IMPLEMENTACIÓN DE UN ALGORITMO EXPERTO PARA LA MONITORIZACIÓN DE SALUD ESTRUCTURAL SOBRE LA PLATAFORMA ODROID-U3

FABIAN EDUARDO ARIZA ARIZA
OSCAR ANDRES RIBERO RAMOS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2015

IMPLEMENTACIÓN DE UN ALGORITMO EXPERTO PARA LA MONITORIZACIÓN DE SALUD ESTRUCTURAL SOBRE LA PLATAFORMA ODROID-U3

FABIAN EDUARDO ARIZA ARIZA
OSCAR ANDRES RIBERO RAMOS

Trabajo de grado presentado como requisito para optar al título de INGENIERO ELECTRÓNICO

DIRECTOR
RODOLFO VILLAMIZAR MEJÍA
Ingeniero Electrónico y Electricista, Ph.D

CODIRECTOR
OSCAR EDUARDO PEREZ GAMBOA
Ingeniero Electrónico, MS.c ©

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICO-MECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2015

AGRADECIMIENTOS

Primero, quiero agradecer a Dios por darme la salud para culminar este proyecto.

A mis padres, Jorge Alberto Ribero Joya y María Smith Ramos Reyes, por brindarme el apoyo, amor, la sabiduría y confianza necesarios para seguir adelante en cada paso de este camino recorrido. A mis hermanos, Sergio Javier Ribero Ramos y Adriana Marcela Ribero, por su apoyo y compañía incondicional. A mi familia, en general, por estar siempre a mi lado.

A mi novia, Lilian Carolina Acevedo Pineda, por brindarme su amor, compañía y paciencia para superar tantos momentos difíciles.

A María Asened Serna, por su cariño y cuidado durante estos años.

A Fabián Eduardo Ariza Ariza, mi compañero de proyecto, carrera y mejor amigo, con quien compartí no solamente días de alegrías, sino también de tristezas. A todos mis amigos, que me ayudaron y acompañaron en este largo proceso.

Oscar Andres Ribero Ramos

AGRADECIMIENTO

Quiero hacer mención de Dios en primera instancia por todo lo brindado, por ser guía y camino, luz y vida, por su sabiduría. “Esfuérzate y se valiente, no temas ni desmayes, porque Jehova tu Dios estará contigo a donde quiera que vayas”… palabras que han sido especiales en mi vida.

Dario Alfonso Ariza y Luz Yaneth Ariza “Morris”, mis papás, gracias por ese apoyo persistente, por sus palabras y esfuerzo, los valores inculcados, la entereza, especialmente por su amor para formarme como una persona de bien. A mis hermanos, Cesar y Juancho, porque son apoyo diario para seguir adelante. No puedo dejar de mencionar a mi tía Mayden porque ha sido mi mamá en esta ciudad que me adopto. A mis primos y familia en general muchas gracias, en todo momento me han motivado y alentado a seguir adelante.

Termino por hacer alusión a mis amigos, porque su compañía, esfuerzo y entusiasmo han sido importantes en este proceso llamado vida. A Laura Andrea Marín Téllez, amiga y compañera, sinónimo de alegría y denuedo, sus memorias permanecen y sus sonrisas también, a pesar de su corta estadía y poco tiempo compartido.

Fabián Eduardo Ariza Ariza

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: IMPLEMENTACION DE UN ALGORITMO EXPERTO PARA LA MONITORIZACION DE SALUD ESTRUCTURAL SOBRE LA PLATAFORMA ODROID-U3[*]

AUTORES FABIAN EDUARDO ARIZA ARIZA
OSCAR ANDRES RIBERO RAMOS[**]

PALABRAS CLAVE: Validación, sistemas embebidos, detección de defectos en estructuras, tiempos de ejecución, piezoeléctricos.

DESCRIPCION

En el presente trabajo de investigación se documentan los resultados de implementación de un algoritmo estadístico basado en Análisis de Componentes Principales (PCA), en su etapa de validación, sobre las plataformas embebidas BeagleBone Black y Odroid-U3, para la detección de fallas en estructuras. Este documento tiene como objetivo mostrar los resultados de desempeño del algoritmo de detección de fallas cuando es ejecutado en dichas plataformas. Para ello, se evaluará la factibilidad de implementación utilizando datos suministrados por el grupo CODALAB de la Universitat Politècnica de Catalunya (UPC, Barcelona) y el grupo CEMOS de la Universidad Industrial de Santander. Estos datos corresponden mediciones piezoeléctricas de dos estructuras básicas que son excitadas por medio de ondas guiadas: el álabe de una turbina de un avión y una tubería de acero-carbono.

Los datos fueron procesados en plataformas embebidas que aprovechan el entorno computacional Linux y el lenguaje de programación Python. Las características que motivaron el uso de dicho lenguaje son: licencia de uso libre (GPL), ejecución multiplataforma, bajo requerimiento de almacenamiento en disco (95 MB aproximadamente), alto desempeño en la ejecución de algoritmos y amplio soporte en línea. El análisis del desempeño de las plataformas embebidas es evidenciado y comparado por medio de los tiempos de procesamiento y recursos de memoria usados, los cuales definen la viabilidad del uso de estos sistemas para los requerimientos de la aplicación de estudio en la Monitorización de Salud Estructural (SHM- por sus siglas en ingles).

[*] Trabajo de grado

[**] Facultad de Ingenierías Fisico-Mecánicas. Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones. Director. Rodolfo Villamizar Mejía. Codirector. Oscar Eduardo Perez Gamboa

ABSTRACT

TITLE: IMPLEMENTATION OF AN EXPERT ALGORITHM FOR STRUCTURAL HEALTH MONITORING ON THE ODROID-U3 PLATFORM [*]

AUTHORS FABIAN EDUARDO ARIZA ARIZA
OSCAR ANDRES RIBERO RAMOS[**]

KEYWORDS: Validation, embedded systems, structural damages detection, runtimes, piezoelectric transducers.

DESCRIPTION

In this research project, the results of implementing a statistic algorithm based on Principal Components Analysis (PCA), in its validation stage, on the embedded platforms BeagleBone Black and Odroid-U3 to detect structural damages, are compiled. The main objective is to show the resulting performance of the Damage Detection Algorithm when it is run in the mentioned platforms. In order to do it, the feasibility of the implementation, using the data supplied by the CODALAB group from the Universitat Politècnica de Catalunya (UPC, Barcelona) and the CEMOS group from Universidad Industrial de Santander, will be evaluated. These data correspond to piezoelectric measurements from two test structures that are excited through guided waves: the blade turbine of a plane and a carbon steel pipe.

The data were processed in embedded platforms that take advantage of the computational environment Linux and the programming language Python. The features that motivated the use of such language are: General Public License (GPL), multi-platform running, minimum storage requirement on the hard disk (95 MB approximately), high performance while running algorithm and sufficient online support. The analysis of the performance of the embedded platforms can be seen and compared through the processing times and the memory resources used, which define the possibility of using these systems to fulfill the requirements of the application in the Structural Health Monitoring (SHM).

[*] Degree work

[**] Faculty of Physical-Mechanical Engineering. School of Electrical and Telecommunications Engineering, Electronics. Director. Villamizar Rodolfo Mejia. Co. Oscar Eduardo Perez Gamboa

INTRODUCCIÓN

La Monitorización de Salud Estructural (SHM) permite evaluar el estado actual de estructuras civiles, mecánicas y aeroespaciales con el objetivo de determinar, localizar y cuantificar fallas estructurales [2]. La identificación de defectos en estructuras se basa en el análisis de la respuesta dinámica estructural. En este sentido, cualquier variación de las propiedades físicas de la estructura está representada en cambios de la respuesta dinámica estructural. Uno de los métodos propuestos en la literatura [3], hace uso de excitaciones mediante ondas guiadas, según lo cual, cualquier daño cambiará la forma de propagación de la onda a través de una estructura. Por ejemplo, en [4] se aplica ondas tipo Lamb en la detección y caracterización de defectos en placas de aluminio y la inspección de soldaduras para análisis de collages de integridad.

Para estudiar la respuesta de onda de propagación se requieren de modelos que faciliten la evaluación de los datos de la respuesta dinámica. Dichos modelos se basan en leyes físicas o corresponden a aproximaciones basadas en mediciones (Data - driven models), estas últimas con amplio campo de estudio para el desarrollo de métodos, tales como: función de base y regresiones Gaussianas, redes neuronales, clasificadores lineales o análisis de componentes principales (PCA-por sus siglas en inglés).

Recientes trabajos ([3], [5]) enfatizan en el estudio de técnicas para el análisis de datos cuantificados por medio de piezoeléctricos, así como de métodos de paradigma de reconocimiento de patrones estadísticos [6] [7]. Consecuente a esto, se ha demostrado la eficiencia del análisis de componentes principales (PCA) para la detección de defectos debido a su capacidad para reducción de datos y revelar patrones o tendencias ocultas y simplificadas.

En instancia a lo descrito, este documento presenta los resultados de implementación de un algoritmo estadístico basado en análisis de componentes principales, cuando es ejecutado sobre dos tarjetas de desarrollo (BeagleBone Black y Odroid-U3) con el fin de detectar fallas en estructuras (Nivel 1 de SHM). Dichos resultados se sintetizan en cuanto al desempeño de los sistemas embebidos para validar las estructuras de estudio, así como en *el tiempo* y *la memoria*[1] requerida para el procesamiento de datos y ejecución del algoritmo desarrollado por medio del software libre Python.

El estudio hace parte de la realización de un proyecto de maestría financiado por COLCIENCIAS[2]: "Monitorización y detección de defectos en estructuras usando algoritmos expertos embebidos"; el cual busca realizar la detección de daños en línea para estructuras tipo tubería.

[1] Tanto el tiempo de procesamiento como la memoria utilizada por los sistemas embebidos, han sido adoptados como Parámetros de rendimiento.
[2] Departamento Administrativo de Ciencias, Tecnología e Innovación COLCIENCIAS.

1. OBJETIVOS

1.1. OBJETIVO GENERAL

Implementar en una plataforma Hardware Odroid-U3 y BeagleBone Black un algoritmo de detección de defectos en estructuras, basado en modelado estadístico mediante Análisis de Componentes Principales.

1.2. OBJETIVOS ESPECÍFICOS

- Analizar e interpretar el algoritmo basado en PCA para detección de defectos en estructuras, realizado en el marco del proyecto MONITORIZACIÓN Y DETECCIÓN DE DEFECTOS EN ESTRUCTURAS USANDO ALGORITMOS EXPERTOS EMBEBIDOS.

- Adaptar en Python™ el algoritmo basado en PCA para detección de defectos en estructuras.

- Validar la implementación del algoritmo usando datos experimentales suministrados por el grupo CODALAB de la UPC Barcelona y pruebas realizadas en los laboratorios de la UIS, sede Guatiguará.

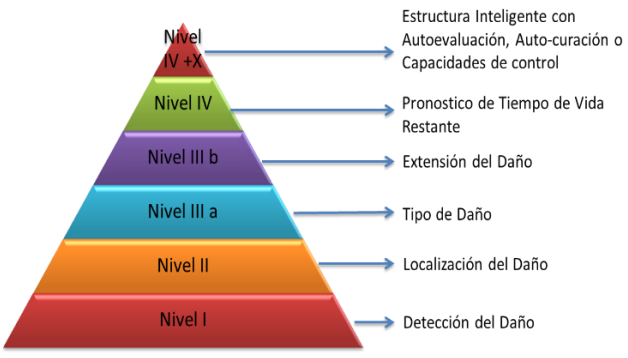
- Evaluar el rendimiento y desempeño (tiempo de ejecución, uso de memoria) de la plataforma ODROID-U3 y BeagleBone Black para la implementación y ejecución del algoritmo PCA en la detección de defectos en estructuras.

2. MARCO TEÓRICO

En las últimas décadas, se ha presentado un gran interés en el reconocimiento de la detección de fallas en estructuras ([8] [9] [10]) y su capacidad para hacer dicha labor de manera remota y con la menor intervención humana; todos estos esfuerzos han hecho que la investigación de la Monitorización de salud estructural (SHM) se oriente en obtener el diseño de estructuras inteligentes [11], sensores, algoritmos para la monitorización de estructuras en tiempo real y detección de fallas en etapas tempranas [3].

La revisión bibliográfica define las técnicas según los cuales puede ser clasificado SHM [12]. Rytter [13] estableció cuatro niveles de técnicas de acuerdo a la información proporcionada al usuario (*Figura 1*). Bajo esta distinción, el nivel 1 determina si el daño ha ocurrido, mientras que el nivel 4 también establece la existencia del daño, la ubicación y severidad de este, y la vida restante de la estructura.

Figura 1 Niveles de SHM



Fuente: Adaptado de [13]

El análisis de SHM puede ser efectuado mediante el procesamiento de señales de vibración e inspección visual y directa a los elementos de la estructura, así como el uso de transductores piezoeléctricos, siendo estos elementos de diagnóstico. Algunos trabajos como los realizados por Yuan et al en [14], argumentan la monitorización de salud estructural y señales vibracionales, basado en el diagnóstico de ondas Lamb como técnica de gran utilidad para la obtención de datos de análisis. A su vez, Lim et al en [15], basa el diagnostico estructural con el uso de piezo-sensores mediante el uso de señales vibracionales.

Para el contexto del presente documento, la metodología de detección de fallas en estructuras [Nivel 1 de SHM] requiere de técnicas para la evaluación de la respuesta dinámica cuando la cantidad de datos adquiridos es muy densa, como lo es análisis de componentes principales (PCA) [3].

Muchas investigaciones demuestran la importancia de PCA para este tipo de aplicaciones como [3] y [6] que aplican análisis de componentes principales para la reducción de datos y hallan tendencias características en el modelado y proyección de los datos de la estructura, en [16] se utiliza PCA como herramienta de visualización y agrupamiento de datos de emisiones acústicas de la caja de la viga de un puente para el seguimiento del desarrollo de grietas, en [17] implementan PCA con la transformada wavelet para la detección de grietas por fatiga y en [3] se define PCA como un modelo basado en datos estadísticos para la condición de funcionamiento normal en estructuras, de modo que cualquier condición de fallo provoca desviación de este modelo que puede ser detectado utilizando índices estadísticos como T2 y Q-estadístico [18].

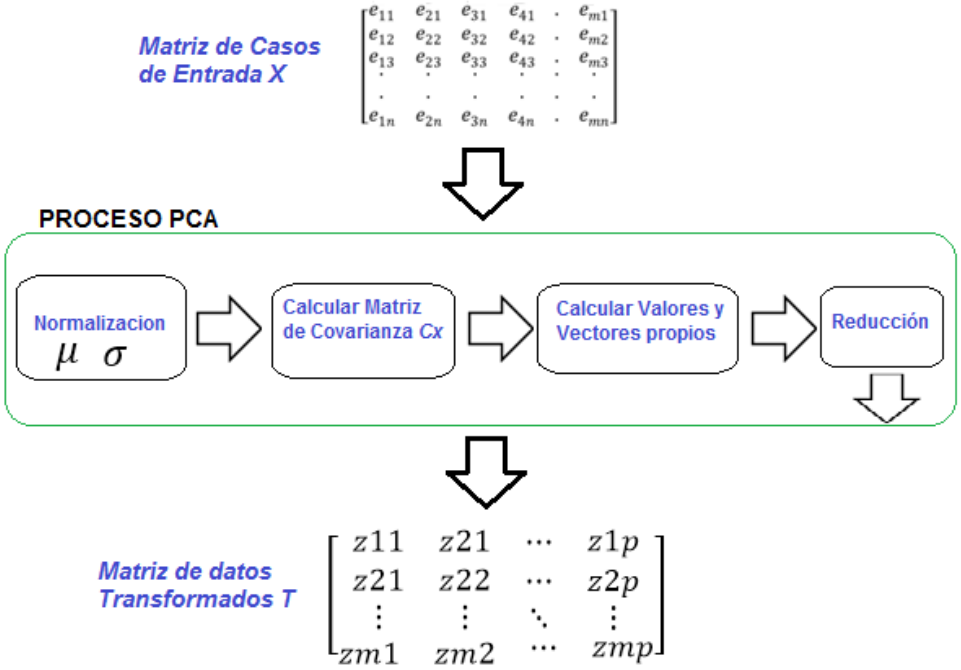
2.1. ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

Análisis de Componentes Principales PCA (por sus siglas en inglés) es una técnica de análisis multivariable simple y no paramétrica para la compresión de datos y la extracción de información, que encuentra combinaciones de variables o factores que describen las principales tendencias de un conjunto de datos confusos [3] [19].

PCA no distingue la naturaleza de los datos para su procesamiento, razón por la cual, su metodología está abierta a cualquier matriz de entrada con valores relacionados a un comportamiento y cuyos componentes de mayor aporte quieran ser develados.

2.1.1. Metodología para el Análisis de Componentes Principales PCA (Clásico). El procedimiento de PCA para la obtención de los valores característicos simplificados y revelar patrones ocultos de un grupo de datos, es posible resumirse bajo el esquema mostrado en la Figura 2. Cada uno de estos recuadros se describe a continuación.

Figura 2 Esquemático del proceso de PCA



Nota: 1. De la Matriz de Casos de Entrada *X* en la Figura 2, e_{ij} representa un dato del sensado de la estructura de estudio (con *i = 1,2,…,m*; y *j = 1,2,…,n*).

2. De la Matriz de datos Transformados *T* en la Figura 2, z_{ij} representa un dato proyectado sobre el plano del modelo (con *i = 1,2,…,m*; y *j = 1,2,…,p*).

- *Normalización:* El grupo de datos adquiridos tienen magnitudes y escalas diferentes, por lo cual se debe normalizar la matriz de datos original X usando la media de cada variable y su desviación estándar a través de alguno de los métodos de escalamiento, entre estos, el AutoScaling, Continuous Scaling y Group Scaling[3].

- *Calcular la matriz de covarianza* C_x que mide el grado de relación lineal entre todos los posibles pares de variables de la serie de datos [19].

[3] Este método de escalamiento se profundiza en el Capítulo 2.2

- *Determinar los vectores y valores propios de la matriz de covarianza* en el que a cada vector propio le corresponde un valor propio, de modo que:

$$C_x\tilde{P} = \tilde{P}\Lambda \qquad \text{(Ecuación 1)}$$

Donde las columnas de $\tilde{P}$ son los vectores propios de la matriz de covarianza y Λ la matriz diagonal con los valores propios. Los vectores propios con los más altos valores propios asociados son los *componentes principales* del grupo de datos, es decir los valores más representativos, de modo que las columnas de $\tilde{P}$ deben de organizarse de mayor a menor de acuerdo a los valores propios asociados [19].

Cuando las dimensiones de la matriz de datos X son elevadas, determinar los valores y vectores propios se vuelve una tarea cada vez más extenuante para el procesamiento, por lo que se es posible adoptar métodos iterativos para la descomposición de un número determinado de componentes principales, como es el caso de Nipals (Non-Linear Iterative Projections by Alternating Least-Squares), Snapshot POD (Proper-orthogonal-decomposition) o SVD (Singular Values Descomposition).

- *Reducir la nueva matriz* $\tilde{P}$ eliminando los componentes de menos significancia, esto con el fin de obtener una nueva matriz P (organizada y reducida) que corresponde a la matriz de "transformación de espacios" o modelo de PCA [19].

- Por último, la matriz de datos transformados T o matriz de resultados [18] representa la proyección de los datos originales sobre la dirección de los componentes principales seleccionados, es decir sobre el nuevo espacio generado, de modo que:

$$T = XP \qquad \text{(Ecuación 2)}$$

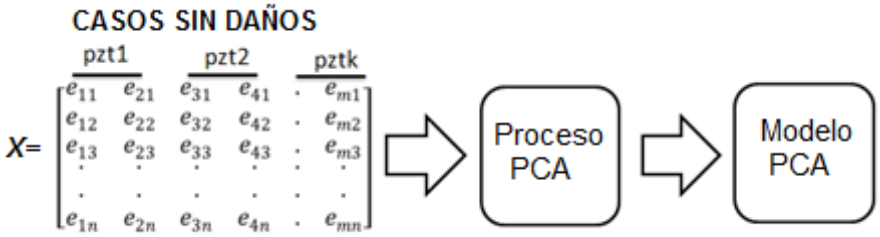
En conclusión T es la matriz de resultados tras la proyección de X en **el modelo estadístico *P***, el cual está representado como una combinación lineal de los vectores propios.

