar*/');
PI_SY=Kp_SY+Ki_SY
Ys_SY=feedback(PI_SY*ft.TF_Pulgar,1);
figure()
step(Ys_SY,stepDataOptions('StepAmplitude',107),'-c')
```

```
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
% usando sintonia LAMBDA
gp=0.4008;
tau=0.135;
Td=0.03342;
taucl=1*tau;
Kp_Lambda=tau/(gp*(taucl+Td));
Ti_Lambda=tau;
Ki_Lambda=(Kp_Lambda/Ti_Lambda)*1/s;
disp('/*Controlador PI-Lambda Pulgar*/');
PI_Lambda=Kp_Lambda+Ki_Lambda
Ys_Lambda=feedback(PI_Lambda*ft.TF_Pulgar,1);
figure()
step(Ys_Lambda,stepDataOptions('StepAmplitude',107),'-k')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
% usando Cohen Coon
Kp_CC=(0.45/gp)*((tau/Td)+0.092);
Ti_CC=3.33*Td*((tau+0.092*Td)/(tau+2.22*Td));
PI_CC=Kp_CC+(Kp_CC/Ti_CC)*1/s;
Ys_CC=feedback(PI_CC*ft.TF_Pulgar,1);
figure()
step(Ys_CC,stepDataOptions('StepAmplitude',107))
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
% % usando PIDtune
% [Cs,info]=pidtune(ft.TF_Pulgar,'PI');
% disp('/*Controlador PI-PITune Pulgar*/');
% Cs
% Ys1=feedback(Cs*ft.TF_Pulgar,1);
% step(Ys1,stepDataOptions('StepAmplitude',107),'-r')
% % % Usando el metodo Manual para el tuneo
% % Kp2=2.24;
% % Ki2=19.02/s;
% % disp('/*Controlador PI-Manual Pulgar*/');
% % PI2=Kp2+Ki2
% % Ys2=feedback(PI2*ft.TF_Pulgar,1);
% % hold on
% % step(Ys2,stepDataOptions('StepAmplitude',107),'-g')
% ylabel('Corriente [mA]')
% xlabel('Tiempo [s]')
% grid on
%% DATA_INDICE
opt_step= stepDataOptions('StepAmplitude',255);
Data=load('OUT_Indice.mat');

Tmues=(1:334)*0.020;
ft=load('FT_planta_indice.mat');
figure()
step(ft.TF_Indice,max(Tmues),opt_step)
hold on
plot(Tmues,Data.y,'-r')
hold off
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
legend('FT Indice','Data Real Indice')
grid on
```

```
%% Kcritico INDICE
Ku=11.89375;
yx=feedback(Ku*ft.TF_Indice,1);
figure()
step(yx,3)
%% PI INDICE
% U=K(e(t)+1/Tis*Int(e(t)))
% Zigler-Nichols Close-loop
s=tf('s');
Ku=11.89375;
Tu=2.84-2.6;
Kp=0.45*Ku;
Ki=(Kp/Tu/1.2)*1/s;
PI_Zn_indi=Kp+Ki;
Ys_indice=feedback(PI_Zn_indi*ft.TF_Indice,1);
figure()
step(Ys_indice,stepDataOptions('StepAmplitude',107),2,'-b')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Tyreus-Luyben
Kp_TL=Ku/3.2;
Ki_TL=(Kp_TL/Tu/0.45)*1/s;
PI_TL_indice=Kp_TL+Ki_TL;
Ys_TL_indice=feedback(PI_TL_indice*ft.TF_Indice,1);
figure()
step(Ys_TL_indice,stepDataOptions('StepAmplitude',107),'-y')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shinskey
Kp_SH_indi=Ku/2;
Ki_SH_indi=(Kp_SH_indi/Tu/2.2)*1/s;
disp('/*Controlador PI-Shinskey Pulgar*/');
PI_SH_indi=Kp_SH_indi+Ki_SH_indi
Ys_SH_indi=feedback(PI_SH_indi*ft.TF_Indice,1);
figure()
step(Ys_SH_indi,stepDataOptions('StepAmplitude',107),'--g')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shen-YU
Kp_SY=Ku/3;
Ki_SY=(Kp_SY/Tu/0.5)*1/s;
PI_SY_indice=Kp_SY+Ki_SY;
Ys_SY_indice=feedback(PI_SY_indice*ft.TF_Indice,1);
figure()
step(Ys_SY_indice,stepDataOptions('StepAmplitude',107),'-c')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando sintonia LAMBDA
gp_indice=0.41534;
Td_indice=0.06754;
tau_indice=0.18;
taucl_indice=1*tau_indice;
Kp_Lambda_indi=tau_indice/(gp_indice*(taucl_indice+Td_indice));
Ti_Lambda_indi=tau_indice;
Ki_Lambda_indi=(Kp_Lambda_indi/Ti_Lambda_indi)*1/s;
PI_Lambda_indi=Kp_Lambda_indi+Ki_Lambda_indi;
Ys_Lambda_indi=feedback(PI_Lambda_indi*ft.TF_Indice,1);
figure()
step(Ys_Lambda_indi,stepDataOptions('StepAmplitude',107),'-k')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
```

```
% usando Cohen Coon
Kp_CC_indi=(0.45/gp_indice)*((tau_indice/Td_indice)+0.092);
Ti_CC_indi=3.33*Td_indice*((tau_indice+0.092*Td_indice)/(tau_indice+2.22*Td
_indice));
PI_CC_indi=Kp_CC_indi+(Kp_CC_indi/Ti_CC_indi)*1/s;
Ys_CC_indi=feedback(PI_CC_indi*ft.TF_Indice,1);
figure()
step(Ys_CC_indi,stepDataOptions('StepAmplitude',107))
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando PIDtune
[Cs_indice,info]=pidtune(ft.TF_Indice,'PI');
Ys_indice1=feedback(Cs_indice*ft.TF_Indice,1);
figure()
step(Ys_indice1,stepDataOptions('StepAmplitude',107),'-r')
% % Usando el metodo Manual para el tuneo
% Kp2=1.7881;
% Ki2=12.9438/s;
% PI2=Kp2+Ki2;
% Ys2=feedback(PI2*ft.TF_Indice,1);
% hold on
% step(Ys2,stepDataOptions('StepAmplitude',113.6),'-g')
hold off
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% legend('Z-N','Tyreus-Luyben','Shinskey','Shen-Yu','1*Lambda','Cohen-
Coon','PItune Matlab')
grid on
%% DATA Medio
opt_step= stepDataOptions('StepAmplitude',255);
Data=load('OUT_Medio.mat');

Tmues=(1:334)*0.020;
ft=load('FT_planta_medio.mat');
figure()
step(ft.TF_Medio,max(Tmues),opt_step)
hold on
plot(Tmues,Data.y,'-r')
hold off
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
legend('FT Medio','Data Real Medio')
grid on
%% Kcritico MEDIO
Ku=8.2655;
yx=feedback(Ku*ft.TF_Medio,1);
figure()
step(yx,3)
%% PI MEDIO
% Zigler-Nichols Close-loop
s=tf('s');
Ku=8.2655;
Tu=3.58-3.29;
Kp=0.45*Ku;
Ki=(Kp/Tu/1.2)*1/s;
PI_ZN_med=Kp+Ki;
Ys_med=feedback(PI_ZN_med*ft.TF_Medio,1);
figure()
step(Ys_med,stepDataOptions('StepAmplitude',107),2,'-b')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
```

```
% Tyreus-Luyben
Kp_TL=Ku/3.2;
Ki_TL=(Kp_TL/Tu/0.45)*1/s;
PI_TL_med=Kp_TL+Ki_TL;
Ys_TL_med=feedback(PI_TL_med*ft.TF_Medio,1);
figure()
step(Ys_TL_med,stepDataOptions('StepAmplitude',107),'-y')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shinskey
Kp_SH_medi=Ku/2;
Ki_SH_medi=(Kp_SH_medi/Tu/2.2)*1/s;
disp('/*Controlador PI-Shinskey Pulgar*/');
PI_SH_medi=Kp_SH_medi+Ki_SH_medi
Ys_SH_medi=feedback(PI_SH_medi*ft.TF_Medio,1);
figure()
step(Ys_SH_medi,stepDataOptions('StepAmplitude',107),'--g')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shen-YU
Kp_SY=Ku/3;
Ki_SY=(Kp_SY/Tu/0.5)*1/s;
PI_SY_med=Kp_SY+Ki_SY;
Ys_SY_med=feedback(PI_SY_med*ft.TF_Medio,1);
figure()
step(Ys_SY_med,stepDataOptions('StepAmplitude',107),'-c')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando sintonia LAMBDA
gp_med=0.50516;
Td_med=0.0799;
tau_med=0.171;
taucl_med=1.25*tau_med;
Kp_Lambda_med=tau_med/(gp_med*(taucl_med+Td_med));
Ti_Lambda_med=tau_med;
Ki_Lambda_med=(Kp_Lambda_med/Ti_Lambda_med)*1/s;
PI_Lambda_med=Kp_Lambda_med+Ki_Lambda_med;
Ys_Lambda_med=feedback(PI_Lambda_med*ft.TF_Medio,1);
figure()
step(Ys_Lambda_med,stepDataOptions('StepAmplitude',107),'-k')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando Cohen Coon
Kp_CC_med=(0.45/gp_med)*((tau_med/Td_med)+0.092);
Ti_CC_med=3.33*Td_med*((tau_med+0.092*Td_med)/(tau_med+2.22*Td_med));
PI_CC_med=Kp_CC_med+(Kp_CC_med/Ti_CC_med)*1/s;
Ys_CC_med=feedback(PI_CC_med*ft.TF_Medio,1);
figure()
step(Ys_CC_med,stepDataOptions('StepAmplitude',107))
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Tune Matlab
PI_matlab_med=pidtune(ft.TF_Medio,'PI');
Ys_matlab_med=feedback(PI_matlab_med*ft.TF_Medio,1);
figure()
step(Ys_matlab_med,stepDataOptions('StepAmplitude',107),'-r')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
%% DATA Anular
opt_step= stepDataOptions('StepAmplitude',255);
```

```
Data=load('OUT_Anular.mat');

Tmues=(1:386)*0.020;
ft=load('FT_planta_anular.mat');
figure()
step(ft.TF_Anular,max(Tmues),opt_step)
hold on
plot(Tmues,Data.y,'-r')
hold off
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
legend('FT Anular','Data Real Anular')
grid on
%% Kcritico ANULAR
Ku=17.297;
yx=feedback(Ku*ft.TF_Anular,1);
figure()
step(yx,3)
%% PI ANULAR
% Zigler-Nichols Close-loop
s=tf('s');
Ku=17.297;
Tu=2.6-2.47;
Kp=0.45*Ku;
Ki=(Kp/Tu/1.2)*1/s;
PI_ZN_anul=Kp+Ki;
Ys_anul=feedback(PI_ZN_anul*ft.TF_Anular,1);
figure()
step(Ys_anul,stepDataOptions('StepAmplitude',107),2,'-b')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Tyreus-Luyben
Kp_TL=Ku/3.2;
Ki_TL=(Kp_TL/Tu/0.45)*1/s;
PI_TL_anul=Kp_TL+Ki_TL;
Ys_TL_anul=feedback(PI_TL_anul*ft.TF_Anular,1);
figure()
step(Ys_TL_anul,stepDataOptions('StepAmplitude',107),'-y')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shinskey
Kp_SH_anul=Ku/2;
Ki_SH_anul=(Kp_SH_anul/Tu/2.2)*1/s;
disp('/*Controlador PI-Shinskey Pulgar*/');
PI_SH_anul=Kp_SH_anul+Ki_SH_anul
Ys_SH_anul=feedback(PI_SH_anul*ft.TF_Anular,1);
figure()
step(Ys_SH_anul,stepDataOptions('StepAmplitude',107),'--g')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shen-YU
Kp_SY=Ku/3;
Ki_SY=(Kp_SY/Tu/0.5)*1/s;
PI_SY_anul=Kp_SY+Ki_SY;
Ys_SY_anul=feedback(PI_SY_anul*ft.TF_Anular,1);
figure()
step(Ys_SY_anul,stepDataOptions('StepAmplitude',107),'-c')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando sintonia LAMBDA
gp_anul=0.54251;
```

```
Td_anul=0.0334;
tau_anul=0.19;
taucl_anul=0.5*tau_anul;
Kp_Lambda_anul=tau_anul/(gp_anul*(taucl_anul+Td_anul));
Ti_Lambda_anul=tau_anul;
Ki_Lambda_anul=(Kp_Lambda_anul/Ti_Lambda_anul)*1/s;
PI_Lambda_anul=Kp_Lambda_anul+Ki_Lambda_anul;
Ys_Lambda_anul=feedback(PI_Lambda_anul*ft.TF_Anular,1);
figure()
step(Ys_Lambda_anul,stepDataOptions('StepAmplitude',107),'- k')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando Cohen Coon
Kp_CC_anul=(0.45/gp_anul)*((tau_anul/Td_anul)+0.092);
Ti_CC_anul=3.33*Td_anul*((tau_anul+0.092*Td_anul)/(tau_anul+2.22*Td_anul));
PI_CC_anul=Kp_CC_anul+(Kp_CC_anul/Ti_CC_anul)*1/s;
Ys_CC_anul=feedback(PI_CC_anul*ft.TF_Anular,1);
figure()
step(Ys_CC_anul,stepDataOptions('StepAmplitude',107))
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Tune Matlab
PI_matlab_anul=pidtune(ft.TF_Anular,'PI');
Ys_matlab_anul=feedback(PI_matlab_anul*ft.TF_Anular,1);
figure()
step(Ys_matlab_anul,stepDataOptions('StepAmplitude',107),'-r')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
%% DATA Menique
opt_step= stepDataOptions('StepAmplitude',255);
Data=load('OUT_Menique.mat');

