



UNIVERSIDAD DE VALLADOLID

INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INFORMÁTICA

ESPECIALIDAD BIG DATA

Una propuesta de arquitectura Big Data para la prognosis de dispositivos electrónicos

Autor:

Mario Cartón González

Tutores:

**Carlos J. Alonso González
J. Belarmino Pulido Junquera**

Valladolid, 20 de Julio de 2018

TÍTULO:	Una propuesta de arquitectura Big Data para la prognosis de dispositivos electrónicos
AUTOR:	Mario Cartón González
TUTORES:	Carlos J. Alonso González J. Belarmino Pulido Junquera
DEPARTAMENTO:	Departamento de Informática

Resumen del trabajo

Los dispositivos electrónicos son una pieza clave y fundamental en la sociedad actual. En gran cantidad de escenarios es de vital importancia que estos dispositivos estén en continuo funcionamiento y la detección de posibles fallos es un factor crucial para mantener en funcionamiento el sistema en el que se integran. Los posibles fallos de estos dispositivos electrónicos normalmente suelen ser causados por fallos en los componentes que integran, por ejemplo, transistores MOSFET.

En este contexto es necesario realizar un estudio y modelado de la vida útil de los dispositivos (RUL), es decir, es necesario descubrir y aplicar técnicas que permitan predecir el tiempo de vida que queda para cada dispositivo, es lo que se conoce como prognosis. Primeramente, para realizar prognosis sobre dispositivos es necesario conocer que característica o características describen la degradación del componente, a partir de aquí se pueden aplicar las distintas técnicas de prognosis. En este trabajo se parte del conocimiento de que se pueden aplicar técnicas como la optimización bayesiana, el filtro de Kalman extendido o el filtro de partículas, pero se exponen otros métodos como pueden ser la media horizontal o un simple ajuste por mínimos cuadrados.

Finalmente, este tipo de métodos pueden llevarse a gran escala por empresas fabricantes de dispositivos. Parece que la solución más razonable para estas empresas es tener una plataforma Big Data que sea capaz de coordinar la gran cantidad de información que se genera en las experimentaciones y testeos de una gran cantidad de dispositivos.

Abstract

Electronic devices are fundamental elements in our society. Some scenarios have an important requirement, the device need to be online full time. Faults in electronic devices are usually caused by the components they have made of, for example, MOSFET transistors.

In this context, it is necessary to carry out a study and to model the remaining useful life of the device (RUL), that is, it's necessary to discover and to apply techniques to predict remaining useful life of devices. First, in order to perform prognosis, it's necessary to know degradation pattern of the component, from which different forecasting techniques can be applied. This work starts from previous experiences applying techniques such as Bayesian optimization, the extended Kalman filter or the particle filter. We propose in this

work to try other techniques other methods such as the horizontal average or a simple regression model using least squares.

Finally, this type of methods could be applied at large scale by device manufacturers. It seems that the most reasonable solution for these companies is to have a Big Data platform that should be able to coordinate the large amount of information that is generated in the experimentation and testing process for very large number of devices.

Agradecimientos

A mi madre, por ser mi primer referente en el mundo de la informática. Siempre serás fuente motivación e inspiración, te echo de menos...

A mi padre, a mi hermano, y al resto de mi familia por la preocupación y el ánimo durante todo el tiempo que llevado la realización del trabajo.

Gracias a Carlos y a Belarmino por ayudarme a lo largo de todo este camino, he aprendido mucho con vosotros. Agradecer a Pedro que nos ayudase a enfocar el cálculo de los intervalos de confianza y nos planteara la reparametrización del modelo para conseguir un modelo lineal; esa parte se la debo a él.

Finalmente, gracias a Isabel, José Luis y el resto del equipo de trabajo por el apoyo, y por darme todas las facilidades posibles para que pudiese acabar este trabajo fin de máster.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Objetivos	1
1.3. Planificación del trabajo	2
1.3.1. Planificación inicial	2
1.3.2. Planificación final	2
1.4. Estructura de la memoria	3
2. Contexto del proyecto: Prognosis de transistores MOSFET	5
2.1. Introducción	5
2.2. Transistores MOSFET	5
2.2.1. ¿Qué son?	5
2.2.2. La influencia de la temperatura en MOSFET	8
2.3. Prognosis. Estado del arte	10
3. Fase experimental. Prognosis	13
3.1. Introducción	13
3.2. Datos. Exploración	13
3.3. Transformación de los datos	18
3.3.1. Muestreo de los <i>transient</i>	22
3.3.2. Cálculo y normalización de la resistencia	25
3.3.3. Media por minuto	29
3.4. Selección de tests	31
3.5. Modelado	33
3.5.1. Media horizontal	34
3.5.2. Ajuste por mínimos cuadrados	34
3.5.3. Filtro de Kalman Extendido	35
3.5.4. Ajuste por mínimos cuadrados para el modelo linealizado	36
3.5.5. Media horizontal para el modelo linealizado	36
3.6. Resultados de la estimación de <i>RUL</i>	37
3.7. Discusión de resultados	58
4. Estudio de una posible plataforma Big Data	65
4.1. Especificación de los requisitos del sistema	66
4.1.1. Roles en el sistema	66
4.1.2. Objetivos	66

4.1.3.	Requisitos funcionales	67
4.1.4.	Requisitos no funcionales	68
4.2.	Arquitecturas Big Data	68
4.2.1.	Arquitectura tradicional	69
4.2.2.	Arquitectura <i>CEP</i> (<i>Complex Event Processing</i>)	71
4.2.3.	Arquitectura lambda	72
4.2.4.	Arquitectura kappa	74
4.2.5.	Arquitectura unificada	75
4.3.	Propuesta de plataforma Big Data	76
4.4.	Diseño de la plataforma	77
4.4.1.	Fuente de los datos	77
4.4.2.	Ingesta de datos	78
4.4.3.	Diseño del <i>Data Lake</i>	79
4.4.4.	Procesado y modelado de datos	80
4.4.5.	Automatización de las tareas	81
4.4.6.	Capa de servicio de datos	82
4.5.	Implementación de la plataforma	83
4.6.	Arquitectura final	83
5.	Conclusiones	85
	Bibliografía	87
A.	Apéndice: Contenido del CD	91

Índice de figuras

2.1.	Estructura MOSFET de canal N	6
2.2.	Estados del transistor	7
2.3.	Representación del transistor MOSFET	7
2.4.	Ejemplo de MOSFET con carcasa <i>TO-220</i>	8
2.5.	Variación de la resistencia respecto de la temperatura	8
2.6.	Variación de la intensidad respecto del potencial aplicado en la puerta y la temperatura	9
2.7.	Comportamiento de I_D respecto de V_{GS} , V_{DS} y la temperatura	9
2.8.	Control del envejecimiento por sobrecarga térmica	11
2.9.	Aparición de huecos en <i>die-attach degradation</i>	11
3.1.	Representación de los valores de control de temperatura en <i>pwmTempControllerState.csv</i> para el <i>Test 36</i> del conjunto de datos.	16
3.2.	Representación de los valores de control de temperatura tras la fase de calentamiento en <i>pwmTempControllerState.csv</i> para el <i>Test 36</i> del conjunto de datos. (Zoom sobre la <i>Figura 3.1</i>)	16
3.3.	Representación de los valores del voltaje aplicado en la puerta y en el dispositivo en <i>pwmTempControllerState.csv</i> para el <i>Test 36</i> del conjunto de datos.	16
3.4.	Representación de algunos datos contenidos en <i>steadyState.csv</i> para el <i>Test 36</i> del conjunto de datos.	17
3.5.	Datos del transient (resistencia, voltage, intensidad entre el <i>drenador</i> y la <i>fuelle</i> , y voltaje aplicado en la <i>puerta</i>) en el instante 1 para el <i>Test 36</i> del conjunto de datos.	19
3.6.	Datos del transient (resistencia, voltage, intensidad entre el <i>drenador</i> y la <i>fuelle</i> , y voltaje aplicado en la <i>puerta</i>) en el instante 2 para el <i>Test 36</i> del conjunto de datos.	20
3.7.	Datos del transient (resistencia, voltage, intensidad entre el <i>drenador</i> y la <i>fuelle</i> , y voltaje aplicado en la <i>puerta</i>) en el instante 3 para el <i>Test 36</i> del conjunto de datos.	21
3.8.	Resultados para la intensidad en el <i>drenador</i> y voltaje entre el <i>drenador</i> y la <i>fuelle</i> tras el muestreo de los <i>transient</i> para <i>Test 36</i>	23
3.9.	Resultados para el voltaje entre la <i>puerta</i> y la <i>fuelle</i> , voltaje de la señal aplicada, y temperaturas <i>package</i> y <i>flange</i> tras el muestreo de los <i>transient</i> para <i>Test 36</i>	24

3.10. Resultados de la normalización de la resistencia y su incremento para el <i>Test 36</i>	27
3.11. Resultados de la normalización de la resistencia y su incremento para el <i>Test 8</i>	28
3.12. ΔR para el <i>Test 36</i>	29
3.13. ΔR para el <i>Test 8</i>	30
3.14. ΔR para los <i>Tests 8, 9, 11, 12, 14</i> y <i>36</i> hasta el fin del experimento	30
3.15. ΔR para los <i>Tests 8, 9, 11, 12, 14</i> y <i>36</i> por debajo del umbral de fallo . . .	30
3.16. Intervalos de confianza para el ajuste al modelo linealizado para el <i>Test 8</i> .	37
3.17. Media horizontal: Predicción en el 64 % (t=90) de la vida del dispositivo del <i>Test 8</i>	39
3.18. Media horizontal: Predicción en el 85 % (t=119) de la vida del dispositivo del <i>Test 8</i>	39
3.19. Media horizontal: Predicción en el 95 % (t=133) de la vida del dispositivo del <i>Test 8</i>	39
3.20. Media horizontal: Predicción en el 64 % (t=63) de la vida del dispositivo del <i>Test 26</i>	40
3.21. Media horizontal: Predicción en el 85 % (t=84) de la vida del dispositivo del <i>Test 26</i>	40
3.22. Media horizontal: Predicción en el 95 % (t=94) de la vida del dispositivo del <i>Test 26</i>	40
3.23. Media horizontal: Predicción en el 64 % (t=151) de la vida del dispositivo del <i>Test 36</i>	41
3.24. Media horizontal: Predicción en el 85 % (t=201) de la vida del dispositivo del <i>Test 36</i>	41
3.25. Media horizontal: Predicción en el 95 % (t=225) de la vida del dispositivo del <i>Test 36</i>	41
3.26. Mínimos cuadrados: Predicción en el 64 % (t=90) de la vida del dispositivo del <i>Test 8</i>	43
3.27. Mínimos cuadrados: Predicción en el 85 % (t=119) de la vida del dispositivo del <i>Test 8</i>	43
3.28. Mínimos cuadrados: Predicción en el 95 % (t=133) de la vida del dispositivo del <i>Test 8</i>	43
3.29. Mínimos cuadrados: Predicción en el 64 % (t=63) de la vida del dispositivo del <i>Test 26</i>	44
3.30. Mínimos cuadrados: Predicción en el 85 % (t=84) de la vida del dispositivo del <i>Test 26</i>	44
3.31. Mínimos cuadrados: Predicción en el 95 % (t=94) de la vida del dispositivo del <i>Test 26</i>	44
3.32. Mínimos cuadrados: Predicción en el 64 % (t=151) de la vida del dispositivo del <i>Test 36</i>	45
3.33. Mínimos cuadrados: Predicción en el 85 % (t=201) de la vida del dispositivo del <i>Test 36</i>	45
3.34. Mínimos cuadrados: Predicción en el 95 % (t=225) de la vida del dispositivo del <i>Test 36</i>	45

3.35. Filtro de Kalman extendido: Predicción en el 64 % (t=90) de la vida del dispositivo del <i>Test 8</i>	47
3.36. Filtro de Kalman extendido: Predicción en el 85 % (t=119) de la vida del dispositivo del <i>Test 8</i>	47
3.37. Filtro de Kalman extendido: Predicción en el 95 % (t=133) de la vida del dispositivo del <i>Test 8</i>	47
3.38. Filtro de Kalman extendido: Predicción en el 64 % (t=63) de la vida del dispositivo del <i>Test 26</i>	48
3.39. Filtro de Kalman extendido: Predicción en el 85 % (t=84) de la vida del dispositivo del <i>Test 26</i>	48
3.40. Filtro de Kalman extendido: Predicción en el 95 % (t=94) de la vida del dispositivo del <i>Test 26</i>	48
3.41. Filtro de Kalman extendido: Predicción en el 64 % (t=151) de la vida del dispositivo del <i>Test 36</i>	49
3.42. Filtro de Kalman extendido: Predicción en el 85 % (t=201) de la vida del dispositivo del <i>Test 36</i>	49
3.43. Filtro de Kalman extendido: Predicción en el 95 % (t=225) de la vida del dispositivo del <i>Test 36</i>	49
3.44. Mínimos cuadrados con el modelo linealizado: Predicción en el 64 % (t=90) de la vida del dispositivo del <i>Test 8</i>	51
3.45. Mínimos cuadrados con el modelo linealizado: Predicción en el 85 % (t=119) de la vida del dispositivo del <i>Test 8</i>	51
3.46. Mínimos cuadrados con el modelo linealizado: Predicción en el 95 % (t=133) de la vida del dispositivo del <i>Test 8</i>	51
3.47. Mínimos cuadrados con el modelo linealizado: Predicción en el 64 % (t=63) de la vida del dispositivo del <i>Test 26</i>	52
3.48. Mínimos cuadrados con el modelo linealizado: Predicción en el 85 % (t=84) de la vida del dispositivo del <i>Test 26</i>	52
3.49. Mínimos cuadrados con el modelo linealizado: Predicción en el 95 % (t=94) de la vida del dispositivo del <i>Test 26</i>	52
3.50. Mínimos cuadrados con el modelo linealizado: Predicción en el 64 % (t=151) de la vida del dispositivo del <i>Test 36</i>	53
3.51. Mínimos cuadrados con el modelo linealizado: Predicción en el 85 % (t=201) de la vida del dispositivo del <i>Test 36</i>	53
3.52. Mínimos cuadrados con el modelo linealizado: Predicción en el 95 % (t=225) de la vida del dispositivo del <i>Test 36</i>	53
3.53. Media horizontal con el modelo linealizado: Predicción en el 64 % (t=90) de la vida del dispositivo del <i>Test 8</i>	55
3.54. Media horizontal con el modelo linealizado: Predicción en el 85 % (t=119) de la vida del dispositivo del <i>Test 8</i>	55
3.55. Media horizontal con el modelo linealizado: Predicción en el 95 % (t=133) de la vida del dispositivo del <i>Test 8</i>	55
3.56. Media horizontal con el modelo linealizado: Predicción en el 64 % (t=63) de la vida del dispositivo del <i>Test 26</i>	56

3.57. Media horizontal con el modelo linealizado: Predicción en el 85 % (t=84) de la vida del dispositivo del <i>Test 26</i>	56
3.58. Media horizontal con el modelo linealizado: Predicción en el 95 % (t=94) de la vida del dispositivo del <i>Test 26</i>	56
3.59. Media horizontal con el modelo linealizado: Predicción en el 64 % (t=151) de la vida del dispositivo del <i>Test 36</i>	57
3.60. Media horizontal con el modelo linealizado: Predicción en el 85 % (t=201) de la vida del dispositivo del <i>Test 36</i>	57
3.61. Media horizontal con el modelo linealizado: Predicción en el 95 % (t=225) de la vida del dispositivo del <i>Test 36</i>	57
3.62. Predicción del <i>RUL</i> a lo largo del tiempo para el <i>Test 8</i>	64
3.63. Predicción del <i>RUL</i> a lo largo del tiempo para el <i>Test 26</i>	64
3.64. Predicción del <i>RUL</i> a lo largo del tiempo para el <i>Test 36</i>	64
4.1. Arquitectura <i>general</i> [9]	69
4.2. Arquitectura <i>Tradicional</i> [8]	70
4.3. Arquitectura <i>CEP</i> [8]	71
4.4. Arquitectura <i>Lambda</i> [8]	73
4.5. Arquitectura <i>Kappa</i> [8]	75
4.6. Arquitectura <i>Unificada</i> [8]	76
4.7. Primera aproximación a la estructura de la plataforma	77
4.8. <i>Apache Kafka</i>	78
4.9. <i>Apache Flume</i>	79
4.10. Diseño de la ingestión de datos	79
4.11. Diseño del procesado y modelado de datos	81
4.12. Diseño del servicio de datos	82
4.13. Arquitectura final del sistema	84

Índice de tablas

3.1. Diferencia entre el <i>RUL</i> real y predicho. Resultados para la media horizontal . <i>Tests 8-24</i>	38
3.2. Diferencia entre el <i>RUL</i> real y predicho. Resultados para la media horizontal . <i>Tests 26-38</i>	38
3.3. Diferencia entre el <i>RUL</i> real y predicho. Resultados para ajuste por mínimos cuadrados . <i>Tests 8-24</i>	42
3.4. Diferencia entre el <i>RUL</i> real y predicho. Resultados para ajuste por mínimos cuadrados . <i>Tests 26-38</i>	42
3.5. Diferencia entre el <i>RUL</i> real y predicho. Resultados para el filtro de Kalman extendido . <i>Tests 8-24</i>	46
3.6. Diferencia entre el <i>RUL</i> real y predicho. Resultados para el filtro de Kalman extendido . <i>Tests 26-38</i>	46
3.7. Diferencia entre el <i>RUL</i> real y predicho. Resultados para mínimos cuadrados con el modelo linealizado . <i>Tests 8-24</i>	50
3.8. Diferencia entre el <i>RUL</i> real y predicho. Resultados para mínimos cuadrados con el modelo linealizado . <i>Tests 26-38</i>	50
3.9. Diferencia entre el <i>RUL</i> real y predicho. Resultados para la media horizontal con el modelo linealizado . <i>Tests 8-24</i>	54
3.10. Diferencia entre el <i>RUL</i> real y predicho. Resultados para la media horizontal con el modelo linealizado . <i>Tests 26-38</i>	54
3.11. Resultados obtenidos en el trabajo de referencia [3]	58
3.12. Error absoluto medio de cada método al 95 % de la vida del dispositivo sobre todo el conjunto de tests	60
3.13. Resultados para el <i>Tests 8</i>	61
3.14. Resultados para el <i>Tests 26</i>	62
3.15. Resultados para el <i>Tests 36</i>	63

Capítulo 1

Introducción

1.1. Contexto

Los dispositivos electrónicos son una pieza clave y fundamental en la sociedad actual. En gran cantidad de escenarios es de vital importancia que estos dispositivos estén en continuo funcionamiento y la detección de posibles fallos es un factor crucial para mantener en funcionamiento el sistema en el que se integran. Los posibles fallos de estos dispositivos electrónicos normalmente suelen ser causados por fallos en los componentes que integran, por ejemplo transistores MOSFET.