2.2. ANÁLISIS DE COMPONENTES PRINCIPALES PCA PARA LA MONITORIZACION DE SALUD ESTRUCTURAL

Básicamente, la aplicación de PCA para SHM requiere separarse en dos etapas para poder determinar la existencia de daños a partir de las matrices de procesamiento adquiridas: **modelado** de la estructura de estudio, y **validación** de la estructura de estudio.

2.2.1. Modelado de la estructura de estudio. Verificar la existencia de fallas en una estructura de estudio no es posible realizarse, si previamente no se tiene un modelo base sobre el cual se contrastan o proyectan los posibles casos de daños. Esto quiere decir que el modelo de la estructura es aquel en el que no existen daños, en otras palabras, es aquel creado a partir de los casos sin fallos (Figura 3).

Figura 3 Modelado de la matriz de casos sin daños



Fuente: Adaptado de [20]

Nota: *pzt1, pzt2,…, pztk* son los piezoeléctricos usados para sensar la estructura, de modo que e_{ij} es un dato registrado por uno de los piezoeléctricos.

La figura anterior, muestra en general la metodología de modelado PCA organizada en diferentes bloques, los cuales poseen importantes distinciones que son descritas como sigue.

- **Matriz de Casos sin Daños**

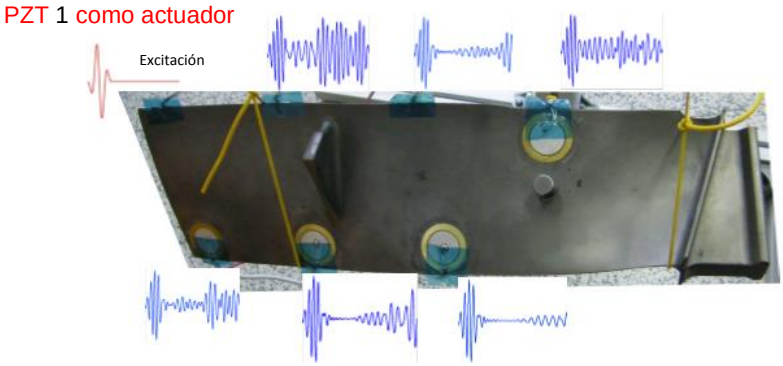
Como se describió anteriormente, PCA no distingue la naturaleza de los datos de entrada, es decir, la forma en cómo estos vienen organizados dentro de la matriz. Sin embargo, realizar el modelo de la estructura de estudio requiere que la matriz de casos de entrada esté organizada de acuerdo al muestreo de los piezo-sensores usados durante los experimentos.

Trabajos como [3], [5], [21] y [22] justifican el uso de piezoeléctricos para sensar las variaciones de la respuesta dinámica de estructuras cuando por estas son emitidas ondas guiadas como las Rayleigh o las Lamb. Dichos piezoeléctricos pueden ser excitados a través de ondas Burst, que aprovechan las propiedades de propagación longitudinal y sirven como señales de excitación de banda estrecha para ondas Lamb, como lo expresa Yuan et al [14]. Conforme a esto, X es una matriz debidamente estructurada para su modelado.

A partir de [3], uno de los N sensores (actuador) puestos sobre una estructura, emite una señal vibracional la cual es captada por los demás (Ver *Figura 4*). Luego, otro sensor hace las veces de actuador mientras que los demás registran en osciloscopio la onda vibracional captada; el proceso se repite para cada uno de los N sensores. De este modo la estructura es analizada en los diversos puntos

que la constituyen para el desarrollo del modelo. A esto se le conoce como Piezo-diagnosis.

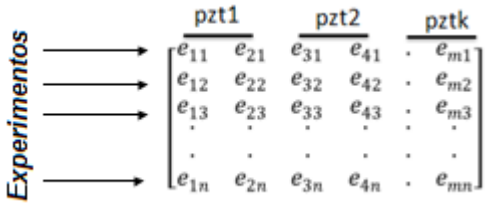
Figura 4 Estructura con piezoeléctricos



Fuente: Adaptado de [3]

En resumen, cada experimento consiste en excitar uno de los N transductores piezoeléctricos (PZT's) y recibir la señal en los otros N-1, de modo que dicho muestreo se organiza en la matriz de casos de acuerdo a lo sensado por cada uno de los PZT´s utilizados (*Figura 5*). Cabe resaltar que todos los piezoeléctricos sensan y actúan con el mismo número de muestras.

Figura 5 Esquema de organización: matriz de casos sin daños



Adaptado de [20]

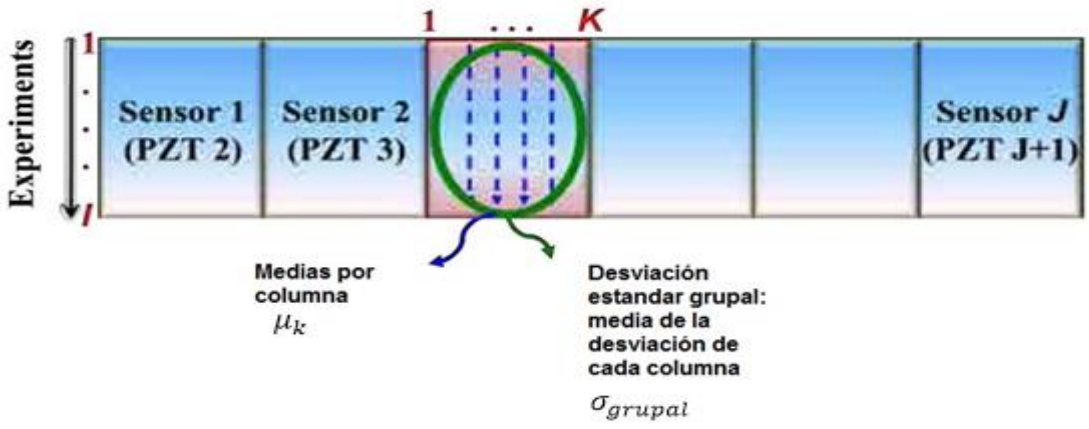
Con lo anterior expuesto, es posible obtener la matriz de casos sin daños para el procesamiento del modelo, constituida con n experimentos en las filas por m muestras por sensor en las columnas.

- **Proceso PCA para SHM**

El procedimiento de PCA inicia en la normalización de los datos de entrada. Dichos datos requieren métodos de escalamientos a través del cálculo estadístico de las medias y desviaciones estándar. Para esto, existen técnicas como el AutoScaling, Continuous Scaling y Group Scaling.

En el desarrollo del presente trabajo de investigación se adoptó **Group Scaling** como técnica de escalamiento, debido a que es de uso común cuando los datos o matriz de datos constan de variables (columnas) distribuidas en varios bloques de igual tamaño, lo que corresponde a la acción de cada piezoeléctrico en la matriz de casos. Este método de pre-procesamiento se lleva a cabo mediante el fraccionamiento de las variables en un número de bloques de igual tamaño y el escalamiento de cada uno por la media de sus desviaciones estándar. La desviación estándar de cada variable, en un bloque, es calculada y la media de esas desviaciones estándar es usada para escalar todas las columnas del bloque (*Figura 6*).

Figura 6 Escalamiento Group Scaling



Fuente: [19]

De modo que:

$$\frac{e_{pk}-\mu_k}{\sigma_{grupal}} = \widehat{e_{pk}} \quad \text{(Ecuación 3)}$$

Es un dato normalizado de la matriz de entrada.

Una vez las variables hayan sido estandarizadas, varios algoritmos para realizar PCA pueden ser usados: NIPALS, POD o SVD [23]. Determinar todas las componentes principales de la matriz de entrada no es práctico, en el sentido de que solo un pequeño número de ellas pueden definir el modelo completamente. En la literatura, NIPALS es el algoritmo de más uso para calcular las principales componentes [23]. Esta técnica es eficiente si únicamente son pocas las componentes principales de PCA requeridas en comparación al tamaño de la matriz [24], ya que dichas componentes son calculadas paso a paso, es decir, una componente al tiempo [25].

Caracterizar una estructura a partir de piezoeléctricos necesita de un significativo número de muestras por sensor. Esto se ve reflejado en la dimensión de la matriz de casos de entrada de la cual solo interesa obtener las principales componentes de mayor aporte de energía dentro del grupo de datos. Lo anterior surge como justificación del uso de NIPALS para la descomposición de valores y reducción de carga computacional en el cálculo de la matriz de datos transformados T, los elementos de la matriz de vectores propios P y los valores propios asociados Λ, conforme al número de componentes seleccionadas para el modelo.

Básicamente, el método de NIPALS empieza cuando una columna de la matriz X es seleccionada como la primera componente principal [23]. Así, las iteraciones comienzan utilizando el método de "mínimos cuadrados alternados" y el correspondiente vector propio de dicha componente se calcula utilizando el método de "regresión". El siguiente paso elimina la varianza asociada a esta

componente y se repite el proceso para la obtener la siguiente componente principal.

Finalmente, para términos de SHM, la **etapa de modelado** se obtiene a partir del análisis de la matriz de mediciones piezoeléctricas correspondiente a la respuesta dinámica de la estructura sin daños X y concluye en el cálculo de T, P y Λ como elementos de salida de NIPALS para la creación de dicho modelo.

2.2.2. Validación de la estructura de estudio. Una nueva matriz de casos con daños D es creada a partir de la estructura de estudio y organizada de forma similar a como se explicó en la matriz de casos sin daños. Grietas, fisuras, masas agregadas o perforaciones en una estructura pueden ser considerados como daños (aunque en la industria se consideran muchos más), de modo que la respuesta dinámica se ve afectada y el proceso de validación podrá concluir sobre dichos daños.

La organización de dicha matriz se resume en la *Figura 7* y se define bajo las siguientes características:

Figura 7 Constitución de la matriz de daños D

$$\begin{array}{c} \\ \text{Daño1} \\ \\ \text{Daño2} \\ \\ \text{DañoF} \end{array} \begin{array}{c} \begin{array}{cccccc} \overbrace{}^{\text{PZT1}} & & \overbrace{}^{\text{PZT2}} & & & \overbrace{}^{\text{PZTK}} \end{array} \\ \begin{bmatrix} d11 & d12 & d13 & d14 & \cdots & d1r \\ d21 & d22 & d23 & d24 & \cdots & d2r \\ d31 & d32 & d33 & d34 & \cdots & d3r \\ d41 & d42 & d43 & d44 & \ddots & d4r \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ dt1 & dt2 & dt3 & dt4 & \cdots & dtr \end{bmatrix} \end{array} = D$$

- Las filas de D corresponden a los experimentos realizados por daño, y las columnas, a las muestras por transductor piezoeléctrico PZT (Piezo-diagnosis).

- La correcta caracterización de cada daño requiere de un número significativo de experimentos, en este sentido, el número de filas de D aumenta.
- Todos los daños poseen el mismo número de experimentos.

En específico, esta etapa consiste en proyectar los elementos propios de la nueva matriz de casos con daños D sobre el plano creado por las principales componentes en la etapa de modelado (*Ecuación 4*). La diferencia entre los elementos proyectados sobre el plano y los elementos del modelo sin daños, permitirá concluir la existencia o no de fallas en la estructura de estudio.

$$T' = DP \qquad \text{(Ecuación 4)}$$

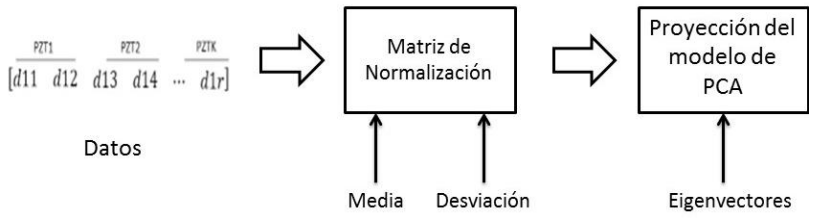
Con: P Matriz de Transformación de espacios

D Matriz de Daños

T' Nueva matriz de datos transformados

Los elementos de salida de la *etapa de modelado* son utilizados en la *etapa de validación* para poder realizar la proyección de los datos de D. Estos elementos corresponden a *las medias* de los datos de la matriz de casos sin daños, las *desviaciones estándar* y los *valores y vectores propios*. La *Figura 8* muestra la incidencia de cada uno de estos elementos dentro del proceso de validación de la estructura de estudio.

Figura 8 Esquemático para validación de datos de estructura



Fuente: Adaptado de [20]

Sin embargo, la estimación de daños presentes en la estructura posee mayor diferenciación con el uso de **índices de daños**. De modo tal que a continuación se hace una reseña de estos.

- **Índices para detección de daños en estructuras basados en PCA.**

En ocasiones, obtener el modelo PCA de una estructura no es suficiente para poder discernir la existencia o no de algún daño. En este sentido, los índices de daño son usados como medidas estadísticas de soporte para hacer un diagnóstico adecuado de la estructura [19].

Existen varios tipos de índices de daño [26] como $\emptyset - index$ e $I^2 - index$. Pero los más usados por su capacidad en la evaluación de fallas son Q-estadístico y T^2 [3].

Q-estadístico es una medida de la diferencia entre una muestra de la matriz residual de datos, producto de la proyección de vuelta al espacio de *X*, y la proyección en el sub-espacio residual.

$$Q = X^T(I - PP^T)X \qquad \text{(Ecuación 5)}$$

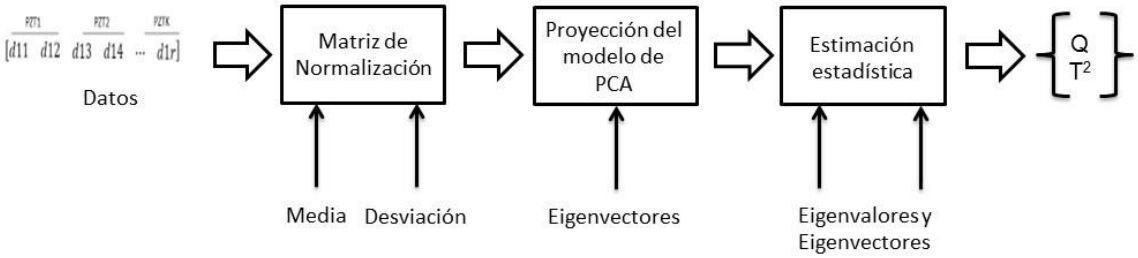
T^2 es una medida de cada muestra de la matriz de datos transformados *T* dentro del modelo PCA.

$$T^2 = X^T(P\Lambda^{-1}P^T)X \qquad \text{(Ecuación 6)}$$

Donde *X* representa la medida de todas las variables en una sola muestra (una fila de la matriz de datos) y Λ es una matriz diagonal cuyos términos son los valores propios de la matriz de covarianza [3].

El esquemático para el proceso de validación se complementa como se muestra en la *Figura 9*.

Figura 9 Esquemático completo para el proceso de validación



2.3. PCA EN SISTEMAS EMBEBIDOS

Los sistemas embebidos son una ventaja para el procesamiento de datos en aplicaciones de baja potencia dado a su reducido tamaño y su capacidad para el desarrollo de funciones computacionales específicas. Mantienen una serie de elementos esenciales para su funcionamiento como lo son una memoria RAM, periféricos de entrada y salida y un microprocesador (CPU), que generalmente es de arquitectura ARM (Advanced RISC Machine) [27].

El interés en este tipo de sistemas, en cuanto a procesos para SHM [9], ha venido siendo objeto de estudio. Bennouna at al en [28], presenta un procedimiento de diagnóstico basado en el uso de la transformada Wavelet con el fin de mejorar la fiabilidad de los sistemas integrados sometidos a vibraciones, dicho procedimiento fue implementado y verificado en tiempo real en la plataforma dSPACE. De igual forma, en [29] utilizan Ruido Multiescalado para Ajustar Resonancia Estocástica (MSTSR, por sus siglas en inglés) para el mejoramiento en el diagnóstico de fallas en los rodamientos y caja de cambios de un tren. El desarrollo del algoritmo se hizo a través de una plataforma embebida por su reducido consumo de memoria y eficiencia en la ejecución del proceso.

Sin embargo, recientes trabajos como [3] y [19], proponen PCA para el estudio de los datos correspondientes a la respuesta dinámica de una estructura. Esta

técnica surge como herramienta de análisis para la descomposición de valores característicos de una serie de datos y su implementación en sistemas embebidos no es algo ajeno. En [30] emplean PCA incremental bidireccional-bidimensional $(I(2D)^2PCA)$ para demandar menos carga computacional de un sistema embebido para el reconocimiento de rostros, de este modo los datos de entrenamiento no requieren estar almacenados en la plataforma y el procesamiento es llevado a cabo en tiempo real. Por su parte [31] investiga el uso de hardware basado en FPGA para aplicaciones de cálculo intensivo en computación portátil y embebida a través del uso de PCA.

2.4. SISTEMAS EMBEBIDOS PARA IMPLEMENTACIÓN DEL ALGORITMO EXPERTO

Gracias a los estudios realizados a partir del uso de sistemas embebidos, se decidió emplear las tarjetas BeagleBone Black y Odroid-U3 para ejecutar el algoritmo experto de monitorización de salud estructural conforme a los objetivos para el presente trabajo de investigación, ya que sus características de hardware y software priman sobre otras tarjetas que han sido usadas para el desarrollo de PCA o SHM. Dichas características se resumen en:

Tabla 1 Especificaciones técnicas de los sistemas embebidos empleados

Especificaciones		
	BeagleBone Black	**Odroid-U3**
CPU	Sitara AM3359AZCZ100 1GHz, 2000 MIPS	1.7GHz Exynos4412 Prime Cortex-A9 Quad-core processor with PoP (Package on Package) 2Gbyte LPDDR2 880Mega Data Rate
RAM	498 [MB]	2072 [MB]
Onboard Flash	2GB, 8bit Embedded MMC	8Gb, eMMC
Power Source	miniUSB USB or DC Jack. 5VDC External Via Expansion Header.	5VDC/2A

Especificaciones		
	BeagleBone Black	**Odroid-U3**
USB 2.0 Host	2 × USB A Host, 1 × ADB/Mass storage (Micro USB)	3 x USB 2.0, 1 x Micro USB
Serial Port	UART0 3.3V TTL Header.	UART 1.8 V
Ethernet	10/100, RJ45	10/100, RJ45
Video Out	16b HDMI	HDMI (480p/720p/1080p)
GPIO	69	5

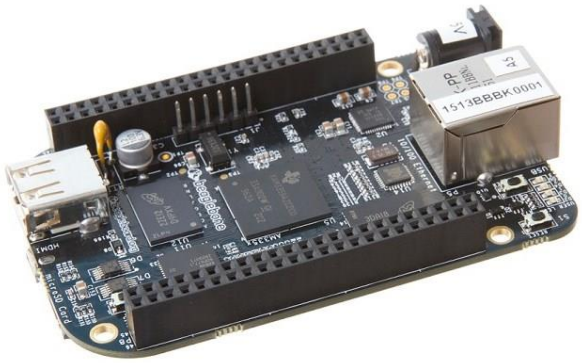
- **Odroid-U3**

Odroid-U3 es un sistema embebido que puede ejecutar los últimos sistemas operativos Ubuntu 13.10 y Jellybean Android para la programación, aprendizaje, juegos, centro de medios de comunicación, servidor web, hardware de plataforma IO y muchas otras aplicaciones. Tiene un potente procesador de cuatro núcleos con 1.7 GHz, de bajo costo, características de eficiencia energética y biblioteca masiva de software [32]. Este tipo de tarjeta ha sido utilizada para aplicaciones en bioingeniería como el rastreo del movimiento del ojo en tiempo real [33] y aplicaciones comerciales de cómputo de alto rendimiento.

- **BeagleBone Black**

BeagleBone Black es un sistema embebido con un procesador de 1 GHz ARM Cortex-A8 que puede ejecutar Linux y Android [34]. Una memoria AM3359 de

hasta 512 MB de memoria, puertos USB 2.0, conector Ethernet para creación de redes, suficientes pines, interfaces de red y puertos periféricos, lo que la hace una gran plataforma de desarrollo para aquellas situaciones en donde se necesita una pantalla o buen poder procesamiento [35]. Han sido utilizadas en procesos como sensor de termocuplas, control de servomotores y motores paso a paso en la industria [36].



