UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

ADAPTACIÓN DE UN SENSOR DE PROXIMIDAD Y TÁCTIL MEDIANTE FPGAS PARA LA INTERACCIÓN HUMANO-ROBOT

Titulación: Ingeniero de Telecomunicación
Autor: Díaz González, Richard
Tutores: Pérez Carballo, Pedro
Escaida Navarro, Stefan

Fecha: Julio de 2017

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

ADAPTACIÓN DE UN SENSOR DE PROXIMIDAD Y TÁCTIL MEDIANTE FPGAS PARA LA INTERACCIÓN HUMANO-ROBOT

HOJA DE FIRMAS

Alumno

Fdo.: Díaz González, Richard

Tutor

Fdo.: Pérez Carballo, Pedro

Tutor

Fdo.: Escaida Navarro, Stefan

FECHA: Julio de 2017

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

ADAPTACIÓN DE UN SENSOR DE PROXIMIDAD Y TÁCTIL MEDIANTE FPGAS PARA LA INTERACCIÓN HUMANO-ROBOT

HOJA DE EVALUACIÓN

CALIFICACIÓN: _____

Presidente

Fdo.: Navarro Mesa, Juan Luis

Vocal

Fdo.: Pérez Álvarez, Iván Alejandro

Secretario

Fdo.: López Feliciano, José

Fecha: Julio de 2017

Quisiera agradecer a mis padres
todo el apoyo y dedicación que
día a día me han brindado,
ya que sin su esfuerzo y trabajo
jamás hubiera llegado a donde estoy.

"Work hard in silence and let your success be your noise"

Resumen

Este proyecto tiene por objeto optimizar el procesamiento de la señal de un sensor de proximidad y táctil que fue desarrollado por el Instituto de Antropomática y Robótica de la división de Automatización Inteligente de Procesos y Robótica del Instituto Tecnológico de Karlsruhe.

Se ha realizado un estudio del sistema para conocer su arquitectura: el sistema de obtención de datos junto con el sensor, la lógica interna de procesamiento de señales con todos sus módulos, el subsistema de envío de datos y el sistema cliente para la recepción de los datos.

Durante el estudio del sistema, el trabajo se ha centrado en el subsistema de procesado de señal. A este fin, se ha optado por crear un modelo en Matlab y simular su comportamiento. Con los resultados obtenidos de esta simulación se ha analizado cada uno de los cambios que el sistema real produce en la señal.

El siguiente paso consiste en profundizar en el conocimiento de cada uno de los submódulos de la lógica de procesamiento de señal. Para ello se ha realizado el análisis de cada submódulo por separado: el submódulo DFT, el submódulo de raíz cuadrada, los submódulos de filtros FIR, el submódulo de diezmado, el submódulo integrador y el submódulo FIFO, entre otros. Como quiera que se trata de módulos ya disponibles en VHDL, se han propuesto mejoras para su rediseño con el objetivo de mejorar sus prestaciones.

Una vez realizadas las tareas descritas, se toman métricas de latencia. Para ello se dispone de un temporizador que se activa y desactiva en función de ciertas condiciones de las señales bajo estudio. Las señales de arranque se configuran para que se activen al tiempo que un dato entra en el sistema y la señal de finalización se establece justo cuando este dato sale del sistema. La segunda forma elegida consiste en configurar un *testbench* para simular el sistema, analizando el flujo de información y conocer así su latencia. Al tiempo calculado con cada uno de estos métodos tenemos que añadirle el tiempo necesario para que los datos salgan del subsistema a través de la interfaz *Ethernet* y lleguen a la máquina cliente o al control del robot.

El trabajo concluye proponiendo un conjunto de mejoras para optimizar el sistema en términos de latencia y procesamiento de datos.

Abstract

This project aims to improve the signal processing of a proximity and touch sensor which was developed by the Division of Intelligent Process Automation and Robotics of Institute of Anthropomatics and Robotics, Karlsruhe Institute of Technology.

A study of the system has been carried out to know its architecture: the data acquisition system together with the sensor, the internal signal processing logic with all its modules, the data sending subsystem and the client system for the data reception.

During the study of the system, the work has been focused on the signal processing subsystem. To this end, we have chosen to create a model in Matlab and simulate its behavior. With the results obtained from this simulation we have analyzed each change that the real system produces in our signal.

Next step is to deepen the knowledge of each submodule of the signal processing logic. For this purpose, each submodule has been analyzed separately: the DFT submodule, the square root submodule, the FIR filter submodule, the decimation submodule, the integrator submodule and the FIFO submodule, among others. As for modules already available in VHDL, improvements have been proposed for its redesign in order to improve its performance.

Once the tasks described are performed, latency metrics are taken. A timer has been used and it is activated and deactivated depending on certain conditions of the signals under study. The start signals are configured to be activated as data enters in the system and the end signal is set just when this data exits the system. The second form chosen is to configure a testbench to simulate the system, analyzing the information flow and knowing its latency. At the time calculated with each of these methods we have to add the necessary time for the data to leave the subsystem through the Ethernet interface and reach the client machine or control the robot.

The work concludes by proposing a set of improvements to optimize the system in terms of latency and data processing capabilities.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Antecedentes

Desde principios del siglo XX el ser humano ha estado buscando la forma de crear una réplica artificial total o parcial del hombre y su comportamiento. A partir de esta idea, ha ido creando máquinas para que realicen ciertas tareas por él, consiguiendo a su vez mejores resultados con menor tiempo y esfuerzo. Aunque en un principio, estas máquinas eran muy rudimentarias, poco a poco se han ido perfeccionando y volviendo cada vez más completas y complejas. Para ello, ha sido necesaria la colaboración de varias ramas de conocimiento durante algunas décadas llegando en la actualidad a la construcción de complejos robots, autómatas y humanoides.

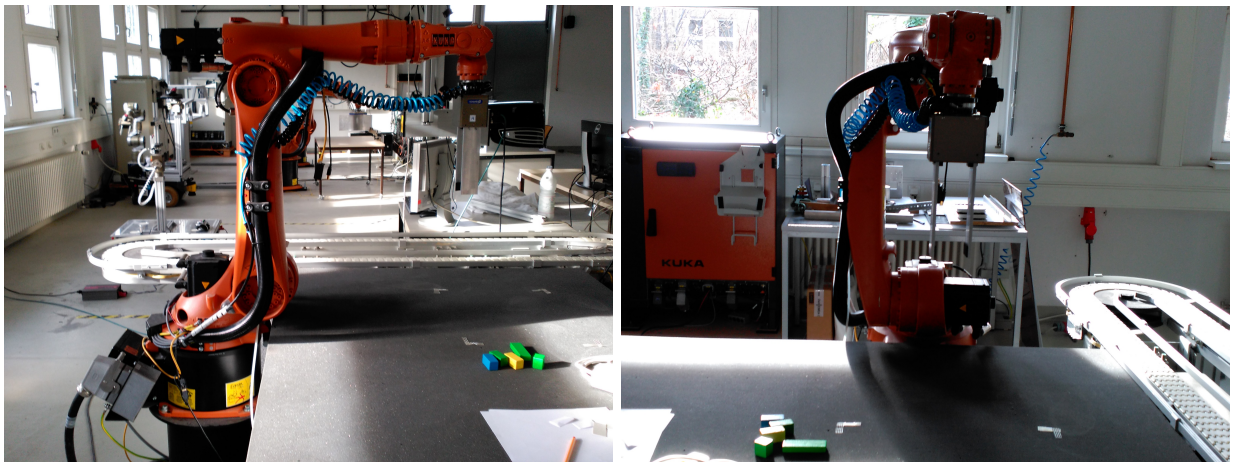
Básicamente un robot se compone de tres grandes bloques funcionales:

- bloque de adquisición de información del exterior,
- bloque de procesamiento de la información y toma de decisiones, y
- bloque para la interactuación con el entorno.

Estos tres bloques coinciden, normalmente, con tres ramas de la ingeniería: la electrónica, la informática y la mecánica. Ningún robot puede estar completo si no se compone de partes que necesiten de cada una de estas ramas de conocimiento. De esta manera, se puede afirmar que la **electrónica** cobra mayor protagonismo en la adquisición de información del exterior y su procesamiento mediante sistemas embebidos o empotrados, la **informática** hace lo propio en cuanto a la creación de algoritmos complejos para el procesamiento de la información y la **mecánica**, en la etapa encargada de la actuación con el exterior, muy de la mano con la electrónica[1].

En este proyecto se trabajará con un brazo robótico de unos 2 metros de longitud fabricado por la empresa **KUKA Roboter GmbH**[2]. Concretamente, el modelo de robot que se ha adquirido de dicha empresa es el **KUKA KR16**[3], el cual se puede ver en la *Figura 1.1*. Este modelo cuenta con hasta 6 ángulos de movimiento totalmente independientes y es capaz de soportar cargas en su extremo de hasta 20 kg. de peso.

Sus especificaciones técnicas se encuentran en el *Anexo A* de la página 109.



(a) Robot KUKA de perfil (b) Robot KUKA de cara

Figura 1.1: Robot KUKA con el que se trabajará en este proyecto.

Además de la posibilidad de programarlo, este brazo robótico, viene dotado con un panel de control remoto alámbrico como el que se ve en la *Figura 1.2*, con el que se puede manejar manualmente y configurar una serie de parámetros y directrices. Este control remoto también tiene otras funciones como la de apagado de emergencia o la de controlar en tiempo real todos los parámetros del robot, tales como motores activos, grados de giro de éstos, etc. Las características principales de este control remoto y sus especificaciones se pueden encontrar en el *Anexo B* de la página 113.

Este trabajo se centrará en el bloque funcional de procesamiento de la información y la toma de decisiones, aunque también se estudiará brevemente la adquisición de los datos desde los sensores y su adaptación para poder ser procesados.

Los orígenes de los sensores táctiles se remontan a la década de los años 60 del siglo pasado, donde se comenzaron a construir los primeros teclados para los ordenadores de la época. Los sensores de estos teclados eléctricos eran enteramente mecánicos y se accionaban por presión. Por supuesto ninguno de ellos era electrónico y mucho menos capacitivo. En 1981 se comenzó a utilizar sensores capacitivos[4] para la tecnología *CRT*.

En la actualidad, con la integración de diferentes tecnologías, tales como sensores capacitivos, táctiles, térmicos, resistivos y mecánicos, entre otros, se está un poco más cerca de conseguir emular el *sentido del tacto*. Sin embargo, un sistema tan perfecto implica una complejidad demasiado elevada y es necesario realizar avances significativos tanto en tecnología como en otras ciencias para obtener un sistema bioinspirado como el del tacto avanzado y fiable que explica Pedreño Molina et al. en su estudio[5].

Figura 1.2: Panel de control remoto del robot KUKA KR 16.

En este sentido, se prevé que el siguiente paso en el desarrollo de este tipo de sistemas sea si, con el mismo sensor capacitivo, fuese posible detectar objetos en la proximidad del espacio pero sin llegar a tocarlos. Ello añade gran funcionalidad al sistema, incrementando su seguridad y su potencialidad ya que dota al robot de la habilidad para detectar previamente el objeto a sujetar y actuar en consecuencia para alinearse con él, buscando la mejor manera de agarrarlo e incluso, evitar colisiones inesperadas con otros objetos del entorno previendo sus trayectorias y adelantándose a ellas.

En este marco de actuación, el grupo de investigación de la División de Automatización de Procesos Inteligentes y Robótica (IPR) del Instituto de Antropomática y Robótica (IAR) del Instituto Tecnológico de Karlsruhe (KIT), Alemania; centra su actividad en la investigación y el desarrollo de diferentes tipos de sensores, actuadores y métodos para conseguir una interacción humano-robot más segura. Este grupo de investigación ha diseñado y construido un prototipo de sensor basado en la capacidad de los materiales, siendo esta propiedad física de especial importancia para este campo de trabajo. La tecnología existente permite detectar variaciones mínimas de esta

propiedad en el espacio lo que la hace útil para aplicarla con éxito en la robótica.

Actualmente, el sensor se encuentra en fase de desarrollo para disminuir su tamaño físico y latencia. Su aspecto físico en este momento es el que se muestra en la *Figura 1.3*[6]. Además, se está trabajando para dotarlo de un amplio conjunto de aplicaciones que prueben su funcionalidad con los robots del laboratorio.

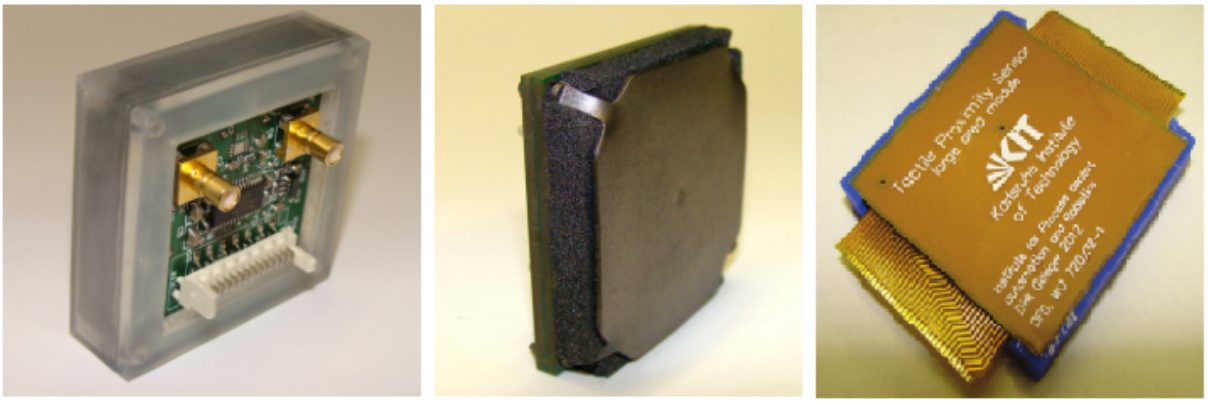


Figura 1.3: Vista interna y externa del prototipo del sensor.

El diseño estructural de este sensor puede verse en la *Figura 1.4* donde se pueden distinguir fácilmente el array de 4×4 *Taxels*, el electrodo superior de goma elástica conductora y la espuma interna que actúa de dieléctrico en el condensador formado. También se indican los flujos de corriente que circularían por él y sus sentidos.

Estos sensores estarán montados en una pinza robótica dual de placas diametralmente opuestas situadas en el extremo superior del brazo robótico y se acercarán y alejarán respectivamente para sujetar o liberar los objetos con los que interaccionará el robot. El controlador utilizado para sujetar dichas pinzas al brazo robótico y dotarlas de movilidad ha sido el **PG 70**[7][8] fabricado por *ROS Components* y sus especificaciones pueden consultarse en el *Anexo C* de la página 121 de esta memoria.

En la *Figura 1.5* se puede ver una vista frontal y trasera de las pinzas que se utilizan para este proyecto. Como puede verse, los sensores se presentan distribuidos sobre la superficie de la pinza, encajados en ella, formando una estructura cuadrangular de 2×2 sensores, lo que significan 4 sensores en cada pinza. Cada sensor mide unos $4cm \times 4cm$ de longitud. Cada una de las pinzas mide $10cm \times 10cm$.

En la parte trasera de la pinza se localiza un pequeño circuito que multiplexa las señales de cada uno de los 4 sensores de su matriz y las envía a la *FPGA*. En esta parte se pueden ver las 3 conexiones necesarias para ello: una para la señal de excitación de las placas de los sensores y otras dos para las medidas de proximidad y tactilidad respectivamente. Una vista previa de la electrónica necesaria para esta configuración de dos matrices de 4 sensores puede verse en la *Figura 1.6*. Cada una de

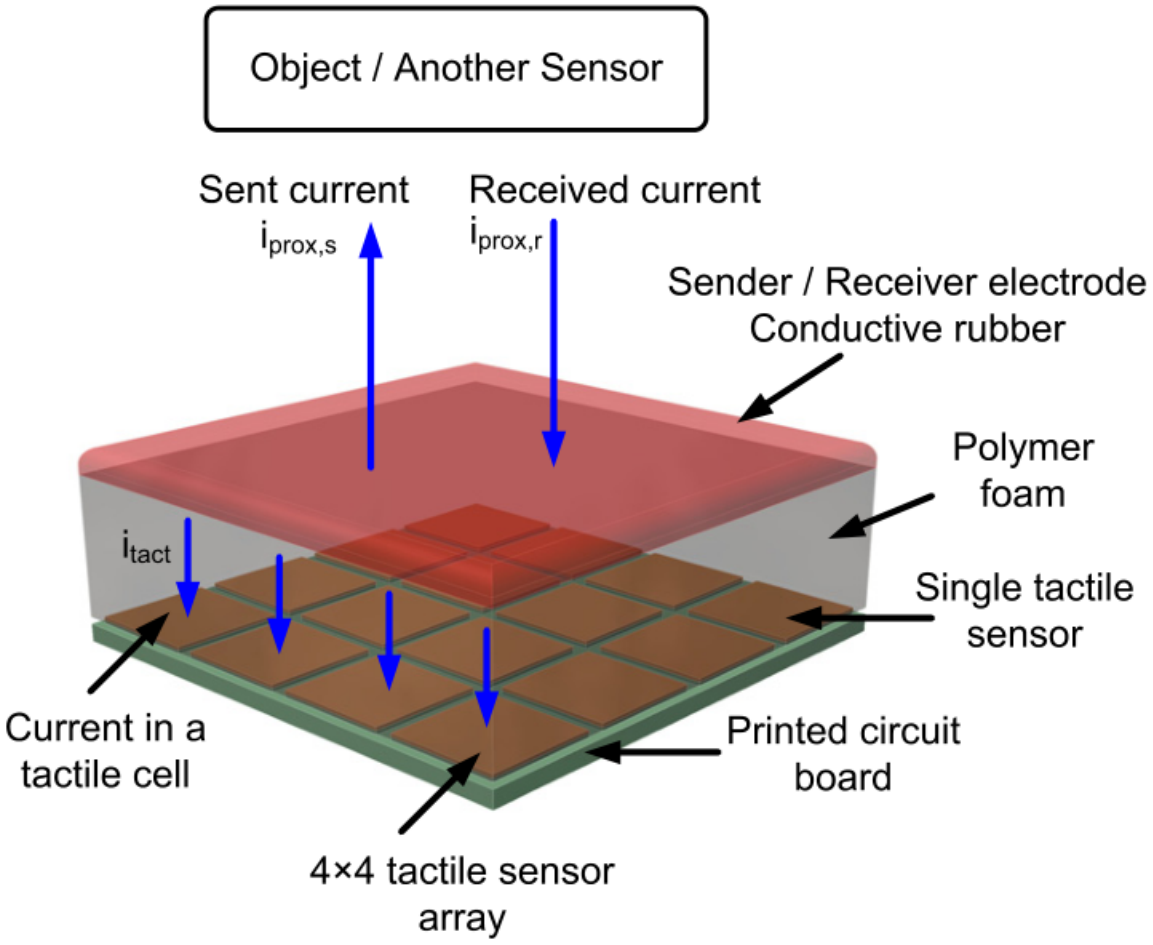
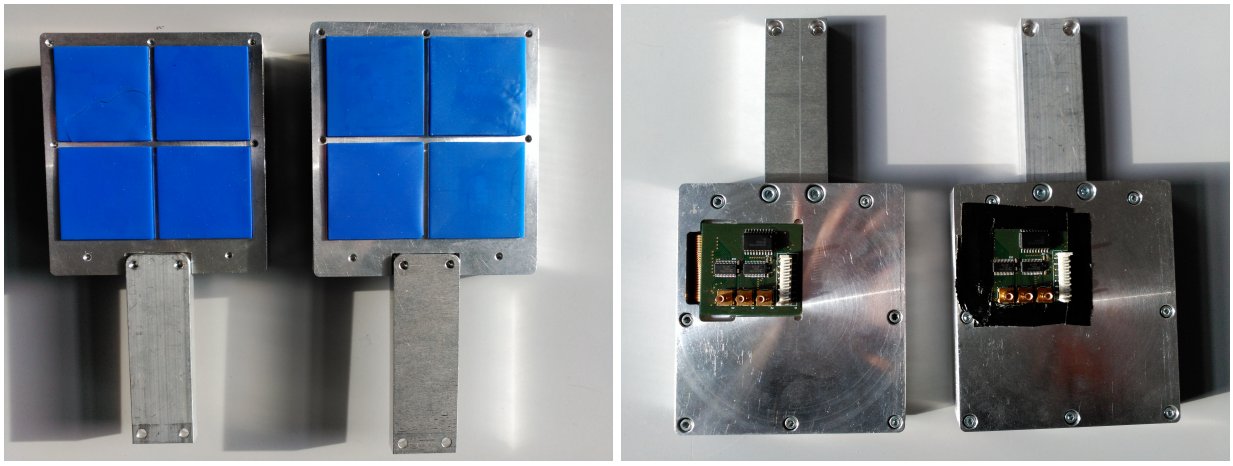


Figura 1.4: Diseño estructural del sensor de proximidad y táctil.



(a) Parte delantera de las pinzas

(b) Parte trasera de las pinzas

Figura 1.5: Visión frontal y trasera de las pinzas de trabajo.

las pinzas se conecta a un canal diferente de la placa de expansión de la *FPGA*. Esta placa de expansión consta de 10 canales multiplexados que pueden ser activados o desactivados para crear diferentes configuraciones y así, no habrá de esperar por el retardo que introduce un canal si éste es desactivado. Esta placa también se encarga de generar la señal de excitación que será enviada hacia los sensores y de encauzar las medidas recabadas hacia la *FPGA* del sistema. Su estructura puede verse en el *Anexo D* (página 133 de esta memoria.

Figura 1.6: Electrónica necesaria para la configuración de una pinza de 4 sensores monocanal.

La idea de este proyecto surge de la necesidad de conseguir métodos cada vez más sofisticados para interaccionar con los robots de manera más rápida, autónoma y segura.

La transformación que implica pasar de dar directrices a un robot mediante un programa a poder interactuar con él directamente mediante el tacto o la voz ha supuesto un avance significativo para estos sistemas avanzados. Ahora se pretende dar un paso adicional, fijando como objetivo conseguir interactuar con el sistema sin tener que recurrir al contacto físico o al habla para que, de esta manera, incluso pueda realizar tareas no predefinidas sin la supervisión de una persona. De la misma manera, se pretende que pueda conocer su entorno próximo e interactuar con él sin que sea necesario “enseñarle”, pudiendo así tomar decisiones y determinar la mejor opción para realizar la tarea encomendada, en definitiva, que el propio robot por si mismo tenga la habilidad de aprender del entorno en el que se encuentre.

Este nuevo enfoque de trabajo está basado de la necesidad de corregir fallos, errores, averías, etc. en cadenas de montaje o fábricas sin la necesidad de parar toda la producción relacionada a los robots adyacentes ya que el robot encargado de mantenimiento puede interactuar con el resto y acometer su tarea sin interferir en la trayectoria de los demás y, lo que es más importante, aún sin que haya necesidad de una interacción física entre ambos. Esto es aplicable también a la interacción con los trabajadores que se hallen en el entorno y a cualquier elemento externo que se introduzca al siste-

ma. El robot tendría la capacidad de reconocer en tiempo real su entorno más cercano, sin necesidad de un contacto físico, evitando de esta manera colisiones e interferencias de trayectorias y su consecuente pérdida de tiempo, recursos y, en definitiva, sin añadir costes a la producción.

Cuando se formuló la idea de realizar este proyecto ya existían varios estudios similares sobre sensores duales de proximidad y táctiles. Un ejemplo es el sensor integrado que utiliza simultáneamente una medida óptica y otra eléctrica[9]. Este sensor consiste en varios electrodos y un fotodiodo emisor de luz. En el método de funcionamiento propuesto se mide la admitancia entre los electrodos. Adicionalmente, las propiedades ópticas de la superficie del sensor se miden a partir de la admitancia del fotodiodo. Este método salva el inconveniente de que las propiedades eléctricas de la mayoría de los objetos varían en función de la distancia entre el sensor y el objeto, ya que la medida se realiza entre los electrodos del sensor y estos electrodos siempre se encuentran a la misma distancia. Además, su composición exacta es conocida a priori. En el experimento se distingue entre 6 tipos de materiales: acrílico, cristal, aluminio y esponjas de 3 densidades diferentes. La detección de la diferencia entre las medidas fue demostrada durante el contacto, mientras la proximidad del objeto era también detectada.

Otro estudio en esta misma línea, basada en trabajos previos trata sobre el concepto de emplear óxido de aluminio anódico nanoporoso (np-AAO) como aislante y una capa dieléctrica para formar un sensor de proximidad integrado-vertical inductivo-capacitivo[10]. Las ventajas de este sensor son las siguientes:

- Amplio rango de la distancia de detección. La detección capacitiva se utilizaría para tener mayor resolución en distancias cortas y en distancias largas se utilizaría la detección inductiva.
- Con este nuevo método, pueden detectarse objetos conductores y no conductores.
- El tamaño del chip se reduce considerablemente al utilizar un diseño integrado verticalmente.
- El efecto de borde se ve reforzado por la espiral de la bobina.
- El np-AAO tiene mejores propiedades eléctricas que otros materiales utilizados anteriormente como el óxido de silicio.

Con el paso de los años, el rápido desarrollo de la tecnología de detección táctil ha contribuido significativamente a la mejora del control táctil intuitivo y la interacción humano-máquina inteligente. A parte de la detección de presión o táctil, la detección de proximidad como función complementaria puede extender el modo de detección de los sensores táctiles de función simple[11]. En este ejemplo se presenta un sensor transparente, capacitivo, dual, con estructura matricial que integra la funcionalidad de detectar presión y proximidad en un sólo dispositivo. La excelente resolución espacial ofrecida por la respuesta aislante de los píxeles capacitivos da la posibilidad al sistema

de realizar una identificación precisa de la localización y proximidad de los objetos y la presión de una carga, con una respuesta rápida y de elevada estabilidad y reversibilidad.

citesohgawa2014 trabaja en el desarrollo de un sensor multimodal con Silicio microvoladizo incrustado en *PDMS*, fabricado y caracterizado para la medición de las fuerzas de proximidad y tactilidad[12]. El sistema incluye un componente fotosensible de Silicio que hace que la impedancia de CA $(> 0,5MHz)$ aumente con el aumento de la distancia entre el sensor y el objeto a causa de la disminución de la intensidad de la luz reflejada en la superficie del objeto. Por otra parte, la impedancia de AC es diferente entre tierra y el objeto próximo debido a la diferencia de la distribución de campo electrostático entre los electrodos, de modo que se sugiere que la proximidad se puede detectar como el cambio de impedancia por la luz, así como el campo eléctrico.

1.2. Objetivos

El objetivo principal de este proyecto es el estudio y la caracterización de retardos, ruta crítica y factores que propician la pérdida de señal del sistema compuesto por los sensores y el procesado de señal necesario para que el robot actúe en respuesta a los estímulos captados. El sistema está capacitado para trabajar con hasta 10 canales y 16 sensores por canal, lo que hace un total de 160 sensores. Para este proyecto se trabaja con una configuración particular de 2 canales activos y 4 sensores por cada canal, lo suficiente para ser aplicado al funcionamiento en una pinza neumática estándar. Finalmente se evaluarán los efectos de escalar esta configuración.

Este objetivo global se ha dividido en los siguientes objetivos operacionales para acometer el trabajo:

1. Realizar el estudio de este tipo de sistemas y sus principios básicos de funcionamiento.

2. Analizar la arquitectura y funcionalidad del procesado de señal.

3. Crear un modelo de alto nivel en Matlab para el procesamiento de las señales del sistema.

4. Caracterizar la latencia del sistema.

5. Proponer alternativas para optimizar el sistema.

6. Evaluar los efectos de las mejoras propuestas.

7. Documentar el proyecto.

1.3. Metodología adoptada

A continuación se describe la metodología utilizada para abordar el problema y alcanzar los objetivos propuestos.

Se parte de la realización de un estudio detallado del conjunto total del sistema para identificar cada uno de sus bloques, que incluyen:

1. El subsistema de obtención de datos, compuesto por el sensor y la electrónica de medición.
2. La lógica de procesamiento de señal interna con todos sus módulos.
3. El subsistema de envío de datos.
4. El subsistema cliente para la recepción y presentación de los datos

En primera instancia, se ha procedido a realizar el estudio del proyecto, incluido código existente y documentación, así como, ejecutando el código y observando su funcionamiento para extraer conclusiones de interés. Asimismo se ha realizado un análisis de sistemas similares a partir de las publicaciones disponibles.

De este primer estudio del proyecto, se ha obtenido el diagrama de bloques presentado en la *Figura 1.7*. A la izquierda se muestra el brazo robótico, con los sensores instalados en las pinzas de su extremo. A cada pinza se conectan 3 cables que van directamente a la placa de multiplexación de canal, donde cada pinza se trata como un canal independiente. Las señales que circulan por cada uno de estos 3 cables son analógicas y se corresponden con el envío a los sensores de una señal de excitación, y la recepción de la señal de corriente correspondiente a la medida de proximidad y a la medida táctil, individualmente. La placa multiplexora está directamente conectada sobre la *FPGA* mediante un banco de pines *GPIO*. En la *FPGA* se realiza todo el procesado de señal, almacenamiento temporal de las muestras, filtrados, etc. y, por último, toda la información procesada se envía a una estación cliente mediante una conexión *Ethernet* para su visualización y estudio.

A partir de este estudio del sistema se procede a analizar el subsistema de procesamiento de datos. Para ello se lleva a cabo su modelado y simulación en Matlab para diferentes movimientos relativos de un objeto con respecto a los sensores. El objetivo de esta simulación es conocer todos los cambios que el sistema producirá en los datos y así poder conocer la señal recibida en el subsistema cliente para cualquier señal que pueda ser recibida en el sensor.

De este análisis del procesamiento de datos se puede concluir que los datos resultantes del objeto de estudio pasan a través de un conjunto de sistemas que responden al diagrama de la arquitectura que se presenta en la *Figura 1.8*. En este diagrama se puede ver un esquema de los módulos más importantes del procesado de señal del

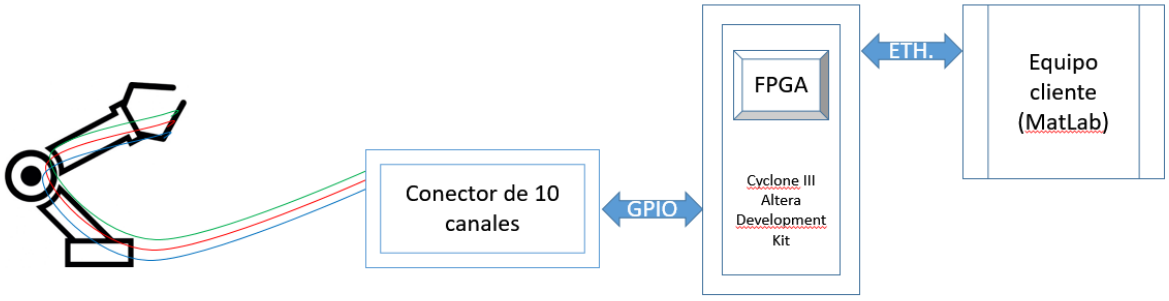


Figura 1.7: Diagrama de bloques del sistema general.

sistema. El **objeto de estudio principal de este trabajo** son estos módulos implementados en la *FPGA* ya que es donde se procesan los datos obtenidos por los sensores para luego ser visualizados. Es por ello que el diagrama ha sido creado tratando de representar los bloques más importantes de esta parte del sistema. De esto se concluye que la optimización de estos módulos producirá mejoras en las prestaciones del sistema.

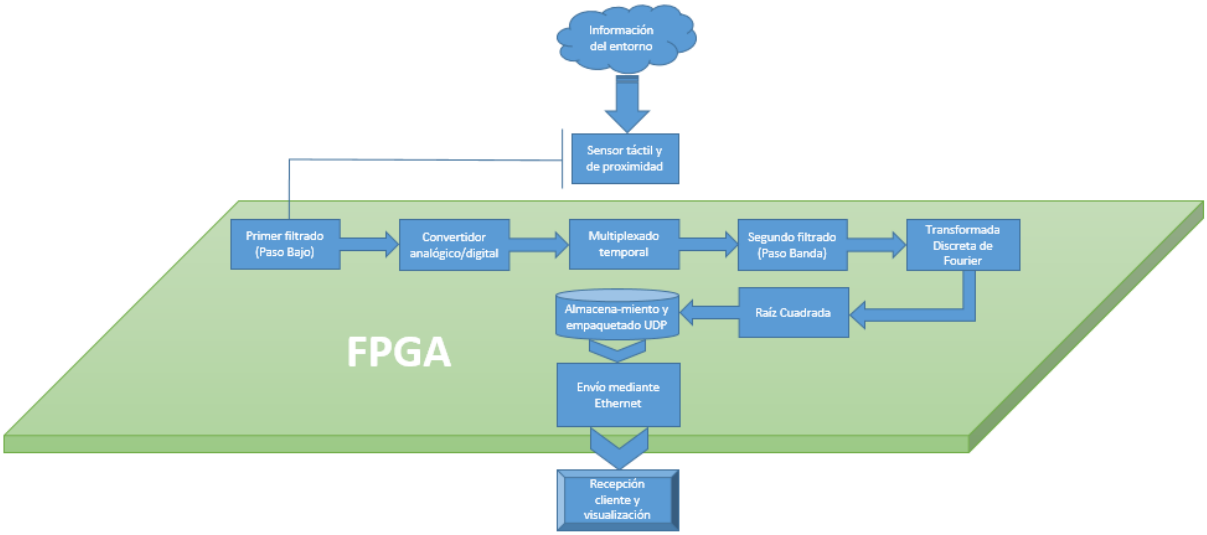


Figura 1.8: Diagrama arquitectónico del sistema.

El siguiente paso consiste en analizar cada bloque de la lógica de procesamiento de datos, a fin de poder plantear alternativas para su optimización. Se analiza la descripción *VHDL* existente de estos bloques, comprendiendo su funcionalidad y planteando las modificaciones que son necesarias para realizar un incremento en la velocidad de procesamiento de los datos, evitando afectar a aspectos claves de la implementación, como por ejemplo la resolución de los datos.

Finalmente, se calcula la latencia del sistema. Para ello se añade un conjunto de *Timers* al modelo del sistema, observando el flujo de datos en el sistema. Cuando el dato entra en el bloque bajo observación, se genera un evento que arranca el correspondiente *Timer* y cuando el dato sale de éste, se generara el evento de parada del *Timer*. El registro de la diferencia de dichos *Timers* proporciona una medida real de la latencia del bloque. El dimensionamiento de los *Timers* se ha realizado a partir de un análisis previo de la latencia del sistema, con lo que se asegura el rango de posibles valores resultantes y la fiabilidad de las mediciones realizadas. Con ello se consigue crear un modelo detallado para la simulación del sistema, configurando un *Testbench* en *VHDL* que permita experimentar diferentes situaciones que se pueden presentar en el sistema final.

1.4. Organización de la memoria

A continuación se describe la organización de esta memoria, realizando una breve descripción de cada uno de los principales apartados.

Una vez introducido el problema y los objetivos del trabajo, el segundo capítulo de esta memoria se va a dedicar al estudio del estado del arte en materia de sensores de proximidad y táctiles.

en el capítulo 3 se realiza el estudio de la arquitectura del sistema, el estudio del sensor, del sistema de procesado de señal y a la transmisión y presentación de las datos.

El cuarto capítulo, está dedicado al estudio y modelado mediante Matlab**??** del sistema.

El capítulo 5 se centra en la búsqueda de la ruta crítica del sistema, el estudio de la latencia de ésta ruta crítica y la elaboración del banco de distintos entornos de test o *Testbenchs* para su posterior simulación y análisis de latencias. El resultado es la propuesta de optimización del sistema.

En los capítulos finales se presentan los resultados obtenidos, se obtienen conclusiones y se detallan algunos trabajos futuros.

El documento se complementa con diferentes anexos que aportan información complementaria al trabajo realizado, incluyendo el código Matlab desarrollado.

Capítulo 2

Estado del arte

2.1. Introducción

Este capítulo esta dedicado a realizar un estudio del estado de la técnica en las líneas de actuación principales del proyecto. En concreto se estudian los sistemas sensores, la metodología de diseño para el sistema de procesado de señal usando *FPGAs*, las técnicas de modelado del sistema, en concreto la creación y evaluación de modelos de alto nivel. Igualmente se estudiarán técnicas de análisis temporal del software empotrado en sistemas embebidos.

2.2. Sistemas sensores de proximidad y táctiles duales

Existen diferentes sistemas integrados que utilizan sensores capacitivos de tecnología similar a la utilizada en este proyecto y que están dedicados a aplicaciones tales como pantallas táctiles, prótesis robóticas, utilidades en la industria aeroespacial para, por ejemplo, medir la intensidad y variación de diversos fenómenos naturales en el espacio y en capas elevadas de la atmósfera, etc. En los párrafos que siguen se hace un resumen de algunas de las tecnologías sensoras alternativas a la utilizada en este proyecto en las que se ha trabajado para obtener un sensor táctil y de proximidad de funcionamiento dual.

Ma et al. describe en [13] los trabajos realizados en la Universidad Nacional de Taiwan, Taipei, donde se experimenta con sensores creados a partir de nanotubos de carbono (CNT). Los CNT poseen unas propiedades eléctricas, mecánicas y térmicas únicas. Este nuevo sensor puede medir fuerzas normales y fuerzas compartidas y el proyecto se centra en la comparación de las fuerzas compartidas medidas por un *array* de sensores que incorpora un patrón denominado *Buckypaper* de gran radio de curvatura. Este tipo de implementación le proporciona capacidad sensora anisótropa, flexibilidad, facilidad de fabricación y bajo coste. Cagatay et al. describe igualmente en [14] el trabajo realizado en la Universidad Técnica de Munich (Alemania) donde tratan

de reproducir una piel sensible utilizando igualmente CNTs.

Por otro lado, Braun et al. muestra en [15] los trabajos que realizan tanto la Universidad Técnica de Darmstadt junto con el Instituto Tecnológico de Fraunhofer, (Alemania) en el desarrollo de un sensor capacitivo de proximidad, similar al utilizado en este trabajo, que aporte la funcionalidad necesaria para interactuar con entornos inteligentes y que facilite diseñar aplicaciones que puedan trabajar en dichos entornos. Este proyecto está orientado sobre todo a la interacción con un ser humano para, por ejemplo, el desarrollo de tareas diarias o el reconocimiento de todo tipo de gestos y acciones.

Otros trabajos en esta misma línea se están realizando en Corea del Sur. Por ejemplo, La Universidad de Inha está trabajando en un nanocristal de celulosa flexible y transparente al que añaden una fina película de óxido de grafeno reducido para detectar objetos próximos[16]. Esta tecnología está orientada a la detección de eventos en pantallas táctiles y debido a sus propiedades químicas lo hace mucho más rápido, sensible y preciso que los anteriores sensores de proximidad opto-electrónicos.

Por otra parte, en la Universidad de Sungkyunkwan (Corea del Sur) se está investigando en un modelo de sensores a partir de microbobinas de carbono[17]. Estos sensores táctiles y de proximidad consisten en multitud de parejas de electrodos activos y un único electrodo de tierra común, situados todos en un mismo plano. De esta manera el sensor es tolerante a repetitivos contactos de fuerzas externas. La capa superior consta de un sustrato elastómero dieléctrico compuesto en un 5 % de microbobinas de carbono. Este sensor ha sido probado y modelado en redes de 4x4 sensores para tareas de investigación.

A parte de los ejemplos anteriores, actualmente se trabaja en el diseño hacia una estructura de sensores táctiles y de proximidad que emulen la disposición antropomorfa de los dedos de la mano. Estos trabajos se podrían identificar como una evolución o mejora en la línea utilizada en este proyecto y que utilizan tecnologías similares para implementar el dispositivo. A continuación se exponen algunos ejemplos.

En la Escuela Politécnica Federal de Lausanne (Suiza) el científico Araromi et al. describe en [18] el trabajo de investigación que trata sobre cómo diseñar y fabricar sensores a partir de diferentes materiales a base de elastómeros y dieléctricos para conseguir sensores que se adapten a cualquier superficie. Esto tiene como fin último intentar reproducir una piel sintética con la que poder recubrir los robots.

Otro ejemplo en esta misma dirección es el descrito por Zhao et al. en el trabajo [19] del Instituto de Nanoenergía y Nanosistemas de Beijin (China) y el Instituto de Tecnología de Georgia (Atlanta, EEUU). Ambos centros investigan sobre la creación de una piel sintética con matrices formadas a partir de *arrays* de sensores flexibles y adaptables que simule el sistema somatosensorial de la piel humana.

En esta búsqueda de los materiales idóneos y de la creación óptima de una piel

sintética, el investigador Mittendorfer et al., expone en su publicación [20] una mejora de la capacidad de manipulación de los dedos robóticos proponiendo el diseño de una punta de dedo blando antropomórfica con receptores distribuidos aleatoriamente en su interior, de la misma manera que ocurre en una piel humana. La yema del dedo consiste en dos capas de silicona de caucho de diferente dureza que contienen dos tipos de receptores, extensómetros y de PVDF (Fluoruro de polivinilideno). La estructura de la yema del dedo es muy similar a la de un ser humano ya que consiste en un hueso, un cuerpo que lo recubre y una fina capa de piel. Los primeros resultados experimentales demuestran la gran capacidad de discriminación de la yema del dedo, ya que puede discriminar cinco materiales diferentes con solo presionar sobre ellos o frotarlos. Este nuevo avance deja abierta una posibilidad ya casi real, de que con ésta técnica se pueda dar sensibilidad a la totalidad de un robot.

En el artículo [21], Xia et al. presenta un sistema de detección capacitiva, en el cual se aborda la cuestión de la prevención de colisiones en ambientes parcialmente asistidos por robots mediante el uso de mediciones de distancia al objeto, seguimiento de movimiento y perfil de superficie de detección. El sensor consiste en una malla de múltiples electrodos, una un módulo de control digital, un convertidor analógico/digital y un módulo de procesamiento de datos. La malla se compone de 16 cuadrados metálicos organizados para formar una matriz de 4×4 condensadores. Las conexiones dentro de la matriz pueden ser reconfiguradas en tiempo de ejecución por la lógica de control digital para proporcionar múltiples funcionalidades de detección. Se utilizan modelos de regresión estadística para derivar la distancia y seguir el movimiento del objeto. A los datos medidos se le aplica un algoritmo de aprendizaje de tipo SVM (Support Vector Machine) para clasificar el perfil de la superficie de detección. Este sistema tiene una capacidad de detectar objetos a distancias de hasta $20cm$ del sensor y muestra una precisión del $90\,\%$ en el reconocimiento del perfiles de superficie.

Han et al. en su artículo [22] publicado en 2016 describe un sensor dual táctil y de proximidad, similar al estudiado en este proyecto, pero basado en CMCs (microbobinas de carbono). El sensor consta de múltiples capas de electrodos impresas en una placa de circuito impreso flexible (FPCB) y un sustrato dieléctrico en el que los CMCs están dispersos. Las propiedades eléctricas del sensor cambian cuando un objeto se aproxima o toca el sensor. Se utiliza un modo de detección capacitiva para la detección táctil y un modo de detección inductiva para la detección de proximidad. Los CMCs cambian impedancia eléctrica del sensor amplificando la señal de detección y, por lo tanto, la la sensibilidad del sensor aumenta. Las dimensiones actuales del prototipo son $30 \times 30 \times 0,6mm^3$ y se ha logrado una resolución espacial de $3mm$. El sensor detecta la presión aplicada hasta $330KPa$ y la distancia de un objeto hasta $150mm$ de distancia.

Otro aspecto importante a tener en cuenta es el modelado del entorno donde se desea trabajar con el sistema sensor ya que, como se ha apuntado anteriormente, la principal utilidad de esta combinación sensora dual es la prevención de colisiones en la interacción humano robot. La única manera que tiene este sistema de predecir la interacción con un ser humano es mediante la conductividad de su cuerpo y, dado que el

cuerpo de un ser humano no es el único objeto que presenta una gran conductividad, es necesario, modelar previamente el entorno de trabajo donde se precisa la actuación de este sensor. En esta linea Hoffmann et al. presenta en su artículo [23] un modelo de entorno que contiene información sobre objetos estáticos en un espacio de trabajo para permitir distinguir con un grado elevado de confiabilidad entre objetos estáticos, donde las consecuencias serían menores si se produjése una colisión y objetos dinámicos, que en general sería un operario, donde la colisión debe evitarse a toda costa ya que las consecuencias, simplemente, no pueden ser asumidas y no deberían producirse.

En el año de redacción de este documento (2017), la tendencia de los sensores de proximidad y táctiles continúa en la misma dirección. Se prioriza la seguridad humano robot ante todo pero también se trabaja en el campo del agarre. Ambos campos deben cumplir ciertos requisitos de rendimiento. En la detección de proximidad se busca ampliar el rango de la detección y lograr una rápida respuesta ante la detección para asegurar un tiempo de respuesta suficiente antes de que la colisión sea inevitable. En cuanto a la función de detección táctil, el sensor necesita ser rápido para mitigar el impacto de la colisión. Para cumplir con estos requisitos, Cho et al. propone un sensor de seguridad integrado donde la detección de proximidad se realiza mediante ultrasonidos y el sensor táctil es piezoeléctrico fabricado a partir de películas de PVDF y compatible con la estructura PDMS. El sensor implementado obtuvo un rango máximo de proximidad de $35cm$ con una tolerancia direccional de aproximadamente $\pm 15^{\circ}$ respecto del eje normal. El sensor táctil integrado PVDF fue capaz de detectar y paliar diversos impactos de hasta $20N$.

Cirillo et al. utiliza la tecnología optoelectrónica con este mismo fin y presenta en [25] unos resultados muy llamativos para conseguir una manipulación fina y detallada de objetos sin perder la seguridad en la interacción con el medioambiente y sobre todo con seres humanos.

Debido a su gran potencial en aplicaciones, en los últimos años también ha despertado gran interés en el mundo cientifico-tecnológico la llamada *piel electrónica* o *E-skin*. En [26], Cheng et al. presenta unos resultados bastante alentadores en esta materia en los cuales intenta buscar una progresiva solución a los dos desafíos claves que se presentan hoy en día: la elasticidad del dispositivo y la detección de presión y tensión lateral multidireccional. Cheng et al. realiza una configuración ortogonal para habilitar tanto el modo capacitivo de detección de presión como el modo resistivo para detección de deformación multidireccional de su prototipo de forma independiente. Para ello utiliza fibras preagrietadas a base de nanocables de plata como electrodos básicos para dotar a la *E-skin* de un estiramiento intrínseco y de detección de la deformación. A través de la optimización de la capa dieléctrica, la sensibilidad ha mejorado hasta un límite de detección de 1,5 Pa.

Actualmente, se estudia la entrada en escena en diferentes campos, como por ejemplo en el campo de la detección del déficit de atención de un conductor en el

mundo del automóvil como el explicado por Mühlbacher-Karrer et al. en [27], donde se utiliza un sensor táctil para medir la presión de las manos en el volante. Igualmente, Kalayeh et al. presenta un modelo de mecánica de deformación para predecir los cambios de capacidad en un elastómero capacitivo para poder realizar un modelo predictivo, validado experimentalmente, ya que la compresión de dicho tipo de elastómeros no tiene una relación lineal con el cambio de capacidad resultante debido a la separación de los electrodos y el espesor de dicha capa [28].

2.3. Técnicas de modelado de sistemas en Matlab

Xia et al. utiliza Matlab en su trabajo [21] en varias ocaciones con el fin de validar el funcionamiento del sistema. En un primer momento, se muestran escenarios típicos de interacción humano-robot como la medición de la distancia vertical de un objeto al sensor, el seguimiento del movimiento de un objeto deslizante paralelo al sensor y el reconocimiento de los perfiles de superficie de diferentes objetos como una placa de aluminio, un tubo y una esfera que serán empleados como modelos sustitutivos de palma de una mano humana, un dedo y un puño.

A partir de los datos recabados de estas simulaciones se aproximan los datos obtenidos mediante modelos de regresión predefinidos y se construye el modelo Matlab asociado mediante funciones de ajuste de curvas propias de este software. El modelo seleccionado para este caso en concreto fue un modelo de regresión exponencial de dos períodos. También se utiliza Matlab para realizar un reconocimiento del perfil de trabajo a partir de los datos medidos por el sensor [29].

En el proyecto descrito en este estudio [29], se tienen unos datos de entrenamiento del sensor de proximidad capacitivo que constan de 7 clases diferentes (Distancias), y cada clase está compuesta por 50 muestras de vectores (valores de capacidad). Se aplicó una solución de Análisis de discriminación lineal (Linear Discriminant Analysis – LDA) en Matlab. Se proporcionan las entradas (muestras de entrenamiento) y variables objetivo (clases), y la función de entrenamiento devuelve el discriminante lineal ajustado. La ventaja más relevante de los LDA de Matlab es su simplicidad, lo que hace posible realizar la clasificación dentro de la FPGA, lo que hará que el sistema de detección sea más portátil y autónomo. Sin embargo, estos resultados deben aplicarse en condiciones muy estrictas: la diferencia entre los datos de las diferentes clases debe ser suficiente para que el algoritmo proporcione resultados válidos, los valores de capacidad de los sensores deben ser pequeños y las diferencias entre las medidas incluso menores. Así, en la práctica, se obtienen resultados clasificadores no válidos cuando se aplican los resultados del experimento original.

Mühlbacher-Karrer et al. utiliza la herramienta Simulink de Matlab para crear un modelo del sistema y luego simularlo estadísticamente para obtener un patrón de muestras apto para compararlo con el modelo de regresión lineal que se obtenga de los

datos medidos en la realidad por su sistema sensor. Además, utiliza la herramienta Matlab PRTools para seleccionar las características más significativas del patrón que se desea seguir, reduciendo así la dimensionalidad de las variables y ayudando a obtener el modelo de patrón que preste una mayor funcionalidad y similitud con el sistema que se desea modelar.

2.4. Metodologías de diseño para procesado de señal usando FPGAs

En su artículo, Han et al., explica que utiliza una *FPGA* para crear todas las operaciones digitales que necesita, como es la generación de señales de exitación, realización de cálculos y comparaciones, etc. En un principio, su sistema genera una ráfaga de cinco impulsos separados por $25\mu s$, lo que corresponde a una frecuencia de $40kHz$. Luego amplifica esta señal para conseguir los $\pm 50V$ necesarios para accionar el PVDF transmisor. Al enviar esta señal por el sensor, se mide la onda sonora reflejada y se convierte en una señal de voltaje a través del receptor, donde es amplificada por un amplificador de bajo nivel de ruido (Low Noise Amplifier - LNA), y luego se compara con el nivel de umbral interno tras ser digitalizada.

Si la señal reflejada es mayor que el umbral interno, la *FPGA* genera una señal de temporización con la que el robot central calcula la distancia al objeto detectado. El tiempo de actualización de la información de proximidad es de $10ms$. La resolución de la medida de proximidad está determinada por la frecuencia de muestreo del ADC $(1MHz)$ y en dicho sistema es de $0,17mm$. El funcionamiento del sensor táctil es mucho más simple. Cuando se ejerce una presión sobre el sensor, éste se deforma y el voltaje se ve reflejado debido al cambio de la carga en la superficie. Este voltaje es amplificado por la relación de la capacidad de realimentación para producir la tensión de salida. La actualización de los valores de información táctil es, también, de 10 ms.

Tal como explica Xia et al. en [29], el objetivo del código *VHDL* es utilizar el reloj interno *FPGA* para generar señales de control apropiadas para los conmutadores de interconexión, y en particular la palabra de instrucción de control de 256 bits que determina la disposición de detección. La *FPGA* tiene 3 señales de entrada y 3 de salida. Las tres entradas son señales de reloj producidas por el oscilador integrado. Las salidas son PCLK, SCLK y SIN. SCLK es la entrada serie del reloj del chip conmutador de punto de cruce y su rango de frecuencias oscila entre $20KHz$ y $5MHz$. Como el oscilador interno *FPGA* funciona a $50MHz$, en el código *VHDL* se construye un divisor de reloj. PCLK es la señal de reloj paralela para el chip de punto de cruce. Sólo se activa después de que toda la instrucción de 256 bits ha sido cargada, y se vuelve a desactivar a los $5ms$. SIN se utiliza para habilitar un registro $R1$. La verificación del módulo de control digital propuesto fue realizado en 3 etapas:

- Se simuló el diseño *VHDL* a nivel *RTL* utilizando Modelsim.

- Se sintetizó el código, se mapeó para la *FPGA* y se optimizó a nivel de puertas volviendo nuevamente a simular con Modelsim extrayendo los retardos de tiempo del diseño desde el motor de análisis temporal del *QuartusII* y asegurando que ningún fallo o retraso afectaría al comportamiento esperado ya verificado a nivel RTL.

- Por último, se cargó el diseño en la *FPGA* y se realizaron las mediciones para comprobar que los resultados concordaban.

En el trabajo presentado en [24] por Cho et al., se explica que la *FPGA* ha sido programada para gestionar todas las funciones digitales de tiempo, cálculo y comparación requeridas. Cuando el robot central del sistema envía una señal de disparo a la placa, el chip *FPGA* genera una ráfaga de cinco impulsos separados por $25\mu s$, lo que corresponde a una frecuencia de $40KHz$. Entonces, el circuito de adaptación amplifica los pulsos a $\pm 50V$ para accionar el transmisor. La onda sonora reflejada se convierte en la señal de voltaje a través del receptor, amplificado por un amplificador de bajo nivel de ruido, se digitaliza y se compara con el nivel umbral. Si la señal del pulso reflejado es mayor que el umbral interno, la *FPGA* genera una señal de temporización con la que el robot central calcula la distancia al objeto detectado.

2.5. Metodologías de desarrollo y análisis de software empotrado

En [29] se describe el sistema sensor como la combinación de los 4 módulos siguientes:

(a) Una matriz de 4×4 que constituye el núcleo de detección del sistema,

(b) Un controlador digital que crea señales para diferentes disposiciones de los electrodos y diferentes patrones de campo eléctrico,

(c) Un convertidor analógico/digital que convierte los valores de capacidad analógicos medidos en señales digitales, y

(d) Un módulo de procesamiento y visualización de señales que indica la distancia real del obstáculo y otra información sobre su perfil físico.

La lógica necesaria para manejar la matriz formada por los electrodos de detección se ha fabricado en una *PCB* de 4 capas. A esta placa se le acoplan los electrodos de detección y otros tres circuitos integrados. Dos de ellos son interruptores que se utilizan para fijar la disposición de los electrodos y el último es un convertidor analógico/digital que convierte los valores de capacidad en señales digitales. La matriz de electrodos se ha fabricado sobre la capa superior de la *PCB* con forma cuadrada. En la segunda capa, se fabrica otra matriz de 4×4 de placas metálicas con forma cuadrada,

que actúan como electrodos de protección activos. Añadiendo una placa conectada a tierra en la tercera capa de la *PCB*, se logra la protección del sistema contra capacidades parásitas.

La estrategia de control de escaneo inteligente adoptada por Cirillo et al. en [25] y el bajo consumo de energía del módulo de detección hace posible el desarrollo del sensor distribuido prototipado en tecnología flexible PCB. La estrategia de escaneado permite una reducción sustancial del número de cables con respecto al número de *Taxels*, lo que hace posible el uso de una *PCB* flexible con un número limitado de capas que corresponden a un grosor reducido. Esto garantiza una alta conformabilidad para el panel optoelectrónico y bajo coste de producción. El diseño de la *PCB* flexible afecta a la flexibilidad máxima alcanzable por el sensor distribuido. También influye en esta propiedad la instalación de los componentes electrónicos necesarios en la *PCB* flexible, dependiendo ésta del numero y la dimensión de los componentes. Además, la flexibilidad depende también de las capas necesarias para el cableado. Los módulos de detección están constituidos únicamente por los componentes optoelectrónicos. La superficie activa del sensor es de, aproximadamente $47mm \times 47mm$.

Según Kalayeh et al., es clara la gran cantidad de beneficios que pueden proporcionar los polímeros conductores a la detección táctil ya que son fáciles de usar, suaves, conformables y robustos. No es para menos si se tiene en cuenta que gracias a la existencia de materiales con estas propiedades, ha podido crear un sensor capacitivo táctil y de proximidad en una placa de circuito impreso flexible (FPCB) como describe en [28] y que amplía sustancialmente el número de aplicaciones de este tipo de sensores. Kalayeh et al. explica también que existen trabajos actualmente donde se intenta trabajar con nanopartículas conductoras en redes pulverizadas a modo de pinturas y el uso de líquidos sintéticos a base de aleaciones que permanecen líquidos a temperatura ambiente.

2.6. Conclusiones

A modo de conclusión en cuanto a materia de sistemas sensores se refiere, se puede afirmar que en la última década los sensores de funcionamiento dual táctil y de proximidad han estado en auge entre la comunidad científica mundial y más aún en los últimos dos años. Se ha realizado un amplio estudio del estado del arte. Existen numerosos equipos científicos que realizan trabajos de investigación sobre nuevos métodos y nuevas tecnologías con el objeto de mejorar y optimizar este tipo de sensores, especialmente con la medición de la proximidad.
No se descarta la introducción en un futuro próximo de nuevas tecnologías sensoras. Se trabaja focalizando los esfuerzos en dos líneas principales:

(a) Seguridad en el entorno, lo cual implica velocidad y amplio rango de visión espacial en lo que a medición de proximidad se refiere, y

(b) la precisión y forma del agarre.

En cuanto a la utilización del software Matlab, se ha visto que su uso se dedica principalmente al modelado de sistemas tipo y variables estadísticas para la consecución de la búsqueda de modelos de regresión que se adapten al sistema que se desea construir. También es de gran utilidad para representar en una estación cliente los datos obtenidos por los sensores de manera gráfica con objeto de tomar decisiones.

Para abordar el diseño de la *FPGA*, se parte de la descripción del sistema a nivel *RTL*, simulando el comportamiento del sistema para comprobar su funcionalidad, que se cumplen las latencias esperadas y que el dimensionado del sistema se ha realizado correctamente, y finalmente volcarlo en la placa para realizar su prototipado para validar el diseño.

En materia de sistemas empotrados, los sensores táctiles y de proximidad se pueden integrar con cierta facilidad en una *PCBs*, consiguiéndose tamaños bastante reducidos en la actualidad. Por lo general se fabrica en la capa superior de la *PCB* la parte táctil del sensor y en las capas internas, la lógica de control que, no suele ser demasiado compleja.