Los transistores MOSFET son dispositivos semiconductores cuyo funcionamiento es similar al de un interruptor, deja pasar la corriente cuando se le aplica un potencial (suficiente) a uno de sus terminales. El uso continuado de estos dispositivos hace que se produzca un efecto de envejecimiento que es el causante de distintas formas de fallo. En este contexto, es importante encontrar las características que describen el envejecimiento y las formas de fallo. Una vez detectadas las características de envejecimiento y fallo se podrá intentar predecir cual es el tiempo de vida restante del dispositivo, lo que generalmente se conoce como pronosis.

La pronosis de transistores MOSFET se encarga de predecir el tiempo de vida útil de uso que le queda a un transistor. Existen diferentes técnicas y métodos para realizar pronosis de dispositivos con los que en el desarrollo de esta práctica se va intentar corroborar si, para los datos que se manejan aquí, funcionan y de que manera.

1.2. Objetivos

En este proyecto se ha intentará afrontar el problema de la pronosis MOSFET. Este trabajo parte de un trabajo inicial realizado en el campo de los transistores MOSFET y que va acompañado de unos datos recogidos mediante experimentación sobre dispositivos con envejecimiento acelerado. En este trabajo se intentará reproducir el proceso realizado en estos trabajos anteriores explorando nuevas técnicas para la predicción del tiempo de vida útil restante para cada dispositivo.

Finalmente se planteará un escenario en el que una empresa ficticia necesita realizar las tareas de experimentación y pronosis de transistores MOSFET a gran escala. Para

ello se diseñará una arquitectura *Big Data* que intente dar respuesta a los problemas que esa empresa ficticia pueda tener.

1.3. Planificación del trabajo

El proyecto se inicia la primera semana de abril de 2018, intentando alargar la finalización del mismo lo máximo posible, entre el 19 y 25 de julio, es decir, se va a disponer de unas 16 semanas y media para su realización. A continuación se expone la planificación proyectada al inicio del proyecto, y la planificación final realmente seguida.

1.3.1. Planificación inicial

Se disponen de 16 semanas y media que se van a distribuir de la siguiente manera:

- Semana 1: Toma de contacto con los datos y el tema que se va a tratar en el proyecto.
- Semana 2-6: Estudio de la teoría de los transistores MOSFET y exploración del conjunto de datos y primeros procesamiento de los datos.
- Semana 7-10: Últimos procesamientos de datos, documentación técnicas y algoritmos de prognosis y documentación sobre arquitecturas *Big Data*.
- Semana 11-16: Aplicación de técnicas de prognosis sobre los datos filtrados, finalización de documentación de todo el proceso y conclusiones.
- Durante las 16 semanas se realiza un proceso de documentación de todo el trabajo realizado.

1.3.2. Planificación final

La planificación inicial no ha tenido grandes cambios. Las variaciones en la planificación con respecto a la inicial se enumeran a continuación:

- La exploración y procesamiento de los conjuntos de datos se ha alargado una o dos semanas más debido a la complejidad del conjunto de datos.
- El procesamiento de datos se retomó hacia la semana 11 para realizar un reprocesado automático de todos los tests ya que sólo se había realizado para un subconjunto de ellos.
- El retraso debido al procesamiento de los datos ha obligado a reducir la cantidad de técnicas que en un principio se pretendían utilizar.
- Problemas con uno de los algoritmos de prognosis ha obligado a rehacer en varias ocasiones los experimentos.

1.4. Estructura de la memoria

Esta memoria recoge el estudio realizado sobre transistores MOSFET orientado al cálculo de la vida útil de estos dispositivos. Se recoge la descripción del procesamiento de los datos extraídos tras la fase de experimentación, y que posteriormente serán utilizados para la aplicación en algoritmos que permitan calcular el tiempo de vida útil restante de los dispositivos. Finalmente, se plantea de forma teórica una arquitectura *Big Data* orientada a una posible empresa fabricante de este tipo de dispositivos y que necesita realizar este tipo de experimentación a gran escala. Todo este contenido se estructura de la siguiente manera:

- **Contexto del proyecto: Prognosis de transistores MOSFET:** En este capítulo se describen de forma breve qué son los transistores MOSFET y sus características, además de resumir otros trabajos previos relacionados con la prognosis de transistores MOSFET.
- **Fase experimental. Prognosis:** En este capítulo se explica el procedimiento de análisis y transformación de los datos realizados. En este capítulo también se explica qué tipos de algoritmos se han utilizado para el cálculo de la vida útil restante de los dispositivos, y cuáles son los resultados obtenidos.
- **Estudio de una posible plataforma Big Data:** En este capítulo se plantea una posible arquitectura *Big Data* para una empresa ficticia fabricante de dispositivos electrónicos semiconductores, como son los transistores MOSFET. En este capítulo se intenta dar solución a la necesidad de realizar todos los procedimientos realizados en el capítulo anterior a gran escala.
- **Conclusiones:** En este capítulo se expone un resumen del contenido del trabajo junto a aspectos a tener en cuenta en futuros trabajos relacionados con la prognosis de transistores MOSFET.

Capítulo 2

Contexto del proyecto: Prognosis de transistores MOSFET

2.1. Introducción

En la sociedad actual vivimos rodeados de dispositivos electrónicos, es decir, vivimos rodeados de dispositivos semiconductores. La industria del semiconductor tiene un nicho de mercado grande, por lo tanto la demanda de dispositivos electrónicos es continua, siendo un claro ejemplo los sectores de la telefonía móvil o la IoT (Internet of Things, o Internet de las Cosas) actualmente en crecimiento.

Este tipo de dispositivos tienen un gran impacto en la economía ya que determinan el coste y la eficiencia de sistemas, por lo tanto cualquier estudio que ayude a determinar su estado actual “su salud” en términos de *Health Management* y su disponibilidad futura es un valor añadido para las empresas.

Este trabajo se centra en un tipo de dispositivo semiconductor, el transistor MOSFET (Metal-Oxide-Semiconductor-Field-Effect Transistor), que son dispositivos esenciales en sistemas con funciones autónomas para el control de vehículos, comunicaciones, navegador o radares.

2.2. Transistores MOSFET

2.2.1. ¿Qué son?

El transistor de efecto de campo metal-óxido-semiconductor o MOSFET es un transistor utilizado para amplificar o conmutar señales electrónicas. Es uno de los transistores más utilizados en la industria microelectrónica, tanto en circuitos analógicos como digitales. Casi todos los microprocesadores comerciales están basados en este tipo de transistor.

Los MOSFET como su propio nombre indica, son transistores de efecto de campo por medio de un óxido semiconductor que se usa como dieléctrico. El campo eléctrico que se crea se utiliza para controlar su conducción. Este tipo de dispositivo está construido con una estructura MOS, compuesta de dos terminales y tres capas, al que se le añaden dos terminales más con un material fuertemente dopado (proceso intencional de agregar impurezas en un semiconductor extremadamente puro o intrínseco, con el fin de cambiar sus

6CAPÍTULO 2. CONTEXTO DEL PROYECTO: PROGNOSIS DE TRANSISTORES MOSFET

propiedades eléctricas, aquellos que están altamente dopados tienen un comportamiento mas cercano al de un conductor), dando lugar a un dispositivo de cuatro terminales. Esta estructura se construye sobre un sustrato de silicio sobre el que se genera una capa de óxido que posee características dieléctricas, y sobre ésta, se coloca una capa de metal de aluminio con capacidades conductoras. A los lados de la interfase *Óxido-Semiconductor* se colocan los terminales fuertemente dopados, como se puede ver en la *Figura 2.1*. El nombre de cada uno de los terminales es: Sustrato (Body,B), Fuente (Source, S), Puerta (Gate, G) y Drenador (Drain, D). Normalmente a pesar de ser un dispositivo de cuatro terminales, el sustrato y la fuente suelen estar conectados dando lugar a un dispositivo de tres terminales (fuente, puerta y drenador).

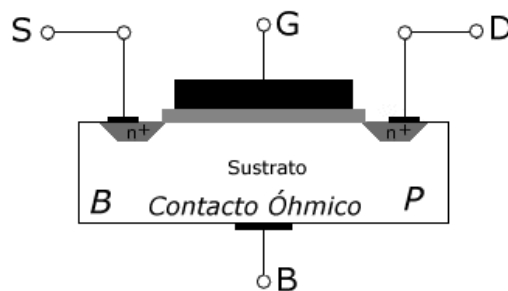


Figura 2.1: Estructura MOSFET de canal N

Antes de continuar, como ya han aparecido algunos términos relacionados con los MOSFET, se van a definir la terminología que se emplea con ellos:

- Fuente o Source (S): Terminal del transistor con menor tensión (tipo N, mayor tensión tipo P).
- Drenador o Drain (D): Terminal del transistor con mayor tensión (tipo N, menor tensión tipo P).
- Puerta o Gate (G): Terminal del transistor donde se aplica tensión para generar la banda de conducción entre la fuente y el drenador.
- Sustrato o Body (B): Terminal del transistor que conecta directamente con el semiconductor y que normalmente esta conectado con la fuente.
- Tensión umbral (V_T): Tensión aplicada en la puerta a partir de la cual se produce el canal de conducción.
- Canal o banda de conducción: Región del semiconductor en la que, por el efecto de aplicar una tensión en la puerta, puede circular la corriente eléctrica.
- Estados del transistor:
 - Corte (OFF state): En este estado la tensión aplicada a la puerta esta por debajo de la tensión umbral del transistor (V_T). El transistor se encuentra apagado, no hay conducción entre la fuente y el drenador, de modo que el MOSFET se podría interpretar como un interruptor abierto.

- Conducción (ON state): En este estado, al aplicar una tensión a la puerta mayor que la tensión umbral, se genera un canal de conducción, creándose una diferencia de potencial entre la fuente y el drenador y por lo tanto dando lugar a una corriente.
- Saturación: En este estado la tensión que se genera entre el drenador y la fuente (V_{DS}) supera un valor fijo que se denomina tensión de saturación (V_{DSSat}). En este estado, el transistor mantiene constante su corriente en el drenador independientemente de la tensión entre la fuente y el drenador. En el estado de saturación el canal se interrumpe o estrangula y sucede cuando $V_{DS} \geq V_T$ y $V_{DS} > (V_{GS} - V_T)$.

Continuando con la explicación del funcionamiento de los transistores MOSFET, cuando se aplica una tensión positiva sobre el terminal de puerta (MOSFET tipo N) se crea un campo eléctrico que si es suficientemente intenso, genera una banda de conducción que permite el paso de corriente entre la fuente y el drenador. Cuanto mayor sea la tensión aplicada, mayor será la banda de conducción.

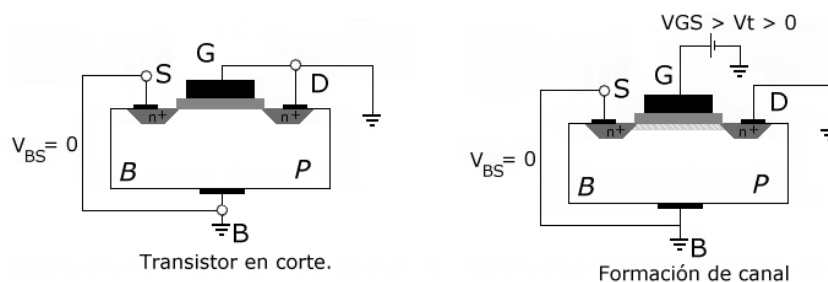


Figura 2.2: Estados del transistor

Para los transistores tipo P el funcionamiento es justamente al contrario, se aplica una tensión negativa y la corriente circula en sentido contrario. En los transistores tipo MOS, los terminales fuente y drenador son intercambiables actuando como drenador siempre el de mayor tensión (para el tipo N, el de menor tensión para el tipo P). La representación a nivel de circuito se puede ver en la Figura 2.3, a la derecha su forma física.

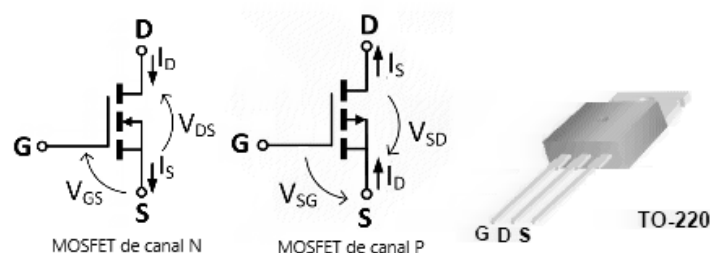


Figura 2.3: Representación del transistor MOSFET

Los dispositivos que se emplean en la experimentación montan la carcasa tipo TO-220 (Figura 2.4). Este tipo de diseño monta tres patillas en la parte inferior (*gate*, *drain*,

8CAPÍTULO 2. CONTEXTO DEL PROYECTO: PROGNOSIS DE TRANSISTORES MOSFET

source) y una placa metálica superior conectada a *drain* que funciona como disipador de calor.

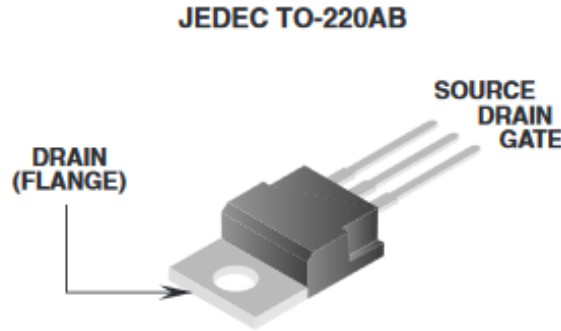


Figura 2.4: Ejemplo de MOSFET con carcasa *TO-220*

2.2.2. La influencia de la temperatura en MOSFET

En esta sección se va hablar de la influencia de la temperatura en el comportamiento de los transistores MOSFET. La temperatura influye en el comportamiento sobre todo porque la movilidad de los electrones y los huecos del semiconductor se ve reducida cuando la temperatura aumenta. Al reducirse la movilidad de electrones y huecos del semiconductor, la resistencia del dispositivo aumenta. En la *Figura 2.5* se puede observar como aumenta la resistencia conforme aumenta la temperatura en el interior del dispositivo.

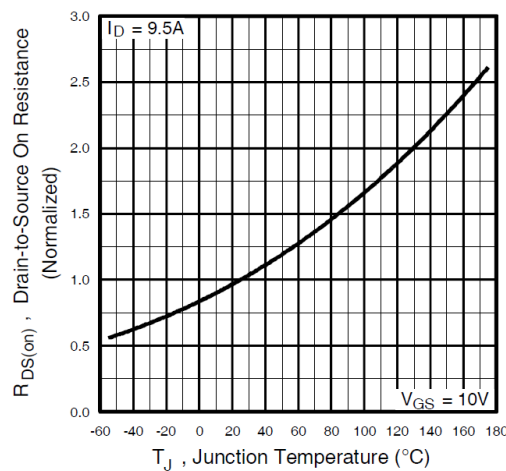


Figura 2.5: Variación de la resistencia respecto de la temperatura

La resistencia del dispositivo en estado *ON* ($R_{DS(on)}$) en una temperatura dada (T en temperatura absoluta) puede ser estimada con la siguiente ecuación:

$$R_{DS(on)}(T) = R_{DS(on)}(25^{\circ}C) \left(\frac{T}{300} \right)^{2,3} \quad (2.1)$$

Aunque la temperatura es un factor importante en la variación de la resistencia (o intensidad) hay otros factores influyentes, como puede ser la diferencia de potencial entre la *puerta* y la *fuentes* del transistor, es decir, el potencial que se le aplica a la puerta. Al aplicar mayor potencial en la puerta se genera una banda de conducción mayor y por lo tanto circula más corriente. La *Figura 2.6* describe este comportamiento.

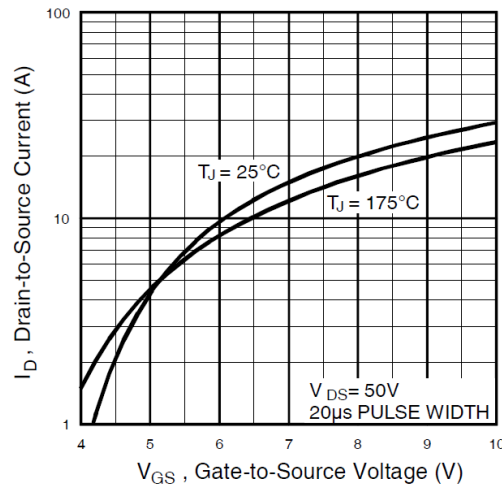


Figura 2.6: Variación de la intensidad respecto del potencial aplicado en la puerta y la temperatura

En resumen, hay dos factores influyentes en la variación de la resistencia o la intensidad (a parte, evidentemente, del potencial aplicado entre la fuente y el drenador), que son la temperatura del dispositivo y el potencial aplicado en la puerta del dispositivo. En la *Figura 2.7* se puede observar como varía la intensidad entre la *fuentes* y el *drenador* en función de estas características.

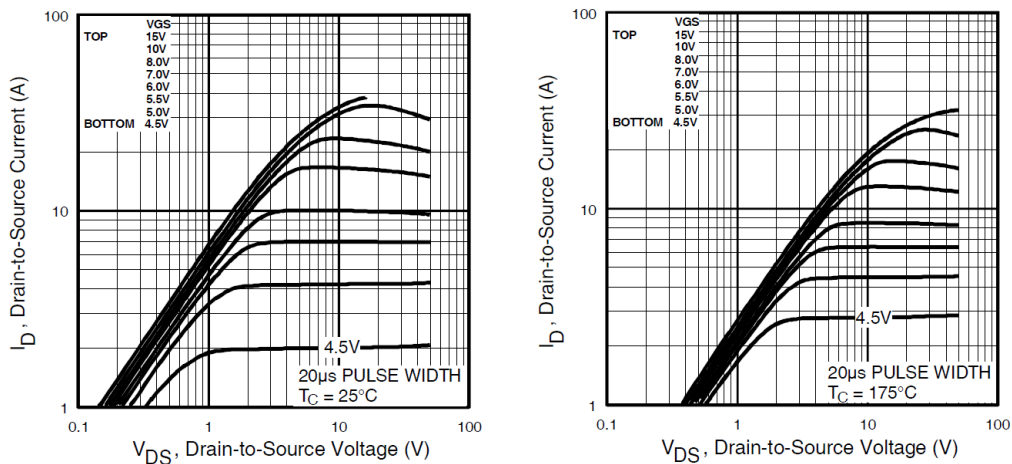


Figura 2.7: Comportamiento de I_D respecto de V_{GS} , V_{DS} y la temperatura

2.3. Prognosis. Estado del arte

El mantenimiento de transistores MOSFET, como el de otros dispositivos similares, normalmente esta basado en la confiabilidad y disponibilidad de los datos del fabricante. Esto funciona bien cuando se trata del mantenimiento de un conjunto amplio de dispositivos, pero no funciona tan bien cuando se trata de dispositivos individuales. En algunos escenarios es crítico conocer el estado de funcionamiento de los dispositivos y posibles fallos derivados, lo que conlleva a aplicar métodos de prognosis para la supervisión de su estado de salud. La prognosis es una ciencia que se encarga del análisis de las formas y condiciones de fallo, además de la detección de signos de desgaste y envejecimiento tempranos. Normalmente la prognosis se encarga de realizar una estimación del tiempo de vida útil (*RUL* o *Remaining Usefull Life*) que le queda al dispositivo. Este tipo de predicciones se realizan *in-situ*, en dispositivos que están en servicio y de los que se tienen ciertas características que se miden durante este servicio.