Además de las características que tienen las tarjetas descritas, uno de los puntos por el cual se escogieron, se debe a que ofrecen un sistema operativo real, como Linux y Android, lo que representa una gran cantidad de programas que pueden ser usados, así como lenguajes de programación fáciles y flexibles, que a diferencia de lenguajes embebidos como para FPGA, tarjetas Arduino o PLC's, están atadas a su propia terminología de programación.

El uso de este tipo de plataformas embebidas, simplifica el número de componentes físicas que constituyen un modelo de investigación, para el caso, SHM en su primera etapa. Las facilidades que se manifiestan por utilizar sistemas de desarrollo especifico se traducen en practicidad, rendimiento y eficiencia en el uso diversos lenguajes de programación.

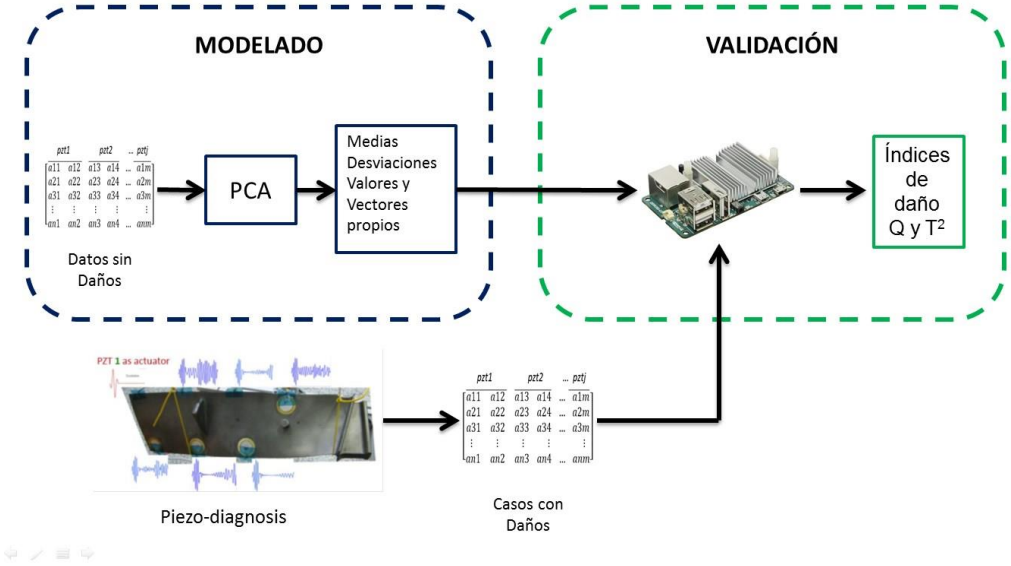
3. METODOLOGÍA PARA EL DESARROLLO DEL TRABAJO

Como se explicó en la sección 2.2.2, PCA para SHM compone de dos fases generales (Modelado y Validación). La Figura 10 es un esquema general del proceso para detección de fallas en estructuras; de éste se recalca la acción de Modelado por medio de la piezo-diagnosis[4] de la estructura sin daños (en modo offline), a fin de generar el modelo con PCA. Sobre este modelo creado serán proyectados los experimentos que representen una variación inusual del modelo o Casos con Daños, lo anterior es la base para hacer el proceso de Validación que culmina en el cálculo de Índices de daño como elementos de estimación ante la existencia de alguna falla.

El presente trabajo de grado, se centra en la implementación offline de la etapa de **validación** en los sistemas embebidos Odroid-U3 y BeagleBone Black, por medio del uso de un algoritmo experto. Esto indica que los elementos de salida de la etapa de Modelado son previamente entregados por grupos de investigación (CEMOS y CODALAB) así como las matrices de Casos con Daños que provienen del análisis de diferentes estructuras.

[4] Acción de sensar o actuar una estructura de estudio por medio de Piezoeléctricos, se utiliza tanto para Modelar la Estructura como para Validarla.

Figura 10 Metodología general de SHM en su primer nivel (Detección de fallas) basado en PCA

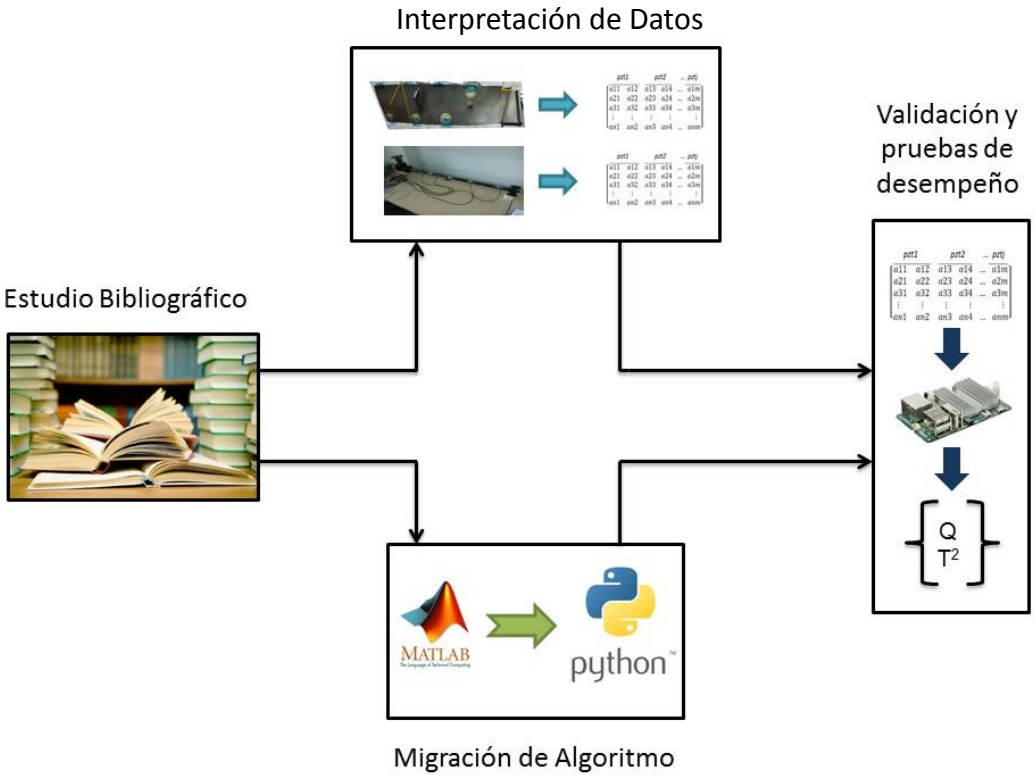


El presente trabajo de investigación se llevó a cabo de acuerdo a los pasos generales mostrados en la *Figura 11*. Como se explicó, la metodología se basó en la validación de los datos experimentales a través de la ejecución del algoritmo experto sobre las plataformas embebidas seleccionadas.

Dicha metodología general se describe bajo cuatro fases:

1) Estudio bibliográfico de la temática.
2) Interpretación de datos para procesamiento y algoritmo base.
3) Migración del algoritmo estadístico y analogías entre lenguajes de programación Matlab y Python.
4) Validación de datos experimentales y pruebas de desempeño sobre las plataformas embebidas.

Figura 11 Esquema básico para metodología de trabajo

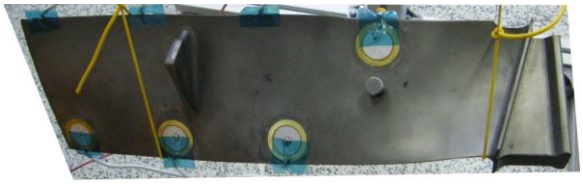


3.1. INTERPRETACIÓN DE DATOS DE PROCESAMIENTO

Parte del estudio de la temática presupone la forma en la que se interpretan los datos suministrados de la etapa de modelado y validación de las estructuras de estudio. Los experimentos de análisis corresponden al álabe la turbina de un avión[5] (*Figura 12*) y a una tubería acero-carbono[6] (*Figura 13*).

[5] Datos experimentales suministrados por el grupo CODALAB de la Universitat Politècnica de Catalunya UPB.
[6] Datos experimentales suministrados por el grupo CEMOS de la Universidad Industrial de Santander UIS.

Figura 12 Álabe de turbina de avión



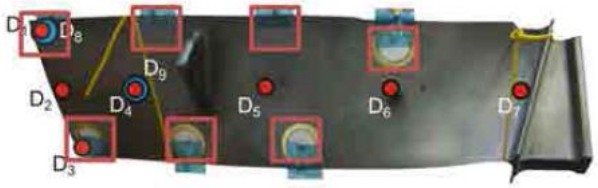
Fuente [3]

Figura 13 Tubería acero-carbono

Como se ha descrito anteriormente, los datos provienen de la cuantificación de la respuesta dinámica capturada por los piezoeléctricos, cuando la estructura se excita por medio de ondas vibracionales. Según lo dicho, cualquier variación de las propiedades físicas de la estructura (masa, amortiguamiento, rigidez) está representada en cambios de la respuesta dinámica estructural.

3.1.1. Datos para modelado PCA. En la *Figura 14* se observa una estructura en buen estado con diversos transductores PZT's sobre esta.

Figura 14 PZT’s sobre estructura se prueba



Fuente: [3]

La excitación de cada uno de los piezoeléctricos se debe al uso de ondas tipo Burst, y la respuesta obtenida por cada uno de estos, está debidamente organizada dentro de la matriz de casos sin daños. Dicha matriz es la utilizada para la creación del modelo estadístico basado en PCA y sus características son las mismas descritas en la *sección 3.2.1*.

Cada una de las estructuras de estudio en el presente trabajo de investigación (Alabe Turbina de Avión y Tubería Acero-Carbono) fue modelada de acuerdo a la actuación de cada uno de los N piezoeléctricos y la respuesta obtenida por los mismos.

3.1.2. Datos para Validación de los Experimentos. De acuerdo a lo descrito en la sección 3.2.2, la matriz de casos con daños es proyectada sobre el modelo previamente elaborado para el análisis de salud estructural. Estos daños para validación, representados dentro de la matriz de casos, pueden considerarse como grietas en la estructura de estudio, orificios, adición de masas, desgaste por efecto corrosivo, entre otros.

Los daños para las estructuras de estudio en el presente trabajo de investigación, fueron constituidos con masas agregadas, para el caso del álabe de la turbina de

avión, y/o filtraciones de aire controladas para la tubería acero-carbono. La acción de dichas anomalías fueron representadas en diferentes puntos de las estructuras, por lo tanto, se realizaron *Y* experimentos por cada daño ubicado.

La organización de la matriz de casos de daños fue descrita en la *sección 3.2.2* y concluye en el hecho de que cada uno de los casos esta ordenado con el fin de discernir uno de otro en la proyección de estos sobre el modelo de PCA.

3.1.2.1. Interpretación de datos Correlacionados: Investigadores están evaluando la forma en cómo es mejor procesar los datos y concluir la existencia de daños en una estructura. Dentro de sus consideraciones está el realizar la correlación de dos variables estadísticas, que en el caso de SHM son dos señales (una actuadora y otra de sensado) dado a que reduce el ruido y componentes de modo común [36]. Este tipo de técnica también es usada como base para la detección de formas modales [1] que son los diferentes tipos de respuesta que tiene una estructura ante entradas que oscilen a determinadas frecuencias cercanas a la frecuencia natural [37]. Con base en estas respuestas, se puede encontrar las propiedades dinámicas de la estructura a partir de análisis matemáticos interpretativos [38].

En la sección 2.2.2 se explicó la organización de la matriz de casos con daños cuyos datos han sido correlacionados previamente. Del análisis de cada fila (o experimentos de la matriz) se determina los respectivos índices de daño.

3.1.2.2. Interpretación de datos no Correlacionados: Cuando los datos para validación no se encuentran correlacionados, estos vienen en archivos de texto generados por PicoScope[7] y son organizados como se muestra:

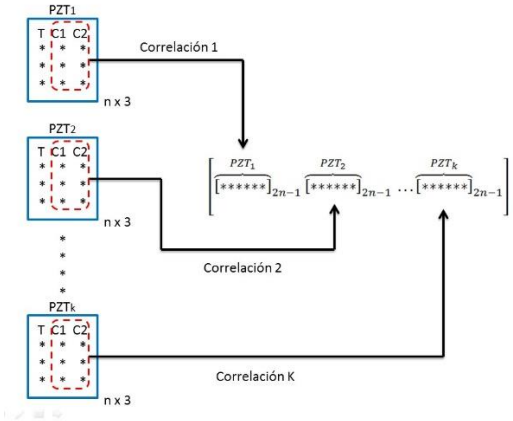
[7] Osciloscopio USB usado para obtener los datos experimentales en el marco del proyecto: “Monitorización y detección de defectos en estructuras usando algoritmos expertos embebidos” financiado por COLCIENCIAS.

Figura 15 Generación archivos de texto PicoScope

Cada archivo de texto corresponde a un registro realizado en un único PZT con un único daño sensado. El archivo contiene tres columnas, una corresponde al Tiempo de los registros, otra (Canal A) donde están las señales actuadoras, y una última columna (Canal B) con las señales sensadas, de modo que la correlación debe realizarse entre los Canales A y B para determinar la acción de la señal actuante sobre la sensada.

Un vector para procesamiento se completa dependiendo del número de registros, de modo que si hay *k* piezoeléctricos, deben correlacionarse *k* registros del mismo daño, como se muestra en la *Figura 16*.

Figura 16 Correlación de registros por daño



Mientras dicho vector se completa, los sub-vectores generados en la correlación se van almacenando en un Buffer hasta obtener el vector para procesamiento.

3.1.3. Derivación de Algoritmo base. El código programado en lenguaje Python para sistemas embebidos está basado en la revisión bibliográfica para la teoría de desarrollo de PCA, y en un algoritmo elaborado en Matlab por Camacho [37] en el marco del proyecto: “Sistema Experto para la Monitorización de Salud Estructural mediante el Reconocimiento de Patrones: Adaptación y Validación Numérica”. El código base en Matlab, contiene las dos etapas de PCA para SHM: Modelado y Validación.

Los resultados obtenidos a partir de [37] demuestran un alto consumo de recursos computacionales debido a la cantidad de datos utilizados para ambas etapas. De modo que para lograr implementar este tipo de algoritmo experto en SHM sobre plataformas embebidas y determinar fallas sobre las estructuras de estudio, se requiere aplicar solo la etapa de validación, mientras que la etapa de modelado se realiza en un sistema de cómputo robusto. Esto a fin de demostrar la aplicabilidad de los sistemas embebidos para requerimientos en SHM y mejorar el procesamiento en línea de una estructura de estudio, si previamente ya se tiene almacenado el modelo de la misma.

3.2. MIGRACIÓN DEL ALGORITMO ESTADISTICO Y ANALOGÍAS ENTRE LENGUAJES DE PROGRAMACIÓN MATLAB Y PYTHON.

Python es un lenguaje de programación multiplataforma de licencia de uso libre (GPL), bajo requerimiento de almacenamiento en disco (95 MB aproximadamente), alto desempeño en la ejecución de algoritmos y amplio soporte en línea [1]. Las ventajas que ofrece sobre otros lenguajes de programación se deben a su estructura de programación bien diseñada y fácil de leer, y la practicidad de implementación sobre plataformas embebidas con sistemas operativos integrados como las usadas en el presente trabajo. De esta

forma, Python se adecua a las características de implementación y ejecución del algoritmo estadístico.

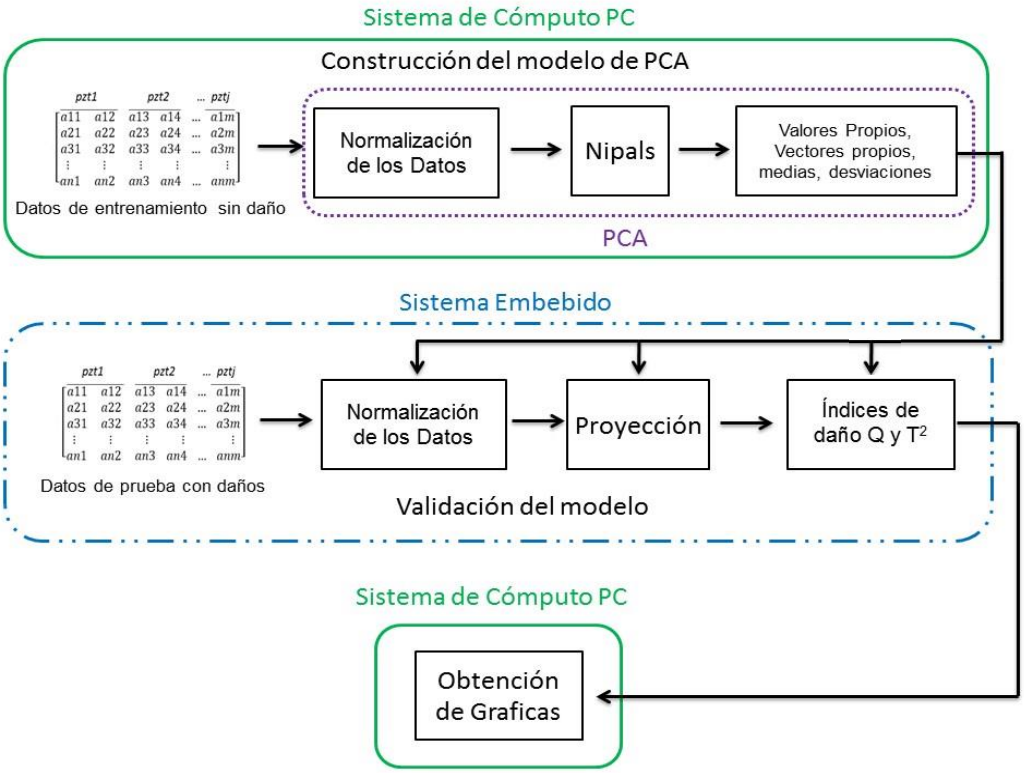
La migración de código consistió en analizar la estructurahost del algoritmo base, verificando de esta manera la analogía entre los comandos de Matlab y Python, si existen o no dichas analogías o si deben crearse funciones en Python equivalentes. Al finalizar este procedimiento, se comprobó la funcionalidad del programa descrito en Python con el diseñado en Matlab; este proceso residió en ejecutar y observar, detenidamente, si los resultados obtenidos del procesamiento de los datos adquiridos son los mismos obtenidos en Matlab, de ser así, el programa en Python desarrolla completamente el procesamiento de validación.

3.3. VALIDACIÓN DE DATOS EXPERIMENTALES Y PRUEBAS DE DESEMPEÑO SOBRE LAS PLATAFORMAS EMBEBIDAS

Dado a que el algoritmo busca la validación de los experimentos, los datos correspondientes al modelado pueden ser almacenados previamente en las tarjetas de desarrollo o enviados a estas desde el *host* o computador (*Figura* 17); en concreto, estos datos corresponden a los vectores de medias, desviaciones estándar, eigenvalores, eigenvectores.

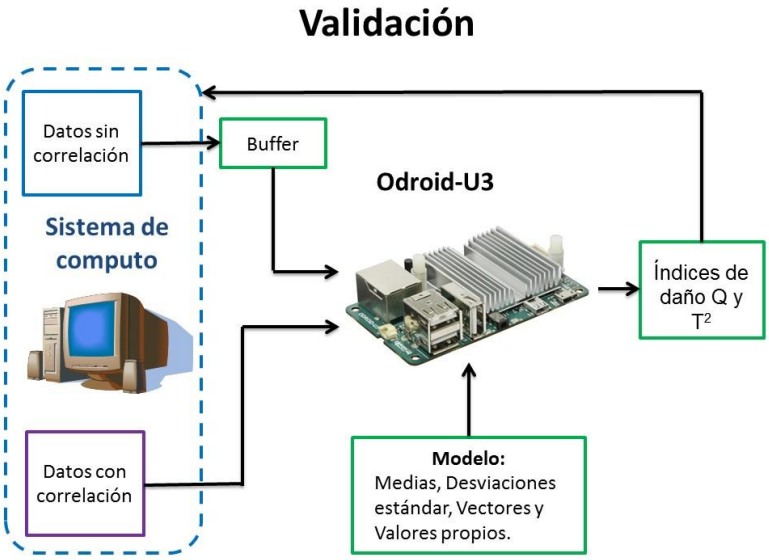
Las pruebas de tipo *Hardware in the loop* fueron realizadas en las tarjetas BeagleBone Black y Odroid-U3 con el fin de determinar rendimiento, que para el caso se define como tiempos de procesamiento y uso de memoria en la ejecución del código sobre las plataformas.