La tendencia actual es el uso de materiales flexibles para la fabricación robusta y fiable de sensores y sistemas empotrados, que soporten el movimiento y la elasticidad. Con ello es posible abarcar una mayor posibilidad de aplicación del dispositivo y preparar el terreno para el objetivo final de fabricar una piel sintética fiable y elástica.

Todos los trabajos analizados buscan como fin último la seguridad en la colaboración en un entorno común de trabajo entre robots y seres humanos. Se ha visto que una cooperación humano-robot en la industria produce mayores beneficios al combinar la automatización y velocidad de una máquina con la supervisión y el manejo de eventos aleatorios presentada por la pericia y la experiencia de un buen empleado.

Capítulo 3

Arquitectura del sistema

3.1. Introducción

En este capítulo se presenta un estudio detallado del sistema sensor ya existente y una serie de conclusiones finales derivadas de éste. En este apartado es tremendamente importante analizar bien el sistema con el que posteriormente se va a trabajar y comprenderlo en su totalidad. Este estudio se estructurará en cuatro fases claramente diferenciadas.

(a) Estudio de la estructura del sistema,

(b) Estudio del sensor utilizado,

(c) Estudio del sistema de procesado de señal,

(d) Estudio de la transmisión y la presentación de los datos.

Este estudio precederá al núcleo de desarrollo del proyecto, donde se expondrán los detalles del trabajo llevado a cabo, el proceso para llegar a los resultados finales y las conclusiones extraídas.

3.2. Estructura del sistema

El sistema preexistente consta de tres bloques principales:

1. el sensor, que recoge los datos del entorno;

2. el bloque de procesado de señal, que se encarga de operar los datos para que éstos puedan ser interpretados debidamente; y

3. el sistema de transmisión de datos para su posterior presentación al usuario.

El estudio sobre el sistema de transmisión de los datos y su presentación al usuario será muy somero y se incluye por completitud de esta memoria, ya que está fuera del alcance de este proyecto.

Este capítulo se ha estructurado siguiendo el esquema mostrado anteriormente, tratando de incluir y de resaltar las dependencias entre las tareas realizadas para el desarrollo del proyecto.

3.3. Estudio del sensor

El sensor utilizado posee funcionalidad dual, táctil y de proximidad. Está diseñado y fabricado por el *Instituto Tecnológico de Karlsruhe* específicamente para este proyecto concreto. Esto proporciona ventajas frente a otros sensores disponibles en el mercado ya que se adapta a los requisitos del proyecto. Entre estas ventajas se puede destacar la alta integración del sensor en el sistema, ya que el procesado de datos y la electrónica disponible para ello estará completamente personalizada y dedicada para su uso en este sistema. Además, permite tener una mayor flexibilidad a la hora de integrarlo en el diseño físico final del sistema.

Como puede verse en la *Figura 3.1* el sensor utilizado en este proyecto está compuesto por dos capas *A* y *B* entre las que se encuentra una esponja o *(foam)* que conforma el material dieléctrico. Esta estructura forma un condensador y mediante la medida de las variaciones de la corriente que circula por cada una de sus capas, se puede determinar la proximidad de un objeto y la presión ejercida sobre el sensor. De esta manera, midiendo la corriente que circula por la *capa A* se podrá determinar la presión que se está ejerciendo sobre el sensor y midiendo la corriente que circula por la *capa B* es posible determinar la proximidad o no de un objeto.

Se ha elegido la parte superior del sensor para ubicar la *capa B* del sensor. Se trata de la parte más externa del sensor, lo que favorece la medida de proximidad. De esta manera, se realiza la medida de proximidad en la zona más cercana al objeto de estudio. Esta capa forma un *condensador virtual* con el entorno, de tal forma que cualquier objeto que se le acerque variará la capacidad de ese condensador y por tanto la corriente que circula por esta capa superior, haciendo posible medir su proximidad al sensor.

La *capa A* se encuentra bajo la esponja o *(foam)* y forma un *condensador real* con la capa superior. Bajo la *capa A* hay un electrodo de guarda, que es donde realmente se toma la medida de la corriente que circula en dicha capa. Bajo éste se ubica una base de toma de tierra. Esta segunda medida se basa en las variaciones de corriente que debidas a la presión se generan en el electrodo de guarda.

Con éstas dos medidas de corriente es posible determinar la presión y proximidad,

datos necesarios sobre los que es posible actuar para determinar el estado del sistema.

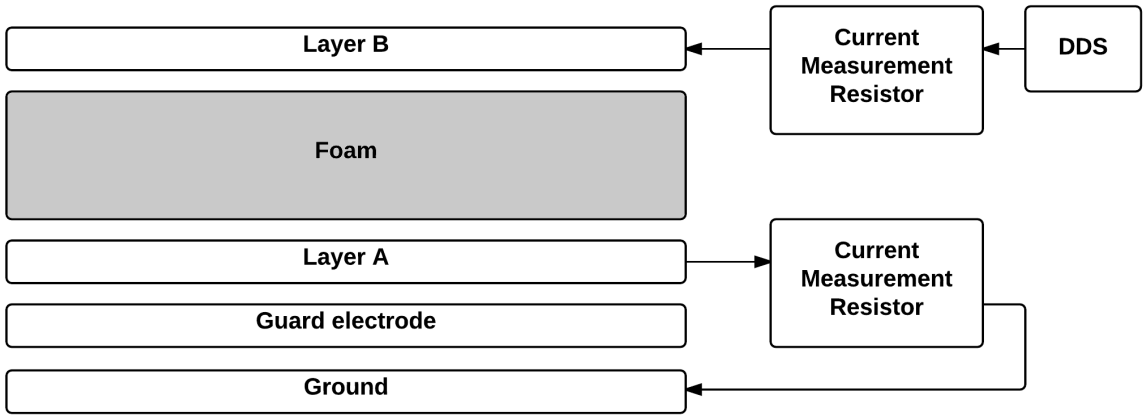


Figura 3.1: Diagrama de las capas estructurales del sensor.

Para ambas funcionalidades se utiliza el mismo principio de detección capacitivo. La parte táctil del sensor consta de un conjunto de sensores táctiles llamados *Taxels* que permite recoger perfiles de presión de la zona de contacto.

Este sensor tiene la capacidad de detectar objetos en su entorno cercano antes de que ocurra el contacto físico siendo esta su principal ventaja. Este avance es posible modificando un sensor táctil normal para que sea capaz de medir las variaciones de corriente que generan en su entorno con los objetos que se encuentran fuera de su alcance físico. Midiendo estas corrientes y conociendo la tensión de excitación se puede calcular la capacidad acoplada a la superficie del sensor y, utilizando varios sensores, la variación de ésta a lo largo del espacio.

Igualmente, el sistema de sensores también incluye la medida del campo eléctrico para ser utilizado en la detección de objetos. Ofrece una modalidad táctil al controlar el agarre durante la manipulación. El sensor utiliza el mismo efecto de campo para percibir la proximidad y la información táctil por lo que no hay necesidad de utilizar diferentes sensores para medir los datos de los múltiples modos. El uso de los dos tipos de sensores en una misma pinza o mano robótica, unido al efecto de campo eléctrico creado por el uso de *arrays* de varios sensores, hace innecesario el uso de cámaras, disminuyendo su coste y aumentando las posibilidades de uso. No obstante, el sistema se puede complementar con el uso de cámaras para funcionalidades de visión o cualquier otro sistema de sensores, con el consiguiente aumento de las capacidades del sistema.

El sensor proporciona **tres tipos de funcionamiento**:

(a) Detección **táctil**,

(b) Detección de **proximidad en modo envío**, y

(c) Detección de **proximidad en modo recepción**.

Para todos los tipos de funcionamiento se utiliza una tensión de excitación en forma de senoide con una **frecuencia (F)** aproximada de **100 KHz**. En el modo de detección **táctil**, el sensor mide las corrientes (i_{Tact}) en una serie de condensadores deformables. En la detección de **proximidad en modo envío**, el sensor envía una corriente (i_{Prox}) fuera de su superficie causada por el acoplamiento capacitivo a su entorno. Por último, en la detección de **proximidad en modo recepción** son necesarios, al menos, dos sensores para poder realizar una medida. Uno de ellos debe estar en **modo envío** y el otro en **modo recepción**. En la detección de **proximidad en modo recepción**, el sensor recoge las corrientes $(i_{Prox,r})$ que se acoplan de forma capacitiva desde el sensor de envío al sensor de recepción.

Estos tipos de funcionamiento pueden ser seleccionados por el ordenador de control para detectar la información que se desea. Para este propósito, se ha desarrollado un circuito analógico configurable que posibilita el cambio entre los diferentes tipos.

La información táctil necesita ser medida por un circuito analógico adicional. Las partes principales del procesamiento de la señal se llevan a cabo en el dominio digital por su flexibilidad. Incluso la tensión de excitación en forma de coseno que se utiliza para alimentar las placas se genera utilizando técnicas digitales.

3.3.1. Diseño mecánico del sensor

Como bien puede observarse en la *Figura 3.2*, el sensor está diseñado principalmente en tres capas.

La capa superior del sensor, en la detección de **proximidad en modo envío** y en la detección **táctil**, sirve como un electrodo de envío de corriente. En la detección de **proximidad en modo recepción** este electrodo recibe la corriente que se envía desde otro sensor que esté en **modo de envío**. Este electrodo está hecho de caucho de silicona conductor *(Elastosil LR 3162)* haciendo que la capa superior sea flexible y deformable, pero a la vez buena conductora de la electricidad para que el sistema no pierda sus propiedades al cubrirlo con ella. En el *Anexo E* de la página 139 se muestra en detalle las especificaciones del material utilizado.

La capa inferior está compuesta por una placa de circuito impreso *(PCB)* multi-capa. Este circuito impreso sirve como soporte para una matriz de 4 x 4 sensores táctiles en la parte superior. En la parte inferior se encuentra la electrónica analógica necesaria para la excitación, adaptación de la señal y realizar su control. La capa conductora entre el caucho y la *PCB* está hecha de una espuma de polímero que funciona principalmente como un resorte elástico. Cuando se aplica una fuerza a la superficie del

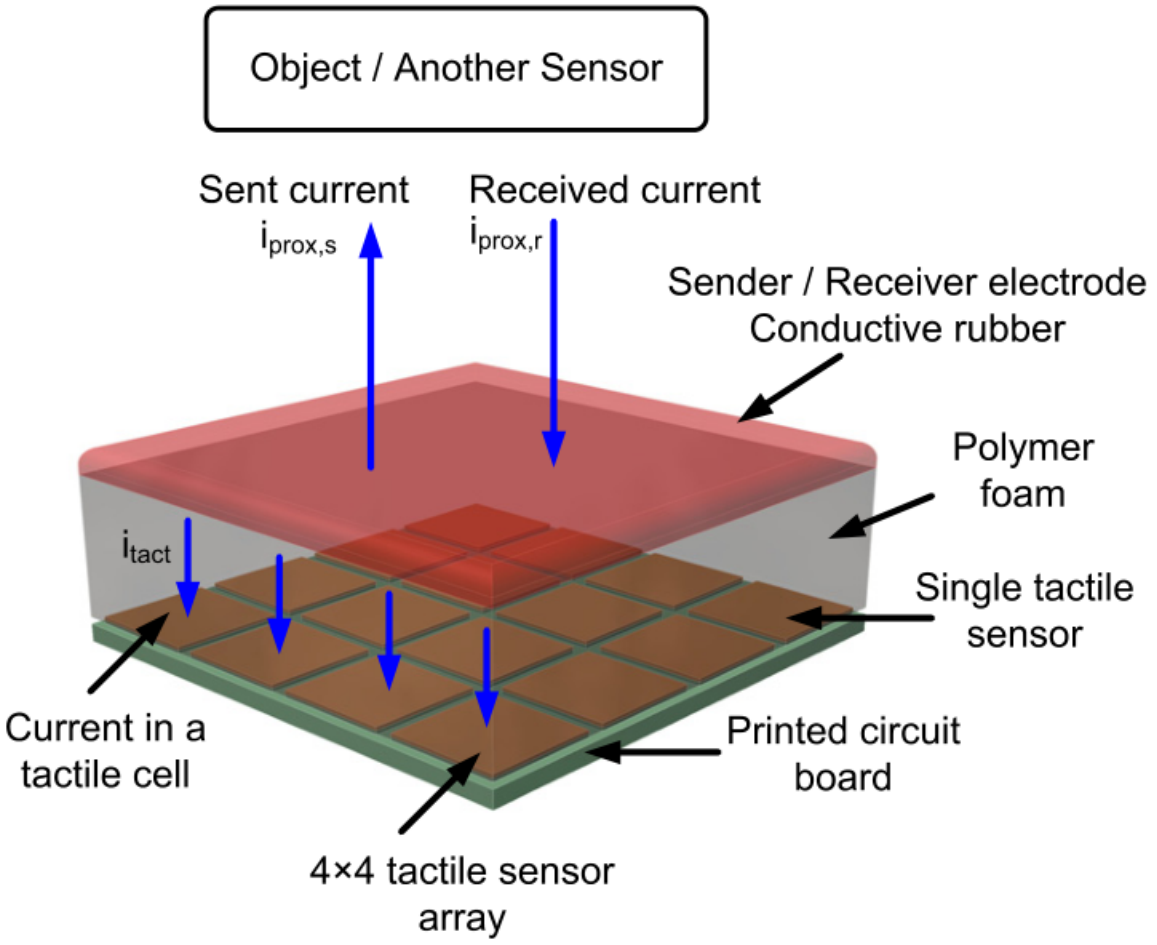


Figura 3.2: Diseño mecánico del sensor.

sensor, la espuma se comprime. Como resultado de esta deformación, la distancia (d_0) entre el caucho conductor y los electrodos de las células táctiles disminuye causando un aumento de la capacidad (C_{tot}) entre la capa superior y el electrodo del sensor táctil.

Las espumas de polímeros celulares son multifase ya que consisten en una fase de polímero y una fase fluida. El fluido utilizado en la espuma es un gas. Para un modelo simple de la fase del polímero, que se puede ver en la *Figura 3.3* se puede pensar en éste como un bloque incompresible con la permitividad relativa del polímero utilizado. Debido a esta aproximación, la capacidad del condensador (C_P) es la máxima que se puede conseguir en cada *Taxel* cuando éste está comprimido al máximo. El gas, que en este caso será aire, constituye otro condensador con capacidad (C_A) y permitividad relativa la unidad. El comportamiento mecánico del condensador está influenciado únicamente por las propiedades mecánicas de éste y la distribución del gas (tamaño y forma de las células del gas y si las células están cerradas o abiertas).

La aplicación de una carga (σ_p) en este sensor táctil produce una compresión de la

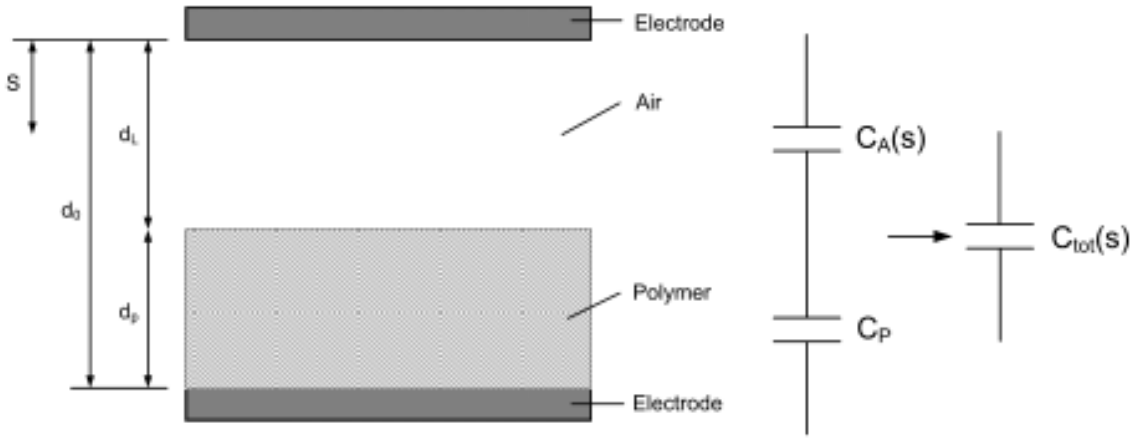


Figura 3.3: Capacidad de un condensador con dos dieléctricos.

espuma que cambia la geometría del condensador lleno de aire. La relación entre la carga aplicada (σ_p) y la capacidad (C_{tot}) es no lineal. Esta relación depende del tiempo *(t)* y de la temperatura *(T)*, así como de la compresión y la profundidad de penetración *(s)*, siendo ésta última una función no lineal. De acuerdo con ello, el condensador (C_A) se convierte en $C_A(\sigma_p, t, T)$. Como la conexión resultante de estos condensadores es una estructura en serie de ambos, el resultado se muestra en la *Ecuación 3.1*.

$$C_{tot}(\sigma) \approx C_{tot}(\sigma_p, t, T) = \frac{C_{tot}(\sigma_p, t, T) + C_A}{C_{tot}(\sigma_p, t, T) \cdot C_A} \tag{3.1}$$

Despreciando la influencia del tiempo y la temperatura, y asumiendo que la capacidad de un condensador de placas es $C = \epsilon_0 \cdot \epsilon_r \frac{A}{d}$, la capacidad total se puede calcular como se muestra en la *Ecuación 3.2*.

$$C_{tot}(\sigma) = \frac{\epsilon_0 \cdot \epsilon_P \cdot \epsilon_A \cdot A}{\epsilon_A \cdot d_P + \epsilon_P \cdot (d_0 - d_P - s)} \tag{3.2}$$

3.3.2. Detección táctil

La *Figura 3.4* muestra el comportamiento de un *Taxel* hecho con una esponja de polímero. En este gráfico se muestra un ciclo de carga de una esponja de polímero, como la utilizada en este proyecto.

La esponja se comprime cuando hay una cierta presión mayor que la que existe cuando no existe carga. El sensor exhibe un comportamiento no lineal. Debido a este comportamiento no lineal, el sensor no es adecuado para una carga en la célula ya que no se ha dotado de un modelo de sensor que sea suficiente para tener en cuenta esta no linealidad.

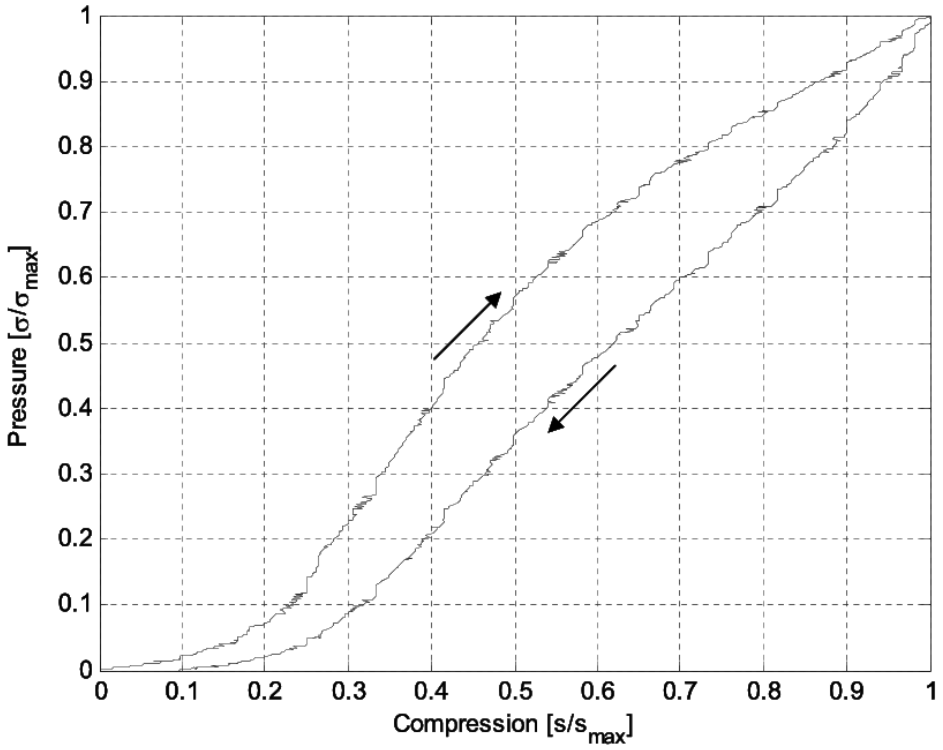


Figura 3.4: Ciclo de carga de una esponja de polímero.

El segundo experimento muestra el comportamiento del sensor táctil con el tiempo. Se aplica una carga y se mide la capacidad resultante. Como se muestra en la *Figura 3.5*, la capacidad aumenta cuando se aplica una carga.

Cuando la carga se mantiene al mismo nivel, la capacidad varía. Este cambio es causado por la dependencia progresiva del efecto del tiempo sobre la esponja de polímero.

3.3.3. Detección de proximidad

Para evaluar la detección de proximidad, se han realizado dos experimentos.

En el primero, se ha enfrentado un sensor en detección de **proximidad en modo envío** y otro en detección de **proximidad en modo recepción** de la misma manera, con un objeto conectado a tierra. En la *Figura 3.6* puede verse el resultado de este experimento. Como se espera, la magnitud de la señal de salida aumenta cuando el sensor se aproxima al objeto conectado a tierra.

En la detección de **proximidad en modo recepción** se ha llevado a cabo un experimento muy similar. De nuevo se han utilizado dos sensores, uno en **modo recepción** y otro en **modo envío**. Aproximando ambos sensores se puede obtener una señal en **modo recepción**. En la *Figura 3.7* se puede ver el resultado de este experimento. De

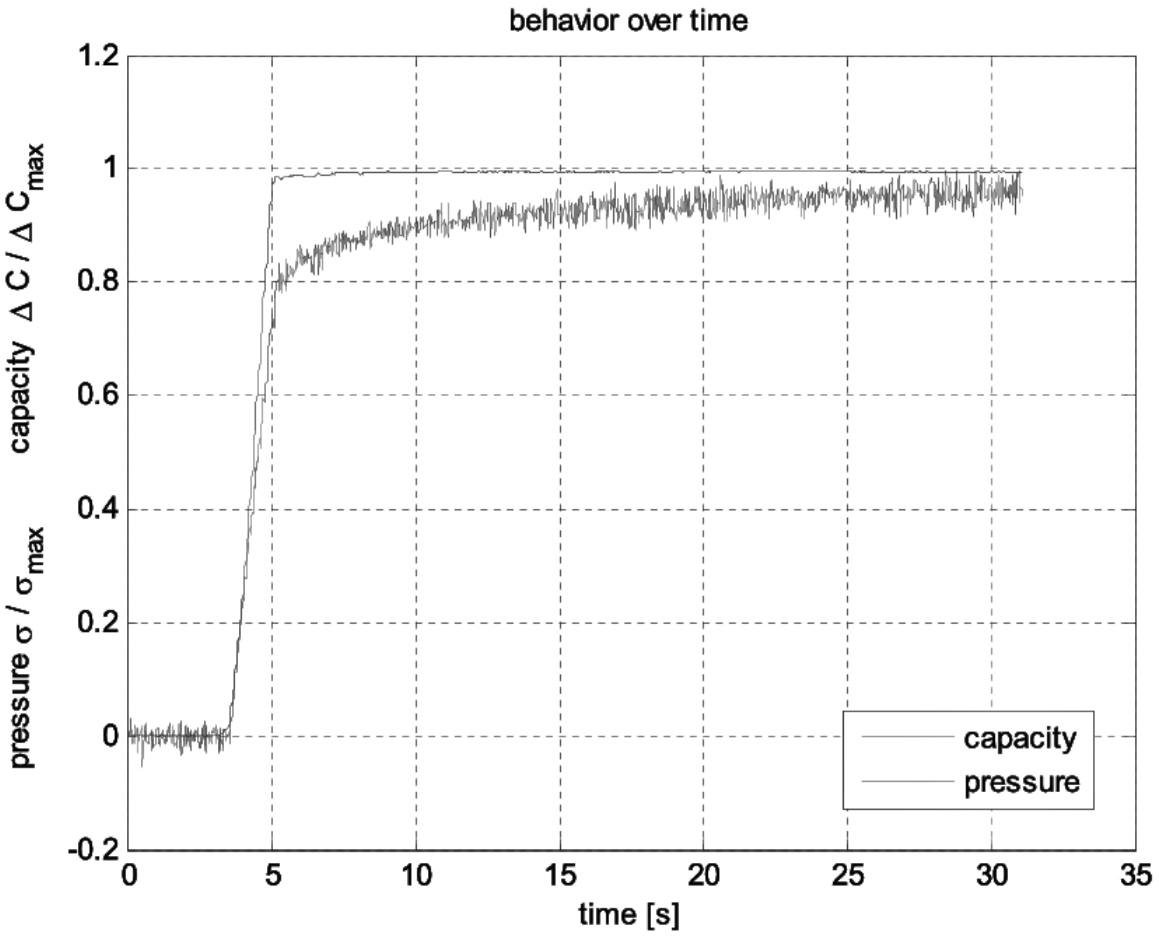


Figura 3.5: Cambio en la capacidad cuando se aplica una carga.

la misma manera que anteriormente, la magnitud aumenta a medida que los sensores se van aproximando cada uno al otro.

3.3.4. Diseño Eléctrico del sensor

En la *Figura 3.8* mostrada a continuación, se puede ver el esquema del circuito eléctrico que se encarga de medir las variaciones de corriente que se producen en cada sensor. Se puede observar la dualidad que presenta el sistema a la hora de medir **presión** *(tactilidad)* y **capacidad** *(proximidad)*.

Este circuito está construido sobre una *PCB* de múltiples capas e integrado bajo la *capa A*, quedando en la parte trasera del sensor. La primera capa se ha estructurado como un sensor táctil y la segunda capa como un electrodo de blindaje activo conmutable. En la *Figura 3.1*, vista anteriormente en esta memoria, se puede ver la pila de capas *PCB* que han sido utilizadas para el diseño de este sensor.

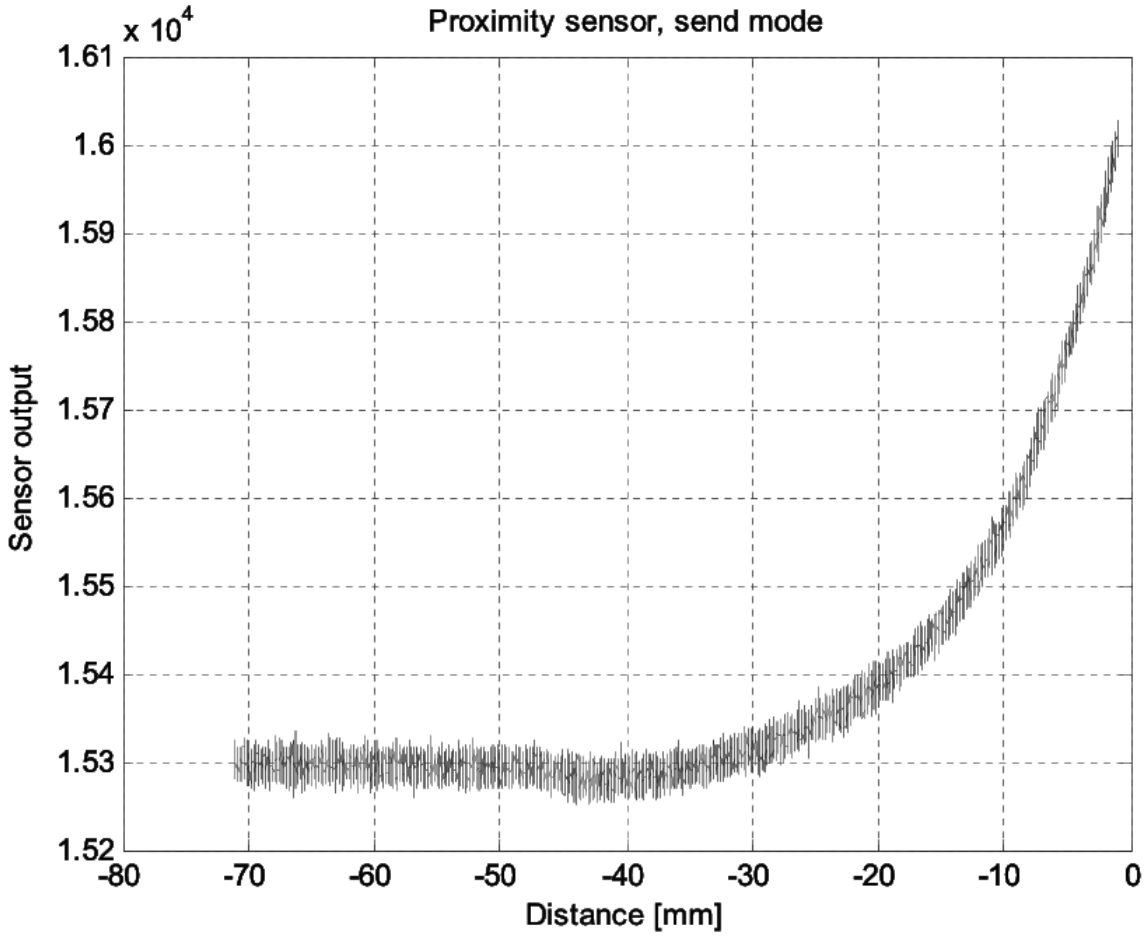


Figura 3.6: Sensor de proximidad en modo envío.

Como puede verse en la *Figura 3.8* existen dos medidas de corriente. Una en la placa superior del sensor (*Capa B Figura 3.1*), la parte más cercana al objeto que se quiere tocar; y otra en la placa base del sensor (*Capa A Figura 3.1*).

La primera de estas medidas, proporciona información sobre la proximidad del objeto, ya que las variaciones de corriente que circulen por ella dependerán de la capacidad de los objetos que se muevan en su entorno y de la cercanía de estos. La segunda medida, va a depender de cuanta fuerza se ejerza sobre la *placa B*, ya que esto variará la distancia relativa entre ambas placas y por tanto se podrá medir en la *placa A* la presión con la que se toca o agarra un determinado objeto.

Es importante notar que la medida de proximidad estará bastante ligada a la naturaleza de los objetos que se acerquen. De esta manera, **no** se obtendrá el mismo valor de corriente para un objeto de madera o plástico que para uno de aluminio o hierro. Como puede intuirse, esto **no** afecta a la medida de la parte táctil del sistema.

El funcionamiento de los interruptores que se puede ver en la *Figura 3.8* es el

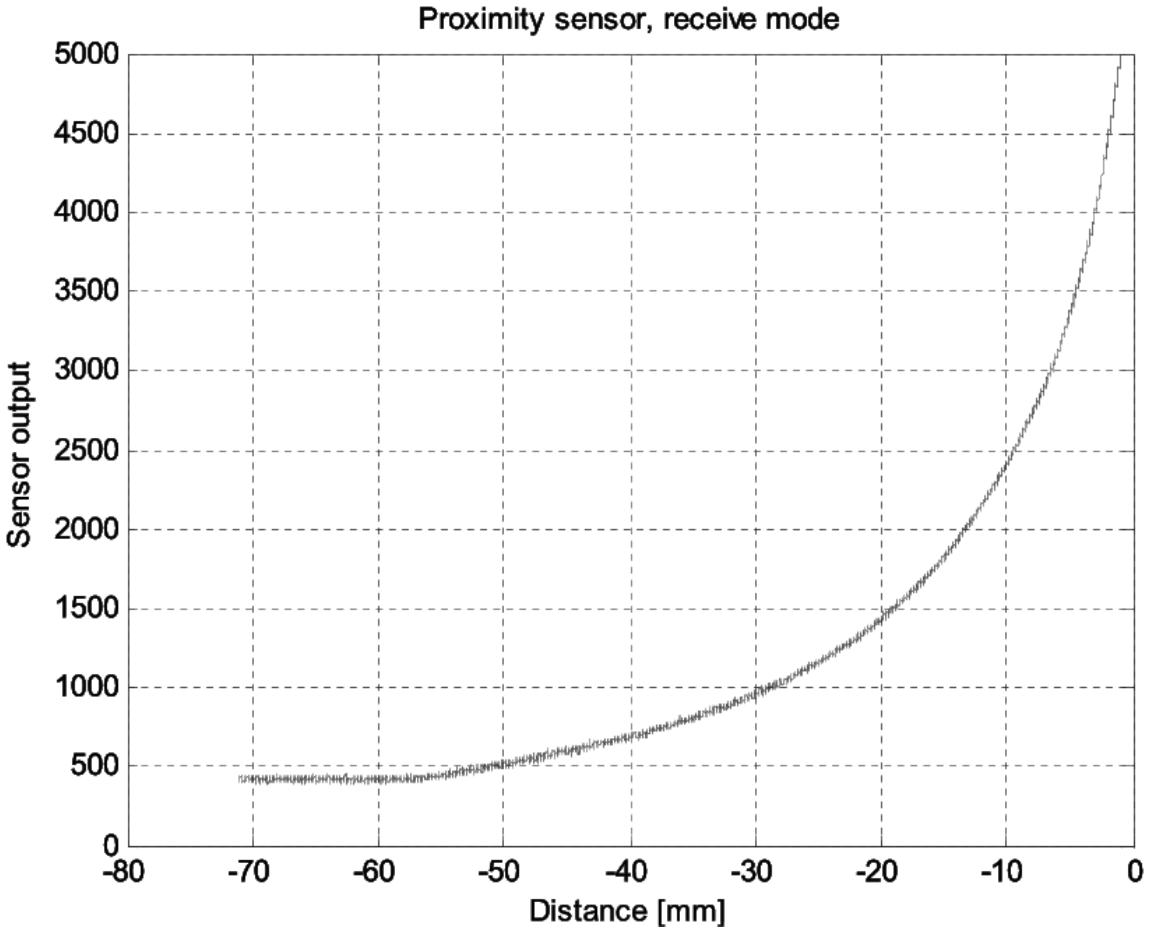


Figura 3.7: Sensor de proximidad en modo recepción.

siguiente:

- En la detección **táctil**, el interruptor 2 se mantiene en posición (d) obligando al *electrodo de protección* a mantenerse al potencial de tierra. Esto mantiene los electrodos de los sensores táctiles, que están acoplados capacitivamente al electrodo de protección, casi a potencial cero. Debido a que todos estos electrodos comparten la misma geometría y se encuentran a la misma distancia de la capa apantallada de puesta a tierra, el campo eléctrico que generan es relativamente homogéneo. En este caso todo el sensor táctil puede ser considerado como un condensador de placas plano-paralelas. Un sólo sensor está dotado de 16 *Taxel*s de dimensiones $1cmx1cm$. La capacidad de un *Taxel* (C_{ov}) es de aproximadamente 1/16 de la capacidad total del sensor (C_{tot}) cuando éste está descargado. En ambos modos de proximidad esta capacidad es indeseada.

- En la detección de **proximidad en modo de envío**, un electrodo de protección o un *Taxel* conectado a tierra se traduciría en una caída significativa en el rendimiento ya que la mayor parte de la corriente de excitación enviada sería redirigida a tierra. Esto se traduciría en una corriente de envío medida mucho mayor que

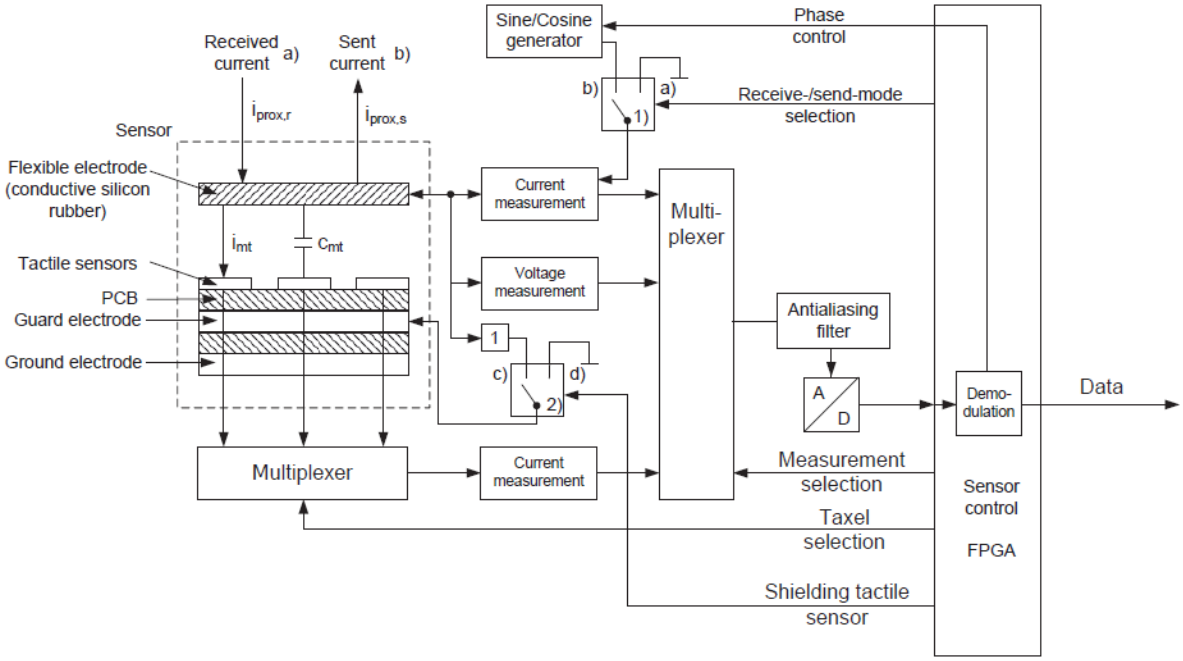


Figura 3.8: Estructura hardware de la parte eléctrica del sensor.

la corriente que está acoplada capacitivamente al medio ambiente. Para evitar esto, el electrodo de protección se puede conectar al potencial de la parte superior, al electrodo de recepción/envío. En este caso, se situará el conmutador 2 en la posición (c). Ahora, el electrodo de protección se lleva a cabo a través de un amplificador con el factor de amplificación 1 para el electrodo superior y se mantiene a la misma magnitud y fase. En este caso no circula corriente a través de (C_{ov}) porque no hay diferencia de potencial entre las placas del condensador. Esta técnica hace que el sensor de proximidad sea mucho más sensible, debido a que la capacidad de carga de los sensores táctiles puede ser eliminada. En los modos de trabajo *proximidad-envío* y *proximidad-recepción* se envía o recibe, respectivamente, una corriente a través del electrodo de silicona conductora y se utiliza un amplificador diferencial para medir la caída de tensión en la resistencia de medición.

En la *Figura 3.9*, puede verse el circuito equivalente a cada uno de los modos de trabajo en cuanto a la detección de proximidad se refiere.

El modo de trabajo es determinado por el interruptor 1, como puede verse en la *Figura 3.8*. Cuando el interruptor 1 está en la posición a) el sensor está en **modo de recepción**. En este modo, el circuito de medición está configurado como se muestra en la *Figura 3.9(a)* y la corriente medida es i_{mp}. Esta corriente medida está acoplada capacitivamente al sensor (en este modo es necesario un segundo sensor en modo de envío). El cambio de este interruptor a la posición b) hace que el sensor se convierta

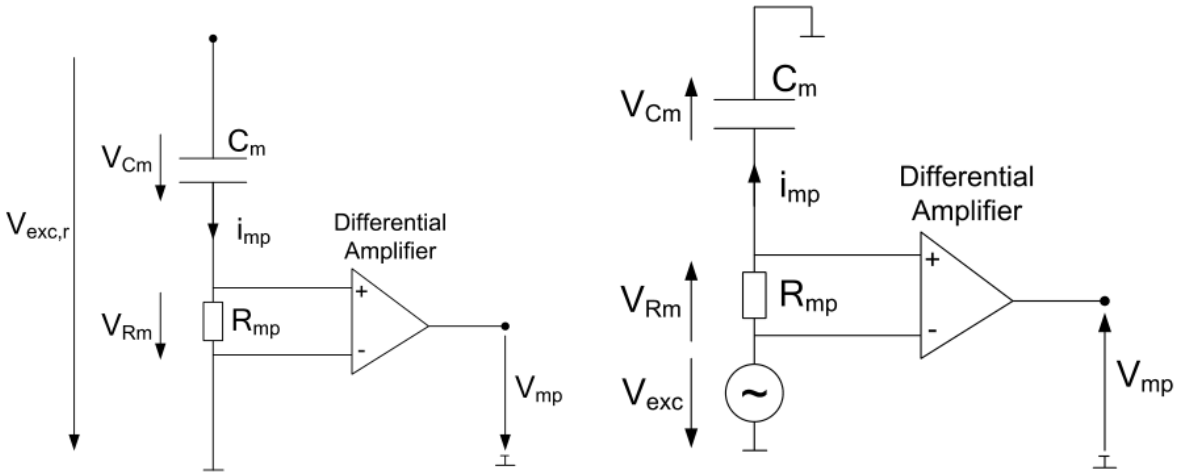


Figura 3.9: Circuito equivalente para la detección de proximidad en modo de recepción (a) y envío (b).

en emisor como se muestra en la *Figura 3.9(b)* y pase a trabajar en **modo de envío**. La corriente i_{mp} se convierte ahora en la corriente enviada y es medida por el mismo amplificador diferencial. En ambos modos el interruptor 2 tiene que estar en la posición c).

En la detección **táctil**, el interruptor 2 debe estar en la posición d) que conecta el electrodo de protección a tierra. Para esta medición se utiliza un amplificador no inversor que mide la caída de tensión en la resistencia de medición (R_{mt}) como puede verse en la *Figura 3.10*.

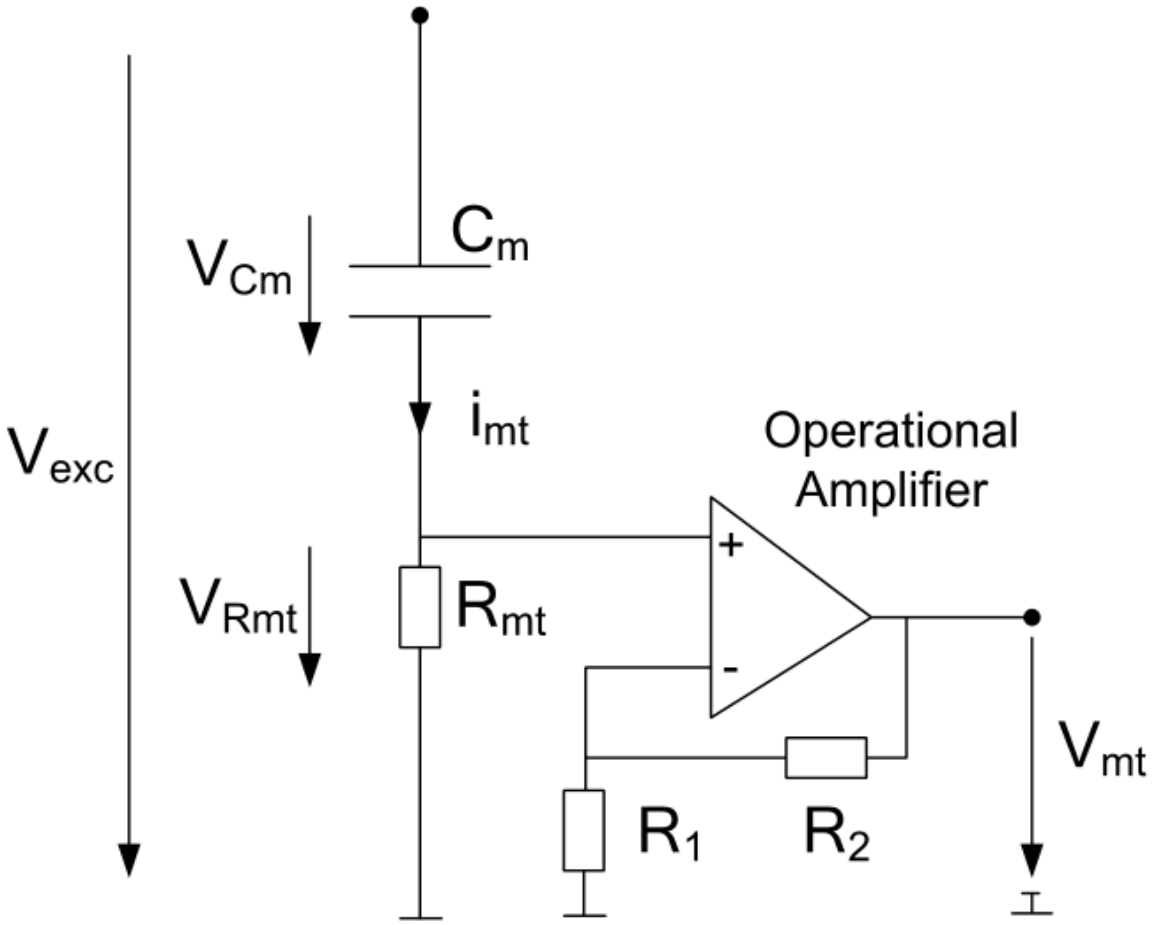


Figura 3.10: Circuito equivalente para la detección táctil.

3.4. Estudio del sistema de procesado de señal

Durante el procesado de la señal de este sistema, las medidas de corriente realizadas por cada sensor, pasan por diferentes bloques que se encargan de modificar la señal con el objetivo de poder cuantificarla. Para ello se realizan varias operaciones: digitalización, filtrado, ponderación, etc. Las señales que se recibirán serán cosenos con frecuencia de excitación (f_{exc}) igual a 100 KHz.

Las variaciones de las capacidades modulan la señal de amplitud (Am) recibida. La señal medida (s_{meas}) será de la forma $s_{meas} = A_m \cdot cos(2\pi f_{exc} t + \varphi)$. El procesamiento digital para demodular la señal (s_{meas}) se representa en la *Figura 3.11*.

En el dominio analógico, las señales de entrada (corriente recibida, corriente enviada, corriente que atraviesa un *Taxel* y tensión de excitación) se introducen en el circuito analógico de recepción y multiplexado. Después del muestreo, las señales alimentan a un filtro paso-banda *FIR* con una atenuación mayor a 100 dB en la banda atenuada y un ancho de banda de 2 kHz. Este filtro atenúa las frecuencias no deseadas y per-

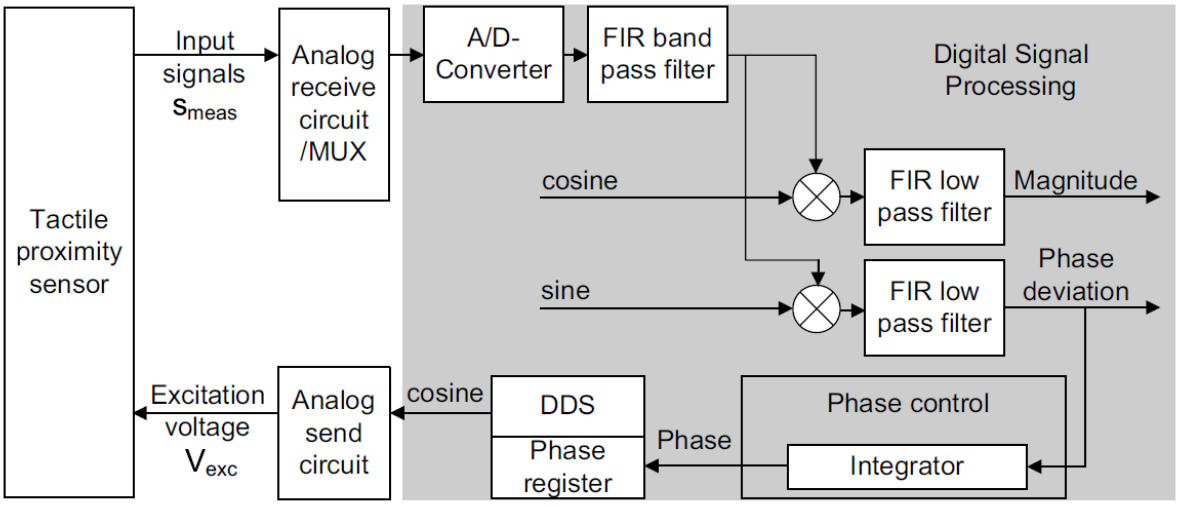


Figura 3.11: Esquema del procesado digital de la señal.

mite el paso de la señal coseno deseada para que su medida. Tras el filtrado, la señal medida será multiplicada con una función seno y una función coseno de $\omega = 2\pi \cdot f_{exc}$, como se muestra en las *Ecuaciones 3.3 y 3.4*:

$$s_M = A_m \cdot cos(\omega t + \varphi) \cdot cos(\omega t) = \frac{A_m(cos(\varphi) + cos(2\omega t + \varphi))}{2} \tag{3.3}$$

$$s_P = A_m \cdot cos(\omega t + \varphi) \cdot sin(\omega t) = \frac{A_m(sin(-\varphi) + sin(2\omega t + \varphi))}{2} \tag{3.4}$$

Después del filtrado paso bajo, solamente la baja frecuencia $s_{Ml} = \frac{1}{2}A_m \cdot cos(\varphi)$ ("magnitud") y $s_{Pl} = \frac{1}{2}A_m \cdot sin(-\varphi)$ ("desviación de fase") se mantiene. Como puede verse, la señal de salida s_{Ml} es una función de la diferencia de fase entre la fase de la señal a demodular y la fase del reloj del convertidor analógico/digital *(ADC)*. Sólo cuando φ es 0 se puede lograr el máximo de la señal demodulada. Debido a esta circunstancia, para reducir al mínimo la diferencia de fase φ, se ha añadido un controlador de fase. Un circuito integrador permite realizar el cálculo de la desviación de fase s_{Pl} y cambia la fase de la tensión de excitación (V_{exc}) hasta que φ se convierte en 0. En este punto, la salida en magnitud se es $s_{Ml} = \frac{1}{2}A_m \cdot cos(0) = \frac{A_m}{2}$.

La *Figura 3.12* muestra el proceso de compensación de fase durante el inicio de una fase aleatoria. La amplitud de la señal de modulación se asume $A_m = 1$ y sin ruido. La desviación de fase (*Phase deviation*) muestra la entrada al control de fase y disminuye con el tiempo. Con el decremento de la diferencia de fase, la magnitud de la señal demodulada es cada vez mayor hasta que el error de magnitud (*Magnitude error*) y la desviación de fase se vuelven aproximadamente cero.

La frecuencia de muestreo del convertidor analógico/digital *(ADC)* (f_s) es 4 veces mayor que la frecuencia de la tensión de excitación (f_{exc}), por lo que cada período

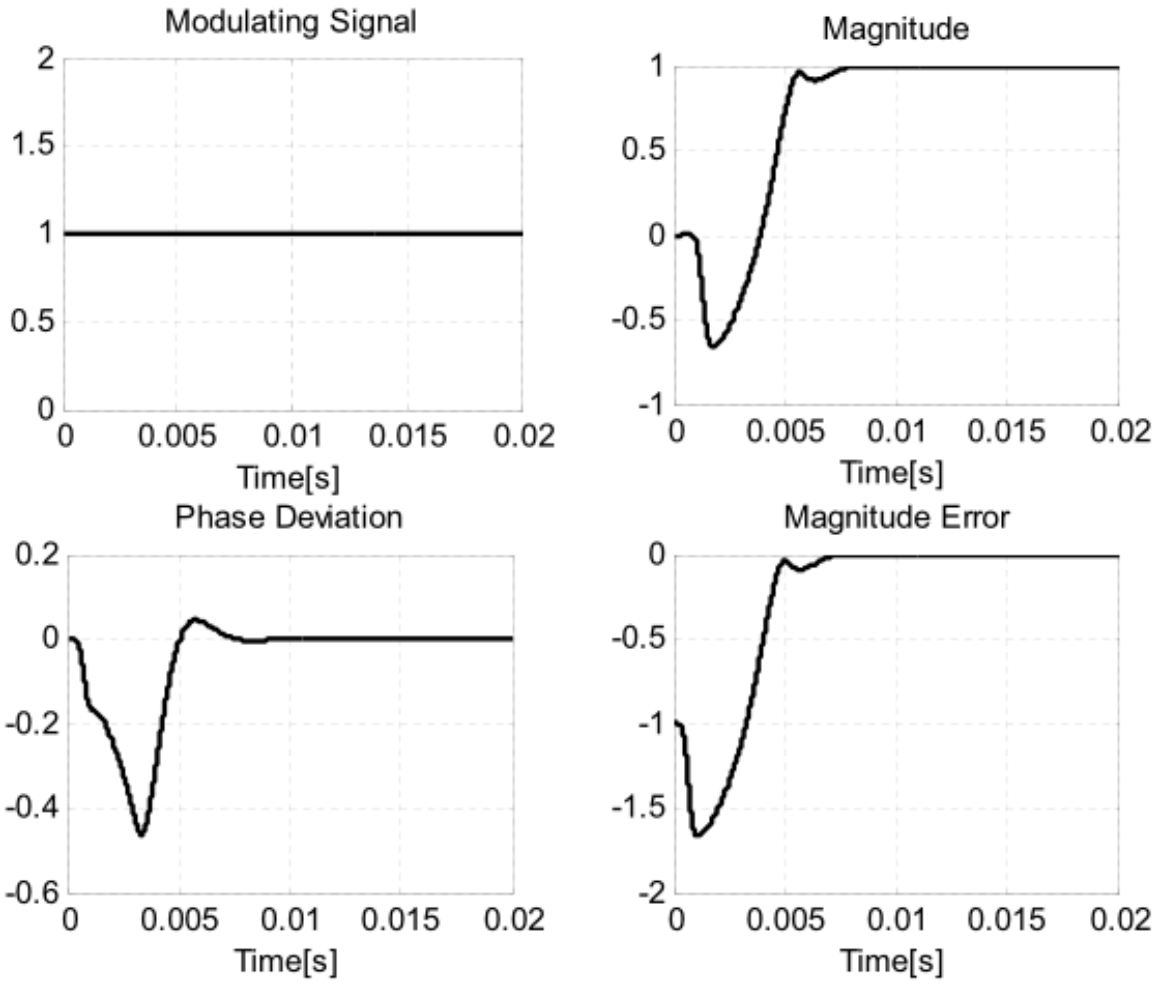


Figura 3.12: Sincronización de fase.

de la señal de entrada es **muestreada 4 veces**. De acuerdo con el seno y el coseno la forma de onda que se multiplica con la señal de entrada necesita ser muestreada en 4 puntos diferentes. Para simplificar el algoritmo, se ha elegido [1, 0, -1, 0] para el coseno y [0, 1, 0, -1] para el seno. Debido a que uno de cada dos valores de s_{Ml} es cero, la señal se puede diezmar por 2, lo que reduce la frecuencia de muestreo de la señal a la mitad. De acuerdo a la naturaleza tan sensible del sensor y el uso previsto en un entorno ruidoso, el demodulador debe ser capaz para suprimir las señales no deseadas de (*ruido*). En la *Figura 3.13* se muestra esta capacidad.

Una señal modulada con amplitud $A_m = 1 + 0,0005 \cdot sin(\omega_m t + \varphi_m)$ alimenta al demodulador. A esta señal se le añade un ruido adicional (*Received Noisy Signal*) que viene del entorno en el que se encuentra el sistema. En este ejemplo la amplitud de las señales interferentes es aproximadamente 7600 veces mayor que la señal necesitada. Magnitud (*Magnitude*) muestra la señal demodulada. En este ejemplo, el error de magnitud absoluto, es menor de $4 \cdot 10^{-5}$.

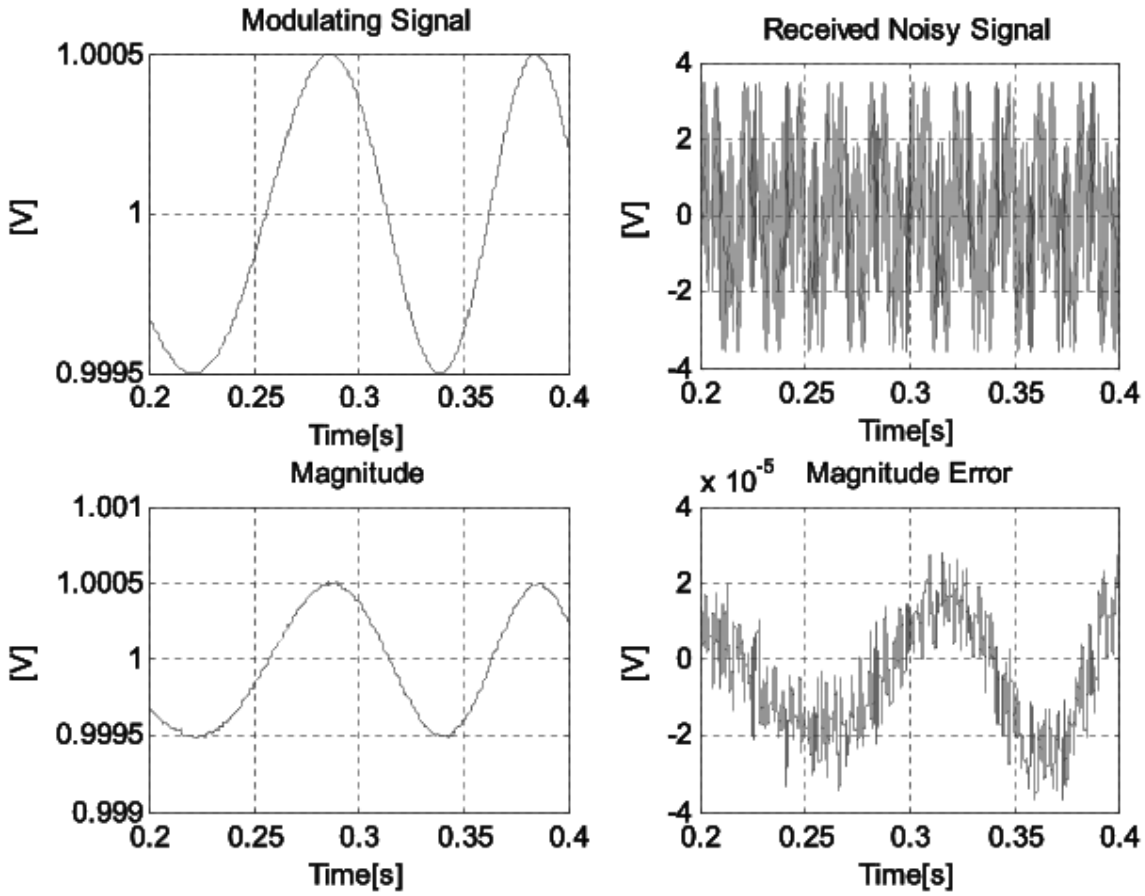


Figura 3.13: Eliminación del ruido en el dominio temporal.