Tmues=(1:386)*0.020;
ft=load('FT_planta_menique.mat');
figure()
step(ft.TF_Meniq,max(Tmues),opt_step)
hold on
plot(Tmues,Data.y,'-r')
hold off
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
legend('FT Menique','Data Real Menique')
grid on
%% Kcritico MENIQUE
Ku=20.575;
yx=feedback(Ku*ft.TF_Meniq,1);
figure()
step(yx,3)
%% PI MENIQUE
% Zigler-Nichols Close-loop
s=tf('s');
Ku=20.575;
Tu=2.61-2.49;
Kp=0.45*Ku;
Ki=(Kp/Tu/1.2)*1/s;
PI_ZN_meniq=Kp+Ki;
Ys_meniq=feedback(PI_ZN_meniq*ft.TF_Meniq,1);
figure()
step(Ys_meniq,stepDataOptions('StepAmplitude',107),2,'-b')
ylabel('Corriente [mA]')
```

```
xlabel('Tiempo [s]')
% Tyreus-Luyben
Kp_TL=Ku/3.2;
Ki_TL=(Kp_TL/Tu/0.45)*1/s;
PI_TL_meniq=Kp_TL+Ki_TL;
Ys_TL_meniq=feedback(PI_TL_meniq*ft.TF_Meniq,1);
figure()
step(Ys_TL_meniq,stepDataOptions('StepAmplitude',107),'-y')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shinskey
Kp_SH_meniq=Ku/2;
Ki_SH_meniq=(Kp_SH_meniq/Tu/2.2)*1/s;
disp('/*Controlador PI-Shinskey Pulgar*/');
PI_SH_meniq=Kp_SH_meniq+Ki_SH_meniq
Ys_SH_meniq=feedback(PI_SH_meniq*ft.TF_Meniq,1);
figure()
step(Ys_SH_meniq,stepDataOptions('StepAmplitude',107),'--g')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Shen-YU
Kp_SY=Ku/3;
Ki_SY=(Kp_SY/Tu/0.5)*1/s;
PI_SY_meniq=Kp_SY+Ki_SY;
Ys_SY_meniq=feedback(PI_SY_meniq*ft.TF_Meniq,1);
figure()
step(Ys_SY_meniq,stepDataOptions('StepAmplitude',107),'-c')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando sintonia LAMBDA
gp_meniq=0.49157 ;
Td_meniq=0.03188;
tau_meniq=0.194;
taucl_meniq=0.7*tau_meniq;
Kp_Lambda_meniq=tau_meniq/(gp_meniq*(taucl_meniq+Td_meniq));
Ti_Lambda_meniq=tau_meniq;
Ki_Lambda_meniq=(Kp_Lambda_meniq/Ti_Lambda_meniq)*1/s;
PI_Lambda_meniq=Kp_Lambda_meniq+Ki_Lambda_meniq;
Ys_Lambda_meniq=feedback(PI_Lambda_meniq*ft.TF_Meniq,1);
figure()
step(Ys_Lambda_meniq,stepDataOptions('StepAmplitude',107),'-k')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% usando Cohen Coon
Kp_CC_meniq=(0.45/gp_meniq)*((tau_meniq/Td_meniq)+0.092);
Ti_CC_meniq=3.33*Td_meniq*((tau_meniq+0.092*Td_meniq)/(tau_meniq+2.22*Td_me
niq));
PI_CC_meniq=Kp_CC_meniq+(Kp_CC_meniq/Ti_CC_meniq)*1/s;
Ys_CC_meniq=feedback(PI_CC_meniq*ft.TF_Meniq,1);
figure()
step(Ys_CC_meniq,stepDataOptions('StepAmplitude',107))
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
% Tune Matlab
PI_matlab_meniq=pidtune(ft.TF_Meniq,'PI');
Ys_matlab_meniq=feedback(PI_matlab_meniq*ft.TF_Meniq,1);
figure()
step(Ys_matlab_meniq,stepDataOptions('StepAmplitude',107),'-r')
ylabel('Corriente [mA]')
xlabel('Tiempo [s]')
grid on
```

Código para la simulación y sintonización de los controladores del dedo pulgar, basados en medición de fuerza mediante sensores FSR.

```
%PROYECTO DE FIN DE TITULACIÓN
%Autor: Santiago Sarmiento
%Analisis de PID
clc
clear all
close all
s = tf('s');
escalon1 = 657;
escalon=1;
aux = 1;%1 para TF en mili Volt
        %0 para TF en Newton
%% Ziegler Nichols CL
%Kp = Kcr*0.5  | 0          | 0
%Kp = 0.45*Kcr | Ti=Tcr/1.2 | 0
%Kp = 0.6*Kcr  | Ti=Tcr/2   | Td=Tcr/8

%PID SOME overshoot
%Kp = 0.33*Kcr  | Ti=Tcr*0.5 | Td=Tcr*0.33

%PID NO overshoot
%Kp = 0.2*Kcr  | Ti=Tcr*0.3 | Td=Tcr*0.5
%% Tyreus-Luyben
%Kp = Kcr/3.2  | Ti=Tcr*2.2 | 0
%Kp = Kcr/2.2  | Ti=Tcr*2.2 | Td=Tcr/6.3
%% Shinskey
%Kp = Kcr/2    | 0          | 0
%Kp = Kcr/2    | Ti=Tcr/2.2 | 0
%Kp = Kcr/4    | Ti=Tcr/2   | Td=Tcr/8.3
%% Lambda
% lambda = 3*Tcr;
% ?y = A2-A1 %A es amplitud
% ?u = 255;
% T1  ->  T2 y sus amplitudes correspondientes
% Gp = ?y/?u
% Tao_p = 1.5(T2-T1)
% Tao_d = T2-Tao_p
% PID parametros
% P =(2*Tao_p+Tao_d)/(2*Gp*(lambda+Tao_d));
% Ti = Tao_p+(Tao_d/2);
% Td = (Tao_p*Tao_d)/(2*Tao_p+Tao_d);
%% Cohen-Coon
%Gp = 1;
%Tao = 0.051; %entre 1% y 63%
%T_d = 0.01272 ;
%PI
%Kp = (0.45/Gp)*((Tao/T_d)+0.092);
%Ti = (3.33*T_d)*((Tao+0.092*T_d)/(Tao+2.22*T_d));
%PID
%Kp = (0.67/Gp)*((Tao/T_d)+0.185);
%Ti = (2.5*T_d)*((Tao+0.185*T_d)/(Tao+0.611*T_d));
%Td = 0.37*T_d*(Tao/(Tao+0.185*T_d));
%% Shen-YU
% Kp_SY=Kcr/3;
% Ki_SY=(Kp_SY/Tcr/0.5)*(1/s);
```

```
% PI_SY=Kp_SY+Ki_SY;

%% Dedo Pulgar
disp('----------------------------DEDO PULGAR----------------------------')
if(aux==0)
    FT_Pulgar=load('TF_dedo_Pulgar');
    FT_Pulgar = FT_Pulgar.TF_Pulgar; %Planta Pulgar
    Kcr = 539.009;
    Tcr = 0.404-0.358;
    FT_Kcr=feedback(FT_Pulgar*Kcr,+1);
    step(FT_Kcr,1); grid on
else
%     FT_Pulgar=load('TF_dedo_Pulgar_Volt');
%     FT_Pulgar = FT_Pulgar.TF_Pulgar_Volt; %Planta Pulgar
%     Kcr = 1.3579;%original volt esta descomentar para redaccion
%     Kcr = 1.35521; volt escrita
%     Tcr = 0.258-0.19;
    FT_Pulgar=load('Pulgar_Nueva_FT');%nueva
    FT_Pulgar = FT_Pulgar.Pulgar_Nueva_FT; %nueva
    FT_Pulgar = exp(-0.0334*s)*((3.022*s+3.339)/(0.08625*s^2+1.172*s+1));
    Kcr=1.5786;
    Tcr = 0.781-0.666;

    FT_Kcr=feedback(FT_Pulgar*Kcr,+1);
    step(FT_Kcr,2); grid on
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ZN CL

%P
Kp = Kcr*0.5;
P_ZN = Kp;
Pulgar_P_ZN=feedback(FT_Pulgar*P_ZN,+1);
% disp('Ziegler Nichols CL   Ganancias          Kp         Ti          Td')
% X=[[ 'Controlador P                        '],num2str(Kp)];
% disp(X)
%PI
            Kp = 0.45*Kcr;
Ti=Tcr/1.2; Ki = (Kp/Ti)*(1/s);
Ki_show = Kp/Ti;
PI_ZN=Kp+Ki;
Pulgar_PI_ZN=feedback(FT_Pulgar*PI_ZN,+1);
[num,~] = tfdata(PI_ZN,'v');
disp(['Ziegler Nichols CL-------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
           Kp = 0.6*Kcr;
Ti=Tcr/2;  Ki = (Kp/Ti)*(1/s);
Td=Tcr/8;  Kd = Kp*Td*s;
Kd_show = Kp*Td;
PID_ZN = Kp+Ki+Kd;
Pulgar_PID_ZN=feedback(FT_Pulgar*PID_ZN,+1);
% X=[[ 'Controlador PID                      '],num2str(Kp),'
',num2str(Ti),'      ',num2str(Td)];
% disp(X)
%PID SOME overshoot
             Kp = 0.33*Kcr;
Ti=Tcr*0.5;  Ki = (Kp/Ti)*(1/s);
Td=Tcr*0.33; Kd = Kp*Td*s;
%Kd_show = Kp*Td;
```

```
PID_minOverShoot_ZN = Kp+Ki+Kd;
Pulgar_PID_minOverShoot_ZN=feedback(FT_Pulgar*PID_minOverShoot_ZN,+1);
%PID NO overshoot
             Kp = 0.2*Kcr;
Ti=Tcr*0.3;  Ki = (Kp/Ti)*(1/s);
Td=Tcr*0.5;  Kd = Kp*Td*s;
%Kd_show = Kp*Td;
PID_NoOverShoot_ZN = Kp+Ki+Kd;
Pulgar_PID_NoOverShoot_ZN=feedback(FT_Pulgar*PID_NoOverShoot_ZN,+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATLAB PID TUNE
%P_Matlab
P_M=pidtune(FT_Pulgar,'p');
Pulgar_P_Matlab=feedback(FT_Pulgar*P_M,+1);

%PI_Matlab
PI_M=pidtune(FT_Pulgar,'pi');
Pulgar_PI_Matlab=feedback(FT_Pulgar*PI_M,+1);
[num,~] = tfdata(PI_M,'v');
disp(['PID tune MATLAB----------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID_Matlab
PID_M=pidtune(FT_Pulgar,'pid');
Pulgar_PID_Matlab=feedback(FT_Pulgar*PID_M,+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Tyreus-Luyben
%PI
              Kp = Kcr/3.2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
PI_TL = Kp+Ki;
Pulgar_PI_TL=feedback(FT_Pulgar*PI_TL,+1);
[num,~] = tfdata(PI_TL,'v');
disp(['Tyreus-Luyben CL---------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
              Kp = Kcr/2.2;
Ti = Tcr*2.2; (Kp/Ti)*(1/s);
Td = Tcr/6.3; Kd = Kp*Td*s;
PID_TL = Kp+Ki+Kd;
Pulgar_PID_TL=feedback(FT_Pulgar*PID_TL,+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shinskey
%P
              Kp = Kcr/2;
P_S = Kp;
Pulgar_P_S=feedback(FT_Pulgar*P_S,+1);

%PI
              Kp = Kcr/2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
PI_S = Kp+Ki;
Pulgar_PI_S=feedback(FT_Pulgar*PI_S,+1);
[num,~] = tfdata(PI_S,'v');
disp(['Shinskey CL--------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
              Kp = Kcr/4;
```

```
Ti = Tcr*2;   Ki = (Kp/Ti)*(1/s);
Td = Tcr/8.3; Kd = Kp*Td*s;
PID_S = Kp+Ki+Kd;
Pulgar_PID_S=feedback(FT_Pulgar*PID_S,+1);
[num,~] = tfdata(PID_S,'v');
disp(['Shinskey CL--------------PID     Kp = ',num2str(num(1)),'      Ki =
',num2str(num(2)),'      Kd = ',num2str(num(3))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lambda
if(aux==0)%Newton
    Gp=0.01291;
    Tao_p = 0.0518;
    Tao_d = 0.01272;
    lambda = 0.7*(Tao_p);
else%Volt
%     Gp=2.575;
%     Tao_p = 0.0374;
%     Tao_d = 0.02085;
%     lambda = 0.8*(Tao_p);% para redaccion
%     Gp=2.6912;
%     Tao_p = 0.115;
%     Tao_d = 0.033165;
%     lambda = 1*(Tao_p);
    Gp=3.3389;
    Tao_p = 0.114;
    Tao_d = 0.033435;
    lambda = 0.4*(Tao_p);
end
                                        Kp
=(2*Tao_p+Tao_d)/(2*Gp*(lambda+Tao_d));
 Ti = Tao_p+(Tao_d/2);                  Ki = (Kp/Ti)*(1/s);
 Td = (Tao_p*Tao_d)/(2*Tao_p+Tao_d);    Kd = Kp*Td*s;
PI_L = Kp+Ki;
Pulgar_PI_L=feedback(FT_Pulgar*PI_L,+1);
% [num,~] = tfdata(PI_L,'v');
% disp(['Lambda-------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%Lambda 2