Figura 17 Esquema general para validación en los sistemas embebidos



La *Figura 18* muestra el procedimiento general para validar los datos suministrados en modo *Hadware in the Loop*. El modelo está almacenado en la tarjeta embebida mientras que los vectores de casos se encuentran en un computador. El algoritmo desarrollado dentro de la Odroid-U3 y BeagleBone llama a dichos vectores para procesamiento, obtiene los respectivos índices de daño y envía, de forma autónoma, los resultados al computador. El procedimiento se repite hasta completar el número de registros por daño almacenados en el computador.

Figura 18 Proceso para validación de datos



Finalmente los índices de daño T2 y Q-estadístico y el tiempo de procesamiento de todos los datos son registrados a fin de realizar gráficas comparativas. El trabajo concluye en la determinación de la aplicabilidad y desempeño de plataformas embebidas para los requerimientos en SHM.

4. RESULTADOS

Los resultados del procesamiento de los datos de las pruebas experimentales de interés para el presente trabajo de investigación se describen como: Tiempo de ejecución, porcentaje de memoria RAM y Comparación de algoritmos.

Es de resaltar las dimensiones de las matrices de procesamiento para cada uno de los elementos de prueba; la adición de masas en diferentes partes de las estructuras así como filtraciones de aire controladas, son consideradas como daños; de modo que las dimensiones de las matrices varían respecto al número de daños, el número de experimentos, el número de sensores y el número de muestras por sensor. Las matrices utilizadas corresponden a:

- [100x24000] datos para los experimentos entregados por CODALAB del alabe de la turbina de un avión. *Los datos ya están correlacionados.* La matriz está distribuida como sigue:
    - 10 experimentos sin daños.
    - 9 daños en la estructura, cada uno con 10 experimentos.
    - 1 piezo-actuador en la estructura
    - 6 piezo-sensores utilizados, 4000 muestras por cada piezo-sensor.

- [3200x19235] datos para los experimentos realizados en la tubería acero-carbono. *Los datos ya están correlacionados*. Igualmente, dicha matriz se distribuye:
    - 200 experimentos sin daños.
    - 15 daños en la estructura, cada uno con 200 experimentos.
    - 1 piezo-actuador en la estructura

- 1 piezo-sensor utilizado, 19235 muestras por piezo-sensor.

Es de observar que las dimensiones de las matrices de procesamiento son elevadas, por lo que el modelo entregado de las estructuras usó NIPALS como método iterativo: *10 componentes principales en los experimentos del Alabe de Turbina de Avión* y *15 componentes de la Tubería Acero-Carbono.* Este número de componentes se seleccionaron a partir del aporte de información de estas sobre el grupo de datos. Cuando una componente calculada aporta menos del 10% de la energía dada por la primera componente principal, su valor se desestima y no es necesario calcular más componentes[8].

4.1. TIEMPO DE EJECUCIÓN EN LAS TARJETAS DE DESARROLLO

El tiempo que tomó cada programa en ejecutarse y procesar los datos, tanto de la tubería acero-carbono y los entregados por el grupo CODALAB, en cada sistema embebido, se halló mediante el uso del comando *time.time* de Python, que permite obtener el tiempo en segundos desde el inicio del comando hasta que encuentra una bandera de cierre al final del algoritmo.

Se evaluó la forma de cómo administrar los datos de validación en los sistemas embebidos para la multiplicación con los valores del modelo previamente almacenado. La Figura 19 y Figura 20 muestran las posibles formas de carga y multiplicación.

[8] Este es el primer criterio para seleccionar el número de componentes principales, aunque investigadores están buscando el mejor método para seleccionar el dicho número.

Figura 19 Multiplicación sub-vectores de datos con sub-matrices de P

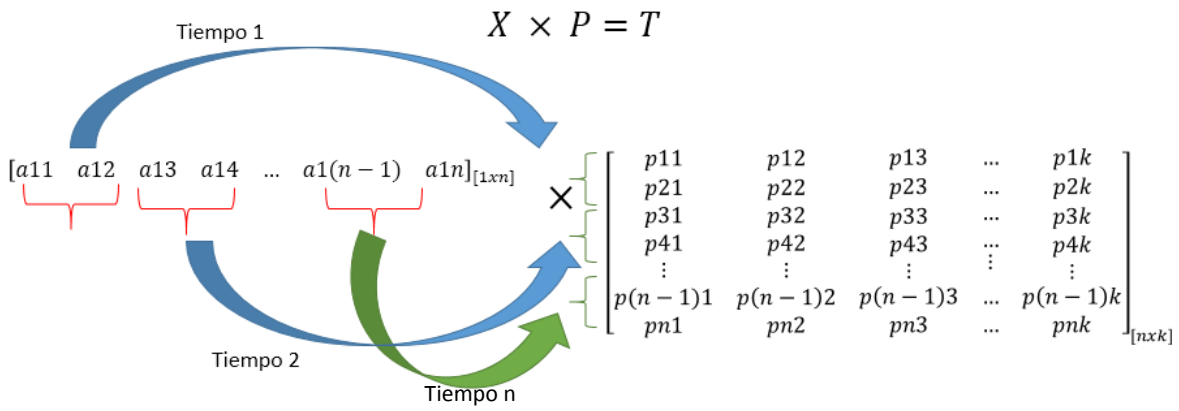
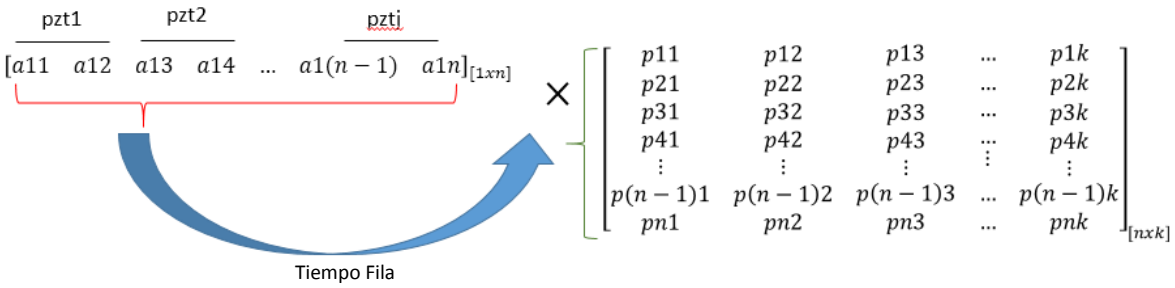


Figura 20 Multiplicación vector de datos con matriz completa de P



Como se explicó en el Capítulo 2, la validación comprende la proyección y contraste de los datos adquiridos en un modelo base sin daños previamente realizado, esto se resume en una multiplicación matricial para obtener los datos transformados *T* o Scores (Ecuación 2). Para los datos mostrados en la Tabla 2 se tomó un vector de datos adquiridos (1 experimento compuesto por 9 piezoeléctricos, cada uno con 8933 muestras) y se multiplicó por una matriz de vectores propios *P* (de dimensiones 80397 x 20) en las formas ilustradas anteriormente.

Tabla 2 Tiempos de multiplicación

	Tiempos [s]
Multiplicación sub-vectores con sub-matrices (*Figura 19*)	Tiempo 1 = 0.46602606 Tiempo 2 = 0.48402690 Tiempo 3 = 0.49002814 Tiempo 4 = 0.54203104 Tiempo 5 = 0.53903102 Tiempo 6 = 0.53903102 Tiempo 7 = 0.51602911 Tiempo 8 = 0.55503082 Tiempo 9 = 0.48502802 Total Fila = 4.61626219
Multiplicación vector completo con matriz completa (*Figura 20*)	Tiempo Fila = 3.90922403

Operar la validación por filas completas o vectores presenta mejores resultados, de forma tal que el modo de operación se ejecutó fila a fila, es decir experimento por experimento, así por cada experimento se obtiene un índice T y un índice Q.

De las estructuras de estudio, fueron entregadas matrices cuyas dimensiones se explicaron al inicio del presente capitulo y resultados en tiempo de procesamiento se presentan en la Tabla 3.

Tabla 3 Tiempos de procesamiento de sistemas embebidos

Tiempo procesamiento datos [s]			
		Tubería Acero-Carbono	**Alabe de Turbina de Avión**
Odroid-U3	Promedio por vector	0.72810490489	0.87602640033
	Total de vectores	2184.31471467	87.6026400328
BeagleBone Black	Promedio por vector	4.1262843855	4.87064123988
	Total de vectores	12378.85316	487.064123988

Los tiempos de ejecución muestran una brecha amplia que depende a las características en hardware de cada sistema embebido y de las dimensiones de las matrices de procesamiento.

4.2. PORCENTAJE DE MEMORIA RAM

Los recursos usados por los sistemas embebidos y que interesan para el análisis de desempeño se dividen en: porcentaje de memoria usada cuando se importa el programa y sus librerías, y cuando se procesan los datos ejecutando el algoritmo. Estos resultados se obtuvieron usando el comando de Linux *htop* que permite observar cuantos recursos está usando un programa determinado cuando se está ejecutando.

Inicialmente el código fue descrito como una función que requiere de carga previa en Python. Este proceso requería cierto tiempo y consumo de memoria de importación al sistema embebido. Es así como se dejó de definir al algoritmo como una función, y los gastos en recursos de memoria en importación no son necesarios.

Los resultados de porcentaje de memoria usado por los sistemas embebidos se muestran en la siguiente tabla:

Tabla 4 Porcentaje de memoria al ejecutar el algoritmo experto

	% Memoria Tubería	% Memoria CODALAB
Odroid-U3	9.4%	6.9%
BeagleBone Black	53.2%	36.1%

Las especificaciones de las tarjetas son claramente diferentes (*Tabla 1*). Se observa que la Odroid U3 posee mejores características que la BeagleBone Black. Equivalente a esto, los recursos utilizados en la Beagle son mayores y el tiempo de procesamiento se extiende, como es posible observar en la *Tabla 3*. No obstante, la ejecución del algoritmo es completada por cada una de las tarjetas y el error obtenido es casi nulo respecto a lo caracterizado previamente en Matlab. En la *sección 4.3.1* se hará mención del cálculo del error de datos de salida por cada uno de los sistemas embebidos.

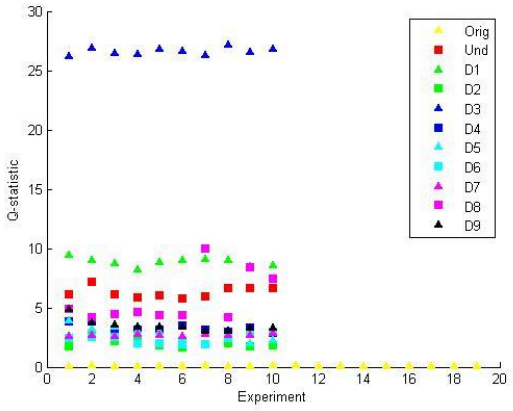
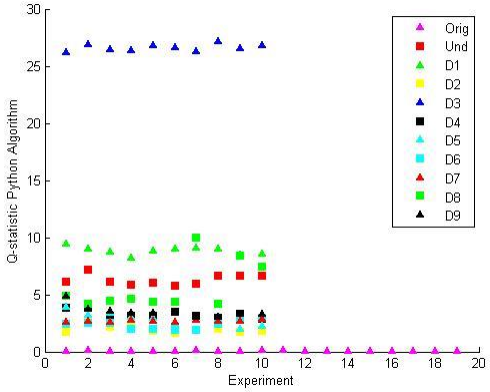
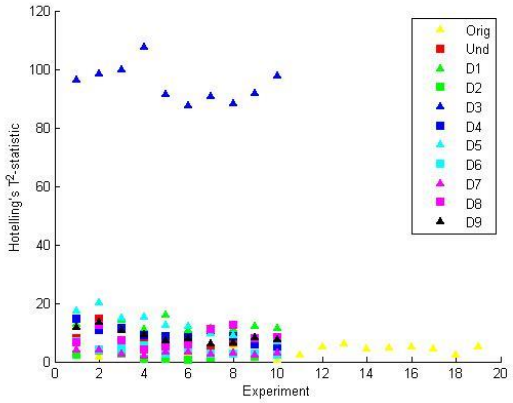
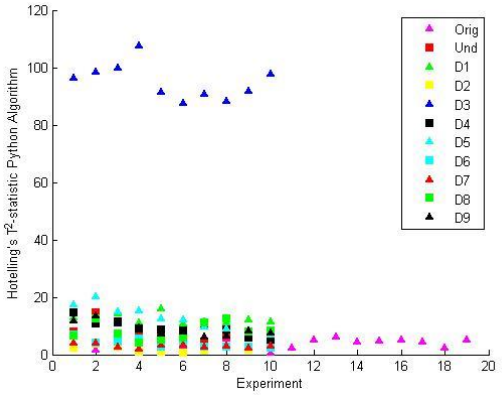
4.3. COMPARACIÓN DE ALGORITMOS

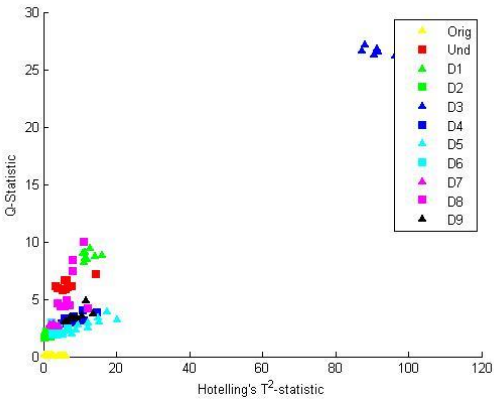
Para comprobar que el funcionamiento del algoritmo realizado en Python es igual al algoritmo base del que se migro, los elementos de salida del algoritmo, para el caso son los índices de daño obtenidos, se contrastan con el modelo base de la estructura en busca de discrepancias. A través del contraste gráfico resulta notorio observar la existencia o no de daños presentes en la estructura.

Las gráficas expuestas en la *Tabla 5* y
Tabla 6 muestran los resultados referentes a los índices de daño para cada una de las estructuras: Alabe de turbina de avión (Grupo CODALAB) y Tubería Acero-Carbono (UIS) respectivamente.

Las leyendas en cada una de las gráficas caracterizan tanto al modelo base (simbolizada *Orig y UND*) como a los daños en las estructuras, de modo que *D1* representa al primer daño, *D2* al segundo daño y así sucesivamente.

Tabla 5 Comparación de gráficas de datos procesados en Matlab y Python: Alabe de turbina de avión.

Gráficas de datos procesados-Alabe de turbina de Avión-CODALAB	
Procesamiento en Matlab	Procesamiento en Python
	
	



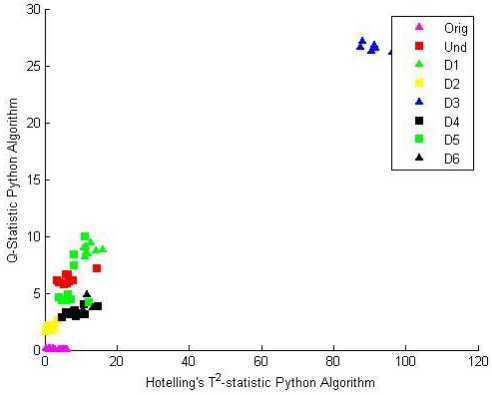
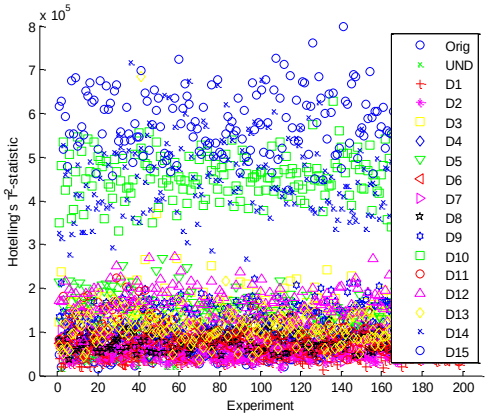
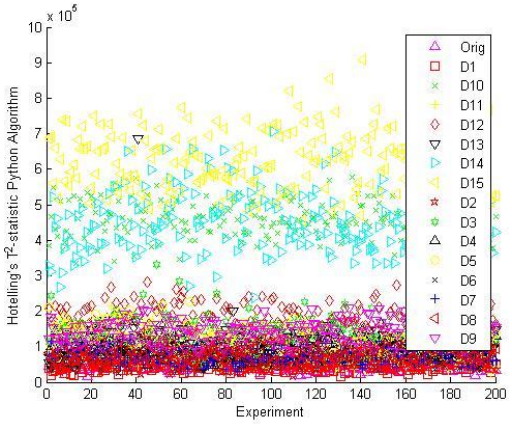
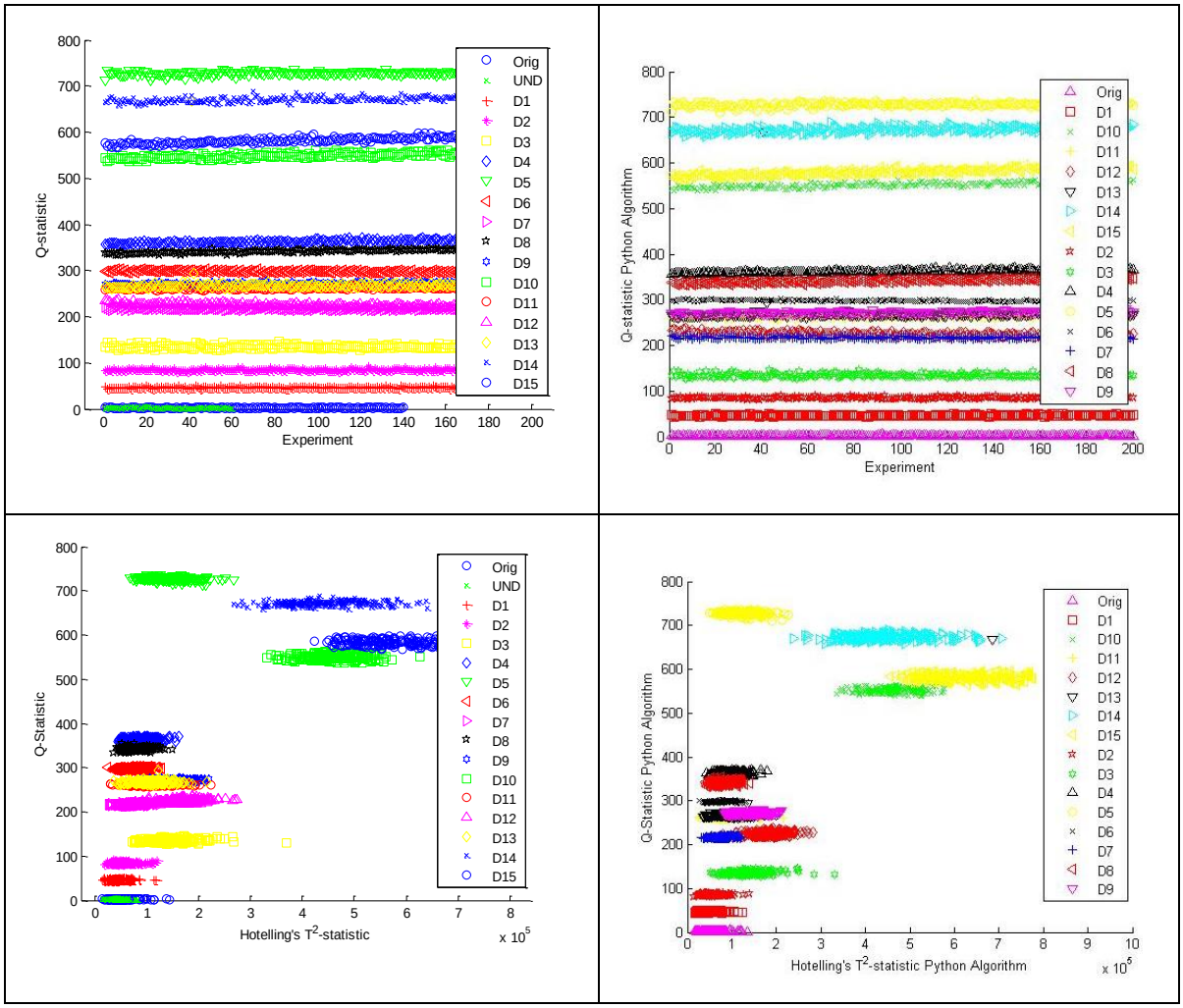


Tabla 6 Comparación de gráficas de datos procesados en Matlab y Python: Tubería Acero-Carbono.