El trabajo de partida, sobre el que aquí se desarrolla la prognosis sobre transistores MOSFET, es el realizado en [3], en el que se exponen tres técnicas con las que poder realizar estimaciones de tiempo de vida útil sobre transistores MOSFET. Pero para poder entender la adquisición de datos y las características que describen el desgaste y los modos de fallo, también es necesario revisar otros trabajos anteriores como son [4] y [5].

El envejecimiento acelerado juega un papel muy importante en el desarrollo de la prognosis y la gestión de la salud de componentes. El envejecimiento acelerado es un método frecuentemente utilizado en dispositivos en los que la vida útil esperada se encuentra alrededor de miles de horas, y en los que no parece razonable esperar su fallo durante el uso normal. Para el envejecimiento acelerado de los MOSFET usados en los artículos citados, se han empleado dos técnicas: sobrecarga térmica (*thermal overstress*) y ciclos de potencia (*power cycling (switching mode)*).

La técnica de ciclos de potencia consiste en poner una señal cuadrada de alta frecuencia en la puerta del dispositivo que hace cambiar de estado continuamente al dispositivo. Las características de la señal son de 15V de amplitud a una frecuencia de 1KHz con un ciclo de trabajo del 40 %. Este tipo de práctica genera una gran cantidad de energía, lo que hace que para la sobrecarga térmica no sea necesario aplicar calor de forma directa. Para evitar que el dispositivo se queme rápidamente se fijan dos umbrales de temperatura, uno que pone a trabajar el dispositivo si la temperatura se encuentra por debajo del umbral fijado (T_{min}) y otro que para completamente el funcionamiento del dispositivo si se supera ese umbral de temperatura (T_{max}). Se crea así algo parecido a un ciclo de histéresis (Figura 2.8).

Algunas de las formas de fallo mas frecuentes son *latch-up*, *die-attach degradation* o la pérdida de control de la puerta del dispositivo. *Latch-up* es la aparición de una estructura parásita que inhabilita el correcto funcionamiento del dispositivo causando normalmente a una sobrecarga. *Die-attach degradation* corresponde a la degradación del material que compone el dispositivo dando lugar a la aparición de huecos (Figura 2.9) y que afecta también el correcto funcionamiento del dispositivo. La pérdida de control de la puerta suele estar causada y precedida por un aumento de la resistencia del dispositivo en estado *ON*. Este aumento de resistencia hace que la corriente en el dispositivo disminuya y falle al cambiar de estado, no circulando corriente en ninguno de los dos estados. La resistencia del dispositivo también está influenciada por la temperatura, un aumento de temperatura

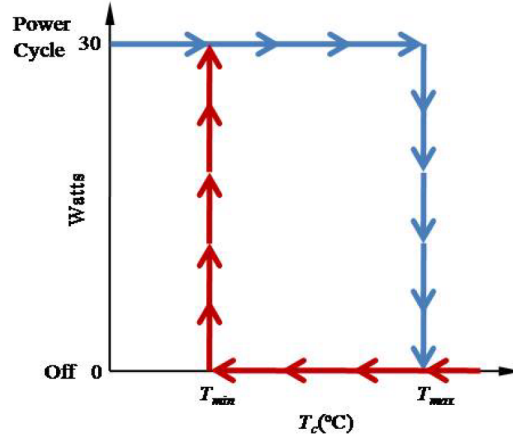


Figura 2.8: Control del envejecimiento por sobrecarga térmica

hace que se reduzca la movilidad de las cargas portadoras. Normalmente se espera que la resistencia aumente cuadráticamente respecto de la temperatura.

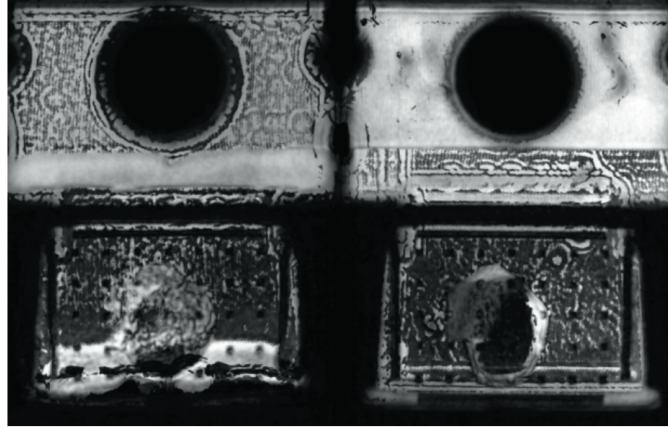


Figura 2.9: Aparición de huecos en *die-attach degradation*

La selección de la resistencia en estado *ON* ($R_{DS(on)}$) como característica que permite detectar el fallo del dispositivo, parece razonable. Como se ha comentado anteriormente, la resistencia varía con respecto a la temperatura, por eso es importante normalizar la resistencia con respecto a la temperatura antes de empezar a trabajar con los valores de la resistencia. Una vez normalizada la resistencia con respecto a la temperatura en la que fue medida, para poder ver el incremento de la resistencia, $R_{DS(on)}$ se normaliza (una segunda vez) con los valores medidos en la condición prístina del dispositivo ($\Delta R_{DS(on)}$). Por lo tanto, $\Delta R_{DS(on)}$ parece la mejor característica para modelar los datos ya que, además de que parece ser el mejor indicador de la salud del dispositivo, puede ser calculada *in-situ* a partir de las medidas de la intensidad en el drenador (I_D), y el voltaje entre el drenador y la fuente (I_{DS}). El comportamiento de $\Delta R_{DS(on)}$ se considera (en [3]) exponencial a lo largo del tiempo, por lo tanto se puede describir como:

$$\Delta R_{DS(on)} = \alpha(e^{\beta t} - 1) \quad (2.2)$$

12CAPÍTULO 2. CONTEXTO DEL PROYECTO: PROGNOSIS DE TRANSISTORES MOSFET

Siendo α y β los parámetros del modelo que hay que estimar. Una vez estimados los parámetros del modelo se utiliza la función para extrapolar los resultados hasta encontrar el punto de ruptura del dispositivo ($\Delta R_{DS(on)} = 0.05$ en [3]) y calcular la distancia hasta el momento de la medida, obteniendo así el *RUL*. Para estimar los parámetros de la función se pueden usar distintos algoritmos de prognosis, en [3] se han probado *Gaussian Process Regression*, *Extended Kalman Filter* y *Particle Filter*.

Capítulo 3

Fase experimental. Prognosis

3.1. Introducción

En este capítulo se parte del conocimiento expuesto en la *sección 2.3*. Se ha presentado el método de experimentación (*thermal overstress* y *power cycling*), se sabe que las formas de fallo que suelen estar precedidas por un aumento de la resistencia, se conoce la característica que describe el estado de funcionamiento, que es la resistencia entre el drenador y la fuente ($R_{DS(on)}$) generalmente normalizada para su condición prístina ($\Delta R_{DS(on)}$) y se conocen algunas técnicas con las que modelar la degradación de los dispositivos, a partir de las cuales se puede realizar una estimación de la vida útil restante del dispositivo.

El objetivo de este capítulo es reproducir la fase experimental realizada en [3], extraer el valor de la característica que determina el estado de salud de los dispositivos durante el proceso de testeo de los dispositivos, reproducir alguna de las técnicas de prognosis expuestas en [3] y finalmente proponer alguna técnica adicional y comparar los resultados.

La tecnología empleada durante toda la fase experimental ha sido *Python* a través de *Anaconda*. *Anaconda* es una *suite* de código abierto que abarca una serie de aplicaciones librerías y conceptos diseñados para el desarrollo de la ciencia de datos. Sólo resaltar cuatro paquetes de esta *suite*: *Numpy* y *Pandas* para trabajar con estructuras de datos avanzadas, *Matplotlib* para la visualización de datos y *Jupyter Notebook* como entorno de desarrollo.

3.2. Datos. Exploración

Los datos con los que se va a trabajar [1] son los generados y utilizados en los trabajos anteriormente comentados, [3], [4] y [5]. Estos datos no vienen acompañados de ningún documento que explique cómo están compuestos y estructurados, o al menos, que ayude a su interpretación. Por lo tanto, el primer paso es explorar el contenido de los mismos y comprobar si es necesaria alguna transformación que facilite el trabajo.

El conjunto de datos está compuesto por *tests* realizados sobre 42 dispositivos, y cada *test* está formado por una serie de ejecuciones (llamadas *runs* en los datos originales). Cada dispositivo ha tenido entre una y siete ejecuciones. Cada fichero del conjunto de datos corresponde a la información generada en una ejecución de un dispositivo, esta

información se almacena en formato *MATLAB* (.mat). La estructura de estos ficheros tiene forma de árbol cuyas hojas son matrices que contienen los datos. La descripción de la estructura es la siguiente:

■ *measurement*:

- *pwmTempControllerState*: Matriz que contiene los valores que controlan el experimento.
 - *time*
 - *timeEpoch*
 - *nanosec_time*
 - *lowTemp*
 - *highTemp*
 - *shutdownTemp*
 - *gateVoltage*
 - *supplyVoltage*
 - *switchingFrequency*
 - *dutyCycle*
 - *gateState*
- *steadyState*: Contiene una matriz que recoge los siguientes valores del transistor independientemente de su estado.
 - *time*
 - *nanosec_time*
 - *timeEpoch*
 - *supplyVoltage*
 - *packageTemperature*
 - *drainSourceVoltage*
 - *drainCurrent*
 - *flangeTemperature*
- *transient*: Tiene las medidas de un ciclo completo de muestreo (40 % del tiempo en estado *ON*, 60 % del tiempo en estado *OFF*), si el muestreo es a 1KHz, cada *transient* contiene mil datos. Este elemento es una matriz de características de cada uno de los *transient* que a su vez dentro de una de sus celdas tiene otra matriz con las medidas tomadas en ese *transient*.
 - *date*
 - *nanosec_time*
 - *timeEpoch*
 - *dt*
 - (matriz con las medidas del transient a 1KHz,):
 - ◇ *gateSignalVoltage*
 - ◇ *gateSourceVoltage*

- ◇ *drainSourceVoltage*
- ◇ *drainCurrent*
- ◇ *nanosec_time*

Como se va a trabajar con *Python* durante toda la fase de experimentación, no resulta práctico trabajar con ficheros *.mat*, a pesar de que *Python* tiene un paquete (*Scipy*) que permite la lectura de este tipo de ficheros. El problema de leer ficheros *.mat* desde *Python* es que si la estructura del fichero *.mat* es compleja, *Python* la hace mucho mas compleja al intentar hacer una conversión de los tipos y estructuras de datos de *MATLAB*, a los tipos y estructuras de datos de *Python*. Para solucionar este problema se ha desarrollado un software en *MATLAB* que lo que hace es transformar la estructura que alberga el fichero *MATLAB* en un directorio con varios ficheros *CSV*. La desventaja de transformar los ficheros *.mat* a varios ficheros *CSV* es que el volumen de los datos pasan a aumentar en tamaño del orden de seis veces más. Se crea un directorio con el mismo nombre del fichero *.mat* origen con el siguiente contenido:

- *pwmTempControllerState.csv*: Fichero en el que se encuentra la matriz de datos del mismo nombre en el fichero *.mat*.
- *steadyState.csv*: Fichero en el que se encuentra la matriz de datos del mismo nombre en el fichero *.mat*.
- *transient.csv*: Fichero en el que se encuentra la matriz de datos del mismo nombre en el fichero *.mat* pero sin la celda que contiene la matriz de medidas.
- *transient*: Directorio que contiene las matrices de datos correspondientes a cada uno de los *transient* (las mil medidas si el muestreo se realiza a 1Khz). Estos ficheros de medidas se conectan con las entradas del fichero *transient.csv* por fecha. Por ejemplo la entrada del fichero *transient.csv* con fecha “08/20/2010 18:22:18.055” tendrá como correspondencia un fichero “08-20-2010_18-22-18-055.csv” en el que se encuentran las medidas para ese *transient*, tal cual se encuentra la matriz en el fichero *.mat*.

Como ya se ha comentado anteriormente el fichero *pwmTempControllerState.csv* recoge los valores de control del experimento, las temperaturas entre las que se quiere mantener el dispositivo así como la temperatura de apagado en caso de sobrecalentamiento, los voltajes aplicados en la puerta y en la fuente (figuras 3.1, 3.2 y 3.3), los datos temporales y otros datos como la frecuencia de muestreo, el porcentaje de tiempo que se va a mantener activo el dispositivo en cada ciclo de muestreo y su estado. Como se puede ver en la Figura 3.1 la temperatura de control va aumentando a lo largo del experimento, esta práctica se realiza para que la temperatura del dispositivo vaya aumentando progresivamente y no se queme de golpe.

El fichero *steadyState.csv* contiene un muestreo de los datos de los *transient* (ciclos de muestreo a 1KHz). Este fichero contiene los datos mezclados del dispositivo en estado *ON* y en estado *OFF*, y además no hay ninguna característica (como el voltaje en la puerta o el estado del dispositivo) que permita diferenciar si el dispositivo se encuentra en estado *ON* u *OFF*. Se podría pensar que el estado del dispositivo se puede extraer

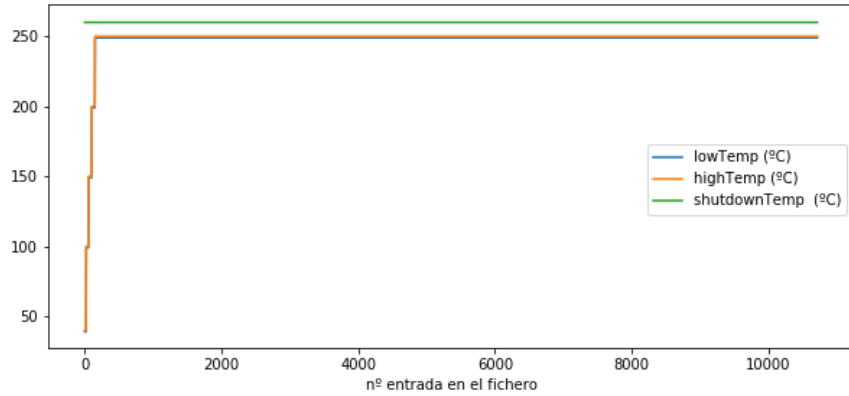


Figura 3.1: Representación de los valores de control de temperatura en *pwmTempControllerState.csv* para el *Test 36* del conjunto de datos.

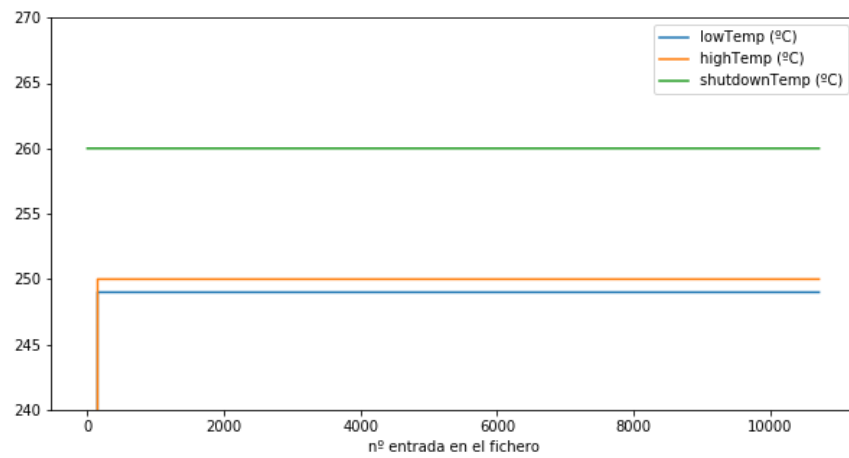


Figura 3.2: Representación de los valores de control de temperatura tras la fase de calentamiento en *pwmTempControllerState.csv* para el *Test 36* del conjunto de datos. (Zoom sobre la *Figura 3.1*)

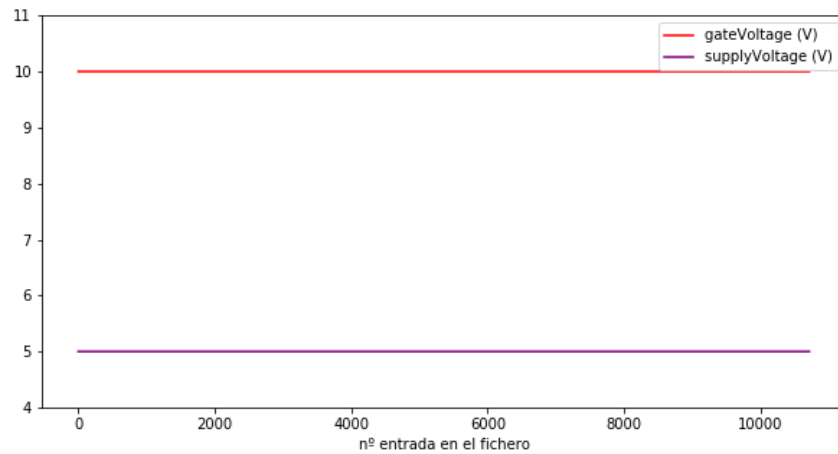


Figura 3.3: Representación de los valores del voltaje aplicado en la puerta y en el dispositivo en *pwmTempControllerState.csv* para el *Test 36* del conjunto de datos.