3.5. Estudio de la transmisión y la presentación de los datos

Los datos son transmitidos a un equipo cliente a través de una conexión *Ethernet* mediante el protocolo de transmisión *UDP*. A continuación se describe el funcionamiento del protocolo y el formato de los paquetes de datos utilizado.

3.5.1. Uso de *UDP* sobre *Ethernet*

UDP es un protocolo de nivel de transporte orientado al envío de mensajes que está documentado en el *RFC 768* de la *IETF*; el cúal se encuentra también en el *Anexo F* de la página 141 de este documento.

Este protocolo se basa en el intercambio de datagramas que se envían a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. No tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros. Tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción. Su uso principal es para protocolos como *DHCP*, *BOOTP*, *DNS* y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores que la información transmitida y no es rentable introducir información de seguridad, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo.

Este protocolo supone que el Protocolo de Internet *(IP)* se utiliza como protocolo subyacente. *UDP* proporciona un procedimiento para enviar mensajes desde programas de aplicación a otros programas con un mínimo mecanismo de protocolo. Está orientado a la transacción, y la entrega de paquetes pero su protección y entrega no están garantizados. Las aplicaciones que requieren una entrega fiable y ordenada de flujos de datos deben utilizar el *Protocolo de Control de Transmisión (TCP)*.

En la familia de protocolos de Internet, *UDP* proporciona una interfaz entre la capa de red y la capa de aplicación. No otorga garantías para la entrega de sus mensajes, por lo que realmente no se debería encontrar en la capa 4, y el origen transmisor no retiene estados de los mensajes que han sido enviados a la red. *UDP* sólo añade multiplexado de aplicación y suma de verificación *Checksum* de la cabecera y la carga útil o *payload*. La comprobación de la transmisión de la información deben ser implementadas en capas superiores.

En la *Figura 3.14* se muestra la estructura de un paquete *UDP* preparado para ser enviado. La cabecera *UDP* consta de 4 campos de los cuales 2 son opcionales (con fondo rojo en la tabla). Los campos de los puertos fuente y destino son campos de 16 bits que identifican el proceso de origen y recepción. Ya que *UDP* carece de un servidor

de estado y el origen *UDP* no solicita respuestas, el puerto origen es opcional. En caso de no ser utilizado, el puerto origen debe ser puesto a cero. A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama *UDP* incluidos los datos. El valor mínimo del tamaño del datagrama es de 8 bytes.

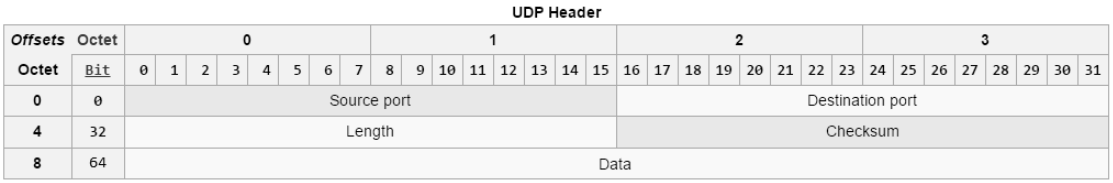


Figura 3.14: Estrucutura de paquete del protocolo *UDP*.

UDP utiliza puertos para permitir la comunicación entre aplicaciones. El campo de puerto tiene una longitud de 16 bits, por lo que el rango de valores válidos va de 0 a 65.535. El puerto 0 está reservado pero, como se ha visto antes, es un valor permitido como puerto origen si el proceso emisor no espera recibir mensajes como respuesta. Los puertos 1 a 1023 se llaman puertos “conocidos” y en sistemas operativos tipo Unix enlazar con uno de estos puertos requiere acceso como superusuario. Los puertos 1.024 a 49.151 son puertos registrados. Los puertos 49.152 a 65.535 son puertos efímeros y son utilizados como puertos temporales, sobre todo por los clientes al comunicarse con los servidores.

La mayoría de las aplicaciones claves de *Internet* utilizan el protocolo *UDP*, incluyendo el *Sistema de Nombres de Dominio (DNS)*, donde las consultas deben ser rápidas y solo contarán de una única solicitud, el *Protocolo de Administración de Red (ARP)*, el *Protocolo de Información de Enrutamiento (RIP)* y *el Protocolo de Configuración Dinámica de Host (DHCP)*.

Las características principales de este protocolo son:

(a) Trabaja sin conexión, es decir que no emplea ninguna sincronización entre el origen y el destino.

(b) Trabaja con paquetes o datagramas enteros, no con bytes individuales como *TCP*. Una aplicación que emplea el protocolo *UDP* intercambia información en forma de bloques de bytes, de manera que por cada bloque de bytes enviado de la capa de aplicación a la capa de transporte, se envía un paquete *UDP*.

(c) No es fiable. No emplea control de flujo ni ordena los paquetes.

(d) Su gran ventaja es que provoca poca carga adicional en la red ya que es sencillo y emplea cabeceras muy simples.

En la *Figura 3.15* se muestra la estructura de un paquete *UDP* sobre *Ethernet* con *IPv4*. Los campos con fondo en rojo son los pertenecientes al protocolo *IPv4* y son, una suma de comprobación *(Checksum)* de 16 bits que abarca una pseudo-cabecera *IP* con las *IPaddress* origen y destino, el protocolo y la longitud del paquete *UDP*. A continuación se encuentra la cabecera *UDP* y los datos que pueden abarcar desde cero bytes hasta completar un múltiplo de 16 bytes. El *Checksum* es opcional en *IPv4*, aunque generalmente se utiliza en la práctica (en *IPv6* su uso es obligatorio). Igualmente existe otra estructura alternativa de paquete *UDP* sobre *IPv6* no usada en este proyecto.

IPv4 Pseudo Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source IPv4 Address																															
4	32	Destination IPv4 Address																															
8	64	Zeroes								Protocol								UDP Length															
12	96	Source Port																Destination Port															
16	128	Length																Checksum															
20	160+	Data																															

Figura 3.15: Estrucutura de paquete del protocolo *UDP* sobre *IPv4*.

3.5.2. Estructura del paquete de datos

A continuación se detalla la estructura del paquete de información enviada por la red mediante una conexión *Ethernet* hasta el equipo cliente. En la *Figura 3.16* se muestra una captura realizada desde el software *Wireshark*, donde se ha filtrado todos los paquetes que son enviados a través de la red en estudio. En este caso se trata de la red que envía los datos desde la *FPGA* hasta el equipo cliente encargado de interpretarlos. Indicar que la captura de los paquetes mostrados han sido capturados durante el funcionamiento del sistema, esto es, no se han capturado los errores de paquetes u otra información relativa a la conexión/desconexión o configuración de la red.

En la *Figura 3.17* se puede ver una doble captura de *Wireshark* donde se muestra un paquete completo de información *UDP* que viaja por la red. Cada paquete *UDP* ocupa 1504 bytes. En dicha imagen pueden verse, en formato hexadecimal a la izquierda y en caracteres imprimibles a la derecha, la información que contiene el paquete. En la columna de la derecha, los puntos (.) quieren decir que el carácter que se desea imprimir no es representable mediante un carácter ASCII en *Wireshark*.

En la *Figura 3.17* se ha redondeado con rojo cada uno de los campos de la cabecera para que sean reconocibles con mayor facilidad y éstos serán explicados a continuación:

- Primeros tres Bytes (0-2) [IG Bit, *Ethernet* IG]: Indica si la conexión es Unicast o Broadcast/Multicast.
- Junto con los siguientes tres Bytes (0-5)[Address, *Ethernet* Address]: Indica la dirección del puerto destino del paquete.

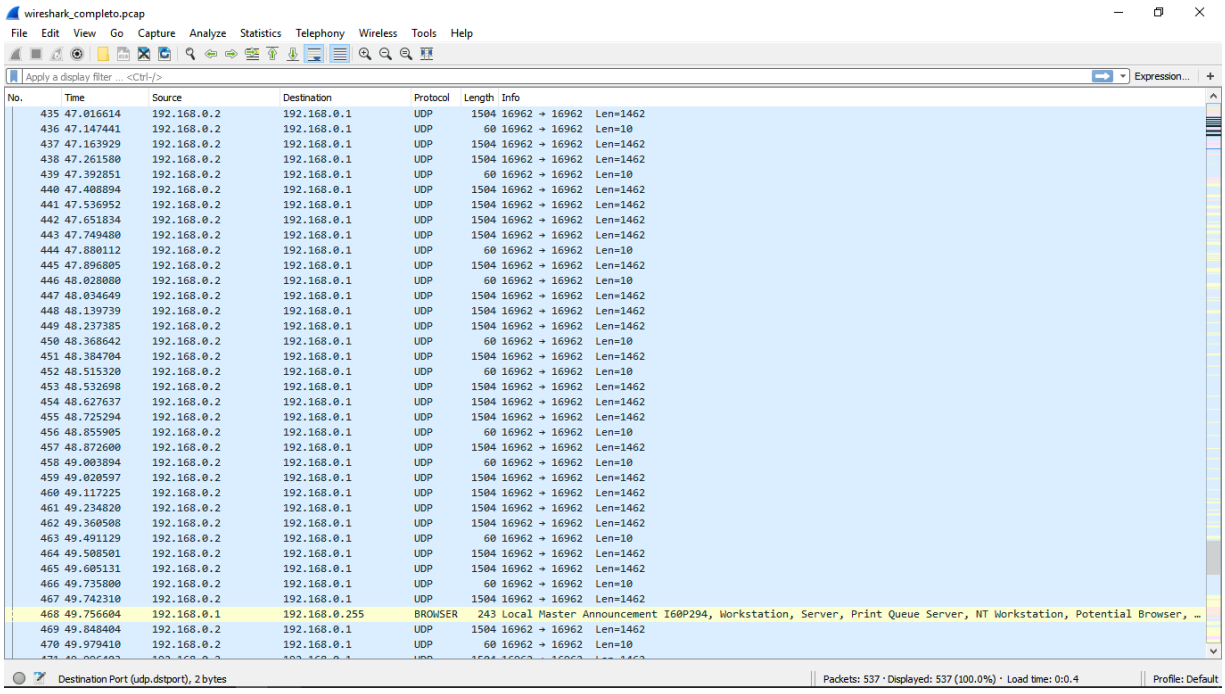


Figura 3.16: Paquetes de datos enviados/recibidos desde el sistema usando la interfaz *Ethernet* de la *FPGA*.

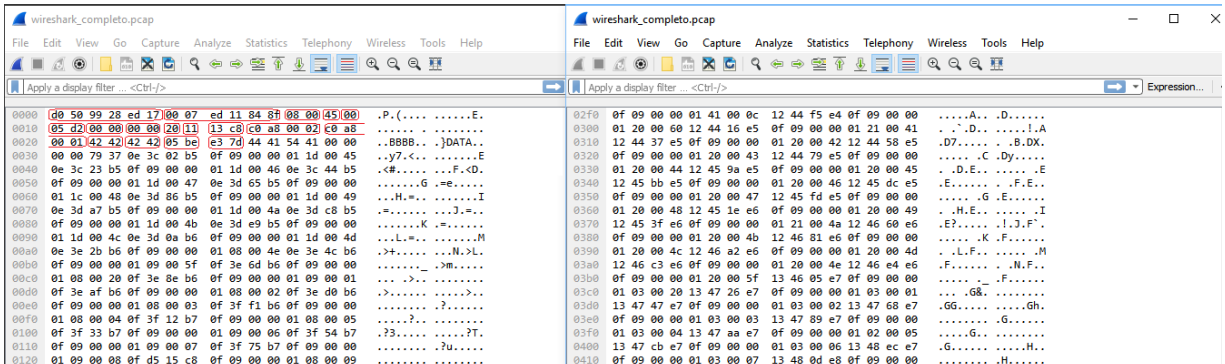


Figura 3.17: Contenido de un paquete *UDP* recibido.

- Siguientes tres Bytes (6-8) [IG Bit, *Ethernet* IG]: Indica si la conexión es Unicast o Broadcast/Multicast.
- Junto con los siguientes tres bytes (6-11) [Address, *Ethernet* Address]: Indica la dirección del puerto fuente del paquete.
- Siguientes dos bytes (12-13) [Type]: Indica el tipo de conexión.
- Siguiente Byte (14) [Longitud de cabecera]: Indica la longitud de la cabecera IP.
- Siguiente Byte (15) [Notificación de congestión]: Indica el estado de la red.
- Siguientes dos Bytes (16-17) [Longitud total]: Indica el tamaño total del paquete IP.
- Siguientes dos Bytes (18-19): [Identificación]: Es un campo de indentificación de paquete.
- Siguientes dos Bytes (20-21): Offset:
- Siguiente Byte (22) [Tiempo de vida]: Indica el tiempo que el paquete va a estar disponible en la red antes de ser descartado definitivamente de ésta.
- Siguiente Byte (23) [Protocolo]: Indica el protocolo de red utilizado para enviar dicho paquete.
- Siguientes dos Bytes (24-25) [*Checksum* de cabecera]: Es una suma de comprobación que asegura que los datos de la cabecera no han sido corrompidos al ser enviados por la red.
- Siguientes cuatro Bytes (26-29) [Fuente]: Indica el puerto *IP* fuente.
- Siguientes cuatro Bytes (30-33) [Destino]: Indica el puerto *IP* destino.
- Siguientes dos Bytes (34-35) [Fuente]: Indica el puerto *UDP* fuente.
- Siguientes dos Bytes (36-37) [Destino]: Indica el puerto *UDP* destino.
- Siguientes dos Bytes (38-39) [Longitud]: Indica el tamaño del paquete *UDP*.
- Siguientes dos Bytes (40-41) [*Checksum UDP*]: Es una suma de comprobación que asegura que los datos del paquete *UDP* no han sido corrompidos al ser enviados por la red.
- Resto del paquete, 1462 Bytes (42-1503) [Constituye los datos del paquete propiamente dichos]: Es la información que se quiere transmitir.
- Cuatro primeros Bytes de los datos del paquete (42-45) [Palabra $"DATA"$]: Es una palabra específica introducida en software que ayuda a nivel de aplicación.
- Resto de los datos del paquete, 1458 Bytes (46-1503) [Información útil o *payload*]: Constituye la carga de información útil del paquete para la aplicación.

En la *Figura 3.18* se puede ver la información de las cabeceras del paquete seleccionado en texto plano, tales como fuentes y destinos, protocolos implementados en dicho paquete, tamaños de los campos y de los paquetes, tipo, tiempo de captura, bytes disponibles y bytes capturados, fecha y hora de captura e información adicional referente a la manera en que se capturó el mensaje.

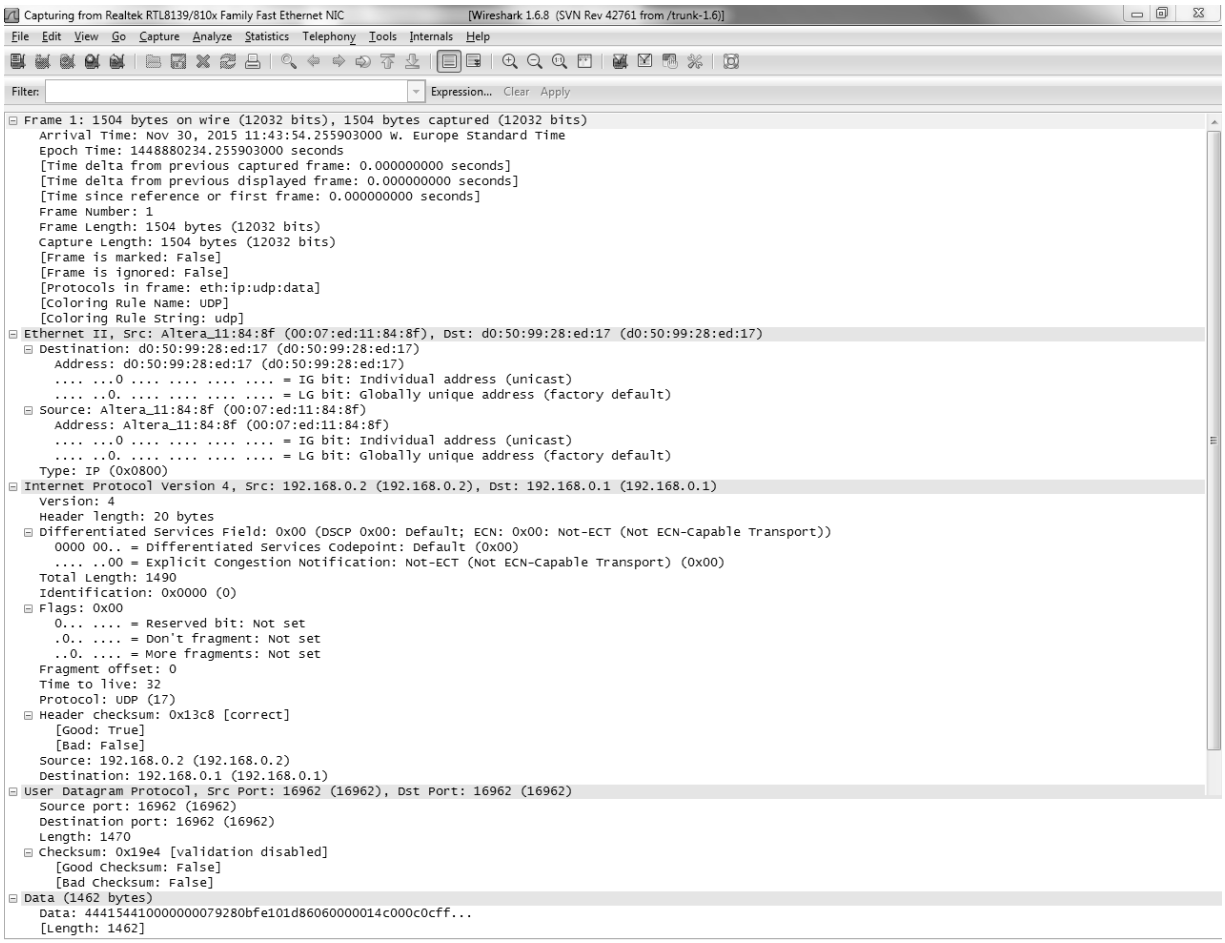


Figura 3.18: Información de uno de los paquetes *UDP*.

3.6. Conclusiones

Como se ha podido ver en este capítulo, el sensor consta de dos placas plano-paralelas que forman un condensador entre sí en el cual se mide la presión que se ejerce sobre él. La placa superior, a su vez, junto con el objeto externo forma condensador en el que se medirá la proximidad relativa del objeto y el sensor descrito.

Este sistema, sigue las leyes básicas de la física y el electromagnetismo. Es posibles medir corrientes generadas en las placas y con ello determinar el estado del sistema. Se han realizado dos medidas, una en la placa superior para determinar la proximidad del objeto de estudio y otra en la placa inferior para conocer la presión ejercida sobre el sensor, en caso de que el objeto se encuentre en contacto con éste.

Las corrientes medidas se digitalizan y pasan por una serie de filtros y sistemas para poder determinar el valor de los datos de interés ya que la señal obtenida es débil, contiene un ruido varios órdenes superior a ella y es analógica e inestable. Es necesario estabilizarla y obtener la señal media de los datos. Todo esto se realiza en

lo que se denomina el procesado de señal del sistema.

Por último, los datos son guardados en un *buffer* para posteriormente introducirlos en paquetes *UDP* y enviarlos mediante una red *Ethernet* hacia un equipo cliente donde se leerán y evaluarán para actuar en consecuencia.

Capítulo 4

Modelado del sistema en *Matlab*

4.1. Introducción

En este capítulo se describe el proceso de creación de un modelo en *Matlab*[30] para simular las transformaciones que sufren los datos que luego van a ser captados por el sistema (Todo el código creado para esta simulación puede encontrarse en el *Anexo G* (página 145).

Esta simulación muestra como afectan las propiedades físicas, eléctricas y magnéticas al sistema y qué es en realidad el dato que se va a leer con el sensor, atendiendo a la naturaleza de la señal que éste capta del exterior. El dato tanto de proximidad como táctil (presión) es una corriente inducida por el campo electromagnético que se crea entre las dos placas del sensor. Como ya se ha explicado, en el caso del sensor de proximidad, una de las placas es el objeto de estudio y la otra la placa forma parte del propio sensor.

Con la simulación de este modelo se tratará de explicar las variaciones que se producen en la señal antes del procesado de datos, así como la manera en que se obtendrá la señal y las bases físicas, magnéticas y eléctricas en las que se basará para demostrar la fiabilidad del modelo.

4.2. Descripción del modelo Matlab

El modelo en *Matlab* desarrollado se representa en la *Figura 4.1*. Se utiliza para simular y analizar las transformaciones que se producen en el dato antes de que se considere como dato captado por el sensor y se proceda a su digitalización. En la *Figura 4.1* se pueden observar dos diagramas de bloques que corresponden al sistema simulado para el caso de trabajar sin señal de excitación (diagrama superior) y para el caso de trabajar con señal de excitación (diagrama inferior).

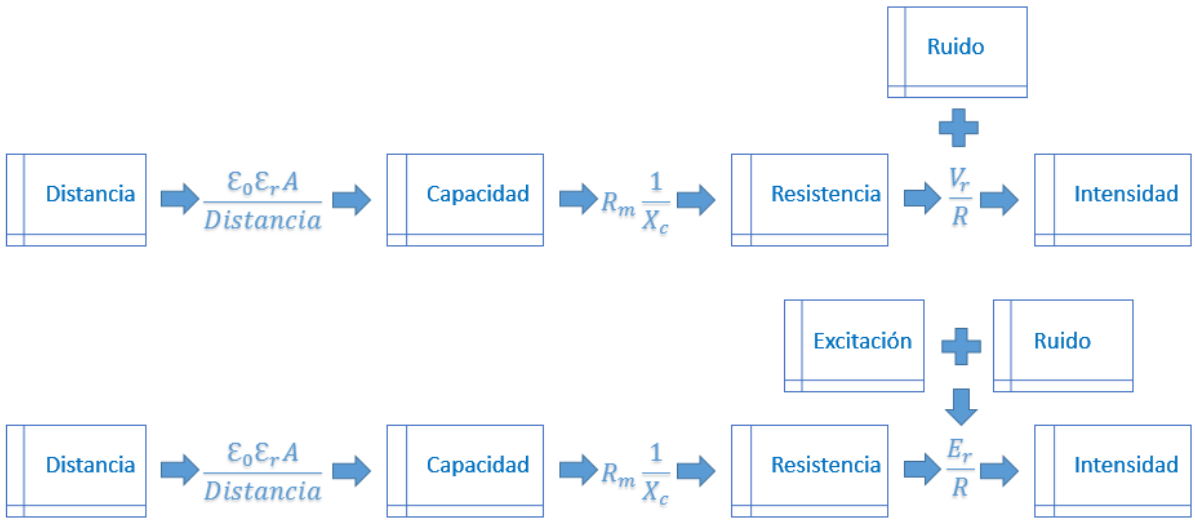


Figura 4.1: Diagrama de bloques de los sistemas simulados en Matlab

Como se puede ver, se parte de una señal que describe el movimiento relativo que realiza el objeto en función del sensor. Esta señal llamada “distancia” se medirá en $milimetros$ frente a $segundos$ y representaría, en dos dimensiones, la velocidad a la cual el objeto se mueve con respecto al sensor. Atendiendo a la fórmula de la capacidad de un condensador de placas plano-paralelas, se obtendrá la capacidad presente en el sistema en función del movimiento anterior. A partir de dicha capacidad el sistema es capaz de obtener la resistencia que presenta el sistema al paso de la corriente y, por último, se obtiene una señal de intensidad de corriente inducida a partir de la tensión presente entre las placas y el valor de resistencia que presenta el sistema para que esta corriente circule.

En el diagrama de la *Figura 4.1* también puede verse el efecto de la tensión que se introduce al sistema para alimentar las placas y en que lugares se introduce ruido al sistema.

4.3. Definición de las condiciones de contorno

Para poder realizar un modelo del sistema es preciso establecer un conjunto de condiciones de contorno que fijen las situaciones en las que se compara el modelo obtenido con los resultados del prototipo.

El modelo reproduce en *Matlab* las transformaciones que sufre el dato de proximidad en el sistema desde que es captado en forma de capacidad por el sensor hasta que llega al sistema cliente para su análisis. Para ello se definen diferentes patrones

de movimiento del robot donde está montado el sensor frente al objeto con el que se va a trabajar.

En la *Figura 4.2* se presenta el movimiento relativo entre el objeto de estudio y el sensor. En el $EjeY$ se representa la distancia en milímetros (mm) y en el $EjeX$ el tiempo en milisegundos (ms) con el que se puede determinar la velocidad de los movimientos.

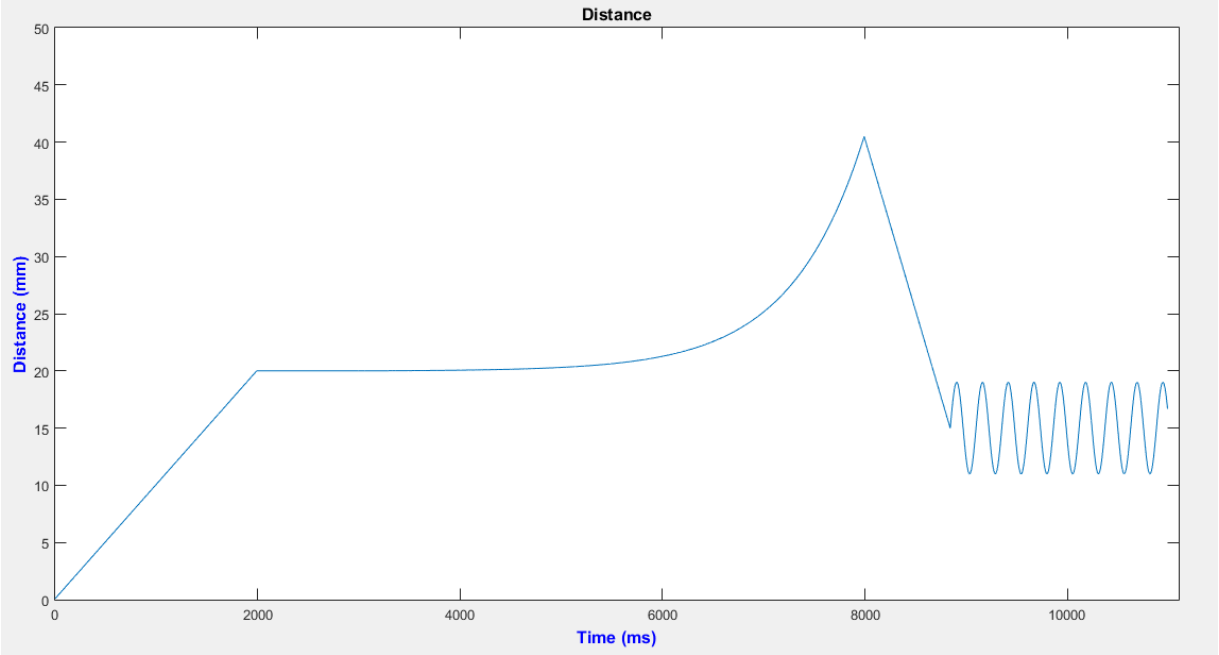


Figura 4.2: Recorrido diseñado para la simulación del sistema.

El diseño de este experimento se ha realizado extrayendo una serie de movimientos diversos de los movimientos más comunes que puede realizar el robot en su funcionamiento normal en el entorno de trabajo. En un único recorrido, como se puede ver, se incluye un acercamiento rectilíneo uniforme con velocidad constante, seguido de unos segundos de espera, luego un movimiento rectilíneo exponencialmente acelerado, seguido de otro alejamiento rectilíneo uniforme con velocidad constante y, por último una sucesión de acercamientos y alejamientos alternados con una frecuencia de 5 Hz.

Esta velocidad de cambio de movimiento resulta significativamente baja para las capacidades del sistema, facilita su análisis, y es una velocidad que se acerca a la velocidad de movimiento normal del ser humano lo que es ideal para este propósito ya que como se ha citado anteriormente, el sistema se quiere destinar a la interacción con las personas en el entorno de trabajo.

En la *Figura 4.3* se puede ver la señal de excitación con la que están alimentadas las placas de los condensadores de todos los sensores. Esta señal es de la forma 4.1.

$$E(t) = A \cdot sen(\omega \cdot t \pm \varphi) \tag{4.1}$$

dónde:

- A es la amplitud de la señal en $voltios$, que en este caso, $A = 12V$,
- ω es la frecuencia de la señal en $radianes/segundos$ (rad/s) y se calcula como $\omega = 2 \cdot \pi \cdot f$ dónde f es la frecuencia de la señal que en este caso vale $f = 98,625KHz$,
- t es la variable de tiempo, y
- φ es una constante de desviación de fase.

Por tanto, la ecuación de la señal quedaría de la siguiente manera 4.2:

$$E(t) = 12 \cdot sen(2 \cdot \pi \cdot 98,625 \cdot 10^3 \cdot t \pm \varphi) \tag{4.2}$$

Esta señal se envía a las placas que forman el sensor para poder medir las diferencias de corriente que se producen en ellas debido a la capacidad que se crea entre éstas y el entorno. Dicho entorno va a depender en gran medida del objeto o los objetos que estén más cercanos al sensor, de la naturaleza de éstos y del electromagnetismo presente en el ambiente. En un entorno ideal, lo único que afecta a la capacidad del entorno y, por consecuencia, a las variaciones de corrientes que se producen en las placas, es el objeto de estudio, pero en el mundo real esto nunca sucede y, a parte del objeto de estudio, influyen el resto de objetos del entorno próximo, la meteorología y cualquier elemento que altere el campo electromagnético del sistema. Por ello es necesario calibrar los sensores de proximidad siempre antes de utilizarlos.

En la *Figura 4.4* se representa un ruido blanco gaussiano con media 0 y varianza 1 que se añade a la señal en el sistema cuando se obtiene ésta desde el exterior en un entorno analógico. Esto ocurre en cualquier sistema que obtenga señales analógicas desde el exterior. Aunque a veces es posible despreciar este ruido porque el nivel de la señal que se necesita estudiar es varios órdenes de magnitud superior al nivel del ruido, en este caso esto no es así y es preciso tenerlo en cuenta ya que la señal dato va a ser **8 órdenes de magnitud** inferior al ruido.

A continuación, en la *Figura 4.5* se puede ver la señal de excitación con la que se alimentan las placas pero con un ruido blanco gaussiano de media 0 y varianza 1 añadido. Esto se debe a que la señal de excitación también es una señal analógica que viaja por un cable de cobre al que se le acoplan distintos ruidos electromagnéticos. En la gráfica puede verse que la señal de excitación resultante que llega a las placas del sensor es la suma de las señales de las *Figuras 4.3* y *4.4*.

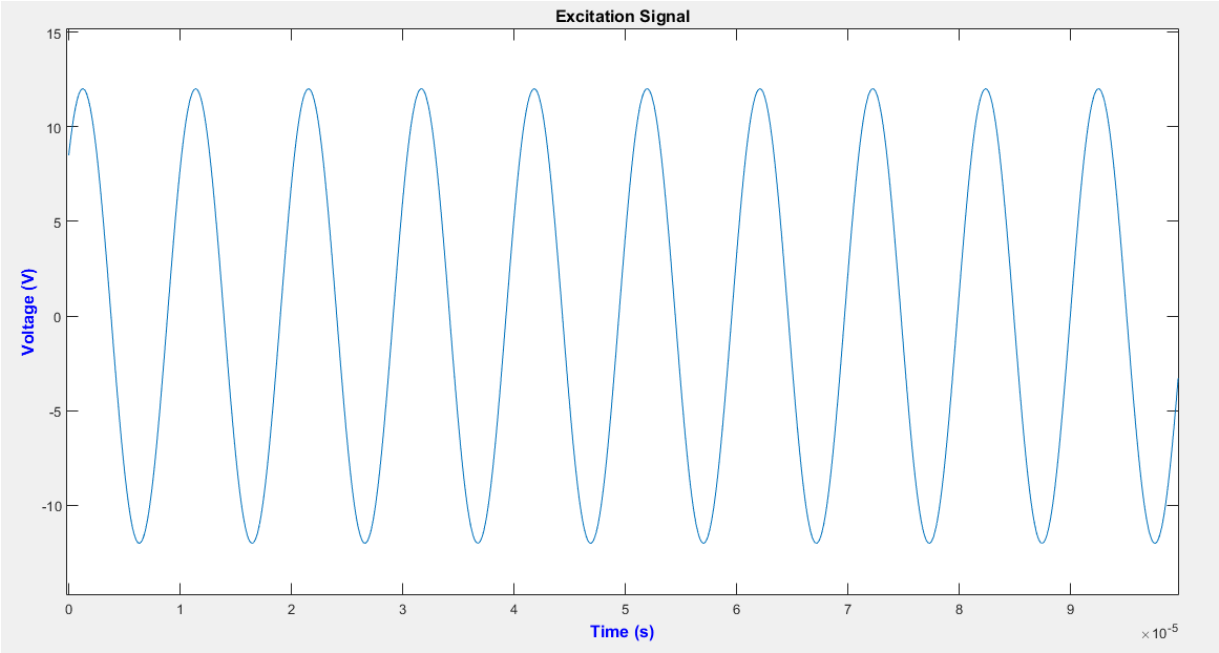


Figura 4.3: Señal de excitación.

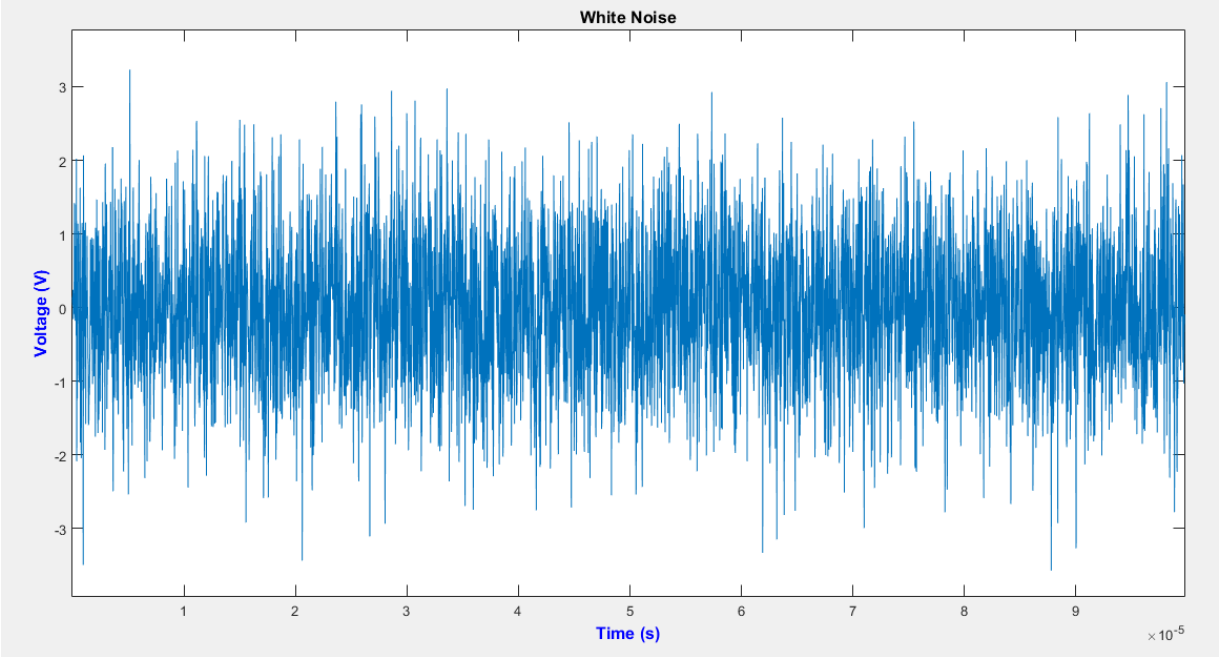


Figura 4.4: Ruido blanco gaussiano.

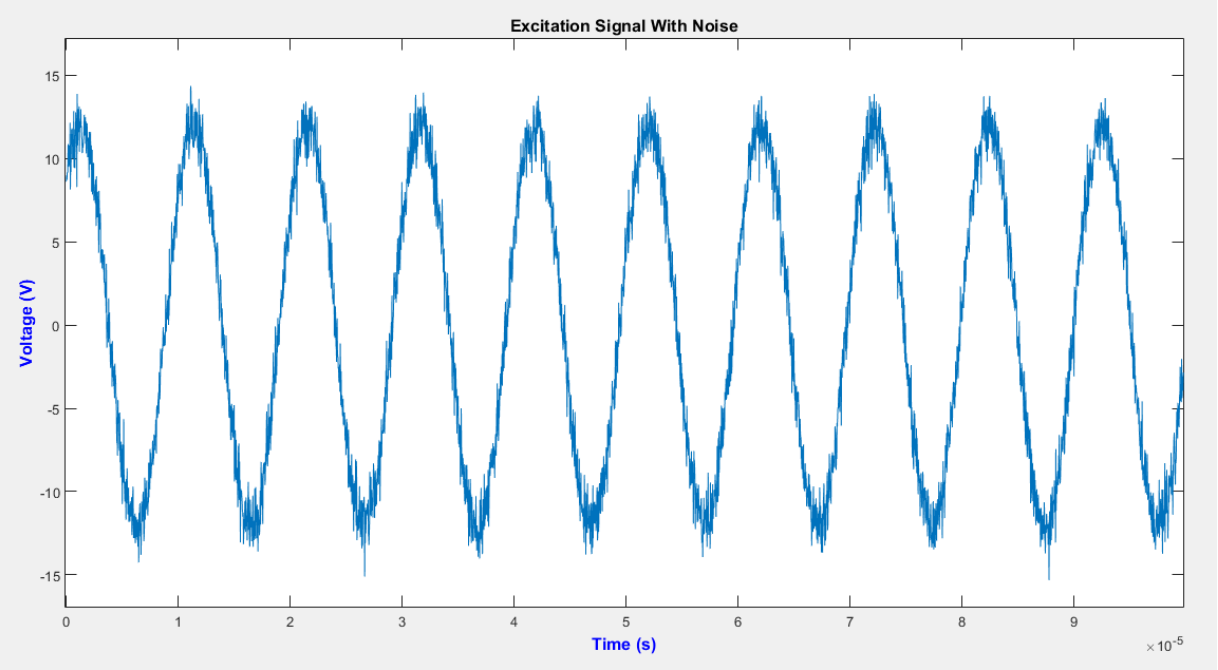


Figura 4.5: Señal de exitación con ruido blanco gaussiano.

4.4. Experimento sin señal de excitación

En la *Figura 4.6* se muestra la capacidad presente en el condensador formado por la placa más externa del sensor y el objeto de estudio para el movimiento relativo de la *Figura 4.2* cuando se utiliza el sistema en modo pasivo, esto es, sin alimentar las placas con la señal de excitación. Esta gráfica se obtiene directamente con la constante dieléctrica del medio y las diferentes distancias que existen entre la placa más externa del sensor y el objeto bajo estudio que actúan como placas de un condensador. Esta relación se explica mediante la expresión 4.3.

$$Capacidad = \frac{\epsilon_0 \cdot \epsilon_r \cdot A}{Distancia} \tag{4.3}$$

dónde:

- ϵ_0 es la constante dieléctrica en el vacío: $(\epsilon_0 = 8{,}85^{(-12)})$,
- ϵ_r es la constante dieléctrica relativa del medio físico situado entre el sensor y el objeto de estudio, que en este caso es el aire $(\epsilon_r = 1)$,
- A es el área mínima de entre el sensor y el objeto, que en este caso siempre va a ser la del sensor $(A = 4cm * 4cm = 16cm^2)$, y
- $Distancia$ es la distancia, que como se ha dicho anteriormente, es la que en cada instante haya entre el objeto de estudio y el sensor.

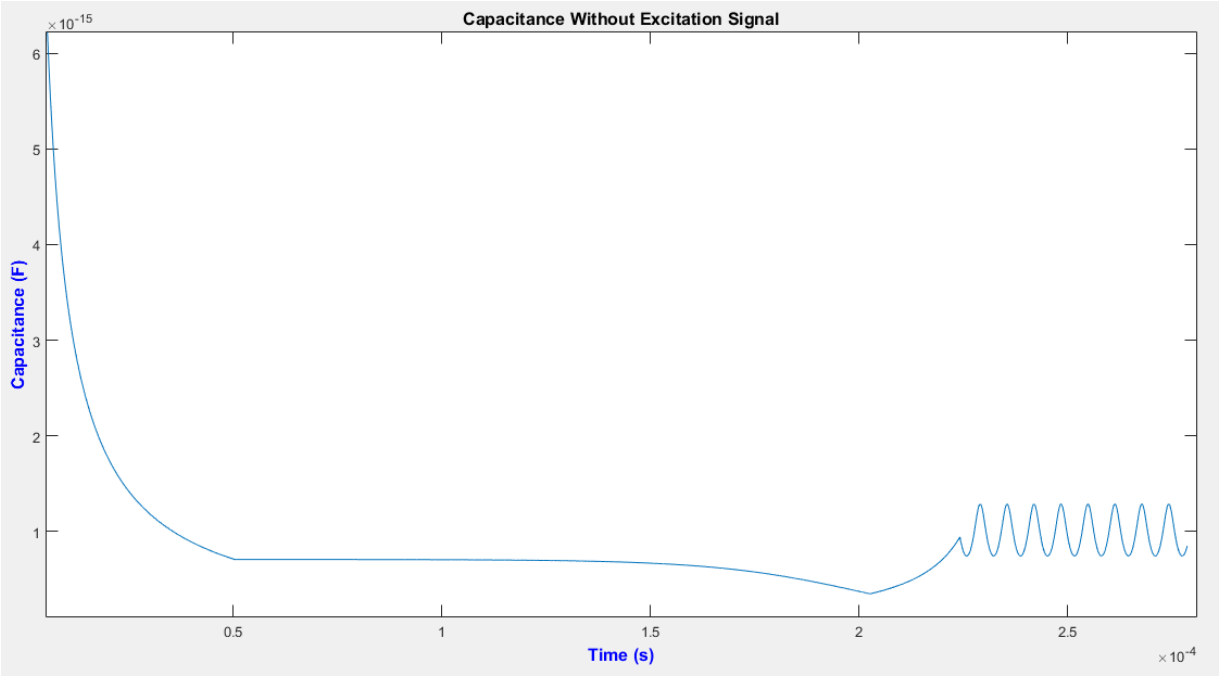


Figura 4.6: Capacidad que presenta el sistema sin señal de excitación.

En la *Figura 4.7* se representa la resistencia que presenta el sistema al paso de la corriente entre el objeto con el que se está trabajando y el sensor cuando se opera con el sistema en modo pasivo. El valor del la resistencia al paso de la corriente es del orden de $G\Omega$. Esto se debe a que no hay contacto físico entre el objeto de estudio y el sensor por donde la corriente pueda circular sino que la corriente que circula lo hace inducida por el campo electromagnético creado a través del aire.

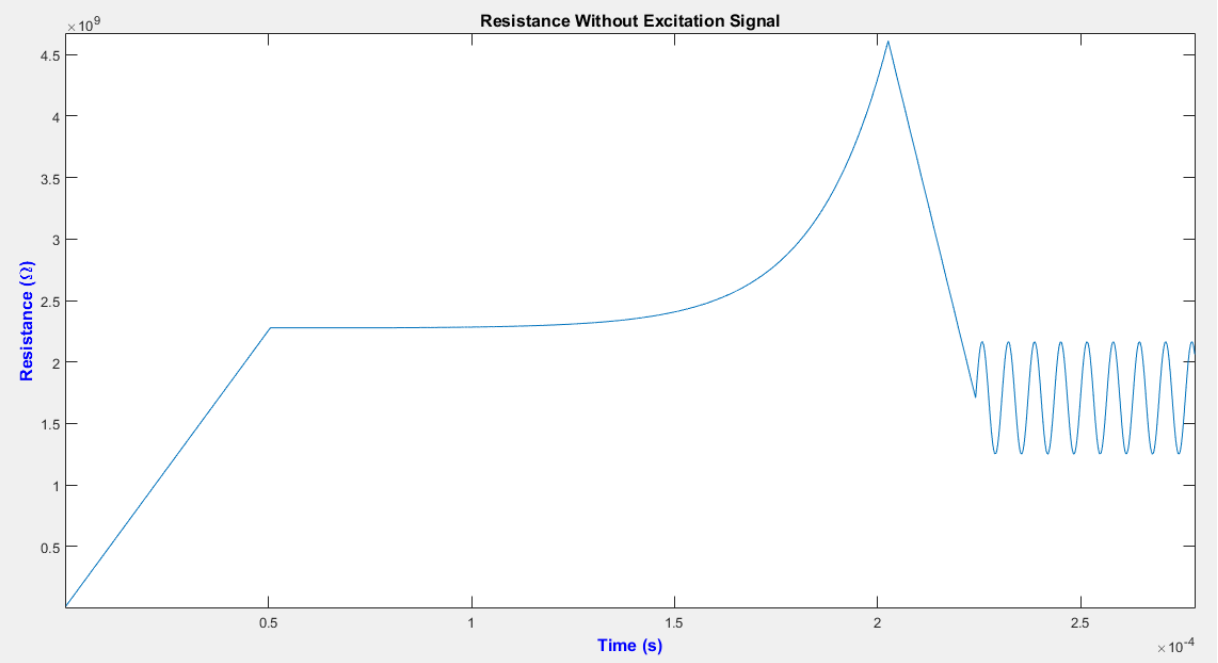


Figura 4.7: Resistencia que presenta el sistema sin señal de excitación.

En la *Figura 4.8* se representa la corriente que el sistema debería medir para el movimiento relativo entre el objeto de estudio y el sensor que se ha presentado en la *Figura 4.2*. Comparando ambas gráficas, se puede observar el aspecto simétrico de la corriente de entrada con un comportamiento correlacionado con la corriente medida. Sin embargo, esta señal no es más que un ruido blanco gaussiano que se ha modulado con la envolvente del movimiento relativo que se ha realizado. Esto se debe a que en el entorno existe energía (capacidad), la cual se ha modificado con el movimiento, pero al no tener señal de excitación, no se recibe más que la deformación que produce esta variación de capacidad en el ruido existente.

Esta variación coincide con lo que se quiere medir pero no se puede extraer del ruido presente en el sistema en la realidad. De aquí se puede deducir que, sin ruido (sistema ideal pero no real) no hay señal a la salida de la cual se puede obtener los datos que se quiere medir y con ruido (sistema real) no se tendría una componente espectral de señal que se pueda diferenciar para eliminar el resto de componentes del ruido y su energía y, así, poder obtener los datos. Por tanto, es necesario introducir una señal de excitación al sistema que adquiera la información de los datos para luego

poder aislarla del ruido y obtener la señal deseada. En el siguiente apartado se puede ver esta misma simulación pero con señal de excitación.

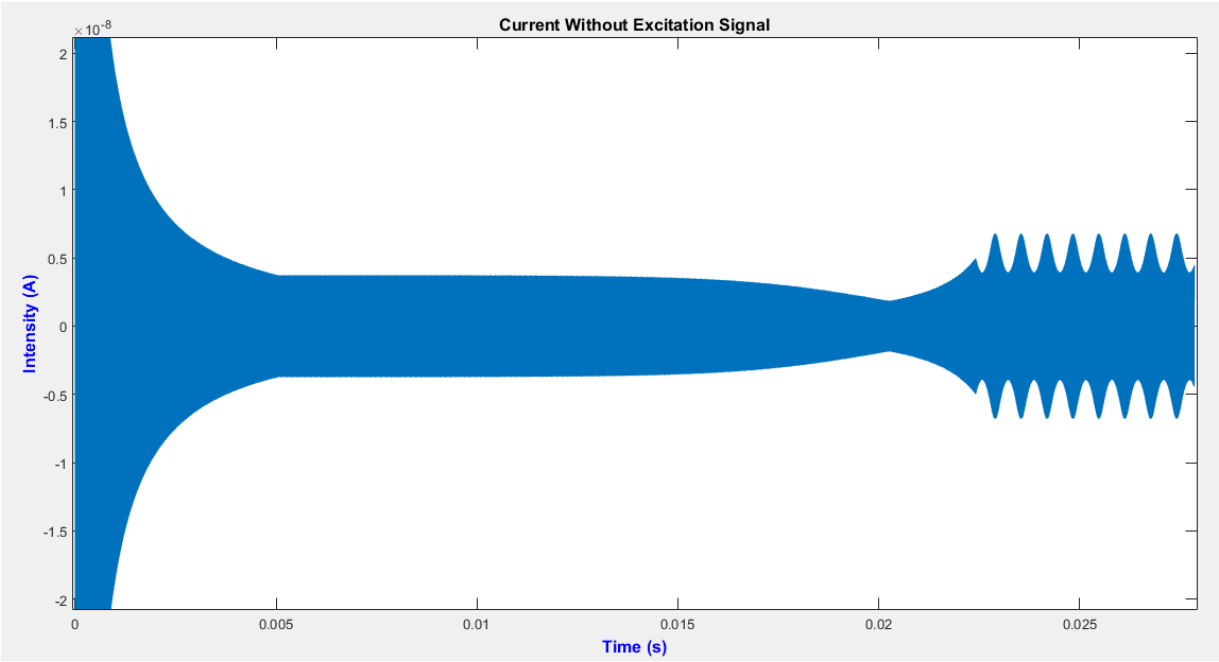


Figura 4.8: Corriente medida sin señal de excitación.

En la realidad, y como se ha explicado con anterioridad, cada vez que se toma una señal del exterior del sistema, se le suma a ésta un ruido blanco gaussiano que no ha sido tenido en cuenta en la *Figura 4.8*. A continuación, en la *Figura 4.9* se ha introducido el efecto e dicho ruido en la señal de corriente medida. Como se puede ver, la envolvente de la señal que comprende los datos a obtener ha desaparecido completamente, esto se debe a que el ruido tiene mucha más energía que la señal de corriente obtenida. Los datos están ahí solo que no se pueden ver.

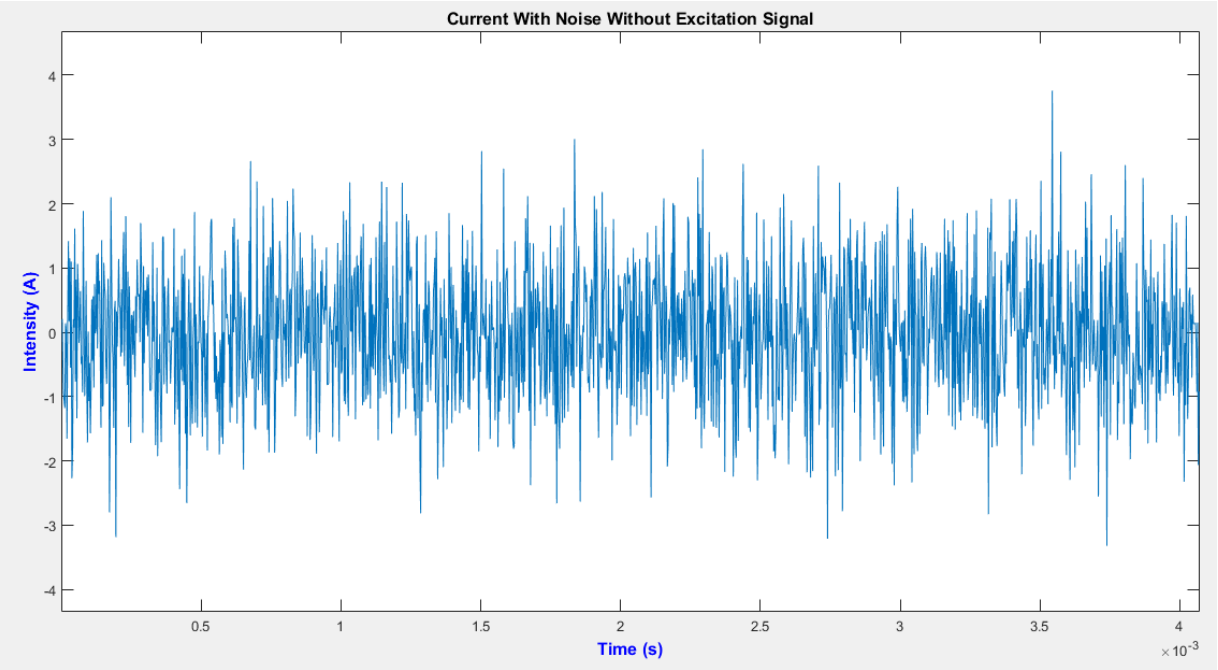


Figura 4.9: Corriente medida sin señal de exitación con ruido blanco gaussiano.

4.5. Introducción de la señal de exitación

A continuación se repetirá el mismo experimento pero con las placas alimentadas con la señal de excitación vista con anterioridad en la *Figura 4.3*.

En la *Figura 4.10* puede verse la capacidad que presenta el sistema definido por el objeto de estudio y el sensor cuando se introduce la señal de excitación que se muestra en la *Figura 4.5*.

En la *Figura 4.11* puede verse una gráfica donde se representa la resistencia que presenta el sistema definido por el objeto de estudio y el sensor cuando se introduce al sensor una señal de excitación como la que se muestra en la *Figura 4.5*.

En la *Figura 4.12* puede verse una gráfica donde se representa la corriente medida por el sistema.

En la *Figura 4.13* puede verse una gráfica donde se representa la corriente medida por el sistema con las interferencias del ruido blanco y gaussiano añadidas. Como puede observarse, la señal aparentemente es ruido blanco y gaussiano, esto se debe a que la señal de corriente medida es muy débil, en general varios órdenes de magnitud menor al ruido que se introduce en el sistema. Esto es algo que era de esperar ya que el $Faradio$ es una unidad muy grande y la capacidad de un sistema de esta forma es del orden de $100pF$, lo que resulta en una resistencia del sistema del orden de $100M\Omega$ lo cual, con una señal de excitación como la que se tiene, de $\pm 12,5V$ lleva

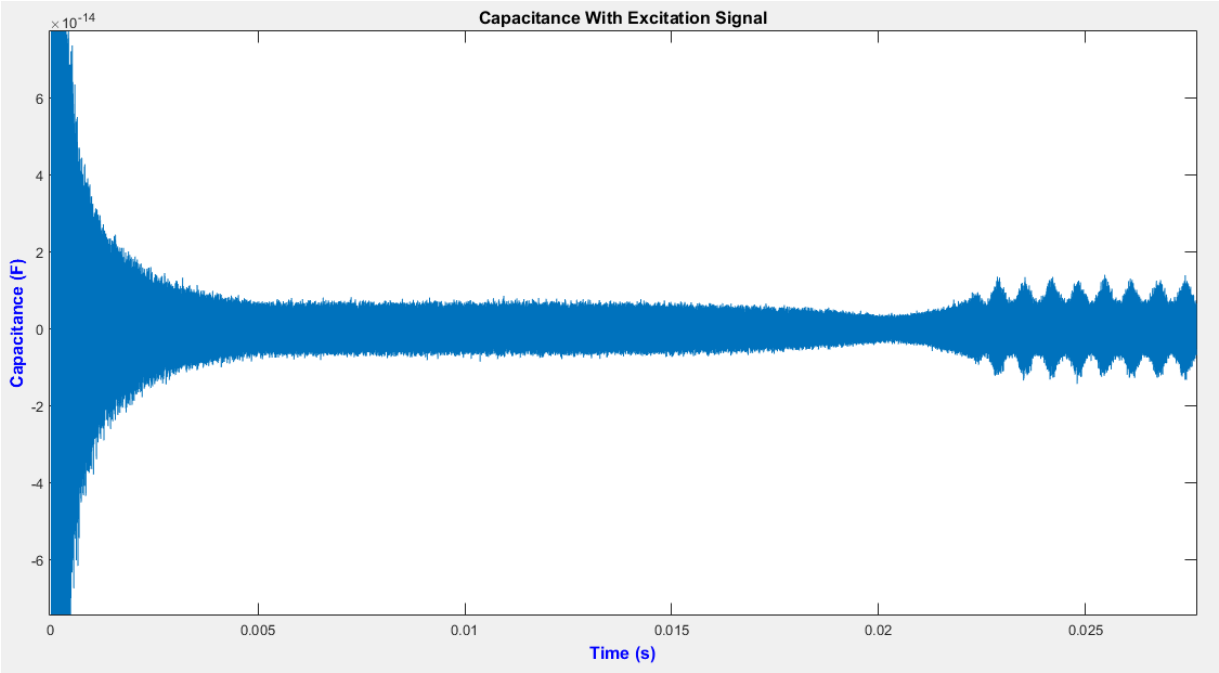


Figura 4.10: Capacidad que presenta el sistema con señal de excitación.