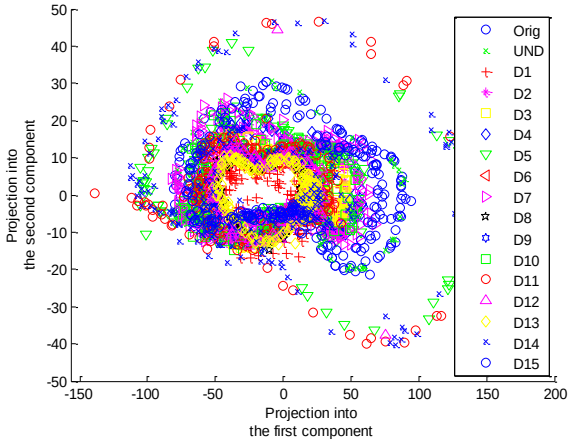
Gráficas de datos procesados-Tubería Acero Carbono-UIS	
Procesamiento en Matlab	Procesamiento en Python
	


Q-statistic
Experiment
Orig
UND
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
Q-statistic Python Algorithm
Experiment
Orig
D1
D10
D11
D12
D13
D14
D15
D2
D3
D4
D5
D6
D7
D8
D9
Q-Statistic
Hotelling's T2-statistic
x 10^5
Orig
UND
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
Q-Statistic Python Algorithm
Hotelling's T2-statistic Python Algorithm
x 10^5
Orig
D1
D10
D11
D12
D13
D14
D15
D2
D3
D4
D5
D6
D7
D8
D9

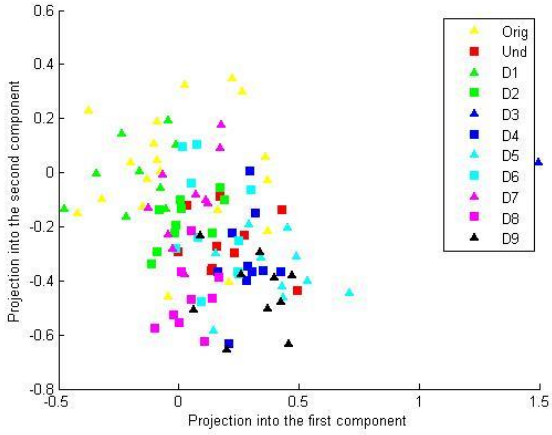
Las imágenes corresponden a la proyección grafica de cada uno de los experimentos realizados y de los cuales es posible discernir la diferencia existente entre cada uno de los daños y la estructura original, en especial si se contrastan los índices de daño T^2 y Q-estadístico en un solo plano. Sin embargo no todas las gráficas son dicientes en lo que reflejan, por ejemplo la *Figura 21* muestra la proyección de los daños sobre el espacio vectorial de las primeras componentes principales luego de hacer PCA. Se observa que la distinción de los daños no es clara por lo que no se lograría constatar en concreto la existencia de alguna falla. En este sentido, el uso de índices de daño reorganiza y clarifica los datos, permitiendo concluir sobre los resultados adquiridos.

Figura 21 Proyección de los daños sobre el espacio vectorial de las primeras componentes principales creado para cada estructura

Tubería Acero-Carbono



Alabe de Turbina de Avión



4.3.1. Calculo del Error en la comparación de datos. En las gráficas anteriores es posible observar que la diferencia entre cada par de gráficas es casi nula. Sin embargo, los resultados de la ejecución del código varían respecto de un lenguaje de programación a otro (Matlab y Python) dado al truncamiento de cifras decimales como característica propia de cada uno de estos. El error obtenido fue

menor al 0.1% como es posible ver en la Figura 22 y la Figura 23 para cada uno de los casos de estudio.

Figura 22 Error de procesamiento de datos: Álabe de turbina de avión

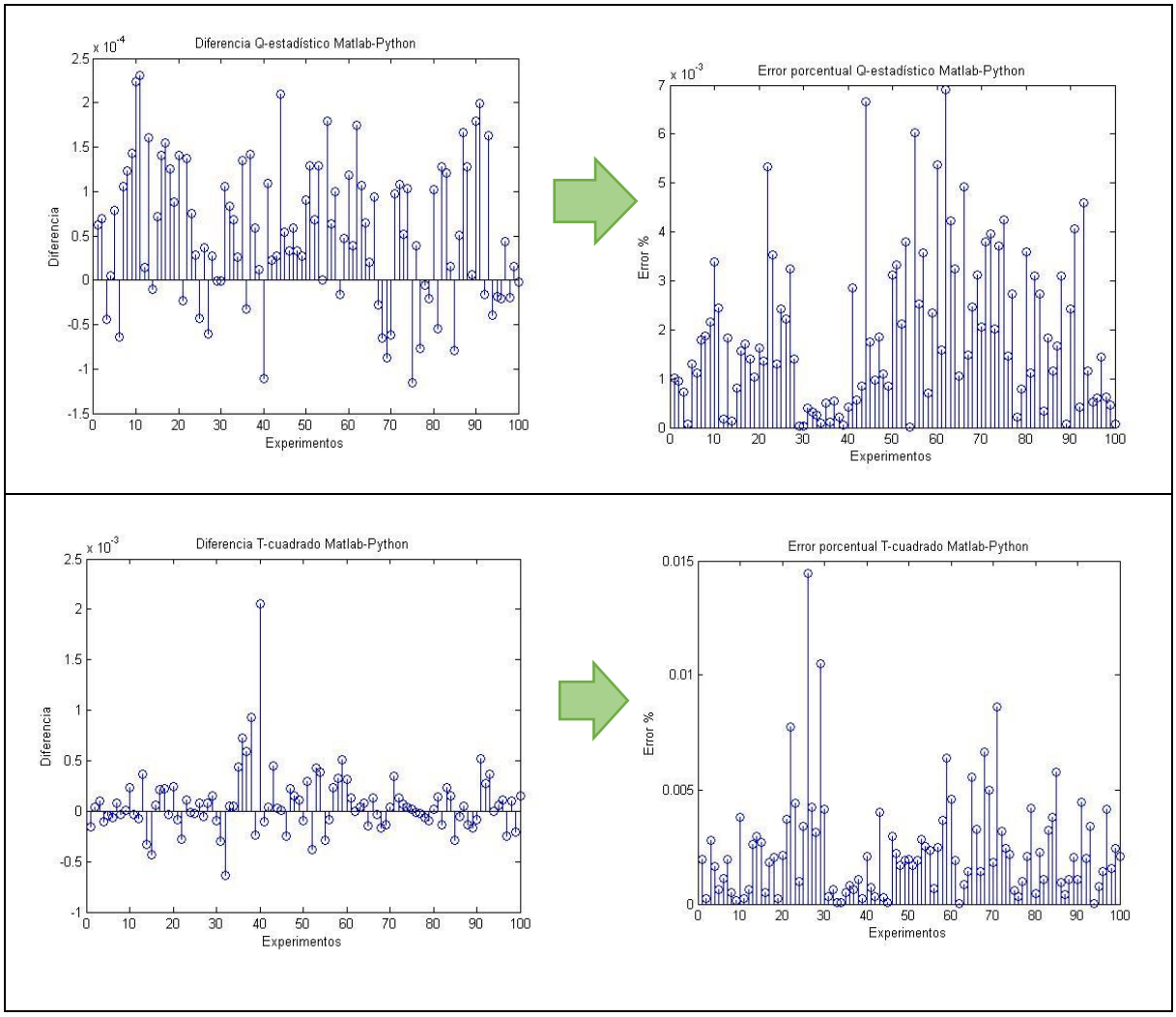
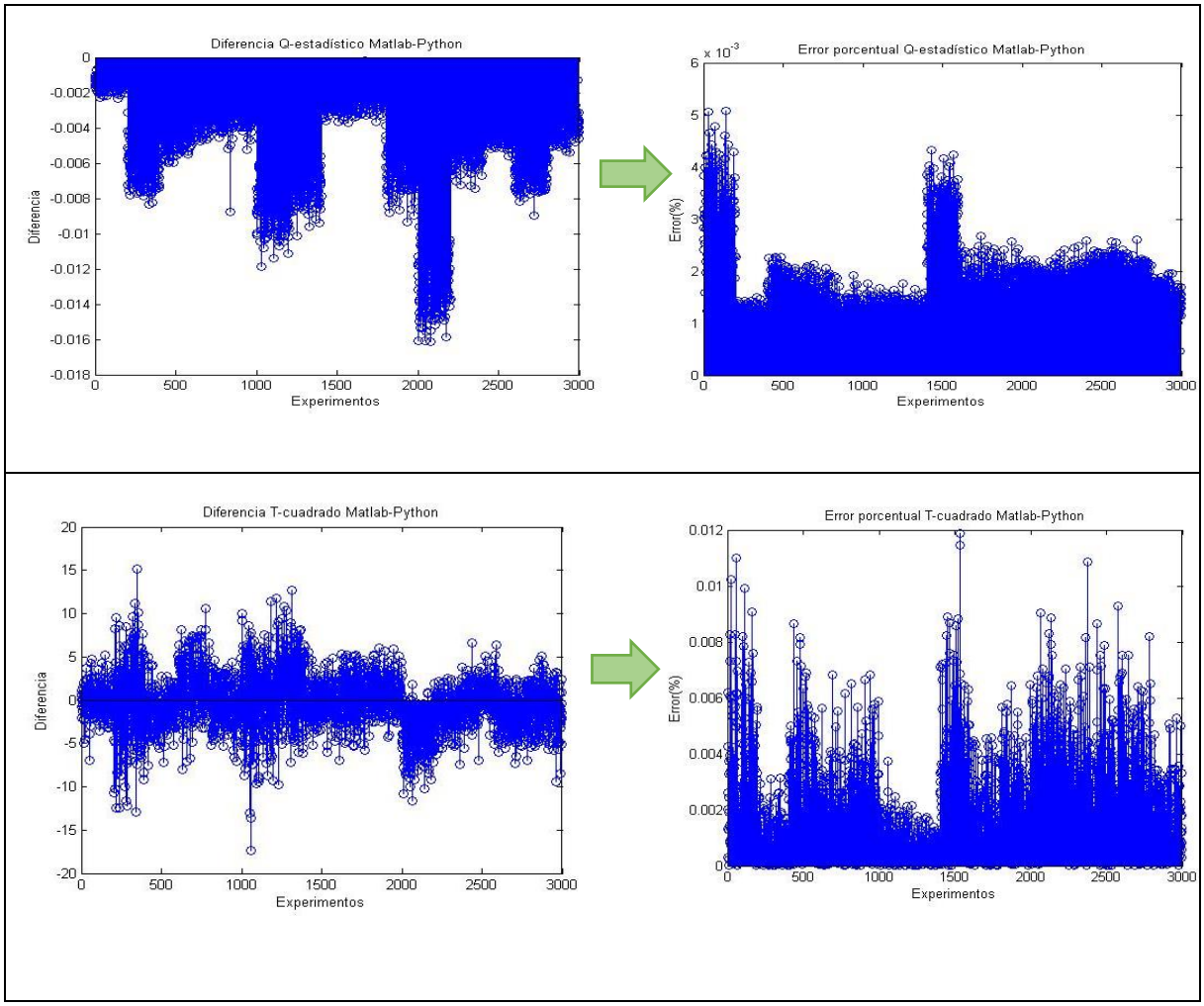


Figura 23 Error de procesamiento de datos: Tubería acero-carbono



4.3.1. Comparación de resultados: Tarjeta Odroid U3 y Computador. El contraste de la ejecución del algoritmo experto entre un sistema embebido y un computador, permite cerciorar cuan practico es el uso de tarjetas de desarrollo en la implementación de código bajo lenguajes multiplataforma. La prueba consistió en ejecutar el algoritmo experto en Python a través del sistema embebido, mientras que en el computador, el código base de Matlab, así pues, se tomaron los datos correspondientes al tiempo de procesamiento y uso de memoria RAM durante la ejecución de ambos códigos. La comparación fue realizada para una tarjeta Odroid U3 y una laptop SONY VAIO PCG-5KFP cuyas especificaciones

primordiales se muestran a continuación (ver *Tabla 7*), los resultados de la prueba se reflejan en la *Tabla 8*.

Tabla 7 Características de sistemas de cómputo

	Laptop SONY VAIO	**ODROID U3**
Sistema Operativo	Microsoft Windows 7	Embedded Linux 32-bits Xubuntu
Procesador	Intel Core™2 Duo Processor T8100 (2.10 GHz)	1.7GHz **Exynos4412 Prime** Cortex-A9 Quad-core processor with PoP (Package on Package) 2Gbyte LPDDR2 880Mega Data Rate
RAM	2 GB	2GB

Tabla 8 Resultados de ejecución de código

		Laptop SONY VAIO	**ODROID U3**
Tiempo de ejecución de datos [segundos]	Álabe Turbina de Avión	21.687860	87.6026400328
	Tubería Acero-Carbono	701.117725	2184.31471467
Memoria RAM utilizada en el proceso [%]	Álabe Turbina de Avión	18.5 %	6.9%
	Tubería Acero-Carbono	65.0 %	9.4%

Nota: Los tiempos corresponden a procesar el total de vectores que componen la matriz de casos con daños

5. CONCLUSIONES

Con el presente trabajo de investigación se logró demostrar la viabilidad del uso de sistemas embebidos para el procesamiento de datos de estructuras sensadas. De esta forma es posible evitar el uso sistemas computacionales robustos para requerimientos en SHM, y por su parte, introducir nuevas técnicas dentro de este campo.

Experimentalmente se evidenció que los resultados de índices de daño obtenidos en la ejecución del código varían mínimamente respecto de un lenguaje de programación a otro (Matlab y Python) por el truncamiento de cifras decimales; esto es característico de cada lenguaje. No obstante, los resultados no varían de una tarjeta embebida a otra, cuando sobre estas se ejecuta el código experto programado en Python, ni cuando se comparan con los obtenidos en un computador con el mismo lenguaje de programación.

A pesar de la diferencia en tiempos de ejecución del algoritmo y porcentajes de memoria utilizada por los sistemas embebidos Odroid-U3 y BeagleBone Black, no es posible hacer un contraste directo en cuanto a la ejecución de código de dichos sistemas, debido a que no poseen los mismos recursos de hardware. Sin embargo, se comprobó que aquellas ejecuciones se cumplen a cabalidad para cada uno de los sistemas mencionados sin utilizar más del 60% de los recursos de memoria. Es posible que a través de un proceso de optimización se pueda aumentar este porcentaje con el fin de disminuir posiblemente el tiempo para la obtención de resultados.

La literatura no muestra términos comparativos en cuanto al rango de aceptación de los tiempos de procesamiento de las plataformas embebidas según sea los

requerimientos para SHM, dado a que actualmente se está explorando este tipo de aplicaciones. Lo anteriormente dicho, justifica el hecho de realizar un contraste de desempeño cuando el algoritmo experto se ejecuta sobre una tarjeta Odroid-U3, con lenguaje de programación Python, y sobre un computador (Sony VAIO en este caso), con lenguaje de programación Matlab y características en hardware similares a la Odroid-U3. Como resultado, se pudo observar que el sistema embebido tarda aproximadamente tres veces más en obtener los índices de daños con respecto al computador, pero la cantidad de recursos de memoria utilizados durante el proceso son inferiores.

Los resultados obtenidos por el algoritmo desarrollado en Python muestran porcentajes de errores menores al 0.07% en contraste a los resultados obtenidos por el algoritmo realizado en Matlab en el proyecto "*Sistema experto para la monitorización de salud estructural mediante el reconocimiento de patrones: Adaptación y validación numérica"* [36], el cual fue base para la realización de este proyecto.

6. TRABAJOS FUTUROS

Una vez culminado este proyecto y tomando como base los resultados obtenidos sobre la viabilidad del uso de sistemas embebidos para la implementación de monitorización de salud estructural, se pueden desarrollar trabajos futuros orientados a:

- Implementación de la etapa de modelado y validación a partir de PCA sobre sistemas embebidos.
- Implementación de algoritmos de salud estructural sobre sistemas embebidos en tiempo real.
- Implementar un algoritmo de salud estructural basado en redes neuronales SOM sobre sistemas embebidos.

REFERENCIAS

[1] A. F. Q. Parra y R. V. Mejía, «Estado del arte en monitorización de salud estructural: Un enfoque basado en agentes inteligentes,» *Ciencia e Ingeniería Neogranadina,* vol. 20, nº 1, pp. 117-132, Junio 2010.

[2] L. E. R. J. F. A. &. G. A. Mujica, «Q-statistic and T2-statistic PCA-based measures for damage assessment in structures,» *Structural Health Monitoring,* vol. 10(5), pp. 539-553, 2010.

[3] M. J. Simões Ferreira dos Santos , Ondas ultra-sonoras guiadas na caracterização e controlo não destrutivo de materiais, Coimbra, 2004.

[4] N. A. Leyla, E. Moulin, J. Assaad, F. Benmeddour, S. Grondel y Y. Zaatar, «Application of piezoelectric transducers in structural health monitoring techniques,» *Advances in Piezoelectric Transducers,* pp. 87-104, 2011.

[5] C. R. Farrar, F. M. Hemez, D. D. Shunk, D. W. Stinemates, B. R. Nadler y J. J. Czarnecki, A review of structural health monitoring literature: 1996-2001, Los Alamos, 2004.

[6] H. Sohn, C. R. Farrar, N. F. Hunter y K. Worden, «Structural health monitoring using statistical pattern recognition techniques,» *Journal of dynamic systems, measurement, and control,* vol. 123(4), pp. 706-711, 2001.

[7] D. A. Tibaduiza, M. A. Torres Arredondo, L. E. Mujica, J. Rodellar y C. P. Fritzen, «A study of two unsupervised data driven statistical methodologies for detecting and classifiying damages in structrual health monitoring,» *Mechanical Systems and Signal Processing,* vol. 41, pp. 467-484, 2013.

[8] M. Lallart, D. Guyomar y T. Monnier, «Low-power computation methods and self-powered systems for Structural Health Monitoring techniques using Lamb waves for embedded sensing,» *Shock and Vibration,* vol. 19, pp. 867-877, 2012.

[9] G. Dib, R. Lattrelf, L. Udpa y G. Yang, «A wireless sensor node for structural health monitoring using guided wave testing,» *Materials Evaluation,* vol. 71(10),

pp. 1232-1241, 2013.

[10] W. J. Staszewski, S. Mahzan y R. Traynor, «Health monitoring of aerospace composite structures–Active and passive approach,» *Composites Science and Technology,* vol. 69(11), pp. 1678-1685, 2009.

[11] D. Giraldo, A structural health monitoring framework for civil structures, Saint Louis, 2006.

[12] A. Rytter, Vibration Based Inspection of Civil Engineering Structures, Aalborg, 1993.

[13] S. Yuan, Y. Xu y G. Peng, «New Developments in Structural Health Monitoring Based on Diagnostic Lamb Wave,» *J. Mater. Sci. Technol.,* vol. 20(05), pp. 490-496, 2004.

[14] Y. Y. Lim, S. Bhalla y C. K. Soh, «Structural identification and damage diagnosis using self-sensing piezo-impedance transducers,» *Smart Materials and Structures,* vol. 15(4), p. 987, 2006.

[15] G. Manson, K. Worden, K. Holford y R. Pullin, «Visualisation and dimension reduction of acoustic emission data for damage detection,» *Journal of Intelligent Material Systems and Structures,,* vol. 12(8), pp. 529-536, 2001.

[16] M. Cammarata, P. Rizzo, D. Dutta y H. Sohn, «Application of principal component analysis and wavelet transform to fatigue crack detection in waveguides,» *Smart structures and systems,* vol. 6, nº 4, pp. 349-362, 2010.

[17] F. Gharibnezhad, L. E. Mujica y J. Rodellar, «Applying robust variant of Principal Component Analysis as a damage detector in the presence of outliers,» *Mechanical Systems and Signal Processing,* vol. 50, pp. 467-479, 2015.

[18] D. A. Tibaduiza Burgos, Design and validation of a structural health monitoring system for aeronautical structures, Barcelona, 2012.

[19] R. Villamizar, J. L. Quiroga, J. Camacho, L. E. Mujica y M. L. Ruiz, «Structural Damage Detection Algorithm Based on Principal Component Indexes and Embedded on a Real Time Platform,» *EWSHM-7th European Workshop on*

Structural Health Monitoring, pp. 1553-1560, 2014.