Kp = Tao_p/(Gp* (lambda + Tao_d));
Ti = Tao_p;
Ki = (Kp/Ti)*(1/s);
PI_L2 = Kp+Ki;
Pulgar_PI_L2=feedback(FT_Pulgar*PI_L2,+1);
[num,~] = tfdata(PI_L2,'v');
disp(['Lambda-------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

T=0.015;

K2 = Kp*(1+(T/Ti))
K1 = -Kp
% % %Lambda 3
% % lambda = 1*Tao_p;
% % Kp = ((2*lambda)+Tao_d)/(Gp*(lambda+Tao_d)^2);
% % Ti = (2*lambda)+Tao_d;
% % Ki = (Kp/Ti)*(1/s);
% % PI_L3 = Kp+Ki;
% % Pulgar_PI_L3=feedback(FT_Pulgar*PI_L3,+1);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cohen-Coon
if(aux==0)
    Gp = 0.01291;
    Tao = 0.0518; %entre 0.1% y 63%
    T_d = 0.01272 ;
else
%     Gp=2.575;
%     Tao = 0.0374;
%     T_d = 0.02085;
    Gp=3.3389;
    Tao = 0.114;
    T_d = 0.033435;

end
%P
    Kp = (0.515/Gp)*((Tao/T_d)+0.34);
Pulgar_P_CC=feedback(FT_Pulgar*Kp,+1);
%PI
    Kp = (0.45/Gp)*((Tao/T_d)+0.092);
Ti = (3.33*T_d)*((Tao+0.092*T_d)/(Tao+2.22*T_d));
    Ki = (Kp/Ti)*(1/s);
PI_CC = Kp+Ki;
Pulgar_PI_CC=feedback(FT_Pulgar*PI_CC,+1);
[num,~] = tfdata(PI_CC,'v');
disp(['Cohen-Coon---------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
    Kp = (0.67/Gp)*((Tao/T_d)+0.185);
Ti = (2.5*T_d)*((Tao+0.185*T_d)/(Tao+0.611*T_d));
Td = 0.37*T_d*(Tao/(Tao+0.185*T_d));
    Ki = (Kp/Ti)*(1/s);
    Kd = Kp*Td*s;
PID_CC = Kp+Ki+Kd;
Pulgar_PID_CC=feedback(FT_Pulgar*PID_CC,+1);
[num,~] = tfdata(PID_CC,'v');
disp(['Cohen-Coon---------------PID     Kp = ',num2str(num(1)),'      Ki =
',num2str(num(2)),'      Kd = ',num2str(num(3))])
%Cohen-Coon 2
Kp = (1/Gp)*(Tao/Td)*(0.9+0.083*(Td/Tao));
Ti = Td*((0.9+0.083*(Td/Tao))/(0.27+0.6*(Td/Tao)));
Ki = (Kp/Ti)*(1/s);
PI_CC2=Kp+Ki;
Pulgar_PI_CC2=feedback(FT_Pulgar*PI_CC2,+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shen-YU
Kp_SY=Kcr/3;
Ki_SY=(Kp_SY/Tcr/0.5)*(1/s);
PI_SY=Kp_SY+Ki_SY;
Pulgar_PI_SY=feedback(FT_Pulgar*PI_SY,+1);
[num,~] = tfdata(PI_SY,'v');
disp(['Shen-YU------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Graficar
figure(1)
step(FT_Kcr,1)
title('Oscilación permanente')
xlabel('Tiempo');ylabel('Amplitud')
```

```
grid on

figure(2)
step(Pulgar_PI_ZN*escalon)
% [Pulgar_P_ZN,tiempo] = step(Pulgar_P_ZN);
% [Pulgar_PI_ZN,~] = step(Pulgar_PI_ZN);
%step(Pulgar_P_ZN,Pulgar_PI_ZN,Pulgar_PID_ZN,Pulgar_PID_minOverShoot_ZN,Pul
gar_PID_NoOverShoot_ZN)
title('Ziegler Nichols C-L')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud (mV)')
xlim([0 0.6])
grid on

figure(3)
step(Pulgar_PI_TL*escalon)
title('Tyreus-Luyben C-L')
% legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud (mV)')
%xlim([0 1])
grid on

figure(4)
step(Pulgar_PI_Matlab*escalon)
title('Sintonización automática MATLAB')
% legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud (mV)')
xlim([0 0.6])
grid on

figure(5)
step(Pulgar_PI_S*escalon)
title('Shinskey Closed Loop')
% legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud (mV)')
xlim([0 0.6])
grid on

figure(6)
step(Pulgar_PI_L2*escalon)
title('Lambda')
% legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud')
ylim([0 1.2]);xlim([0 0.6])
grid on

figure(7)
step(Pulgar_PI_CC*escalon);
title('Cohen-Coon')
% legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud');grid on
xlim([0 0.6])

figure(8)
step(Pulgar_PI_SY*escalon);
title('Shen-Yu')
% legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud');grid on
xlim([0 0.6])
%% Mejores Controladores
% DEDO PULGAR
```

```
figure(13)
step(Pulgar_PI_ZN,Pulgar_PI_TL,Pulgar_PI_S,Pulgar_PI_L2,Pulgar_PI_CC,Pulgar
_PI_SY)
title('Controladores PID dedo Pulgar')
legend('Ziegler-Nichols PI','Tyreus-Luyben PI','Shinskey PI','Lambda
PI','Cohen-Coon PI','Shen-Yu PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud')
grid on


%% Ajuste Manual
%pidTuner(FT_Pulgar,PID_M)
```

Código para la simulación y sintonización de los controladores del dedo índice, basados en medición de fuerza mediande sensores FSR.

```
%PROYECTO DE FIN DE TITULACIÓN
%Autor: Santiago Sarmiento
%Analisis de PID dedo Indice
clc
clear all
close all
s = tf('s');
aux = 1;%1 para TF en mili Volt
        %0 para TF en Newton
%% Dedo Indice
disp('----------------------------DEDO INDICE----------------------------
')
% FT_Indice=load('TF_dedo_Indice');
% FT_Indice = FT_Indice.TF_Indice; %Planta Indice
if(aux==0)
    FT_Indice=load('TF_dedo_Indice');
    FT_Indice = FT_Indice.TF_Indice; %Planta Indice
    Kcr = 213.647;
    Tcr = 0.386-0.341;
    FT_Kcr=feedback(FT_Indice*Kcr,+1);
else
%     FT_Indice=load('TF_dedo_Indice_Volt');
%     FT_Indice = FT_Indice.TF_Indice_Volt; %Planta Indice
%     FT_Indice = exp(-
0.015*s)*((1.068*s+5.322)/(0.00497*s^2+0.2539*s+1));
    FT_Indice=load('Indice_Nueva2_FT');
    FT_Indice = FT_Indice.Indice_Nueva2_FT; %Planta Indice
    Kcr = 0.59679;
    %Kcr = 0.633045;
    Tcr = 0.605-0.527;
    FT_Kcr=feedback(FT_Indice*Kcr,+1);
    step(FT_Kcr,2); grid on
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ZN CL
%P
 Kp = Kcr*0.5;
P_ZN = Kp;
Indice_P_ZN=feedback(FT_Indice*P_ZN,+1);

%PI
```

```
            Kp = 0.45*Kcr;
Ti=Tcr/1.2; Ki = (Kp/Ti)*(1/s);
Ki_show = Kp/Ti;
PI_ZN=Kp+Ki;
Indice_PI_ZN=feedback(FT_Indice*PI_ZN,+1);
[num,~] = tfdata(PI_ZN,'v');
disp(['Ziegler Nichols CL-------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
           Kp = 0.6*Kcr;
Ti=Tcr/2;  Ki = (Kp/Ti)*(1/s);
Td=Tcr/8;  Kd = Kp*Td*s;
Kd_show = Kp*Td;
PID_ZN = Kp+Ki+Kd;
Indice_PID_ZN=feedback(FT_Indice*PID_ZN,+1);
% X=[[ 'Controlador PID                        '],num2str(Kp),'
',num2str(Ti),'      ',num2str(Td)];
% disp(X)
%PID SOME overshoot
             Kp = 0.33*Kcr;
Ti=Tcr*0.5;  Ki = (Kp/Ti)*(1/s);
Td=Tcr*0.33; Kd = Kp*Td*s;
%Kd_show = Kp*Td;
PID_minOverShoot_ZN = Kp+Ki+Kd;
Indice_PID_minOverShoot_ZN=feedback(FT_Indice*PID_minOverShoot_ZN,+1);
%PID NO overshoot
             Kp = 0.2*Kcr;
Ti=Tcr*0.3;  Ki = (Kp/Ti)*(1/s);
Td=Tcr*0.5;  Kd = Kp*Td*s;
%Kd_show = Kp*Td;
PID_NoOverShoot_ZN = Kp+Ki+Kd;
Indice_PID_NoOverShoot_ZN=feedback(FT_Indice*PID_NoOverShoot_ZN,+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATLAB PID TUNE
%P_Matlab
P_M=pidtune(FT_Indice,'p');
Indice_P_Matlab=feedback(FT_Indice*P_M,+1);

%PI_Matlab
PI_M=pidtune(FT_Indice,'pi');
Indice_PI_Matlab=feedback(FT_Indice*PI_M,+1);
[num,~] = tfdata(PI_M,'v');
disp(['PID tune MATLAB----------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2)),])

%PID_Matlab
PID_M=pidtune(FT_Indice,'pid');
Indice_PID_Matlab=feedback(FT_Indice*PID_M,+1);
[num,~] = tfdata(PID_M,'v');
disp(['PID tune MATLAB----------PID     Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2)),'     Kd = 0'])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Tyreus-Luyben
%PI
              Kp = Kcr/3.2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
PI_TL = Kp+Ki;
Indice_PI_TL=feedback(FT_Indice*PI_TL,+1);
[num,~] = tfdata(PI_TL,'v');
disp(['Tyreus-Luyben CL---------PI      Kp = ',num2str(num(1)),'     Ki =
```

```
',num2str(num(2))])
T=0.015;

K2 = Kp*(1+(T/Ti))
K1 = -Kp
%PID
                Kp = Kcr/2.2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
Td = Tcr/6.3; Kd = Kp*Td*s;
PID_TL = Kp+Ki+Kd;
Indice_PID_TL=feedback(FT_Indice*PID_TL,+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shinskey
%P
                Kp = Kcr/2;
P_S = Kp;
Indice_P_S=feedback(FT_Indice*P_S,+1);

%PI
                Kp = Kcr/2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
PI_S = Kp+Ki;
Indice_PI_S=feedback(FT_Indice*PI_S,+1);
[num,~] = tfdata(PI_S,'v');
disp(['Shinskey CL--------------PI      Kp = ',num2str(num(1)),'    Ki =
',num2str(num(2))])

%PID
                Kp = Kcr/4;
Ti = Tcr*2;    Ki = (Kp/Ti)*(1/s);
Td = Tcr/8.3; Kd = Kp*Td*s;
PID_S = Kp+Ki+Kd;
Indice_PID_S=feedback(FT_Indice*PID_S,+1);
[num,~] = tfdata(PID_S,'v');
disp(['Shinskey CL--------------PID     Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2)),'       Kd = ',num2str(num(3))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lambda
if(aux==0)%En newton
    Gp = 0.019314;
    Tao_p = (0.0939);
    Tao_d = 0.01488;
    lambda = 0.5*(Tao_p);%0.5
else%en Volt
%     Gp = 5.3217;
%     Tao_p = 0.0287;
%     Tao_d = 0.01497;
%     lambda = 0.7*(Tao_p);% para redaccion tesis
%     Gp = 4.2424;
%     Tao_p = 0.0566;
%     Tao_d = 0.012255;
%     lambda = 1*(Tao_p);%Nueva FT
    Gp = 4.0506;
    Tao_p = 0.0283;
    Tao_d = 0.02514;
    lambda = 1*(Tao_p);% Nueva 2 FT
end
```

Kp

```
=(2*Tao_p+Tao_d)/(2*Gp*(lambda+Tao_d));
 Ti = Tao_p+(Tao_d/2);                Ki = (Kp/Ti)*(1/s);
 Td = (Tao_p*Tao_d)/(2*Tao_p+Tao_d);  Kd = Kp*Td*s;
PI_L = Kp+Ki;
Indice_PI_L=feedback(FT_Indice*PI_L,+1);
% [num,~] = tfdata(PI_L,'v');
% disp(['Lambda-------------------PI      Kp = ',num2str(num(1)),'     Ki
= ',num2str(num(2))])

%Lambda 2

Kp = Tao_p/(Gp* (lambda + Tao_d));
Ti = Tao_p;
Ki = (Kp/Ti)*(1/s);
PI_L2 = Kp+Ki;
Indice_PI_L2=feedback(FT_Indice*PI_L2,+1);
[num,~] = tfdata(PI_L2,'v');
disp(['Lambda-------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

T=0.015;

K2 = Kp*(1+(T/Ti))
K1 = -Kp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cohen-Coon
if(aux==0)
    Gp = 0.019314;
    Tao = 0.0939; %entre 1% y 63%
    T_d = 0.01488;
else
%     Gp = 5.3217;
%     Tao = 0.0285;
%     T_d = 0.01497;
    Gp = 4.0506;
    Tao = 0.0283;
    T_d = 0.02514;
end
%P
    Kp = (0.515/Gp)*((Tao/T_d)+0.34);
Indice_P_CC=feedback(FT_Indice*Kp,+1);
%PI
    Kp = (0.45/Gp)*((Tao/T_d)+0.092);
Ti = (3.33*T_d)*((Tao+(0.092*T_d))/(Tao+(2.22*T_d)));
    Ki = (Kp/Ti)*(1/s);
PI_CC = Kp+Ki;
Indice_PI_CC=feedback(FT_Indice*PI_CC,+1);
[num,~] = tfdata(PI_CC,'v');
disp(['Cohen-Coon---------------PI      Kp = ',num2str(num(1)),'    Ki =
',num2str(num(2))])