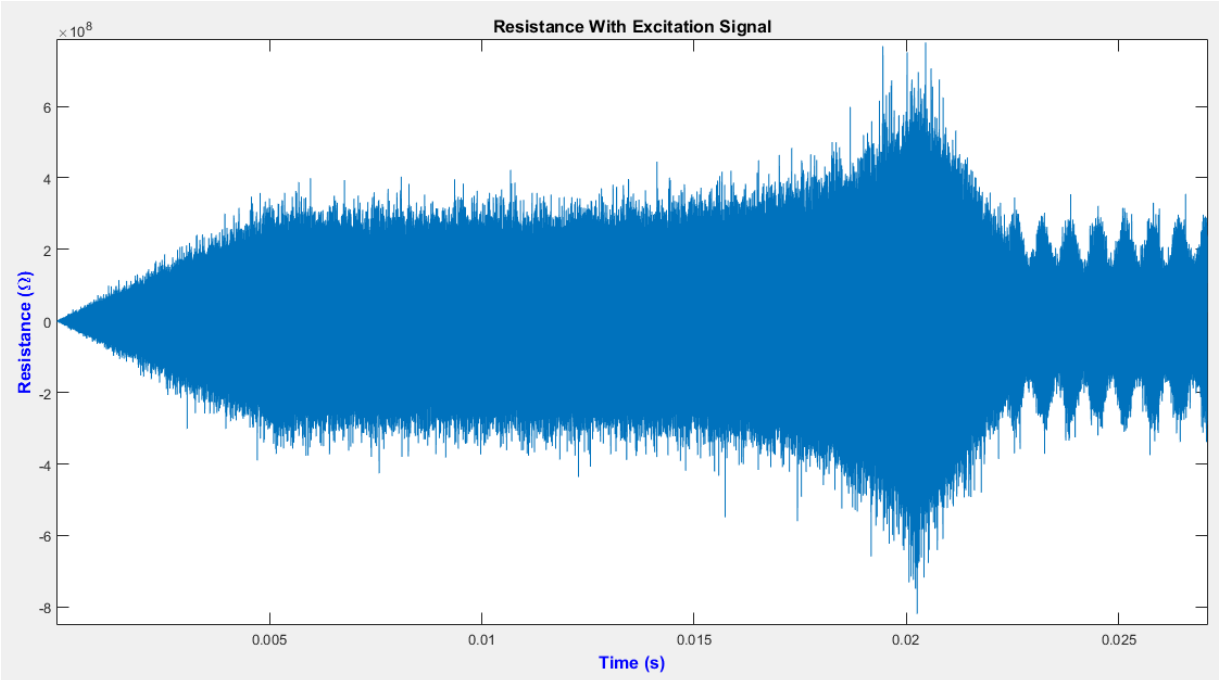


Figura 4.11: Resistencia que presenta el sistema con señal de excitación.

a tener una corriente del orden de $10\mu A$. Esto hace que al añadir el ruido blanco a la señal, éste prevalezca y la señal sea ininteligible. Para poder observar la señal, posteriormente en el sistema de procesado de señal, se realiza un filtrado en banda que elimina la mayoría de las componentes del espectro del ruido y con ellas la mayoría

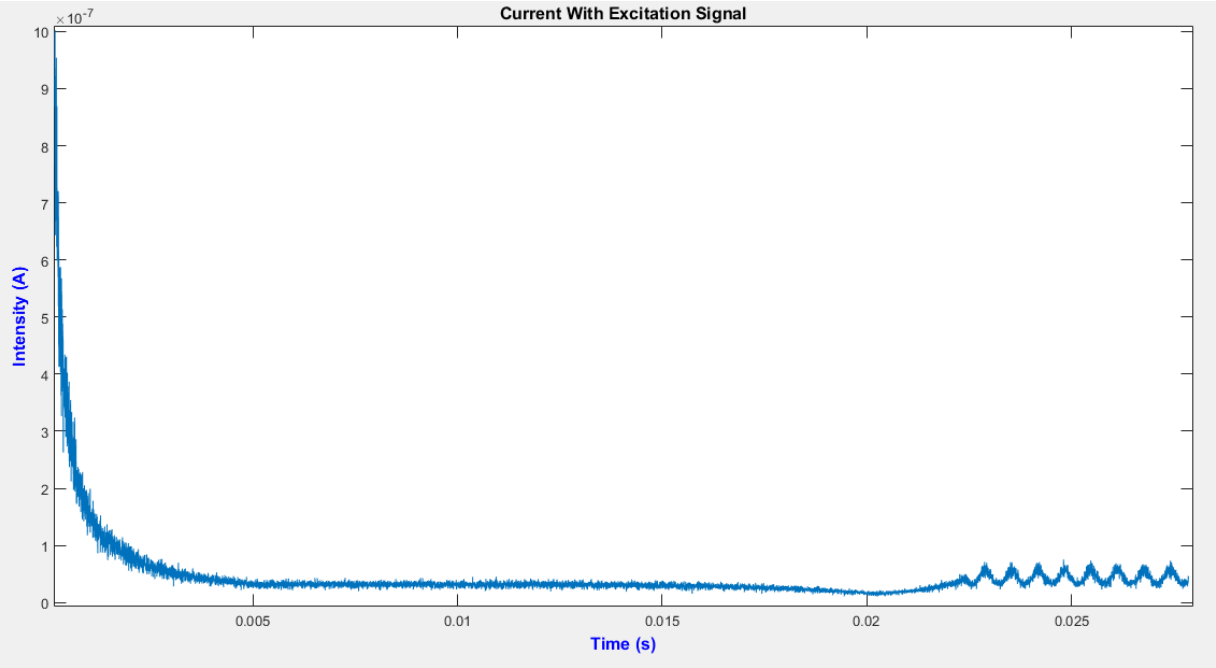


Figura 4.12: Corriente medida en el sistema con señal de excitación.

de su energía, dándonos la posibilidad de poder observar la señal bajo estudio.

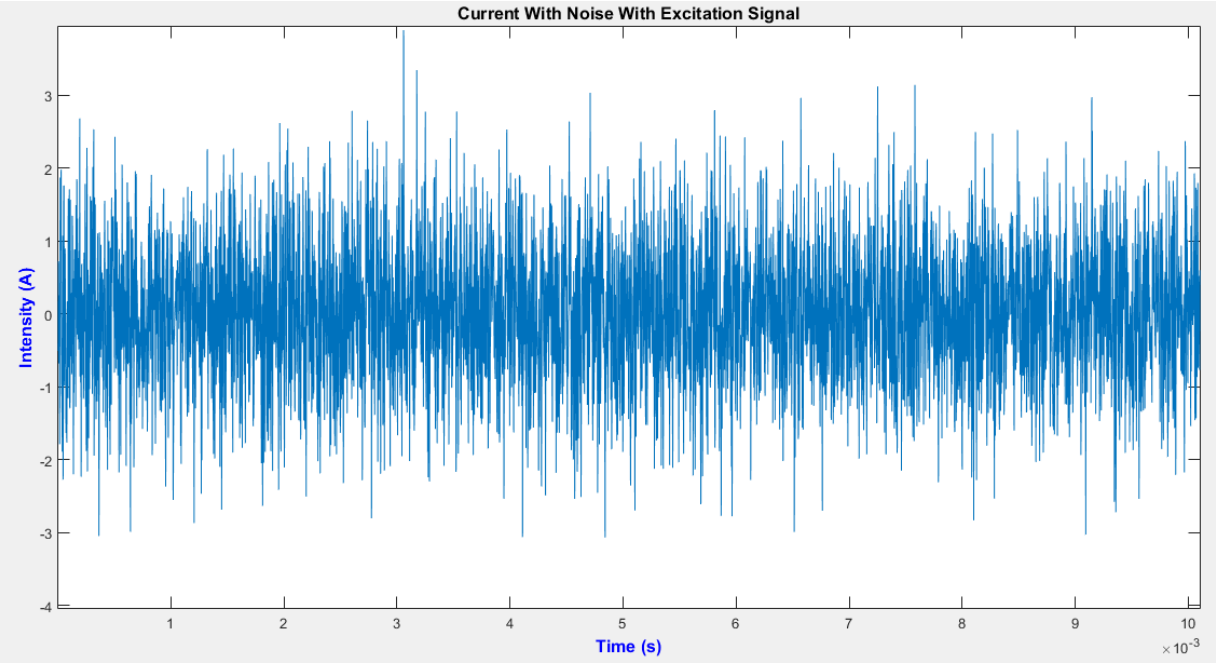


Figura 4.13: Corriente medida en el sistema con señal de excitación y con ruido blanco gaussiano.

4.6. Conclusiones

Como se ha indicado, el modelo realizado en *Matlab* trata de estudiar el sistema en cuanto a los datos recibidos de los sensores. Da una visión general de los datos resultantes de la medida de proximidad del sistema y como varían en función de las interacciones entre el objeto externo y el sensor, si se tocan, se alejan o se acercan.

De la misma manera, este estudio sirve también para estudiar la función táctil del sensor ya que el condensador se rige por la mismas leyes físicas y electromagnéticas y lo que continúa variando es, de la misma manera, la distancia entre placas. Si el objeto, ya está en contacto con el sensor de proximidad, puede ejercer una presión y acercar las placas del condensador que genera la medida táctil, o por el contrario, dejar de ejercerla y permitir que ambas se alejen hasta volver a obtener la distancia relativa entre placas normal del sistema.

Existe una gran diferencia entre ambos casos. En el condensador que mide la proximidad, una de las placas es conocida, la placa más externa del sensor, y la otra placa es desconocida. Se desconoce su material y su distancia máxima. En el condensador táctil, ambas placas son conocidas y se conoce también la distancia máxima a la que se encuentran.

El modelo Matlab realizado permite tener un modelo en alto nivel que representa el comportamiento del sistema y facilita el análisis rápido de distintas situaciones de uso.

Capítulo 5

Implementación del sistema

5.1. Introducción

En este capítulo se describe el proceso de análisis del sistema de procesado de señal así como su implementación y los procesos seguidos para el análisis y cálculo de su latencia. El sistema ha sido diseñado utilizando las herramientas de *Altera Corp.* para crear sistemas basados en *Nios II*[31].

La herramienta principal utilizada es *Quartus II*. En la *Figura 5.1* se muestra la interfaz de usuario de la herramienta con el proyecto objeto de trabajo de esta memoria abierto.

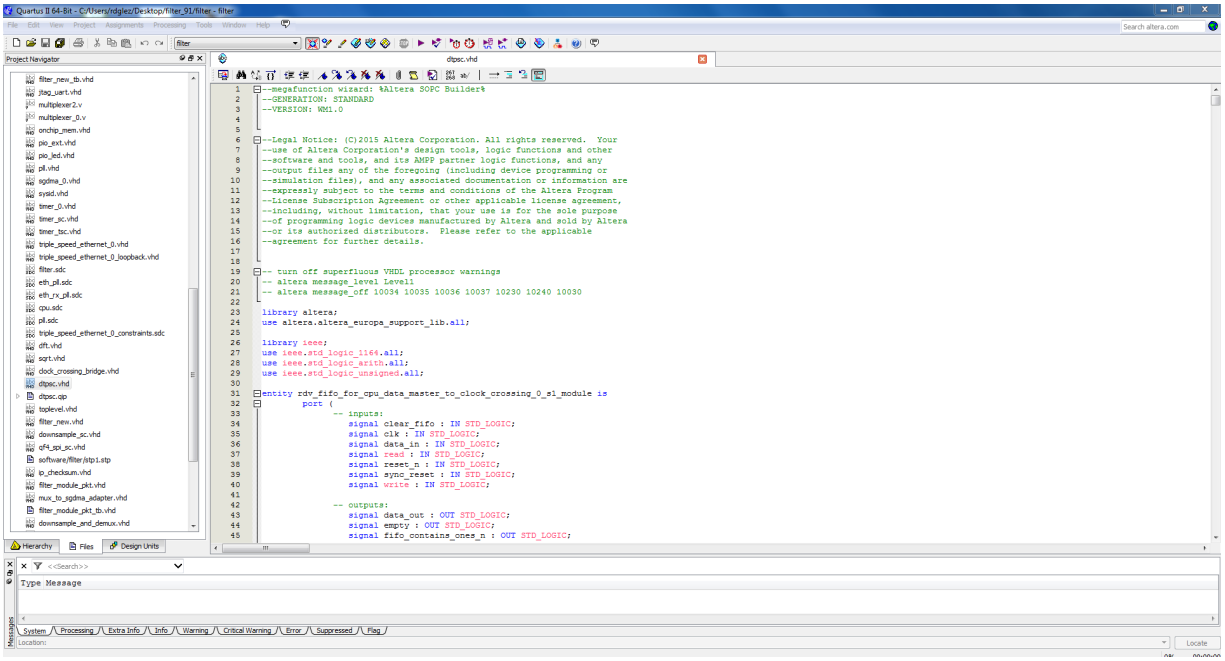


Figura 5.1: Vista preliminar del software de diseño *VHDL Quartus II*.

Como puede verse, en esta herramienta de diseño *EDA* se trabaja principalmente describiendo en lenguaje *VHDL* los distintos subsistemas de los que consta este proyecto e interconectándolos entre sí mediante subsistemas que engloben los anteriores e interconecten sus distintas señales lógicas. Pero también hay otra manera de construir sistemas más complejos a partir de entidades y componentes previamente creados, como los que ofrece la propia empresa *Altera Corp*. Estos módulos descritos con anterioridad, pueden ser incluidos en el diseño a través de una herramienta muy potente, interna a este software, cuyo nombre es *SOPCBuilder*.

En la *Figura 5.2* puede verse una vista previa de esta herramienta interna del *Quartus II* con el diseño acabado.

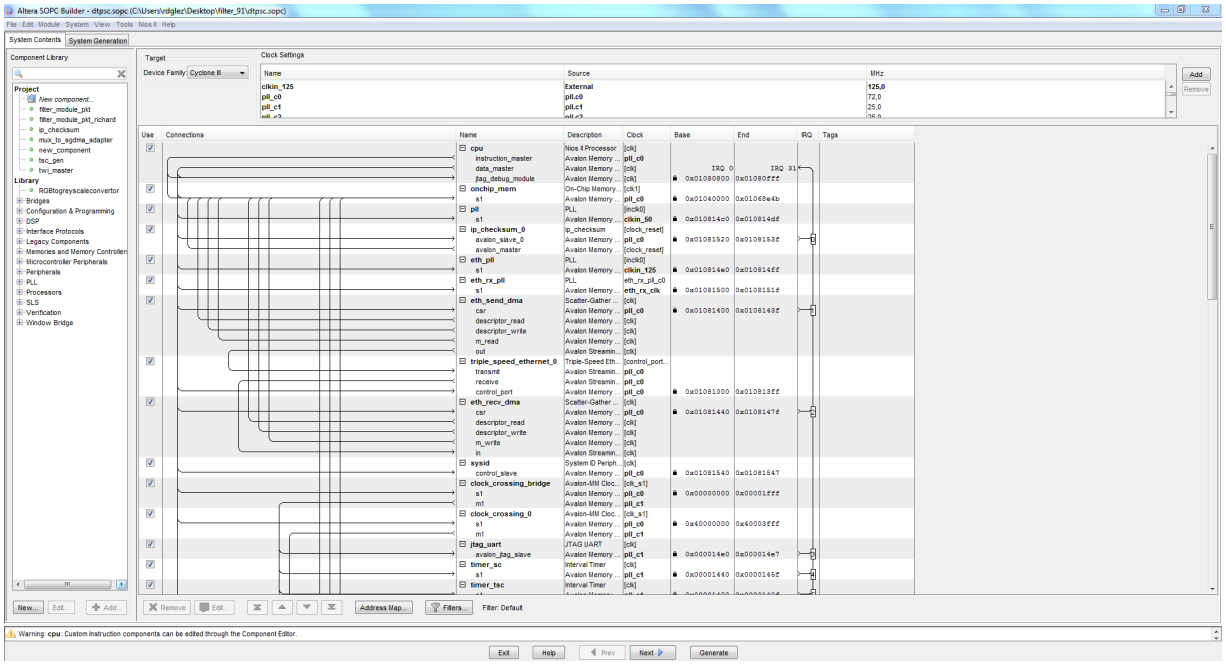


Figura 5.2: Vista preliminar del software *SOPC*Builder.

Esta herramienta consta de una gran cantidad de componentes predefinidos (Bloques *IP*) por *Altera Corp.* a los que se les puede configurar todo tipo de variables y características con el fin de obtener el componente deseado de entre un gran número de posibilidades. Además, *SOPCBuilder* consta de una interfaz gráfica para interconectar todos los componentes del sistema creado, tanto los componentes predefinidos como los que se describen enteramente en este proyecto. Una vez se ha terminado de diseñar este sistema, la herramienta creará los archivos *.vhd* con el código *VHDL* correspondiente al sistema creado y los incluirá en el proyecto en uso en *Quartus II*, dejando todo listo para el compilado y volcado del sistema en la *FPGA*.

El sistema creado utiliza la *CPU Nios II*[31] para controlar el flujo de datos a través de los diferentes módulos a los que está inteconectada mediante un *Bus Avalon-MM*.

A continuación, mediante estas herramientas se estudiará como ha sido construido el sistema y se analizará detenidamente para buscar vías alternativas de funcionamiento y optimización del mismo.

5.2. Implementación FPGA

El sistema está construido y diseñado para su volcado en una *FPGA Cyclone III de Altera*[32][33][34], que incluye un microprocesador *NIOS II*[31]. Este microprocesador será el encargado de controlar el flujo de datos por los diferentes subsistemas hardware implementados en la *FPGA*. El microprocesador será programado en *C++* mediante el entorno *Eclipse* proporcionado por *Altera Corp.* En la *Figura 5.3* puede verse un diagrama de bloques del sistema donde se puede ver la organización de los módulos y, de esta manera, la organización de la descripción del sistema. Estos bloques han sido desarrollados en *VHDL*[35][36][37][38]. El sistema incluye interfaces digitales hacia los conversores *(ADC)* y *(DAC)* y un bloque de comunicación *Ethernet* 10/100/1000.

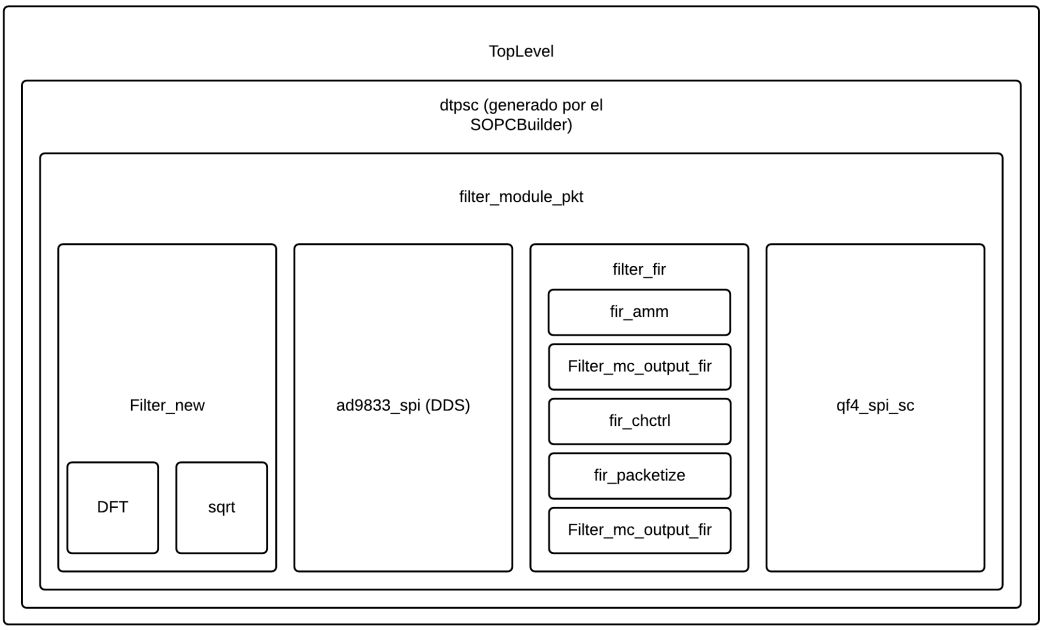


Figura 5.3: Diagrama de bloques del sistema *VHDL*.

El prototipo descrito es funcionalmente correcto, habiendo probado su funcionamiento utilizando los sensores disponibles. Se espera, sin embargo, realizar una optimización temporal del sistema con objeto de reducir su latencia de procesamiento y

por tanto mejorar los tiempos de respuesta del sistema.

5.2.1. Arquitectura de la implementación FPGA

La estructura de la arquitectura de la *FPGA* se muestra en la *Figura 5.4*.

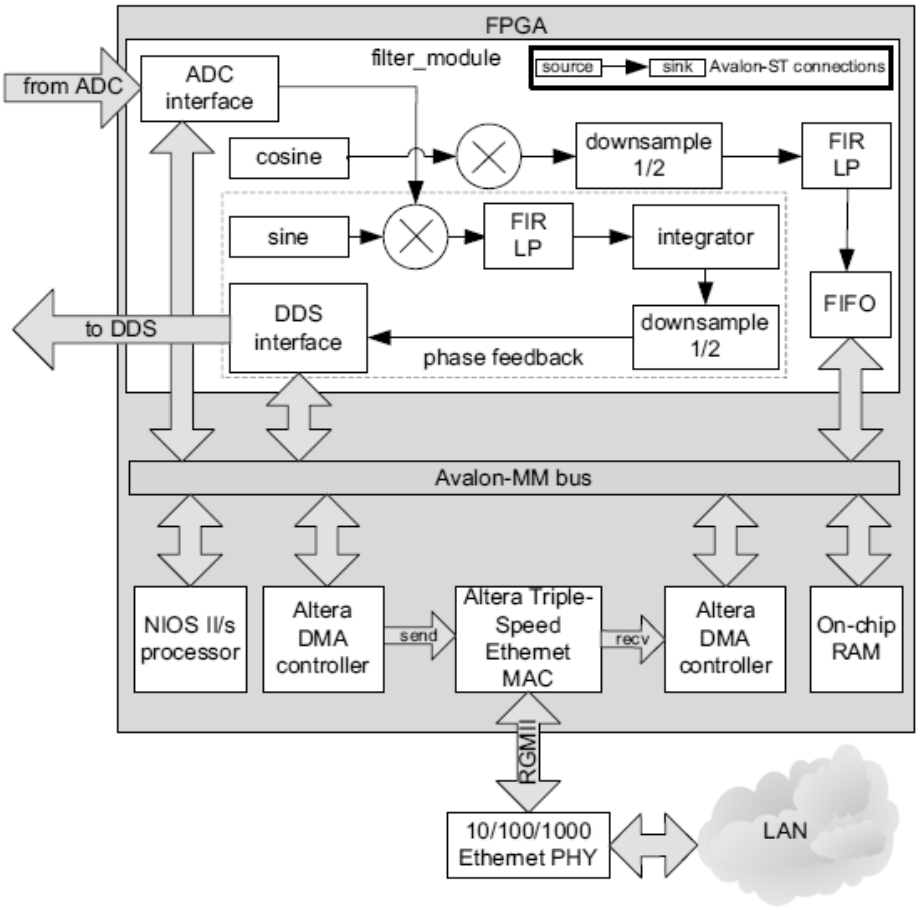


Figura 5.4: Diagrama de bloques del diseño implementado en la *FPGA*[39].

El bloque principal, llamado *filter_module_pkt*, implementa partes del esquema del procesado digital de la señal *(DSP)* mostrado en la *Figura 3.11*. El primer filtro paso banda *FIR* después del convertidor analógico/digital está integrado en el chip *ADC*. Desde ahí la muestra es transferida vía *SPI* a la *FPGA*. El módulo de raíz cuadrada, el filtro *FIR* paso bajo y el control de fase también están implementados por el bloque *filter_module_pkt*.

La arquitectura de la *FPGA* contiene también un sistema programable en chip *(SOPC)*, el cual consta de un procesador *NIOS II*, bloques *RAM on chip* y un controlador *Ethernet* con soporte para *DMA*. Todos estos componentes están interconectados

por un *Bus Avalon-MM* al cual también está conectado el bloque *filter_module_pkt*.

El paquete *filter_module_pkt* acepta datos desde el *convertidor analógico/digital* a través de la interfaz *ADC*. Esta interfaz conecta las interfaces *SPI* del *ADC* al *Bus Avalon-ST*, utilizado internamente por el *filter_module_pkt*. Se provee también a la *CPU* de una interfaz de conexión al sistema de buses para el envío de comandos y datos de configuración al *ADC*.

Los bloques de generación de senos y cosenos emiten una ráfaga *(stream)* de muestras [0, 1, 0, -1] y [1, 0, -1, 0] respectivamente, lo que representa una función seno o coseno con una amplitud de 1, que muestrea a una velocidad que es 4 veces su frecuencia. Este *stream* generado se multiplica con el de muestras de entrada procedente del interfaz *ADC*. La velocidad del *stream* que resulta de esta multiplicación se controla desde el mecanismo de control de flujo de datos que se encuentra en la interfaz del *Bus Avalon-ST*. La interfaz *ADC* emite una muestra de datos por cada muestra recibida. Los bloques multiplicadores solamente aceptan una muestra desde el generador de senos/cosenos cuando éstos reciben una muestra desde el *ADC*. Así, el *stream* seno/coseno funciona a la misma velocidad de muestreo que el *ADC*. Este hecho se traduce también en una fase fija entre el bloque generador del *stream* y la frecuencia de excitación.

El *stream*, que fue multiplicado por el coseno, es luego diezmado, filtrado y pasado a la *FIFO* de salida como la magnitud de la medida. Antes de introducirlos en la *FIFO*, a los datos se les añade una marca de tiempo. La *CPU* puede de esta manera leer la marca de tiempo del dato desde el *búfer* implementado en la *FIFO*.

El otro *stream*, el que ha sido multiplicado por el seno, solamente se filtra y se integra. Este integrador consta de un acumulador que va acumulando las muestras de entrada (interpretadas como valores con signo). Después de cada suma el integrador emite un nuevo valor de acumulación en su puerto de salida. La salida del acumulador luego es diezmada y enviada a la interfaz *DDS*. La interfaz de salida del *DDS* envía esta desviación de fase al circuito *DDS*. Esto también provee una interfaz al sistema de buses de la *CPU* para escribir en los registros de configuración del chip *DDS*.

El propósito de este *sistema on-chip* es empaquetar las muestras recibidas, enviarlas fuera del sistema vía *Ethernet* para procesarlas y evaluarlas por otro sistema. Esto se suele utilizar también para escribir un dato de configuración inicial para los componentes externos (*ADC* y *DDS*), y para conmutar entre los diferentes modos posibles de funcionamiento de los sensores.

El rendimiento máximo teórico del *pipeline* del *DSP* es de 17 **ciclos de reloj por muestra**. En la práctica, el rendimiento está limitado por el bus *SPI* que se utiliza para obtener las muestras desde el *ADC* ya que este bus funciona con el mismo reloj que el *pipeline*.

Como **una muestra tiene 24 bits** de longitud en el bus *SPI*, **8 bits de cabecera y** 16 **bits de muestra**; y el *ADC* transmite como máximo 3 de los canales configurados, incluso pensando que se utiliza sólo uno al mismo tiempo, el reloj debe funcionar como mínimo 80 veces por encima de la frecuencia de muestreo deseada. Con una frecuencia de muestreo de $400KHz$, esto resulta en una frecuencia mínima de trabajo de $32MHz$.

Si la potencia consumida fuera un problema cuando se integren estos sensores en un robot, se podría considerar separar la interfaz *ADC* y el *pipeline* y trabajar con dos frecuencias de reloj diferentes para poder trabajar en el *pipeline* del *DSP* a frecuencias menores y, por tanto, reducir el consumo de potencia.

El *pipeline* del *DSP* usa $2{,}800$ células lógicas de la FPGA, mientras que el *SOC* utiliza unas $11{,}000$. Una configuración de muestra con dos módulos de filtrado y el *SOC* utilizan $17{,}000$ células lógicas, lo que significa un $14\,\%$ de la *FPGA Altera Cyclone III EP3C120* como la utilizada para este proyecto.

5.3. Análisis del módulo *DFT*

Este módulo realiza la operación *DFT* de las muestras que entran al sistema para obtener el valor de amplitud de la señal en cada instante. Para ello, éste necesita recopilar un cierto número de muestras y, en tiempo real, operar con ellas y dar a su salida el resultado de dicha operación. El número de muestras a recopilar deberá coincidir con el factor de relación entre la frecuencia de la señal y la frecuencia de conversión del convertidor analógico/digital. En este caso el factor es de 4.

En la ecuación 5.1 se puede ver la Fórmula de Síntesis de la Transformada de Fourier de una señal discreta y aperiódica en el dominio temporal y que, debido a ello, se convertirá en una señal continua y periódica en el dominio de la frecuencia al realizar dicha operación.

$$x[n] = \sum_{k=0}^{N-1} a_k e^{j\frac{2\pi}{N}kn} \tag{5.1}$$

En la ecuación 5.2 se puede ver la Fórmula de Análisis de esta misma Transformada.

$$a_k = \frac{1}{N}\sum_{k=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{5.2}$$

La señal de entrada a este módulo es discreta en el tiempo y aperiódica, por lo tanto, como ya se ha explicado anteriormente, su Transformada de Fourier resultará en

una señal continua y periódica. Esto simplificará mucho las operaciones que se tendrá que realizar ya que en la suma de convolución de las muestras entrantes se anularán unas con otras por estar en contrafase y tener el mismo valor modular.

Si se estudia la ecuación de análisis presentada sobre estas líneas 5.2 y se desarrolla para una ventana de muestras de $N = 4$, que es la ventana que se utiliza en el sistema *VHDL*, se llega a la fórmula cerrada que se puede ver a continuación en la ecuación 5.3.

$$a_k = \frac{1}{4} \sum_{k=0}^{3} x[n] e^{-j\frac{\pi}{2}kn} \tag{5.3}$$

En dicha ecuación se puede ver que, como $N = 4$, resultará una suma de 4 sumandos a_0, a_1, a_2, a_3. Basándonos en las propiedades de la Transformada de Fourier se puede simplificar el cálculo de esta sumatoria ya que realmente se necesita conocer solo la amplitud de la señal en un determinado instante y no como variará ésta con el tiempo ya que de nada valdrá ésta posible interpretación debido a que el sistema varía aleatoriamente con él y cualquier predicción puede quedar obsoleta en el instante siguiente. Para conocer la amplitud de la señal en el instante "presente" bastará con conocer el valor de a_1. Si se desarrolla la ecuación anterior para valores fijos de $k = 1$ y $N = 4$ quedará la ecuación 5.4, donde $x[0]$, $x[1]$, $x[2]$ y $x[3]$ son las muestras de la señal procedente del convertidor analógico/digital en los instantes $t = 0$, $t = 1$, $t = 2$ y $t = 3$.

$$a_1 = \frac{1}{4} \left(x[0] - jx[1] - x[2] + jx[3] \right) \tag{5.4}$$

Cómo el objetivo final es conocer la amplitud de la señal en un determinado instante, aún en esta ecuación queda información que no se necesita y que, de eliminarla, simplificaría los cálculos y en consecuencia disminuiría el tiempo de procesado y el área utilizada por el sistema. Este es el caso de la fase de la señal, que viene implícita en la ecuación anterior mediante la unidad imaginaria. Si se eleva al cuadrado toda la ecuación y luego se realiza la raíz cuadrada del resultado, se elimina dicha información y y se simplifican los cálculos. De la misma manera, al elevar al cuadrado el valor fraccionario del inicio de la ecuación $(1/4)$ quedaría $(1/16)$ lo que simplemente haría que el valor que se busca fuera mucho menor pero no cambiaría la forma de onda de la señal formada por los sucesivos valores que se obtendrían en posteriores resultados de este cálculo. Esto último sólamente se puede realizar ya que el fin es reconstruir la señal con los valores absolutos de las muestras obtenidos en este cálculo, y un factor numérico multiplicativo no ofrece información adicional que pueda influir.

Una vez se ha realizado las últimas simplificaciones descritas en el párrafo anterior, la ecuación del valor final que se busca quedaría de la siguiente manera 5.5, donde $x[0]$, $x[1]$, $x[2]$ y $x[3]$ son las muestras de la señal procedente del convertidor analógi-

co/digital en los instantes $t = 0$, $t = 1$, $t = 2$ y $t = 3$.

$$a_1 = (x[0] - x[2])^2 + (x[1] - x[3])^2 \tag{5.5}$$

La implementación del módulo puede verse en la *Figura 5.5*.

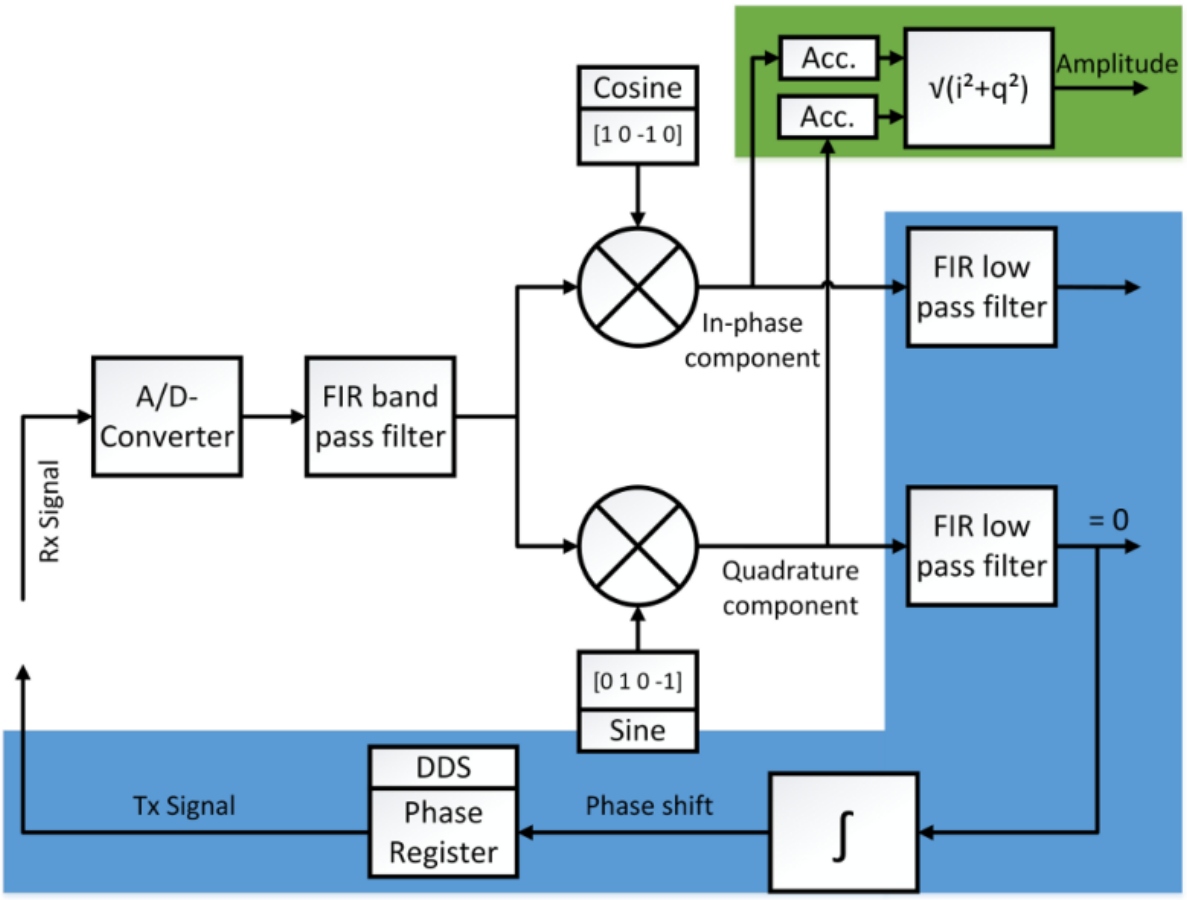


Figura 5.5: Diagrama de bloques de la implementación de la *DFT* en el sistema.

Este módulo trabaja con señales de 16 bits de ancho, el mismo ancho de resolución de muestra que el resto del sistema. Con cada ciclo de reloj de la *CPU* se realiza una *DFT* con una ventana de 4 muestras, esto quiere decir que al iniciar el sistema se tendría que esperar hasta el quinto ciclo de reloj de la *CPU* para obtener un resultado válido de la *DFT*. A partir de este quinto ciclo, cada ciclo siguiente se tendrá un dato válido a la salida. Esto va a ocurrir no solo al iniciar el sistema sino en cada uno de los instantes en que *discard_count* culmine su cuenta y descarte todas las muestras para cambiar de sensor. En este caso, *discard_count* toma el valor 8 y seguirá así siempre. Por tanto, cada 8 ciclos se cambia de dato o de sensor.

Una vez inicializado el sistema, se necesitan, como ya se ha explicado anteriormente, 4 ciclos para obtener un valor de *DFT* válido a la salida. Tras obtener este primer resultado válido, se tienen otros 4 ciclos para capturar el dato y guardarlo en memoria hasta que la señal *discard_count* llegue a su valor máximo. En este momento, se cambia de tipo de medida y se vuelven a necesitar otros 4 ciclos para obtener un valor de *DFT* válido a la salida. De nuevo se tienen 4 ciclos para capturar el dato y guardarlo en memoria y se pasa al siguiente sensor donde el proceso a seguir para la obtención de datos seguiría siendo el mismo.

Al aumentar la ventana de muestras de la *DFT*, el número de ciclos de espera inicial hasta que haya un dato válido a la salida del sistema aumenta, al pasar de 4 muestras por ventana a 8, el número de ciclos inicial hasta que haya un dato útil a la salida, se duplica, de la misma manera que lo hace el número de ciclos necesarios al final para vaciar el *búfer* de datos. En la ecuación 5.3 se puede ver la fórmula matemática a la que obedece el código escrito en *VHDL* para realizar una *DFT* con un ancho de ventana de 4 muestras.

Las posibles variaciones más sencillas que se puede realizar al sistema para experimentar si se podría en velocidad de procesado o en mejora de los resultados sería ampliar o disminuir el ancho de la ventana de muestras de esta operación. A simple vista, si se ampliara dicha ventana se puede entrever que la exactitud de la muestra mejora pero el sistema necesita una mayor velocidad de muestreo y se tardaría más tiempo en realizar la *DFT* ya que el número de sumandos de la ecuación 5.3 aumentaría al número de muestras de la ventana.

5.4. Análisis del módulo raíz cuadrada

La función de éste módulo es únicamente realizar la raíz cuadrada del dato que sale del módulo de la *DFT* para que la relación de proporcionalidad entre las muestras se conserve y, con ello, la proporcionalidad de la señal.

Este módulo trabaja con señales de 32 bits de ancho, el doble del ancho de resolución de muestra que el resto del sistema pero este ancho coincide con el número de bits que se producen a la salida del módulo predecesor para cada muestra, la *DFT*. Con cada ciclo de reloj de la *CPU* se realiza una *SQRT* del dato situado a la entrada del módulo y a la salida se produce un dato de 16 bits por muestra.

5.5. Análisis de los filtros *FIR*

Los filtros *FIR* tienen como función eliminar el ruido de la señal, acotar la energía que entra al sistema y maximizar la energía de la señal que entra al sistema en la parte

útil de ésta para el objetivo del proyecto. En este caso, la parte útil de la señal será en torno a los $98,625KHz$ en los que se encuentra la señal de excitación que se ha introducido al sistema. Debido a esta señal de excitación se tendrá una banda con bastante energía sobre la cual estarán presentes las modificaciones causadas por la capacidad presente en el entorno. Estas modificaciones son las que se quieren caracterizar ya que compondrán la información útil.

Para crear estos filtros se dispone de una herramienta muy potente que permite dimensionarlos en términos de tipo de filtro, frecuencia central, ancho de banda, etc. y, además, proporciona información a la hora de implementarlos en código *VHDL* sobre la latencia en ciclos de reloj, el área ocupada o la potencia consumida.

Esta herramienta la proporciona la propia *Altera Corp.*, se llama *Parametrize FIR Compiler*, disponible en el software *Quartus II* bajo el menú $Tools \rightarrow MegawizardPlugInManager \rightarrow Parametrize$ como se puede ver en la *Figura 5.6*.

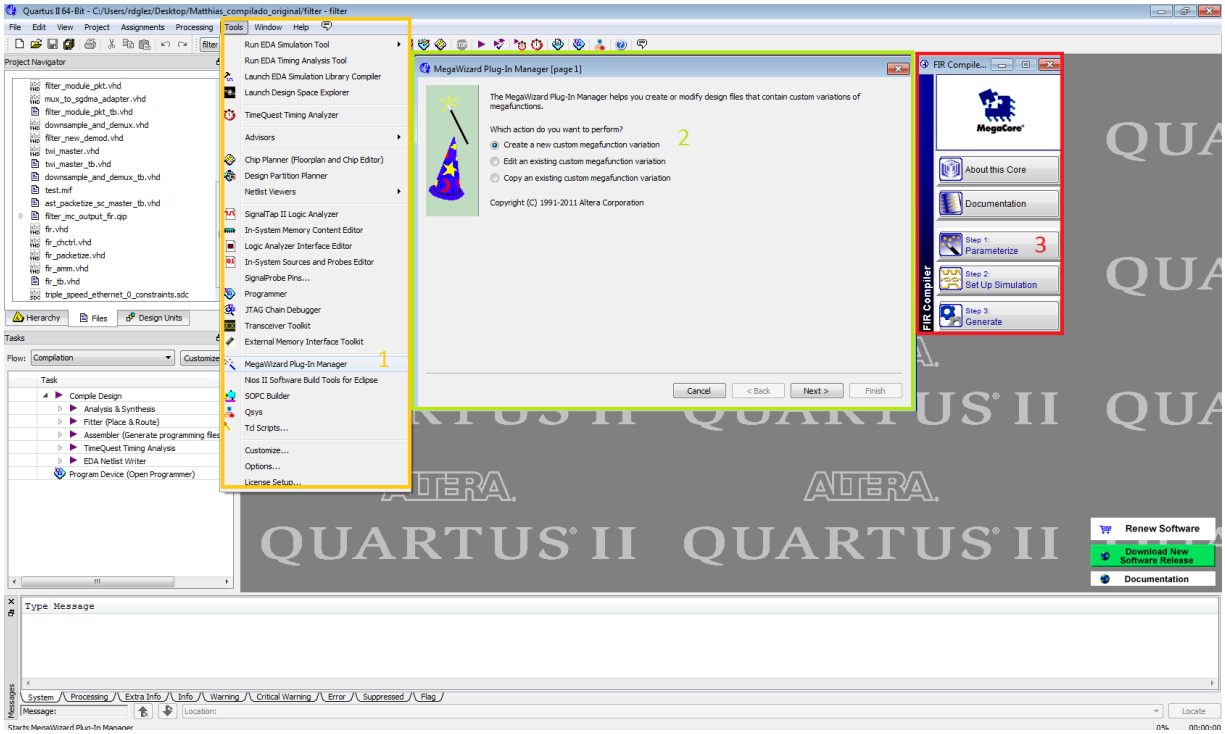


Figura 5.6: Vista del menú desplegable donde se sitúa la herramienta *Parametrize FIR Compiler*.

Una vez abierta la herramienta, en la *Figura 5.7* se muestra el aspecto que ésta presenta. A partir de los menús interactivos que ésta herramienta, se diseña el filtro *FIR* deseado y la herramienta genera una serie de archivos *VHDL* para implementarlo.

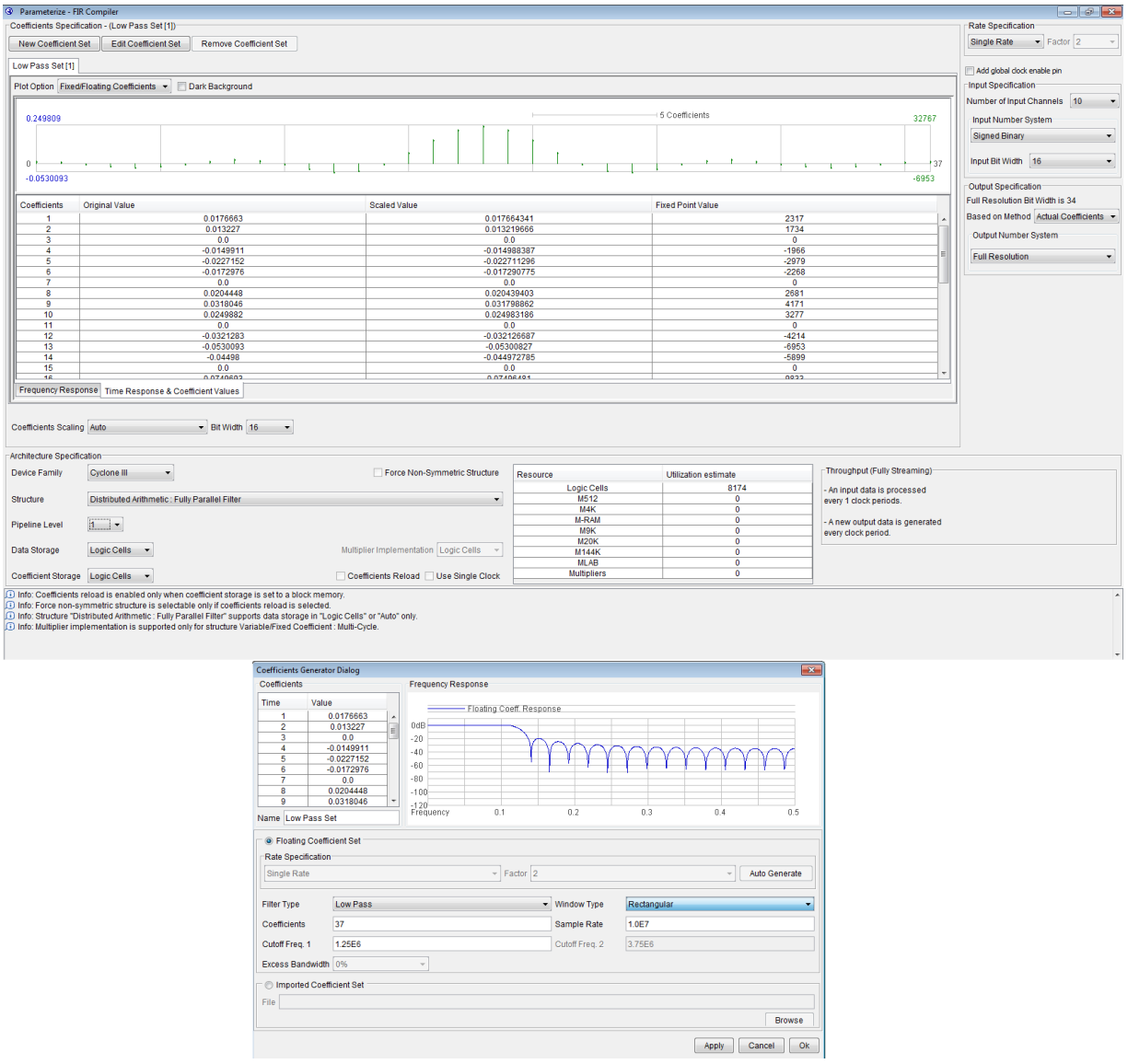


Figura 5.7: Vista de previa de la herramienta de diseño de filtros *FIR Parametrize FIR Compiler*.

5.6. Caracterización de la latencia del sistema

Para caracterizar la latencia del sistema se ha actuado siguiendo dos procesos totalmente diferenciados.

1. El primero de ellos ha sido la implementación en lenguaje *C++* de varios *Timers* localizados en la *CPU* y con acceso a las variables (los bits) de señalización de los buses. Estas señales situadas de manera estratégica en el sistema, dan la capacidad de controlar todo lo que está ocurriendo en él sin alterar, al menos en gran medida, la latencia de éste, ya que actúan fuera del *path* de datos del sistema.

2. El segundo proceso que se ha seguido para determinar el tiempo de procesado del sistema es la simulación de éste mediante *Testbenchs* que se encarguen de simular cada uno de los diferentes módulos de los que éste consta.

A continuación se detallan cada uno de los procesos llevados a cabo, indicando los problemas que han surgido, las soluciones adoptadas y los resultados obtenidos.

5.6.1. Estudio de la latencia mediante *Timers*

A grandes rasgos, éste método para conocer la latencia del sistema se basa en tres pasos fundamentales:

(a) Añadir una señal *IRQ* al *Bus Avalon-ST* del *Filter_Module*.

(b) Crear un *Timer* y aprender a utilizarlo.

(c) Modificar el software para controlar el *Timer* en función de su propósito.

Antes de comenzar con la descripción de los pasos seguidos en este punto del proyecto para estudiar la latencia del sistema, es importante definir la estructura de bloques formada por la descripción *VHDL* de los módulos dentro de la *FPGA* y con la cuál se trabajará, así como el camino que siguen los datos a través de estos módulos.

Para facilitar la comprensión de este estudio, a continuación, en la *Figura 5.8* se puede ver cada uno de los bloques del sistema, donde los módulos que son llamados por otros módulos superiores, se representan dentro de los mismos. Atendiendo a dicha figura, se describe el recorrido que cada muestra hace por el sistema antes de ser empaquetada y enviada al equipo cliente:

- En primera instancia, las muestras entran a través del *TopLevel*, el *dtpsc*, el *filter_module_pkt* y el *Filter_new* y llegan hasta el módulo *DFT* sin ser modificadas.
- A partir de aquí se empieza a trabajar con ellas. Primero se calcula su *Transformada de Fourier Discreta* en el tiempo con un ancho de ventana que siempre va a ser el mismo. Luego pasan al módulo *SQRT* para calcular su raíz cuadrada.
- Pasado este punto, salen del *Filter_new* y entran al módulo de filtrado, *filter_fir*, pasando por cada uno de los submódulos que se encuentran dentro de éste filtro siguiendo el orden de arriba a abajo según el orden en que figuran los bloques en el esquema visto. Por último, salen del módulo de filtrado para pasar al módulo *qf4_spi_sc*.
- Al salir de éste módulo, la muestra ya está preparada para ser empaquetada y enviada por la red hasta el equipo cliente.

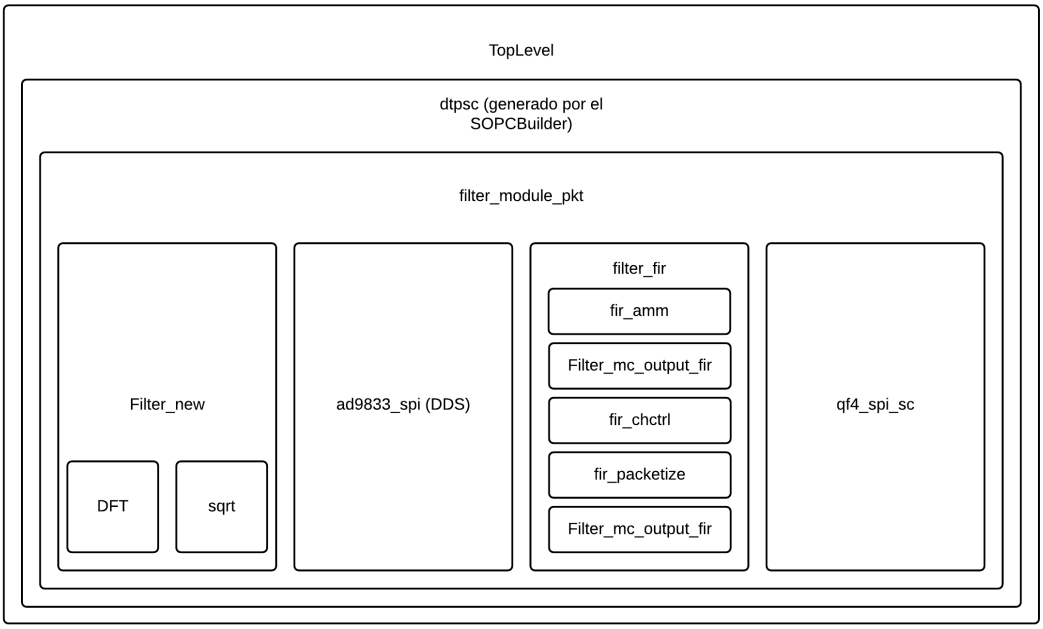


Figura 5.8: Diagrama de bloques del sistema *VHDL*.

El módulo *DDS* se encarga de generar la función coseno de $98,625KHz$ que se introduce a los sensores para excitar la *placa B* de estos y conseguir medir el valor de proximidad del objeto. Como se ha podido ver anteriormente en la *Figura 3.11*, el bloque *DDS* obtiene esta señal coseno a partir de la fase que resulta de integrar en el bloque de control de fase previo, la desviación de fase a la salida del *filtro paso bajo FIR* cuando a la entrada de este filtro se tiene el producto de la muestra de proximidad y el seno obtenido a partir del *DDS* en la muestra anterior.

Los pasos a dar son los siguientes:

1. Para el primer paso de este método, se modifica la descripción *hardware* del módulo *Filter_Module_pkt*, añadiendo una señal lógica de un bit de ancho e interconectarla desde éste módulo, a través de los diferentes módulos que atraviesa el dato hasta el final del procesado.

2. El segundo paso, es poner la señal al valor lógico 1 al inicio del procesado, justo después de digitalizar la señal. A su vez se modifica el dato con un valor previamente fijado, por ejemplo poniendo todos los bits del bus de datos al valor lógico 0.

3. El tercer y último paso a realizar en la descripción *hardware* para este método es

ir al final del procesado de señal y comprobar que el dato tiene todos sus bits a cero. La primera vez que esto se cumpla, significa que el primer dato que se ha puesto a cero ha llegado al final del procesado y como, coincidiendo con esto se ha activado la señal al valor lógico 1, se vuelve a resetear al valor lógico 0 ya que el tiempo del *timer* deberá finalizar ahí.

En la herramienta *SOPC*Builder dentro del *Quartus II* y la señal lógica que se ha utilizado para leer el estado del dato, se le asigna al *Bus Avalon-MM* como una señal IRQ. Esta acción hace que la señal de control interconecte la entidad principal *Filter_Module* con la *CPU* y el *Nios II*.

Una vez realizados estos cambios en el *hardware* es preciso modificar el software empotrado del sistema (que se ubicar en la memoria *ROM* del *Nios II*) y se crea el *timer* gestionado con la *API* que proporciona *Altera Corp*. Una vez definido el *timer*, sólamente es necesario modificar el código de la rutina principal para que al activarse la IRQ correspondiente, el *timer* se active o se desactive respectivamente.

Adicionalmente, se ha añadido una subrutina al programa principal para que muestre en el terminal de ejecución del sistema, el tiempo en μs que el dato ha tardado en recorrer el sistema. En la *Figura 5.9* se puede ver el tiempo que tarda el sistema en procesar cada dato.

Como se ha visto, los tiempos para cada muestra oscilan entre los $158,000\mu s$ y $196,000\mu s$, adoptando un valor pesimista de $200\mu s$ por muestra.

Aunque este valor es esperado y no supone un problema de retardo para el sistema, es necesario mejorarlo ya que si se tiene en cuenta que un instante en el sistema son 320 muestras (10 canales x 16 sensores x 2 muestras/sensor), para que un instante del sistema esté caracterizado al 100 %, se necesita que todas esas muestras sean procesadas y enviadas al servidor cliente para su interpretación. Bajo estas premisas, el tiempo necesario para caracterizar el sistema en un determinado instante de tiempo aumenta considerablemente hasta los $997,500\mu s$. Este valor se obtiene de sumar la latencia del path y el tiempo que tarda cada muestra siguiente en entrar al sistema hasta llegar al total de 320 muestras. Este tiempo es de $2,5\mu s$. Con todo esto se tiene que, $200,000\mu s + 2,500\mu s \times 319 muestras = 997,500\mu s/instante$.

El tiempo necesario para procesar las 320 muestras es menor a $1ms$, por tanto, la velocidad de respuesta del sistema por ahora cumple con las especificaciones. Pero, a este tiempo, aún se tiene que añadir el tiempo de transmisión necesario para transmitir los datos desde la *FPGA* hasta el servidor cliente del sistema de respuesta mediante una red *Ethernet* y el protocolo *UDP*. Este tiempo adicional, se ha calculado teóricamente ya que al ser un sistema ya probado y muy estándar, el error de cálculo está acotado. Para dicho cálculo se ha de tener en cuenta que, un paquete *UDP* tiene un tamaño estándar de 1504 Bytes, de los cuales, 1458 bytes son datos.

Puesto que un instante en el sistema se corresponde con 320 muestras (10 canales x 16 sensores x 2 muestras/sensor) y que cada muestra son 16 bits, es decir, 2 bytes. Un instante en el sistema serán entonces 640 bytes de información (320 muestras x 2 bytes), esto es, en cada paquete *UDP* caben al menos dos instantes. Rellenando el resto con la tercera muestra y siguiendo a continuación, y para una velocidad de la conexión *Ethernet* de 100MB/s, la velocidad teórica de los instantes en el sistema sería la que se expresa en la ecuación 5.6.

$$T_{medio_{Tx}} = 640B/_{instante} \cdot \frac{1s}{1 \cdot 10^8 B} \Rightarrow T_{medio_{Tx}} = 6,4\mu s/_{instante} \tag{5.6}$$

También hay que añadir el tiempo que necesita el sistema de respuesta del robot para actuar en consecuencia a los datos obtenidos. Este tiempo de respuesta del robot es desconocido ya que aún no se ha desarrollado esta parte del sistema y no es objetivo de este proyecto la estimación de su latencia.

Si se considera únicamente las pinzas duales, es decir, con la configuración vista en la introducción de este proyecto, el resultado obtenido se explica a continuación. Un instante en este nuevo sistema son 16 muestras (2 canales x 4 sensores x 2 muestras/sensor). Por lo tanto, el nuevo tiempo de incertidumbre es $200,000\mu s + 2,500\mu s \times 15 muestras = 237,500\mu s$ lo que supone una gran mejora con respecto a la configuración anterior pero que aún así no mejora lo necesario para las especificaciones del sistema.

Por lo tanto, si se utiliza el sistema con todos los canales y todos los sensores activos, 320 muestras, el tiempo requerido para tener un instante en el servidor cliente será de $997,500\mu s + 6,400\mu s = 1{,}003,900\mu s/instante$ aproximadamente.

Si se utiliza el sistema con la configuración de dos pinzas de cuatro sensores cada pinza, el resultado será mucho menor, de $200\mu s + 2,5\mu s \times 15 muestras = 237,5\mu s/instante$. De nuevo, si se añade el retardo al transmitir los datos por la red *Ethernet* (32 Bytes) a una velocidad de 100 MBytes/s, lo que resultará en $320,000ns$. Por tanto, el retardo final de un instante con la configuración de las dos pinzas de cuatro sensores cada pinza será de $237,500\mu s + 0,320\mu s = 237,820\mu s/instante$.

```
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

DTPSC/NiosII v0.6, HW rev5, mode ECTv2
Copyright (c) 2010-2013 IPR, Karlsruher Institut fuer Technologie
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Found QF4A512 rev. C2
Altera Triple Speed Ethernet rev. 901
Link up, 100 Mb/s full-duplex

Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 162us, 760ns.
Runtime: 0s, 0ms, 159us, 440ns.
Runtime: 0s, 0ms, 159us, 480ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 162us, 720ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 159us, 480ns.
Runtime: 0s, 0ms, 159us, 440ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 159us, 480ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 159us, 480ns.
Runtime: 0s, 0ms, 159us, 440ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 158us, 360ns.
Runtime: 0s, 0ms, 159us, 480ns.
Runtime: 0s, 0ms, 159us, 440ns.
Runtime: 0s, 0ms, 163us, 400ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 159us, 360ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 196us, 600ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 159us, 360ns.
Runtime: 0s, 0ms, 193us, 760ns.
Runtime: 0s, 0ms, 161us, 600ns.
Runtime: 0s, 0ms, 158us, 280ns.
Runtime: 0s, 0ms, 159us, 400ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 196us, 0ns.
Runtime: 0s, 0ms, 159us, 440ns.
Runtime: 0s, 0ms, 158us, 320ns.
Runtime: 0s, 0ms, 158us, 280ns.
Runtime: 0s, 0ms, 162us, 720ns.
```

Figura 5.9: Tiempo que tarda el dato en ser procesado según el *timer*.