[20] V. Giurgiutiu, Structural health monitoring: with piezoelectric wafer active sensors, Academic Press-Elsevier, 2007.

[21] Fraunhofer-Institut für Silicatforschung ISC, *Piezoelectric transducers for structural health monitoring,* Fraunhofer ISC.

[22] M. L. Ruiz Ordóñez, Multivariate statistical process control and case-based reasoning for situation assessment of sequencing batch reactors, Girona, 2008.

[23] K. Varmuza y P. Filzmoser, Introduction to multivariate statistical analysis in chemometrics, Boca raton: CRC Press-Taylor & Francis Group, 2009.

[24] K. H. Esbensen, D. Guyot, F. Westad y L. P. Houmolle, Multivariate data analysis in practice: an introduction to multivariate data analysis and experimental design, camo, 2002.

[25] C. F. Alcala y S. J. Qin, «Unified analysis of diagnosis methods for process monitoring,» *Proceedings of the 7th IFAC Symposium on Fault detection, supervision and safety of technical processes,* pp. 1-3, 2009.

[26] A. M. Escobar Arias y A. J. Flórez Martínez, Manuales Proyecto de grado Cámara Elphel y Brazo Puma, Bucaramanga, 2013.

[27] O. Bennouna y J. P. Roux, «Real Time Diagnosis & Fault Detection for the Reliability Improvement of the Embedded Systems,» *Journal of Signal Processing Systems,* vol. 73, nº 2, pp. 153-160, 2013.

[28] S. Lu, Q. He, F. Hu y F. Kong, «Sequential Multiscale Noise Tuning Stochastic Resonance for Train Bearing Fault Diagnosis in an Embedded System,» *Instrumentation and Measurement, IEEE Transactions,* vol. 63, nº 1, pp. 106 - 116, 2013.

[29] B. Kim y H. M. Park, «Efficient face recognition based on MCT and I(2D)^2PCA,» *IEEE International conference on Systems, Man and Cybernetics,* pp. 2585-2590, 2012.

[30] D. G. Perera y K. F. Li, «Embedded hardware solution for principal component

analysis,» *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference,* pp. 730-735, 2011.

[31] Hardkernel co., Ltd., «Magazine Odroid,» Enero 2015. [En línea]. Available: http://magazine.odroid.com/assets/201401/pdf/ODROID-Magazine-201401.pdf. [Último acceso: Diciembre 2014].

[32] L. Świrski, A. Bulling y N. Dodgson, «Robust real-time pupil tracking in highly off-axis images,» *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA,* pp. 173-176, 2012.

[33] J. Pavlus, «MIT Technology Review,» 2013 Abril 2013. [En línea]. Available: http://www.technologyreview.com/view/514036/beaglebone-black-a-makers-dream/. [Último acceso: Agosto 2014].

[34] T. Smith, «The Register,» 11 Junio 2013. [En línea]. Available: http://www.theregister.co.uk/2013/06/11/review_beagleboard_beaglebone_black/. [Último acceso: Agosto 2014].

[35] Adafruit, «Adafruit-BeagleBone,» [En línea]. Available: https://learn.adafruit.com/category/beaglebone. [Último acceso: Septiembre 2014].

[36] J. ZHANG, S. ZHU, Y. XU y Z. CHEN, «SHM-based Correlation Study of Trainload-induced Response in Tsing Ma Bridge,» *14 Asia Pacific Vibration Conference ,* pp. 113-122, 2011.

[37] P. AVIATABLE, «Experimental Modal Analysis (A Simple Non-Mathematical Presentation),» *Modal Analysis and Control Laboratory, Sound and Vibration Magazine, University Of Massachusetts.*

[38] J. M. Caicedo, «Structural Health Monitoring for flexible Covil Structures,» *Washington University, Department Of Civil Engineer Doctoral Thesis.*

[39] J. Camacho Navarro, Sistema Experto para la Monitorización de Salud Estructural mediante el Reconocimiento de Patrones: Adaptación y Validación Numérica, Bucaramanga, 2010.

[40] Python Software Foundation, «Python,» 2001-2015. [En línea]. Available: https://www.python.org/.

BIBLIOGRAFÍA

A. F. Q. Parra y R. V. Mejía, «Estado del arte en monitorización de salud estructural: Un enfoque basado en agentes inteligentes,» Ciencia e Ingeniería Neogranadina, vol. 20, nº 1, pp. 117-132, Junio 2010.

A. M. Escobar Arias y A. J. Flórez Martínez, Manuales Proyecto de grado Cámara Elphel y Brazo Puma, Bucaramanga, 2013.

A. Rytter, Vibration Based Inspection of Civil Engineering Structures, Aalborg, 1993.

Adafruit, «Adafruit-BeagleBone,» [En línea]. Available: https://learn.adafruit.com/category/beaglebone. [Último acceso: Septiembre 2014].

B. Kim y H. M. Park, «Efficient face recognition based on MCT and I(2D)^2PCA,» IEEE International conference on Systems, Man and Cybernetics, pp. 2585-2590, 2012.

C. F. Alcala y S. J. Qin, «Unified analysis of diagnosis methods for process monitoring,» Proceedings of the 7th IFAC Symposium on Fault detection, supervision and safety of technical processes, pp. 1-3, 2009.

C. R. Farrar, F. M. Hemez, D. D. Shunk, D. W. Stinemates, B. R. Nadler y J. J. Czarnecki, A review of structural health monitoring literature: 1996-2001, Los Alamos, 2004.

D. A. Tibaduiza Burgos, Design and validation of a structural health monitoring system for aeronautical structures, Barcelona, 2012.

D. A. Tibaduiza, M. A. Torres Arredondo, L. E. Mujica, J. Rodellar y C. P. Fritzen, «A study of two unsupervised data driven statistical methodologies for detecting and classifiying damages in structrual health monitoring,» Mechanical Systems and Signal Processing, vol. 41, pp. 467-484, 2013.

D. G. Perera y K. F. Li, «Embedded hardware solution for principal component analysis,» Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference, pp. 730-735, 2011.

D. Giraldo, A structural health monitoring framework for civil structures, Saint Louis, 2006.

F. Gharibnezhad, L. E. Mujica y J. Rodellar, «Applying robust variant of Principal Component Analysis as a damage detector in the presence of outliers,» Mechanical Systems and Signal Processing, vol. 50, pp. 467-479, 2015.

Fraunhofer-Institut für Silicatforschung ISC, Piezoelectric transducers for structural health monitoring, Fraunhofer ISC.

G. Dib, R. Lattrelf, L. Udpa y G. Yang, «A wireless sensor node for structural health monitoring using guided wave testing,» Materials Evaluation, vol. 71(10), pp. 1232-1241, 2013.

G. Manson, K. Worden, K. Holford y R. Pullin, «Visualisation and dimension reduction of acoustic emission data for damage detection,» Journal of Intelligent Material Systems and Structures,, vol. 12(8), pp. 529-536, 2001.

H. Sohn, C. R. Farrar, N. F. Hunter y K. Worden, «Structural health monitoring using statistical pattern recognition techniques,» Journal of dynamic systems, measurement, and control, vol. 123(4), pp. 706-711, 2001.

Hardkernel co., Ltd., «Magazine Odroid,» Enero 2015. [En línea]. Available: http://magazine.odroid.com/assets/201401/pdf/ODROID-Magazine-201401.pdf. [Último acceso: Diciembre 2014].

J. Camacho Navarro, Sistema Experto para la Monitorización de Salud Estructural mediante el Reconocimiento de Patrones: Adaptación y Validación Numérica, Bucaramanga, 2010.

J. M. Caicedo, «Structural Health Monitoring for flexible Covil Structures,» Washington University, Department Of Civil Engineer Doctoral Thesis.

J. Pavlus, «MIT Technology Review,» 2013 Abril 2013. [En línea]. Available: http://www.technologyreview.com/view/514036/beaglebone-black-a-makers-dream/. [Último acceso: Agosto 2014].

J. ZHANG, S. ZHU, Y. XU y Z. CHEN, «SHM-based Correlation Study of Trainload-induced Response in Tsing Ma Bridge,» 14 Asia Pacific Vibration Conference , pp. 113-122, 2011.

K. H. Esbensen, D. Guyot, F. Westad y L. P. Houmolle, Multivariate data analysis in practice: an introduction to multivariate data analysis and experimental design, camo, 2002.

K. Varmuza y P. Filzmoser, Introduction to multivariate statistical analysis in chemometrics, Boca raton: CRC Press-Taylor & Francis Group, 2009.

L. E. R. J. F. A. &. G. A. Mujica, «Q-statistic and T2-statistic PCA-based measures for damage assessment in structures,» Structural Health Monitoring, vol. 10(5), pp. 539-553, 2010.

L. Świrski, A. Bulling y N. Dodgson, «Robust real-time pupil tracking in highly off-axis images,» Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA, pp. 173-176, 2012.

M. Cammarata, P. Rizzo, D. Dutta y H. Sohn, «Application of principal component analysis and wavelet transform to fatigue crack detection in waveguides,» Smart structures and systems, vol. 6, nº 4, pp. 349-362, 2010.

M. J. Simões Ferreira dos Santos , Ondas ultra-sonoras guiadas na caracterização e controlo não destrutivo de materiais, Coimbra, 2004.

M. L. Ruiz Ordóñez, Multivariate statistical process control and case-based reasoning for situation assessment of sequencing batch reactors, Girona, 2008.

M. Lallart, D. Guyomar y T. Monnier, «Low-power computation methods and self-powered systems for Structural Health Monitoring techniques using Lamb waves for embedded sensing,» Shock and Vibration, vol. 19, pp. 867-877, 2012.

N. A. Leyla, E. Moulin, J. Assaad, F. Benmeddour, S. Grondel y Y. Zaatar, «Application of piezoelectric transducers in structural health monitoring techniques,» Advances in Piezoelectric Transducers, pp. 87-104, 2011.

O. Bennouna y J. P. Roux, «Real Time Diagnosis & Fault Detection for the Reliability Improvement of the Embedded Systems,» Journal of Signal Processing Systems, vol. 73, nº 2, pp. 153-160, 2013.

P. AVIATABLE, «Experimental Modal Analysis (A Simple Non-Mathematical Presentation),» Modal Analysis and Control Laboratory, Sound and Vibration Magazine, University Of Massachusetts.

Python Software Foundation, «Python,» 2001-2015. [En línea]. Available: https://www.python.org/.

R. Villamizar, J. L. Quiroga, J. Camacho, L. E. Mujica y M. L. Ruiz, «Structural Damage Detection Algorithm Based on Principal Component Indexes and Embedded on a Real Time Platform,» EWSHM-7th European Workshop on Structural Health Monitoring, pp. 1553-1560, 2014.

S. Lu, Q. He, F. Hu y F. Kong, «Sequential Multiscale Noise Tuning Stochastic Resonance for Train Bearing Fault Diagnosis in an Embedded System,» Instrumentation and Measurement, IEEE Transactions, vol. 63, nº 1, pp. 106 - 116, 2013.

S. Yuan, Y. Xu y G. Peng, «New Developments in Structural Health Monitoring Based on Diagnostic Lamb Wave,» J. Mater. Sci. Technol., vol. 20(05), pp. 490-496, 2004.

T. Smith, «The Register,» 11 Junio 2013. [En línea]. Available: http://www.theregister.co.uk/2013/06/11/review_beagleboard_beaglebone_black/. [Último acceso: Agosto 2014].

V. Giurgiutiu, Structural health monitoring: with piezoelectric wafer active sensors, Academic Press-Elsevier, 2007.

W. J. Staszewski, S. Mahzan y R. Traynor, «Health monitoring of aerospace composite structures–Active and passive approach,» Composites Science and Technology, vol. 69(11), pp. 1678-1685, 2009.

Y. Y. Lim, S. Bhalla y C. K. Soh, «Structural identification and damage diagnosis using self-sensing piezo-impedance transducers,» Smart Materials and Structures, vol. 15(4), p. 987, 2006.

ANEXOS

ANEXO A. MANUAL DE USUARIO PARA EJECUCION DEL CODIGO EN PYTHON SOBRE PLATAFORMAS EMBEBIDAS

1. Descripción General

La presente investigación permite, mediante el uso de un equipo de cómputo y de sistemas embebidos, la identificación de los índices de detección de daño (T^2 y Q-estadístico) a través de un código de ejecución programado en Python. Esto a partir del desarrollo de análisis de componentes principales para la detección de fallas en estructuras.

2. Requerimientos previos

Para poder realizar la implementación del algoritmo de salud estructural en los sistemas de cómputo, se necesitan ciertos requerimientos para cada tarjeta y computador:

2.1. Computador Personal

Requerimientos de Software:

- Ubuntu con versiones mayores o iguales a la 12.04.

2.2. BeagleBone Black

Requerimientos para conexión:

- Cable de conexión USB a mini USB
- Cable de conexión Ethernet
- Router

Requerimientos de Software:

- Python 2.7.6

Librerías de Python:

- Numpy
- StringIO
- Os

2.3. Odroid-U3

Requerimientos para conexión:

- Cable de conexión HDMI tipo A a mini HDMI tipo D
- Cable de conexión USB a mini USB
- MicroSD 8GB UHS-1 con U Linux
- Cable de alimentación 5V/2A
- Cable de conexión Ethernet
- Pantalla con puerto HDMI o entrada VGA.
- Mouse USB
- Teclado USB
- Router

Requerimientos de software:

- Python 2.7.6

Librerías de Python:

- Numpy
- StringIO
- Os

3. Comandos

- ***Ctrl+Alt+T:*** Este atajo de teclado permite abrir un terminal, el cual será nuestro espacio de trabajo y comunicación con los sistemas embebidos.
- ***cd****:* Se usa para cambiar el directorio de trabajo actual indicando a cual se desea ir. Por ejemplo: *root@beaglebone:~#* ***cd /ubicación deseada***, cd también puede usarse de la siguiente manera *cd..*, para cambiar la ubicación actual subiendo un nivel.

- ***ls**:* Este comando es bastante útil, ya que con él se puede generar una lista del contenido que se encuentra en el directorio actual, *root@beaglebone:/# ls*, si se desea es posible obtener más información del contenido del directorio adicionando *-l*, *root@beaglebone:/# ls -l*.
- ***mkdir**:* Se usa para crear un nuevo directorio o carpeta. *root@beaglebone:~#* ***mkdir myProject***
- ***nano**:* Se usa para ver el contenido de un archivo y editarlo*. root@beaglebone:~#* ***nano hello.txt***
- ***rm**:* Se usa para eliminar archivos *root@beaglebone:~/myProject#* ***rm saludos.txt*** , o para eliminar directorios agregando –r, *root@beaglebone:~#* ***rm -r archive.***
- ***opkg**:* Es un comando con varias funcionalidades, ya que con él se pueden descargar paquetes disponibles (*opkg update)*, actualizar software (*opkg upgrade)*, instalar un software determinado (*opkg install nombredelarchivo)* y actualizar software específicos *opkg upgradenombredelarchivo.*
- ***Ifconfig***: Es un comando que se usa para configurar las interfaces de red residentes en el kernel, si no se le da ningún argumento muestra el estado de las interfaces activas en el momento; si se le da un solo argumento de interfaz, muestra el estado de esa interfaz solamente. Es normalmente usado para conocer la dirección IP del servidor.
- ***ssh***: Es un programa que permite acceder a otro ordenador a través de una red, ejecutar comandos en la máquina remota y mover ficheros entre dos máquinas. Para acceder a una maquina remota, se debe conocer su nombre de usuario, la dirección IP con la cual se va a trabajar y una clave para terminar de establecer la conexión:

  ssh sistemaremoto@direcciónIPsistemaremoto.
- ***scp***: Es un comando que copia archivos o directorios entre un sistema local y un sistema remoto, o entre dos sistemas remotos. El comando scp utiliza ssh para la transferencia de datos. Por lo tanto, el comando scp usa la misma autenticación. Con este comando se puede, copiar un archivo o

directorio del sistema local a un sistema remoto y copiar un archivo o directorio de un sistema remoto a su sistema local.

Se puede usar de las siguientes maneras:

- Desde el sistema local enviar archivos al sistema remoto:

scp archivoaenviar sistemaremoto@direcciónIPsistemaremoto:/direcciónsistemalocal

- Desde el sistema remoto enviar archivos al sistema local:

scp archivoaenviar sistemalocal@direcciónIPsistemalocal:/direcciónsistemaremoto

- Desde el sistema local pedir archivos al sistema remoto:

scp sistemalocal@direcciónIPsistemalocal:/direcciónarchivosistemaremoto/archivoaenviar /direcciónsistemalocal

- Desde el sistema remoto pedir archivos al sistema local:

scp sistemaremoto@direcciónIPsistemaremoto:/direcciónarchivosistemalocal/archivoaenviar /direcciónsistemaremoto

- ***Python:*** Este comando es usado para abrir el programa o entorno interactivo de lenguaje de programación, Python.

4. Conexión y almacenamiento de datos.

Para poder ejecutar el algoritmo experto, es necesario realizar la conexión de los sistemas embebidos con un computador o conectarlo de tal manera que funcione como un equipo de cómputo local; cada una de las tarjetas tiene un tipo de conexión diferente:

4.1. BeagleBone Black:

- Para realizar la conexión, el computador debe estar funcionando con el sistema operativo Ubuntu igual o superior a la versión 12.04.

- La tarjeta se debe conectar a uno de los puertos USB del computador y a su puerto mini USB por medio del cable USB a mini USB; ésta será la fuente de alimentación para el funcionamiento del sistema embebido.
- Después de que la tarjeta se encuentra conectada, se abre un terminal (Ctrl+Alt+T) y se ingresa a ella por medio del comando *ssh*, en este caso el sistema remoto es "*root"* y su dirección IP es "*192.168.7.2*", lo que significa que el comando debe ser ejecutado de la siguiente manera: *ssh root@192.168.7.2*. Una vez ejecutado el comando, el sistema pedirá una clave de acceso, a lo cual se presionará la tecla *enter* para poder acceder.
- Una vez dentro del sistema embebido, se crea una carpeta para guardar los datos necesarios para las pruebas, esto con el comando *mkdir,* así: *mkdir Datos_Tuberia*
- Se abre otro terminal y por medio de los comandos *ls* y *cd* se busca en el sistema local la ubicación del algoritmo y los archivos de texto que se deben enviar[9].
- Para enviar los archivos es necesario usar el comando *scp*, teniendo en cuenta que los archivos solamente pueden ser enviados uno por uno.

4.2. Odroid-U3:

La conexión del sistema embebido se puede realizar de dos métodos. En el primero método, se trabaja el sistema embebido como un sistema de cómputo independiente; en el segundo método, se conecta la tarjeta a un sistema de cómputo local y se maneja por medio de comandos de terminal.

Método 1:

- Se conecta el sistema embebido a una pantalla o monitor por medio del cable HDMI a mini HDMI.

[9] Los archivos necesarios se describen en la sección 5 del Manual de Usuario.

- Se conectan en los puertos USB un mouse y un teclado; si se tiene la disponibilidad, se puede conectar el cable Ethernet a un router para, de esta manera, brindar internet a la tarjeta.
- Se conecta el cable de 5V al plug de alimentación para energizar la tarjeta. Después de esto, el sistema embebido encenderá automáticamente.
- Una vez encendida la tarjeta ésta mostrará la pantalla de inicio con un cuadro de Usuario y Contraseña para poder iniciar sesión, en este caso el usuario es “odroid” y la contraseña es “odroid”, obteniendo de esta manera la funcionalidad completa de un sistema de cómputo.
- Los archivos pueden ser introducidos al sistema embebido por medio de una memoria USB, guardándolos todos en una misma carpeta.

Método 2:

- Se conecta el sistema embebido al computador por medio del cable USB a mini USB.
- Se conecta el cable de 5V al plug de alimentación para energizar la tarjeta.
- Se conecta el cable Ethernet al router y al sistema embebido, para que de esta manera la tarjeta comparta con el computador la IP de la red.
- Después de que la tarjeta se encuentra conectada, se abre un terminal (Ctrl+Alt+T) y se ingresa a ella por medio del comando *ssh*, en este caso el sistema remoto es “*odroid”* y su dirección IP es la del internet.

  Nota: En caso de desconocer la dirección IP de la tarjeta, en una terminal aparte escribir *ifconfig* para conocerla.

5. Ejecución del algoritmo

Ya que la ejecución del algoritmo se hace desde el terminal de comandos, este procedimiento es el mismo para ambas tarjetas, solamente se debe estar en la ubicación donde se encuentran los archivos necesarios para el procesamiento.