%PID
    Kp = (0.67/Gp)*((Tao/T_d)+0.185);
Ti = (2.5*T_d)*((Tao+(0.185*T_d))/(Tao+(0.611*T_d)));
Td = (0.37*T_d)*(Tao/(Tao+(0.185*T_d)));
    Ki = (Kp/Ti)*(1/s);
    Kd = Kp*Td*s;
PID_CC = Kp+Ki+Kd;
Indice_PID_CC=feedback(FT_Indice*PID_CC,+1);
% [num,~] = tfdata(PID_CC,'v');
% disp(['Cohen-Coon---------------PI      Kp = ',num2str(num(1)),'
```

```
Ki = ',num2str(num(2)),'      Kd = ',num2str(num(3))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shen-YU
Kp_SY=Kcr/3;
Ki_SY=(Kp_SY/Tcr/0.5)*(1/s);
PI_SY=Kp_SY+Ki_SY;
Indice_PI_SY=feedback(FT_Indice*PI_SY,+1);
[num,~] = tfdata(PI_SY,'v');
disp(['Shen-YU------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Graficar
figure(1)
step(FT_Kcr,10)
title('Oscilación permanente')
xlabel('Tiempo');ylabel('Amplitud')
grid on

figure(2)
step(Indice_PI_ZN)
title('Ziegler Nichols C-L')
%legend('Controlador P','Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
xlim([0 0.6])
grid on

figure(3)
step(Indice_PI_TL)
title('Tyreus-Luyben C-L')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
xlim([0 1])
grid on

figure(4)
step(Indice_P_Matlab,Indice_PI_Matlab)
title('Sintonización automática MATLAB')
legend('Controlador P','Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud')
xlim([0 0.6])
grid on

figure(5)
step(Indice_PI_S)
title('Shinskey C-L')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
grid on

figure(6)
step(Indice_PI_L2)
title('Lambda')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
ylim([0 1.2])
grid on

figure(7)
step(Indice_PI_CC);
```

```
title('Cohen-Coon')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)');grid on

figure(8)
step(Indice_PI_SY);
title('Shen-YU')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)');grid on

%% Mejores Controladores
% DEDO INDICE
figure(13)
step(Indice_PI_TL,Indice_PI_S,Indice_PID_S,Indice_PI_L2)
title('Controladores PID dedo Indice')
legend('Tyreus-Luyben PI','Shinskey PI','Shinskey PID','Lambda
PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud')
grid on


%% Ajuste Manual
%pidTuner(FT_Indice,PID_M)
```

Código para la simulación y sintonización de los controladores del dedo medio, basados en medición de fuerza mediande sensores FSR.

```
%PROYECTO DE FIN DE TITULACIÓN
%Autor: Santiago Sarmiento
%Analisis de PID dedo Indice
clc
clear all
close all
s = tf('s');
aux = 1;%1 para TF en mili Volt
        %0 para TF en Newton
%% Dedo Medio
disp('----------------------------DEDO MEDIO---------------------------
')
if(aux==0)%Newton
    FT_Medio=load('TF_dedo_Medio');
    FT_Medio = FT_Medio.TF_Medio; %Planta Medio
    Kcr = 172.4633;
    Tcr = 0.484-0.422;
    FT_Kcr=feedback(FT_Medio*Kcr,+1);%Oscilacion permanente
else%Volt
%     FT_Medio=load('TF_dedo_Medio_Volt');
%     FT_Medio = FT_Medio.TF_Medio_Volt; %Planta Medio
%     FT_Medio = exp(-0.028*s)*((2.141*s+5.708)/(0.01629*s^2+0.446*s+1));
    FT_Medio=load('Medio_Nueva_FT');
    FT_Medio = FT_Medio.Medio_Nueva_FT; %Planta Medio
    Kcr = 6.41838;
    Tcr = 0.272-0.248;
    FT_Kcr=feedback(FT_Medio*Kcr,+1);%Oscilacion permanente
    step(FT_Kcr,2); grid on
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ZN CL
%P
Kp = Kcr*0.5;
```

```
P_ZN = Kp;
Medio_P_ZN=feedback(FT_Medio*P_ZN,+1);

%PI
            Kp = 0.45*Kcr;
Ti=Tcr/1.2; Ki = (Kp/Ti)*(1/s);
Ki_show = Kp/Ti;
PI_ZN=Kp+Ki;
Medio_PI_ZN=feedback(FT_Medio*PI_ZN,+1);
[num,~] = tfdata(PI_ZN,'v');
disp(['Ziegler Nichols CL-------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
           Kp = 0.6*Kcr;
Ti=Tcr/2;  Ki = (Kp/Ti)*(1/s);
Td=Tcr/8;  Kd = Kp*Td*s;
Kd_show = Kp*Td;
PID_ZN = Kp+Ki+Kd;
Medio_PID_ZN=feedback(FT_Medio*PID_ZN,+1);

%PID SOME overshoot
              Kp = 0.33*Kcr;
Ti=Tcr*0.5;  Ki = (Kp/Ti)*(1/s);
Td=Tcr*0.33; Kd = Kp*Td*s;
%Kd_show = Kp*Td;
PID_minOverShoot_ZN = Kp+Ki+Kd;
Medio_PID_minOverShoot_ZN=feedback(FT_Medio*PID_minOverShoot_ZN,+1);
%PID NO overshoot
              Kp = 0.2*Kcr;
Ti=Tcr*0.3;  Ki = (Kp/Ti)*(1/s);
Td=Tcr*0.5;  Kd = Kp*Td*s;
%Kd_show = Kp*Td;
PID_NoOverShoot_ZN = Kp+Ki+Kd;
Medio_PID_NoOverShoot_ZN=feedback(FT_Medio*PID_NoOverShoot_ZN,+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATLAB PID TUNE
%P_Matlab
P_M=pidtune(FT_Medio,'p');
Medio_P_Matlab=feedback(FT_Medio*P_M,+1);

%PI_Matlab
PI_M=pidtune(FT_Medio,'pi');
Medio_PI_Matlab=feedback(FT_Medio*PI_M,+1);
[num,~] = tfdata(PI_M,'v');
disp(['PID tune MATLAB----------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID_Matlab
PID_M=pidtune(FT_Medio,'pid');
Medio_PID_Matlab=feedback(FT_Medio*PID_M,+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Tyreus-Luyben
%PI
              Kp = Kcr/3.2;
Ti = Tcr*2.2; Ki =(Kp/Ti)*(1/s);
PI_TL = Kp+Ki;
Medio_PI_TL=feedback(FT_Medio*PI_TL,+1);
[num,~] = tfdata(PI_TL,'v');
disp(['Tyreus-Luyben CL---------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])
```

```
%PID
               Kp = Kcr/2.2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
Td = Tcr/6.3; Kd = Kp*Td*s;
PID_TL = Kp+Ki+Kd;
Medio_PID_TL=feedback(FT_Medio*PID_TL,+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shinskey
%P
               Kp = Kcr/2;
P_S = Kp;
Medio_P_S=feedback(FT_Medio*P_S,+1);

%PI
               Kp = Kcr/2;
Ti = Tcr*2.2; Ki = (Kp/Ti)*(1/s);
PI_S = Kp+Ki;
Medio_PI_S=feedback(FT_Medio*PI_S,+1);
[num,~] = tfdata(PI_S,'v');
disp(['Shinskey CL--------------PI      Kp = ',num2str(num(1)),'      Ki =
',num2str(num(2))])

%PID
               Kp = Kcr/4;
Ti = Tcr*2;   Ki = (Kp/Ti)*(1/s);
Td = Tcr/8.3; Kd = Kp*Td*s;
PID_S = Kp+Ki+Kd;
Medio_PID_S=feedback(FT_Medio*PID_S,+1);
[num,~] = tfdata(PID_S,'v');
disp(['Shinskey CL--------------PID     Kp = ',num2str(num(1)),'      Ki =
',num2str(num(2)),'      Kd = ',num2str(num(3))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lambda
if(aux==0)%Newton
    Gp = 0.038465;
    Tao_p = (0.0688);
    Tao_d = 0.017385;
    lambda = 0.4*(Tao_p);
else %Volts
%     Gp = 5.7077;
%     Tao_p = 0.0457;
%     Tao_d = 0.02799;
%     lambda = 0.7*(Tao_p);
%     Gp = 2.9498;
%     Tao_p = 0.106;
%     Tao_d = 0.016005;
%     lambda = 2*(Tao_p);
    Gp = 3.6565;
    Tao_p = 0.102;
    Tao_d = 0.006585;
    lambda = 0.5*(Tao_p);
end

                                        Kp
=(2*Tao_p+Tao_d)/(2*Gp*(lambda+Tao_d));
 Ti = Tao_p+(Tao_d/2);                  Ki = (Kp/Ti)*(1/s);
 Td = (Tao_p*Tao_d)/(2*Tao_p+Tao_d);    Kd = Kp*Td*s;
PI_L = Kp+Ki;
```

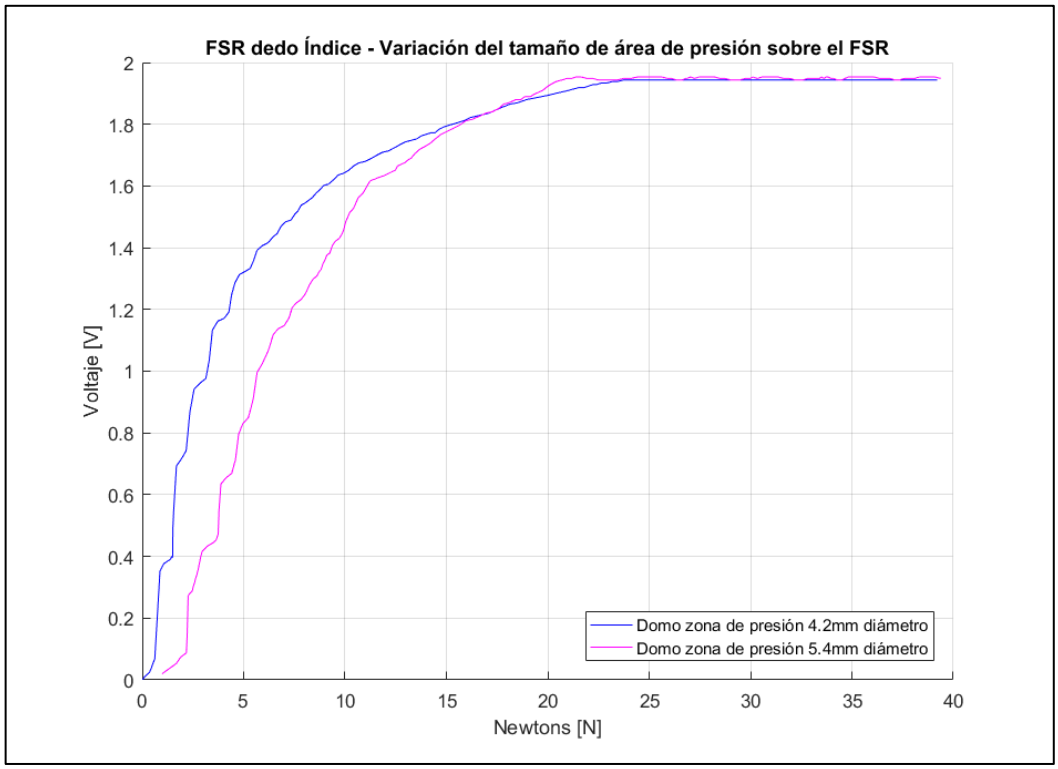
```
Medio_PI_L=feedback(FT_Medio*PI_L,+1);
% [num,~] = tfdata(PI_L,'v');
% disp(['Lambda-------------------PI      Kp = ',num2str(num(1)),'     Ki
= ',num2str(num(2))])

%Lambda 2

Kp = Tao_p/(Gp* (lambda + Tao_d));
Ti = Tao_p;
Ki = (Kp/Ti)*(1/s);
PI_L2 = Kp+Ki;
Medio_PI_L2=feedback(FT_Medio*PI_L2,+1);
[num,~] = tfdata(PI_L2,'v');
disp(['Lambda-------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