5.6.2. Estudio de la latencia mediante *Testbenchs*

Este segundo método es, a priori, menos intrusivo ya que para implementarlo no se tiene que modificar la descripción *(VHDL)* del *hardware* del proyecto, ni modificar el *software* empotrado en *(C++)*. Por esta razón, será también más preciso.

El estudio de la latencia mediante *Testbenchs*, es la forma natural de simular los proyectos construidos mediante lenguajes de descripción de *hardware* como lo es *VHDL*.

En la *Figura 5.8*, mostrada y explicada en el método anterior de estudio de la latencia, se puede ver todo el sistema *VHDL* organizado en bloques. En este caso, se utilizará para explicar las partes del sistema que pueden ser simuladas mediante *Testbenchs* y cuáles no, debido a las restricciones adoptadas por *Altera Corp*.

Ninguno de los submódulos de filtros *FIR* es necesario simularlo ya que son generados de forma automática por la herramienta $Parametrize - FirCompiler$ de *Altera Corp*. Como se ha indicado, se accede a dicha herramienta desde el menú $Tools \Longrightarrow MegawizardPlugInManager \Longrightarrow CreatenewcustomMegafunctionVariation \Longrightarrow Parametrize$ y cuya interfaz principal se ha presentado anteriormente (*Figura 5.7*).

En esta herramienta, se muestra el valor en ciclos de reloj de la latencia para cualquiera de los filtros o combinaciones de ellos que se utilicen en el diseño. Para ello, es preciso seleccionar todos los parámetros de los filtros del sistema, el número de canales, el ancho en número de bits de cada canal, el tipo de filtro, el número de coeficientes (grado del filtro), el tipo de ventana de filtrado, etc.

Una vez seleccionados todos los parámetros de los filtros, se generan los ficheros necesarios en lenguaje *VHDL* (click en $Generate$) para llevar a cabo los filtros que se ha definido para el diseño. La herramienta informa en una ventana (como la que se muestra en la *Figura 5.10*) con información sobre los archivos generados y su contenido.

Para este caso, todo el submódulo de filtros (*Figura 5.8*) bajo el nombre de *filter_fir* y todos los submódulos internos a éste, han sido generados y parametrizados mediante esta herramienta. Los resultados obtenidos para ello han sido los que se muestran en la *Figura 5.11*. En dicha figura se puede ver el retardo total que introduce este módulo al sistema expresado en ciclos de reloj, que asciende a 17 ciclos. Como se ha visto que el reloj utilizado es de $125MHz$ lo que significa $8ns$ por ciclo, esto quiere decir que el retardo máximo introducido por todos los filtros *FIR* del sistema es de $retardo_{FIR} = 8ns \cdot 17 = 136ns$.

A este tiempo hay que añadirle el tiempo que en simulación se ve que tarda el dato en entrar al sistema y volver a salir de este. Al realizar la simulación del sistema, se ha podido ver que el tiempo que tarda un dato en entrar al sistema y volver a salir de éste, es de $\approx 184,48\mu s$ en el peor de los casos y que normalmente, este dato se sitúa entre

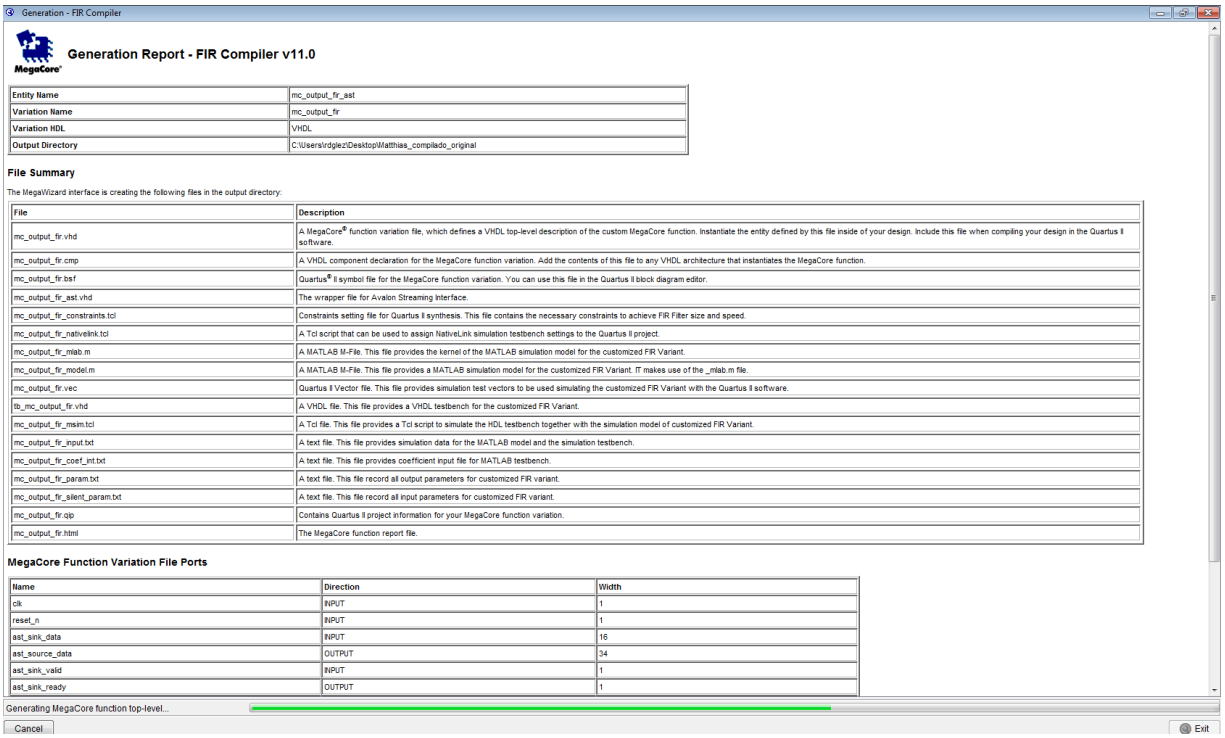


Figura 5.10: Ventana de generación de ficheros y descripción de cada uno de ellos mediante la herramienta $Parametrize - FirCompiler$.

los $144\mu s$ aproximadamente.

Esto se debe a que normalmente el sistema conmuta entre sensores de un mismo canal, pero cuando recopila todos los datos de los sensores de un determinado canal, el multiplexor tiene que cambiar de canal y vaciar completamente el búfer de datos para comenzar a leer los del nuevo canal. Como los datos entran al sistema cada $2,5\mu s$, se realiza el cálculo para saber el tiempo total en obtener las muestras de todos los sensores y determinar un instante completo del sistema, que es de $185\mu s + 2,5\mu s \times 319 muestras = 982,5\mu s/instante$.

Realizando esta misma operación pero para la configuración de 2 pinzas y 4 sensores por pinza, se obtiene el tiempo correspondiente: $185\mu s + 2,5\mu s \times 15 muestras = 222,5\mu s/instante$.

A este tiempo hay que añadir el tiempo que tarda el sistema en enviar los datos a través de la red *Ethernet*, que para la configuración de 160 sensores es de $6,4\mu s$ y para la configuración de las dos pinzas de cuatro sensores, es de $320ns$ como se ha calculado anteriormente.

También es necesario añadir a este tiempo, como ya se explicó con anterioridad, el tiempo que necesita sistema de respuesta del robot para actuar en consecuencia a los datos obtenidos.

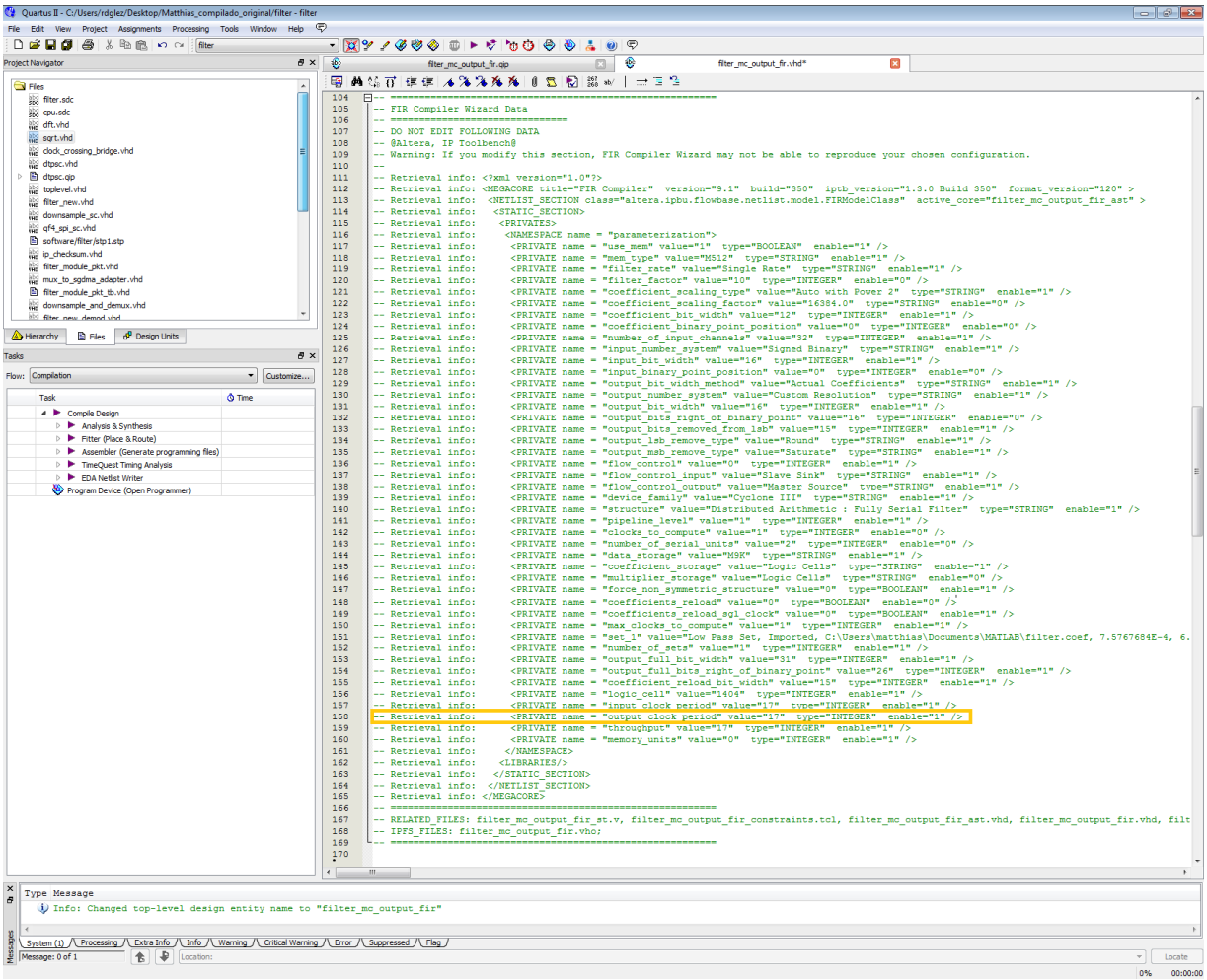


Figura 5.11: Retardo total introducido por el submódulo de filtros *FIR*.

Por tanto, según los datos obtenidos y, según como está el sistema construido, el tiempo total que el sistema necesita para procesar la información de un único instante es de $982,5\mu s + 6,4\mu s + 136ns = 980,036\mu s/instante$ para la configuración completa de los 160 sensores y de $222,5\mu s + 320ns + 136ns = 222,956\mu s/instante$ para la configuración de las dos pinzas de 4 sensores.

5.7. Alternativas de optimización el sistema

Una vez estudiado el sistema desde el punto de vista de su comportamiento temporal, se proponen algunos cambios que mejoren el funcionamiento y los resultados del sistema.

Las modificaciones propuestas son las siguientes:

(a) Modificar el multiplexado de las señales para disminuir las pérdidas de tiempo al llenar o vaciar los búfers.

(b) Modificar los parámetros de la *DFT* para ampliar la ventana de muestras.

(c) Modificar el módulo *SQRT* optimizando esta operación.

(d) Modificar los filtros *FIR* del sistema cambiando el tipo, el orden o el ancho de ventana para comprobar la mejoría o el empeoramiento de la fiabilidad de los datos y compararlo con el gasto temporal.

(e) Modificar el empaquetado de muestras.

(f) Añadir más búfers en paralelo y replicar la lógica de procesado de señal para procesar varios sensores o canales en paralelo y disminuir así la latencia del sistema ya que el uso en área de la *FPGA* no llega aún al 30 %, por lo que se podría replicar tres veces y disminuir la latencia a un tercio.

(g) Se puede tratar de cambiar la *FPGA* a otra con mayores prestaciones para aumentar aún más el replicado paralelo de la lógica o incluso para aumentar la frecuencia de procesado de la *CPU* y disminuir así la latencia del sistema.

5.8. Conclusiones

La ruta crítica del sistema se encuentra en el camino de datos que circulan desde el *ADC* hasta la interfaz de salida *Ethernet* pasando por los módulos de *DFT*, *SQRT* y *FIR* entre otros. El retardo total calculado está comprendido entre $184,48\mu s$ y $196,60\mu s$. Se ha seleccionado un retardo de $200\mu s$ para considerar un escenario más pesimista.

Estas aproximaciones son necesarias porque interesa calcular el retardo del sistema en el peor de los casos y como el tiempo que tarda el sistema en procesar los datos tiene una cierta variabilidad, es necesario adoptar un valor para el caso pesimista para calcular, tras el estudio de una gran cantidad de muestras, para que el sistema tenga una mayor tasa de probabilidad de funcionamiento dentro de la estimación realizada.

El objetivo de optimización del sistema se ha fijado para que éste sea capaz de anticipar cualquier posible amenaza para el ser humano. De esta manera, el sistema tiene que ser capaz de reaccionar a cualquier movimiento o posible movimiento o cambio de estado del entorno por muy brusco que sea. Como el tiempo mínimo de reacción

del ser humano en el mejor de los casos es de $1ms$ (para una persona entrenada que ejecuta un movimiento brusco deliberado), se ha fijado este tiempo como valor de referencia que se tendrá que superar para que el sistema cumpla las especificaciones de seguridad requeridas. El sistema cumple de forma ajustada las especificaciones requeridas. Por cuestiones de seguridad, antes de poner el sistema en funcionamiento en un entorno real, se ha estipulado que este tiempo debe ser al menos un orden de magnitud, es decir, hasta los $0,1ms$ o, lo que es lo mismo, $100\mu s$. Con este fin, se proponen las alternativas para optimizar el sistema que se han dado a conocer en el apartado anterior.

Es importante tener en cuenta también que a los valores de tiempo obtenidos tras el estudio realizado en este proyecto falta añadirle un sobrecoste temporal estimado del orden de *"nanosegundos (ns)"* que tardará el sistema inteligente de decisión que controlará en un futuro el brazo robótico en leer la información recibida y enviar una orden que actúe en consecuencia a ella a los motores del robot y lo que tarden éstos en llevar a cabo esa orden.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

Se ha realizado un amplio estudio del estado del arte en el campo de los sensores. Existen numerosos equipos científicos que realizan trabajos de investigación sobre nuevos métodos y nuevas tecnologías con el objeto de mejorar y optimizar este tipo de sensores, especialmente con la medición de la proximidad. No se descarta la introducción en un futuro próximo de nuevas tecnologías sensoras. Tal como se ha indicado, se trabaja focalizando los esfuerzos en dos líneas principales:

(a) Seguridad en el entorno, lo cual implica velocidad y amplio rango de visión espacial en lo que a medición de proximidad se refiere, y

(b) la precisión y forma del agarre.

La tendencia actual es el uso de materiales flexibles para la fabricación robusta y fiable de sensores y sistemas empotrados, que soporten el movimiento y la elasticidad. Con ello es posible abarcar una mayor posibilidad de aplicación del dispositivo y preparar el terreno para el objetivo final de fabricar una piel sintética fiable y elástica.

Todos los trabajos analizados buscan como fin último la seguridad en la colaboración en un entorno común de trabajo entre robots y seres humanos. Se ha visto que una cooperación humano-robot en la industria produce mayores beneficios al combinar la automatización y velocidad de una máquina con la supervisión y el manejo de eventos aleatorios presentada por la pericia y la experiencia de un buen empleado.

Para el proyecto en cuestión, el sensor consta de dos placas plano-paralelas que forman un condensador entre sí en el cual se mide la presión que se ejerce sobre él. La placa superior, a su vez, junto con el objeto externo forma condensador en el que se medirá la proximidad relativa del objeto y el sensor descrito.

Este sistema, sigue las leyes básicas de la física y el electromagnetismo. Es posible medir corrientes generadas en las placas y con ello determinar el estado del sistema. Se han realizado dos medidas, una en la placa superior para determinar la proximidad

del objeto de estudio y otra en la placa inferior para conocer la presión ejercida sobre el sensor, en caso de que el objeto se encuentre en contacto con éste. Las corrientes medidas se digitalizan y pasan por una serie de filtros y sistemas para poder determinar el valor de los datos de interés ya que la señal obtenida es débil, contiene un ruido varios órdenes superior a ella y es analógica e inestable. Es necesario estabilizarla y obtener la señal media de los datos. Todo esto se realiza en lo que se denomina el procesado de señal del sistema. Los datos son guardados en un *buffer* para posteriormente introducirlos en paquetes *UDP* y enviarlos mediante una red *Ethernet* hacia un equipo cliente donde se leerán y evaluarán para actuar en consecuencia.

Como se ha indicado a lo largo del documento, se ha realizado un modelo en *Matlab* para estudiar los datos recibidos de los sensores. Da visión general de los datos resultantes de la medida de proximidad del sistema y como varían en función de las interacciones entre el objeto externo y el sensor, si se tocan, se alejan o se acercan.

De la misma manera, este estudio sirve también para estudiar la función táctil del sensor ya que el condensador se rige por la mismas leyes físicas y electromagnéticas, y lo que continúa variando es la distancia entre placas. Si el objeto, ya está en contacto con el sensor de proximidad, puede ejercer una presión y acercar las placas del condensador que genera la medida táctil, o por el contrario, dejar de ejercerla y permitir que ambas se alejen hasta volver a obtener la distancia relativa entre placas normal del sistema.

Existe una gran diferencia entre ambos casos. En el condensador que mide la proximidad, una de las placas es conocida, la placa más externa del sensor, y la otra placa es desconocida. Se desconoce su material y su distancia máxima. En el condensador táctil, ambas placas son conocidas y se conoce también la distancia máxima a la que se encuentran. El modelo Matlab realizado permite tener un modelo en alto nivel que representa el comportamiento del sistema y facilita el análisis rápido de distintas situaciones de uso.

Igualmente se ha estudiado y propuesto mejoras en la latencia del sistema. Se han descrito dos métodos de cálculo, ya sea desde el dominio *hardware*, mediante la utilización de bancos de test, o desde el *software* empotrado prsente en el sistema, utilizando un conjunto de *timers* y el mecanismo de interrupciones presente en el microporcesador NIOS-II que controla el sistema de sensores.

La ruta crítica del sistema se encuentra en el camino de datos que circulan desde el *ADC* hasta la interfaz de salida *Ethernet* pasando por los módulos de *DFT*, *SQRT* y *FIR* entre otros. El retardo total calculado está comprendido entre $184,48\mu s$ y $196,60\mu s$. Se ha seleccionado un retardo de $200\mu s$ para considerar un escenario más pesimista. Estas aproximaciones son necesarias porque interesa calcular el retardo del sistema en el peor de los casos y como el tiempo empleado por el sistema en procesar los datos tiene una cierta variabilidad, es necesario adoptar un valor para el caso pesimista para calcular, tras el estudio de una gran cantidad de muestras, para que el sistema

tenga una mayor tasa de probabilidad de funcionamiento dentro de la estimación realizada.

El objetivo de optimización del sistema se ha fijado para que éste sea capaz de anticipar cualquier posible amenaza para el ser humano. De esta manera, el sistema tiene que ser capaz de reaccionar a cualquier movimiento o posible movimiento o cambio de estado del entorno por muy brusco que sea. Como el tiempo mínimo de reacción del ser humano en el mejor de los casos es de $1ms$ (para una persona entrenada que ejecuta un movimiento brusco deliberado), se ha fijado este tiempo como valor de referencia que se tendrá que superar para que el sistema cumpla las especificaciones de seguridad requeridas. El sistema cumple de forma ajustada las especificaciones requeridas. Por cuestiones de seguridad, antes de poner el sistema en funcionamiento en un entorno real, se ha estipulado que este tiempo debe ser al menos un orden de magnitud, es decir, hasta los $0,1ms$ o, lo que es lo mismo, $100\mu s$. Con este fin, se proponen las alternativas para optimizar el sistema.

6.2. Trabajos futuros

En este capítulo se presentan las líneas de trabajo futuras a desarrollar que completan el presente proyecto y dan validez y utilidad al trabajo desarrollado en él. A continuación se enunciará y describirá cada uno de los trabajos futuros y estudios se sugieren o en los que se ha pensado trabajar como complemento del sistema presentado en esta memoria y en los que se cree que el presente sistema puede ser de gran utilidad. Se resumen en las siguientes propuestas:

1. El principio de detección de este sistema sensor presenta algunos desafíos al relacionar la señal de proximidad con los valores de distancia, sin embargo, el alcance de aplicación es más amplio que sus homólogas ópticas. Estos sensores son capaces de cerrar una brecha de percepción de campo cercano existente en la robótica. El reto que se plantea para un futuro próximo es modelar eventos en el entorno cercano del robot, es decir, el rastreo de objetos y análisis de su movimiento para la predicción de parada y contacto. Esta predicción de parada y contacto tiene varias aplicaciones para la seguridad robótica y puede ser utilizada para implementar estrategias para evitar colisiones y así prevenir daños debido a choques causados por el movimiento del robot. Cuando el contacto es obligatorio, ya que se busca la sujeción de un objeto o algo similar, también puede ser útil, ya que mejora el control basado en la proximidad y el contacto del robot.

2. Se ha planteado también la opción de investigar cómo relacionar el movimiento del objeto detectado por los sensores y el tipo de objeto. El movimiento de las manos humanas es muy dinámico y armónico, mientras que el movimiento controlado por el robot suele estar limitado a aceleración constante y velocidad

objetivo, por tanto, los modelos de regresión indirecta pueden ser ajustados para estimar mejor la distancia y el tamaño.

3. La integración de los sensores en una pinza o sobre una plataforma robótica y su control de movimiento, así como el desarrollo de más aplicaciones y la exploración de objetos *pretouch* es igualmente un reto. En general, todas las capacidades que puedan ser desarrolladas e incluidas en una biblioteca de habilidades serían muy útiles para que puedan ser utilizados dentro de planes de ejecución y colaboración complejos y así hacer más completa las utilidades del sistema.

4. Otros trabajos futuros más avanzados que se proponen serían, por ejemplo, utilizar el sensor para diseñar una piel robótica sintética y un sensor que se pueda utilizar en pinzas de robot ya que ambas aplicaciones se basan en el mismo concepto y principio de detección.

5. En este proyecto se han investigado distintos tipos de espumas poliméricas con diferentes resultados para que el dieléctrico del condensador sea compresible. El modelo final muestra en condiciones de sobrecarga una gran diferencia a cuando se aplican sólo pequeñas fuerzas al sensor. Por supuesto, el modelo de sensor actual está todavía lejos de una célula de carga que haya sido construida expresamente para medir fuerzas. Continúa siendo un modelo que no incluye la temperatura y los efectos del envejecimiento. Además este modelo necesita ser probado para comprobar que las propiedades son aplicables a sensores cilíndricos.

6. Por otro lado, el modelo proporcionado utiliza un modelado de tiempo lineal invariante, lo que significa que los coeficientes no cambian con el tiempo, por ello, este modelo no puede tomarse como un modelo realista. Las espumas poliméricas usadas como resorte elástico también cambian sus propiedades mecánicas después de haber estado usando los sensores durante algún tiempo, por ello, el modelo será cada vez más inexacto con el paso de éste. Consecuentemente con esto se propone el desarrollo futuro de un método para detectar cambios significativos de los parámetros del sensor. Cuando se haya producido un cambio significativo en el parámetro, es necesario volver a identificar los parámetros del modelo.

7. Es posible también, conseguir que el sistema no sólo detecte un objeto en la proximidad del sensor, sino varios e incluso es posible lograr una característica de sensor asimétrico que ayude a localizar el objeto en el rango de detección del sensor. Por otro lado, se deberá profundizar también en el desarrollo de los algoritmos de agarre utilizando este tipo de sensores en las pinzas de robot por un lado y usando los sensores como una piel de robot por otro lado.

Bibliografía

[1] Raúl Martín Delgado et al. Desarrollo de un prototipo de sensor táctil de tres ejes. *Lectura de Tesis en Universidad Carlos III de Madrid. Departamento de Ingeniería de Sistemas y Automática*, 2009. URL http://hdl.handle.net/10016/8698.

[2] *KUKA Roboter GmbH*, 2017. URL https://www.kuka.com/.

[3] *KUKA KR 6-2, KR 16-2*. KUKA ROBOTER GMBH, Global Sales Center Hery-Park 3000 86368 Gersthofen Germany, 2002. URL http://roboticturnkeysolutions.com/robots/kuka/datasheet/kr_16.pdf.

[4] Carlos Yañez Duran, Gloria Castro León, and Jose Perez Quintanilla. Tecnología multitouch, presente y futuro. *Revista de investigación de Sistemas e Informática*, 10(1):75–85, 2013. URL http://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/download/5716/4947.

[5] Juan Luis Pedreño Molina, Antonio Guerrero González, Juan López Coronado, et al. Estudio de los sensores táctiles artificiales aplicados a la robótica de agarre. *Comité Español de Automática (CEA)*, 2000. URL http://intranet.ceautomatica.es/old/actividades/jornadas/XXI/documentos/ja00_003/ja00_003.pdf.

[6] Stefan Escaida Navarro, Maximiliano Marufo, Yitao Ding, Stephan Puls, Dirk Goger, Bjorn Hein, and Heinz Worn. Methods for safe human-robot-interaction using capacitive tactile proximity sensors. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1149–1154. IEEE, 2013. URL http://ai2-s2-pdfs.s3.amazonaws.com/af82/c2ed090df9e2087333af069c13e6ec0a465f.pdf.

[7] *Servo Electric 2-Finger-Parallel Gripper Type PG 70 Assembly and Operating Manual*. SCHUNK GmbH & Co. KG., 09.02 / 14.08.2012 / en edition, 2012. URL http://www.schunk.com/schunk_files/attachments/OM_AU_PG__EN.pdf. Assembly and Operating Manual.

[8] *Elektrisch 2-Finger-Parallelgreifer Universalgreifer Datasheet*, 2012. URL http://www.schunk.com/schunk_files/attachments/PG_70_DE.pdf.

[9] S. Tsuji, A. Kimoto, and E. Takahashi. A multifunction tactile and proximity sensing method by optical and electrical simultaneous measurement. *IEEE Transactions on Instrumentation and Measurement*, 61(12):3312–3317, Dec 2012. ISSN 0018-9456. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6295659.

[10] Pei-Hsuan Lo, Chitsung Hong, Shih-Hsiung Tseng, Jen-Hao Yeh, and Wei-leun Fang. Implementation of vertical-integrated dual mode inductive-capacitive proximity sensor. In *Micro Electro Mechanical Systems (MEMS), 2012 IEEE 25th International Conference on*, pages 640–643. IEEE, 2012. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6170268.

[11] Bo Zhang, Zemin Xiang, Siwei Zhu, Qiyi Hu, Yuanzhi Cao, Junwen Zhong, Qize Zhong, Bo Wang, Yunsheng Fang, Bin Hu, et al. Dual functional transparent film for proximity and pressure sensing. *Nano Research*, 7(10):1488–1496, 2014. URL http://www.thenanoresearch.com/upload/justPDF/0510.pdf.

[12] Masayuki Sohgawa, Akito Nozawa, Hokuto Yokoyama, Takeshi Kanashima, Masanori Okuyama, Takashi Abe, Haruo Noma, and Teruaki Azuma. Multimodal measurement of proximity and touch force by light-and strain-sensitive multifunctional mems sensor. In *IEEE SENSORS 2014 Proceedings*, pages 1749–1752. IEEE, 2014. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6985362.

[13] Cheng-Wen Ma, Li-Sheng Hsu, Jui-Chang Kuo, and Yao-Joe Yang. A flexible tactile and shear sensing array fabricated using a novel buckypaper patterning technique. *Sensors and Actuators A: Physical*, 2014. URL http://ac.els-cdn.com/S0924424714004117/1-s2.0-S0924424714004117-main.pdf?_tid=ddd37900-24fe-11e7-9827-00000aacb35f&acdnat=1492606452_7a9efc5246a4acdf08040183f1075605.

[14] Engin Cagatay, Philipp Kohler, Paolo Lugli, and Alaa Abdellah. Flexible capacitive tactile sensors based on carbon nanotube thin films. *Sensors Journal, IEEE*, 15(6):3225–3233, 2015. URL http://vm4ms9mb9q.scholar.serialssolutions.com/?sid=google&auinit=E&aulast=Cagatay&atitle=Flexible+capacitive+tactile+sensors+based+on+carbon+nanotube+thin+films&id=doi:10.1109/JSEN.2015.2404342&title=IEEE+sensors+journal&volume=15&issue=6&date=2015&spage=3225&issn=1530-437X.

[15] Andreas Braun, Reiner Wichert, Arjan Kuijper, and Dieter W Fellner. Capacitive proximity sensing in smart environments. *Journal of Ambient Intelligence and Smart Environments*, 7(4):483–510, 2015. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.714.842&rep=rep1&type=pdf.

[16] Kishor Kumar Sadasivuni, Abdullahil Kafy, Lindong Zhai, Hyun-U Ko, Seongcheol Mun, and Jaehwan Kim. Transparent and flexible cellulose nanocrystal/reduced graphene oxide film for proximity sensing. *small*, 11(8):994–1002,

2015. URL http://vm4ms9mb9q.scholar.serialssolutions.com/?sid=google&auinit=KK&aulast=Sadasivuni&atitle=Transparent+and+flexible+cellulose+nanocrystal/reduced+graphene+oxide+film+for+proximity+sensing&id=doi:10.1002/smll.201402109&title=Small+(Weinheim+an+der+Bergstrasse,+Germany)&volume=11&issue=8&date=2015&spage=994&issn=1613-6810.

[17] Junwoo Park, Tien Dat Nguyen, Uikyum Kim, Dong-Hyuk Lee, Canh Toan Nguyen, Hoa Phung, Jaedo Nam, Hyungpil Moon, Ja Choon Koo, and Hyouk Ryeol Choi. Design and fabrication of compliant proximity-tactile sensor using carbon micro coils. In *SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring*, pages 94302T–94302T. International Society for Optics and Photonics, 2015. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=2239773.

[18] Oluwaseun A Araromi, Samuel Rosset, and Herbert R Shea. High-resolution, large-area fabrication of compliant electrodes via laser ablation for robust, stretchable dielectric elastomer actuators and sensors. *ACS applied materials & interfaces*, 7(32):18046–18053, 2015. URL http://pubs.acs.org/doi/pdf/10.1021/acsami.5b04975.

[19] Xiaoli Zhao, Qilin Hua, Ruomeng Yu, Yan Zhang, and Caofeng Pan. Flexible, stretchable and wearable multifunctional sensor array as artificial electronic skin for static and dynamic strain mapping. *Advanced Electronic Materials*, 2015. URL https://www.researchgate.net/profile/Caofeng_Pan/publication/277977932_Flexible_Stretchable_and_Wearable_Multifunctional_Sensor_Array_as_Artificial_Electronic_Skin_for_Static_and_Dynamic_Strain_Mapping/links/557a294a08aeb6d8c0205dab.pdf.

[20] P Mittendorfer, E Yoshida, and G Cheng. Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot. *Advanced Robotics*, 29(1):51–67, 2015. URL http://www.tandfonline.com/doi/pdf/10.1080/01691864.2014.952493?needAccess=true.

[21] Fan Xia, Behraad Bahreyni, and Fabio Campi. Multi-functional capacitive proximity sensing system for industrial safety applications. In *SENSORS, 2016 IEEE*, pages 1–3. IEEE, 2016. URL https://www.researchgate.net/profile/Fan_Xia12/publication/312325643_Multi-functional_capacitive_proximity_sensing_system_for_industrial_safety_applications/links/587d5e1908ae9275d4e74e60.pdf.

[22] Hyo Seung Han, Junwoo Park, Tien Dat Nguyen, Uikyum Kim, Canh Toan Nguyen, Hoa Phung, and Hyouk Ryeol Choi. A highly sensitive dual mode tactile and proximity sensor using carbon microcoils for robotic applications. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 97–102. IEEE, 2016. URL https://www.researchgate.net/profile/

Tien_Nguyen73/publication/303466591_A_Highly_Sensitive_Dual_Mode_Tactile_Proximity_Sensor_Using_Carbon_Microcoils_for_Robotic_Applications/links/5763a7f208ae9964a16bad23.pdf.

[23] Alwin Hoffmann, Alexander Poeppel, Andreas Schierl, and Wolfgang Reif. Environment-aware proximity detection with capacitive sensors for human-robot-interaction. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 145–150. IEEE, 2016. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7759047.

[24] Il-Joo Cho, Hyung-Kew Lee, Sun-Il Chang, and Euisik Yoon. Compliant ultrasound proximity sensor for the safe operation of human friendly robots integrated with tactile sensing capability. *Journal of Electrical Engineering & Technology*, 12(1): 310–316, 2017. URL http://www.jeet.or.kr/LTKPSWeb/uploadfiles/be/201611/09112016154345312625O.pdf.

[25] Andrea Cirillo, Pasquale Cirillo, Giuseppe De Maria, Ciro Natale, and Salvatore Pirozzi. Force/tactile sensors based on optoelectronic technology for manipulation and physical human–robot interaction. In *Advanced Mechatronics and MEMS Devices II*, pages 95–131. Springer, 2017. URL https://www.researchgate.net/profile/Andrea_Cirillo/publication/303738265_ForceTactile_Sensors_Based_on_Optoelectronic_Technology_for_Manipulation_and_Physical_Human-Robot_Interaction/links/5750178508aeb753e7b4a0ae/Force-Tactile-Sensors-Based-on-Optoelectronic-Technology=for-Manipulation-and-Physical-Human-Robot-Interaction.pdf.

[26] Yin Cheng, Ranran Wang, Haitao Zhai, and Jing Sun. Stretchable electronic skin based on silver nanowire composite fiber electrodes for sensing pressure, proximity, and multidirectional strain. *Nanoscale*, 9(11):3834–3842, 2017. URL http://pubs.rsc.org/en/content/articlepdf/2017/nr/c7nr00121e.

[27] Stephan Mühlbacher-Karrer, Ahmad Haj Mosa, Lisa-Marie Faller, Mouhannad Ali, Raiyan Hamid, Hubert Zangl, and Kyandoghere Kyamakya. A driver state detection system-combining a capacitive hand detection sensor with physiological sensors. *IEEE Transactions on Instrumentation and Measurement*, 66(4): 624–636, 2017. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7831367.

[28] Kourosh M Kalayeh, Alexi Charalambides, Sarah Bergbreiter, and Panos G Charalambides. Development and experimental validation of a non-linear, all-elastomer in-plane capacitive pressure sensor model. *IEEE Sensors Journal*, 17(2):274–285, 2017. URL http://vm4ms9mb9q.scholar.serialssolutions.com/?sid=google&auinit=KM&aulast=Kalayeh&atitle=Development+and+Experimental+Validation+of+a+Non-Linear,+All-Elastomer+In-Plane+Capacitive+Pressure+Sensor+Model&id=doi:10.1109/

JSEN.2016.2628200&title=IEEE+sensors+journal&volume=17&issue=2&date=2017&spage=274&issn=1530-437X.

[29] Fan Xia, Behraad Bahreyni, and Fabio Campi. Design of digital modules for capacitive proximity sensing system applications. In *Electrical and Computer Engineering (CCECE), 2016 IEEE Canadian Conference on*, pages 1–4. IEEE, 2016. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7726660.

[30] Ivan Graham. Matlab manual and introductory tutorials. *Bath University Computing Service, Bath, UK*, 2005. URL http://www.meteo.psu.edu/holocene/public_html/Mann/courses/ENNEC472SPR09/MatlabTutorial.pdf.

[31] *Nios II Classic Software Developers Handbook*. ALTERA, 101 Innovation Drive San Jose, CA 95134, nii5v2 edition, May 2015. URL https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2sw_nii5v2.pdf.

[32] *Cyclone III 3C120 Development Board Reference Manual*. ALTERA, 101 Innovation Drive San Jose, CA 95134, 1.4 edition, March 2009. URL https://www.altera.com/en_US/pdfs/literature/manual/rm_cycloneiii_dev_kit_host_board.pdf.

[33] *Cyclone III Development Kit User Guide*. ALTERA, 101 Innovation Drive San Jose, CA 95134, 1.4 edition, September 2010. URL https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_ciii_devkit.pdf.

[34] *Data Conversion HSMC Reference Manual*. ALTERA, 101 Innovation Drive San Jose, CA 95134, March 2008. URL file:///C:/Users/Richard/Downloads/data_conversion_hsmc_reference_manual.pdf.

[35] Pong P. Chu. *RTL hardware design using VHDL : coding for efficiency, portability, and scalability*. Wiley-Interscience, Hoboken, NJ, 2006. ISBN 0-471-72092-5; 978-0-471-72092-8. URL http://swbplus.bsz-bw.de/bsz250874865cov.htm;http://swbplus.bsz-bw.de/bsz250874865inh.htm.

[36] Pong P. Chu. *FPGA prototyping by VHDL examples : Xilinx Spartan-3 version*. Wiley-Interscience, Hoboken, NJ, 2008. ISBN 0-470-18531-7; 978-0-470-18531-5. URL http://swbplus.bsz-bw.de/bsz277383927cov.htm;http://www.gbv.de/dms/ilmenau/toc/537760342.PDF. Includes bibliographical references and index.

[37] Pong P. Chu. *Embedded SOPC design with NIOS II processor and VHDL examples*. John Wiley & Sons, Hoboken, N.J, online-ausg. edition, 2011. ISBN 978-1-283-28288-8. URL http://lib.myilibrary.com/detail.asp?id=328288.

[38] Eduardo Augusto Bezerra and Djones Vinicius Lettnin. *Synthesizable VHDL Design for FPGAs*. Springer, Cham, 2014. ISBN 978-331-90254-7-6. URL https://github.com/kratsg/ZynqDocumentation/blob/master/Eduardo%20Augusto%20Bezerra%2C%20Djones%20Vinicius%20Lettnin%20(auth.)-Synthesizable%20VHDL%20Design%20for%20FPGAs-Springer%20International%20Publishing%20(2014).pdf.

[39] Dirk Goeger, Matthias Blankertz, and Heinz Woern. A tactile proximity sensor. In *Sensors, 2010 IEEE*, pages 589–594. IEEE, 2010. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5690450.

[40] *Quartus Prime Standard Edition Handbook Volume 1: Design and Synthesis*. ALTERA, 101 Innovation Drive San Jose, CA 95134, 15.1.0 edition, November 2015. URL https://www.altera.com/en_US/pdfs/literature/hb/qts/qts-qps-handbook.pdf.

[41] *Quartus II Handbook Volume 2: Design Implementation and Optimization*. ALTERA, 101 Innovation Drive San Jose, CA 95134, 14.0.0 edition, June 2014. URL https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/qts/qts_qii5v2.pdf.

[42] *Quartus II Scripting Reference Manual For Command-Line Operation & Tool Command Language (Tcl) Scripting*. ALTERA, 101 Innovation Drive San Jose, CA 95134, mnl-q2101904-9.1.1 edition, July 2013. URL https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/tclscriptrefmnl.pdf.

[43] *DATA SHEET QF4A512 4-Channel Programmable Signal Converter (PSC)*. Quickfilter Technologies, Inc, 1024 S. Greenville Ave Suite 100 Allen, TX 75002-3344, rev d8 edition, August 2009. URL http://www.quickfiltertech.com/files/QF4A512revD8.pdf.

[44] *MSP-430 Host Software Example for the QF4A512*. Quickfilter Technologies, Inc, 1024 S. Greenville Ave Suite 100 Allen, TX 75002-3344, rev a2 edition, July 2006. URL http://www.quickfiltertech.com/files/QFAN003%20-%20MSP-430%20Host%20Software%20Example%20for%20the%20QF4A512.pdf.

[45] *DAC Connections to the QF4A512 Programmable Signal Converter*. Quickfilter Technologies, Inc, 1024 S. Greenville Ave Suite 100 Allen, TX 75002-3344, rev a1 edition, September 2006. URL http://www.quickfiltertech.com/files/QFAN010-DAC%20Connections.pdf.

[46] *SOPC Builder User Guide*. ALTERA, 101 Innovation Drive San Jose, CA 95134, 1.0 edition, December 2010. URL https://www.altera.com/en_US/pdfs/literature/ug/ug_sopc_builder.pdf.

[47] *Designing with the Nios II Processor and SOPC Builder*. ALTERA, 101 Innovation Drive San Jose, CA 95134, 2010. URL http://ei.hust.edu.cn/teacher/zengyj/2013/EI0821361/Labs%20&%20Homeworks/Homeworks/Homework2_NII_SOPCBuilder_1Day_9_1_v2.pdf.

[48] *ModelSim AE Tutorial Software Version 6.4a*. Menthor Graphics Corporation, 8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777., 2015. URL http://courses.cs.washington.edu/courses/csep567/10wi/labs/modelsim_tut.pdf.

[49] Joachim Schlosser. Wissenschaftliche arbeiten schreiben mit latex. *Leitfaden für Einsteiger. Deutsch*, 3, 2008. URL https://books.google.es/books?hl=es&lr=&id=gAzCBQAAQBAJ&oi=fnd&pg=PA1&dq=Wissenschaftliche+Arbeiten+schreiben+mit+LATEX&ots=4K3UmbBPP4&sig=J3Xx7KgS6CUv6YSji0-QPTGrfjc#v=onepage&q=Wissenschaftliche%20Arbeiten%20schreiben%20mit%20LATEX&f=false.

[50] Herbert Voß. *Einführung in LATEX:[unter Berücksichtigung von pdfLATEX, XeLATEX und LuaLATEX]*. Lehmanns Media, 2012. URL https://books.google.es/books?hl=es&lr=&id=z793CwAAQBAJ&oi=fnd&pg=PR1&dq=Einf%7B%5C%22u%7Dhrung+in+LATEX:%5Bunter+Ber%7B%5C%22u%7Dcksichtigung+von+pdfLATEX,+XeLATEX+und+LuaLATEX%5D&ots=w529mJjRCa&sig=f5ulFGKOBM1DxXeofQKNI9n9k1A#v=onepage&q&f=false.

Glosario

Símbolos | A | B | C | D | E | F | G | I | L | P | R | S | T | U | V

Símbolos

Address Resolution Potocol

Es un protocolo de comunicaciones de la capa de enlace, responsable de encontrar la dirección de hardware que corresponde a una determinada dirección IP. Para ello se envía un paquete (ARP request) a la dirección de difusión de la red (broadcast, MAC = FF FF FF FF FF FF) que contiene la dirección IP por la que se pregunta, y se espera a que la máquina en cuestión responda (ARP reply) con la dirección *Ethernet* que le corresponde. Cada máquina mantiene una caché con las direcciones traducidas para reducir el retardo y la carga. Este protocolo permite a la dirección de Internet ser independiente de la dirección *Ethernet*, pero esto solo funciona si todas las máquinas lo soportan. 93, 102

Analogic to Digital Converter

Se trata de un convertidor analógico digital, que como su propio nomtre indica, convierte una señal analógica en su correspondiente señal digital. 93, 102

Application Programming Interface

La interfaz de programación de aplicaciones es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. 93, 102

BOOTstrap Protocol

También conocido como *protocolo de arranque*, es un protocolo de red utilizado por los clientes para obtener su dirección IP automáticamente. Normalmente se utiliza en el proceso de arranque de las computadoras o del sistema operativo y permite a los ordenadores sin disco obtener una dirección IP antes de cargar un sistema operativo avanzado. Históricamente ta sido utilizado por las estaciones de trabajo sin disco y por empresas para introducir una instalación preconfigurada de Windows en PCs recién comprados. Originalmente requería el uso de un disquete de arranque para establecer las conexiones de red iniciales, pero el protocolo se integró en la BIOS de algunas tarjetas de red y en muchas placas tase modernas para permitir el arranque directo desde la red. 93, 103

Carbon Micro Coils

Se trata de estructuras de carbono formando estructuras en bobina. 93, 103

Carbon NanoTubes

Se denominan nanotubos a estructuras tubulares (cilíndricas), cuyo diámetro es del tamaño del nanómetro. Existen nanotubos de muchos materiales, tales como silicio o nitruro de boro pero, generalmente, el término se aplica a los nanotubos de carbono. 93, 103

Cattode Ray Tute

El tubo de rayos catódicos es una tecnología que permite visualizar imágenes mediante un haz de rayos catódicos constante dirigido contra una pantalla de vidrio recubierta de fósforo y plomo. 93, 103

Central Process Unit

En electrónica, la unidad central de procesamiento es el hardware que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida. 93, 103

Digital Signal Processing

Se refiere al procesado digital de la señal, conjunto de técnicas y operaciones que se le pueden realizar a una señal para mejorar su exactitud y fiabilidad con el fin de obtener mejores comunicaciones y una tasa de error menor. 93, 104

Digital to Analogic Converter

Convertidor digital analógico, que como su propio nombre indica, convierte una señal digital en su correspondiente señal analógica. 93, 104

Direct Digital Synthesizer

Un DDS es un sintetizador digital de señal, esto es, un circuito que dado unos parámetros como una frecuencia, una forma de onda y un valor de voltaje, etc. es capaz de generar una señal digital que obedezca a éstos. 93, 104

Direct Memory Access

En electrónica, el módulo de acceso directo a memoria permite a cierto tipo de componentes de una computadora acceder a la memoria del sistema para leer o escribir sin tener que interactuar con el procesador principal. Muchos sistemas hardware utilizan estos tipos de módulos, incluyendo controladores de unidades de disco, tarjetas gráficas y tarjetas de sonido. Es una característica esencial en todos los ordenadores modernos, ya que permite a dispositivos de diferentes velocidades comunicarse sin someter al procesador a una carga masiva de interrupciones. 93, 104

Discrete Fourier Transform

Es un tipo de transformada discreta utilizada en el análisis de Fourier. Transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo. La DFT requiere que la función de entrada sea una secuencia discreta y de duración finita. Utilizar la DFT implica que el segmento que se analiza es un único período de una señal periódica que se extiende de forma infinita; si esto no se cumple, se debe utilizar una ventana para reducir los elementos espúreos del espectro. 93, 104

Domain Name System

Es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o redes privadas. Este sistema asocia información variada con el nombre de dominio asignado a cada uno de los participantes. Su función más importante es "traducir" nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente. 93, 104

Dynamic Host Configu- ration Protocol

Es un servidor que usa protocolo de red de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstas van quedando litres, sabiendo en todo momento quién ta estado en posesión de esa IP, cuánto tiempo la ta tenido y a quién se la ta asignado después. Así los clientes de una red IP pueden conseguir sus parámetros de configuración automáticamente. 93, 104

Electronic Design Automation

La automatización de diseño electrónico se refiere a una categoría de herramientas de software enfocadas en el proyecto, concepción, y producción de sistemas electrónicos, abarcando desde el proyecto de circuitos integrados hasta el desarrollo de placas de circuito impreso. 93, 104

Field Programmable Gates Array

Campo de vectores de puertas programables, es decir, una FPGA es un dispositivo que alberga un conjunto de bloques lógicos que pueden ser programados “in situ” mediante un lenguaje de descripción de hardware. 93, 105

Finite Impulse Response

En teoría de la señal, un sistema FIR es un sistema con respuesta finita al impulso. Esto es que, dada una señal o un impulso de entrada a ese sistema, la respuesta de este siempre será finita, llegando a ser cero en un tiempo determinado. 93, 105

First Input First Output

Estas siglas se utilizan para denominar un tipo específico de búffer o pila atendiendo al orden en que los datos entran y salen de ésta. En una pila FIFO, el orden seguido va a ser que el primer dato en llegar al búffer va a ser el primer dato en salir de él. También existen pilas LIFO (Last Input First Output) o FILO (First Input Last Output). 93, 104

Flexible Printed Circuit Board

Placa de circuito impreso flexible. 93, 105

General Purpose Input Output

Indica que un pin o un conjunto de pines son de propósito no especificado, esto significa que el programador puede darle el uso que él estime conveniente. Estos pines pueden ser de entrada, salida o funcionar de un modo mixto, entrada y salida indistintamente. 93, 105

Intelectual Property

Un bloque de Propiedad Intelectural es un bloque reutilizable y listo para ser utilizado en el diseño. Soporta diferentes tipos de configuraciones en función del procedimiento de desarrollo. 93, 105

Internet Engineering Task Force

Es una institución sin fines de lucro y abierta a la participación de cualquier persona, cuyo objetivo es velar para que la arquitectura de Internet y los protocolos que la conforman funcionen correctamente. Se la considera como la organización con más autoridad para establecer modificaciones de los parámetros técnicos tajo los que funciona la red. El IETF se compone de técnicos y profesionales en el área de redes, tales como investigadores, integradores, diseñadores de red, administradores, vendedores, entre otros. 93, 105

Internet Protocol

Es un protocolo de comunicación de datos digitales clasificado funcionalmente en la capa de red según el modelo internacional OSI. Su función principal es el uso bidireccional en origen o destino de comunicación para transmitir datos mediante un protocolo no orientado a conexión que transfiere paquetes conmutados a través de distintas redes físicas previamente enlazadas según la norma OSI de enlace de datos. 93, 105

Internet Protocol versión 4

Es la cuarta versión del protocolo de internet. Es uno de los protocolos centrales de los métodos estándares de interconexión de redes tasados en Internet, y fue la primera versión implementada para la producción de ARPANET, en 1983. IPv4 usa direcciones de 32 bits, limitándola a $2^{32} \times 2^{32} = 4{,}294{,}967{,}296$ direcciones únicas, muchas de las cuales están dedicadas a redes locales (LAN). Debido al

enorme crecimiento que ta tenido Internet unido al hecho de que hay desperdicio de direcciones en muchos casos, tace ya varios años se vio que escaseaban las direcciones IPv4. Las direcciones disponibles en la reserva global de IANA pertenecientes al protocolo IPv4 se agotaron oficialmente el lunes 31 de enero de 2011. Los Registros Regionales de Internet deben, desde ahora, manejarse con sus propias reservas, que se estima, alcanzarán hasta el 2020. 93, 105

Internet Protocol versión 6

Es la versión 6 del protocolo de internet y fue diseñada para reemplazar a su versión 4 que desde 2016 se ha estado implementado en la gran mayoría de dispositivos que acceden a Internet. Está sujeto a todas las normativas de la versión 4 y destinado a sustituirla. El nuevo estándar mejorará el servicio globalmente proporcionando entre otros, direcciones suficientes como para que los dispositivos móviles y telefónicos puedan tener sus direcciones IP propias y permanentes. IPv6 admite 340.282.366.920.938.463.463.374.607.431.
768.211.456 (340 sextillones de direcciones) cerca de $6,7 \times 10^{17}$ (670 mil trillones) de direcciones por cada milímetro cuadrado de la superficie de la Tierra. 93, 105

Internet Protocol address

Una dirección IP es un número que identifica, de manera lógica y jerárquica, a una interfaz en red (elemento de comunicación/conexión) de un dispositivo (computadora, tableta, portátil, smartphone) que utilice el protocolo IP, que corresponde al nivel de red del modelo TCP/IP. 93, 105

Interrupt ReQuest

Es una petición de interrupción, esto es, la petición de una suspensión temporal de la ejecución de un proceso, para pasar a ejecutar una subrutina de servicio de interrupción, la cual, por lo general, no forma parte del programa, sino que pertenece al sistema operativo o al BIOS. Una vez finalizada dicha subrutina, se reanuda la ejecución del programa. 93, 105

Linear Discriminant Analysis

Sistema de clasificación ideado por Ronal Fisher, disponible como función de Matlab. 93, 105

Low Noise Amplifier

se trata de un amplificador para señales de baja potencia que minimizan la degradación de la relación señal-ruido. 93, 106

PoliDiMetilSiloxano

También es conocido como Dimeticona. Es el polímero lineal del dimetilsiloxano. Pertenece al grupo de los compuestos de organosilicio, sustancias comúnmente conocidas como siliconas. Es transparente y, generalmente inerte, inocuo y no inflamable y se utilizar, por ejemplo, en lentes de contacto y artilugios médicos basta elastómeros. 93, 106

PolyVinyliDene Fluoride

El Fluoruro de polivinilideno un fluoropolímero termoplástico altamente inerte químicamente. Se suele emplear en condiciones que requieren mucha pureza, fortaleza y elevada resistencia a ácidos, bases y disolventes, a altas temperaturas, al envejecimiento y a los rayos ultravioleta. 93, 106

Printed Circuit Board

En electrónica, una *placa de circuito impreso* es la superficie constituida por caminos, pistas o tuses de material conductor laminadas sobre una tase no conductora. El circuito impreso se utiliza para conectar eléctricamente a través de pistas conductoras, y sostener mecánicamente, por medio de la tase, un conjunto de componentes electrónicos. Las pistas son generalmente de cobre mientras que la tase se fabrica generalmente de resinas de fibra de vidrio reforzada, cerámica, plástico, teflón o polímeros como la baquelita. También se fabrican de celuloide con pistas de pintura conductora cuando se requiere que sean flexibles. 93, 106

Random Access Memory

En electrónica, la memoria de acceso aleatorio se utiliza como memoria de trabajo en tiempo real para el sistema operativo, los programas y la mayor parte del software. En este tipo de memorias se cargan todas las instrucciones que ejecuta el procesador y otras unidades del computador. Se denominan *de acceso aleatorio* porque el tiempo de lectura o escritura es el mismo para cualquier posición de la memoria, no siendo necesario seguir un orden para acceder a la información de la manera más rápida posible. 93, 106

Read Only-Memory

La memoria de sólo lectura es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solamente la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía. Cabe recordar que esta es una memoria de acceso secuencial. 93, 106

Register-Transfer Level

En el diseño de circuitos digitales, el nivel de transferencia de registros es una abstracción de diseño que modela un circuito digital síncrono en términos del flujo de señales digitales entre los registros de hardware y las operaciones lógicas realizadas en esas señales. 93, 106

Request For Comments

Son una serie de publicaciones del grupo de trabajo de ingeniería de Internet que describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como protocolos, procedimientos, etc. y comentarios e ideas sobre estos. Cada RFC constituye un monográfico o memorando que ingenieros o expertos en la materia tan hecho llegar al IETF, el consorcio de colaboración técnica más importante en Internet, para que éste sea valorado por el resto de la comunidad. 93, 106

Routing Information Potocol

Es un protocolo utilizado por los routers o encaminadores para intercambiar información acerca de las redes de Internet a las que se encuentran conectados. Su algoritmo de encaminamiento está tasado en el vector de distancia, ya que calcula la métrica o ruta más corta posible hasta el destino a partir del número de equipos intermedios que los paquetes IP deben atravesar. El límite máximo de saltos en este protocolo es de 15, de forma que al llegar a 16 se considera una ruta como inalcanzable o no deseable. A diferencia de otros protocolos, éste es un protocolo litre es decir que puede ser usado por diferentes routers y no únicamente por un solo propietario con uno como es el caso de EIGRP en sistemas Cisco. 93, 106

SQuare RooT

En español “raíz cuadrada”. En éste documento este acrónimo hace referencia al módulo que realiza dicha operación sobre la señal de datos en el sistema. 93, 106

Serial Peripheral Interface

El interfaz periférico serie es un protocolo estándar para tuses que ta sido diseñado principalmente para la comunicación entre circuitos integrados. 93, 106

Support Vector Machine

Las máquinas de soporte vectorial, máquinas de vectores de soporte o máquinas de vector soporte (Support Vector Machines, SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T. 93, 107

System On Chip

Un sistema en chip es un circuito integrado compuesto por sensores, un procesador y actuadores que ta sido diseñado y fabricado para la realización de una serie de tareas específicas. 93, 106

System On Programmable Chip

Se denomina sistema en chip programable al conjunto formado por una FPGA donde se localiza la memoria y los elementos lógicos, y un núcleo procesador con propiedad intelectual (IP) para la implementación de un computador y un hardware específicos diseñado para aplicaciones de sistemas en chip (SOC). 93, 106

Tactile Pixel

Se refiere a cada una de las células que en un sensor táctil puede sentir individualmente una determinada presión que puede ser medida sin tener en cuenta la presión ejercida en las células sensoras de presión adyacentes. 93, 107

Testing Workbench

Es un modelo virtual que simula un sistema de señales dentro del cual poder ubicar el hardware que se desea testear y así poder comprobar cómo se comporta el sistema diseñado frente al conjunto de señales y así ser capaces de verificar el correcto funcionamiento del modelo diseñado. 93, 107

Transmission Control Protocol

Es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de datos compuesta por redes de computadoras, pueden usar este protocolo para crear conexiones entre sí a través de las cuales pueda enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto. 93, 107

User Datagram Protocol

Es un protocolo del nivel de transporte tasado en el intercambio de datagramas que permite el envío de éstos a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ta llegado correctamente, ya que no hay confirmación de entrega o recepción. 93, 107