- Para poder ejecutar el programa, se debe estar en la carpeta o directorio donde se encuentran los archivos de la estructura a estudiar y que son necesarios para el procesamiento. Estos archivos son:
    * “MU_G.txt”: Vector de medias de la matriz de casos sin daños, usada para modelado de la estructura.
    *”latent.txt”: Vector de valores propios o de principales componentes del proceso de modelado.
    *”coeffp.txt”: Vectores propios del proceso de modelado.
    *”desv_G.txt”: Vector de desviaciones estándar por piezoeléctrico utilizado en el proceso de modelado.
    *”act_train2D.txt”: Matriz de entrenamiento o de casos sin daños.
    *”act_test2D.txt”: Matriz de validación o de casos con daños.
    *”Nsensores.txt”: Numero de piezoeléctricos o sensores usados.
    *”NormalizV4.py”: Algoritmo experto.

- Una vez adentro, se usa el comando *nano* para poder abrir el programa NormalizV4, verificar los archivos que queremos procesar y hacer cambios en él de la siguiente manera: *nano NormalizV4*. Una vez usado el comando, se muestra el código para SHM en donde se cambiará el nombre de la matriz que se quiere procesar, que para el caso son: act_train2D (para verificar la valores de modelado) o act_test2D (para obtener los resultados de validación). Al finalizar los cambios, se pulsa *Ctrl+x* y se le da *Y* para salir y guardar lo realizado.
- Ingresar a Python, por medio del comando *Python.* Esto abrirá el entorno interactivo del programa.
- Una vez adentro, se importa el algoritmo y las librerías necesarias para su ejecución, por medio de los siguientes comandos, primero *import NormalizV4*, y luego *from NormalizV4 import **
- Luego de lo anterior, se usa el comando *Normaliz(Data, MU_G, desv_G, coeffp, latent, Nsensores)*, lo que hará que se ejecute el algoritmo de salud

estructural. Una vez terminado este proceso, los datos de tiempo e índices de daño serán encontrados como archivos *.txt* en la carpeta en donde se ejecutó el algoritmo. Estos archivos pueden ser devueltos al sistema de cómputo a través del comando *scp.*

ANEXO B PRUEBAS DE VALIDACIÓN

Se realizaron pruebas de validación del código experto basado en PCA sobre la tarjeta Odroid-U3 para una nueva estructura. En la *Figura 24* se muestra el esqueleto (‘Skeleton’) del ala de un avión sometido a pruebas experimentales a través de la excitación de piezoeléctricos, esto con el fin de modelar el tramo de la estructura y validar los datos obtenidos a partir del análisis de daños presentes.

Figura 24. Esqueleto de Ala de Avión

Como se observa en la *Figura 24*, en una reducida área del esqueleto del ala de avión fueron adicionadas masas (representación de los daños) junto con piezoeléctricos para la caracterización del tramo de la estructura. En total fueron 4 masas adicionadas, es decir, 4 daños sobre el tramo, y 9 transductores piezoeléctricos ubicados en la misma. La estructura de la matriz de datos se organiza de la forma como se explicó en la *sección 2.2.2* y sus dimensiones se describen como sigue:

[750x80397], que corresponden a:

- 150 experimentos sin daños.
- 4 daños en la estructura, cada uno con 150 experimentos.

- 1 piezo-actuador usado en la estructura.
- 9 piezosensores utilizados, 8933 muestras por piezo-sensor.

Nota: El modelo para la presente estructura fue creado con 20 componentes principales.

RESULTADOS PARA EL ESQUELETO DE ALA DE AVIÓN

Las pruebas de ejecución del código elaborado en el presente trabajo de investigación con los datos del esqueleto del ala de avión, fueron realizadas únicamente sobre la tarjeta Odroid-U3 a fin de que ésta es la tarjeta sobre la cual se centra el presente trabajo, así como el proyecto global realizado para Colciencias: “MONITORIZACIÓN Y DETECCIÓN DE DEFECTOS EN ESTRUCTURAS USANDO ALGORITMOS EXPERTOS EMBEBIDOS”. Los resultados obtenidos se muestran a continuación:

Tabla 9. Tiempo de procesamiento de datos 'Skeleton'

Tiempo procesamiento datos [s]		
		Ala Skeleton
Odroid-U3	Promedio por vector	3.51746838808
	Total de vectores	2638.101291061

PORCENTAJE DE MEMORIA RAM

Debido a la cantidad de datos procesados, la tarjeta Odroid-U3 consume más recursos de memoria para la ejecución del algoritmo experto. La *Tabla 10* permite observar dichos porcentajes.

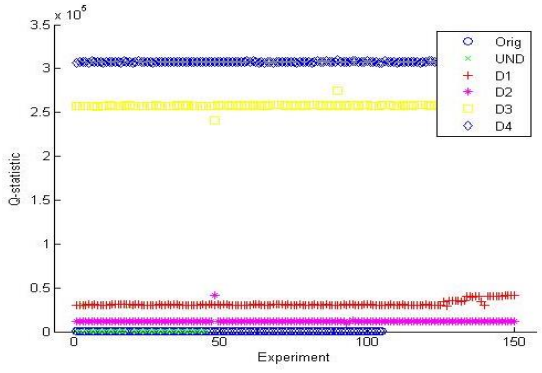
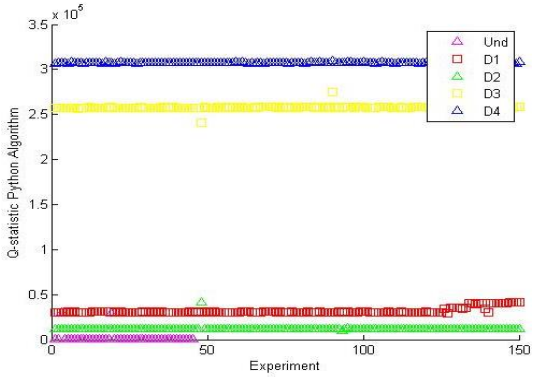
Tabla 10. Porcentajes de RAM ´Skeleton´

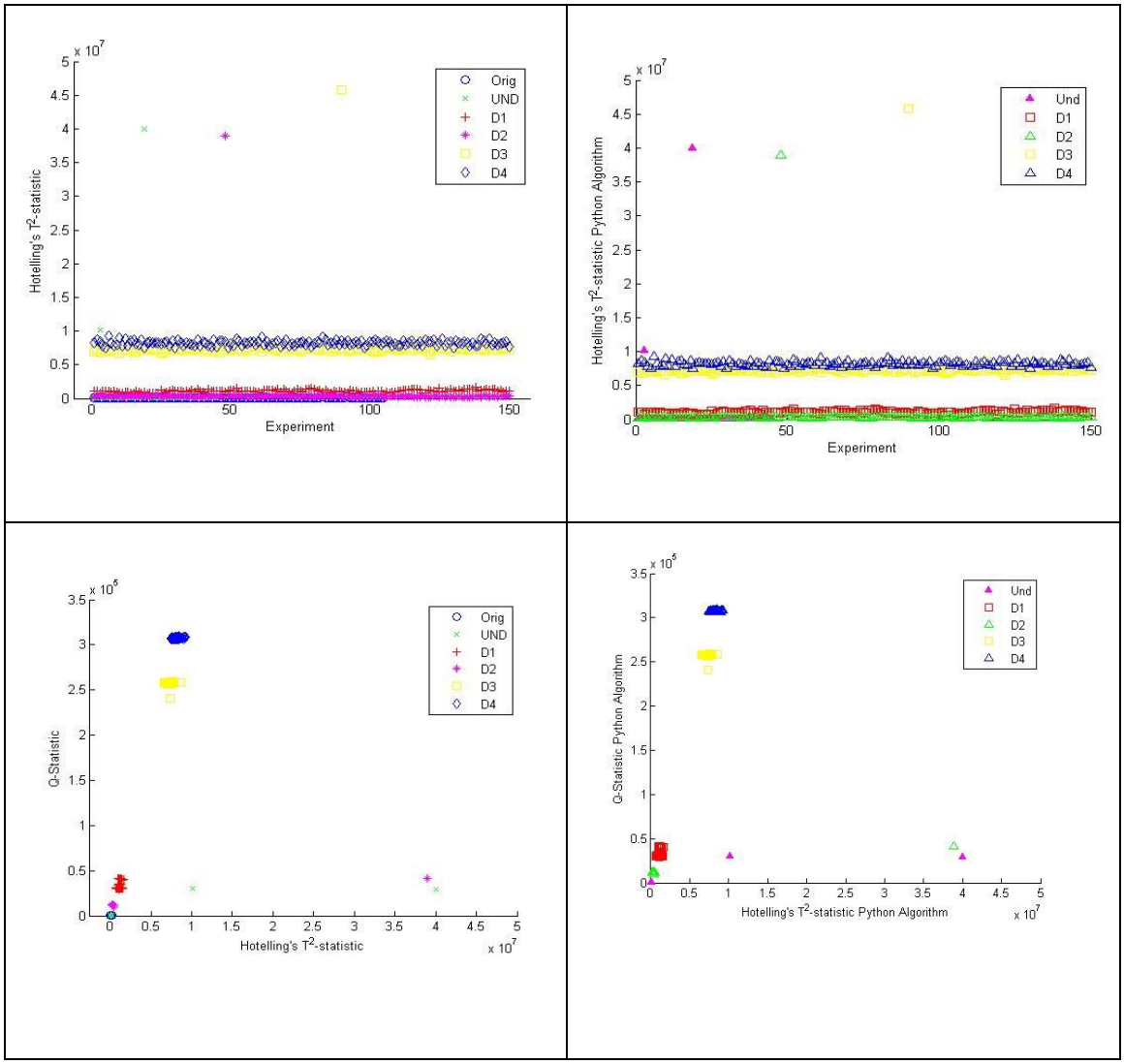
	% Memoria al importar	**% Memoria al ejecutar**
Odroid-U3	47%	50,9%

COMPARACION DE ALGORITMOS

Nuevamente se realiza la comparación grafica de resultados para determinar la existencia de daños presentes en la estructura de estudio y definir la precisión del código elaborado en Python respecto al realizado en Matlab. Los resultados se muestran como sigue:

Tabla 11. Comparación de gráficas de datos procesados en Matlab y Python: Esqueleto del Ala de Avión

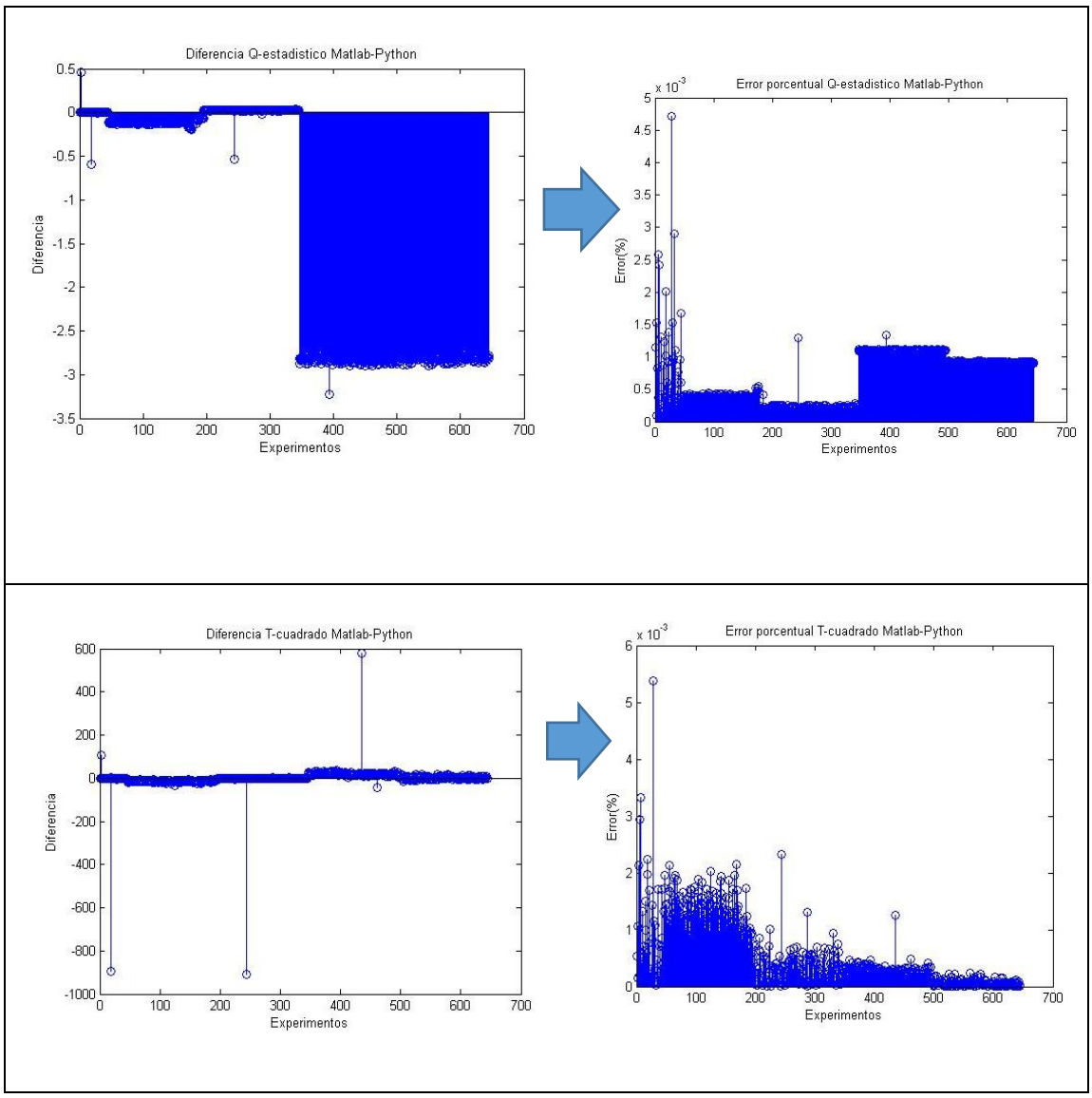
Gráficas de datos procesados - Skeleton Ala de Avión	
Procesamiento en Matlab	Procesamiento en Python
	



ERROR EN LA COMPARACION DE DATOS

Para verificar la precisión de los resultados obtenidos en Python con los obtenidos en Matlab, no solo basta con el análisis grafico sino también con la diferenciación y cálculo del error de ambos grupos de datos de cada lenguaje de programación. De este modo, la *Figura 25* muestra dicho análisis y permite concluir que el error de la precisión de datos es menor al 0.007%.

Figura 25 Error porcentual para el cálculo de índices de daño Ala ‘Skeleton’


Diferencia Q-estadistico Matlab-Python
Diferencia
Experimentos
Error porcentual Q-estadistico Matlab-Python
x 10^-3
Error(%)
Experimentos
Diferencia T-cuadrado Matlab-Python
Diferencia
Experimentos
Error porcentual T-cuadrado Matlab-Python
x 10^-3
Error(%)
Experimentos

ANEXO C. PRECISION DEL CALCULO DE INDICES DE DAÑO EN PYTHON

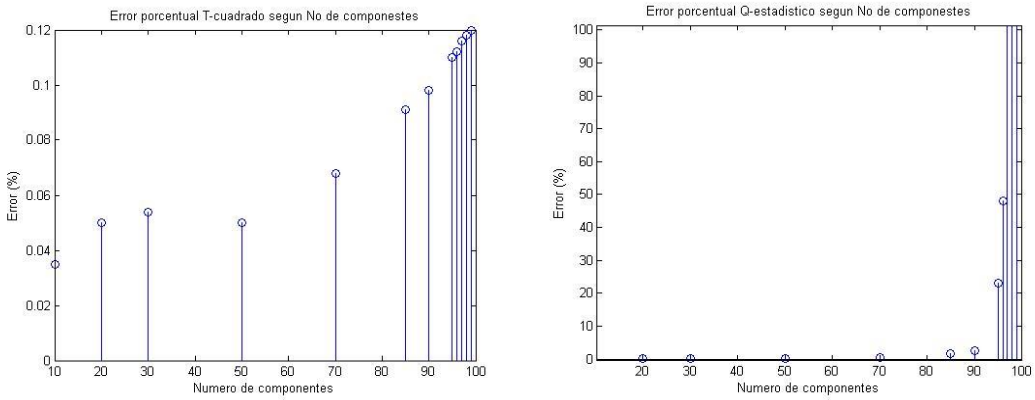
Cada software de programación opera los datos en forma diferente, por lo que la precisión con la que se obtiene un dato es variable. Justamente, el cálculo del error y la diferencia de índices de daños de las estructuras estudiadas en el presente trabajo de grado, fue presentado en el Capítulo 4 para un número específico de componentes principales (10 comp-Alabe de Turbina de avión, 15 comp-Tubería Acero Carbono, 20 comp- Ala Skeleton). Sin embargo, a medida que se aumenta el número de componentes, las dimensiones de los elementos de salida del modelo cambian y por tanto la validación también, haciendo sustancial el análisis del uso de cifras decimales que el software de programación (Python) utiliza para obtener los índices de daño.

La *Figura 26* muestra la variación del error a medida que se aumenta el número de componentes principales para obtener los índices de daño (T^2 y Q-estadístico)[10], cada medida corresponde al índice con máximo valor de error, obtenido de acuerdo al procesamiento de cada componente analizada. La precisión de Python se mide respecto a los índices de daño calculados en Matlab[11].

[10] Las imágenes mostradas corresponden a la validación de datos de la Tubería Acero-Carbono: 4 Daños sobre la estructura, cada uno con 100 experimentos; dos piezoeléctricos usados, 2007 muestras por sensor. Dichos datos fueron suministrados por el Laboratorio de Investigación de la UIS, sede Guatiguara.

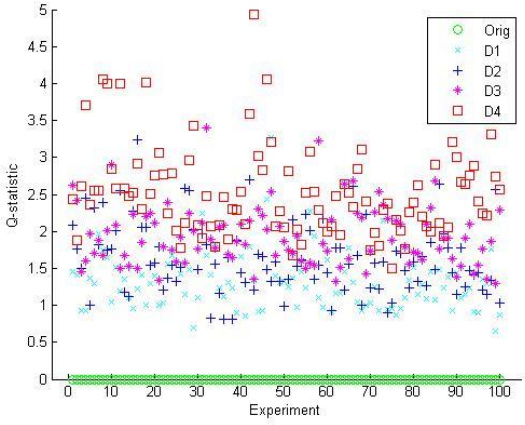
[11] Tanto Python como Matlab se basan en la norma IEEE-754.

Figura 26 Errores porcentuales de Índices de Daño Vs. Número de componentes



El error porcentual es mínimo para el cálculo del índice T^2, pero dicho error se acentúa cuando se calcula Q-estadístico. Esto se explica dado a que los valores del índice Q, para los casos sin daños (*Figura 27* - Verde), son muy cercanos a cero y la precisión de Python respecto a Matlab varía por lo general a partir de la cuarta cifra decimal, de este modo, existen diferencias en cifras decimales pero el valor entero se mantiene.

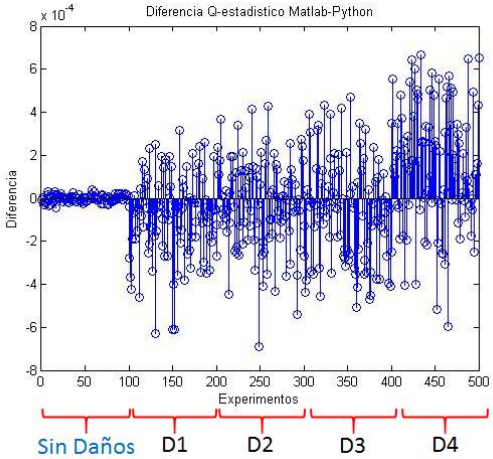
Figura 27 Tabulación Grafica Índice Q-estadístico

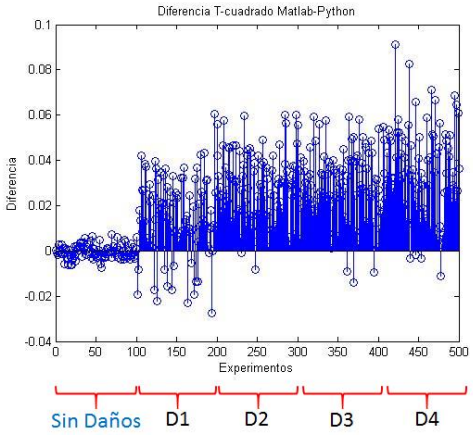


La media cuadrática RMS de las diferencias entre los resultados de índices de daño de Matlab y Python determina cuan distinta puede ser la precesión de este último en el sistema embebido. Las siguientes figuras muestran los resultados de

validación para diferente número de componentes y sus respectivas medias cuadráticas calculadas.