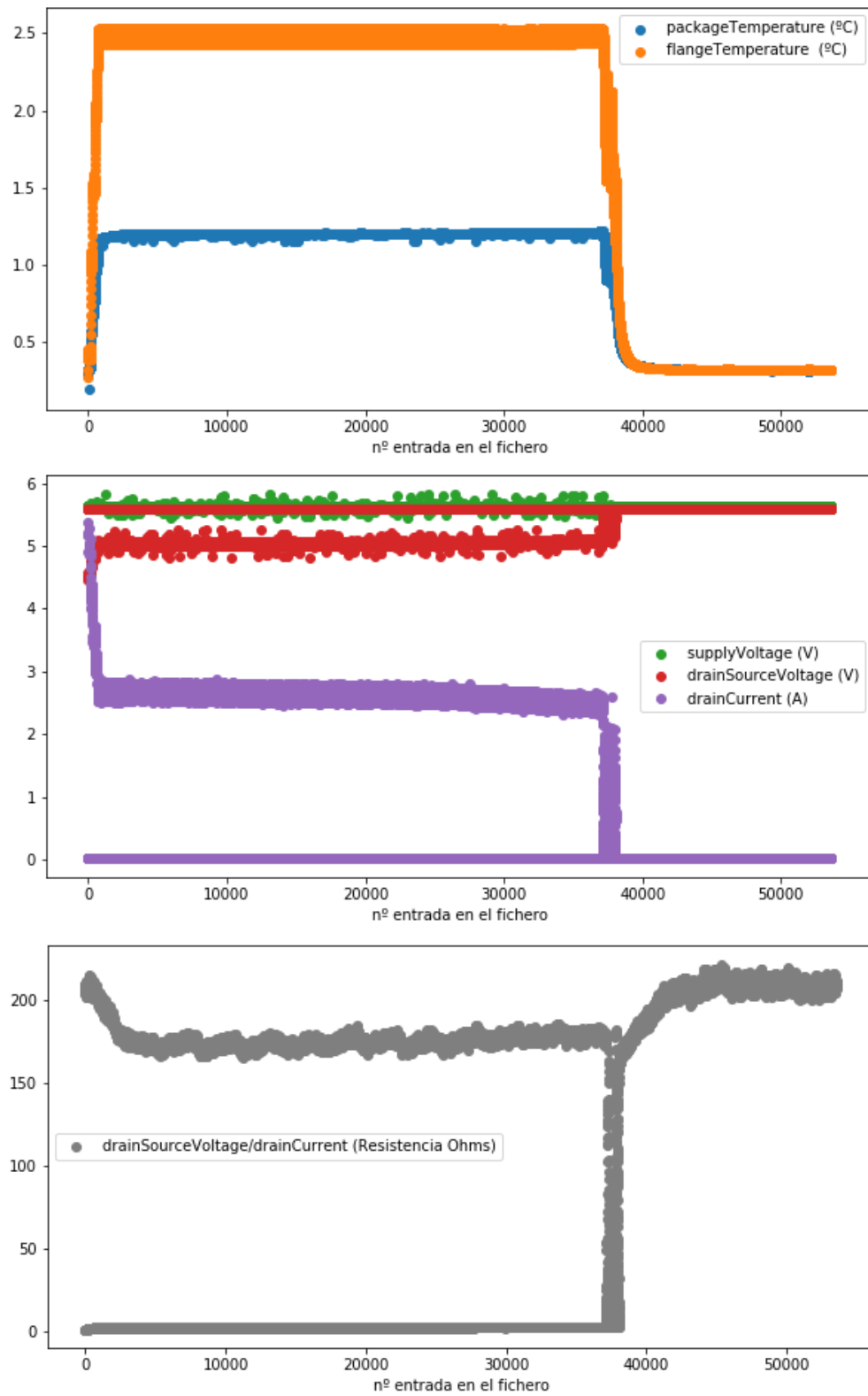


Figura 3.4: Representación de algunos datos contenidos en *steadyState.csv* para el *Test 36* del conjunto de datos.

de *pwmTempControllerState.csv* que tiene una característica que lo describe, pero como se puede ver en las figuras 3.1, 3.2 y 3.3, *pwmTempControllerState.csv* tiene algo mas de 10.000 datos frente a *steadyState.csv* que tiene algo mas de 50.000 datos. Parece razonable pensar que no va a haber correspondencia directa para todos los datos de *steadyState.csv*. En Figura 3.4 se pueden distinguir las dos franjas de los distintos estados y el punto de ruptura. En el estado *OFF* la intensidad del dispositivo (*drainCurrent* en morado) 0A y el voltaje (*drainSourceVoltage* en rojo) igual al suministrado (*supplyVoltage* en verde) alrededor de 5.6V. En el estado *ON* la intensidad entre 2.5 y 3 A y el voltaje alrededor de 5V. En el punto de ruptura se ve como hay una variación grande de los valores, para finalmente dejar de circular la corriente. En la Figura 3.4 se ha representado también la resistencia ($\text{drainSourceVoltage} / \text{drainCurrent}$), en la que también se pueden distinguir los dos estados y el punto de ruptura.

El directorio *transient* contiene los ficheros que tienen los datos de cada ciclo completo de muestreo. Las figuras 3.5, 3.6 y 3.7 representan los datos de los *transient* en tres instantes, el primer instante corresponde a uno de los primeros *transients*, el segundo instante corresponde a un *transient* recogido hacia la mitad del experimento, el tercero corresponde a un *transient* recogido en la parte final del experimento, cuando el dispositivo ya ha fallado.

Durante el proceso de exploración de los ficheros se ha percibido que los valores observados en *steadyState* y los observados en los *transient* no concuerdan. Por ejemplo, las intensidades en estado *ON* del dispositivo en *steadyState*, una vez se estabiliza la temperatura, se encuentran entre 2 y 3 amperios, mientras que los valores de la intensidades en estado *ON* del dispositivo en el instante correspondiente a la mitad el experimento se encuentran entre alrededor de 7 amperios. Por tanto el cálculo de la resistencia va a dar resultados distintos en *steadyState* y en los *transient*.

Visto éste problema, se ha decidido no usar los datos de *steadyState* para el cálculo de la resistencia y dar como válidos los datos de los *transient*. Los valores de los datos recogidos en los *transient* son abundantemente repetitivos para trabajar con ellos tal y como están, con trabajar con un porcentaje significativo de los mismos es suficiente. Por lo tanto, se ha realizado un muestreo sobre los datos de los ficheros *transient*.

3.3. Transformación de los datos

En la sección anterior se ha visto que los datos de los *transient* son abundantemente repetitivos y con trabajar con un porcentaje significativo de ellos es suficiente. A través de un procedimiento de muestreo sobre los *transient* se consigue un conjunto de datos representativo y suficiente. Este muestreo implica, además, una reducción del volumen de datos y una mayor facilidad de uso del conjunto de datos. En esta sección se va a describir las transformaciones que se han realizado a los datos y por qué. Ha sido necesario realizar tres transformaciones:

- Muestreo de los datos de los *transient* generados en el experimento.
- Cálculo de la resistencia del muestreo anterior y la normalización respecto de la temperatura en la que se ha realizado la muestra.

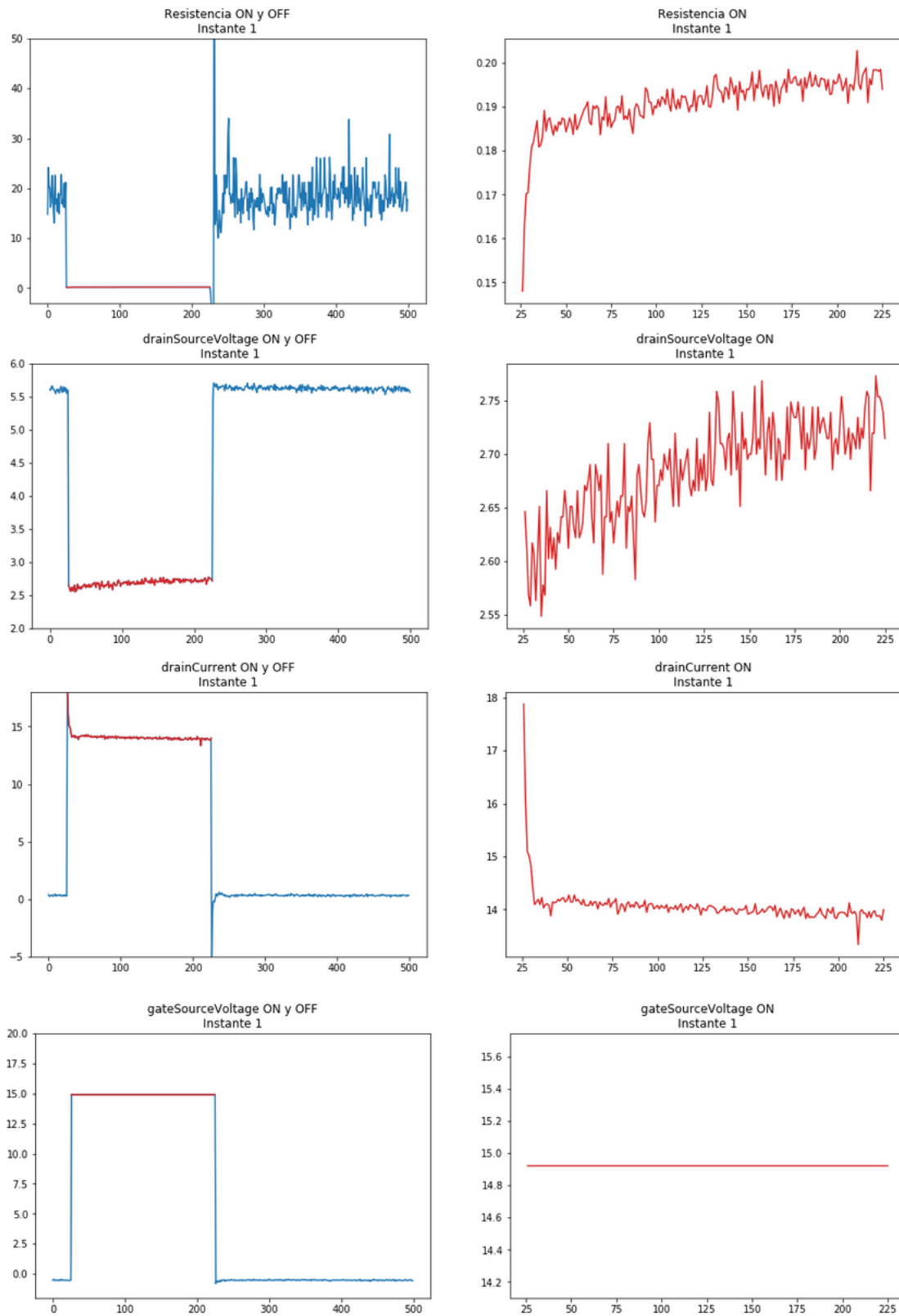


Figura 3.5: Datos del transient (resistencia, voltage, intensidad entre el *drenador* y la *fuelle*, y voltaje aplicado en la *puerta*) en el instante 1 para el *Test 36* del conjunto de datos.

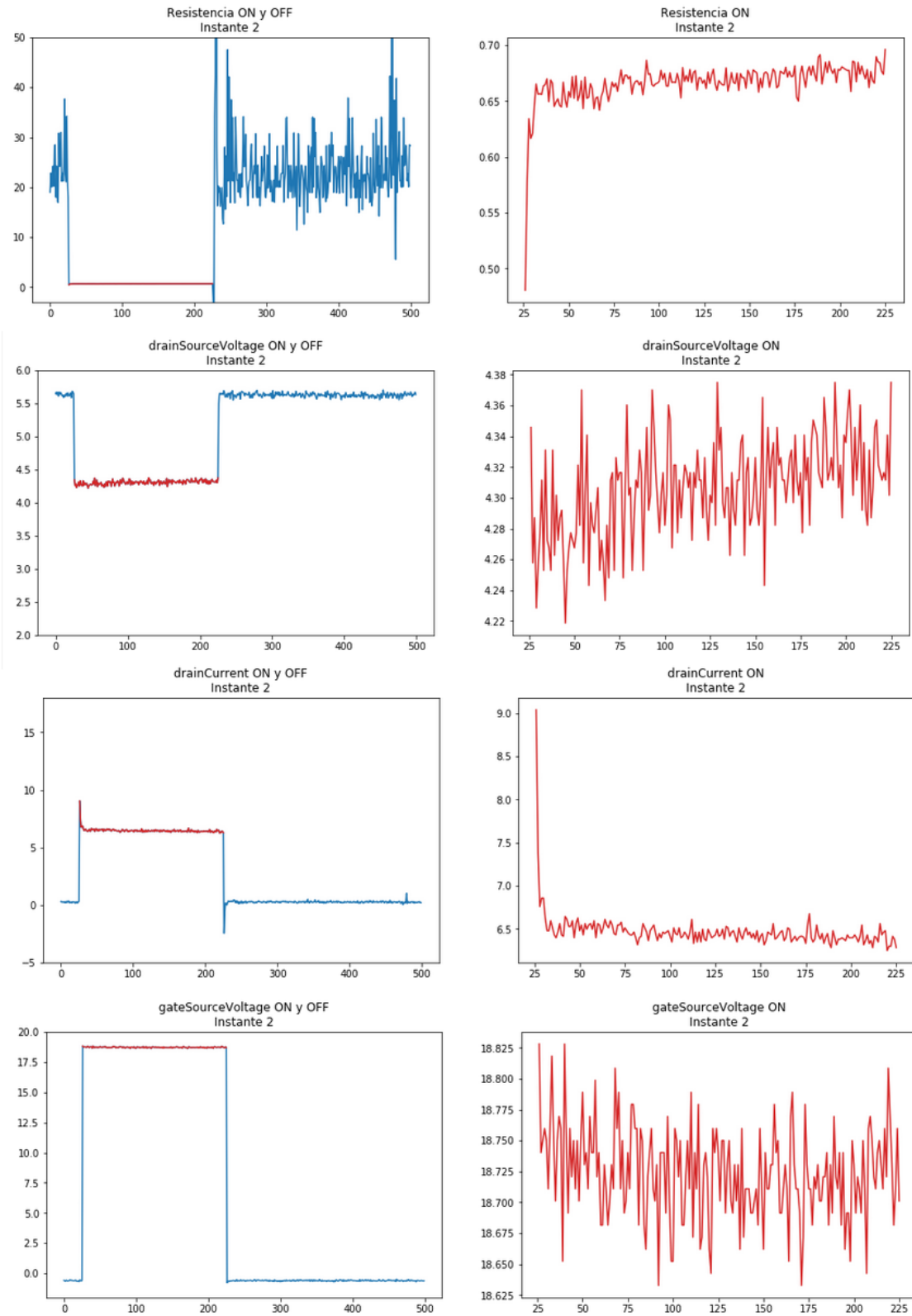


Figura 3.6: Datos del transient (resistencia, voltage, intensidad entre el *drenador* y la *fuelle*, y voltaje aplicado en la *puerta*) en el instante 2 para el *Test 36* del conjunto de datos.

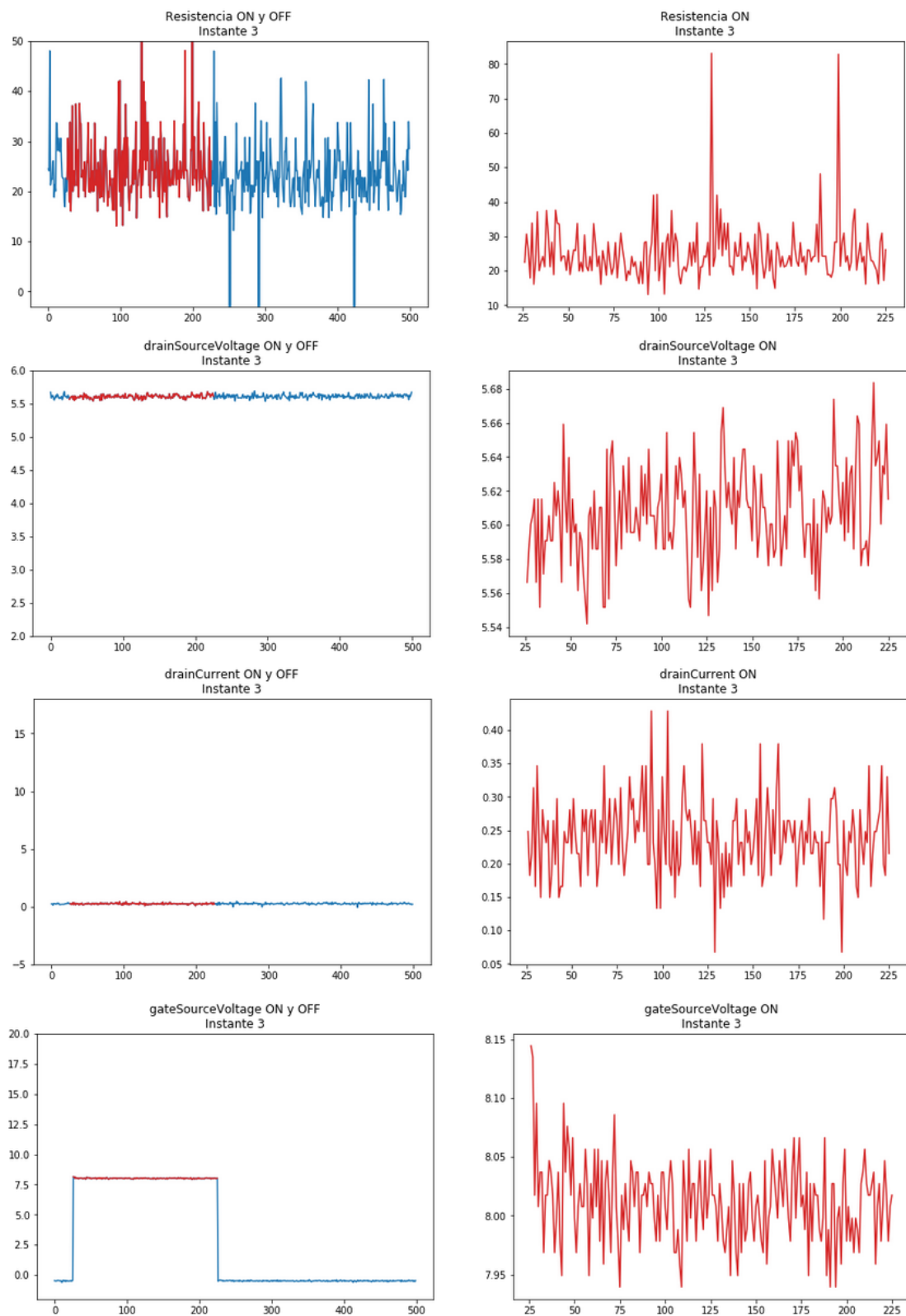


Figura 3.7: Datos del transient (resistencia, voltage, intensidad entre el *drenador* y la *fuelle*, y voltaje aplicado en la *puerta*) en el instante 3 para el *Test 36* del conjunto de datos.

- Cálculo de la media por minuto orientado al modelado de los datos.