VHSIC Hardware Description Language

Este nombre surge de la combinación de VHSIC y HDL, donde VHSIC es el acrónimo de Very High Speed Integrated Circuit y HDL es, a su vez, el acrónimo de Hardware Description Language. Se trata de un lenguaje de descripción de hardware tasado en ADA y desarrollado por el departamento de defensa de los Estados Unidos a inicios de los años 80, con el fin de realizar simulación de circuitos eléctricos digitales. Posteriormente se desarrollaron las herramientas de síntesis e implementación en hardware a partir de los archivos $.VHD$. 93, 107

A

ADC

Analogic to Digital Converter. 18, 36, 63–66, 80, 84, 93, 102, *Glosario:* Analogic to Digital Converter

API

Application Programming Interface. 74, 93, 102, *Glosario:* Application Programming Interface

ARP

Address Resolution Potocol. 40, 93, 102, *Glosario:* Address Resolution Potocol

B

BOOTP

BOOTstrap Protocol. 39, 93, 103, *Glosario:* BOOTstrap Protocol

C

C++

C++. 63, 71, 77, 93, 103, *Glosario:* C++

C++

Es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup con la intención de dotar al lenguaje de programación C de mecanismos que permitieran la manipulación de objetos. 63, 71, 77, 93, 103

Checksum

Checksum. 39, 41, 43, 93, 103, *Glosario:* Checksum

Checksum

Es una suma de verificación o suma de chequeo que en telecomunicación e informática se trata de una función *Hash* que tiene como propósito principal detectar cambios accidentales en una secuencia de datos para proteger la integridad de estos, verificando que no haya discrepancias entre los valores obtenidos al hacer una comprobación inicial, antes de la transmisión y otra final, tras la transmisión. La idea es que se transmita el dato junto con su valor *Hash*, de esta forma el receptor puede calcular dicho valor y compararlo así con el valor recibido. Si hay una discrepancia se pueden rechazar los datos o pedir una retransmisión. 39, 41, 43, 93, 103

CMCs

Carbon Micro Coils. 15, 93, 103, *Glosario:* Carbon Micro Coils

CNT

Carbon NanoTubes. 13, 93, 103, *Glosario:* Carbon NanoTubes

CPU

Central Process Unit. 62, 65, 68, 69, 71, 74, 80, 93, 103, 205, *Glosario:* Central Process Unit

CRT

Cattode Ray Tute. 2, 93, 103, *Glosario:* Cattode Ray Tute

D

DAC

Digital to Analogic Converter. 63, 93, 104, *Glosario:* Digital to Analogic Converter

DDS

Direct Digital Synthesizer. 65, 73, 93, 104, *Glosario:* Direct Digital Synthesizer

DFT

Discrete Fourier Transform. 66, 68, 69, 72, 80, 84, 93, 104, 150, 167, 168, 170, *Glosario:* Discrete Fourier Transform

DHCP

Dynamic Host Configu- ration Protocol. 39, 40, 93, 104, *Glosario:* Dynamic Host Configu- ration Protocol

DMA

Direct Memory Access. 64, 93, 104, *Glosario:* Direct Memory Access

DNS

Domain Name System. 39, 93, 104, *Glosario:* Domain Name System

DSP

Digital Signal Processing. 64–66, 93, 104, *Glosario:* Digital Signal Processing

E

EDA

Electronic Design Automation. 62, 93, 104, *Glosario:* Electronic Design Automation

Ethernet

Ethernet. 9, 39, 41–43, 45, 63–65, 74, 75, 78, 80, 84, 93, 95, 104, 197, 205, *Glosario:* Ethernet

Ethernet

Es un estándar de redes de área local para computadores con acceso al medio por detección de la onda portadora y con detección de colisiones. Su nombre viene del concepto físico de *ether*. *Ethernet* define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI. 9, 39, 41–43, 45, 63–65, 74, 75, 78, 80, 84, 93, 95, 104, 197, 205

F

FIFO

First Input First Output. 65, 93, 104, *Glosario:* First Input First Output

FIR

Finite Impulse Response. 35, 64, 69–71, 73, 77, 79, 80, 84, 93, 105, *Glosario:* Finite Impulse Response

FPCB

Flexible Printed Circuit Board. 15, 93, 105, *Glosario:* Flexible Printed Circuit Board

FPGA

Field Programmable Gates Array. 4, 6, 9, 10, 13, 17–19, 21, 41, 42, 62–64, 66, 72, 74, 80, 93, 105, 197, 199, 205, *Glosario:* Field Programmable Gates Array

G

GPIO

General Purpose Input Output. 9, 93, 105, *Glosario:* General Purpose Input Output

I

IETF

Internet Engineering Task Force. 39, 93, 105, *Glosario:* Internet Engineering Task Force

IP

Intelectual Property. 62, 93, 105, *Glosario:* Intelectual Property

IP

Internet Protocol. 39, 41, 43, 93, 105, *Glosario:* Internet Protocol

IPaddress

Internet Protocol address. 41, 93, 105, *Glosario:* Internet Protocol address

IPv4

Internet Protocol versión 4. 41, 93, 105, *Glosario:* Internet Protocol versión 4

IPv6

Internet Protocol versión 6. 41, 93, 105, *Glosario:* Internet Protocol versión 6

IRQ

Interrupt ReQuest. 72, 74, 93, 105, *Glosario:* Interrupt ReQuest

L

LDA

Linear Discriminant Analysis. 17, 93, 105, *Glosario:* Linear Discriminant Analysis

LNA

Low Noise Amplifier. 18, 93, 106, *Glosario:* Low Noise Amplifier

P

PCB

Printed Circuit Board. 19–21, 26, 30, 93, 106, *Glosario:* Printed Circuit Board

PDMS

PoliDiMetilSiloxano. 8, 16, 93, 106, *Glosario:* PoliDiMetilSiloxano

PVDF

PolyVinyliDene Fluoride. 15, 16, 93, 106, *Glosario:* PolyVinyliDene Fluoride

R

RAM

Random Access Memory. 64, 93, 106, 205, *Glosario:* Random Access Memory

RFC

Request For Comments. 39, 93, 106, *Glosario:* Request For Comments

RIP

Routing Information Potocol. 40, 93, 106, *Glosario:* Routing Information Potocol

ROM

Read Only-Memory. 74, 93, 106, *Glosario:* Read Only-Memory

RTL

Register-Transfer Level. 18, 21, 93, 106, *Glosario:* Register-Transfer Level

S

SOC

System On Chip. 66, 93, 106, *Glosario:* System On Chip

SOPC

System On Programmable Chip. 62, 64, 74, 93, 106, 198, 206, *Glosario:* System On Programmable Chip

SPI

Serial Peripheral Interface. 64–66, 93, 106, *Glosario:* Serial Peripheral Interface

SQRT

SQuare RooT. 69, 72, 80, 84, 93, 106, *Glosario:* SQuare RooT

SVM

Support Vector Machine. 15, 93, 107, *Glosario:* Support Vector Machine

T

Taxel

Tactile Pixel. 4, 20, 25, 27, 28, 32, 35, 93, 107, *Glosario:* Tactile Pixel

TCP

Transmission Control Protocol. 39, 40, 93, 107, *Glosario:* Transmission Control Protocol

Testbench

Testing Workbench. 11, 72, 77, 93, 107, *Glosario:* Testing Workbench

U

UDP

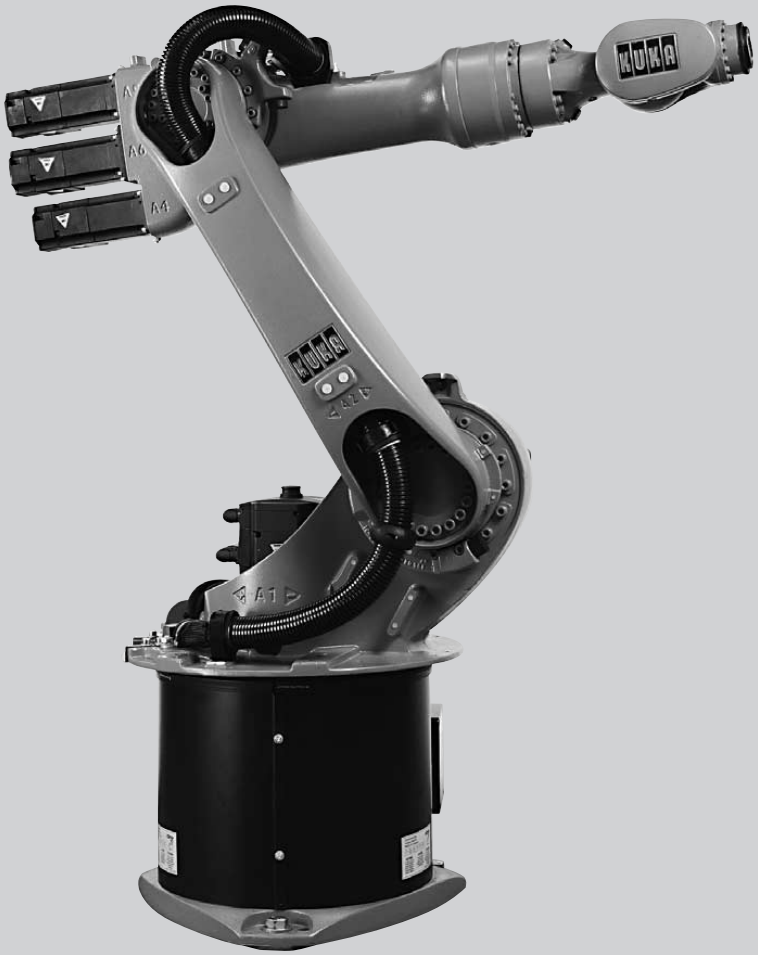
User Datagram Protocol. 39–45, 74, 75, 84, 93, 107, *Glosario:* User Datagram Protocol

V

VHDL

VHSIC Hardware Description Language. 10, 11, 18, 61–63, 67, 69, 70, 72, 73, 77, 93, 107, *Glosario:* VHSIC Hardware Description Language

Anexo A: Especificaciones técnicas del robot KUKA KR 16



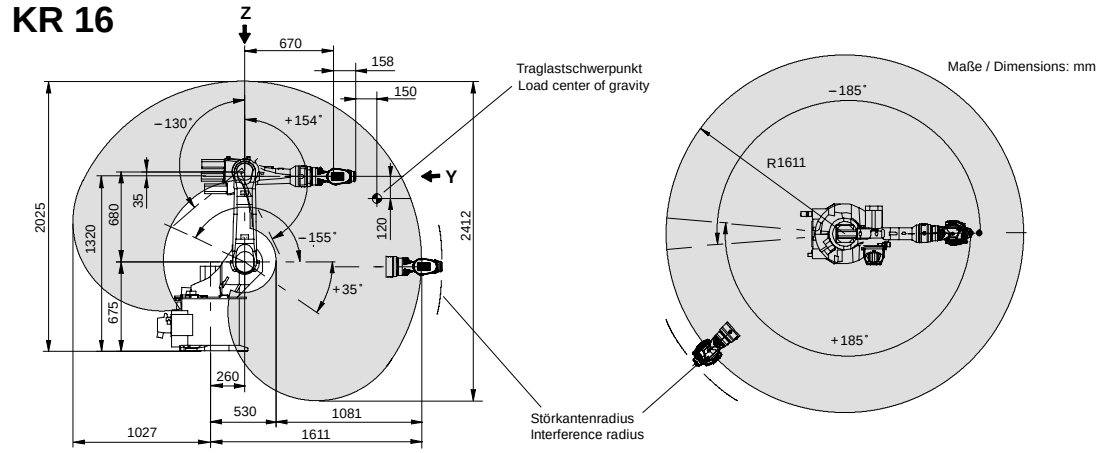
TECHNICAL DATA

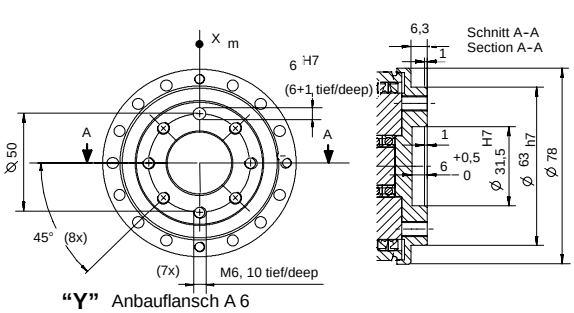
KR 16

WORKING IDEAS

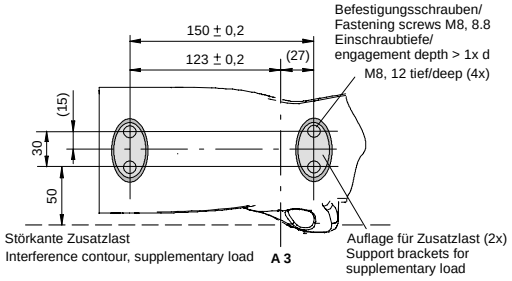
KUKA

KR 16





"Y" Anbauflansch A 6
Mounting flange A 6



"Z" Lochbild für Zusatzlast Arm
Attachment holes for supplementary load on arm

Traglast / Payload:	16 kg	
Zusatzlast Arm / Schwinge / Karussell Supplementary load on arm / link arm / rotating column:	10 kg / variabel/variable / 20 kg	
Max. Gesamtlast / Total distributed load:	46 kg	
Anzahl der Achsen / Number of axes:	6	
Handvariante / Wrist variant:	Zentralhand 16 kg / In-line wrist 16 kg	
Handvariante / Wrist variant:	Zentralhand 16 kg F/ In-line wrist 16 kg F (foundry)	
Anbauflansch A 6 / Mounting flange A 6:	DIN ISO 9409-1-A50	
Einbaulage / Mounting position:	Boden, Wand, Decke / Floor, wall, ceiling	
Wiederholgenauigkeit / Repeatability:	± 0,1 mm	
Steuerung / Controller:	KR C2	
Gewicht (ohne Steuerung) ca. / Weight (excl. controller) approx.:	235 kg	
Arbeitsraumvolumen / Work envelope volume:	14,5 m^3 1)	
Achsdaten / Axis data:	**Bereich (Software) / Range (software)**	**Geschwindigkeit / Speed**
Achse / Axis 1 (A 1)	± 185° 2)	156° /s
Achse / Axis 2 (A 2)	+ 35° /- 155°	156° /s
Achse / Axis 3 (A 3)	+ 154° /- 130°	156° /s
Achse / Axis 4 (A 4)	± 350°	330° /s
Achse / Axis 5 (A 5)	± 130°	330° /s
Achse / Axis 6 (A 6)	± 350°	615° /s

1) Bezogen auf Schnittpunkt Achse 4/5. / Referred to intersection of axes 4 and 5.

2) Einschränkung des Bewegungsbereiches bei Wandmontage / Limitation of the range of motion in case of wall mounting

D Antriebssystem elektro-mech. mit bürstenlosen AC-Servomotoren. / Drive system electromechanical, with brushless AC servomotors.

D Wegmeßsystem digital-absolut. / Position sensing system digital-absolute.

RoMeDBKR16-04.03.04

E KUKA Roboter GmbH, Germany

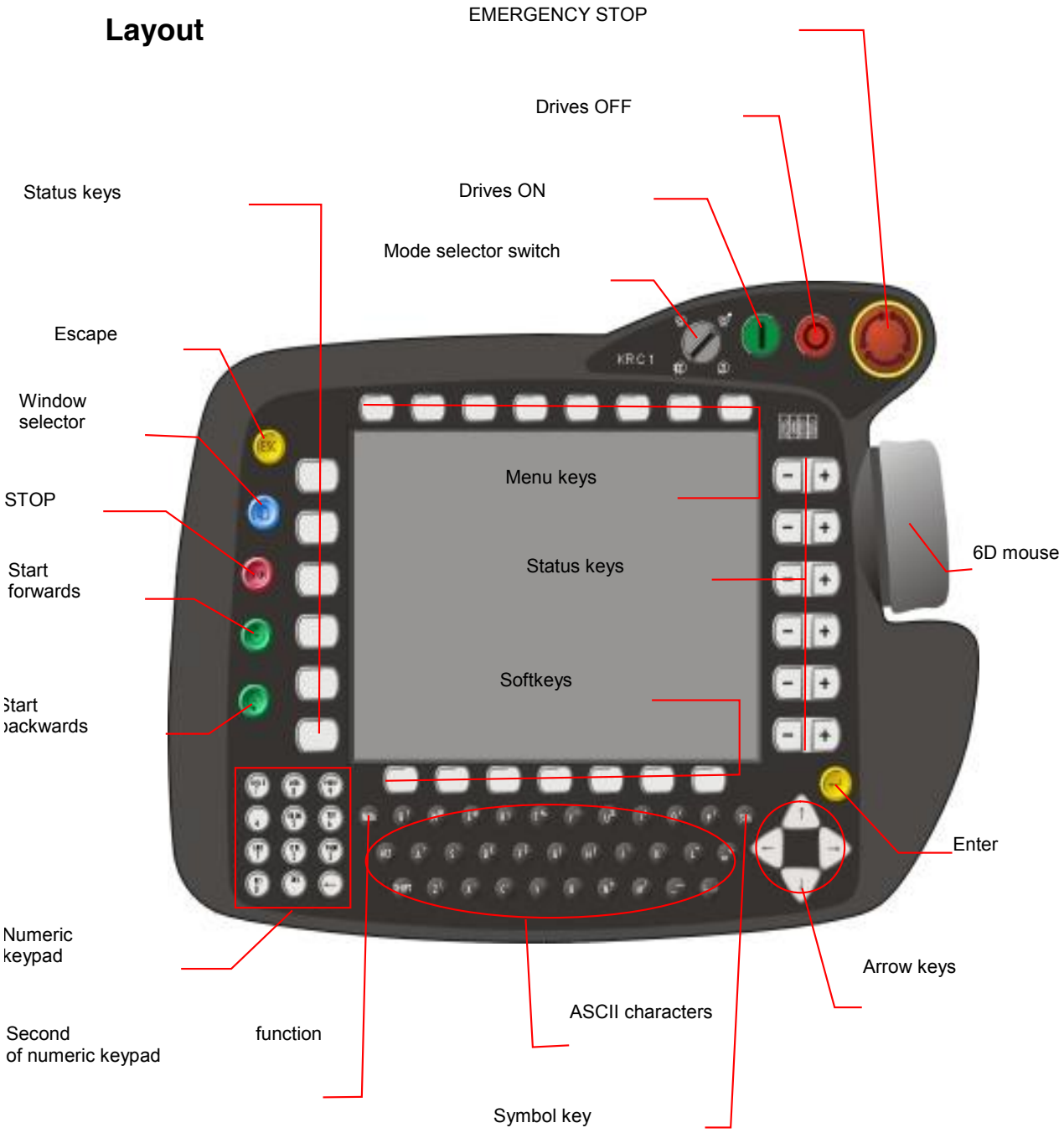
WM-Nr.841612-70/D+E/5/10.04

Anexo B: Panel de Control KUKA

KUKA Control Panel (KCP)

The KUKA control panel is the interface to the robot controller and to the robot. This document gives an schematic overview of the buttons and symbols. Do note however that this document does not replace the KUKA documentation. This document is valid for most KR C1 and all KR C2 robot controllers.

Layout



KUKA Control Panel (KCP)

Important function keys

An action that has been started can be aborted at any time using the **ESC key**.

It is possible to toggle between the program, status and message windows using the **window selection key**. The active window is indicated by a blue background.

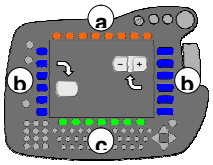
Pressing the **Stop key** stops a program that is running in automatic mode.

Pressing the **Program start forwards key** starts a program that has been selected.

If the **Program start backwards key** is pressed, the motion blocks are executed step by step towards the beginning of the program.

The **Enter key** is used, for example, to complete commands or to confirm entries in forms, etc.

The **menu keys** (**a**) are used to open menus in the menu bar. Navigation is accomplished with the aid of the arrow keys.

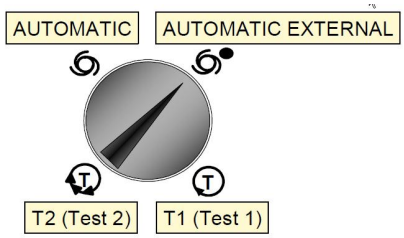


The **status keys** (**b**) are used for selecting operating options, switching individual functions and setting values.

The functions of the **softkeys** (**c**) are dynamically adapted to the current requirements, i.e. the assignment of the softkey bar is altered.

Mode selector switch

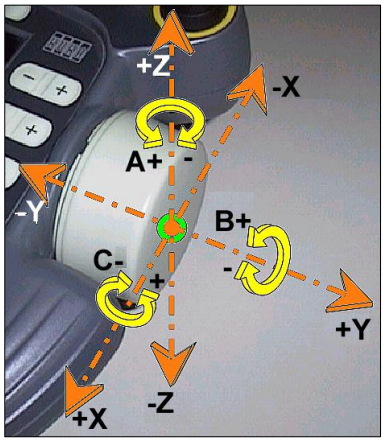
Mode selector switch



Mode selector switch	T1	T2	AUTOMATIC	AUTOMATIC EXTERNAL
Jogging using keys or Space Mouse HOV	250 mm/s Enabling switch (dead man function)	250 mm/s Enabling switch (dead man function)	Jogging not active	Jogging not active
Program execution POV	250 mm/s Enabling switch (dead man function) START key pressed	Prog. velocity Enabling switch (dead man function) START key pressed	Prog. velocity Drives ON START key --> PULSE	Prog. velocity Drives ON external External start

Jog mode and program execution

Jogging using the status keys



Jogging using the 6D mouse

Program mode:
Jogging deactivated

GO mode: Program execution is continued as long as the “Program start forwards” key remains pressed.

Single Step mode: For each motion command, the “Program start forwards” key must be released and pressed again.

Backward move: Appears automatically when the “Program start backwards” key is pressed.

Program velocity in % (POV):
Motion velocity of the robot during program execution.
☞ T1: max. 250 mm/s; T2: max. process velocity

Jog velocity in % (HOV):
Motion velocity of the robot during jogging using the jog keys or 6D mouse.
☞ T1 & T2: max. 250 mm/s

Status bar

1 – Numeric keypad

2 – Upper / lower-case letters

3 – Status Submit-Interpreter:

S	Submit interpreter deselected
S	Submit interpreter stopped
S	Submit interpreter running

Status Drives:

O	Drives not ready
I	Drives ready (600 ms)

Status Program:

R	No program is selected
R	The block pointer is situated on the first line of the selected program
R	A program has been selected and is currently being executed
R	The selected and started program has been stopped
R	The block pointer is situated on the last line of the selected program

4 – Selected program

5 – Current block number

6 – Current operating mode:

T1	Test 1: max. 250 mm/s
T2	Test 2: process velocity
AUT	Automatic
EXT	Automatic external

7 – Current override:

HOV	Hand override (hand jogging velocity)
POV	Program override (program velocity)

8 – Robot name

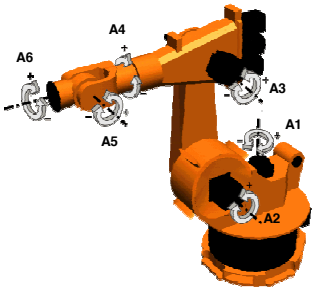
9 – Time

Coordinate systems

Axis-specific jogging:

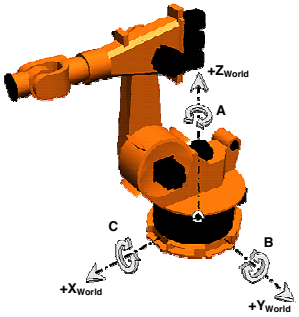
Each axis can be moved individually in a positive or negative direction.

Area of application: Rough method for addressing points. Only way of moving an unmastered robot. Moving away from a software limit switch.



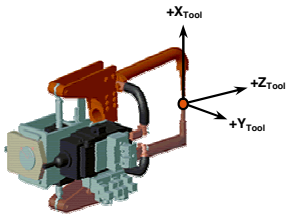
WORLD – Coordinate system

Fixed coordinate system whose origin is located at the base of the robot.
Area of application: Motion of the robot to any point in space using the keyboard or 6D mouse. Tool and BASE calibration.



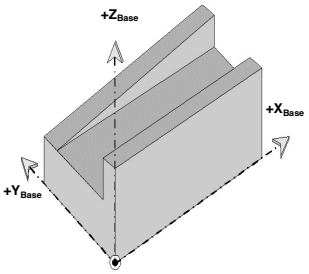
TOOL – Coordinate system

Coordinate system whose origin is located in the robot tool.
Area of application: Motion with the tool along a straight line if the orientation of the tool is inclined. Spot welding on the work piece. Gripper functions on the work piece. Motion of a work piece under an external TCP.

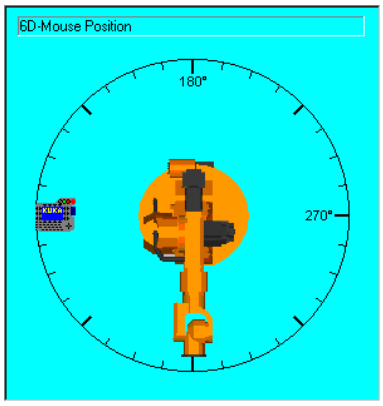


BASE – Coordinate system

Coordinate system whose origin is located in the work piece.
Area of application: Motion of work piece with defined BASE. “Mouse” jogging of work piece if the positive X axis of the BASE coordinate system is pointing towards the programmer.



Setting the mouse position



The menu item:

Configure
Jogging
Mouse position

is used by the operator to communicate his position to the controller.

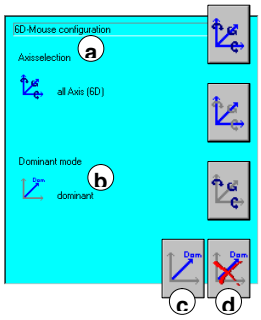
When switching to “Automatic” mode, the mouse position is automatically reset to 0 degrees!

Setting the mouse configuration

The menu item:

Configure Jogging Mouse configuration

is used by the operator to limit the degrees of freedom (**a**) and to switch the Dominant mode (**b**) on or off.



Axis-spec.: all axes 1 to 6
Cartesian: all directions X,Y,Z and all angles A,B,C

Axis-spec.: only main axes 1 to 3
Cartesian: only directions X,Y,Z

Axis-spec.: only wrist axes 4 to 6
Cartesian: only angles A,B,C

Dominant mode on (**c**)
Dominant mode off (**d**)

Anexo C:
Especificaciones técnicas de la pinza robótica PG 70

PG

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

Bus Capable. Flexible. High Performance Density.

PG Universal Gripper

Servo-electric 2-finger parallel gripper with highly precise gripping force control and long stroke

Field of Application

All-purpose, ultra-flexible gripper for great part variety and sensitive components in clean environments

Advantages – Your benefit

Electrically controlled gripper force adjustment for the delicate gripping of sensitive workpieces

Long stroke of 68 mm for flexible workpiece handling

Fully integrated control and power electronics for creating a decentralized control system

Versatile actuation options for simple integration into existing servo-controlled concepts via Profibus-DP, or CAN bus

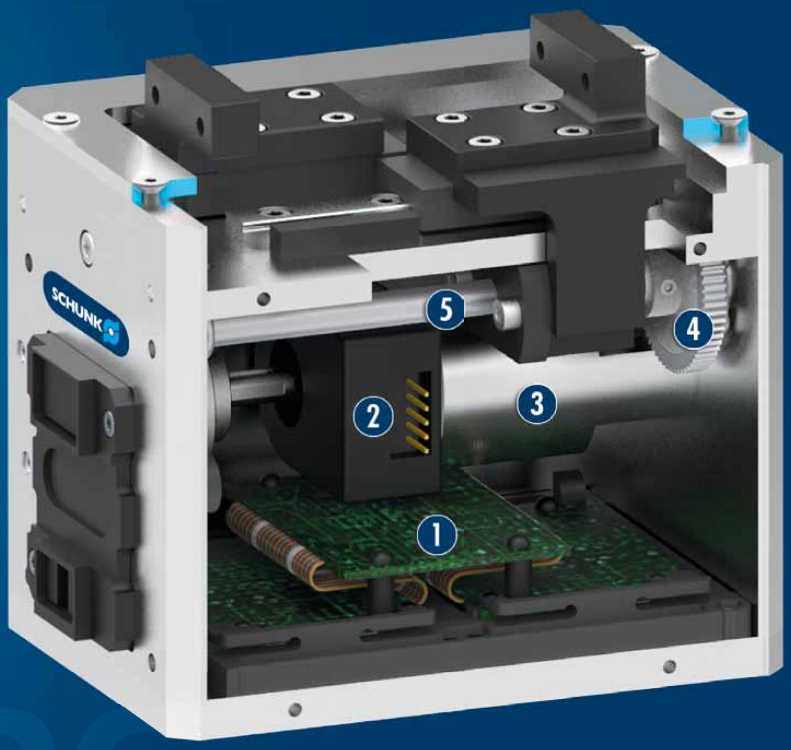
Standard connecting elements and integrated control concept for extensive combination possibilities with other mechatronic modules

Sizes	Weight	Gripping force	Stroke per jaw	Workpiece weight
Quantity: 1	1.4 kg	200 N	34 mm	1 kg

1712

Functional Description

The brushless servo motor drives the ball screw via a toothed belt drive.
The rotational movement is transformed into the linear movement of the base jaw by base jaws mounted on the spindle nuts.



① **Control electronics**
integrated control and power electronics for decentralized actuation of the servomotor

② **Encoder**
for gripper positioning and position evaluation

③ **Drive**
Brushless DC servomotor

④ **Gear mechanism**
Force transmission from the servomotor to the drive spindle

⑤ **Spindle**
Transforms the rotational movement into a linear movement

CAD data, operating manuals and other current product documents are available at **www.schunk.com**

PG

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

General Notes about the Series

Operating principle: Spindle drive

Housingmaterial: Aluminum alloy, coated

Base jaw material: Aluminum alloy, anodized

Actuation: servo-electric, via brushless DC servomotor

Warranty: 24 months (details, general terms and conditions and operating manuals can be downloaded at www.schunk.com)

Scope of delivery: Enclosed pack with centering sleeves, assembly and operating manual with declaration of incorporation, DVD with SCHUNK software and commissioning assistant, functional module for control via Siemens S7-300 / 400. A DMI or MMI electric cap is required for operation of the gripper. This is not included in the scope of delivery and must be ordered separately.

Gripping force: is the arithmetic total of the gripping force applied to each gripper jaw at distance P (see illustration).

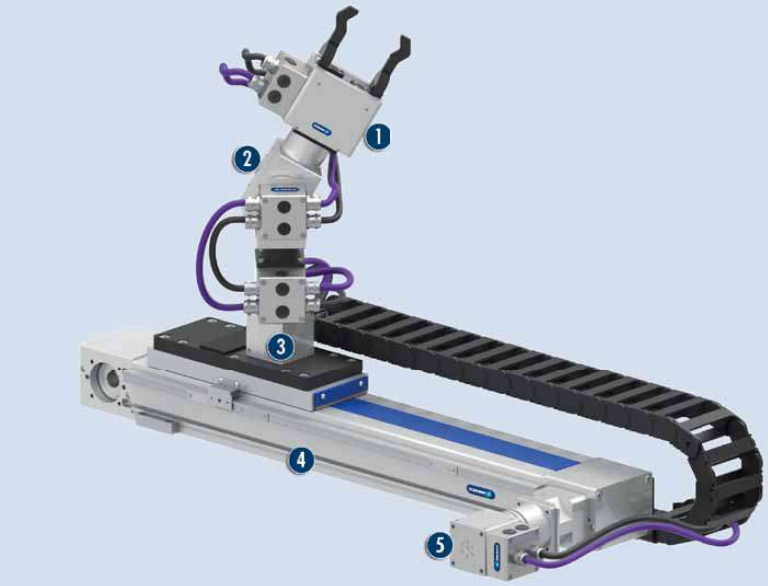
Finger length: is measured from the upper edge of the gripper housing in the direction of the main axis. The breach of the max. permitted finger length can bring higher abrasion.

Repeat accuracy: is defined as the spread of the limit position after 100 consecutive strokes.

Workpiece weight: is calculated for a force-fit connection with a coefficient of friction of 0.1 and a safety factor of 2 against slippage of the workpiece on acceleration due to gravity g. Considerably heavier workpiece weights are permitted with form-fit gripping.

Closing and opening times: Minimum closing and opening times are only the movement times of the base jaws at max. speed, max. acceleration without electrical restrictions (maximum current) and observance of the maximum permissible mass per finger.

Nominal currents: may be permanently applied. The information in the respective product documentation must be observed for all current levels beyond the rated current up to the maximum current.



Application example

Electrically powered gripper solution with linear axis and rotary modules for the handling of sensitive workpieces

1. PG Universal Gripper
2. PW Pan-Tilt Actuator
3. PR Rotary Unit electric
4. HSB Beta Belt-driven Axis
5. PDU Drive

SCHUNK offers more ...

The following components make the product PG even more productive – the suitable addition for the highest functionality, flexibility, reliability, and controlled production.

MMI Connection Cap

DMI Connection Cap

Power- / and Data Cable

Centering Sleeves

Beta electrical Linear Module

PRH Rotary Actuator

Finger Blanks

ERS Rotary Actuator

ERD Rotary Actuator

PW electrical Rotary Pan-Tilt Actuator

PR Rotary Unit electric

PDU Rotary Unit electric

ⓘ Further information regarding the products can be found on the following product pages or at www.schunk.com. Please contact us for further information: SCHUNK technical hotline +49-7133-103-2696

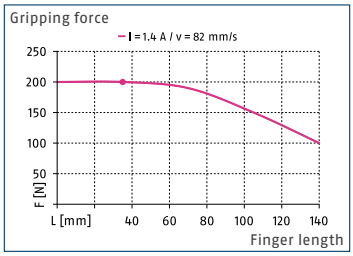
Options and special Information

Integrated electronics: The electrical control of the gripper takes place via the fully integrated control and power electronics. Therefore, no additional external control units are required for the module.
Easy integration: There is a varied range of interfaces available, such as Profibus-DP or CAN bus as methods of communication. This enables the assembly of industrial bus networks and ensures easy integration into existing control systems.
Connection caps DMI and MMI: The DMI or MMI connection caps are available for connection of the gripper to the voltage supply or superordinate control unit. They are not included in the scope of delivery and must be ordered separately.

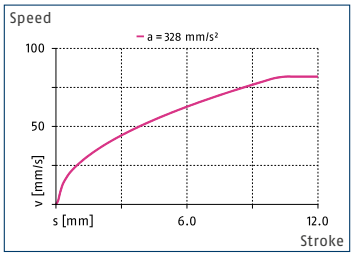
PG 70

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

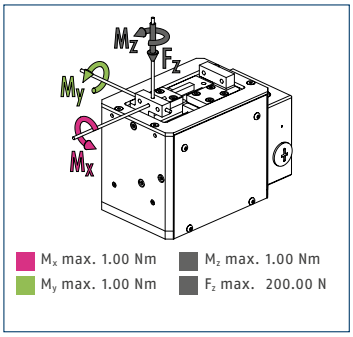
Gripping force



Speed



Finger load



ⓘ The specified torques and forces are static values, apply for each base jaw, and may occur simultaneously. M_y may occur in addition to the torque generated by the gripping force.

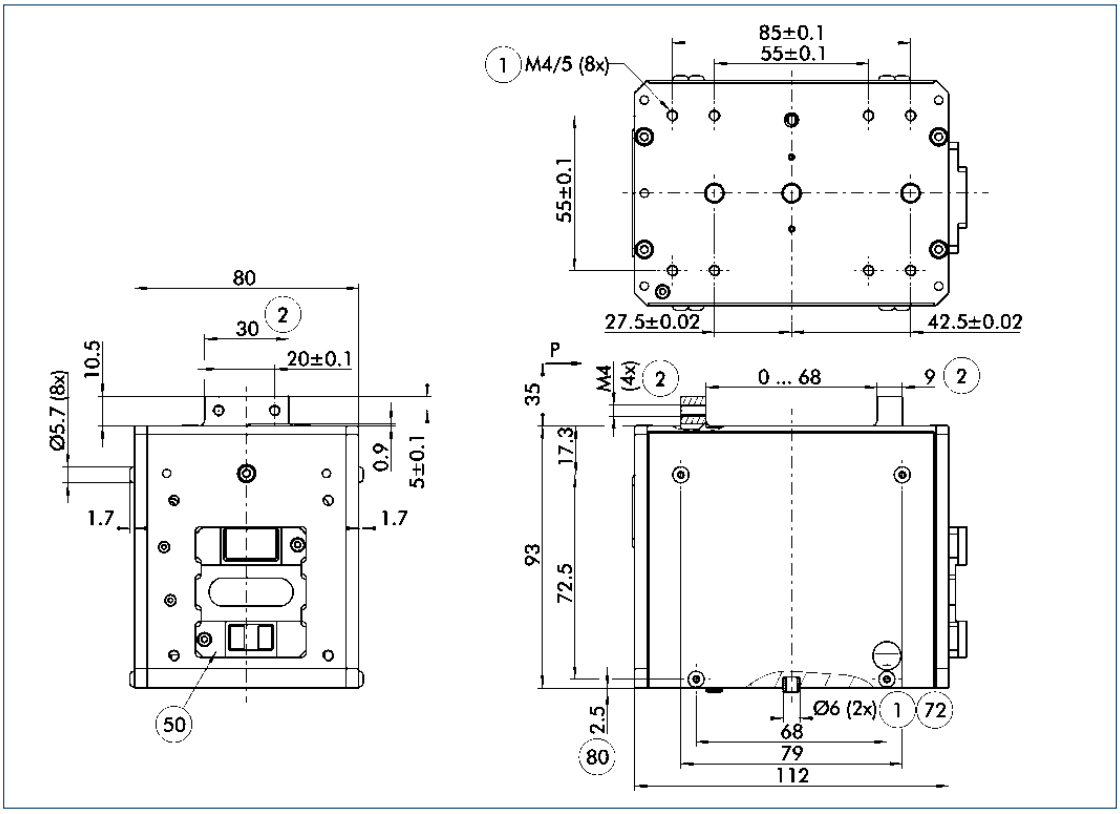
Technical data

Description		PG 70
ID		0306095
General operating data		
Stroke per jaw	[mm]	34
min. / max. gripping force	[N]	30/200
Recommended workpiece weight	[kg]	1
max. permitted finger length	[mm]	140
Repeat accuracy	[mm]	0.05
Closing- / opening time	[s]	1.1/1.1
max. speed	[mm/s]	82
max. acceleration	[mm/s²]	328
Weight	[kg]	1.4
min. / max. ambient temperature	[°C]	5/55
IP class		20
Electrical operating data		
Controller electronics		integrated
Nominal voltage	[V DC]	24
Nominal current	[A]	1.4
max. power supply	[A]	1.8
Communication interface		Profibus, CAN bus, Digital I/O
Profibus interface	[Mbit/s]	1.5
CAN interface	[Mbit/s]	1
Number of digital inputs/outputs		4/4
Parametrized interface		RS232

PG 70

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

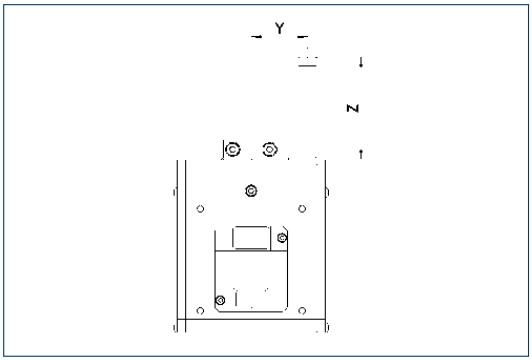
Main view

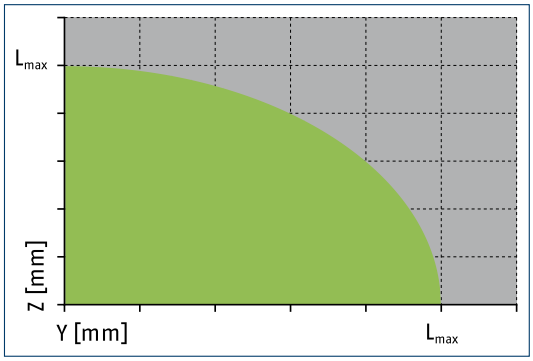


The drawing shows the basic version of the gripper with open jaws, without dimensional consideration of the options described below.

1 Gripper connection
2 Finger connection
50 Electrical connection
72 Fit for centering sleeves
80 Depth of the centering sleeve hole in the mating part

Maximum permitted finger projection





Permitted range

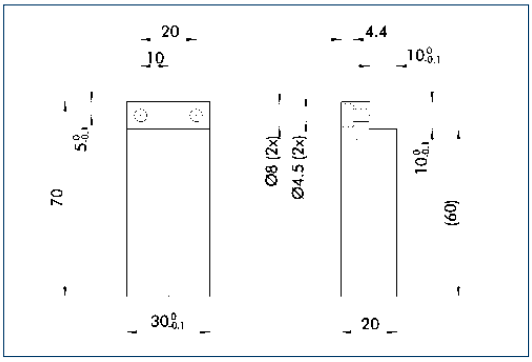
Inadmissible range

L_{max} is equivalent to the maximum permitted finger length, see the chart of technical specifications.

PG 70

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

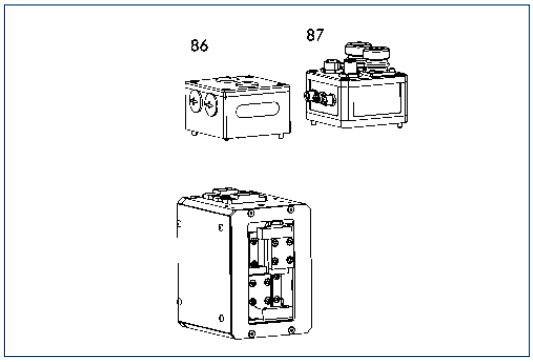
ABR-MPG-plus 70 finger blanks



Finger blanks for customized subsequent machining.

Description	ID	Material	Scope of delivery
Finger blanks			
ABR-PG 70	0307850	Aluminum	1

Connection caps



(86) DMI connection cap

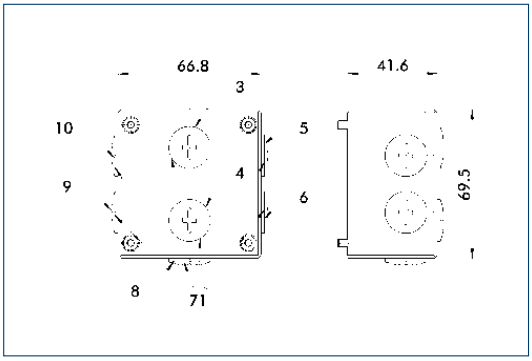
(87) MMI connection cap

The DMI or MMI connection caps are required for connection of the modules to the voltage supply or superordinate control unit. For the DMI the connection of the cable wires takes place via connection terminals. The MMI enables convenient connection via plug connector.

Description	ID	
Connection caps		
DMI 070-V05-B	0307732	
MMI 070-V05-E-CN	0307500	
MMI 070-V05-D-CN	0307501	
MMI 070-V05-E-PB	0307502	
MMI 070-V05-D-PB	0307503	

ⓘ Further information and accessories can be found in the following displays.

DMI connection cap

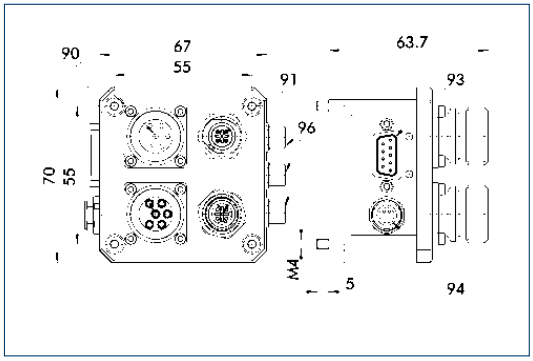


(71) M16x1.5 for cable guide penetrating screw connection

For the DMI the connection of the cable wires takes place via connection terminals. The DMI is prepared for Profibus and CAN bus communication interfaces.

Description	ID	
Connection caps		
DMI 070-V05-B	0307732	

MMI connection cap



(90) Connection power supply (logic / load)

(91) Connection fieldbus M12

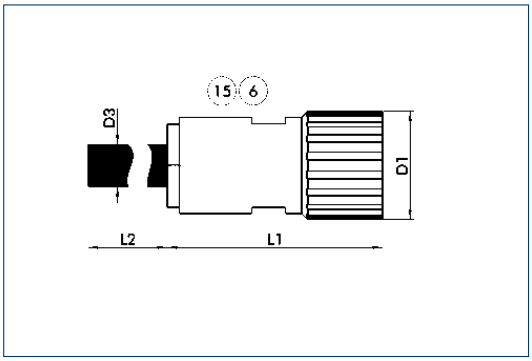
(93) Parametrized interface RS232

(94) Connection power supply service box (SSB)

(96) Connection ext. M8 limit switch or digital I/Os

Description	ID	
Connection caps		
MMI 070-V05-E-CN	0307500	
MMI 070-V05-D-CN	0307501	
MMI 070-V05-E-PB	0307502	
MMI 070-V05-D-PB	0307503	

Power cable for SCHUNK MMI



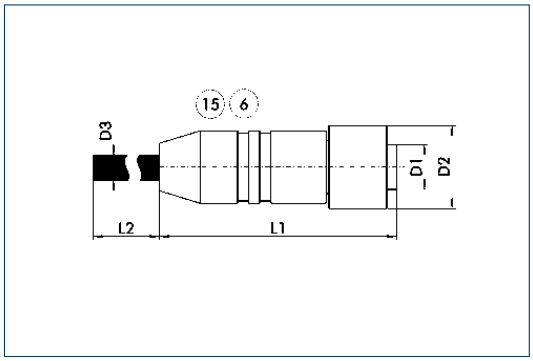
⑥ Connection module side ⑮ Socket

The power cable for the MMI connection cab is available in various lengths (L2). The power cable has an M23 connection plug on the module side. The cable can be optionally fitted with a matching mating plug (GG) or open wires (GL) on the other side.

Description	ID	L_2 [m]	D_1
Power cable for SCHUNK MMI			
KA GGN2304-LK-00150-H	0349874	1.5	M23
KA GGN2304-LK-00300-H	0349875	3	M23
KA GGN2304-LK-00500-H	0349876	5	M23
KA GGN2304-LK-01000-H	0349877	10	M23
KA GLN2304-LK-00150-H	0349870	1.5	M23
KA GLN2304-LK-00300-H	0349871	3	M23
KA GLN2304-LK-00500-H	0349872	0.5	M23
KA GLN2304-LK-01000-H	0349873	1	M23

ⓘ Please observe the bending radius (7.5 times the cable diameter).

CAN bus cable



⑥ Connection module side ⑮ Socket

The CAN bus cable is pre-assembled for our mechatronic modules with MMI connection cap and the RPH rotary module. It has an M12 connector plug on both sides.

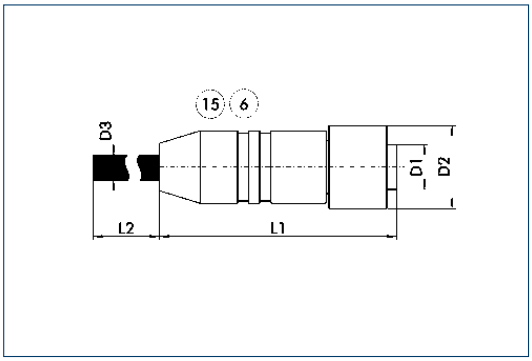
Description	ID	L_2 [m]	D_1
CAN bus cable			
KA GGN1204-CN-00150-A	0349770	1.5	M12
KA GGN1204-CN-00300-A	0349771	3	M12
KA GGN1204-CN-00500-A	0349772	5	M12
KA GGN1204-CN-01000-A	0349773	10	M12

ⓘ Please observe the bending radius (7.5 times the cable diameter).

PG 70

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

Profibus cable



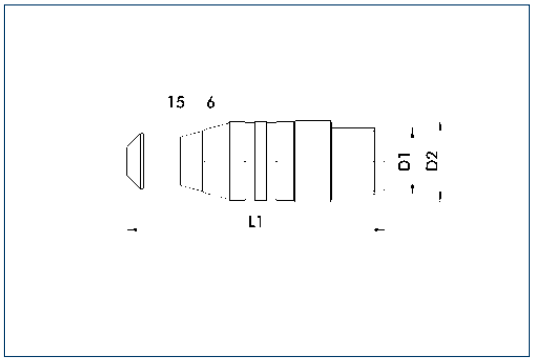
(6) Connection module side (15) Socket

The Profibus cable is pre-assembled for our mechatronic modules with MMI connection cap and PRH rotary module. It has an M12 connector plug on both sides.

Description	ID	L_2 [m]	D_1
Profibus cable			
KA GGN1204-PB-00150-A	0349750	1.5	M12
KA GGN1204-PB-00300-A	0349751	3	M12
KA GGN1204-PB-00500-A	0349752	5	M12
KA GGN1204-PB-01000-A	0349753	10	M12

ⓘ Please observe the bending radius (7.5 times the cable diameter).

Terminators



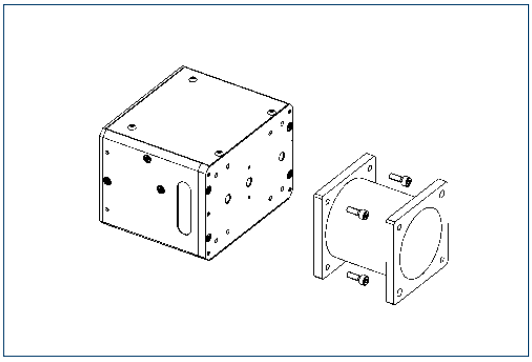
(6) Connection module side (15) Socket

The ST terminating resistors are provided for the termination of the bus string directly at the SCHUNK module. The terminating resistors are available for the Profibus (RB) or CAN bus (CN) bus systems.

Description	ID	D_1
Terminators		
ST SG1204-CN-A-A	0349660	M12
ST SG1204-PB-A-A	0349650	M12

ⓘ A suitable terminator must be mounted on the last module in the CAN or Profibus line.

Connecting element – straight

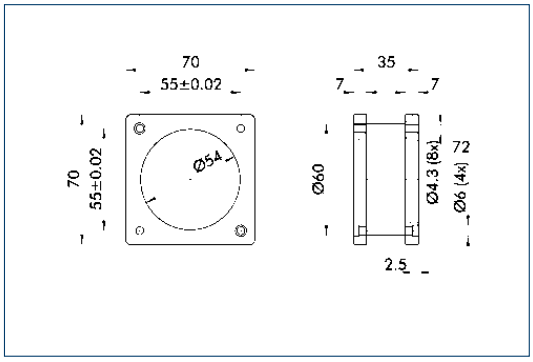


Straight standard element for connection of mechatronic modules PG, PR, PDU, and PSM in size 70.

Description	ID	Dimensions
Connecting element		
PAM 100	0307800	70x70/35/70x70 mm
PAM 101	0307801	70x70/70/70x70 mm

ⓘ Special lengths are available on request. Please contact us.

PAM 100 – straight



(72) Fit for centering sleeves

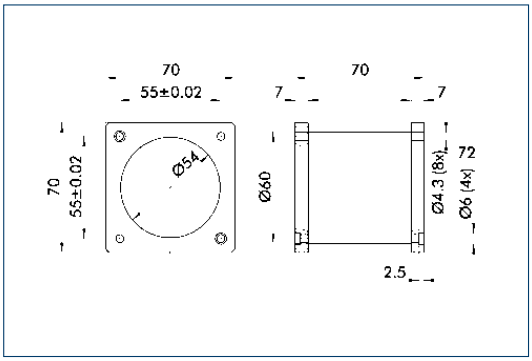
Suitable for mechatronic modules PG, PR, PDU, and PSM in size 70.

Description	ID	Dimensions
Connecting element		
PAM 100	0307800	70x70/35/70x70 mm

PG 70

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

PAM 101 – straight

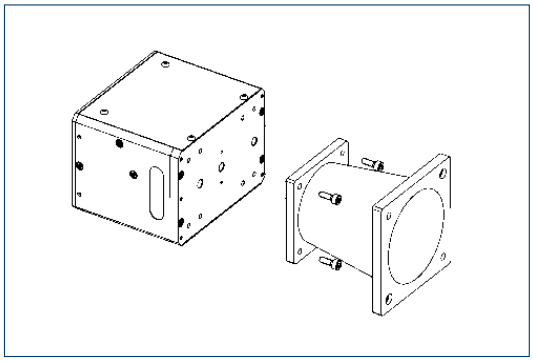


(72) Fit for centering sleeves

Suitable for mechatronic modules PG, PR, PDU, and PSM in size 70.

Description	ID	Dimensions
Connecting element		
PAM 101	0307801	70x70/70/70x70 mm

Connecting element – conical

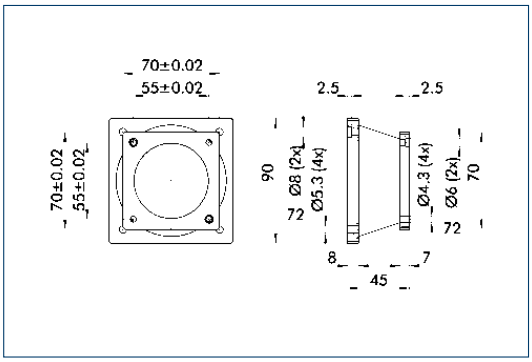


Straight standard element for connection of mechatronic modules PG, PR, PDU, and PSM in size 70/90.

Description	ID	
Connecting element		
PAM 110	0307810	
PAM 111	0307811	

ⓘ Special lengths are available on request. Please contact us.

PAM 110 – conical

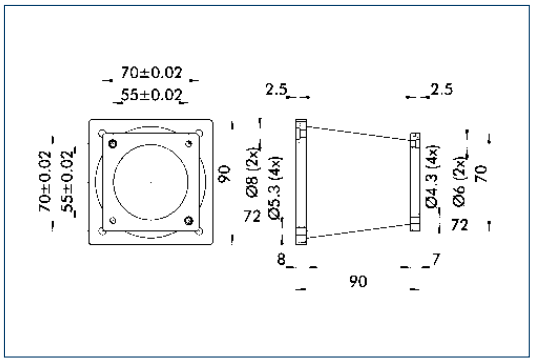


(72) Fit for centering sleeves

Suitable for mechatronic modules PG, PR, PDU, and PSM in size 70/90.

Description	ID	
Connecting element		
PAM 110	0307810	

PAM 111 – conical



(72) Fit for centering sleeves

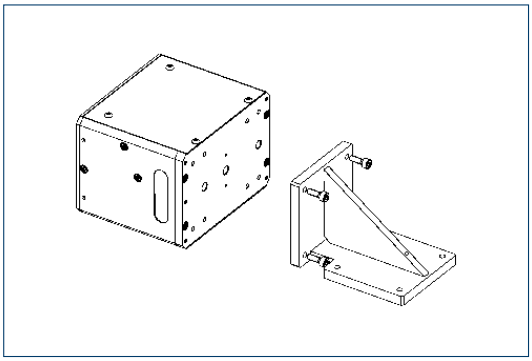
Suitable for mechatronic modules PG, PR, PDU, and PSM in size 70/90.

Description	ID	
Connecting element		
PAM 111	0307811	

PG 70

SCHUNK Grippers electric | 2-Finger Parallel Grippers | Universal Gripper

Connecting element – angle

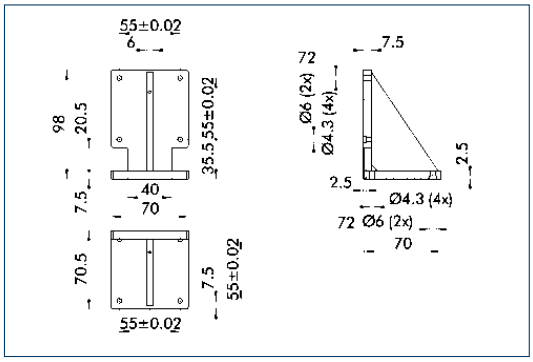


Angled standard element for connection of mechatronic modules PG, PR, PDU, and PSM in size 70.

Description	ID	
Connecting element		
PAM 120	0307820	

ⓘ Special lengths are available on request. Please contact us.

PAM 120 – angle



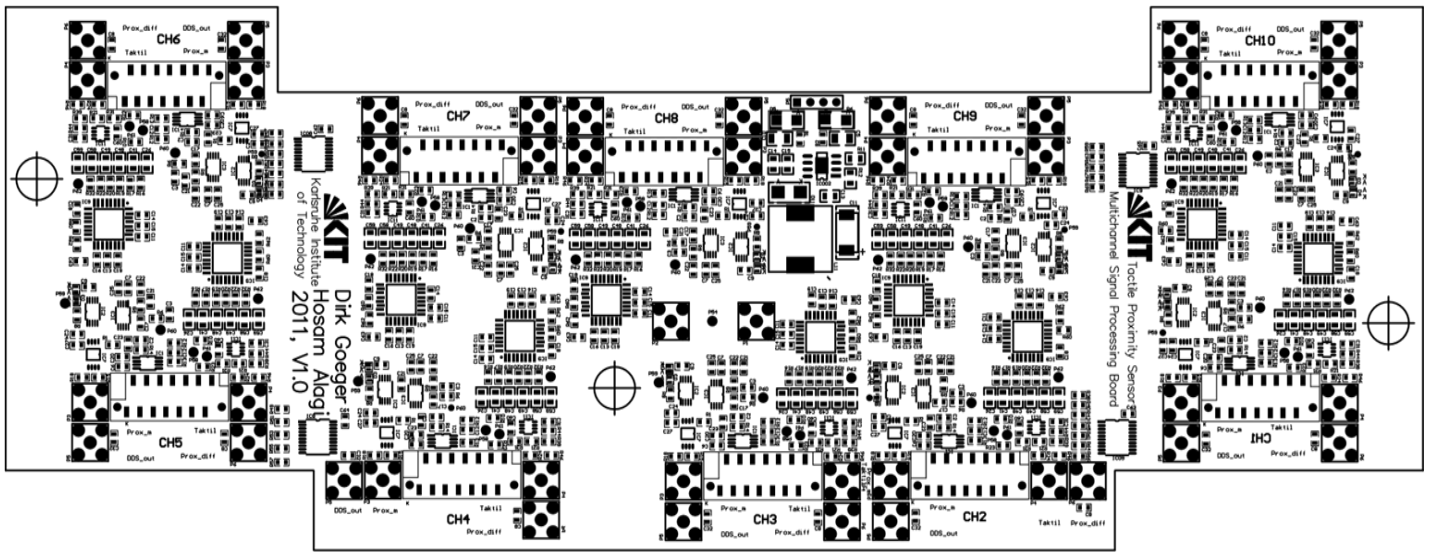
(72) Fit for centering sleeves

Suitable for mechatronic modules PG, PR, PDU, and PSM in size 70.