20 Componentes Principales:

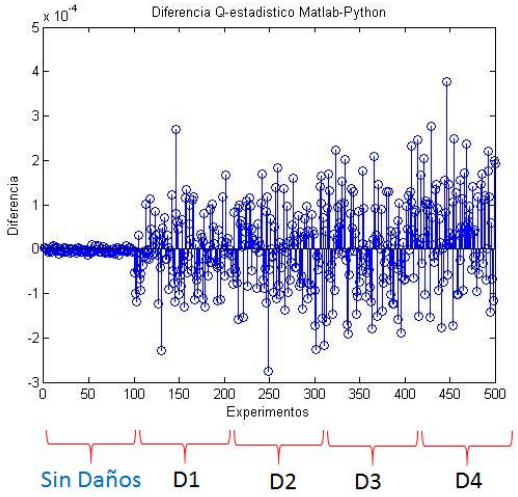


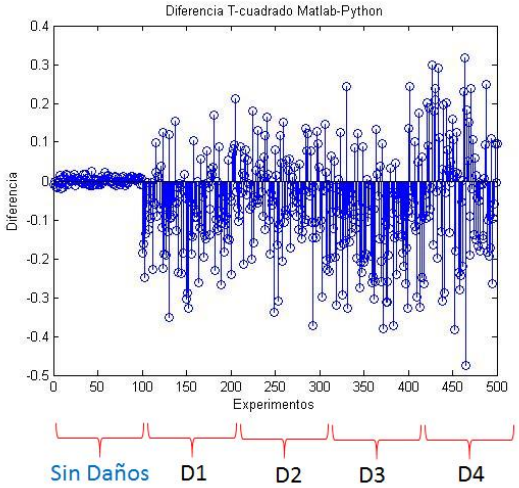


Índice Q-estadístico	
Media RMS (Sin Daños)	1.5770e-05
Media RMS (D1,…,D4)	2.5293e-04

Índice T-cuadrado	
Media RMS (Sin Daños)	0.0030
Media RMS (D1,…,D4)	0.0321

50 Componentes Principales:

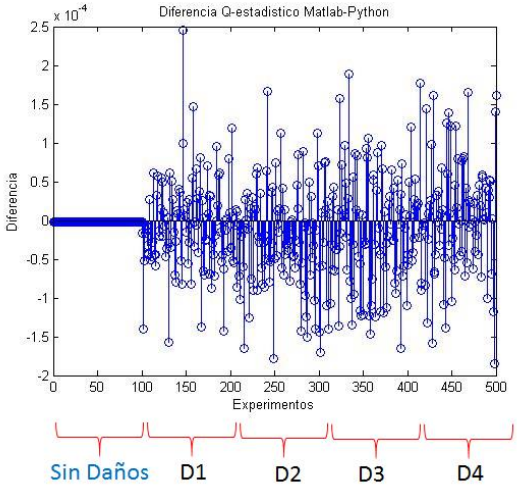


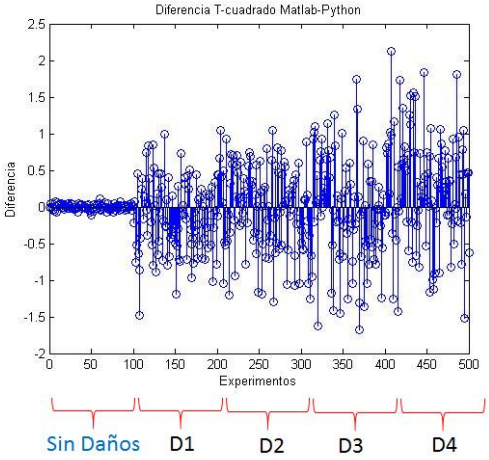


Índice Q-estadístico	
Media RMS (Sin Daños)	5.3084e-06
Media RMS (D1,…,D4)	9.6841e-05

Índice T-cuadrado	
Media RMS (Sin Daños)	0.0091
Media RMS (D1,…,D4)	0.1483

99 Componentes Principales:





Índice Q-estadístico	
Media RMS (Sin Daños)	2.2039e-06
Media RMS (D1,…,D4)	6.9631e-05

Índice T-cuadrado	
Media RMS (Sin Daños)	0.0379
Media RMS (D1,…,D4)	0.6595

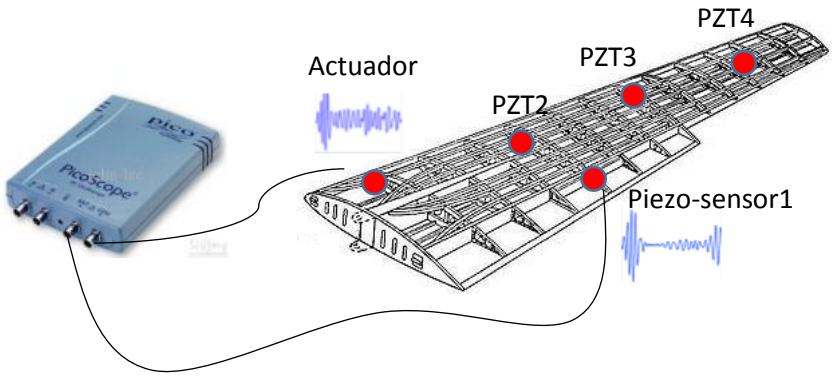
A pesar del aumento del número de componentes, la dispersión de resultados observados a partir de las medias cuadráticas es pequeña, de modo que lo anterior demuestra la no inferencia sustancial de los errores de precisión para el cálculo de los índices de daño en Python.

ANEXO D. VALIDACION DE DATOS SIN CORRELACION

El Capítulo 4 y los Anexos del presente trabajo, abarcan el análisis y resultados de datos adquiridos cuando los elementos de entrada en la etapa de validación (Vectores de casos, Medias, Desviaciones Estándar, Eigenvalores, Eigenvectores) han sido previamente correlacionados.

La correlación, en resumen, es una medida de la similitud entre dos señales, frecuentemente usada para encontrar características relevantes en una señal desconocida (sensor) por medio de la comparación con otra que sí se conoce (actuador).

Figura 28 Osciloscopio PicoScope conectado a la estructura

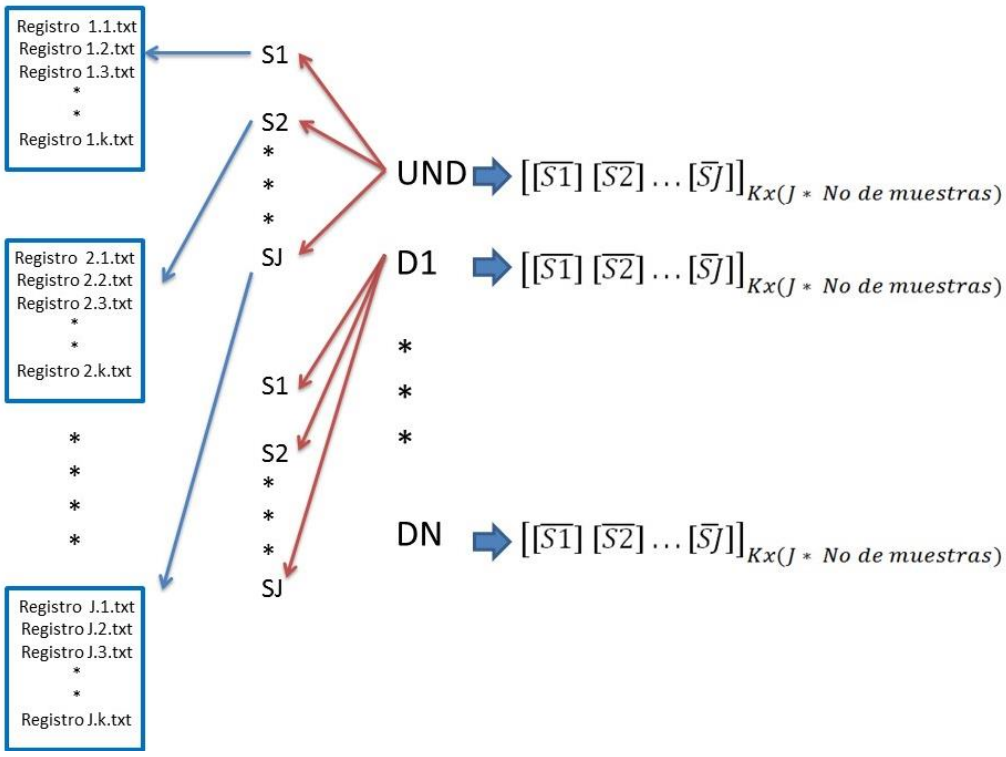


En el laboratorio, el osciloscopio PicoScope genera archivos de texto con los valores que captura. Dichos valores se organizan en tres columnas: Una de Tiempo, otra del Canal A (Actuador) y finalmente Canal B (sensor). Cabe resaltar que por cada sensor, el PicoScope genera un número significativo de archivos generados para correlacionar (eso depende de la configuración del osciloscopio en la definición del número de muestras).

Considerando que las pruebas de validación hechas en el presente trabajo de investigación son Off-line, se emuló a través de código Python la acción de un sistema de adquisición de datos para ordenar archivos generados por PicoScope y luego procesarlos, a fin de buscar daños en la estructura.

La *Figura 29* hace alusión a la forma de organización de datos y construcción de la matriz de casos con y sin daños. Los daños simulados están separadas en carpetas que los distinguen. Dentro de cada carpeta de daños están los sensores utilizados para el análisis de la estructura y dentro de cada carpeta de sensores, los registros de osciloscopio en formato tipo texto.

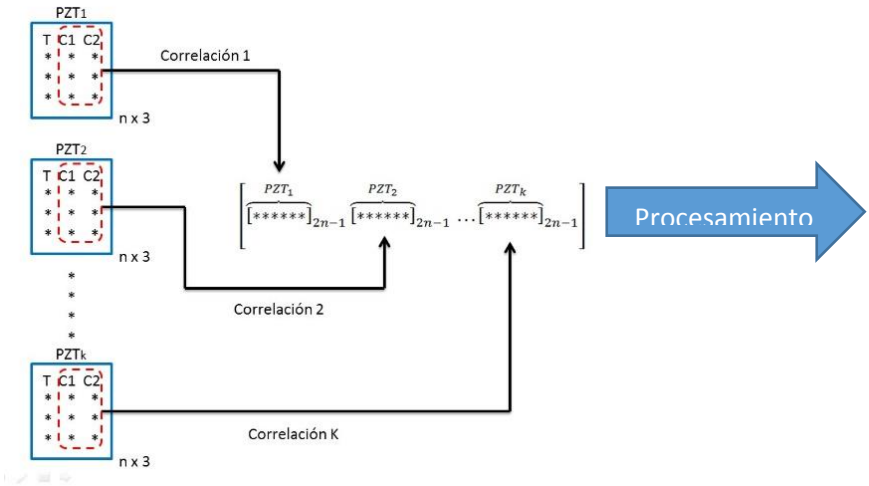
Figura 29 Emulación de sistema de adquisición de datos



En el capítulo 3.1.2.2 se explicó cuándo un vector, cuyos datos ya han sido correlacionados, está listo para procesamiento a través del algoritmo experto y

determinar sus respectivos índices de daño T y Q. Básicamente lo dicho se resumió en la siguiente figura.

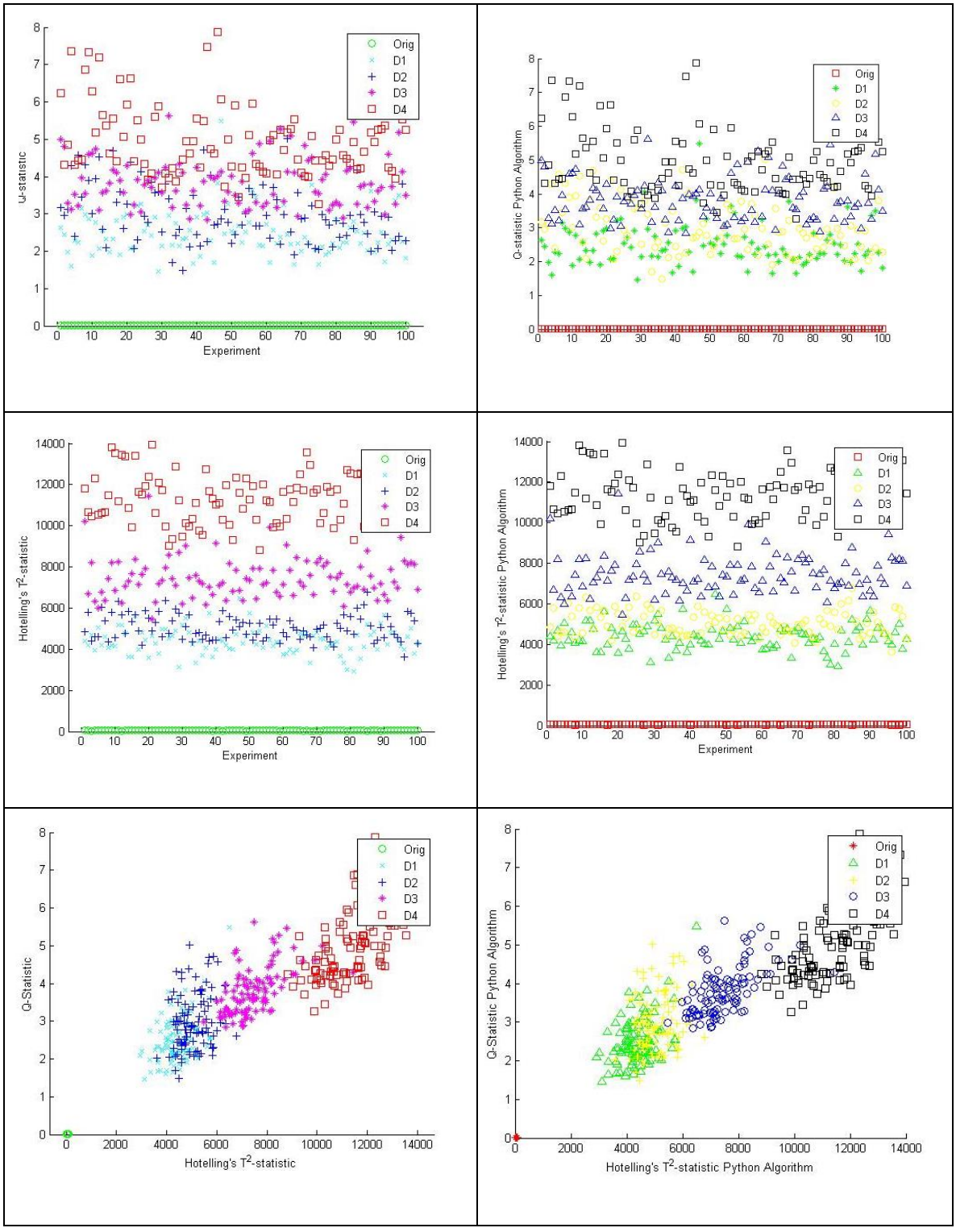
Figura 30 Correlación de registros por daño



Por ejemplo, una fila de Undamage (UND) está compuesta por la correlación de las columnas del *Registro 1.1.txt* del Sensor 1 (S1), luego se hace la correlación de las columnas del *Registro 2.1.txt* del Sensor 2 (S2), el proceso continua hasta correlacionar las columnas del *Registro J.1.txt* del *J-ésimo* sensor piezoeléctrico (de aquí el hecho de utilizar un Buffer para ir almacenando los resultados de correlación). Cuando este proceso termina, la fila o vector está completa y el procesamiento se lleva a cabo. Lo anteriormente descrito se repite para cada uno de los registros por daño.

RESULTADOS

Gráficas de datos procesados-Tubería Acero Carbono	
Procesamiento en Matlab	Procesamiento en Python


Orig
D1
D2
D3
D4
Experiment
Q-statistic
Q-statistic Python Algorithm
Hotelling's T2-statistic
Hotelling's T2-statistic Python Algorithm
Q-Statistic
Q-Statistic Python Algorithm

Se realizó la implementación del emulador del sistema de adquisición con los registros generados por osciloscopio de la estructura Tubería Acero-Carbono. Estos se describen como sigue:

- 4 Daños sobre la estructura (Representadas como fugas de aire).
- Dos sensores utilizados.
- Cada sensor con 100 Registros en archivos de texto.
- Cada archivo de texto contiene 1004 muestras.

Estos datos, al igual que los propios de la etapa de modelo, fueron suministrados por el grupo CEMOS en el Laboratorio de Investigación de la UIS, sede Guatiguara. Dicho modelo fue elaborado para 50 componentes principales. Los resultados gráficos de la implementación se muestran a continuación.

Figura 31 Diferencia y Error porcentual de datos

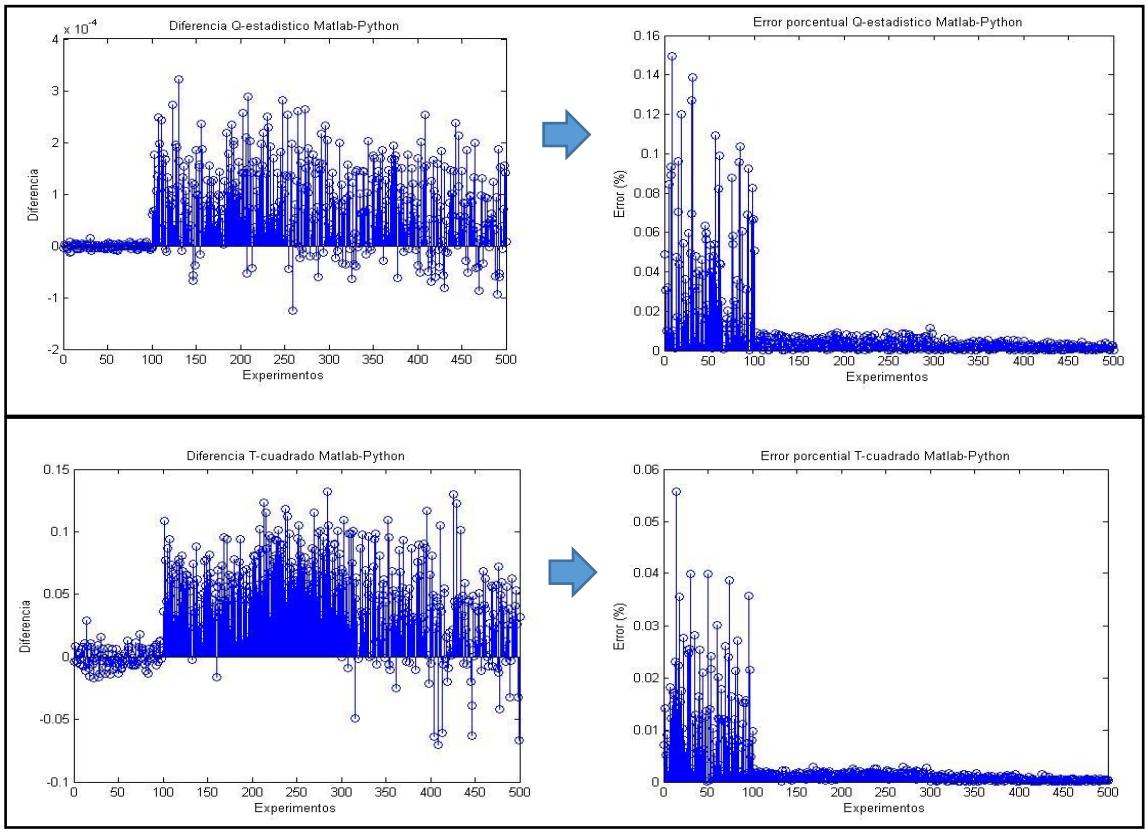


Tabla 12 Tiempo de procesamiento

		Tubería Acero-Carbono
Odroid-U3	Promedio por vector	0,189666
	Total de vectores	94,8329995

Tabla 13 Máximo pico de porcentaje de uso de memoria RAM en el procesamiento

	% Memoria al ejecutar
Odroid-U3	54,1%