T=0.015;
K2 = Kp*(1+(T/Ti))
K1 = -Kp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cohen-Coon
if(aux==0)
    Gp = 0.038465;
    Tao = 0.0688; %entre 1% y 63%
    T_d = 0.017385;
else
%     Gp = 5.7077;
%     Tao = 0.0457; %entre 1% y 63%
%     T_d = 0.02799;
    Gp = 3.6565;
    Tao = 0.102; %entre 1% y 63%
    T_d = 0.006585;
end
%P
    Kp = (0.515/Gp)*((Tao/T_d)+0.34);
Medio_P_CC=feedback(FT_Medio*Kp,+1);
%PI
    Kp = (0.45/Gp)*((Tao/T_d)+0.092);
Ti = (3.33*T_d)*((Tao+0.092*T_d)/(Tao+2.22*T_d));
    Ki = (Kp/Ti)*(1/s);
PI_CC = Kp+Ki;
Medio_PI_CC=feedback(FT_Medio*PI_CC,+1);
[num,~] = tfdata(PI_CC,'v');
disp(['Cohen-Coon---------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%PID
    Kp = (0.67/Gp)*((Tao/T_d)+0.185);
Ti = (2.5*T_d)*((Tao+0.185*T_d)/(Tao+0.611*T_d));
Td = 0.37*T_d*(Tao/(Tao+0.185*T_d));
    Ki = (Kp/Ti)*(1/s);
    Kd = Kp*Td*s;
PID_CC = Kp+Ki+Kd;
Medio_PID_CC=feedback(FT_Medio*PID_CC,+1);
[num,~] = tfdata(PID_CC,'v');
disp(['Cohen-Coon---------------PID     Kp = ',num2str(num(1)),'      Ki
= ',num2str(num(2)),'      Kd = ',num2str(num(3))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Shen-YU
Kp_SY=Kcr/3;
```

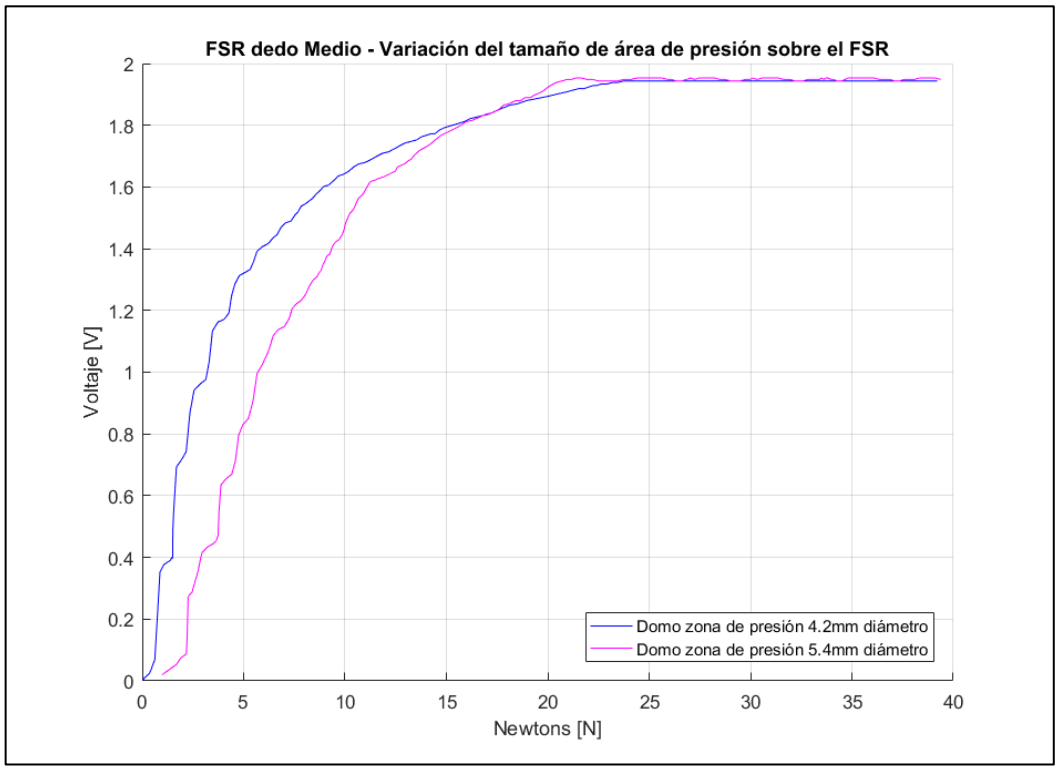
```
Ki_SY=(Kp_SY/Tcr/0.5)*(1/s);
PI_SY=Kp_SY+Ki_SY;
Medio_PI_SY=feedback(FT_Medio*PI_SY,+1);
[num,~] = tfdata(PI_SY,'v');
disp(['Shen-YU------------------PI      Kp = ',num2str(num(1)),'     Ki =
',num2str(num(2))])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Graficar
figure(1)
step(FT_Kcr,1)
title('Oscilación permanente')
xlabel('Tiempo');ylabel('Amplitud')
grid on

figure(2)
step(Medio_PI_ZN)
title('Ziegler Nichols C-L')
%legend('Controlador P','Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
xlim([0 0.6])
grid on

figure(3)
step(Medio_PI_TL)
title('Tyreus-Luyben C-L')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
xlim([0 1])
grid on

figure(4)
step(Medio_P_Matlab,Medio_PI_Matlab)
title('Sintonización automática MATLAB')
legend('Controlador P','Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
xlim([0 0.6])
grid on

figure(5)
step(Medio_PI_S)
title('Shinskey C-L')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
grid on

figure(6)
step(Medio_PI_L2)
title('Lambda')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)')
ylim([0 1.2])
grid on

figure(7)
step(Medio_PI_CC);
title('Cohen-Coon')
%legend('Controlador P','Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)');grid on

figure(8)
```

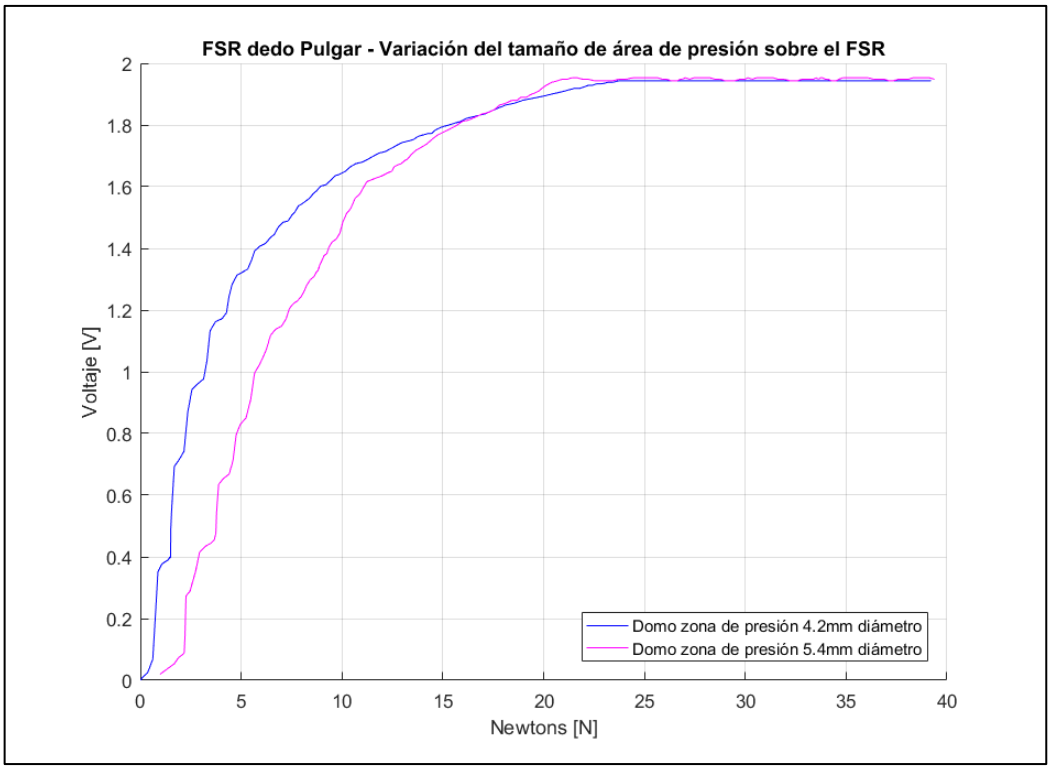
```
step(Medio_PI_SY);
title('Shen-YU')
%legend('Controlador PI','Location','southeast');
xlabel('Tiempo');ylabel('Voltaje (mV)');grid on

%% Mejores Controladores
% DEDO MEDIO
figure(13)
step(Medio_PI_TL,Medio_PID_S,Medio_PI_L2)
title('Controladores PID dedo Medio')
legend('Tyreus-Luyben CL PI','Shinskey CL PID','Lambda
PI','Location','southeast');
xlabel('Tiempo');ylabel('Amplitud')
grid on