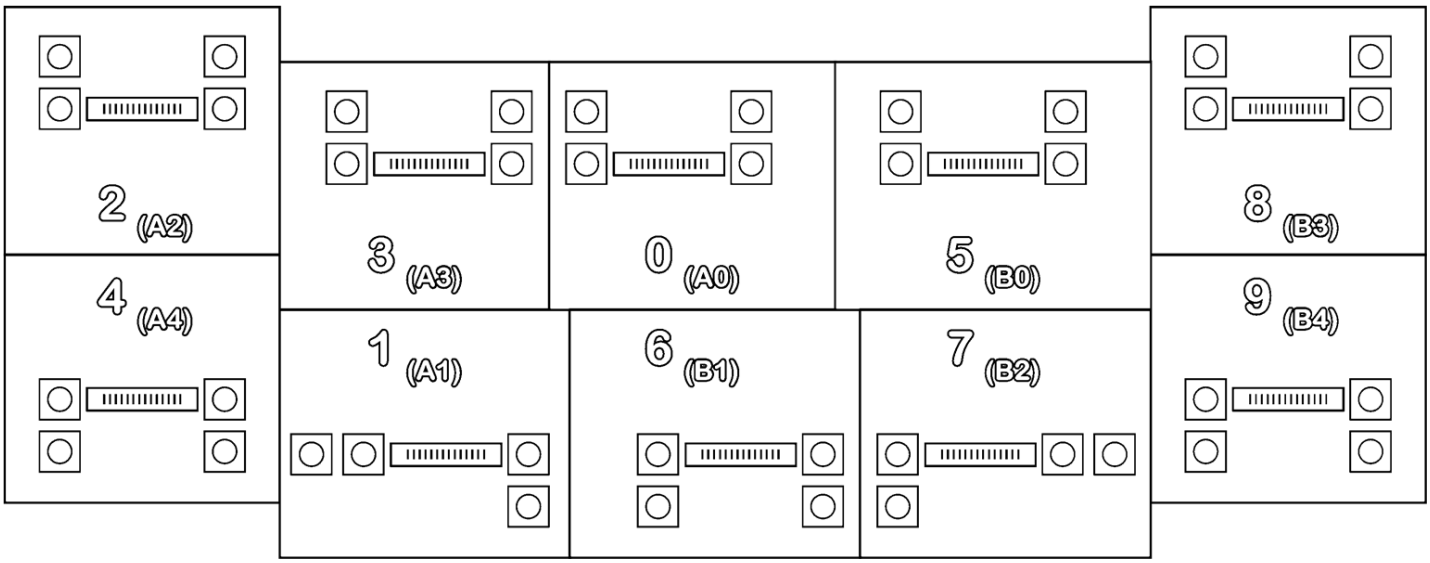
Description	ID	
Connecting element		
PAM 120	0307820	

Anexo D: Placa de expansión de 10 canales

10 – Channels FPGA
Top Overlay Footprint

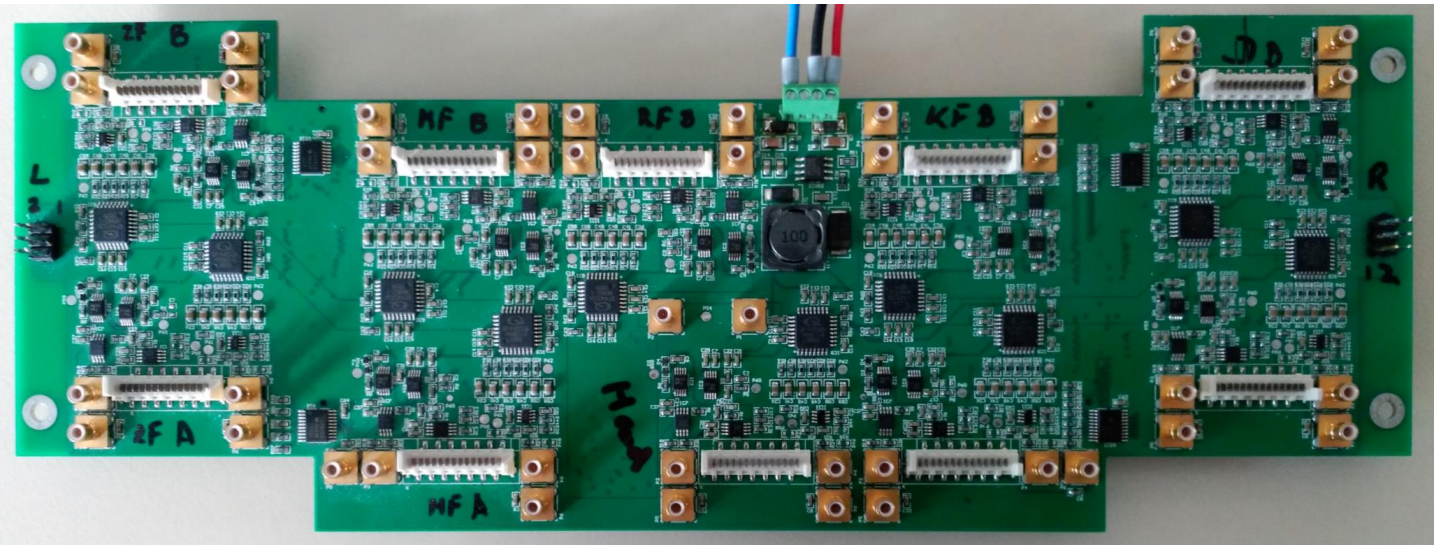


10 – Channels FPGA
Bottom Overlay Footprint

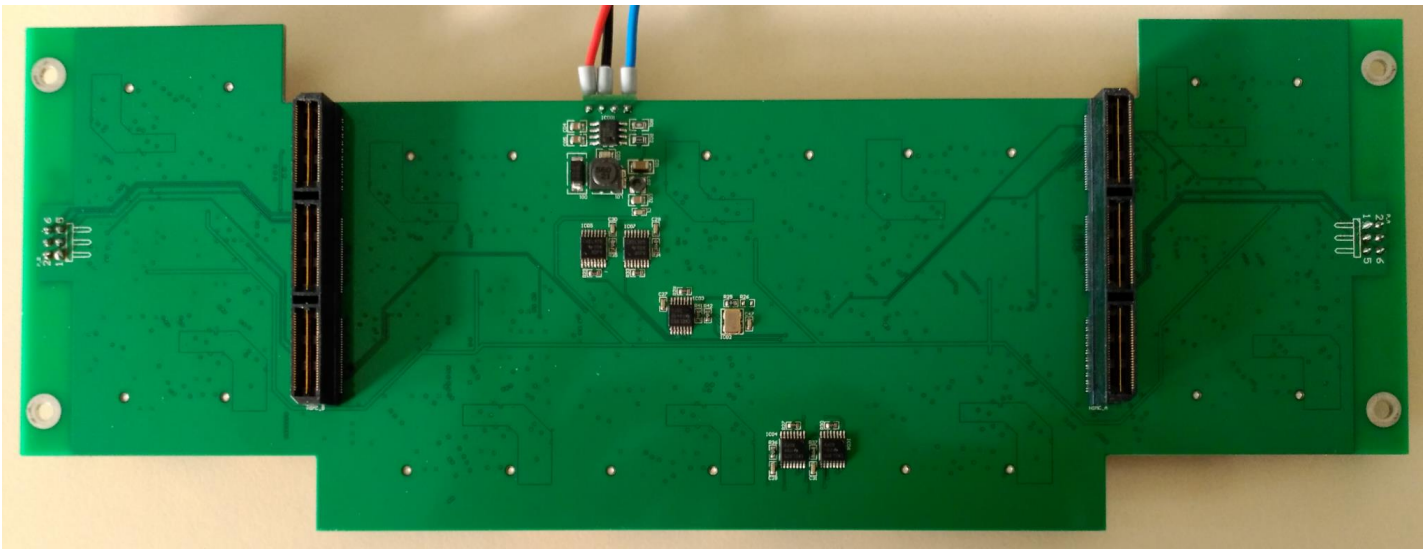


10 – Channels FPGA

Top Physical View



10 – Channels FPGA
Bottom Physical View



Anexo E:
Elastosil LR 3162

WACKER ELASTOSIL®

ELASTOSIL® LR 3162 A/B

LIQUID SILICONE RUBBER

Product description

Electrically conductive liquid silicone rubber. ELASTOSIL® LR 3162 A/B is a paste-like, two-component compound with short curing times.

Properties

The vulcanizates are noted for good mechanical as well as electrical properties and for good heat resistance within a temperature range of -55°C to+210°C.

Application

These grades are particularly suitable for the economical production of large series of injection-moulded electrically conductive articles. Parts made from ELASTOSIL® LR 3162 A/B can be used for technical applications, but do not comply with regulations concerning use in the pharmaceutical and food industry.

Processing

The A and B components are delivered ready to use in 20 and 200 litre drums. With adequate metering equipment, they can be pumped directly from the original containers into the injection molding machine and mixed by a static mixer. The A and B components are delivered ready to use in 20 litre drums. With adequate metering equipment, they can be pumped directly from the original containers into the injection molding machine and mixed by a static mixer. The mixing ratio is 1 : 1.
At room temperature, mixtures of A and B components have a pot life of at least 8 hours.

For detailed information refer to our brochure "SOLID AND LIQUID SILICONE RUBBER - MATERIAL AND PROCESSING GUIDELINES".

Storage

The 'Best use before end' date of each batch is shown on the product label.

Storage beyond the date specified on the label does not necessarily mean that the product is no longer usable. In this case however, the properties required for the intended use must be checked for quality assurance reasons.

Safety notes

Comprehensive instructions are given in the corresponding Material Safety Data Sheets. They are available on request from WACKER subsidiaries or may be printed via WACKER web site http://www.wacker.com.

Product data

Typical general characteristics	Inspection Method	Value
Hardness Shore A	DIN 53505	53
Appearance		black
Density at 23 °C	ISO 1183-1 A	approx. 1,12 g/cm³
Viscosity (shear rate 0.9 s^{-1})	DIN 53019	6600000 mPa s
Tensile strength	DIN 53504 S 1	5,50 N/mm^2
Elongation at break	DIN 53504 S 1	400 %
Tear strength	ASTM D 624 B	18 N/mm
Rebound resilience	DIN 53512	54 %
Volume resistivity	DIN VDE 0303	11 Ω cm

Cure conditions: 10 min / 165 °C in press
These figures are only intended as a guide and should not be used in preparing specifications.

Anexo F:
RFC 768 de la IETF. Estándar UDP

RFC 768 J. Postel
ISI
28 August 1980

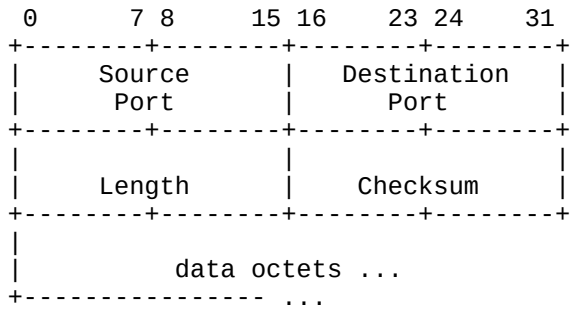
User Datagram Protocol

Introduction

This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format



User Datagram Header Format

Fields

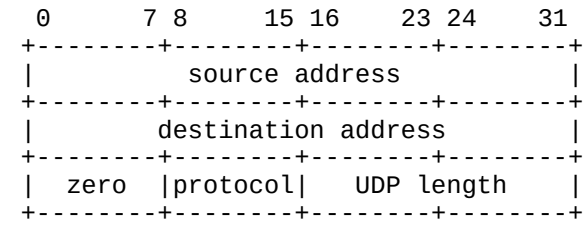
Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted.

Destination Port has a meaning within the context of a particular internet destination address.

Length is the length in octets of this user datagram including this header and the data. (This means the minimum value of the length is eight.)

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

The pseudo header conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length. This information gives protection against misrouted datagrams. This checksum procedure is the same as is used in TCP.



If the computed checksum is zero, it is transmitted as all ones (the equivalent in one's complement arithmetic). An all zero transmitted checksum value means that the transmitter generated no checksum (for debugging or for higher level protocols that don't care).

User Interface

A user interface should allow

the creation of new receive ports,

receive operations on the receive ports that return the data octets and an indication of source port and source address,

and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent.

IP Interface

The UDP module must be able to determine the source and destination internet addresses and the protocol field from the internet header. One possible UDP/IP interface would return the whole internet datagram including all of the internet header in response to a receive operation. Such an interface would also allow the UDP to pass a full internet datagram complete with header to the IP to send. The IP would verify certain fields for consistency and compute the internet header checksum.

Protocol Application

The major uses of this protocol is the Internet Name Server [3], and the Trivial File Transfer [4].

Protocol Number

This is protocol 17 (21 octal) when used in the Internet Protocol. Other protocol numbers are listed in [5].

References
---

[1] Postel, J., "Internet Protocol," RFC 760, USC/Information Sciences Institute, January 1980.

[2] Postel, J., "Transmission Control Protocol," RFC 761, USC/Information Sciences Institute, January 1980.

[3] Postel, J., "Internet Name Server," USC/Information Sciences Institute, IEN 116, August 1979.

[4] Sollins, K., "The TFTP Protocol," Massachusetts Institute of Technology, IEN 133, January 1980.

[5] Postel, J., "Assigned Numbers," USC/Information Sciences Institute, RFC 762, January 1980.

Anexo G: Código Matlab para Simulación

El archivo “start.m” que podemos ver bajo estas líneas, tiene como función limpiar por completo el entorno de Matlab, llamar a la función “generate_signal” para crear una señal de excitación con y sin ruido y calcular su señal equivalente de corriente medida, con y sin ruido también, de la correspondiente medida de proximidad asociada a la señal de excitación creada. Este archivo también define ciertos parámetros para los correspondientes filtros por los que posteriormente pasarán las señales medidas, tales como tipo de filtro, orden de éste, y sus frecuencias de corte. Una vez definidos estos parámetros, se realizan una serie de tests.

```
1  %Signal generation
2  [current, current_noise, current_exc, current_exc_noise, samples] =
       generate_signal;
3
4  %Parameters
5  filt_type  = 'cheby1';
6  order = 6;
7
8  %Anti Aliasign Filter
9  wc2   = 0.25;
10
11 %DFT Filter.
12 %wc4   = 0.51;
13 wc4   = [0.49 0.51];
14
15 %Final Filter
16 wc6   = 0.002;
17
18 %%modificar la siguiente línea para ejecutar el test correspondiente
19 %test_name (signal, signal_noise, samples, window)
20 amplitude_test_1 (current, current_noise, samples, 2);
21 amplitude_test_1 (current_exc, current_exc_noise, samples, 2);
```

El archivo “generate_signal” que se muestra a continuación, es una función que

crea y muestra en diferentes gráficas cada una de las señales explicadas en el capítulo dedicado al modelo Matlab del sistema y genera a su salida una señal de excitación con y sin ruido y su señal correspondiente de corriente medida, con y sin ruido.

```
% Samples should be only two powers to simplify calculations.
function [current, current_noise, current_exc, current_exc_noise, samples]
    = generate_signal

%% %%%%%%%%%%%%%%%%%%%%
%% DISTANCE PATTERN  %%
%%%%%%%%%%%%%%%%%%%%%%%%
%  Datos para generar el patrón de movimiento.
%  El músculo más rápido del cuerpo humano es el de los párpados,
%  capaz de abrirse y cerrarse hasta 5 veces por segundo. Por tanto,
%  para nuestra aplicación ésta es la máxima frecuencia posible.
%  fmax = 5 Hz
freq_mov_hertz = 0.005;
%  Generación del patrón de movimiento
lot1           = 10:2000;               % v=cte   , a=0
lot2           = 2000*ones(1,1000);     % v=0     , a=0
lot3           = 1998+2.^(1:0.002:11);  % v=x     , a=cte
lot4           = 4049:-3:1500;          % v=3*cte , a=0

freq_mov_rad   = 2*pi*freq_mov_hertz;
axis_sine      = (0:pi/4:pi/4*2147);
sine           = 1500 + 400*sin(freq_mov_rad.*axis_sine);
lot5           = sine;                  % v=x     , a=sinusoide
% divido cada muestra por 100 para que las medidas cuadren con mm
distance       = [lot1 lot2 lot3 lot4 lot5]./10^2;

%% %%%%%%%%%%%%%%%%%%%%
%% EXCITATION SIGNAL  %%  (((EXCITATION SIGNAL TO ADC)))
%%%%%%%%%%%%%%%%%%%%%%%%
%  Data:
amplitude   = 12;                   % +/-12V --NO ES NECESARIO--
hertzios    = 98625;                % 98,625 KHz
radianes    = 2*pi*hertzios;
Rm          = 3.6e3 + 11e3 + 360;

Eo          = 8.8541878176e-12;  % Permitividad eléctrica en el vacío
Er          = 1;                 % Permitividad eléctrica del aire
area        = (4e-2)^2;          % 4cm^2

phase       = pi/4;
```

```
ad_relation = pi/2;
samples= length (distance);

%% CAUTION %%
%  Nitidez have to be 1 to the signal has 4 samples per periode and we can do
%  calculates. Others values like 100 or 10 can be possible only to see the
%  EXCITATION SIGNAL MORE CLREARLY.
nitidez = 100;
axis_n       = (0:ad_relation/nitidez:ad_relation*(samples-1)/nitidez)./ radianes;
%  Voltage
voltage      = amplitude * sin((radianes.*axis_n) + phase);
%  White Noise
white_noise  = random ('norm', 0, 1, [1,length(axis_n)]);
%  Excitation signal to ADC
excitation   = amplitude * sin((radianes.*axis_n) + phase) + white_noise;

%%%%%%%%%%%%%%%%%%%%
%% REPRESENTATION  %%
%%%%%%%%%%%%%%%%%%%%
% Distance
figure;
plot (1:length(distance), distance);
axis([0,11100,0,50]);
xlabel('Time (ms)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Distance (mm)','FontSize',12,'FontWeight','bold','Color','b');
title('Distance','FontSize',12,'FontWeight','bold','Color','k');

% Excitation Signal
figure;
plot (axis_n, voltage);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Voltage (V)','FontSize',12,'FontWeight','bold','Color','b');
title('Excitation Signal','FontSize',12,'FontWeight','bold','Color','k');

%  White Noise
figure;
plot (axis_n, white_noise);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Voltage (V)','FontSize',12,'FontWeight','bold','Color','b');
title('White Noise','FontSize',12,'FontWeight','bold','Color','k');

%  Excitation Signal With Noise
```

```
figure;
plot (axis_n, excitation);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Voltage (V)','FontSize',12,'FontWeight','bold','Color','b');
title('Excitation Signal With Noise','FontSize',12,'FontWeight','bold','
   Color','k');


%% CAUTION
%  Nitidez have to be 1 to the signal has 4 samples per periode and we can
    do
%  calculates. Others values like 100 or 10 can be possible only to see
   the
%  EXCITATION SIGNAL MORE CLREARLY.
nitidez = 1;
axis_n       = (0:ad_relation/nitidez:ad_relation*(samples-1)/nitidez)./
   radianes;
%  Voltage
voltage      = amplitude * sin((radianes.*axis_n) + phase);
% White Noise
white_noise  = random ('norm', 0, 1, [1,length(axis_n)]);
%  Excitation signal to ADC
excitation   = amplitude * sin((radianes.*axis_n) + phase) + white_noise;

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   WITHOUT EXCITATION SIGNAL    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Capacitance without excitation signal
capacitance  = (Er*Eo*area)./distance;

%  Resistance without excitation signal[(condensador (Xc) en serie con una
    resistencia (Rm)]
resistance   = Rm + 1./(radianes.*capacitance);

%  Current without excitation signal
current       = voltage./resistance;
current_noise = current + white_noise;

% REPRESENTATIONS %
%  Capacitance without excitation signal
figure;
plot (axis_n, capacitance);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Capacitance (F)','FontSize',12,'FontWeight','bold','Color','b');
title('Capacitance Without Excitation Signal','FontSize',12,'FontWeight','
```

```
   bold','Color','k');

% Resistance without excitation signal
figure;
plot (axis_n, resistance);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Resistance (\Omega)','FontSize',12,'FontWeight','bold','Color','b'
   );
title('Resistance Without Excitation Signal','FontSize',12,'FontWeight','
   bold','Color','k');

% Current without excitation signal
figure;
plot (axis_n, current);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Intensity (A)','FontSize',12,'FontWeight','bold','Color','b');
title('Current Without Excitation Signal','FontSize',12,'FontWeight','bold
   ','Color','k');

% Current with noise without excitation signal
figure;
plot (axis_n, current_noise);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Intensity (A)','FontSize',12,'FontWeight','bold','Color','b');
title('Current With Noise Without Excitation Signal','FontSize',12,'
   FontWeight','bold','Color','k');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% WITH EXCITATION SIGNAL  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Capacitance with excitation signal
capacitance_exc  = excitation.*(Er*Eo*area)./distance;

% Resistance with excitation signal[(condensador (Xc) en serie con una
   resistencia (Rm)]
resistance_exc   = Rm + 1./(radianes.*capacitance_exc);

% Current with excitation signal
current_exc       = voltage./resistance_exc;
current_exc_noise = current_exc + white_noise;

% REPRESENTATIONS %
% Capacitance with excitation signal
figure;
plot (axis_n, capacitance_exc);
```

```
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Capacitance (F)','FontSize',12,'FontWeight','bold','Color','b');
title('Capacitance With Excitation Signal','FontSize',12,'FontWeight','
    bold','Color','k');

% Resistance with excitation signal
figure;
plot (axis_n, resistance_exc);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Resistance (\Omega)','FontSize',12,'FontWeight','bold','Color','b'
    );
title('Resistance With Excitation Signal','FontSize',12,'FontWeight','bold
    ','Color','k');

% Current with excitation signal
figure;
plot (axis_n, current_exc);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Intensity (A)','FontSize',12,'FontWeight','bold','Color','b');
title('Current With Excitation Signal','FontSize',12,'FontWeight','bold','
    Color','k');

% Current with noise with excitation signal
figure;
plot (axis_n, current_exc_noise);
xlabel('Time (s)','FontSize',12,'FontWeight','bold','Color','b');
ylabel('Intensity (A)','FontSize',12,'FontWeight','bold','Color','b');
title('Current With Noise With Excitation Signal','FontSize',12,'
    FontWeight','bold','Color','k');
```

Los tres archivos de test siguientes, *amplitude_test_1.m, amplitude_test_2.m y amplitude_test_3.m* son diferentes implementaciones más o menos optimizadas de la *DFT* para ventanas de N=2, 4, 8 y 16.

```
%{A LA SALIDA TENEMOS 1 MUESTRA CADA N MUESTRAS A LA ENTRADA:%}
function [] = amplitude_test_1 (signal, signal_noise, samples, window)
%%N=2;
if window == 2
axis_n_2        = 1:samples/window;
out_add_2       = zeros (1,0);
out_dif_2       = zeros (1,0);
out_add_noise_2 = zeros (1,0);
out_dif_noise_2 = zeros (1,0);

for m = 1:window:samples-window+1 %should be while true;
```

```
    var0 = signal(m);
    var1 = signal(m+1);
    ampl_add_2 = var0^2+var1^2;
    ampl_dif_2 = var0^2-var1^2;
    out_add_2  = [out_add_2 ampl_add_2];
    out_dif_2  = [out_dif_2 ampl_dif_2];

    var0_noise = signal_noise(m);
    var1_noise = signal_noise(m+1);
    ampl_add_noise_2 = var0_noise^2+var1_noise^2;
    ampl_dif_noise_2 = var0_noise^2-var1_noise^2;
    out_add_noise_2  = [out_add_noise_2 ampl_add_noise_2];
    out_dif_noise_2  = [out_dif_noise_2 ampl_dif_noise_2];
end

figure;
stem (axis_n_2, out_add_2);
title('N=2. Add Without Noise');
figure;
stem (axis_n_2, out_dif_2);
title('N=2. Dif Without Noise');
figure;
stem (axis_n_2, out_add_noise_2);
title('N=2. Add With Noise');
figure;
stem (axis_n_2, out_dif_noise_2);
title('N=2. Dif With Noise');

%%N=4;
elseif window == 4
axis_n_4        = 1:samples/window;
out_add_4       = zeros (1,0);
out_dif_4       = zeros (1,0);
out_add_noise_4 = zeros (1,0);
out_dif_noise_4 = zeros (1,0);

for m = 1:window:samples-window %should be while true;
    var0 = signal(m);
    var1 = signal(m+1);
    var2 = signal(m+2);
    var3 = signal(m+3);
    ampl_add_4 = (var0+var2)^2+(var3+var1)^2;
    ampl_dif_4 = (var0-var2)^2+(var3-var1)^2;
    out_add_4  = [out_add_4 ampl_add_4];
    out_dif_4  = [out_dif_4 ampl_dif_4];
```

```
57
58      var0_noise = signal_noise(m);
59      var1_noise = signal_noise(m+1);
60      var2_noise = signal_noise(m+2);
61      var3_noise = signal_noise(m+3);
62      ampl_add_noise_4 = (var0_noise+var2_noise)^2+(var3_noise+var1_noise)
            ^2;
63      ampl_dif_noise_4 = (var0_noise-var2_noise)^2+(var3_noise-var1_noise)
            ^2;
64      out_add_noise_4  = [out_add_noise_4 ampl_add_noise_4];
65      out_dif_noise_4  = [out_dif_noise_4 ampl_dif_noise_4];
66  end
67
68  figure;
69  stem (axis_n_4, out_add_4);
70  title('N=4. Add Without Noise');
71  figure;
72  stem (axis_n_4, out_dif_4);
73  title('N=4. Dif Without Noise');
74  figure;
75  stem (axis_n_4, out_add_noise_4);
76  title('N=4. Add With Noise');
77  figure;
78  stem (axis_n_4, out_dif_noise_4);
79  title('N=4. Dif With Noise');
80
81  %%N=8;
82  elseif window == 8
83  axis_n_8        = 1:samples/window;
84  out_add_8       = zeros (1,0);
85  out_dif_8       = zeros (1,0);
86  out_add_noise_8 = zeros (1,0);
87  out_dif_noise_8 = zeros (1,0);
88
89  for m = 1:window:samples-window+1 %should be while true;
90      var0 = signal(m);
91      var1 = signal(m+1);
92      var2 = signal(m+2);
93      var3 = signal(m+3);
94      var4 = signal(m+4);
95      var5 = signal(m+5);
96      var6 = signal(m+6);
97      var7 = signal(m+7);
98      ampl_add_8 = (var0+var4+var2+var6)^2+(var3+var7+var1+var5)^2;
99      ampl_dif_8 = (var0+var4-var2-var6)^2+(var3+var7-var1-var5)^2;
```

```
    out_add_8  = [out_add_8 ampl_add_8];
    out_dif_8  = [out_dif_8 ampl_dif_8];

    var0_noise = signal_noise(m);
    var1_noise = signal_noise(m+1);
    var2_noise = signal_noise(m+2);
    var3_noise = signal_noise(m+3);
    var4_noise = signal_noise(m+4);
    var5_noise = signal_noise(m+5);
    var6_noise = signal_noise(m+6);
    var7_noise = signal_noise(m+7);
    ampl_add_noise_8 = (var0_noise+var4_noise+var2_noise+var6_noise)^2+(
       var3_noise+var7_noise+var1_noise+var5_noise)^2;
    ampl_dif_noise_8 = (var0_noise+var4_noise-var2_noise-var6_noise)^2+(
       var3_noise+var7_noise-var1_noise-var5_noise)^2;
    out_add_noise_8  = [out_add_noise_8 ampl_add_noise_8];
    out_dif_noise_8  = [out_dif_noise_8 ampl_dif_noise_8];
end

figure;
stem (axis_n_8, out_add_8);
title('N=8. Add Without Noise');
figure;
stem (axis_n_8, out_dif_8);
title('N=8. Dif Without Noise');
figure;
stem (axis_n_8, out_add_noise_8);
title('N=8. Add With Noise');
figure;
stem (axis_n_8, out_dif_noise_8);
title('N=8. Dif With Noise');

%%N=16;
elseif window == 16
axis_n_16        = 1:samples/window;
out_add_16       = zeros (1,0);
out_dif_16       = zeros (1,0);
out_add_noise_16 = zeros (1,0);
out_dif_noise_16 = zeros (1,0);

for m = 1:window:samples-window+1 %should be while true;
    var0  = signal(m);
    var1  = signal(m+1);
    var2  = signal(m+2);
    var3  = signal(m+3);
```

```
143     var4  = signal(m+4);
144     var5  = signal(m+5);
145     var6  = signal(m+6);
146     var7  = signal(m+7);
147     var8  = signal(m+8);
148     var9  = signal(m+9);
149     var10 = signal(m+10);
150     var11 = signal(m+11);
151     var12 = signal(m+12);
152     var13 = signal(m+13);
153     var14 = signal(m+14);
154     var15 = signal(m+15);
155     ampl_add_16 = (var0+var4+var8+var12+var2+var6+var10+var14)^2+(var3+
            var7+var11+var15+var1+var5+var9+var13)^2;
156     ampl_dif_16 = (var0+var4+var8+var12-var2-var6-var10-var14)^2+(var3+
            var7+var11+var15-var1-var5-var9-var13)^2;
157     out_add_16  = [out_add_16 ampl_add_16];
158     out_dif_16  = [out_dif_16 ampl_dif_16];
159
160     var0_noise  = signal_noise(m);
161     var1_noise  = signal_noise(m+1);
162     var2_noise  = signal_noise(m+2);
163     var3_noise  = signal_noise(m+3);
164     var4_noise  = signal_noise(m+4);
165     var5_noise  = signal_noise(m+5);
166     var6_noise  = signal_noise(m+6);
167     var7_noise  = signal_noise(m+7);
168     var8_noise  = signal_noise(m+8);
169     var9_noise  = signal_noise(m+9);
170     var10_noise = signal_noise(m+10);
171     var11_noise = signal_noise(m+11);
172     var12_noise = signal_noise(m+12);
173     var13_noise = signal_noise(m+13);
174     var14_noise = signal_noise(m+14);
175     var15_noise = signal_noise(m+15);
176     ampl_add_noise_16 = (var0_noise+var4_noise+var8_noise+var12_noise+
            var2_noise+var6_noise+var10_noise+var14_noise)^2+(var3_noise+
            var7_noise+var11_noise+var15_noise+var1_noise+var5_noise+var9_noise
            +var13_noise)^2;
177     ampl_dif_noise_16 = (var0_noise+var4_noise+var8_noise+var12_noise-
            var2_noise-var6_noise-var10_noise-var14_noise)^2+(var3_noise+
            var7_noise+var11_noise+var15_noise-var1_noise-var5_noise-var9_noise
            -var13_noise)^2;
178     out_add_noise_16  = [out_add_noise_16 ampl_add_noise_16];
179     out_dif_noise_16  = [out_dif_noise_16 ampl_dif_noise_16];
```

```
end

figure;
stem (axis_n_16, out_add_16);
title('N=16. Add Without Noise');
figure;
stem (axis_n_16, out_dif_16);
title('N=16. Dif Without Noise');
figure;
stem (axis_n_16, out_add_noise_16);
title('N=16. Add With Noise');
figure;
stem (axis_n_16, out_dif_noise_16);
title('N=16. Dif With Noise');
end
```

```
%{A LA SALIDA TENEMOS 1 MUESTRA CADA N MUESTRAS A LA ENTRADA: %}
function [] = amplitude_test_3 (signal, signal_noise, window)
%%N have to be power of 2 and between 2 and 16;
samples       = length (signal);
axis_n        = 1:samples/window;
out_add       = zeros (1,0);
out_dif       = zeros (1,0);
out_add_noise = zeros (1,0);
out_dif_noise = zeros (1,0);

for m = 1:window:samples %should be while true;
    for p = 1:window
        var(p)       = signal (m+p-1);
        var_noise(p) = signal_noise (m+p-1);
    end
    if window == 2
        ampl_add = var(1)^2+var(2)^2;
        ampl_dif = var(1)^2-var(2)^2;
        ampl_add_noise = var_noise(1)^2+var_noise(2)^2;
        ampl_dif_noise = var_noise(1)^2-var_noise(2)^2;
    elseif window == 4
        ampl_add = (var(1)+var(3))^2+(var(4)+var(2))^2;
        ampl_dif = (var(1)-var(3))^2+(var(4)-var(2))^2;
        ampl_add_noise = (var_noise(1)+var_noise(3))^2+(var_noise(4)+
           var_noise(2))^2;
        ampl_dif_noise = (var_noise(1)-var_noise(3))^2+(var_noise(4)-
           var_noise(2))^2;
    elseif window == 8
        ampl_add = (var(1)+var(5)+var(3)+var(7))^2+(var(4)+var(8)+var(2)+
           var(6))^2;
        ampl_dif = (var(1)+var(5)-var(3)-var(7))^2+(var(4)+var(8)-var(2)-
           var(6))^2;
        ampl_add_noise = (var_noise(1)+var_noise(5)+var_noise(3)+var_noise
           (7))^2+(var_noise(4)+var_noise(8)+var_noise(2)+var_noise(6))^2;
        ampl_dif_noise = (var_noise(1)+var_noise(5)-var_noise(3)-var_noise
           (7))^2+(var_noise(4)+var_noise(8)-var_noise(2)-var_noise(6))^2;
    elseif window == 16
        ampl_add = (var(1)+var(5)+var(9)+var(13)+var(3)+var(7)+var(11)+var
           (15))^2+(var(4)+var(8)+var(12)+var(16)+var(2)+var(6)+var(10)+
           var(14))^2;
        ampl_dif = (var(1)+var(5)+var(9)+var(13)-var(3)-var(7)-var(11)-var
           (15))^2+(var(4)+var(8)+var(12)+var(16)-var(2)-var(6)-var(10)-
           var(14))^2;
        ampl_add_noise = (var_noise(1)+var_noise(5)+var_noise(9)+var_noise
```

```
            (13)+var_noise(3)+var_noise(7)+var_noise(11)+var_noise(15))^2+(
            var_noise(4)+var_noise(8)+var_noise(12)+var_noise(16)+var_noise
            (2)+var_noise(6)+var_noise(10)+var_noise(14))^2;
        ampl_dif_noise = (var_noise(1)+var_noise(5)+var_noise(9)+var_noise
            (13)-var_noise(3)-var_noise(7)-var_noise(11)-var_noise(15))^2+(
            var_noise(4)+var_noise(8)+var_noise(12)+var_noise(16)-var_noise
            (2)-var_noise(6)-var_noise(10)-var_noise(14))^2;
    end
        out_add = [out_add ampl_add];
        out_dif = [out_dif ampl_dif];
        out_add_noise = [out_add_noise ampl_add_noise];
        out_dif_noise = [out_dif_noise ampl_dif_noise];
end

figure;
stem (axis_n, out_add);
title('Add Without Noise');

figure;
stem (axis_n, out_dif);
title('Dif Without Noise');

figure;
stem (axis_n, out_add_noise);
title('Add With Noise');

figure;
stem(axis_n, out_dif_noise);
title('Dif With Noise');

figure;
plot (axis_n, out_add);
title('Add Without Noise');

figure;
plot (axis_n, out_dif);
title('Dif Without Noise');

figure;
plot (axis_n, out_add_noise);
title('Add With Noise');

figure;
plot(axis_n, out_dif_noise);
title('Dif With Noise');
```

```
%{A LA SALIDA TENEMOS 1 MUESTRA CADA N MUESTRAS A LA ENTRADA:%}
function [] = amplitude_test_4 (signal, signal_noise, window)
%%N have to be power of 2 and bigger or same than 4;
samples       = length (signal);
axis_n        = 1:samples/window;
out_add       = zeros (1,0);
out_dif       = zeros (1,0);
out_add_noise = zeros (1,0);
out_dif_noise = zeros (1,0);

for m = 1:window:samples %should be while true;
    sum1       = 0;
    sum2       = 0;
    sum3       = 0;
    sum4       = 0;
    sum_noise1 = 0;
    sum_noise2 = 0;
    sum_noise3 = 0;
    sum_noise4 = 0;

    for p = 1:window
        var(p)       = signal (m+p-1);
        var_noise(p) = signal_noise (m+p-1);
    end

    for k = 1:4:window
        sum1       = sum1 + var(k);
        sum2       = sum2 + var(k+2);
        sum3       = sum3 + var(k+3);
        sum4       = sum4 + var(k+1);
        sum_noise1 = sum_noise1 + var_noise(k);
        sum_noise2 = sum_noise2 + var_noise(k+2);
        sum_noise3 = sum_noise3 + var_noise(k+3);
        sum_noise4 = sum_noise4 + var_noise(k+1);
    end

    ampl_add       = (sum1 + sum2) ^2 + (sum3 + sum4) ^2;
    ampl_dif       = (sum1 - sum2) ^2 + (sum3 - sum4) ^2;
    ampl_add_noise = (sum_noise1 + sum_noise2) ^2 + (sum_noise3 +
       sum_noise4) ^2;
    ampl_dif_noise = (sum_noise1 - sum_noise2) ^2 + (sum_noise3 -
       sum_noise4) ^2;

    out_add       = [out_add ampl_add];
```

```
    out_dif       = [out_dif ampl_dif];
    out_add_noise = [out_add_noise ampl_add_noise];
    out_dif_noise = [out_dif_noise ampl_dif_noise];
end

figure;
stem (axis_n, out_add);
title('Add Without Noise');

figure;
stem (axis_n, out_dif);
title('Dif Without Noise');

figure;
stem (axis_n, out_add_noise);
title('Add With Noise');

figure;
stem(axis_n, out_dif_noise);
title('Dif With Noise');

figure;
plot (axis_n, out_add);
title('Add Without Noise');

figure;
plot (axis_n, out_dif);
title('Dif Without Noise');

figure;
plot (axis_n, out_add_noise);
title('Add With Noise');

figure;
plot(axis_n, out_dif_noise);
title('Dif With Noise');
```

Los dos archivos de test siguientes, *amplitude_test_3_new.m y amplitude_test_4_new.m* son una mejora de las implementaciones de los test 3 y 4.

```
%{A LA SALIDA TENEMOS [SAMPLES-(N-1) = SAMPLES-N+1] MUESTRAS%}
function [] = amplitude_test_3_new (signal, signal_noise, window)
%%N have to be power of 2 and between 2 and 16;
samples       = length (signal);
axis_n        = 1:samples-window+1;
out_add       = zeros (1,0);
out_dif       = zeros (1,0);
out_add_noise = zeros (1,0);
out_dif_noise = zeros (1,0);

for m = 1:samples-window+1 %should be while true;
    for p = 1:window
        var(p)       = signal (m+p-1);
        var_noise(p) = signal_noise (m+p-1);
    end
    if window == 2
        ampl_add = var(1)^2+var(2)^2;
        ampl_dif = var(1)^2-var(2)^2;
        ampl_add_noise = var_noise(1)^2+var_noise(2)^2;
        ampl_dif_noise = var_noise(1)^2-var_noise(2)^2;
    elseif window == 4
        ampl_add = (var(1)+var(3))^2+(var(4)+var(2))^2;
        ampl_dif = (var(1)-var(3))^2+(var(4)-var(2))^2;
        ampl_add_noise = (var_noise(1)+var_noise(3))^2+(var_noise(4)+
           var_noise(2))^2;
        ampl_dif_noise = (var_noise(1)-var_noise(3))^2+(var_noise(4)-
           var_noise(2))^2;
    elseif window == 8
        ampl_add = (var(1)+var(5)+var(3)+var(7))^2+(var(4)+var(8)+var(2)+
           var(6))^2;
        ampl_dif = (var(1)+var(5)-var(3)-var(7))^2+(var(4)+var(8)-var(2)-
           var(6))^2;
        ampl_add_noise = (var_noise(1)+var_noise(5)+var_noise(3)+var_noise
           (7))^2+(var_noise(4)+var_noise(8)+var_noise(2)+var_noise(6))^2;
        ampl_dif_noise = (var_noise(1)+var_noise(5)-var_noise(3)-var_noise
           (7))^2+(var_noise(4)+var_noise(8)-var_noise(2)-var_noise(6))^2;
    elseif window == 16
        ampl_add = (var(1)+var(5)+var(9)+var(13)+var(3)+var(7)+var(11)+var
           (15))^2+(var(4)+var(8)+var(12)+var(16)+var(2)+var(6)+var(10)+
           var(14))^2;
        ampl_dif = (var(1)+var(5)+var(9)+var(13)-var(3)-var(7)-var(11)-var
```

```
            (15))^2+(var(4)+var(8)+var(12)+var(16)-var(2)-var(6)-var(10)-
            var(14))^2;
        ampl_add_noise = (var_noise(1)+var_noise(5)+var_noise(9)+var_noise
            (13)+var_noise(3)+var_noise(7)+var_noise(11)+var_noise(15))^2+(
            var_noise(4)+var_noise(8)+var_noise(12)+var_noise(16)+var_noise
            (2)+var_noise(6)+var_noise(10)+var_noise(14))^2;
        ampl_dif_noise = (var_noise(1)+var_noise(5)+var_noise(9)+var_noise
            (13)-var_noise(3)-var_noise(7)-var_noise(11)-var_noise(15))^2+(
            var_noise(4)+var_noise(8)+var_noise(12)+var_noise(16)-var_noise
            (2)-var_noise(6)-var_noise(10)-var_noise(14))^2;
    end
        out_add = [out_add ampl_add];
        out_dif = [out_dif ampl_dif];
        out_add_noise = [out_add_noise ampl_add_noise];
        out_dif_noise = [out_dif_noise ampl_dif_noise];
end

figure;
stem (axis_n, out_add);
title('Add Without Noise');

figure;
stem (axis_n, out_dif);
title('Dif Without Noise');

figure;
stem (axis_n, out_add_noise);
title('Add With Noise');

figure;
stem(axis_n, out_dif_noise);
title('Dif With Noise');

figure;
plot (axis_n, out_add);
title('Add Without Noise');

figure;
plot (axis_n, out_dif);
title('Dif Without Noise');

figure;
plot (axis_n, out_add_noise);
title('Add With Noise');

```

```
figure;
plot(axis_n, out_dif_noise);
title('Dif With Noise');
```

```
%{A LA SALIDA TENEMOS [SAMPLES-(N-1) = SAMPLES-N+1] MUESTRAS%}
%%Especifica que señales van a ser procesadas por la DFT %%
%signal = current;
%signal_noise = current_noise;
function [] = amplitude_test_4_new (signal, signal_noise, window)
%%N have to be power of 2 and bigger or same than 4;
samples       = length (signal);
axis_n        = 1:samples-window+1;
out_add       = zeros (1,0);
out_dif       = zeros (1,0);
out_add_noise = zeros (1,0);
out_dif_noise = zeros (1,0);

for m = 1:samples-window+1 %should be while true;
    sum1       = 0;
    sum2       = 0;
    sum3       = 0;
    sum4       = 0;
    sum_noise1 = 0;
    sum_noise2 = 0;
    sum_noise3 = 0;
    sum_noise4 = 0;

    for p = 1:window
        var(p)       = signal (m+p-1);
        var_noise(p) = signal_noise (m+p-1);
    end

    for k = 1:4:window
        sum1       = sum1 + var(k);
        sum2       = sum2 + var(k+2);
        sum3       = sum3 + var(k+3);
        sum4       = sum4 + var(k+1);
        sum_noise1 = sum_noise1 + var_noise(k);
        sum_noise2 = sum_noise2 + var_noise(k+2);
        sum_noise3 = sum_noise3 + var_noise(k+3);
        sum_noise4 = sum_noise4 + var_noise(k+1);
    end

    ampl_add       = (sum1 + sum2) ^2 + (sum3 + sum4) ^2;
    ampl_dif       = (sum1 - sum2) ^2 + (sum3 - sum4) ^2;
    ampl_add_noise = (sum_noise1 + sum_noise2) ^2 + (sum_noise3 +
       sum_noise4) ^2;
    ampl_dif_noise = (sum_noise1 - sum_noise2) ^2 + (sum_noise3 -
```

```
       sum_noise4) ^2;

    out_add       = [out_add ampl_add];
    out_dif       = [out_dif ampl_dif];
    out_add_noise = [out_add_noise ampl_add_noise];
    out_dif_noise = [out_dif_noise ampl_dif_noise];
end

figure;
stem (axis_n, out_add);
title('Add Without Noise');

figure;
stem (axis_n, out_dif);
title('Dif Without Noise');

figure;
stem (axis_n, out_add_noise);
title('Add With Noise');

figure;
stem(axis_n, out_dif_noise);
title('Dif With Noise');

figure;
plot (axis_n, out_add);
title('Add Without Noise');

figure;
plot (axis_n, out_dif);
title('Dif Without Noise');

figure;
plot (axis_n, out_add_noise);
title('Add With Noise');

figure;
plot(axis_n, out_dif_noise);
title('Dif With Noise');
```

El siguiente archivo genera el espectro dinámico de la capacidad del sistema creado.

```
% A LA SALIDA TENEMOS [SAMPLES-(N-1) = SAMPLES-N+1] MUESTRAS
% Window have to be power of 2 same or less than 2^14;

close all;

window         = 2^6; %
samples        = length (cap_1);

axis_time      = 0:5e-6:(samples-1)*5e-6;

%axis_fft       = -(4e5-(4e5/samples))/2:4e5/samples:(4e5-(4e5/samples))
   /2;
axis_fft       = -(4e5-(4e5/window))/2:4e5/window:(4e5-(4e5/window))/2;

spectrum       = zeros (1,0);
spectrum_noise = zeros (1,0);

for m = 1:samples-window+1 %should be while true;
    part           = fft(cap_1(m:m+window-1), window);
    part_noise     = fft(cap_noise_1(m:m+window-1), window);
    spectrum       = [spectrum part'];
    spectrum_noise = [spectrum_noise part_noise'];
end

spectrum       = abs(spectrum).^2;
spectrum_noise = abs(spectrum_noise).^2;

for k = 1:window-1 %should be not exist.
temp           = zeros(1, window);
temp_noise     = zeros(1, window);
spectrum       = [spectrum temp'];
spectrum_noise = [spectrum_noise temp_noise'];
end

figure;
mesh (axis_time, axis_fft, spectrum);
figure;
mesh (axis_time, axis_fft, spectrum_noise);
```

El siguiente archivo calcula el error absoluto de las *DFTs* realizadas.

```
1
2  %Total error
3  error_total = sum((signal_noise_6 - signal_1).^2);
4
5  %Medium quadratic initial error.
6  error_1       = sum((signal_1 - signal_1).^2);
7  error_noise_1 = sum((signal_noise_1 - signal_1).^2);
8
9  %Medium quadratic error.
10 error_4 = sum((signal_4 - signal_1).^2);
11 error_noise_4 = sum((signal_noise_4 - signal_4).^2);
12
13 %Medium quadratic error
14 error_6 = sum((signal_6 - signal_5).^2);
15 error_noise_6 = sum((signal_noise_6 - signal_6).^2);
```

El siguiente archivo realiza la *DFTs* con el comando proporcionado por Matlab.

```
% A LA SALIDA TENEMOS [SAMPLES-(N-1) = SAMPLES-N+1] MUESTRAS
% Window have to be power of 2 and bigger or same than 4;
function [out_dif, out_dif_noise] = fft_test (signal, signal_noise, window
   )
samples       = length (signal);
out_dif       = zeros (1,0);
out_dif_noise = zeros (1,0);

for m = 1:samples-window+1 %should be while true;
    sum1       = 0;
    sum2       = 0;
    sum3       = 0;
    sum4       = 0;
    sum_noise1 = 0;
    sum_noise2 = 0;
    sum_noise3 = 0;
    sum_noise4 = 0;

    for p = 1:window
        var(p)       = signal (m+p-1);
        var_noise(p) = signal_noise (m+p-1);
    end

    for k = 1:4:window
        sum1       = sum1 + var(k);
        sum2       = sum2 + var(k+2);
        sum3       = sum3 + var(k+3);
        sum4       = sum4 + var(k+1);
        sum_noise1 = sum_noise1 + var_noise(k);
        sum_noise2 = sum_noise2 + var_noise(k+2);
        sum_noise3 = sum_noise3 + var_noise(k+3);
        sum_noise4 = sum_noise4 + var_noise(k+1);
    end

    ampl_dif       = (sum1 - sum2) ^2 + (sum3 - sum4) ^2;
    out_dif        = [out_dif ampl_dif];
    ampl_dif_noise = (sum_noise1 - sum_noise2) ^2 + (sum_noise3 -
       sum_noise4) ^2;
    out_dif_noise  = [out_dif_noise ampl_dif_noise];
end
temp           = ampl_dif * ones(1, window-1);
temp_noise     = ampl_dif_noise * ones(1, window-1);
```

```
41 out_dif        = [out_dif temp];
42 out_dif_noise  = [out_dif_noise temp_noise];
```

El siguiente archivo realiza la *DFTs* con el comando proporcionado por Matlab pero optimizada realizandola primero a las muestras pares y luego a las impares.

```
%% TIME OPTIMIZED
%Data:
samples = 1024;
window = 4;
amplitude = 11.5;
analogic_frequency = 2*pi*100e3;
digital_frequency = 2*pi*500e3;

%%Analogic Signal comes TPS:
axis_t = (0:samples-1)./analogic_frequency;
white_noise = random ('norm', 0, 1, [1,samples]);
analogic_signal=amplitude*exp(1i*analogic_frequency.*axis_t);
%PODRÍA MODELAR EL RUIDO DEL CABLE AQUÍ (NO INTERESA PARA EL ESTUDIO
%ACTUAL)
analogic_signal_with_noise = analogic_signal + white_noise;

%Digital Signal after ADC:
order = ceil(digital_frequency/analogic_frequency);
axis_n = 0:(samples*order)-1;
digital_signal = interp(analogic_signal_with_noise,order);

%%
%Cálculo de la DFT de secuencias reales (PAG 25 LIBRO TDS)
%Un procedimiento para ahorrarse operaciones en el cálculo de la DFT es convertir una
%señal real x[n] de N puntos en una señal compleja g[n] de N/2 puntos y realizar la DFT
%de N/2 puntos. Para verificar esta posibilidad, realice los siguientes pasos:
%1. Genere 64 muestras de un seno s[n] de pulsación ?=0.3?. Calcule su DFT S[k]
%2. A partir de la señal s[n], obtenga la señal g[n]=s[2n]+js[2n+1]
%3. Obtenga G[k] como la fft de g[n].
%4. Obtenga la DFT de s[2n] a partir de G[k] como Spares[k]= (G[k]+G*[-k])/2,
%k=0,1,....,31.
%5. Obtenga la DFT de s[2n+1] como Simpares[k]=-j (G[k]-G*[-k])/2, k=0,1,....,31.
%6. Recupere la DFT S[k] de s[n] como: Spares[k]+e-j2?k/N Simpares[k] , k=0,1,....,63.
%7. Para comprobar que ambos procedimientos son equivalentes, calcule el
```

```
   error
%cuadrático medio entre S[k] y Spares[k]+e−j2?k/N Simpares[k]

tic;
fft_8192 = fft(digital_signal,8192);
stem (0:8191, fft_8192);
toc;
% Elapsed time is 0.303567 seconds.

tic;
digital_signal_2 = digital_signal(1:2:end) + 1i* digital_signal(2:2:end);
fft_8192 = fft(digital_signal_2,8192);
pares = (fft_8192 (1:4096) + conj(fft_8192(1:4096)))/2;
impares = −1i*(fft_8192 (1:4096) − conj(fft_8192(1:4096)))/2;
fft_final = pares + exp(−2i*pi*(1:4096)/4096).*impares;
stem (0:4095, fft_final);
toc;
% Elapsed time is 0.046858 seconds.
```

El siguiente archivo modela todo el sistema y como varía la capacidad a través de los diferentes subsistemas del procesado de señal, los filtros, etc.

```
%%wc = 1 correspondig to half the sample rate.
%See Capacitance.
ploter (cap_1, cap_noise_1);
title('Capacitance 1');
%See capacitance spectrum.
plot_spectrum (cap_1, cap_noise_1);
title('Capacitance spectrum 1');

%See signal.
signal_1       = sqrt(-cap_1);
signal_noise_1 = sqrt(-cap_noise_1);
ploter (signal_1, signal_noise_1);
title('Signal 1');
%See signal spectrum.
plot_spectrum (signal_1, signal_noise_1);
title('Signal spectrum 1');

%% AAF
if strcmp (filt_type, 'butter')
    [B, A] = butter (order, wc2, 'low');
elseif strcmp (filt_type, 'cheby1')
    [B, A] = cheby1 (order, 0.5, wc2, 'low');
elseif strcmp (filt_type, 'cheby2')
    [B, A] = cheby2 (order, 60, wc2, 'low');
elseif strcmp (filt_type, 'ellip')
    [B, A] = ellip (order, 0.5, 20, wc2, 'low');
end

%Transfer function
[Haaf,w2] = freqz(B,A,500);
Haaf      = abs(Haaf).^2;
figure;
%plot(0:2*pi/499:2*pi, Haaf);
plot(0:4e5/500:4e5-(4e5/500), Haaf);
title('Transfer Function AAF');

%Filter.
cap_2       = filter (B, A, cap_1);
cap_noise_2 = filter (B, A, cap_noise_1);

%See capacitance.
```

```
ploter (cap_2, cap_noise_2);
title('Capacitance 2');
%See capacitance spectrum.
plot_spectrum (cap_2, cap_noise_2);
title('Capacitance spectrum 2');

%See signal.
signal_2       = sqrt(-cap_2);
signal_noise_2 = sqrt(-cap_noise_2);
ploter (signal_2, signal_noise_2);
title('signal 2');
%See signal spectrum.
plot_spectrum (signal_2, signal_noise_2);
title('Signal spectrum 2');

%%ADC
cap_3       = cap_2;
white_noise = random ('norm', 0, 0.1, [1,length(axis_n)]);
cap_noise_3 = cap_noise_2;% + white_noise;

%See capacitance.
ploter (cap_3, cap_noise_3);
title('Capacitance 3');
%See capacitance spectrum.
plot_spectrum (cap_3, cap_noise_3);
title('Capacitance spectrum 3');

%See signal.
signal_3       = sqrt(-cap_3);
signal_noise_3 = sqrt(-cap_noise_3);
ploter (signal_3, signal_noise_3);
title('signal 3');
%See signal spectrum.
plot_spectrum (signal_3, signal_noise_3);
title('Signal spectrum 3');
%%Before DFT Filter.
if strcmp (filt_type, 'butter')
    [C, D] = butter (order, wc4);
elseif strcmp (filt_type, 'cheby1')
    [C, D] = cheby1 (order, 0.5, wc4);
elseif strcmp (filt_type, 'cheby2')
    [C, D] = cheby2 (order, 60, wc4);
elseif strcmp (filt_type, 'ellip')
    [C, D] = ellip (order, 0.5, 20, wc4);
end
```

```
87
88  %Transfer function
89  [Hdft,w4] = freqz(C,D,500);
90  Hdft      = abs(Hdft).^2;
91  figure;
92  %plot(0:2*pi/499:2*pi, Hdft);
93  plot(0:4e5/500:4e5-(4e5/500), Hdft);
94  title('Transfer Function pre DFT Filter');
95
96  %Filter.
97  cap_4       = filter (C, D, cap_3);
98  cap_noise_4 = filter (C, D, cap_noise_3);
99
100 %See capacitance.
101 ploter (cap_4, cap_noise_4);
102 title('Capacitance 4');
103 %See capacitance spectrum.
104 plot_spectrum (cap_4, cap_noise_4);
105 title('Capacitance spectrum 4');
106
107 %See signal.
108 signal_4       = sqrt(-cap_4);
109 signal_noise_4 = sqrt(-cap_noise_4);
110 ploter (signal_4, signal_noise_4);
111 title('signal 4');
112 %See signal spectrum.
113 plot_spectrum (signal_4, signal_noise_4);
114 title('Signal spectrum 4');
115
116 %%DFT
117 [cap_5, cap_noise_5] = fft_test(cap_4, cap_noise_4, 512);
118
119 %See capacitance.
120 ploter (cap_5, cap_noise_5);
121 title('Capacitance 5');
122 %See capacitance spectrum.
123 plot_spectrum (cap_5, cap_noise_5);
124 title('Capacitance spectrum 5');
125
126 %See signal.
127 signal_5       = sqrt(-cap_5);
128 signal_noise_5 = sqrt(-cap_noise_5);
129 ploter (signal_5, signal_noise_5);
130 title('signal 5');
131 %See signal spectrum.
```

```
plot_spectrum (signal_5, signal_noise_5);
title('Signal spectrum 5');

%%After DFT Filter.
if strcmp (filt_type, 'butter')
    [E, F] = butter (order, wc6);
elseif strcmp (filt_type, 'cheby1')
    [E, F] = cheby1 (order, 0.5, wc6);
elseif strcmp (filt_type, 'cheby2')
    [E, F] = cheby2 (order, 60, wc6);
elseif strcmp (filt_type, 'ellip')
    [E, F] = ellip (order, 0.5, 20, wc6);
end

Transfer function
[Hend,w4] = freqz(E,F,500);
Hend      = abs(Hend).^2;
figure;
%plot(0:2*pi/499:2*pi, Hend);
plot(0:4e5/500:4e5-(4e5/500), Hend);
title('Transfer Function final filter');

Filter.
cap_6       = filter (E, F, cap_5);
cap_noise_6 = filter (E, F, cap_noise_5);

See capacitance.
ploter (cap_6, cap_noise_6);
title('Capacitance 6');
%See capacitance spectrum.
plot_spectrum (cap_6, cap_noise_6);
title('Capacitance spectrum 6');

%See signal.
signal_6       = sqrt(-cap_6);
signal_noise_6 = sqrt(-cap_noise_6);
ploter (signal_6, signal_noise_6);
title('signal 6');
%See signal spectrum.
plot_spectrum (signal_6, signal_noise_6);
title('Signal spectrum 6');
```