3.3.1. Muestreo de los *transient*

En secciones anteriores se ha visto que hay ciertas discrepancias entre los valores de las mediciones en *steadyState* y los *transient*. Se ha optado por elegir las medidas recogidas en los *transient* ya que las medidas en *steadyState* parecen ser un muestreo de estos *transient* para el que se desconoce el procedimiento. Las figuras 3.5, 3.6 y 3.7 describen el contenido de los *transient* a lo largo de un experimento. En estas tres figuras se puede observar la forma que tienen los datos, apreciándose perfectamente los dos estados (salvo en el caso de ruptura). En las figuras se representan los valores del estado *OFF* en azul y los valores del estado *ON* en rojo, además se representa a la izquierda el estado *ON* separado del estado *OFF* para poder ver con mas detalle, y de forma ampliada, su comportamiento.

En este muestreo se realiza la media de los valores en cada estado de un *transient*, es decir, para cada *transient* se tendrá la media de las medidas en estado *ON* y la media de las medidas en estado *OFF* con la marca temporal del *transient* en las que se han recogido. Para delimitar qué valores del *transient* corresponden al estado *ON* del dispositivo y cuáles al estado *OFF*, se calcula la media entre el máximo valor de *gateSignalVoltage* y el mínimo (se supone que la señal de la *puerta* no sufre ruido ni variaciones al ser una señal que se aplica y no que se mide). Una vez calculado el límite entre estados, todo valor que se encuentre por debajo de ese límite se considera en estado *OFF*, y todo valor que se encuentre por encima, en estado *ON*.

Como ya se ha visto en la *sección 3.2* los *transient* tienen cuatro medidas que interesan para el muestreo: *gateSignalVoltage*, *gateSourceVoltage*, *drainSourceVoltage* y *drainCurrent*. Los *transient* no tienen medidas para la temperatura y es un dato deseable dentro del conjunto de datos resultante del muestreo. *SteadyState* tiene valores para la temperatura y es la única referencia de temperatura que se tiene en todo el conjunto de datos, por lo tanto, la temperatura (tanto *packageTemperature* como *flangeTemperature*) se extraen de aquí. Como cada *transient* y cada entrada (o fila) del *steadyState* tienen una marca temporal, a cada *transient* se le asigna el valor de temperatura de la entrada *steadyState* más cercana temporalmente. En resumen, se tiene como entrada las siguientes características medidas en el experimento:

- ***gateSignalVoltage (transient)***: Señal aplicada en la *puerta*. Describe el estado del dispositivo (*ON-OFF*).
- ***gateSourceVoltage (transient)***: Diferencia de potencial entre la *puerta* y la *fuentes*.
- ***drainSourceVoltage (transient)***: Diferencia de potencial entre el *drenador* y la *fuentes*.
- ***drainCurrent (transient)***: Corriente que pasa por el *drenador*.
- ***packageTemperature (steadyState)***: Temperatura medida en la carcasa o parte plástica del dispositivo.
- ***flangeTemperature (steadyState)***: Temperatura medida en la parte metálica plana superior del dispositivo.

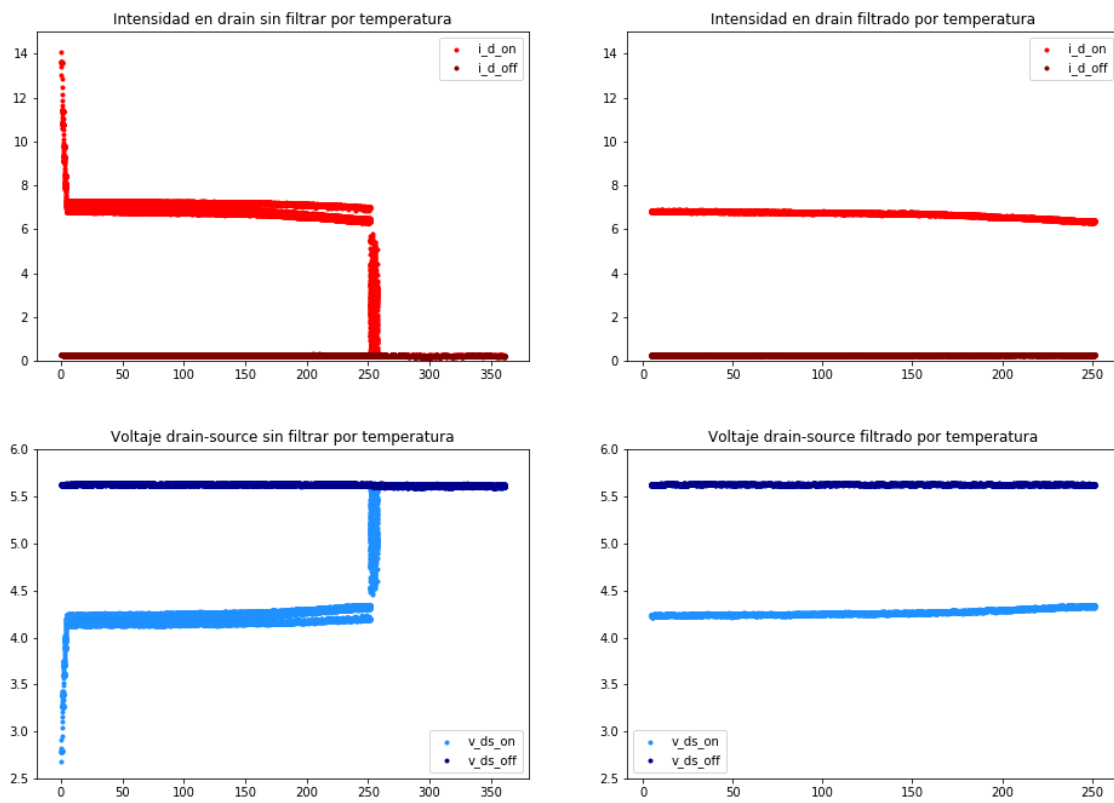


Figura 3.8: Resultados para la intensidad en el *drenador* y voltaje entre el *drenador* y la *fuelle* tras el muestreo de los *transient* para *Test 36*

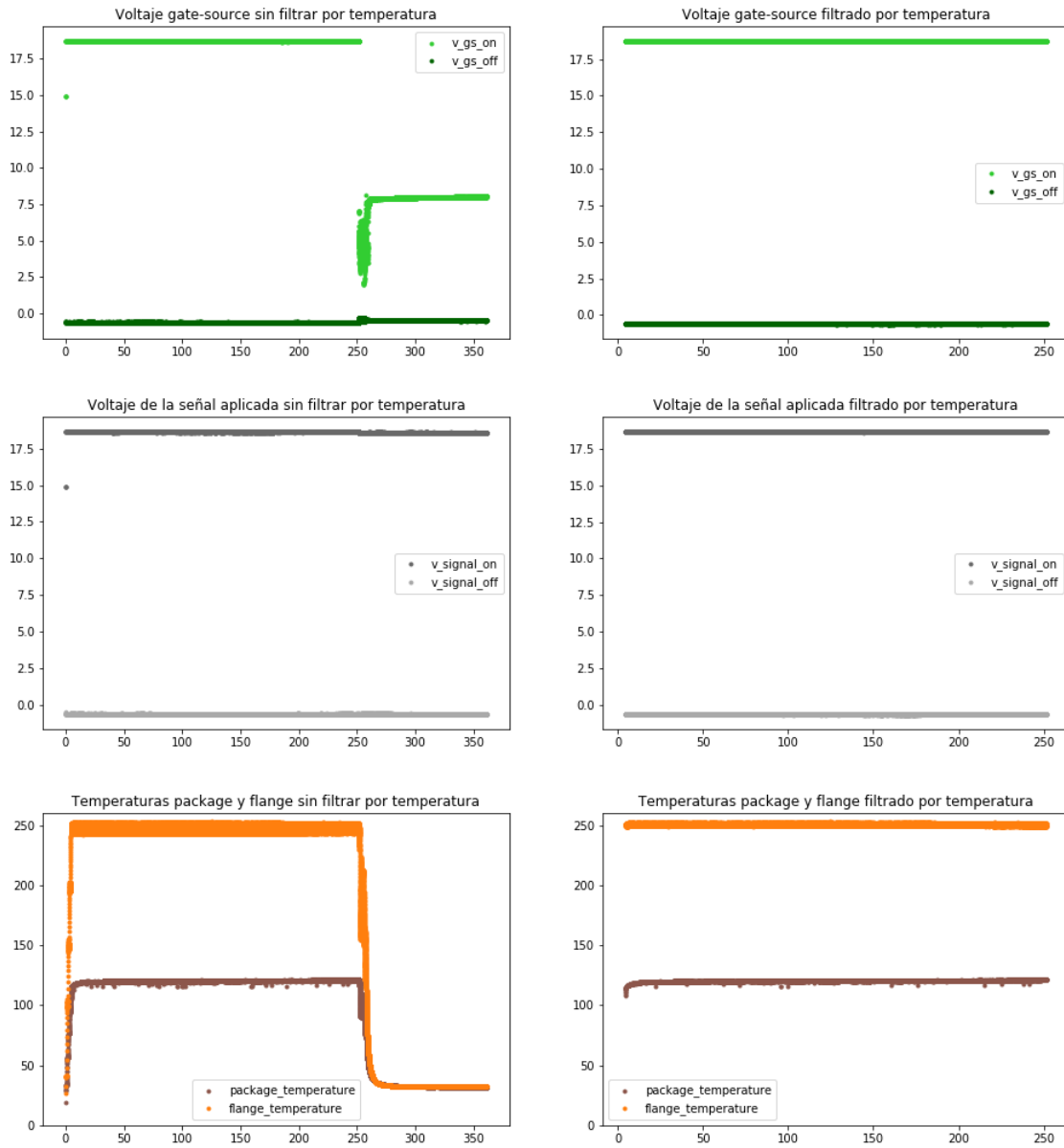


Figura 3.9: Resultados para el voltaje entre la *puerta* y la *fuentes*, voltaje de la señal aplicada, y temperaturas *package* y *flange* tras el muestreo de los *transient* para *Test 36*

Se genera como salida un conjunto de datos en los que se tiene dos valores por cada una de las características observadas en el *transient* (una por estado *ON-OFF*), además de la marca temporal y sus temperaturas asociadas:

- ***acum_time_mins***: Tiempo transcurrido en minutos desde el primer *transient* observado hasta el *transient* actual.
- ***date***: Marca temporal que corresponde al *transient* del que han sido extraídos los valores.
- ***i_d_on***: Corriente que pasa por el *drenador* en estado *ON*.
- ***i_d_off***: Corriente que pasa por el *drenador* en estado *OFF*.
- ***v_ds_on***: Diferencia de potencial entre el *drenador* y la *fuelle* en estado *ON*.
- ***v_ds_off***: Diferencia de potencial entre el *drenador* y la *fuelle* en estado *OFF*.
- ***v_gs_on***: Diferencia de potencial entre la *puerta* y la *fuelle* en estado *ON*.
- ***v_gs_off***: Diferencia de potencial entre la *puerta* y la *fuelle* en estado *OFF*.
- ***v_signal_on***: Valor del potencial aplicado en la *puerta* en estado *ON*.
- ***v_signal_off***: Valor del potencial aplicado en la *puerta* en estado *OFF*.
- ***package_temperature***: Temperatura medida en la carcasa o parte plástica del dispositivo.
- ***flange_temperature***: Temperatura medida en la parte metálica plana superior del dispositivo.

Una vez construido este conjunto de datos es necesario realizar un filtrado por temperatura. En este filtrado se van a descartar los datos relativos a la fase de calentamiento preliminar de cada *run* o ejecución de cada test junto con los datos que están por debajo del umbral de temperatura mínimo (*lowTemp*) marcado en *pwmTempControllerState*. La temperatura de filtrado empleada es *flange_temperature*, es la única que se encuentra cerca de los valores fijados en *pwmTempControllerState*. El resultado del filtrado de los datos se puede ver en las figuras 3.8 y 3.9.

3.3.2. Cálculo y normalización de la resistencia

Una vez realizado el muestreo para los valores de los *transient* y el filtrado por temperatura de estos mismos valores, se va a calcular la resistencia del dispositivo en estado *ON*. Como ya se ha comentado en el capítulo anterior, la resistencia depende de la temperatura en la que se encuentre el dispositivo (un aumento de la temperatura genera una disminución de la movilidad de las cargas portadoras), por eso es necesario aplicar una normalización a la resistencia tras su cálculo. Esta normalización se va a generar para dos temperaturas, una a temperatura ambiente (25°C) y otra a 230°C. Se ha cogido la temperatura de normalización 230°C porque en [5] se habla de que se intenta mantener la

temperatura del dispositivo a 230°C durante todo el ciclo de envejecimiento acelerado, además en [4] se habla de rangos de temperatura del dispositivo de entre 200°C y 260°C (temperatura en la que se apaga el circuito de experimentación para que no se queme). Esta normalización de la resistencia respecto de la temperatura se realiza con la fórmula 2.1 vista en el capítulo anterior.

La característica de interés no es en sí misma la resistencia (R), si no su incremento (ΔR) a lo largo del tiempo con respecto de su valor en las fases iniciales del experimento (sin envejecer), también denominado condición prístina. Por lo tanto una vez realizado el cálculo de la resistencia y realizadas las normalizaciones para las dos temperaturas, se calcula el incremento de la resistencia con respecto al primer valor del conjunto de datos. En resumen, se parte de los datos obtenidos en el muestreo de los *transient*, y se calcula la resistencia y su incremento junto con sus normalizaciones por temperatura, todo para el estado *ON* del dispositivo:

- ***acum_time_mins***: Tiempo transcurrido en minutos desde el inicio del experimento hasta el instante actual.
- ***package temperature***: Temperatura medida en la carcasa o parte plástica del dispositivo.
- ***flange temperature***: Temperatura medida en la parte metálica plana superior del dispositivo.
- ***i_d_on***: Corriente que pasa por el *drenador* en estado *ON*.
- ***v_ds_on***: Diferencia de potencial entre el *drenador* y la *fuelle* en estado *ON*.
- ***v_gs_on***: Diferencia de potencial entre la *puerta* y la *fuelle* en estado *ON*.
- ***v_signal_on***: Valor del potencial aplicado en la *puerta* en estado *ON*.
- ***r_ds_on***: Resistencia entre el *drenador* y la *fuelle* en estado *ON*. Se calcula como la división de *v_ds_on* entre *i_d_on* (*Ley de Ohm*).
- ***r_ds_on_25***: Resistencia entre el *drenador* y la *fuelle* en estado *ON* normalizada a temperatura ambiente (25°C). Se calcula aplicando la fórmula 2.1 sobre *r_ds_on*.
- ***r_ds_on_230***: Resistencia entre el *drenador* y la *fuelle* en estado *ON* normalizada a temperatura 230°C. Se calcula aplicando la fórmula (ref) sobre *r_ds_on_25* y fijando a la temperatura que se quiere llevar, en este caso 230°C.
- ***delta_r_ds_on_25***: Incremento de la resistencia a 25°C respecto del primer valor. Se calcula restando todos los elementos de *r_ds_on_25* del primero.
- ***delta_r_ds_on_230***: Incremento de la resistencia a 230°C respecto del primer valor. Se calcula restando todos los elementos de *r_ds_on_230* del primero.

En la *Figura 3.10* se puede ver la resistencia del *Test 36* junto con el resultado de normalizarla a 25°C y a 230°C. En esta figura no parece ser muy importante la normalización

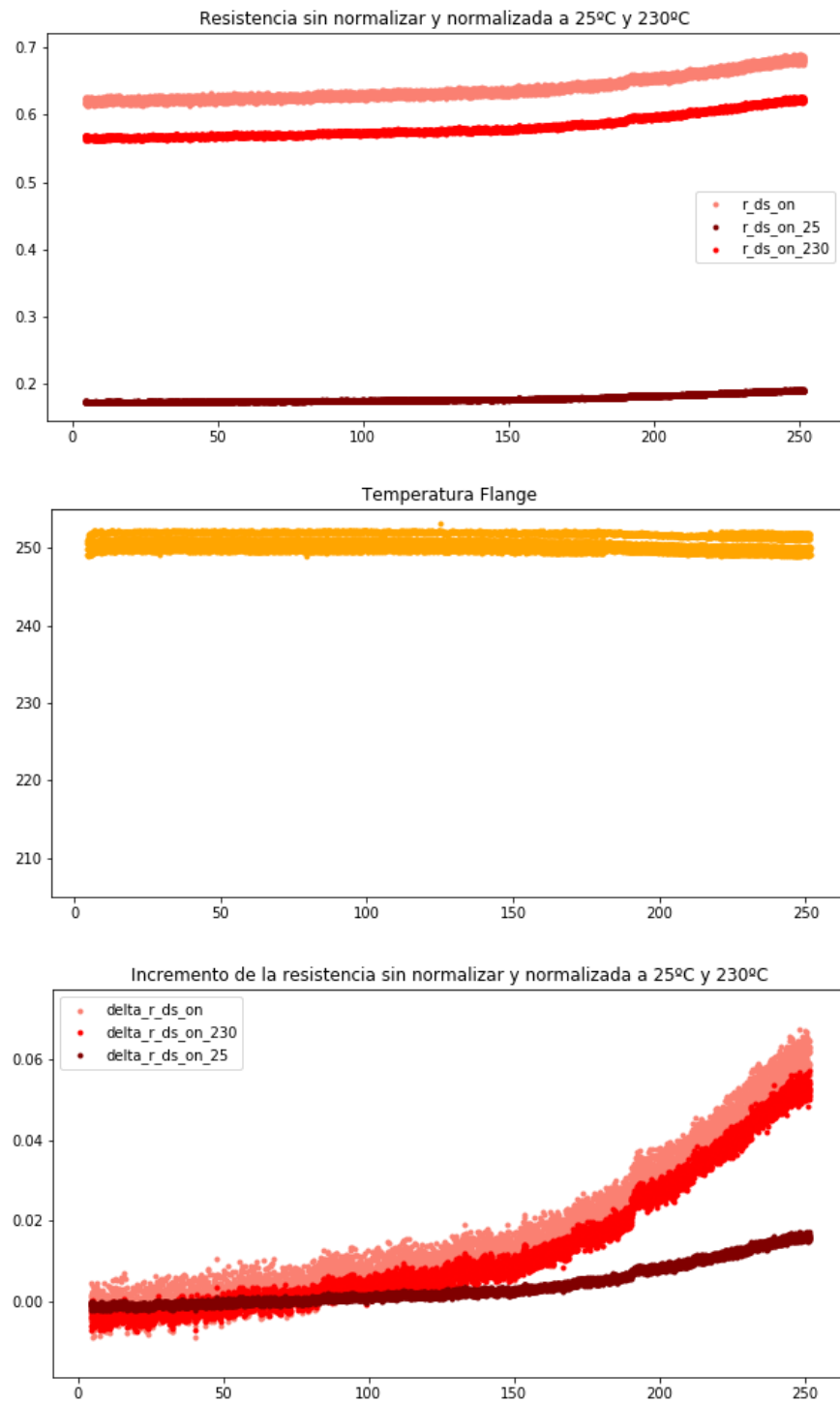


Figura 3.10: Resultados de la normalización de la resistencia y su incremento para el *Test*