%% Ajuste Manual
%pidTuner(FT_Medio,PID_M)
```

Anexo C

Comparación de la respuesta de los sensores FSR al cambio del área de presión del domo.

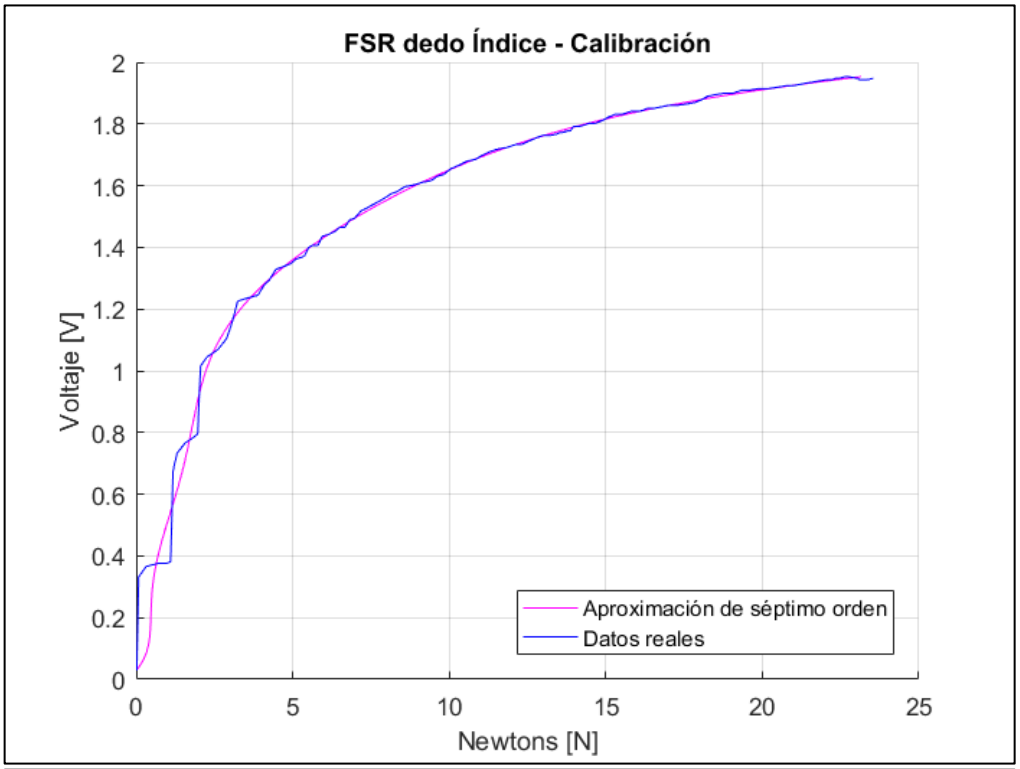


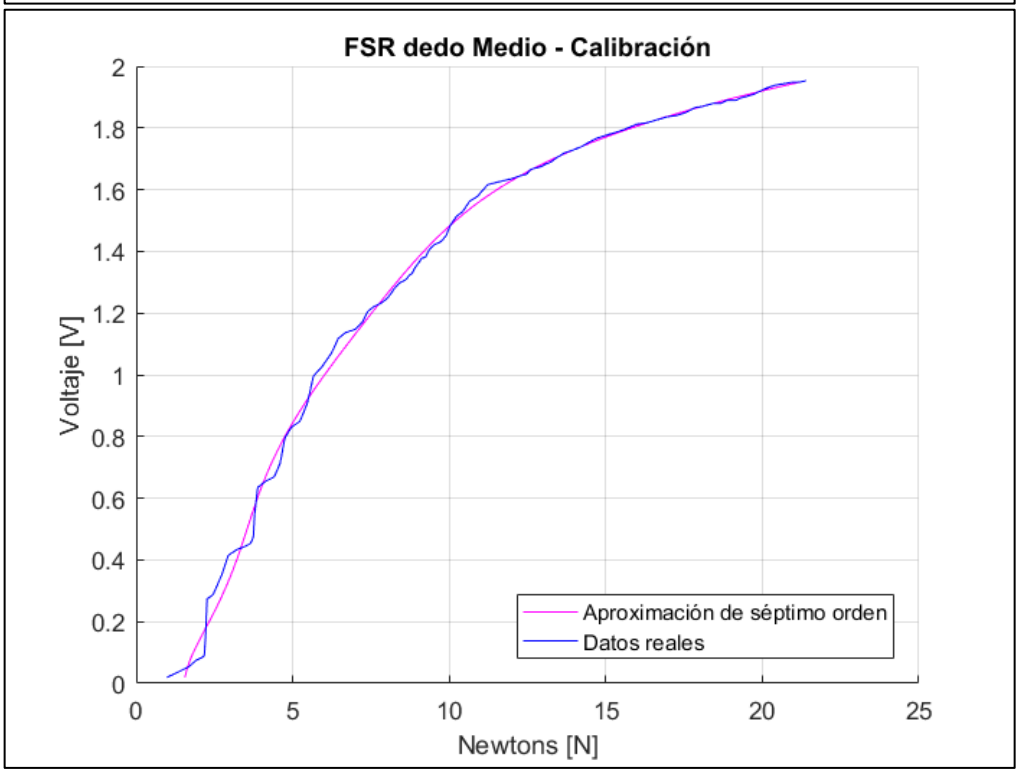


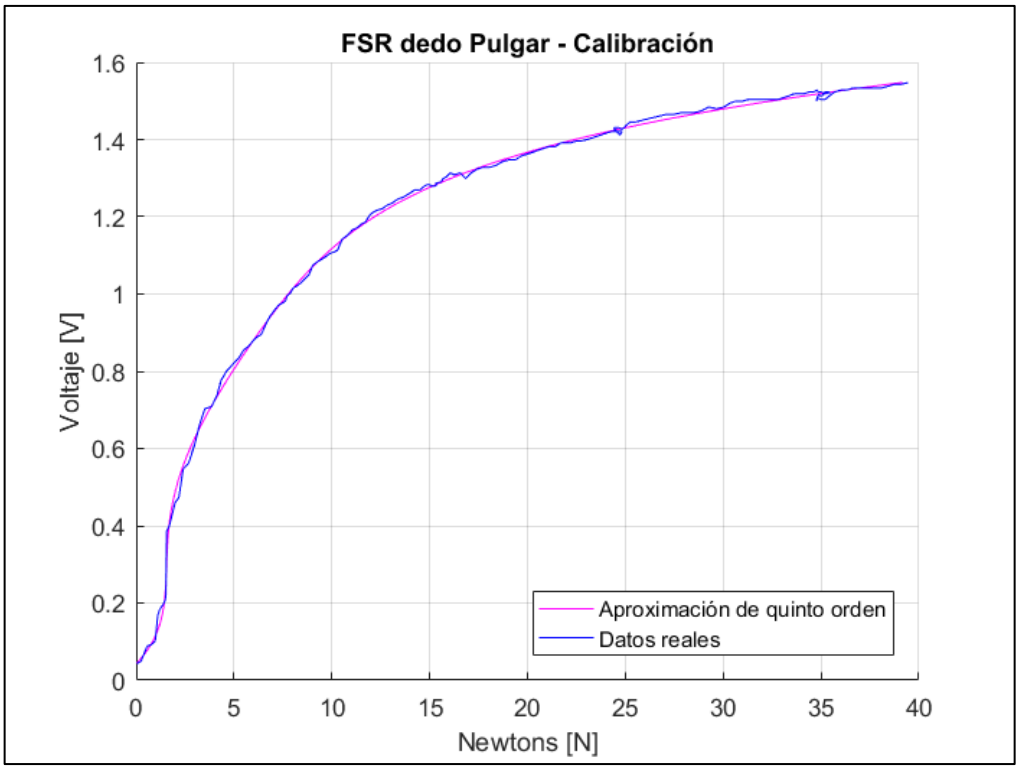

FSR dedo Pulgar - Variación del tamaño de área de presión sobre el FSR
Voltaje [V]
Newtons [N]
Domo zona de presión 4.2mm diámetro
Domo zona de presión 5.4mm diámetro
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
5
10
15
20
25
30
35
40

Anexo D

Calibración de los sensores FSR de la curva con los datos completos, eliminando la saturación en el sensor que lo requiera.

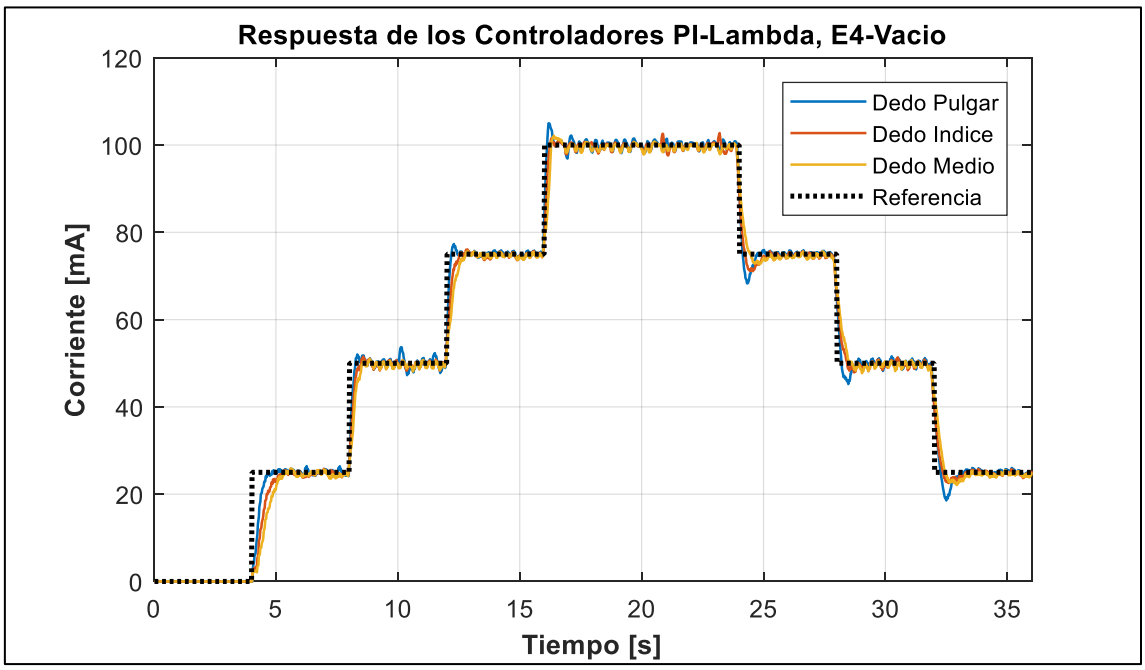


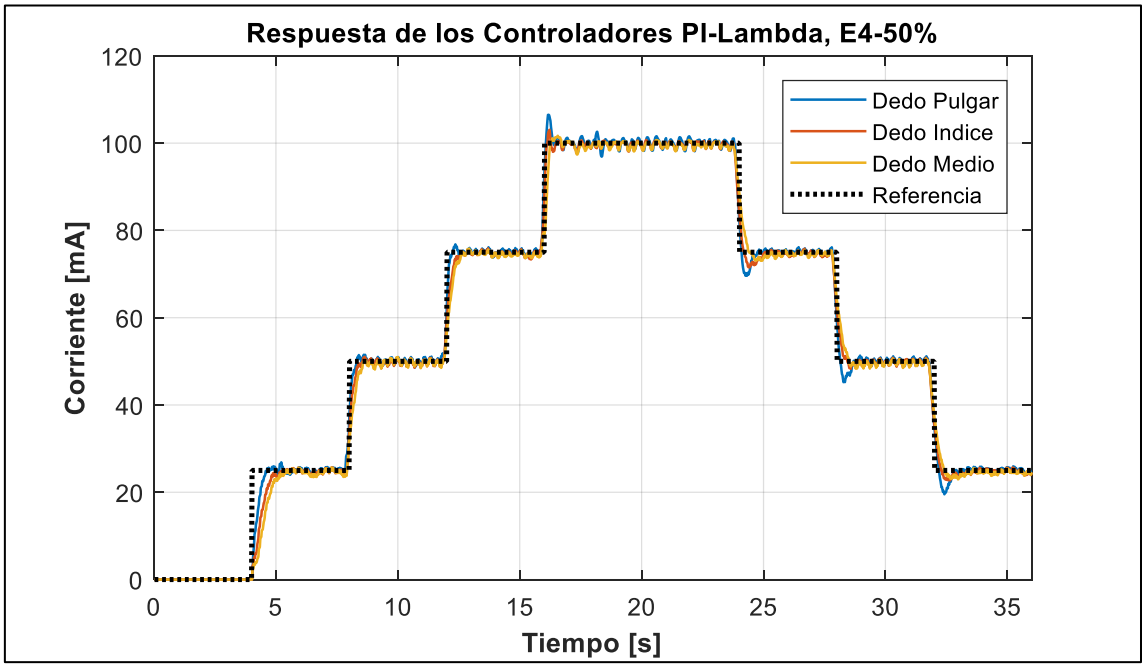


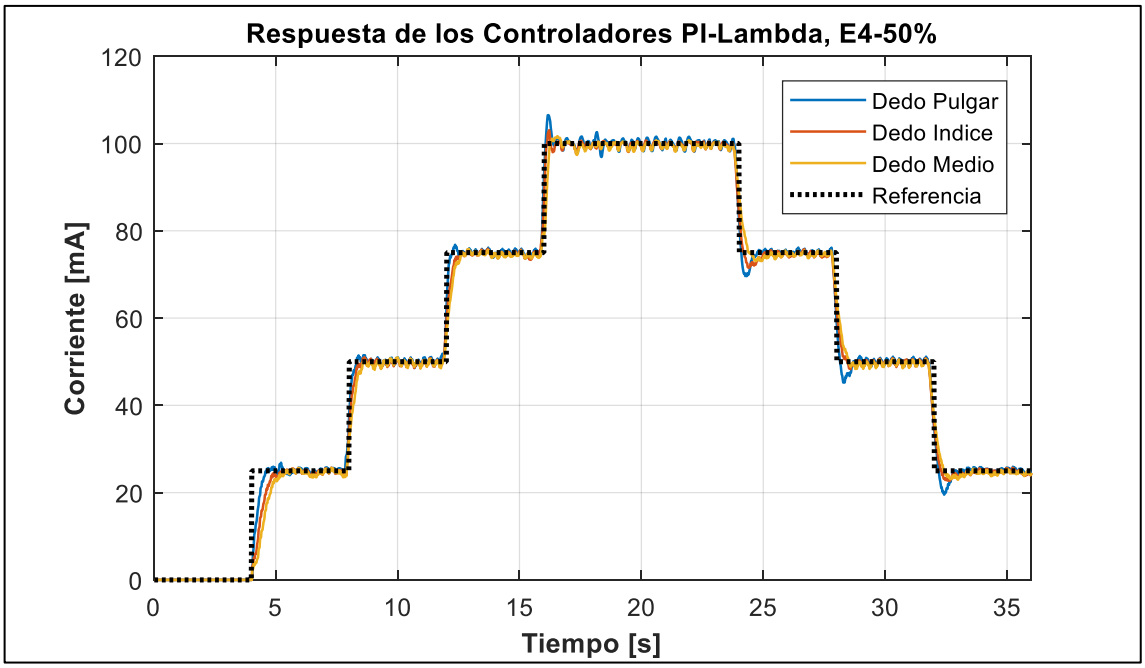

FSR dedo Pulgar - Calibración
Voltaje [V]
Newtons [N]
Aproximación de quinto orden
Datos reales
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
5
10
15