El siguiente archivo modela todo el sistema y como varía la medida de la distancia a través de los diferentes subsistemas del procesado de señal, los filtros, etc.

```
%%wc = 1 correspondig to half the sample rate.
close all;

%See Capacitance.
cap_1 = -(abs(signal_1).^2);
cap_noise_1 = -(abs(signal_noise_1).^2);
ploter (cap_1, cap_noise_1);
title('Capacitance 1');
%See signal.
ploter (signal_1, signal_noise_1);
title('Signal 1');
%See signal spectrum.
espectro (signal_1, signal_noise_1);
title('Spectrum 1');

%%AAF
if strcmp (filt_type, 'butter')
    [B, A] = butter (order, wc2, 'low');
elseif strcmp (filt_type, 'cheby1')
    [B, A] = cheby1 (order, 0.5, wc2, 'low');
elseif strcmp (filt_type, 'cheby2')
    [B, A] = cheby2 (order, 60, wc2, 'low');
elseif strcmp (filt_type, 'ellip')
    [B, A] = ellip (order, 0.5, 20, wc2, 'low');
end

%Transfer function
[Haaf,w2] = freqz(B,A,500);
Haaf = abs(Haaf).^2;
figure;
plot(0:2*pi/499:2*pi, Haaf);
title('Transfer Function AAF');

%Filter.
signal_2 = filter (B, A, cap_1);
signal_noise_2 = filter (B, A, signal_noise_1);

%See signal.
ploter (signal_2, signal_noise_2);
title('Signal 2');

```

```
%See Capacitance.
cap_2 = -(abs(signal_2).^2);
cap_noise_2 = -(abs(signal_noise_2).^2);
ploter (cap_2, cap_noise_2);
title('Capacitance 2');

%See signal spectrum.
espectro (signal_2, signal_noise_2);
title('Spectrum 2');

%% ADC
signal_3       = signal_2;
white_noise    = random ('norm', 0, 0.5, [1,length(axis_n)]);
signal_noise_3 = signal_noise_2; % + white_noise;

%See signal.
ploter (signal_3, signal_noise_3);
title('Signal 3');

%See Capacitance.
cap_3 = -(abs(signal_3).^2);
cap_noise_3 = -(abs(signal_noise_3).^2);
ploter (cap_3, cap_noise_3);
title('Capacitance 3');

%See signal spectrum.
espectro (signal_3, signal_noise_3);
title('Spectrum 3');

%% Before DFT Filter.
if strcmp (filt_type, 'butter')
    [C, D] = butter (order, wc4);
elseif strcmp (filt_type, 'cheby1')
    [C, D] = cheby1 (order, 0.5, wc4);
elseif strcmp (filt_type, 'cheby2')
    [C, D] = cheby2 (order, 60, wc4);
elseif strcmp (filt_type, 'ellip')
    [C, D] = ellip (order, 0.5, 20, wc4);
end

%Transfer function
[Hdft,w4] = freqz(C,D,500);
Hdft = abs(Hdft).^2;
figure;
plot(0:2*pi/499:2*pi, Hdft);
```

```
title('Transfer Function pre DFT Filter');

%Filter.
signal_4 = filter (C, D, signal_3);
signal_noise_4 = filter (C, D, signal_noise_3);

%See signal.
ploter (signal_4, signal_noise_4);
title('Signal 4');

%See Capacitance.
cap_4 = -(abs(signal_4).^2);
cap_noise_4 = -(abs(signal_noise_4).^2);
ploter (cap_4, cap_noise_4);
title('Capacitance 4');

%See signal spectrum.
espectro (signal_4, signal_noise_4);
title('Spectrum 4');

%% DFT
[signal_5, signal_noise_5] = fft_test(signal_4, signal_noise_4, 16);

%See signal.
ploter (signal_5, signal_noise_5);
title('Signal 5');

%See Capacitance.
cap_5 = -(abs(signal_5).^2);
cap_noise_5 = -(abs(signal_noise_5).^2);
ploter (cap_5, cap_noise_5);
title('Capacitance 5');

%See signal spectrum.
espectro (signal_5, signal_noise_5);
title('Spectrum 5');

%% After DFT Filter.
if strcmp (filt_type, 'butter')
    [E, F] = butter (order, wc6);
elseif strcmp (filt_type, 'cheby1')
    [E, F] = cheby1 (order, 0.5, wc6);
elseif strcmp (filt_type, 'cheby2')
    [E, F] = cheby2 (order, 60, wc6);
elseif strcmp (filt_type, 'ellip')
```

```
    [E, F] = ellip (order, 0.5, 20, wc6);
end

%Transfer function
[Hend,w4] = freqz(E,F,500);
Hend = abs(Hend).^2;
figure;
plot(0:2*pi/499:2*pi, Hend);
title('Transfer Function final filter');

%Filter.
signal_6 = filter (E, F, signal_5);
signal_noise_6 = filter (E, F, signal_noise_5);

%See signal.
ploter (signal_6, signal_noise_6);
title('Signal 6');

%See Capacitance.
cap_6 = -(abs(signal_6).^2);
cap_noise_6 = -(abs(signal_noise_6).^2);
ploter (cap_6, cap_noise_6);
title('Capacitance 6');

%See signal spectrum.
espectro (signal_6, signal_noise_6);
title('Spectrum 6');
```

El siguiente archivo muestra la comparativa entre la señal y el ruido en cada tramo del sistema.

```
%% Noise vs Signal Test.
%Data:
samples = 16384;
amplitude = 11.5;
analogic_frequency = 2*pi*100e3;
digital_frequency = 2*pi*400e3;

%% Analogic Signal comes TPS:
axis_t = (0:samples-1)./analogic_frequency;
white_noise = random ('norm', 0, 1, [1,samples]);
analogic_signal=20+ amplitude * sin(analogic_frequency.*axis_t);
analogic_signal_noise = analogic_signal + white_noise;

figure;
subplot (2,1,1)
plot (axis_t, analogic_signal);
subplot (2,1,2)
plot (axis_t, analogic_signal_noise);

%% Digital Signal after ADC:
order = ceil(digital_frequency/analogic_frequency);
axis_n = 0:(samples*order)-1;
digital_signal = interp(analogic_signal,order);
digital_signal_noise = interp(analogic_signal_noise,order);

figure;
subplot (2,1,1)
stem (axis_n, digital_signal);
subplot (2,1,2)
stem (axis_n, digital_signal_noise);

%% N=2;
window = 2;
axis_n = 1:2*samples/window;
out = zeros (1,samples/window);
out_noise = zeros (1,samples/window);

for m = 1:2:samples %should be while true;
    var0 = digital_signal(m);
    var1 = digital_signal(m+1);
    %ampl = var0^2+var1^2;
```

```
    ampl = var0^2-var1^2;
    out = [out ampl];

    var0_noise = digital_signal_noise(m);
    var1_noise = digital_signal_noise(m+1);
    %ampl_noise = var0_noise^2+var1_noise^2;
    ampl_noise = var0_noise^2-var1_noise^2;
    out_noise = [out_noise ampl_noise];
end

figure;
subplot (2,1,1)
stem (axis_n, out);
subplot (2,1,2)
stem (axis_n, out_noise);

%%N=4;
window = 4;
axis_n = 1:2*samples/window;
out = zeros (1,samples/window);
out_noise = zeros (1,samples/window);

for m = 1:window:samples %should be while true;
    var0 = digital_signal(m);
    var1 = digital_signal(m+1);
    var2 = digital_signal(m+2);
    var3 = digital_signal(m+3);
    ampl = (var0-var2)^2+(var3-var1)^2;
    %ampl = (var0+var2)^2+(var3+var1)^2;
    out = [out ampl];

    var0_noise = digital_signal_noise(m);
    var1_noise = digital_signal_noise(m+1);
    var2_noise = digital_signal_noise(m+2);
    var3_noise = digital_signal_noise(m+3);
    ampl_noise = (var0_noise-var2_noise)^2+(var3_noise-var1_noise)^2;
    %ampl_noise = (var0_noise+var2_noise)^2+(var3_noise+var1_noise)^2;

    out_noise = [out_noise ampl_noise];
end

figure;
subplot (2,1,1)
stem (axis_n, out);
subplot (2,1,2)
```

```
stem (axis_n, out_noise);

%%N=8;
window = 8;
axis_n = 1:2*samples/window;
out = zeros (1,samples/window);
out_noise = zeros (1,samples/window);

for m = 1:window:samples %should be while true;
    var0 = digital_signal(m);
    var1 = digital_signal(m+1);
    var2 = digital_signal(m+2);
    var3 = digital_signal(m+3);
    var4 = digital_signal(m+4);
    var5 = digital_signal(m+5);
    var6 = digital_signal(m+6);
    var7 = digital_signal(m+7);
    ampl = (var0+var4-var2-var6)^2+(var3+var7-var1-var5)^2;
    %ampl = (var0+var4+var2+var6)^2+(var3+var7+var1+var5)^2;
    out = [out ampl];

    var0_noise = digital_signal_noise(m);
    var1_noise = digital_signal_noise(m+1);
    var2_noise = digital_signal_noise(m+2);
    var3_noise = digital_signal_noise(m+3);
    var4_noise = digital_signal_noise(m+4);
    var5_noise = digital_signal_noise(m+5);
    var6_noise = digital_signal_noise(m+6);
    var7_noise = digital_signal_noise(m+7);
    ampl = (var0_noise+var4_noise-var2_noise-var6_noise)^2+(var3_noise+
       var7_noise-var1_noise-var5_noise)^2;
    %ampl = (var0_noise+var4_noise+var2_noise+var6_noise)^2+(var3_noise+
       var7_noise+var1_noise+var5_noise)^2;
    out_noise = [out_noise ampl_noise];
end

figure;
subplot (2,1,1)
stem (axis_n, out);
subplot (2,1,2)
stem (axis_n, out_noise);
```

El siguiente archivo muestra el espectro de la señal en cada tramo del sistema.

```
function [] = plot_spectrum (signal, signal_noise)

N = length(signal);
%Radian axis.
%axis_fft = -(pi-(2*pi/N)):2*pi/N:pi-(2*pi/N);
%Linear axis.
axis_fft = -(4e5-(4e5/N))/2:4e5/N:(4e5-(4e5/N))/2;

%See signal spectrum.
fft_signal = (abs(fft(signal,N)).^2);
fft_signal = [fft_signal(length(fft_signal)/2+1:end) fft_signal(1:length(
   fft_signal)/2)];
figure;
subplot (2,1,1)
plot (axis_fft, fft_signal);
%axis ([-2e5 2e5 -20 1e5]);
axis ([0 2e5 -20 1e5]);

fft_signal_noise = (abs(fft(signal_noise,N)).^2);
fft_signal_noise = [fft_signal_noise(length(fft_signal_noise)/2+1:end)
   fft_signal_noise(1:length(fft_signal_noise)/2)];
subplot (2,1,2)
plot (axis_fft, fft_signal_noise);
%axis ([-2e5 2e5 -20 1e5]);
axis ([0 2e5 -20 1e5]);
```

Anexo H: Datasheet Quickfilter Pro QF4A512

DATA SHEET

QF4A512

4-Channel Programmable Signal Converter (PSC)

APPLICATIONS

- **Industrial Control**
- **Wireless Sensor Networks**
- **Machine Monitoring**
- **Smart Sensors**
- **Medical Monitoring and Diagnostics**
- **Homeland Security**

DESCRIPTION

The QF4A512 Programmable Signal Converter is a 4-channel, signal conditioner and signal converter. Each channel can be individually programmed for the gain, anti-aliasing filter cutoff frequency, A-to-D sampling frequency, and unique filter requirements. This is accomplished with 4 separate high-precision 512-tap FIR filters.

Quickfilter Pro software has been created for rapid device configuration and filter design at performance levels unattainable with analog components.

ORDERING INFORMATION

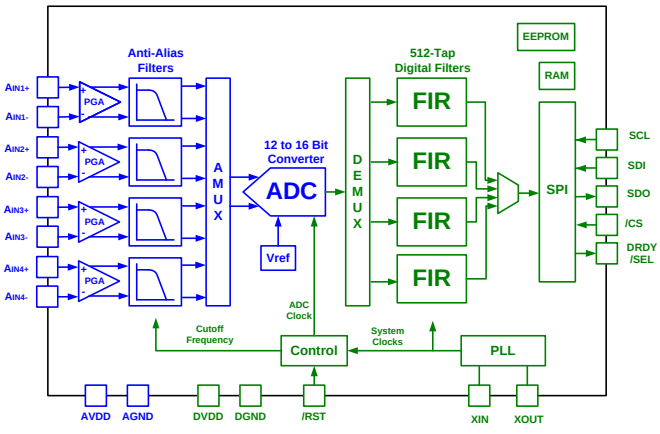
Device	Package
QF4A512A-LQ--T QF4A512A-LQ--B	32-Pin LQFP - Tape & Reel (Reel qty 1000) - Trays
QF4A512-DK	Development Kit

FEATURES

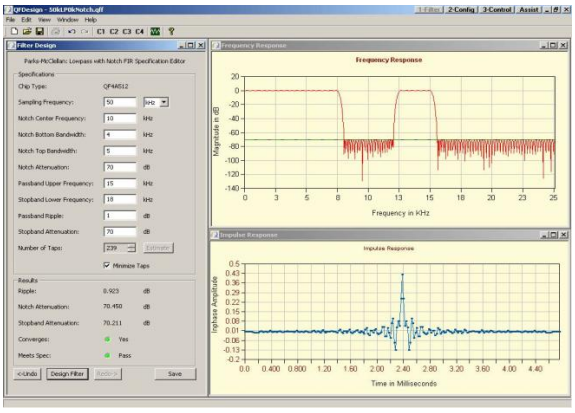
- **4 Channel Analog 16-bit Programmable A/D Converter**
- **Differential or Single Ended Inputs**
- **4 Programmable (1x, 2x, 4x, 8x) Gain Amplifiers**
- **Anti-Aliasing Filter Per Channel, 3rd Order Bessel**
- **Analog DC – 900kHz, up to 2Msps Sampling rate**
- **Internal Precision Voltage Reference**
- **4 Individual Programmable 512-tap Digital FIR Filters**
- **SPI Port Interface**
- **3.3V Digital I/O, 5 Volt Tolerant**
- **32-Pin LQFP Package**
- **Industrial Temp -40C to +85C**
- **4K Byte EEPROM for filter coefficient, chip configuration and calibration.**
  - 128 bytes of EEPROM User Data Space
  - 384 Bit Masks for IEEE 1451.4 TEDS on 4 Channels

QUICKFILTER DEVELOPMENT KIT (QF4A512-DK)

- **Quickfilter Pro Windows®-based Software for Rapid Filter Design and IC Configuration**
- **In-System Programmability (ISP) trough SPI Port**
- **Evaluation board for verification of device performance**



Functional Block Diagram



Filter Design Screen

1. SPECIFICATIONS

1.1 Absolute Maximum Ratings

Stresses above those listed under Absolute Maximum Ratings can cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

Parameter	Min	Max	Units
Storage Temperature	-60	125	ºC
Supply Voltage, V_{DD18} to DGND	-0.2V	2.2V	V
Supply Voltage, V_{DD33} with respect to DGND	-0.2V	4.0V	V
Digital Input Voltage with respect to DGND	-0.3	7	V
Analog Input Voltage with respect to AGND	-0.3	V_{DD33} + 0.3	V
ESD Immunity (Human Body Model, JESD222 Class 1C))	1000		V

This integrated circuit can be damaged by ESD. Quickfilter Technologies recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

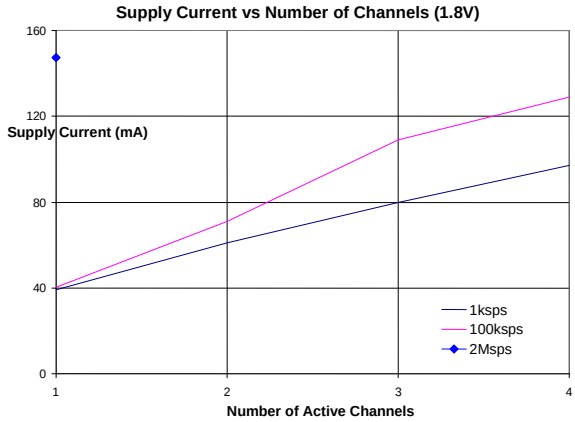
1.2 Package Assembly

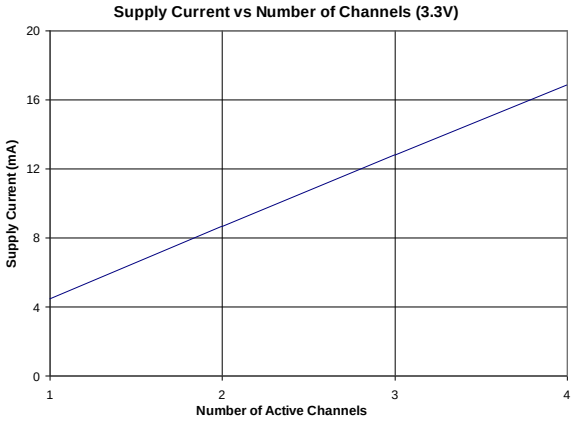
The QF4A512 is offered in a “green” package (RoHS & no Sb/Br), assembled with enhanced environmentally compatible Pb-free and halide-free materials. The leads possess a matte-tin plating which is compatible with conventional board assembly processes or newer lead-free board assembly processes. The peak soldering temperature should not exceed 260°C during printed circuit board assembly.

1.3 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Units
Supply Voltage 1.8	V_{DD18}	1.6	1.8	2.0	V
Supply Voltage 3.3	V_{DD33}	3.0	3.3	3.6	V
Digital Input Voltage		0		5.5	V
Analog Input Voltage	A_{IN}	0.2		2.5	V
Clock Frequency	f_0	5	20	200	MHz
Ambient Temperature	T_A	-40	25	85	ºC
ADC Clock Rate	f_{ADC}	10		100	MHz

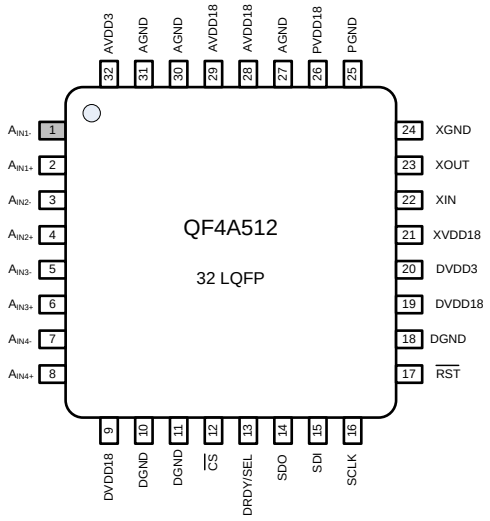
Note: Quickfilter guarantees the performance of this device over specified ranges by conducting electrical characterization over each range and by conducting a production test with single insertion coupled to periodic sampling.





2. PINOUT and PIN DESCRIPTIONS

2.1 Pinout



2.2 Pin Descriptions

Pin #	Pin Name	Type	Description
1	A_{IN1-}	Input	Analog Input Channel 1 Differential-, or DC bias input (SE)
2	A_{IN1+}	Input	Analog Input Channel 1, Single-Ended or Differential+ (No phase shift)
3	A_{IN2-}	Input	Analog Input Channel 2 Differential-, or DC bias input (SE)
4	A_{IN2+}	Input	Analog Input Channel 2 , Single-Ended or Differential+ (No phase shift)
5	A_{IN3-}	Input	Analog Input Channel 3 Differential-, or DC bias input (SE)
6	A_{IN3+}	Input	Analog Input Channel 3, Single-Ended or Differential+ (No phase shift)
7	A_{IN4-}	Input	Analog Input Channel 4 Differential-, or DC bias input (SE)
8	A_{IN4+}	Input	Analog Input Channel 4, Single-Ended or Differential+ (No phase shift)

3. GENERAL DESCRIPTION

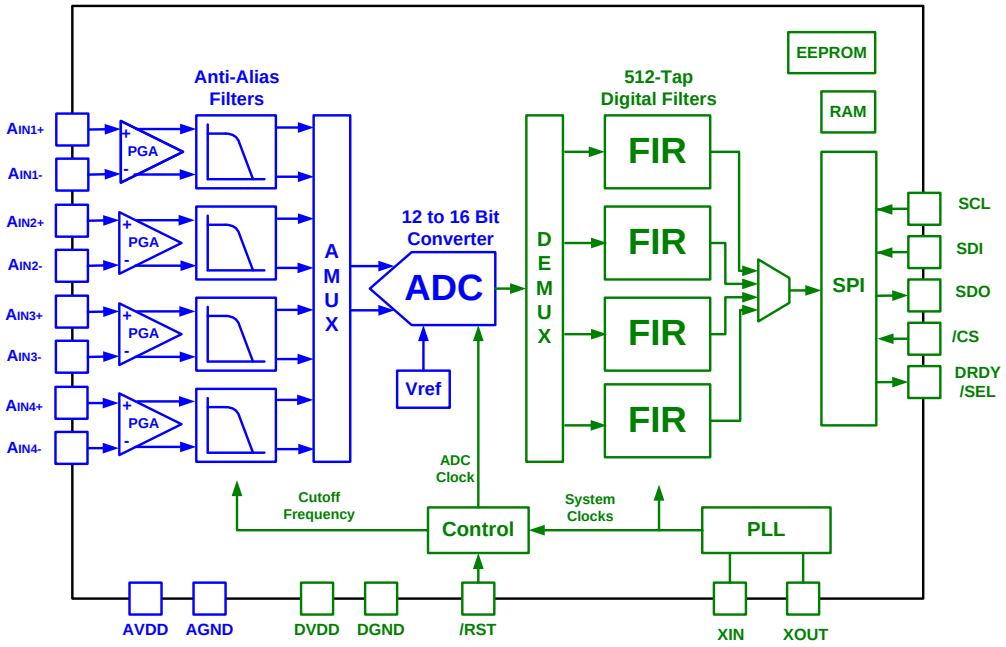


Figure 1. Functional Block Diagram

3.1 Analog Front End (AFE)

The AFE consists of a Programmable Gain Amplifier (PGA), a chopper-stabilized amplifier and an Anti-Aliasing Filter.

Programmable Gain Amplifier (PGA)

The PGA can be set at gains of 1X, 2X, 4X, and 8X. The input impedance of the PGA is 10kΩ on both the positive and negative inputs. The PGA can be configured as either single ended or differential and can receive inputs of up to 2.0V p-p directly. With a single scaling resistor in each channel, two if configured differentially, the PGA can receive signals of up to +/- 10Vp-p or higher.

Chopper-stabilized Amplifier (not shown in diagram)

This circuit minimizes correlated (1/f) noise and dc offset within the chip. For sampling rates less than 200kHz this circuit will maximize signal-to-noise performance, and hence SINAD and ENOB. (see Section 6.2)

Anti-Aliasing Filter (AAF)

The Anti-Aliasing Filter is designed to reject frequencies that are higher than the band of interest. If those frequencies are sent to the ADC, they can alias back into the band of interest and can cause erroneous readings to result. The AAF is a 3rd order Bessel function (linear phase) and is set to the appropriate cut-off frequency based on the filter design that is implemented. The AAF has two available cut off frequencies, 500kHz and 3MHz.

3.2 Analog to Digital Converter (ADC)

The ADC has a pipeline architecture that is 12 bits in hardware and runs at up to 100Msps. Resolutions of up to 16 bits are achieved by oversampling the input and averaging the resultant conversions. During Chip Configuration and Filter Design, the exact sampling speed of the ADC is determined (based on the highest sampling rate required for any one of the four channels).

Not shown in the block diagram are the following sub-blocks which handle decimation/down-conversion of the oversampled data:

CIC (Cascaded Integrator Comb Filters)

The purpose of the CIC filter is the integration of 16 bits, and adjustment of the proper sample rate through decimation. The digitized signal is then processed in the CIH (Cascaded Integrator Halfband Filters)

CIH (Cascaded Integrator Halfband Filters)

5. STARTUP

The behavior of the QF4A512 during power up or after a reset can be determined by the configuration of 2 bits in the **STARTUP_1** register. The **auto_config** bit, if set, will initiate transfer of EEPROM contents to the control registers and FIR filter coefficient RAMs.

The **auto_start** bit will determine whether the chip starts in Run mode, filtering and sending data out on the SPI bus (**auto_start**=1), or the chip will wait in configure mode until manually started (**auto_start**=0).

If the **auto_config** bit is not set, the chip will wait in configure mode until externally programmed.

5.1 Power up / Reset Sequence

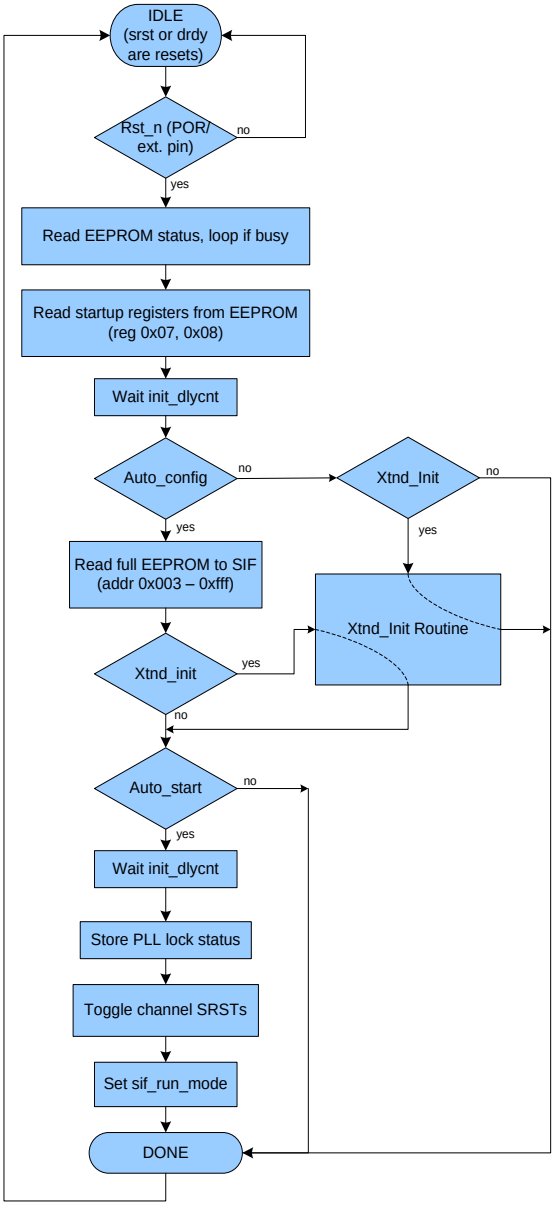


Figure 3. Power up / Reset Sequence

In Configure mode 14-bit address words are used. In Run mode 8-bit addressing is used. These differences are shown graphically in the following diagram. Consequently the host controller must use the appropriate timing depending on which mode of operation is active. (At power up the mode of operation is determined by the value of the **auto_start** bit (register **STARTUP_1**).

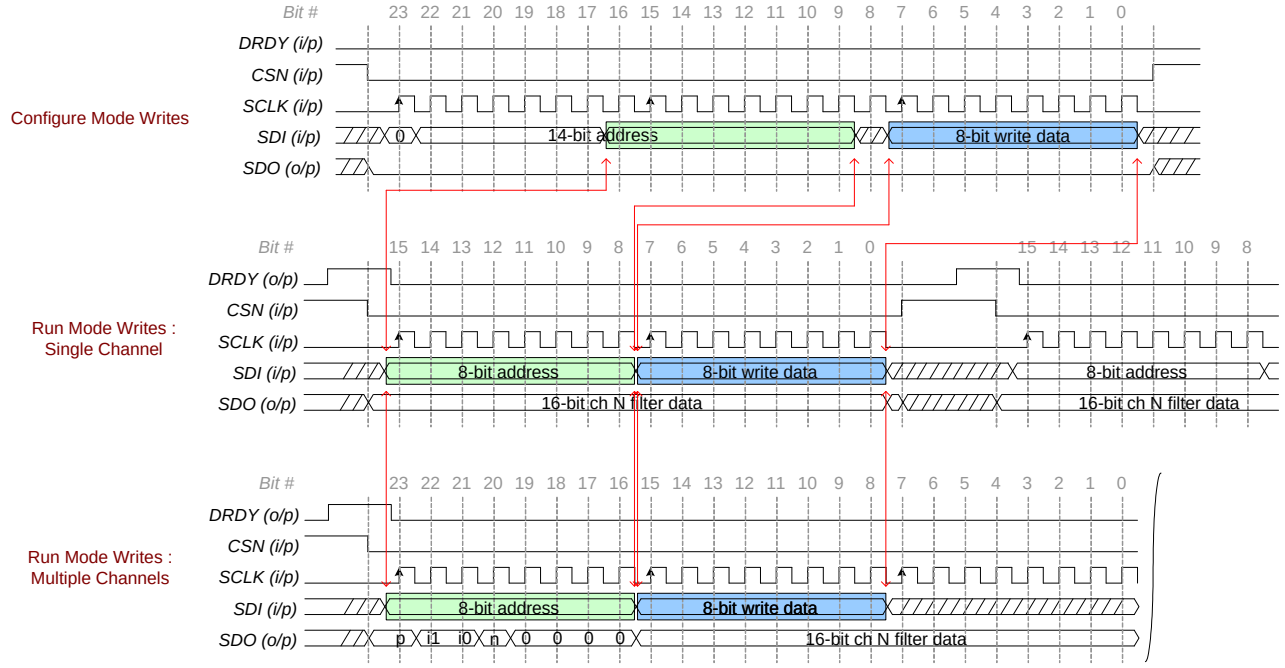


Figure 13. Comparison of Configure and Run Mode Timing

If it is unknown which mode is active there are several ways to determine this. If in Run mode there should be a continuous stream of conversions occurring which should be reflected by changes in level of the DRDY pin, furthermore if SCLK is active and CSN is held low there should be transitions occurring on the SDO pin. These pins can me monitored for activity within a pre-determined timeout interval. These methods are summarized below.

1. If the QF4A512 is in Run mode, DRDY will be driven high when the fastest sample is available. If in Configure mode, DRDY is an input (internal pull-down in the chip). So, by monitoring DRDY you can determine the mode of operation.
2. Holding SDI low (to prevent inadvertently writing any bad info to the chip), toggle SCLK and look for data on SDO. If in Configure mode, SDO will be low the whole time. If in Run mode, SDO will eventually toggle.

10.5 Multiple QF4A512s and Synchronous Sampling

A single QF4A512 can sample 4 analog signals simultaneously. However, some applications sample more than 4 channels using multiple QF4A512 chips. Multi-chip configurations of QF4A512 are described in App Note QFAN005 “Multiple Chip Configuration”. Some systems additionally require that the samples across multiple chips be tightly synchronized. App Note QFAN020 “Synchronized Sampling Using Multiple QF4A512 Programmable Signal Converters” describes a simple additional procedure to synchronize multiple QF4A512 devices so that all analog signals are sampled at the same moment and all DRDY signals assert synchronously.

13. APPLICATION CIRCUITS

For more information please see Application note QFAN004 at http://www.quickfiltertech.com/html/app_notes.php

13.1 AC Coupled, Single-ended

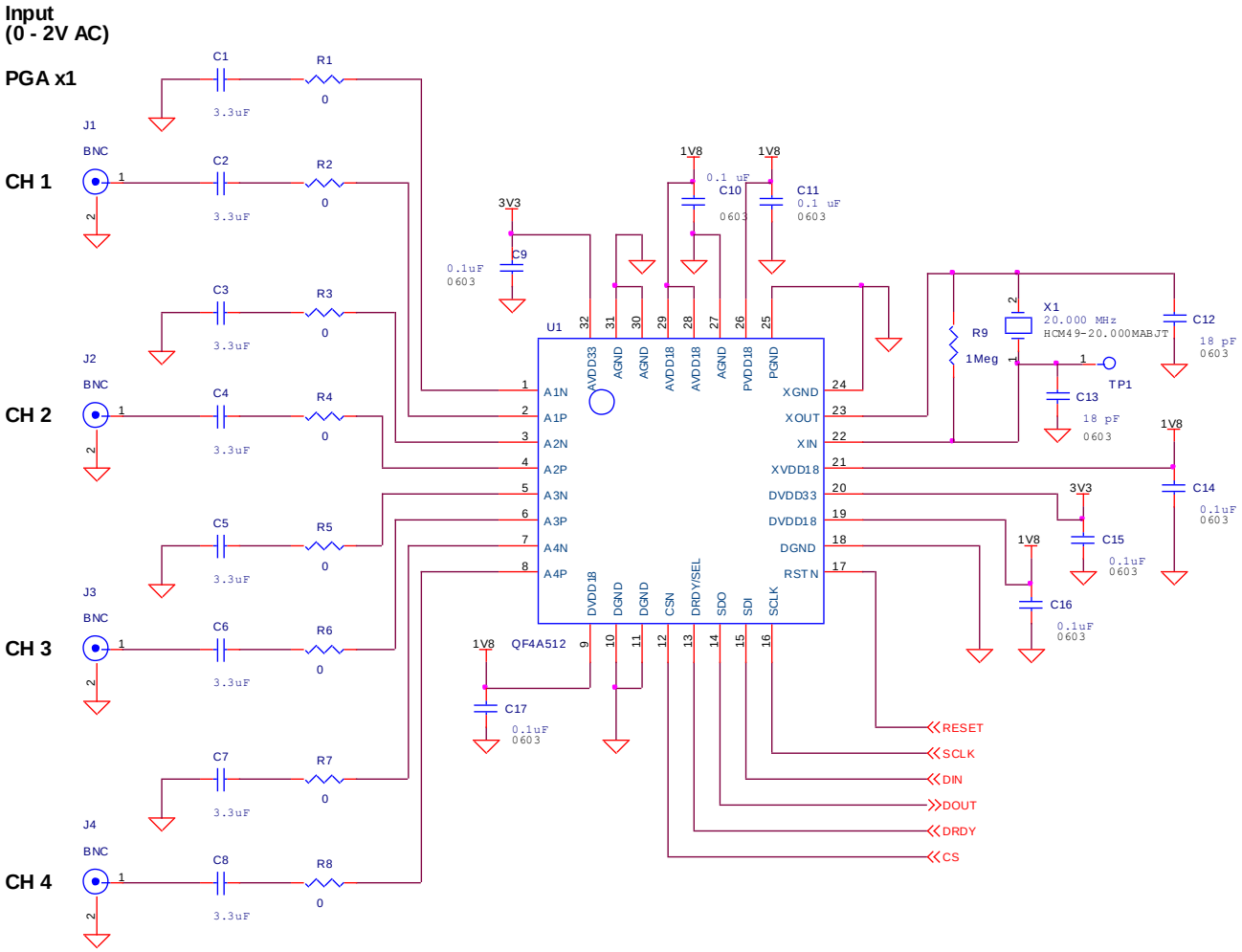


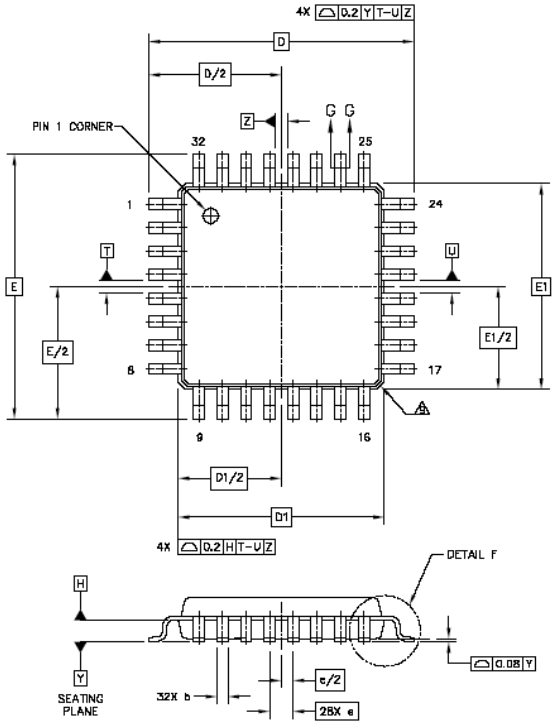
Figure 14. Application Circuit – Single-ended, ac coupled

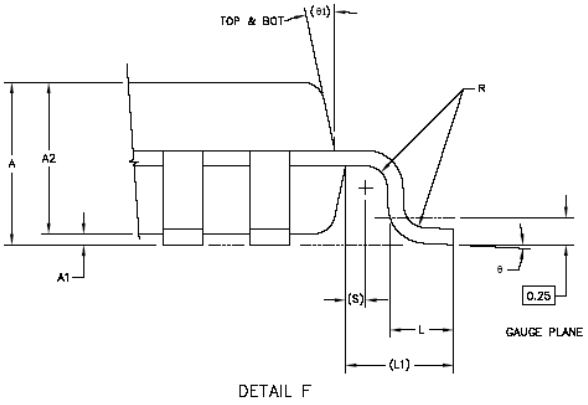
Table 20. Example resistor values for ac-coupled, single-ended

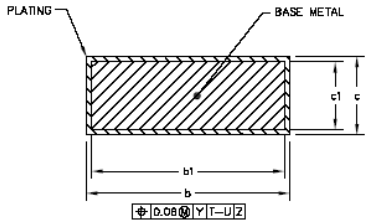
AC-coupled, singled-ended		
Input level	R1 - R8 value	PGA setting
1V ptp	0k	x2
3.3V ptp	6.5k	x1
5V ptp	15k	x1
10V ptp	40k	x1

14. PACKAGING INFORMATION

7x7x1.4mm, LQFP 32, 0.8 mm Pitch POD (JEDEC)







Dimension	Minimum (mm)	mm	Maximum (mm)
A	1.4		1.6
A1	0.05		0.15
A2	1.35		1.45
b	0.3		0.45
b1	0.3		0.4
c	0.09		0.2
c1	0.09		0.178
D		9 BSC	
D1		7 BSC	
e		0.8 BSC	
E		9 BSC	
E1		7BSC	
L	0.5		0.7
L1		1 REF	
R	0.15		0.25
S		0.2 REF	
θ	1°		5°
θ1		12° REF	

Notes:

1. Dimensions are in millimeters.
2. Interpret dimensions and tolerance per ASME Y14.5M-1994
3. Datum Plane H is located at bottom of lead and is coincident with the lead where the lead exits the plastic body at the bottom of the parting line.
4. Datum T, U, and Z to be determined at Datum Plane H.
5. Dimensions D and E to be determined at seating Plane Y.
6. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 do include mold mismatch and are determined at datum plane H.
7. Dimension b does not include Dam Bar protrusion. Dam Bar protrusion shall not cause the b dimension to exceed 0.533 mm.
8. Minimum solder plate thickness shall be 0.0076 mm.
9. Exact shape of each corner may vary from depiction.

Anexo I: Documentación contenida en el CD

Adjunto a esta memoria se entrega un CD cuyo contenido se describe a continuación. Los recursos software mínimos necesarios para poder ejecutar el sistema pueden verse a continuación en el siguiente listado:

- Memoria del proyecto final de carrera con título “Adaptación de un sensor de proximidad y táctil mediante FPGAs para la interacción humano-robot” en formato *.PDF* cuyo nombre de archivo es “Memoria.pdf”.
- Un resumen del proyecto final de carrera en español con título “Resumen” en formato *.PDF* cuyo nombre de archivo es “Resumen.pdf”.
- Un resumen del proyecto final de carrera en inglés con título “Abstract” en formato *.PDF* cuyo nombre de archivo es “Abstract.pdf”.
- Una colección de los archivos *Matlab* que han sido escritos en el transcurso de este proyecto para ayudar a la consecución del objetivo del “Capítulo 4. Modelado del sistema en *Matlab*” que se encuentran en el directorio raíz dentro de la capeta “matlab”, en formato *.m* y que pasaremos a nombrar a continuación:

  - start.m
  - generate_signal.m
  - amplitude_test_1.m
  - amplitude_test_2.m
  - amplitude_test_2_new.m
  - amplitude_test_3.m
  - amplitude_test_3_new.m
  - dynamic_spectrum.m
  - error.m
  - fft_test.m
  - full_system_capacitance.m
  - full_system_distance.m
  - lot_spectrum.m
  - noise_vs_signal_test.m
  - plot_spectrum.m
  - ploter.m
  - time_optimized_fft.m

Pliego de condiciones

En este apartado se explica en primer lugar el pliego de condiciones técnicas, es decir, la información referente a la instalación y ejecución del código. Posteriormente, en el pliego de condiciones legales, se informa de los acuerdos de licencia del código.

1. Pliego de condiciones técnicas

1.1. Requisitos hardware

Los recursos hardware mínimos para ejecutar el software específico necesario para la realización de éste proyecto pueden verse listado a continuación:

- R.1. Ordenador de sobremesa con sistema operativo de usuario.
- R.2. Robot Industrial KUKA KR16 con Panel de Control[3]*Anexo A*.
- R.3. Pinza Robótica PG70[7][8].
- R.4. 160 sensores tactiles y de proximidad.
- R.5. Cyclone III *FPGA* Development Kit de Altera[32][33][34].
- R.6. Placa Convertidora Analógica-Digital y Generadora de la Señal de Referencia de 10 Canales.*Anexo C*.
- R.7. Placa de Ethernet Intel® 82567LM-3 Gigabit Network Connection (para conectar con el servidor).
- R.8. Placa Realtek PCI GBE Family Controller (para conectar con la *FPGA*).
- R.9. Fuente de Alimentación Laboratory DC Power Supply GW INSTEK GPS-4303.
- R.10. 2 Cables Ethernet Gigabit.

1.2. Requisitos software

Los recursos software mínimos necesarios para poder ejecutar el sistema pueden verse a continuación en el siguiente listado:

- R.11. MATLAB R2014b con SIMULINK para MATLAB R2014b[30].
- R.12. Nios II 9.1 Software Build Tools for Eclipse[31].
- R.13. Quartus II versión 8.0 Build 231 SJ Full Version[40][41][42].

1.3. Ejecución del código

Para ejecutar el sistema se recomienda seguir los pasos que se describen a continuación:

1. Ejecutar **Quartus II**
   a) Abrir **"quartus/filter_91/dtpsc.qpf"**
2. Ejecutar **SOPC Builder**
   a) Abrir **"quartus/filter_91/dtpsc.sopc"**
   b) Ir a **System Generation tab**
   c) **Generate**
3. Ejecutar **Nios II 9.1 Software Build Tools** para **Eclipse**
   a) Si es la primera vez que compilas este proyecto, añade el proyecto a un espacio de trabajo de **Eclipse**
   b) Seleccionar **File** ⇒ **Import**
   c) Seleccionar **General** ⇒ **Existing project into Workspace**
   d) Click **Next**
   e) Click en el botón **Browse** al lado de **Select root directory**
   f) Navegar hasta **"fpga/quartus/filter_91/software"**
   g) Click **OK**
   h) Asegurarse de que todos los proyectos que aparecen en el cuadro **Projects:** están seleccionados. La lista debería contener al menos **filter** y **filter_bsp**
   i) Click **Finish**
4. Click derecho en **filter_bsp** en el explorador de proyecto
   a) Seleccionar **Nios II** ⇒ **Generate BSP**
5. Click derecho en **filter** en el explorador de proyecto
   a) Seleccionar **Build Project**.

   El fichero **"quartus/filter_91/software/filter/filter.elf"** contiene ahora la imagen del firmware.
6. Abrir el terminal de comandos de **Nios II** (**Start** ⇒ **Altera** ⇒ **Nios II EDS 9.1** ⇒ **Nios II Command Shell**)
   a) Ir al directorio **"quartus/filter_91/software/filter"**
   b) Escribir **"make mem_init_install"**

7. Abrir **“quartus/filter_91/triple_speed_ethernet_constraints.sdc”**
   a) Editar la línea “ **set FIFO_CLOCK_FREQUENCY “100 MHz”** ”
   b) Escribir en ella “ **set FIFO_CLOCK_FREQUENCY “100 MHz”** ”
   c) Editar la línea “ **set DEFAULT_SYSTEM_CLOCK_SPEED “66 MHz”** ”
   d) Escribir en ella “ **set DEFAULT_SYSTEM_CLOCK_SPEED “100 MHz”** ”
8. En **Quartus II**, seleccionar **“Tools ⇒ Megawizard PlugIn Manager”**
   a) Seleccionar **“Edit an existing custom megafunction variation”**
   b) Click **Next**
   c) Select **“filter_mc_output_fir.vhd”**
   d) Click **Next**
   e) En la ventana que se abre, seleccionar **Generate**
   f) Click **OK** si aparece la ventana emergente con título *“File Overwrite / Conflict Warning”*
9. En **Quartus II**, seleccionar **Processing ⇒ Compilation**
   El fichero **“quartus/filter_91/filter_time_limited.sof”** contiene ahora la imagen de la FPGA
10. En **Matlab[30]**, necesitamos tener una JDK compatible con la versión de **Matlab[30]** utilizada.
    a) Abrir **“MATLAB/java”** en una ventana de consola
    b) Compilar todos los ficheros **“.java”** con **javac**
    c) Opcionalmente, ejecutar **“javadoc.sh”** para generar la documentación **JavaDoc**
    d) Abrir **Matlab[30]**
    e) Cambiar el directorio actual al subdirectorio del repositorio llamado **“/MATLAB”**
    f) Ejecutar el comando **“javaaddpath(’java’)”**
    g) Construir un **“ThreadedConnection”** ejecutando el comando **“conn = ThreadedConnection(16962, [false true false true false true false true false true false true false true false true], 4096, false 65535)”** para uso **Normal**, o el comando **“conn = ThreadedConnection(16962, [true true true true true true true true true true true true true true true true], 4096, false 65535)”** para el modo **Endeffektor**
    h) Iniciar el **thread** de recepción con el comando **“conn.startReceive()”**
    i) Si la *FPGA* está configurada y con el **firmware** inicializado, el modelo de **Simulink “simulink_test.mdl”** puede ahora ser usado para ver los datos provenientes de los sensores. Para la adquisición de datos para su posterior procesamiento, utilizar los métodos del comando **“ThreadedConnection”**

j) Introducir **set_mode_gui** en la línea de comandos para abrir la configuración **GUI** del controlador del sensor.

2. Pliego de condiciones legales

2.1. Concesión de licencia

Como ya se ha indicado, este proyecto se realizará en el Instituto Tecnológico de Karlsruhe (KIT) y, concretamente, en la división de Automatización de Procesos Inteligentes y Robótica (IPR) del Instituto de Antropomática y Robótica (IAR).

Debido a este hecho, el alumno se ve obligado a cumplir una **cláusula de confidencialidad** en la que se compromete a guardar bajo secreto profesional la mayor parte de la información manejada relacionada con el proyecto. Esta información, por tanto, no podrá ser detallada en la memoria ni en la exposición de este proyecto final de carrera.

Esta cláusula de confidencialidad afecta a la totalidad del software, no pudiendo éste ser copiado ni eliminado y mucho menos impreso o expuesto. Por lo tanto, no se podrá dar ninguna información relevante a los detalles del software del proyecto en la memoria ni en la exposición de ésta.

2.2. Derechos de autor

Tanto el código como la información que se adjunta están protegidos por las leyes de propiedad intelectual que les sean aplicables, así como las disposiciones de los tratados internacionales. Por tanto, el código se considera un producto protegido por derechos de autor. Los ficheros de código empleados en este trabajo y desarrollados por otros autores también están protegidos por las leyes mencionadas.

2.3. Restricciones

No se permite el uso de ingeniería inversa. No se permite la transferencia del código a un tercero ni la copia de éste, incluyendo actualizaciones y material adicional como histogramas, desarrollos matemáticos, etc.

2.4. Garantía

El autor del proyecto lo presenta “AS IS” (tal cual), sin garantía implícita de ningún tipo. No se responsabiliza de los daños que pudieran causar a equipos o personas por

el uso del código o la documentación. El autor no asegura, garantiza, o realiza ninguna declaración sobre el uso y resultados derivados de la utilización del código y/o del resto de la información proporcionada. El código no está exento de errores y no está diseñado para entornos de riesgo que requieran de un funcionamiento a prueba de fallos. El autor rechaza expresamente cualquier garantía explícita e implícita de adecuación del código para actividades de riego.

2.5. Limitación de responsabilidad

En ningún caso el autor, los tutores, la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de las Palmas de Gran Canaria (ULPGC), ni la división de Automatización de Procesos Inteligentes y Robótica (IPR) del Instituto de Antropomática y Robótica (IAR) del Instituto Tecnológico de Karlsruhe (KIT) serán responsables de los perjuicios directos, indirectos, incidentales o consiguientes, gastos, lucro cesante, pérdida de ahorros, interrupción de negocios, pérdida de información comercial o de negocio, o cualquier otra pérdida que resulte del uso o de la incapacidad de usar el código o la documentación. El usuario conoce y acepta este riesgo, así como el resto de cláusulas y restricciones. El autor no reconoce otra garantía que no haya sido indicada anteriormente.

2.6. Otras consideraciones

En el supuesto de que cualquier disposición de esta licencia sea declarada total o parcialmente inválida, las cláusulas afectadas serán modificadas convenientemente de manera que sea ejecutable una vez modificada, permaneciendo el resto de este contrato en vigencia. Este contrato se rige por las leyes de la Unión Europea. El usuario acepta la jurisdicción exclusiva de los tribunales europeos en relación a las diputas derivadas de la presente licencia.

Presupuesto

A continuación detallaremos los recursos necesarios para realizar este proyecto, y las condiciones en las que se llevará a cabo, así como el gasto necesario invertido en mano de obra cualificada.

1. Recursos humanos

Los recursos humanos necesarios para completar este proyecto ha sido contabilizado para la contratación de un único Ingeniero Superior de Telecomunicaciones. Bajo esta premisa, han sido necesarios al menos 235 días con una jornada laboral de 7,5 horas de trabajo, lo que hace un total de 1.762,5 horas trabajadas. Si contabilizamos una media de 22 días laborales al mes, esto hace un total de 10,68 meses trabajados, o lo que es lo mismo, 10 meses y 3 semanas.

Para poder calcular el costo de estas horas trabajadas, hemos consultado la tabla de retribuciones y categorías para la contratación de personal con cargo a proyectos y convenios de investigación donde se presentan las retribuciones en función de la categoría a la que el personal contratado por la ULPGC pertenece y que ha sido publicada en el BOULPGC. En dicha tabla, podemos ver que el ingeniero contratado corresponde a la celda de *Personal Investigador* $\Longrightarrow$ *Técnico en Proyecto* $\Longrightarrow$ *Licenciado, Arquitecto o Ingeniero* ***TCP3*** $\Longrightarrow$ *TC 37,5* lo que indica que el empleado ha dedicado a este proyecto 37,5 horas semanales y que la retribución mensual que le corresponde a un empleado con éstas características por dicho trabajo es de 1.921,64 €.

Con el dato de la retribución mensual que corresponde a un empleado con estas características y el dato, ya obtenido, de las horas semanales dedicadas; podemos obtener el coste en euros de una hora de trabajo, a partir del desarrollo de la ecuación 6.1, de la siguiente manera:

$$Coste_{hora}(€/h) = \frac{Coste_{mensual}(€/mes)}{Horas_{laborales}(h/mes)} \tag{6.1}$$

$$Coste_{hora}(€/h) = \frac{Coste_{mensual}(€/mes)}{Dias_{laborales}(d/mes) \cdot Horas_{laborales}(h/d)} \tag{6.2}$$

$$Coste_{hora}(€/h) = \frac{1{,}921,64(€/mes)}{22(d/mes) \cdot 7,5(h/d)} \tag{6.3}$$

$$Coste_{hora}(€/h) = \frac{1{,}921,64(€/mes)}{165(h/mes)} \tag{6.4}$$

$$Coste_{hora}(€/h) = 11,6463(€/h) \approx 11,65(€/h) \tag{6.5}$$

Como podemos observar en el resultado 6.5 al desarrollar la ecuación 6.1, el coste por hora del proyecto realizado es de 11,65 €. Como ya sabemos, el número de horas finales trabajadas ha sido de 1.762,5 horas por lo que, como puede verse en 6.8, **el costo total de las horas trabajadas en este proyecto es de 20.533,13€**.

$$Coste_{total_{trabajado}}(€) = Coste_{hora}(€/h) \cdot Horas_{trabajadas}(h) \tag{6.6}$$

$$Coste_{total_{trabajado}}(€) = 11,65(€/h) \cdot 1{,}762,5(h) \tag{6.7}$$

$$Coste_{total_{trabajado}}(€) = 20{,}533,125(€) \approx 20{,}533,13(€) \tag{6.8}$$

2. Recursos *hardware*

Para la realización de este proyecto son necesarios los recursos materiales y sistemas *hardware* que se especifican junto a su coste en la *Tabla 6.1*.

Cómo el *hardware* citado es utilizado única y exclusivamente para este proyecto, el coste total que figura al final de la *Tabla 6.1* coincidiría con el coste aplicable al proyecto en cuanto a recursos *hardware* se refiere.

Recurso *Hardware*	**Coste del Recurso (€)**
Ordenador de sobremesa Intel® Core™2 Quad *CPU* Q9400 @ 2.67GHz, 64 bits, 8,00GB *RAM* con Windows 7 Professional	**459,00 €**
Ordenador portátil HP ENVY 15 Notetook PC con Intel® Core™i7-4702MQ @ 2.20GHz, 64 bits, 16,00 GB *RAM* con Windows 10 Home	**999,00 €**
Robot Industrial KUKA KR 16 con Panel de Control[3]*Anexo A*	**9.500,00 €**
Pinza Robótica PG 70[7][8]	**2.780,00 €**
160 sensores tactiles y de proximidad	**88,48 €**
Cyclone III *FPGA* Development Kit de Altera[32][33][34]	**1.070,00 €**
5 Quickfilter Pro QF4A512 Configuration[43][44][45]	**420,00 €**
Placa Convertidora Analógica-Digital y Generadora de la Señal de Referencia de 10 Canales.*Anexo C*	**75,00 €**
Placa de Ethernet Intel® 82567LM-3 Gigabit Network Connection (para conectar con el servidor)	**260,00 €**
Placa Realtek PCI GBE Family Controller (para conectar con la *FPGA*)	**13,95 €**
Acceso a kits de diseño	**535,00 €**
Osciloscopio y Analizador Lógico Tektronix MSO 2013 Mixed Signal, 16 canales, 100 MHz, 1GS/s	**2.855,00 €**
Fuente de Alimentación DC GW INSTEK GPS-4303	**469,48 €**
2 Cables Ethernet Gigabit	**35,00 €**
Coste total	**19.559,91 €**

Tabla 6.1: Coste de adquisición de los recursos *hardware*.

3. Recursos *software*

Para la realización de este proyecto es necesario tener acceso al *software* que se presenta en la *Tabla 6.2* junto a los costes específicos de adquisición anual de cada uno de ellos.

Como el 100 % del coste total de estas herramientas corresponde a un año de soporte, 12 meses; 10 meses y 3 semanas corresponderán a $10,75 \cdot 100/12 = 89,5833\,\% \approx 89,6\,\%$. Por tanto, el coste final de las herramientas software utilizadas durante el tiempo del proyecto será de $Coste_{final} = 89,6\,\% Coste_{total} \Longrightarrow Coste_{final} = 0,896 \cdot 3{,}501,32 \Longrightarrow Coste_{final} = 3{,}137,18272 \approx 3{,}137,18$.

Además, este coste debe ser prorrateado por los 12 usuarios que diariamente tienen acceso a dichas licencias y que, por tanto, también contribuyen con su trabajo a la amortización de dicho pago. Teniendo este hecho en cuenta, el costo final de las

Recurso *Software*	**Precio del Recurso (€)**
MATLAB R2014b con SIMULINK para MATLAB R2014b[30]	**427,83 €**
Quartus II versión 8.0 Build 231 SJ Full Version[40][41][42]	***214,00 €**
*SOPC*Builder 9.1sp2 Build 350[46][47]	**Incluido en ***
Nios II 9.1 Software Build Tools for Eclipse[31]	**Incluido en ***
ModelSim-Altera Edition 6.4a Software [48]	**1.872,50 €**
Paquete ofimático de Microsoft Office 2016 Professional	**344,99 €**
Wireshark versión 1.2.6 (SVN Rev. 31702)	**Software Libre**
TeXnicCenter versión 2.02[49][50]	**Software Libre**
Active Perl versión 5.24.1.2402	**Software Libre**
MiKTeX version 2.9.6200	**Software Libre**
Notepad++ versión 6.7.7	**Software Libre**
Cygwin64	**Software Libre**
JabRef versión 3.8.2	**Software Libre**
Acceso a soporte de herramientas	**642,00 €**
Coste total	**3.501,32 €**

Tabla 6.2: Coste anual aplicado a los recursos *software*.

herramientas software quedaría de la siguiente manera 6.9:

$$Coste_{final_{software}}(€) = \frac{Coste_{final}(€)}{N^{\circ}_{usuarios}} \tag{6.9}$$

$$Coste_{final_{software}}(€) = \frac{3{,}137{,}18(€)}{12} \tag{6.10}$$

$$Coste_{final_{software}}(€) = 261{,}43€ \tag{6.11}$$

4. Coste total del proyecto

El coste total del proyecto se calcula como la suma de los costes directos e indirectos más un 7 % correspondiente al impuesto de IGIC sobre la base imponible total de este y su valor final asciende a **58.725,38 € Cincuenta y ocho mil setecientos veinticinco euros con treinta y ocho céntimos de euro**.

Los costes directos corresponderán a la suma de los costes de los diferentes recursos empleados como son, los recursos humanos, los recursos hardware, los recursos software, el material fungible y los costes de impresión.

Por último, los costes indirectos se estiman como un porcentaje de los costes directos, que en este proyecto es del 35 %.

Todos estos datos pueden verse desglosados a continuación en la *Tabla 6.3*.

Tipo de Recurso	**Coste (€)**
Recursos Humanos	20.533,13 €
Hardware	19.559,91 €
Software	261,43 €
Material fungible	200,00 €
Costes de impresión	100,00 €
Costes Directos	40.654,47 €
Costes Indirectos (35 % de los Costes Directos)	14.229,06 €
Coste del Proyecto antes de impuestos	**54.883,53 €**
IGIC (7 %)	**3.841,85 €**
Coste total del proyecto	**58.725,38 €**

Tabla 6.3: Coste total del proyecto.

Las Palmas de Gran Canaria, a 12 de julio de 2017

Fdo.: Díaz González, Richard