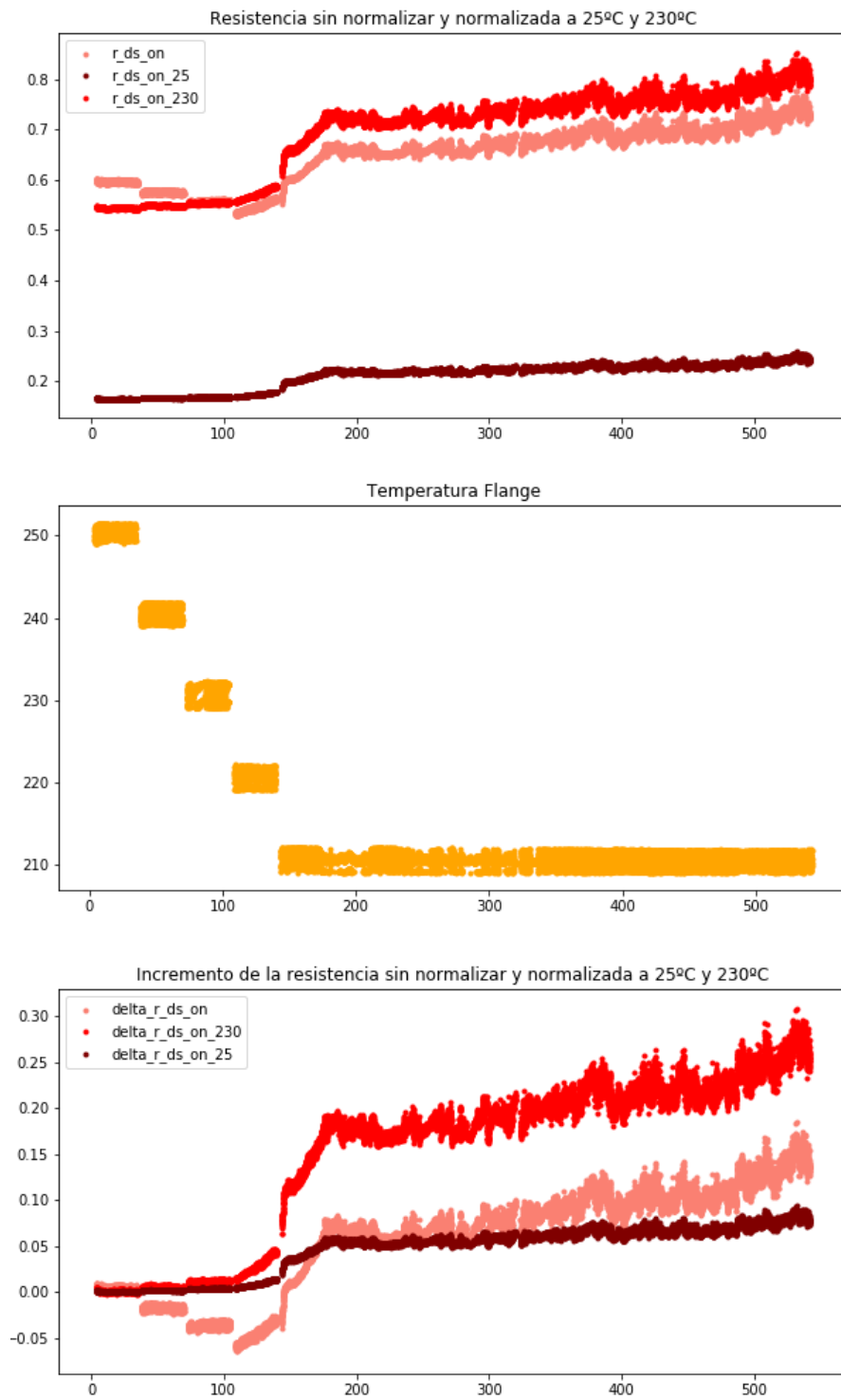


Figura 3.11: Resultados de la normalización de la resistencia y su incremento para el *Test* 8

respecto de la temperatura si no es para desplazar verticalmente los valores de la resistencia (o incremento de la resistencia), esto es así porque este *test* está realizado a la misma temperatura siempre, alrededor de 250°C. Si se observa la *Figura 3.11* correspondiente al *Test 8* se puede ver que esta normalización es mucho más importante. En este *test* se han realizado varias ejecuciones (o *runs*) a distintas temperaturas (250°C, 240°C, 230°C, 220°C y 210°C), por lo tanto, y como se ve en la figura, la resistencia decrece cuando la temperatura también decrece. Sin embargo, si se normaliza la resistencia respecto de la temperatura, en la figura se puede observar que la resistencia tiene una tendencia creciente debido al efecto de la degradación.

En definitiva es necesario quitar la influencia que ejerce la temperatura sobre la resistencia para poder observar realmente el efecto de degradación en el dispositivo. A la hora de elegir que normalización es la mejor para el modelado de datos, se va a escoger la resistencia normalizada a 230°C ya que en los *test*, los dispositivos se encuentran calientes y no a temperatura ambiente.

3.3.3. Media por minuto

En la sección anterior se ha fijado como característica descriptiva de la degradación del dispositivo el incremento de la resistencia respecto de su valor prístino normalizada a temperatura 230°C. Pero antes de proceder al modelado hay que eliminar el ruido del incremento de la resistencia, para ello se va a realizar la media por minuto de los valores de *delta_r_ds_on_230*, calculados en la sección anterior. El resultado de este filtrado es un conjunto de datos con menos ruido y con muchas menos observaciones; esto facilitará el cómputo en la fase de modelado. Aunque pertenece a la fase de modelado de datos, es necesario tener en cuenta que hay que fijar un umbral de incremento de la resistencia en el que se considera que el dispositivo ha fallado. Se fija el umbral en $\Delta R = 0.05$.

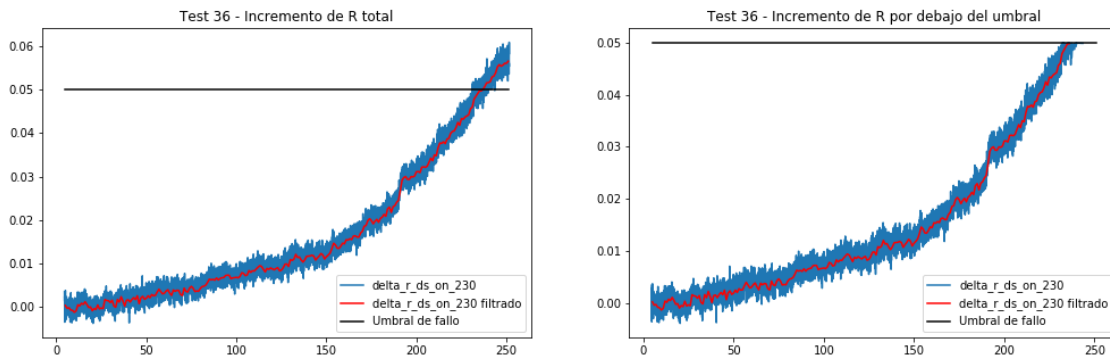
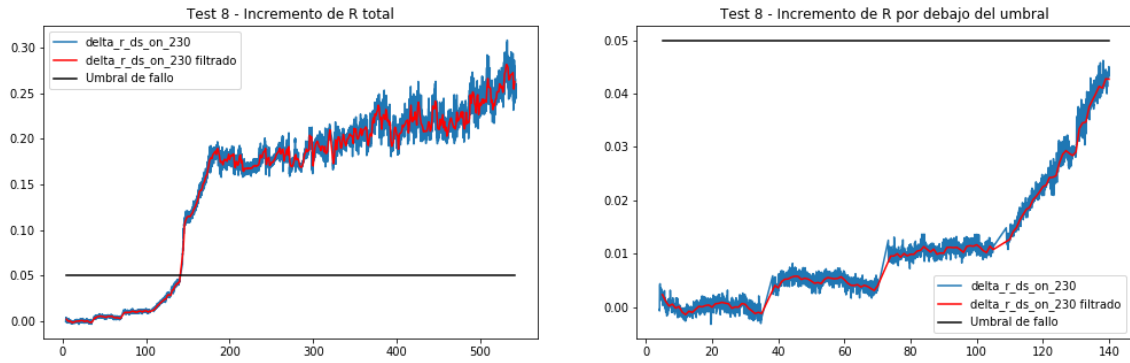
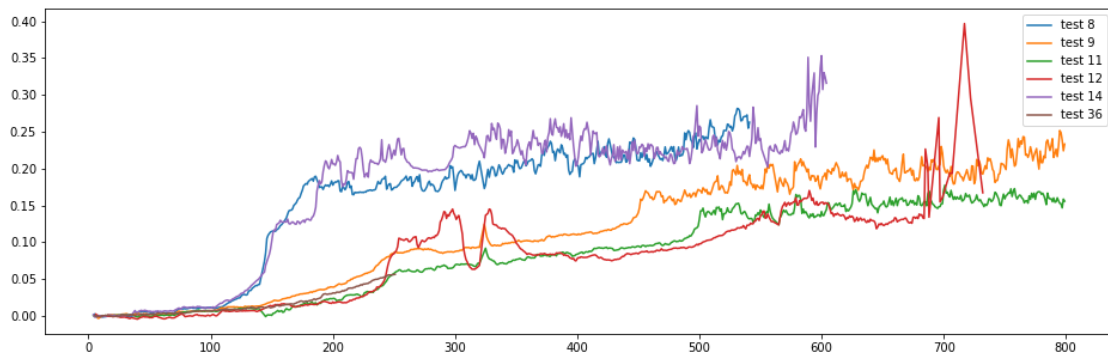
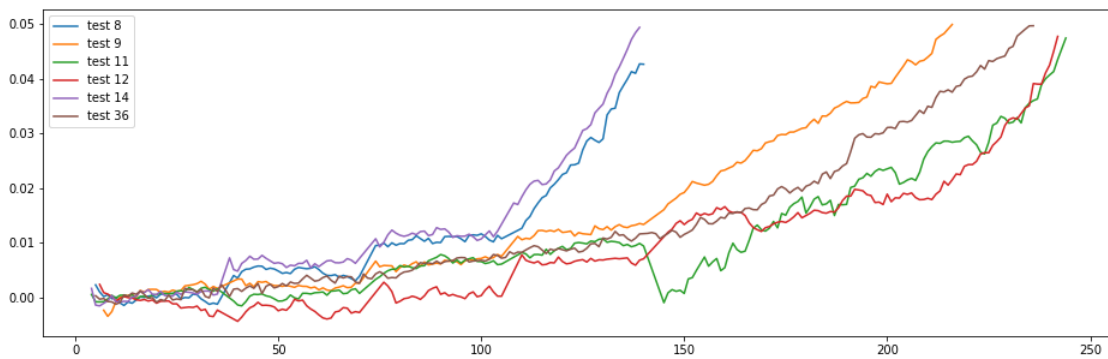


Figura 3.12: ΔR para el *Test 36*

En las figuras 3.12 y 3.13 se puede ver el resultado de aplicar el filtrado de media por minuto tanto para todo el conjunto de datos de ΔR para el *Test 36* y el *Test 8* respectivamente, como sólo para los que están por debajo del umbral. En la *Figura 3.14* se muestra cómo es la forma que muestra el ΔR para un conjunto de experimentos. En la *Figura 3.15* se muestra ΔR hasta alcanzar el valor umbral, para el mismo conjunto de experimentos que en la *Figura 3.14*.

Figura 3.13: ΔR para el *Test 8*Figura 3.14: ΔR para los *Tests 8, 9, 11, 12, 14 y 36* hasta el fin del experimentoFigura 3.15: ΔR para los *Tests 8, 9, 11, 12, 14 y 36* por debajo del umbral de fallo

3.4. Selección de tests

Tras el procesado de los tests se ha realizado una inspección de los resultados de cada test. Las características que se han revisado para la selección de los tests válidos para el modelado han sido la temperatura *flange*, la resistencia entre el *drenador* y la fuente y el incremento de la resistencia entre el *drenador* y la fuente respecto de los valores iniciales. El comportamiento de estas tres características va a determinar si un test es seleccionado para la fase de modelado o no.

- Test 3: **No seleccionado**. Pocos datos. La duración del experimento es de 30 minutos .
- Test 4: **No seleccionado**. Pocos datos. La duración del experimento es de 40 minutos.
- Test 5: **No seleccionado**. Pocos datos. La duración del experimento es de 40 minutos y se quema muy rápido.
- Test 6: **No seleccionado**. Se quema muy rápido, tarda 40 minutos en quemarse.
- Test 7: **No seleccionado**. Se quema muy rápido, tarda 50 minutos en quemarse.
- Test 8: **Seleccionado**. Hay suficientes datos. Tarda 140 minutos en sobrepasar el umbral de ruptura fijado y mas de 500 minutos en finalizar experimento completo. El comportamiento que describe parece razonable hasta el minuto 200 a pesar de un crecimiento abrupto al final.
- Test 9: **Seleccionado**. Hay suficientes datos. Tarda más de 200 minutos en sobrepasar el umbral de ruptura fijado y mas de 800 minutos en finalizar el experimento completo. La curva que describe bien hasta 200 minutos.
- Test 10: **No seleccionado**. Patrón contrario al habitual, no hay casi datos por debajo del umbral de ruptura. Tiene 800 datos en los que o no se quema y mantiene una resistencia muy alta o esta estropeado durante todo el experimento.
- Test 11: **Seleccionado**. Hay suficientes datos. Tarda más de 250 minutos en sobrepasar el umbral de ruptura fijado y mas de 800 minutos en finalizar el experimento completo. La curva que describe está bien a pesar de tener un salto descendente alrededor del minuto 150.
- Test 12: **Seleccionado**. Hay suficientes datos. Tarda algo menos de 250 minutos en sobrepasar el umbral de ruptura fijado y alrededor de 800 minutos en finalizar el experimento completo, aunque se queme alrededor del minuto 700.
- Test 13: **Seleccionado**. Hay suficientes datos. Se quema completamente alrededor del minuto 140.
- Test 14: **Seleccionado**. Hay suficientes datos. Tarda algo menos de 140 minutos en sobrepasar el umbral de ruptura fijado y alrededor de 600 minutos en finalizar el experimento completo, parece que no se quema en todo el experimento.

- Test 15: **No seleccionado**. Pocos datos. La duración del experimento es de 35 minutos.
- Test 16: **No seleccionado**. Pocos datos. La duración del experimento es de 35 minutos.
- Test 17: **No seleccionado**. Pocos datos. La duración del experimento es de 35 minutos.
- Test 18: **No seleccionado**. Pocos datos. La duración del experimento es de 70 minutos y se quema en torno al minuto 35.
- Test 19: **No seleccionado**. Pocos datos. La duración del experimento es de 14 minutos.
- Test 20: **No seleccionado**. El experimento dura alrededor de 200 minutos pero la mayoría del tiempo la resistencia se mantiene por encima del umbral fijado.
- Test 21: **No seleccionado**. Pocos datos. La duración del experimento es de 20 minutos y se quema muy rápidamente.
- Test 22: **No seleccionado**. Pocos datos. La duración del experimento es de 60 minutos.
- Test 23: **No seleccionado**. Hay suficientes datos. El experimento dura mas de 200 minutos pero tiene un crecimiento muy abrupto alrededor del minuto 180 en el que pasa de tener unos valores muy pequeños a sobrepasar el umbral de ruptura fijado, es decir, prácticamente no se degrada sino que se rompe de golpe.
- Test 24: **Seleccionado**. Hay suficientes datos. El experimento dura mas de 200 minutos. El experimento tiene una anomalía los 15 primeros minutos del experimento con un incremento de la resistencia muy alto. Sobrepasa el umbral de ruptura en torno al minuto 180.
- Test 25: **No seleccionado**. A pesar de que hay suficientes datos de experimento (600 minutos), se mantiene alrededor de 400 minutos muy próximo a cero y posteriormente un salto muy abrupto hasta el final del experimento.
- Test 26: **Seleccionado**. Comportamiento habitual de la curva de degradación a pesar de que sobrepasa el umbral de ruptura alrededor del minuto 100 y se quema alrededor del minuto 120.
- Test 27: **No seleccionado**. Suficientes datos pero no sigue el comportamiento habitual de degradación y no se quema.
- Test 28: **No seleccionado**. Pocos datos. La duración de experimento es de 45 minutos.
- Test 29: **Seleccionado**. Hay datos suficientes. El dispositivo se quema en trono al minuto 160 y sobrepasa el umbral de ruptura en trono al minuto 140. Sigue el patrón habitual de degradación.

- Test 30: **No seleccionado**. Hay datos suficientes pero tiene un comportamiento anómalo en trono al minuto 300.
- Test 31: **No seleccionado**. Hay datos suficientes pero no se quema ni sigue el patrón habitual de degradación.
- Test 32: **Seleccionado**. Se quema en torno al minuto 90 y supera el umbral de ruptura en torno al minuto 70. Sigue el patrón habitual de degradación.
- Test 33: **No seleccionado**. Hay suficientes datos pero tiene un comportamiento anómalo entre el minuto 125 y el minuto 200, parece que se quema pero luego sigue funcionando hasta el minuto 375 que es donde parece que finalmente se quema.
- Test 34: **No seleccionado**. No se quema ni alcanza el umbral de degradación.
- Test 35: **Seleccionado**. Hay suficientes datos. Se quema en torno al minuto 125 y sigue el patrón habitual de degradación.
- Test 36: **Seleccionado**. Hay datos suficientes. El patrón de degradación que describe es particularmente bueno.
- Test 37: **Seleccionado**. Hay datos suficientes. Se quema de forma brusca en torno al minuto 250.
- Test 38: **Seleccionado**. Hay datos hasta el minuto 90. Sigue el patrón habitual de degradación.
- Test 39: **No seleccionado**. Pocos datos. La duración del experimento es de 35 minutos. Experimento fallido.
- Test 40: **No seleccionado**. No hay datos. Experimento Fallido.
- Test 41: **No seleccionado**. Hay suficientes datos pero el dispositivo no se quema.
- Test 42: **No seleccionado**. Pocos datos. La duración del experimento es de 35 minutos

3.5. Modelado

En esta sección se van a plantear distintos algoritmos con los que realizar pronósticos y que resultados se obtienen con cada uno de ellos. Se van a usar los tests seleccionados en la sección anterior para realizar los experimentos.