20
25
30
35
40

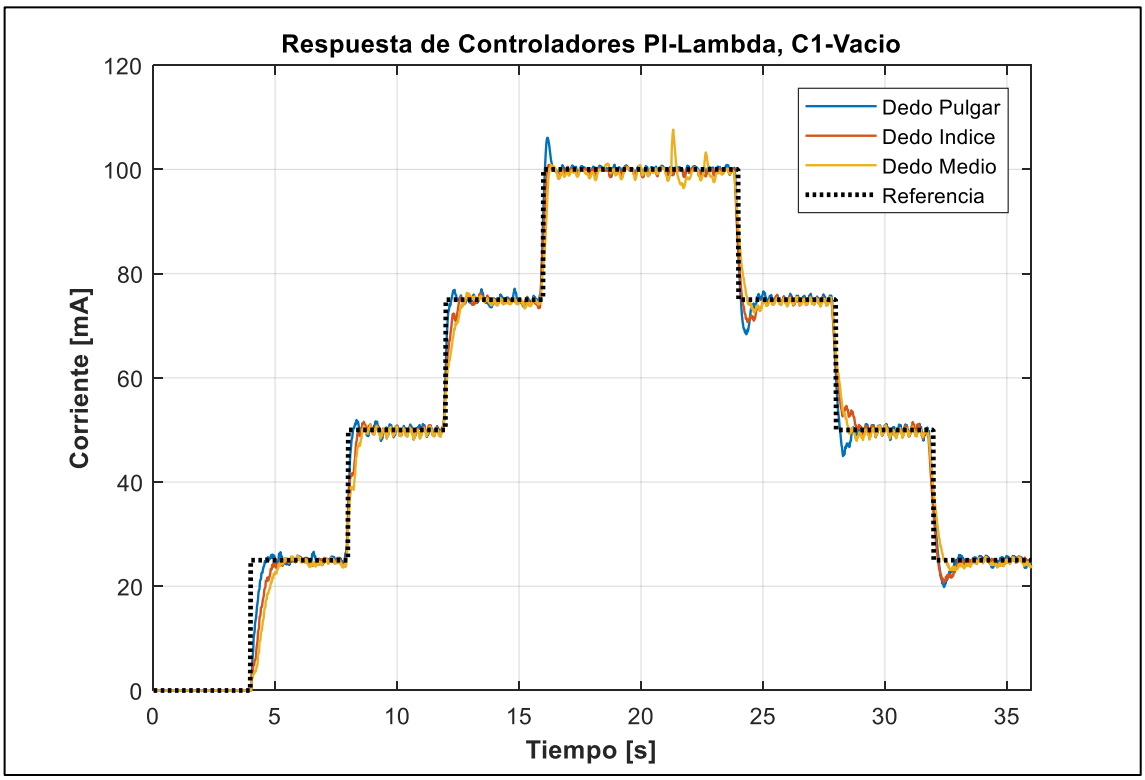
Anexo E

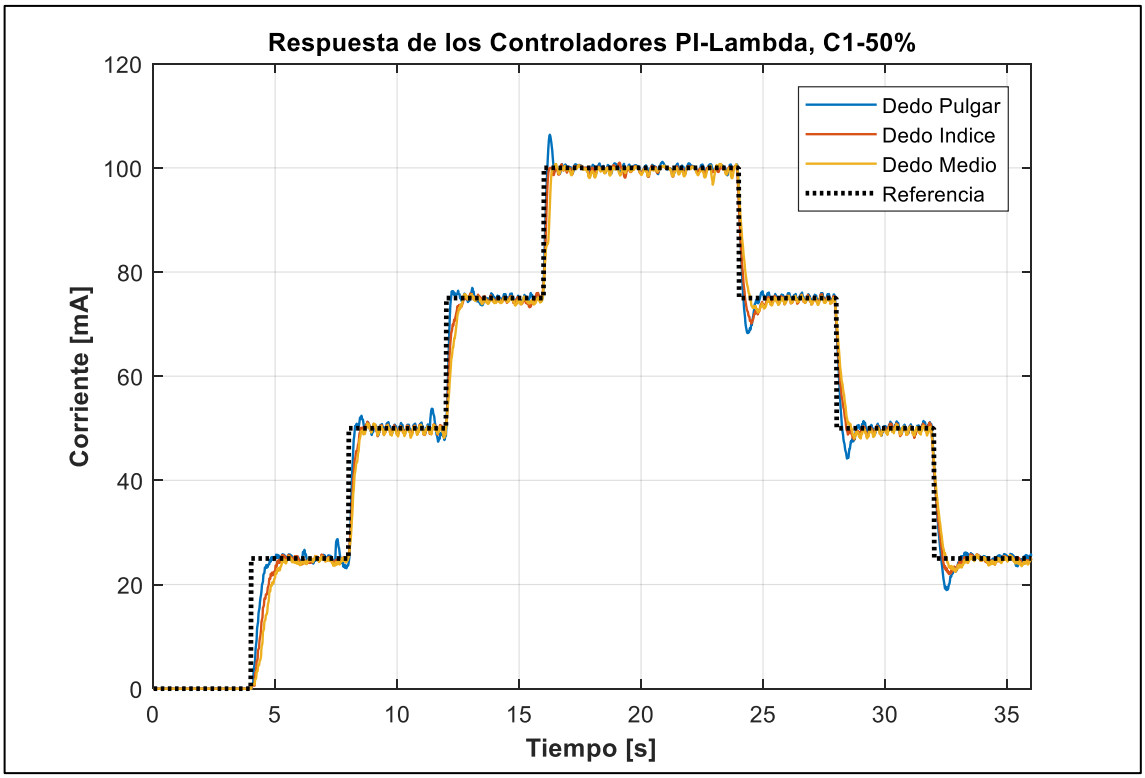
Respuesta del controlador basado en medición de corriente, con los objetos de prueba E4, C1, C2, C3

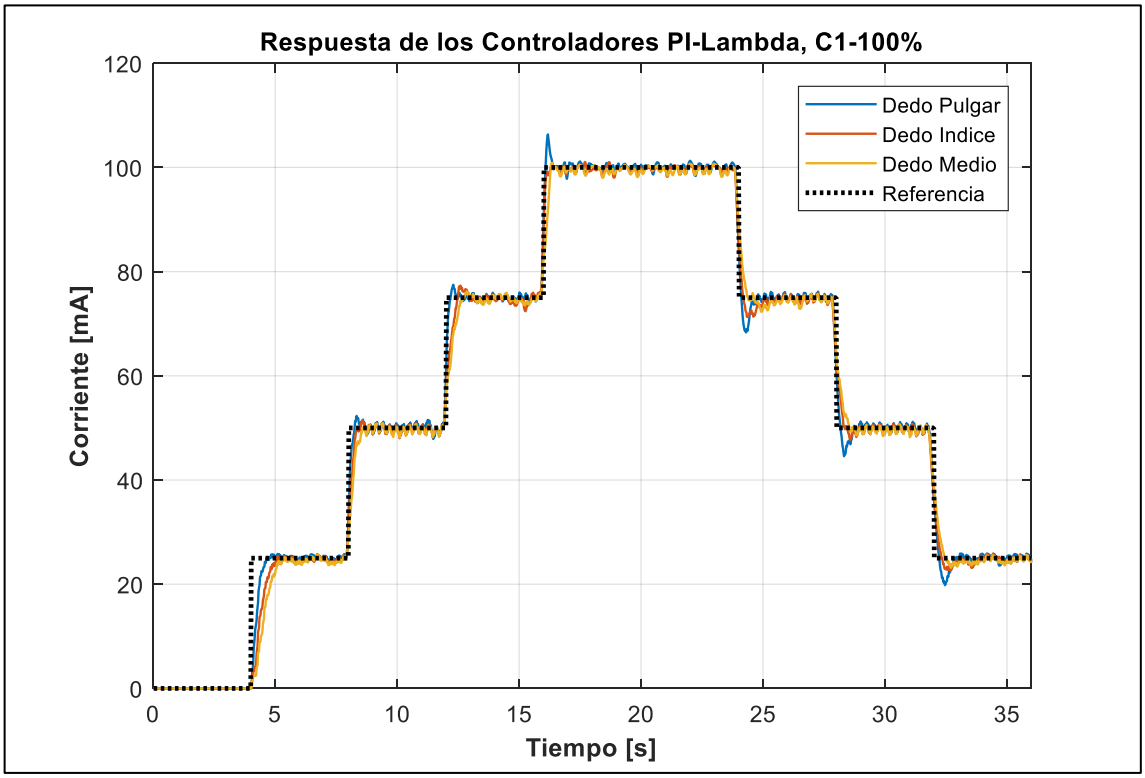


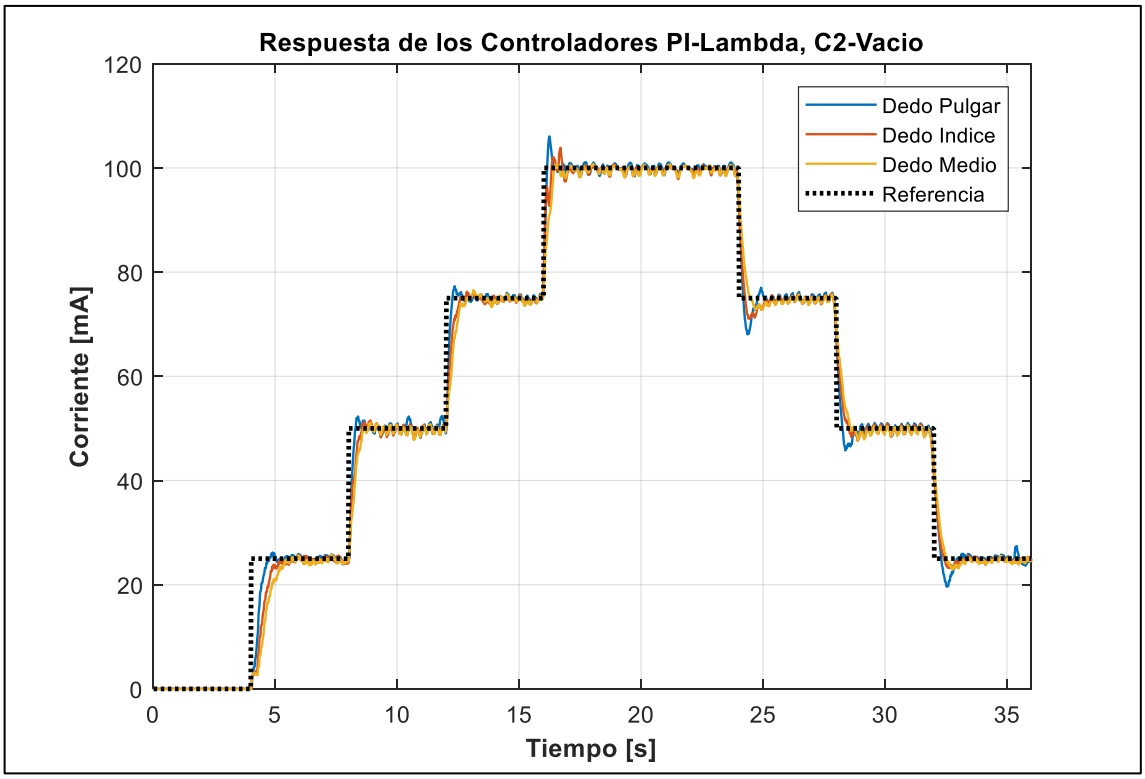


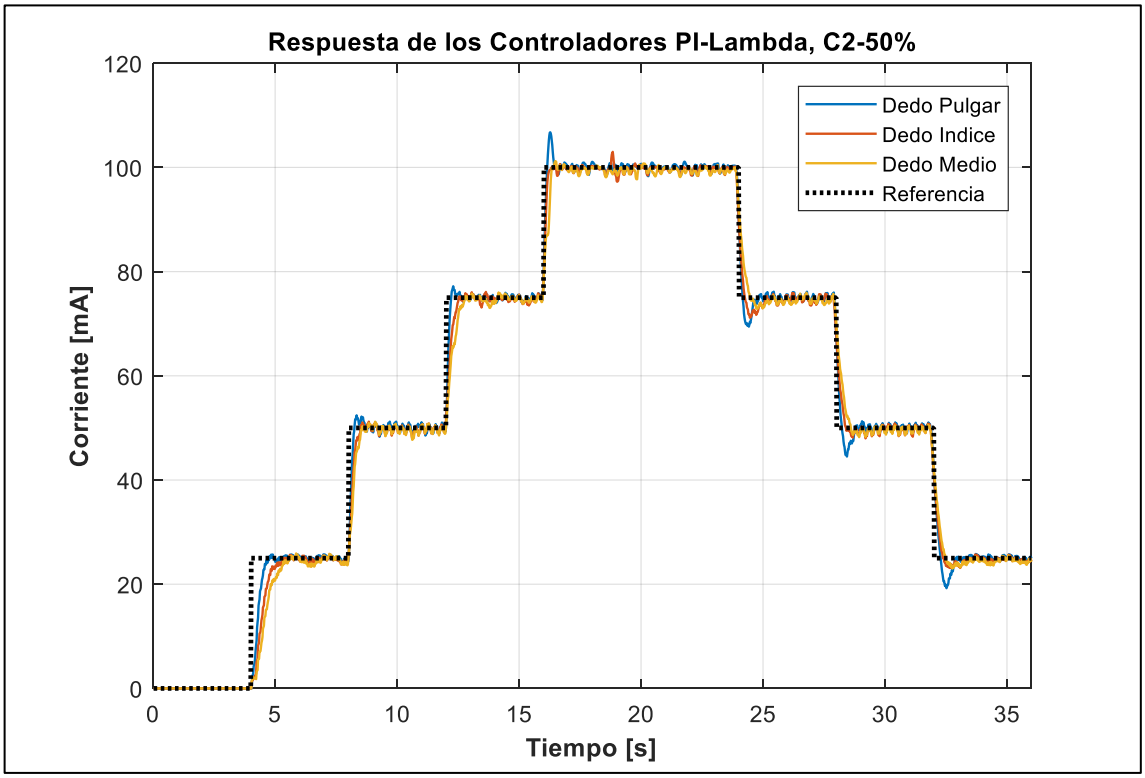

Respuesta de los Controladores PI-Lambda, E4-50%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

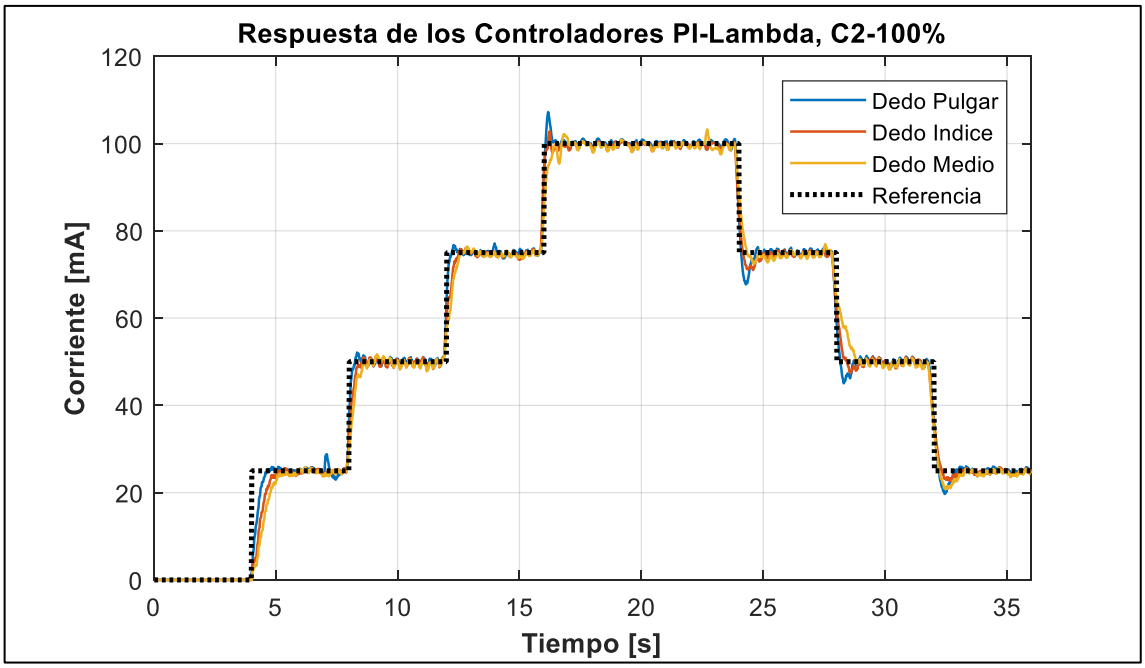

Respuesta de Controladores PI-Lambda, C1-Vacio
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

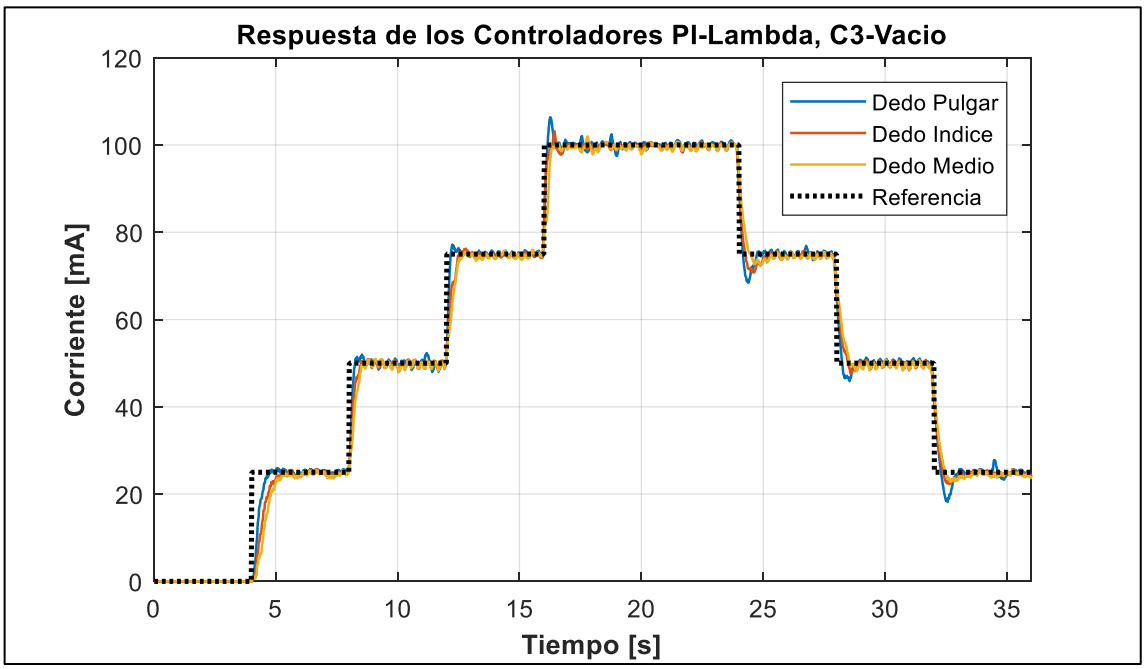

Respuesta de los Controladores PI-Lambda, C1-50%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

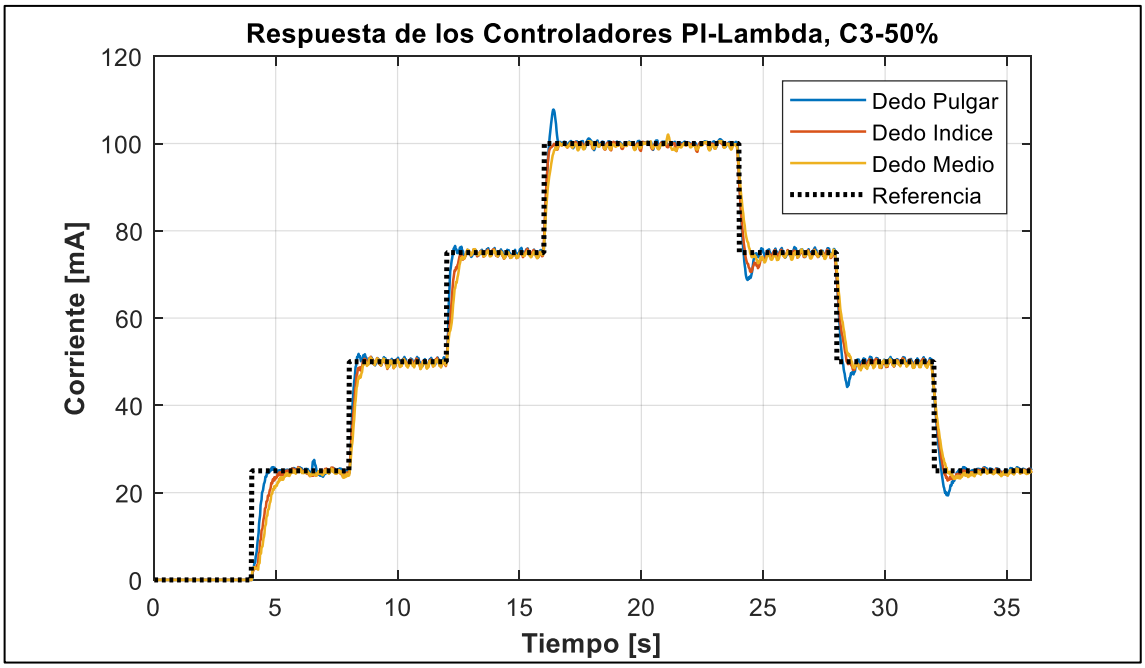

Respuesta de los Controladores PI-Lambda, C1-100%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

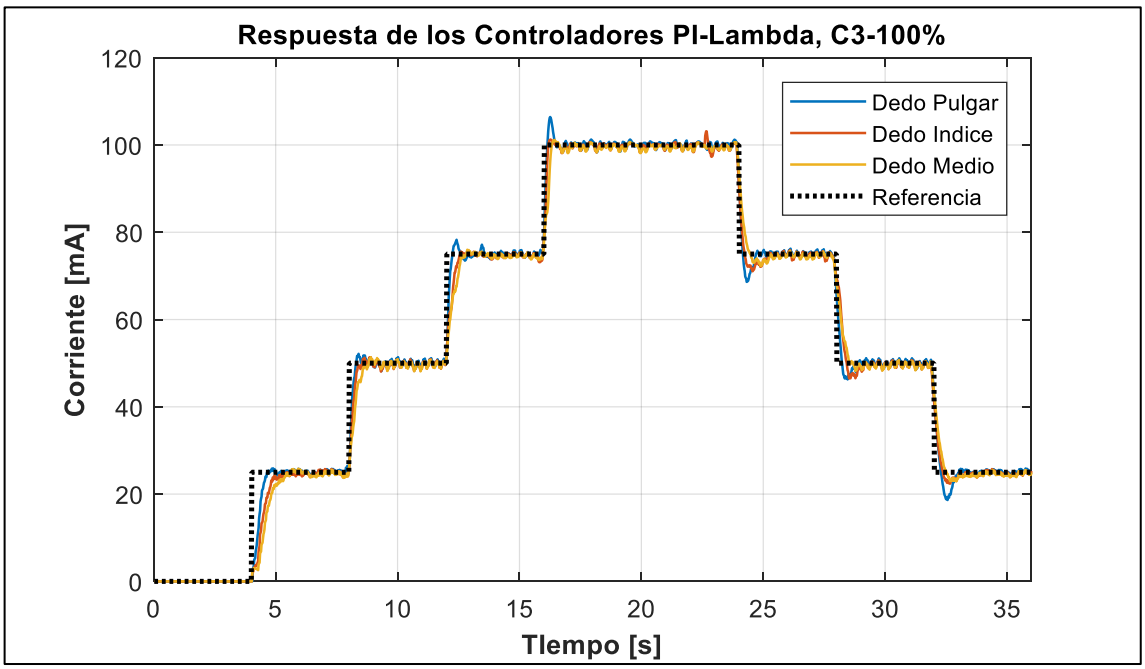

Respuesta de los Controladores PI-Lambda, C2-Vacio
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35


Respuesta de los Controladores PI-Lambda, C2-50%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

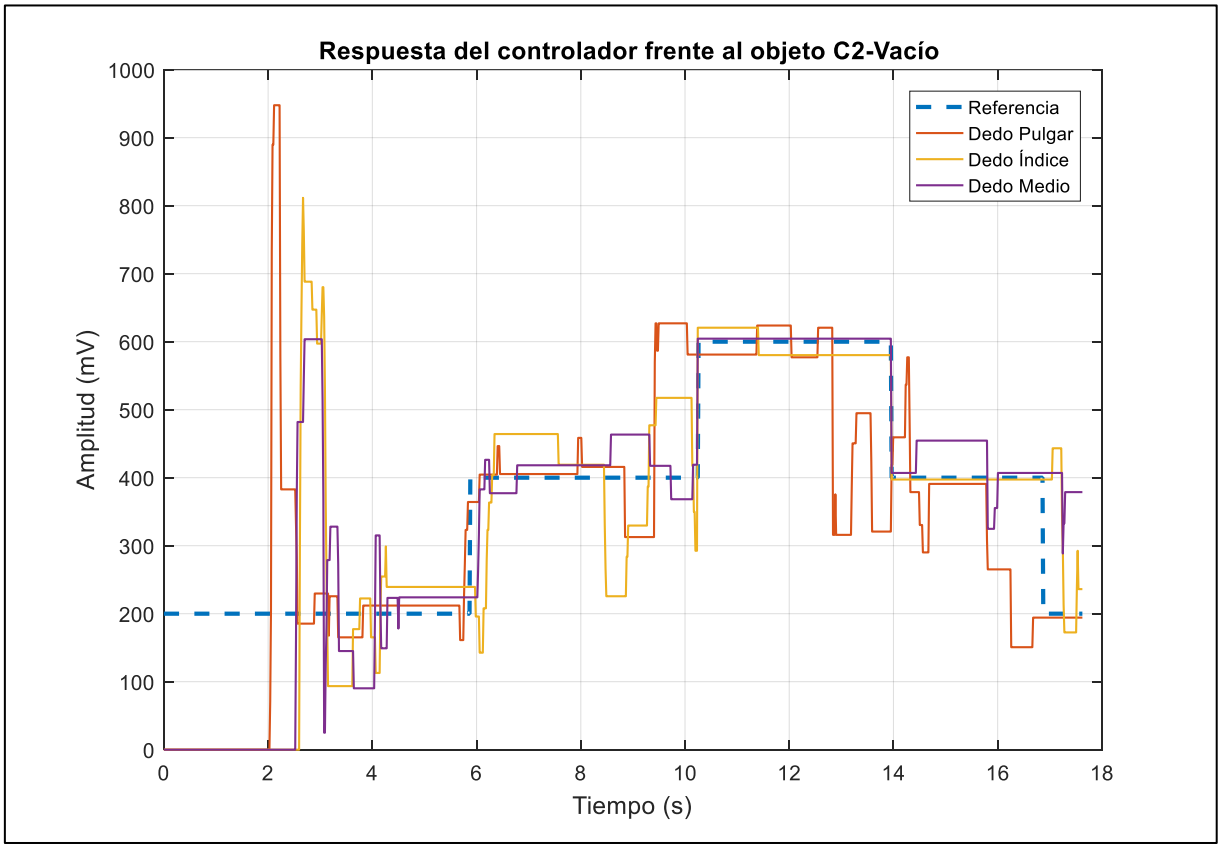

Respuesta de los Controladores PI-Lambda, C2-100%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

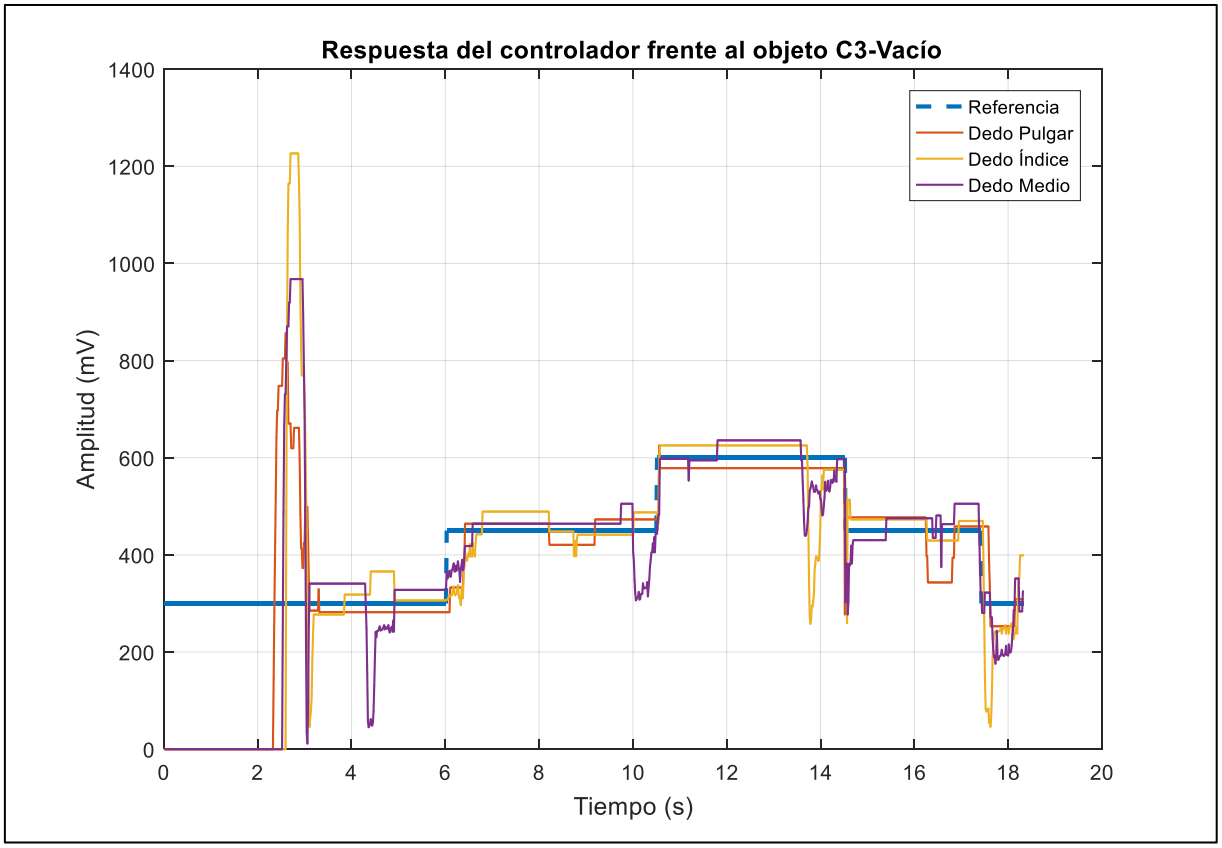

Respuesta de los Controladores PI-Lambda, C3-Vacio
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35


Respuesta de los Controladores PI-Lambda, C3-50%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35


Respuesta de los Controladores PI-Lambda, C3-100%
Corriente [mA]
Tiempo [s]
Dedo Pulgar
Dedo Indice
Dedo Medio
Referencia
0
20
40
60
80
100
120
0
5
10
15
20
25
30
35

Respuesta del controlador basado en medición de fuerza mediante los sensores FSR, con los objetos de prueba C2 y C4 sin peso adicional.





Agarre de los objetos C2, C3, E1, E3