En esta fase de modelado se parte de la hipótesis de que la resistencia entre el *drenador* y la *fuelle* en estado de *puerta* ON se comporta según la función (2.2) descrita en el capítulo anterior. El objetivo de los algoritmos aplicados es calcular los parámetros α y β que mejor se ajusten al comportamiento de la resistencia del dispositivo en cada instante de tiempo. Una vez se tienen los parámetros se realiza una estimación, colocando en la función la resistencia en la que se considera que el dispositivo falla (al igual que en [3] se

fija $\Delta R_{DS(on)} = 0.05$) y despejando el tiempo, que es el instante en el que se predice que el dispositivo va a fallar.

Los métodos que se van a emplear para realizar prognosis de transistores MOSFET son la media horizontal, regresión ajustada o ajuste por mínimos cuadrados y el filtro de Kalman extendido. También se ha probado a linealizar los métodos de la media horizontal y el ajuste por mínimos cuadrados a través de una transformación.

3.5.1. Media horizontal

La media horizontal ha sido el primer método planteado como una aproximación sencilla. Este método consiste en que para todo el conjunto de tests que se tiene, se selecciona uno con el que se valida y el resto se utilizan de entrenamiento. Para cada test de validación, el entrenamiento se compone de tres pasos:

- Ajustar por mínimos cuadrados todos los test de entrenamiento a la función (2.2) obteniendo los parámetros de α y β para cada test.
- Se recorren los valores de $\Delta R_{DS(on)}$ en el intervalo (0,0.05). Cada valor del intervalo se evalúa para cada test, con los parámetros calculados en el paso anterior, y se calculan los distintos tiempos (uno por test) en los que se alcanza ese valor y se hace la media. En resumen, se hace la media los valores del tiempo resultantes al evaluar $\Delta R_{DS(on)}$ en cada uno de los ajustes.
- El tercer paso es ajustar de nuevo por mínimos cuadrados los valores calculados en el apartado anterior para conocer qué α y qué β modelan esa media horizontal.

La predicción del *RUL* consiste en desplazar la media horizontal a lo largo del eje temporal hasta el valor $\Delta R_{DS(on)}$ actual de la observación y ver en que instante de tiempo la función alcanza $\Delta R_{DS(on)} = 0.05$. La diferencia entre el instante en el que la media horizontal sobrepasa el umbral fijado y el instante en el que se realiza la predicción, es el valor del *RUL* predicho.

Los resultados de la experimentación con este método están en las *tablas 3.1* y *3.2* de la *sección 3.6*.

3.5.2. Ajuste por mínimos cuadrados

Como segundo método se ha elegido el ajuste por mínimos cuadrados, que es una técnica de análisis numérico enmarcada dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados (variable independiente, variable dependiente) y una familia de funciones, se intenta encontrar la función continua, dentro de dicha familia, que mejor se aproxime a los datos (que tenga un mejor ajuste), de acuerdo con el criterio de mínimo error cuadrático. En la implementación del ajuste por mínimos cuadrados de *Scipy (Python)* se usa el algoritmo de Levenberg–Marquardt como algoritmo de minimización.

Este método consiste en ajustar por mínimos cuadrados los valores observados a la función 2.2 para calcular los valores de α y β . Una vez se tienen los valores de los parámetros se comprueba en que instante de tiempo la función alcanza $\Delta R_{DS(on)} = 0.05$ y se

predice ese instante como punto de ruptura. La diferencia entre el instante en el que la función sobrepasa el umbral fijado y el instante en el que se realiza la predicción, es el valor del *RUL* predicho.

Los resultados de la experimentación con este método están en las *tablas 3.3 y 3.4* de la *sección 3.6*.

3.5.3. Filtro de Kalman Extendido

Como tercer método se ha usado una implementación del Filtro de Kalman Extendido para estimar los parámetros de α y β . El filtro de Kalman extendido es una versión no lineal del filtro de Kalman que linealiza sobre una estimación de la media y la covarianza actuales. En el filtro de Kalman extendido los modelos de transición y observación de estado no necesitan ser funciones lineales del estado, si no que pueden ser funciones diferenciables.

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \quad (3.1)$$

$$z_k = h(x_k) + v_k \quad (3.2)$$

La forma general de las ecuaciones de estado que admite el filtro de Kalman extendido esta descrita por las ecuaciones 3.1 y 3.2. Donde w_k y v_k son los ruidos del proceso y observación, se supone que estos ruidos siguen una distribución Gausiana multivariante de media cero con covarianza Q_r y R_k respectivamente. El vector u_k es el vector de control.

La función f puede usarse para calcular el estado predicho a partir de la estimación previa y, de manera similar, la función h puede usarse para calcular la medida predicha a partir del estado predicho. Sin embargo, f y h no pueden aplicarse directamente a la covarianza, lo que se hace es calcular una matriz de derivadas parciales (jacobiana).

En cada paso de tiempo, el la matriz jacobiana se evalúa con los estados predichos actuales. Estas matrices se pueden usar en las ecuaciones de filtro de Kalman. Este proceso esencialmente lo que hace es linealizar la función no lineal alrededor de la estimación actual. Igual que en los métodos anteriores se utilizan los parámetros de α y β para calcular en qué instante de tiempo la función alcanza $\Delta R_{DS(on)} = 0.05$ y se predice ese instante como punto de ruptura. En este caso también, la diferencia entre el instante en el que la función sobrepasa el umbral fijado y el instante en el que se realiza la predicción, es el valor del *RUL* predicho.

Para este caso concreto se ha realizado una implementación propia a partir de la definición en el artículo [6]. Se ha inicializado los parámetros de α y β con los valores obtenidos en la media horizontal y se han utilizado valores para la covarianza muy pequeños. En esta implementación se ha usado el modelo de degradación de la ecuación (2.2), usando como variable de estado $x = \{R, \alpha, \beta\}$ donde $R = \Delta R_{DS(on)}$, por lo tanto f es una función de valor vectorial. La ecuación de transición para la variable R está descrita por

$$\frac{dR}{dt} = R\beta + \alpha\beta \quad (3.3)$$

considerando α y β constantes pero con la necesidad de ser estimadas, por lo tanto $\frac{d\alpha}{dt} = \frac{d\beta}{dt} = 0$ como parte de la función de transición f .

Los resultados de la experimentación con este método están en las *tablas 3.5 y 3.6* de la *sección 3.6*.

3.5.4. Ajuste por mínimos cuadrados para el modelo linealizado

Lo que se plantea en este método es una reparametrización de la función (2.2) que modela el comportamiento de $\Delta R_{DS(on)}$. La motivación de realizar una reparametrización sobre la ecuación (2.2) es encontrar un modelo lineal en el que se puedan calcular fácilmente los intervalos de confianza para las predicciones. La reparametrización propuesta es de la forma (3.4)

$$\Delta R_{DS(on)} = \alpha' e^{\beta t} \quad (3.4)$$

Pero como esta reparametrización no es lineal, es necesario realizar una transformación tomando logaritmos (3.5).

$$\ln(\Delta R_{DS(on)}) = \ln(\alpha') + \beta t \quad (3.5)$$

dónde la relación entre α y α' es (3.6).

$$\alpha' e^{\beta t} = \alpha(e^{\beta t} - 1) \quad (3.6)$$

En este método se plantea realizar los mismos pasos que el método de la *sección 3.5.2* pero para el modelo lineal (3.4). Se ajusta por mínimos cuadrados la función calculando en este caso α' y β . Una vez se tienen los valores de los parámetros se comprueba en qué instante de tiempo la función alcanza $\ln(\Delta R_{DS(on)}) = \ln(0,05)$ y se predice ese instante como punto de ruptura. La diferencia entre el instante en el que la función sobrepasa el umbral fijado y el instante en el que se realiza la predicción, es el valor del *RUL* predicho.

Como ya se ha comentado, la motivación de reparametrizar el modelo es el cálculo de los intervalos de confianza. El cálculo de los intervalos se puede realizar tal y como se indica en [7]. Se pueden calcular dos tipos de intervalos, el intervalo de confianza para un valor medio de la variable o el intervalo para un valor individual.

En la *figura 3.16* se pueden observar los intervalos de confianza para un valor medio de la variable $\ln(\Delta R_{DS(on)})$ (en verde) y los intervalos de confianza para un valor individual de la variable $\ln(\Delta R_{DS(on)})$.

Los resultados de la experimentación con este método están en las *tablas 3.3 y 3.4* de la *sección 3.6*.

3.5.5. Media horizontal para el modelo linealizado

Como la *sección 3.5.4* plantea un modelo lineal a partir de una transformación de la ecuación (2.2) se va a probar como funciona el método de la media horizontal explicado en la *sección 3.5.1* con esta nueva ecuación.

Los resultados de la experimentación con este método están en las *tablas 3.9 y 3.10* de la *sección 3.6*.

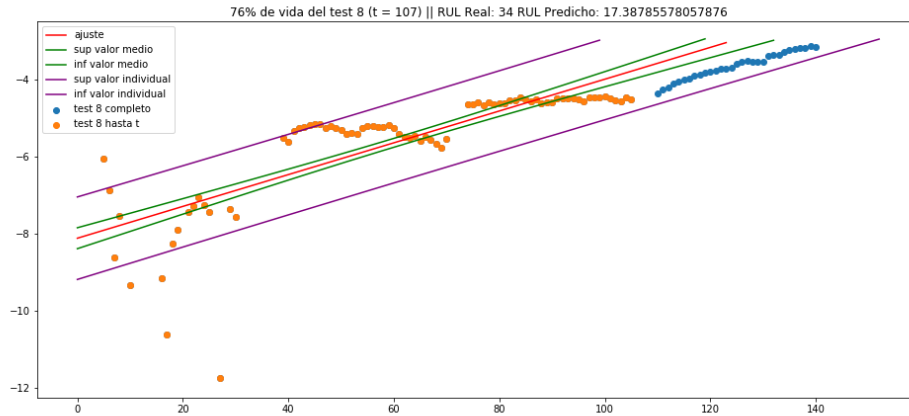


Figura 3.16: Intervalos de confianza para el ajuste al modelo linealizado para el *Test 8*

3.6. Resultados de la estimación de *RUL*

En esta sección se presentan en forma de tabla los resultados obtenidos con cada uno de los métodos para los test seleccionados en la *sección 3.4*. Como ningún test tiene la misma duración se ha optado por realizar las predicciones en función del porcentaje de vida consumida (55 %, 59 %, 64 %, 68 %, 72 %, 76 %, 81 %, 83 %, 85 %, 87 %, 89 %, 90 % y 95 %). La vida que ha tenido cada test en minutos se aporta debajo del nombre de cada test en la tabla.

La métrica empleada para la evaluación de los tests es la diferencia entre el *RUL* real del dispositivo y el *RUL* predicho por cada método. Cuánto mas próximo a cero sea el valor de la métrica mejor es la predicción del *RUL*.

Los resultados para cada test seleccionado por cada uno de los métodos descritos anteriormente se muestran a continuación. Además se exponen las gráficas que describen la predicción, en naranja el los valores observados, en azul los valores por observar y en rojo el ajuste. En el caso del filtro de Kalman extendido se añade la predicción para $\Delta R_{DS(on)}$ que realiza el método, y en el caso del ajuste por mínimos cuadrados para el modelo linealizado se incluyen las bandas del intervalo de confianza para la predicción.

min. vivos	Test 8 141 min.	Test 9 217 min.	Test 11 245 min.	Test 12 243 min.	Test 13 65 min.	Test 14 140 min.	Test 24 189 min.
55 %	-5.77	39.35	45.74	30.51	-1.74	-6.13	9.77
59 %	-9.77	28.46	32.63	20.56	-5.15	-6.86	17.57
64 %	-18.20	23.67	11.59	37.74	-8.15	-12.71	12.90
68 %	-19.10	25.32	6.66	29.23	10.83	-19.50	1.51
72 %	-23.35	23.79	19.12	15.26	11.11	-27.74	-4.47
76 %	-32.65	23.48	14.32	11.01	9.01	-27.59	2.94
81 %	-22.38	20.24	14.49	5.94	7.58	-11.86	-4.45
83 %	-19.40	16.78	9.11	1.12	6.08	-14.97	-3.40
85 %	-16.91	14.02	1.47	-3.94	3.11	-13.04	-1.27
87 %	-15.70	13.52	7.98	-10.54	2.01	-11.93	6.16
89 %	-15.46	9.68	3.34	-10.42	1.62	-10.75	1.80
90 %	-12.36	8.60	2.35	-8.51	1.73	-7.87	2.81
95 %	-8.58	5.10	-5.73	-6.64	0.03	-4.39	0.47

Tabla 3.1: Diferencia entre el *RUL* real y predicho. Resultados para la **media horizontal**.
Tests 8-24

min. vivos	Test 26 99 min.	Test 29 136 min.	Test 32 77 min.	Test 35 145 min.	Test 36 237 min.	Test 37 245 min.	Test 38 72 min.
55 %	-88.72	-29.49	-51.48	-21.40	38.19	46.25	-58.62
59 %	-57.12	-56.34	-53.72	-20.96	35.17	43.26	-54.13
64 %	-55.27	-30.95	-49.17	-16.33	25.58	42.34	-53.51
68 %	-58.42	-25.98	-39.55	-14.82	24.65	35.00	-53.99
72 %	-54.37	-33.47	-39.16	-14.22	22.06	29.57	-38.99
76 %	-39.51	-19.17	-40.20	-16.00	17.58	22.18	-36.68
81 %	-44.08	11.23	-36.41	-3.77	16.05	16.79	-39.50
83 %	-49.80	8.27	-36.76	-0.64	18.56	12.64	-40.76
85 %	-42.81	9.99	-34.65	1.00	15.97	8.11	-32.75
87 %	-48.25	6.32	-41.10	-1.18	12.31	4.80	-33.91
89 %	-49.92	6.53	-40.42	0.55	10.30	3.00	-29.91
90 %	-44.59	4.29	-38.42	2.90	10.78	0.87	-29.91
95 %	-34.51	1.98	-11.34	0.76	4.40	-8.13	-16.93

Tabla 3.2: Diferencia entre el *RUL* real y predicho. Resultados para la **media horizontal**.
Tests 26-38

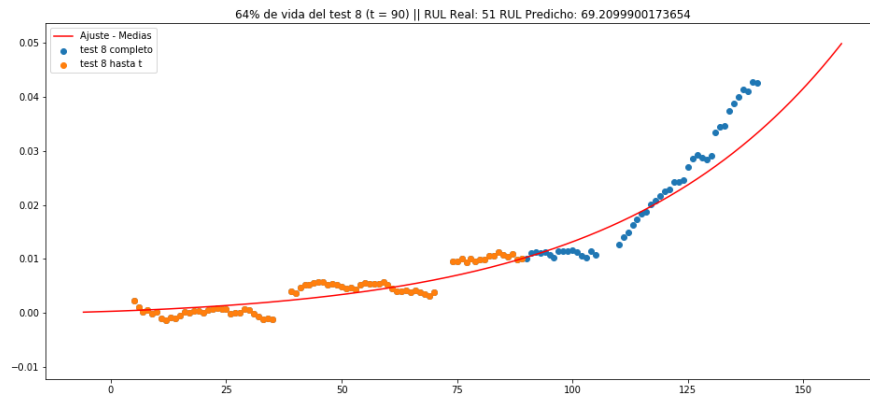


Figura 3.17: **Media horizontal:** Predicción en el 64 % (t=90) de la vida del dispositivo del *Test 8*

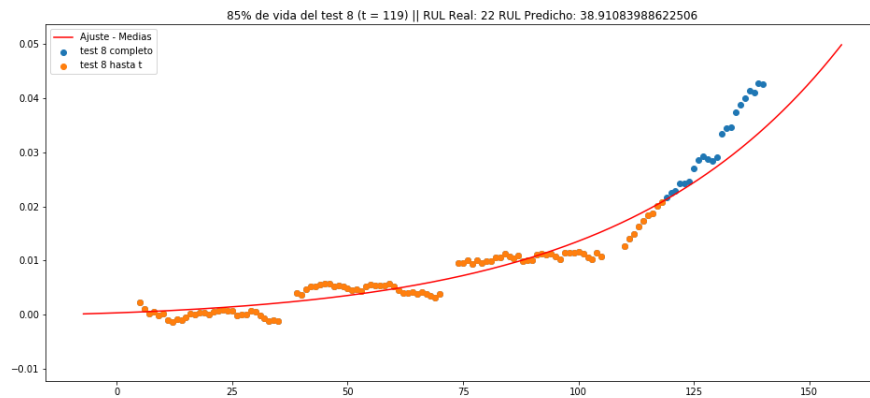


Figura 3.18: **Media horizontal:** Predicción en el 85 % (t=119) de la vida del dispositivo del *Test 8*

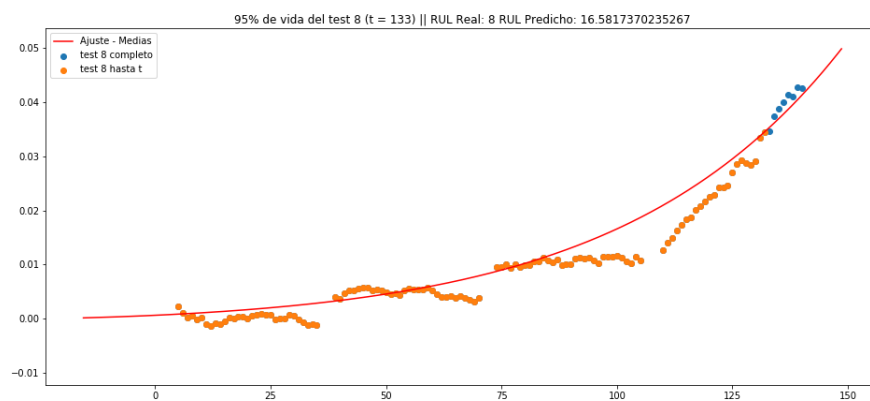


Figura 3.19: **Media horizontal:** Predicción en el 95 % (t=133) de la vida del dispositivo del *Test 8*

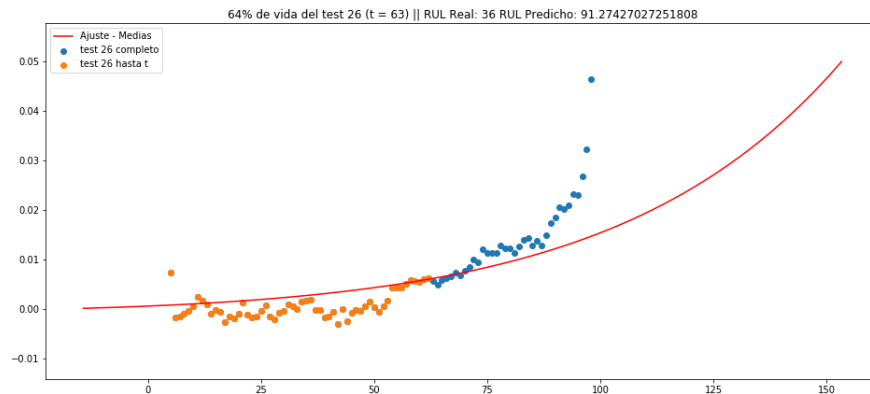


Figura 3.20: **Media horizontal:** Predicción en el 64 % ($t=63$) de la vida del dispositivo del *Test 26*

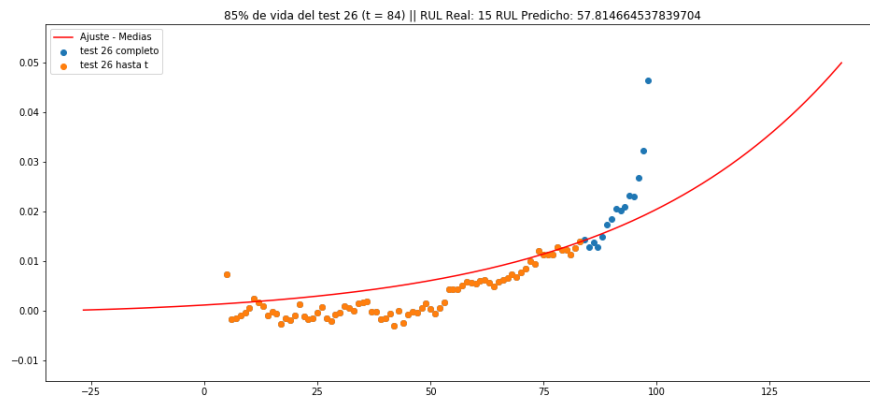


Figura 3.21: **Media horizontal:** Predicción en el 85 % ($t=84$) de la vida del dispositivo del *Test 26*

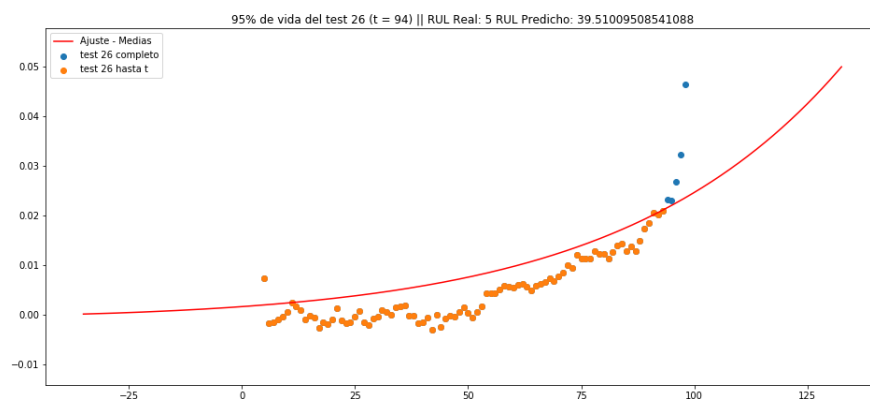


Figura 3.22: **Media horizontal:** Predicción en el 95 % ($t=94$) de la vida del dispositivo del *Test 26*

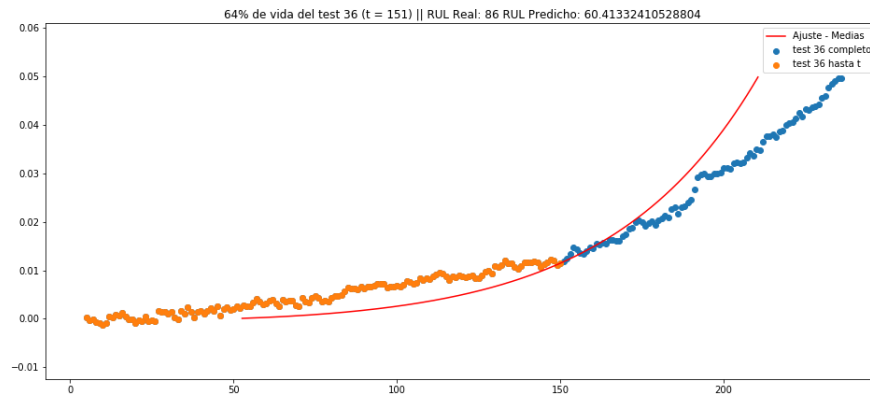


Figura 3.23: **Media horizontal:** Predicción en el 64 % (t=151) de la vida del dispositivo del *Test 36*

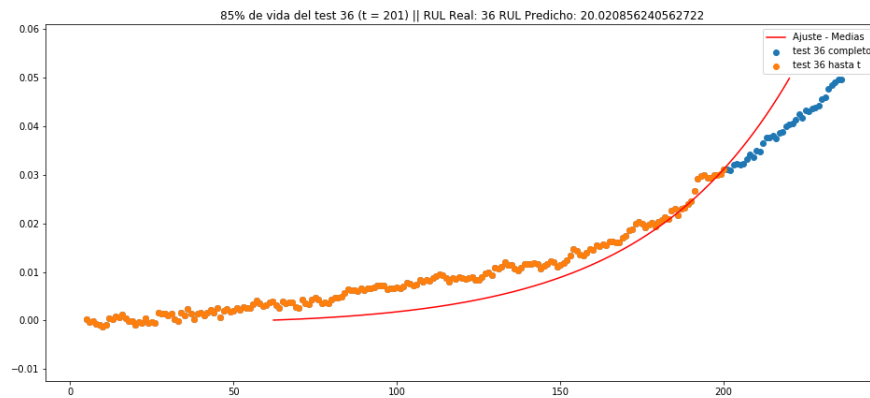


Figura 3.24: **Media horizontal:** Predicción en el 85 % (t=201) de la vida del dispositivo del *Test 36*

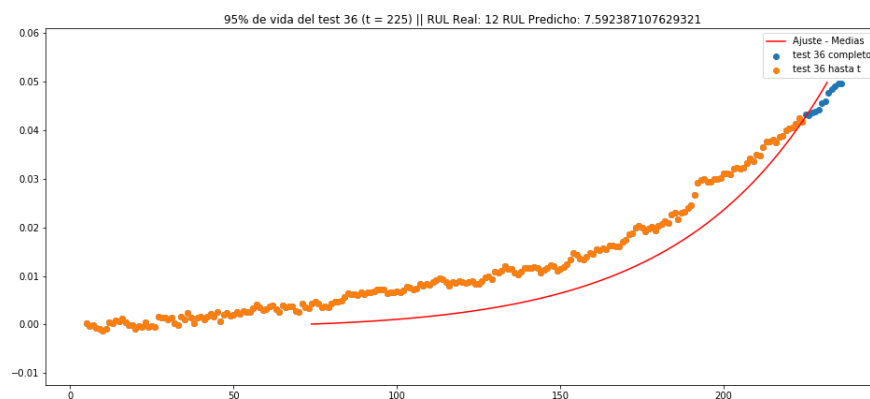


Figura 3.25: **Media horizontal:** Predicción en el 95 % (t=225) de la vida del dispositivo del *Test 36*

min. vivididos	Test 8 141 min.	Test 9 217 min.	Test 11 245 min.	Test 12 243 min.	Test 13 65 min.	Test 14 140 min.	Test 24 189 min.
55 %	-8.39	34.21	33.65	79.96	27.27	-24.98	66.00
59 %	4.49	18.73	10.77	62.66	25.07	-11.18	53.65
64 %	-3.66	1.56	-558.56	59.71	25.07	-22.50	43.16
68 %	-13.46	-3.59	-705.07	49.85	21.03	-39.87	30.88
72 %	-29.03	6.52	-224.29	30.24	19.58	-70.52	17.35
76 %	-48.97	9.78	-67.56	13.96	17.44	-103.39	13.49
81 %	-49.90	12.35	-25.43	1.07	15.22	-56.71	2.85
83 %	-40.10	11.58	-16.14	-6.27	14.43	-41.51	0.65
85 %	-33.37	10.46	-19.40	-13.54	12.72	-34.25	-1.60
87 %	-25.46	9.40	-14.76	-21.81	11.84	-28.98	1.42
89 %	-20.16	7.91	-11.66	-28.33	11.00	-21.97	2.68
90 %	-18.21	7.02	-11.43	-29.56	10.22	-17.56	2.83
95 %	-10.37	3.91	-14.87	-28.40	8.00	-7.76	1.78

Tabla 3.3: Diferencia entre el *RUL* real y predicho. Resultados para **ajuste por mínimos cuadrados**. *Tests 8-24*

min. vivididos	Test 26 99 min.	Test 29 136 min.	Test 32 77 min.	Test 35 145 min.	Test 36 237 min.	Test 37 245 min.	Test 38 72 min.
55 %	-48918.05	-2.67	-11.35	-8.27	-17.85	9.33	27.40
59 %	35.30	-33.20	-37.68	3.59	-21.08	25.57	22.57
64 %	25.44	-19.54	-52.85	5.49	-43.28	26.27	14.91
68 %	16.11	-6.86	-43.19	6.58	-44.33	23.40	11.52
72 %	9.78	-19.15	-28.64	4.91	-37.95	14.67	10.06
76 %	7.48	-16.84	-29.33	1.80	-24.04	6.26	7.64
81 %	2.12	20.98	-28.04	5.25	-16.24	-2.70	1.11
83 %	-1.36	20.58	-27.32	6.44	-6.16	-5.47	-0.38
85 %	-3.73	19.18	-25.25	7.38	-1.85	-9.73	-1.34
87 %	-6.44	16.97	-27.19	7.21	-0.01	-13.92	-2.24
89 %	-9.75	15.14	-29.87	6.83	0.57	-17.62	-3.23
90 %	-10.82	14.39	-30.40	6.96	1.06	-18.66	-3.23
95 %	-11.29	9.87	-10.55	5.45	1.92	-25.52	-2.64

Tabla 3.4: Diferencia entre el *RUL* real y predicho. Resultados para **ajuste por mínimos cuadrados**. *Tests 26-38*

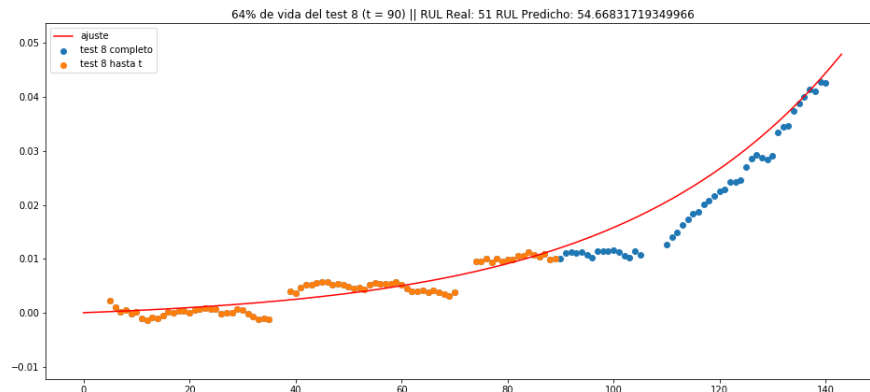


Figura 3.26: **Mínimos cuadrados:** Predicción en el 64 % (t=90) de la vida del dispositivo del *Test 8*

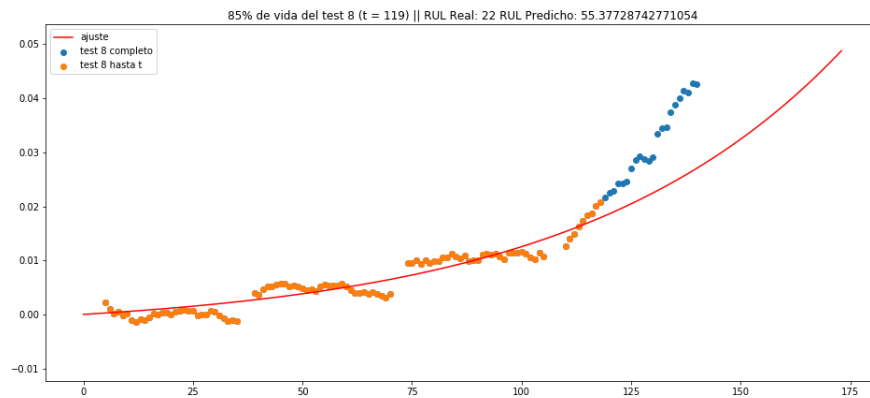


Figura 3.27: **Mínimos cuadrados:** Predicción en el 85 % (t=119) de la vida del dispositivo del *Test 8*

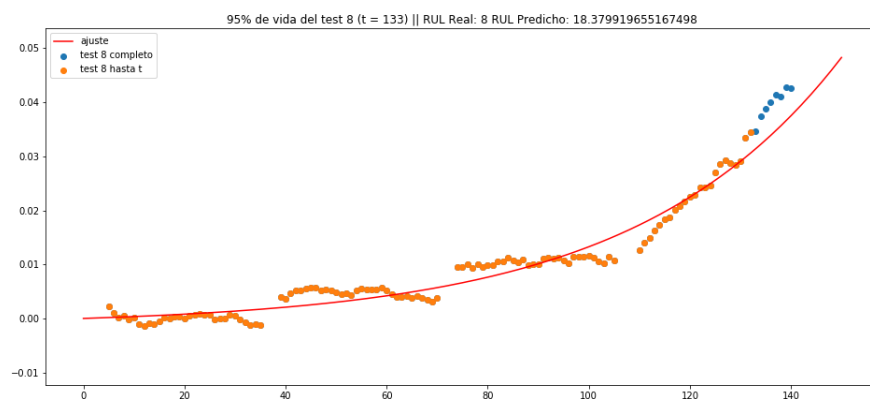


Figura 3.28: **Mínimos cuadrados:** Predicción en el 95 % (t=133) de la vida del dispositivo del *Test 8*

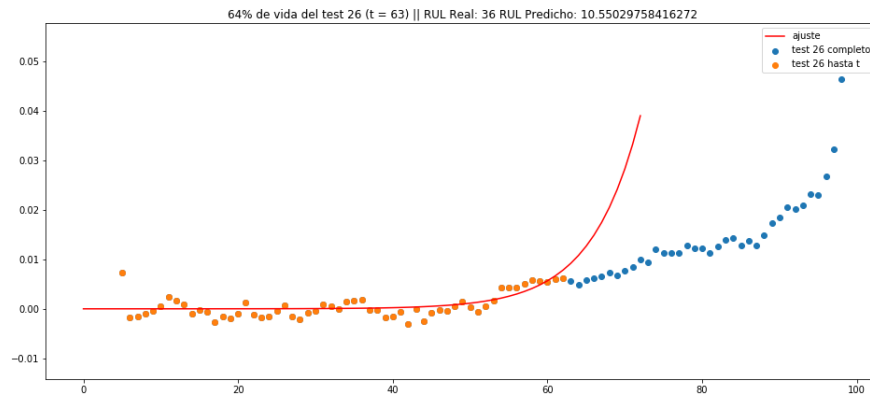


Figura 3.29: **Mínimos cuadrados:** Predicción en el 64 % ($t=63$) de la vida del dispositivo del *Test 26*

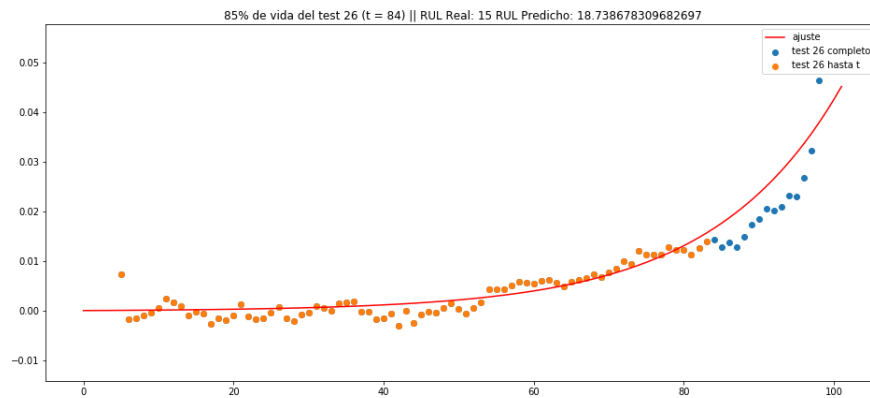


Figura 3.30: **Mínimos cuadrados:** Predicción en el 85 % ($t=84$) de la vida del dispositivo del *Test 26*

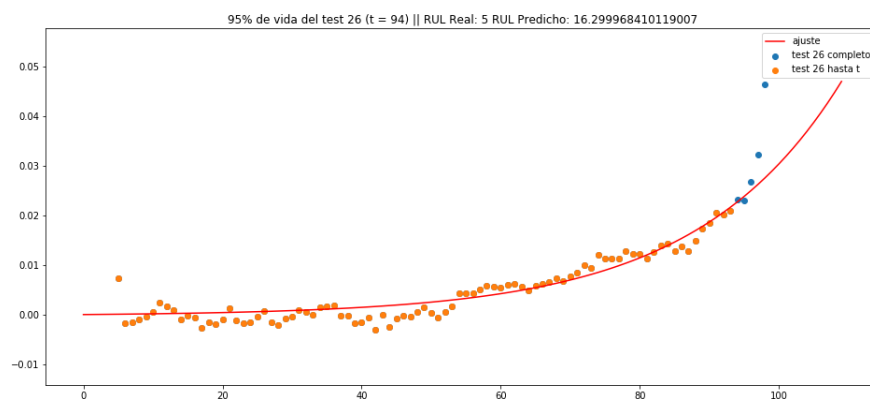


Figura 3.31: **Mínimos cuadrados:** Predicción en el 95 % ($t=94$) de la vida del dispositivo del *Test 26*

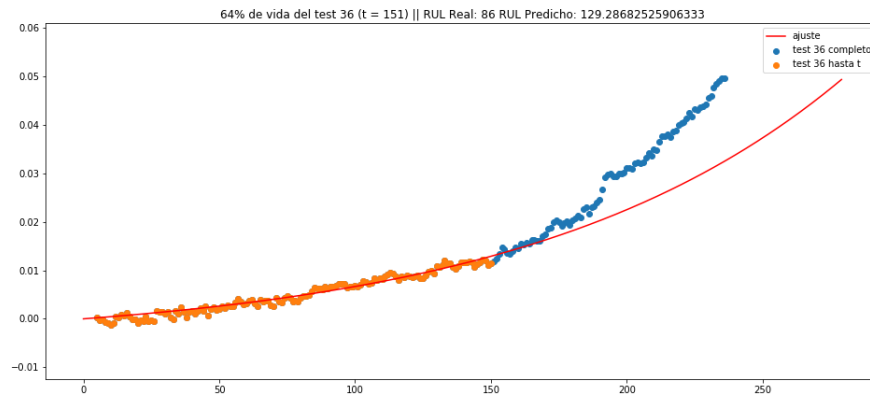


Figura 3.32: **Mínimos cuadrados:** Predicción en el 64 % ($t=151$) de la vida del dispositivo del *Test 36*

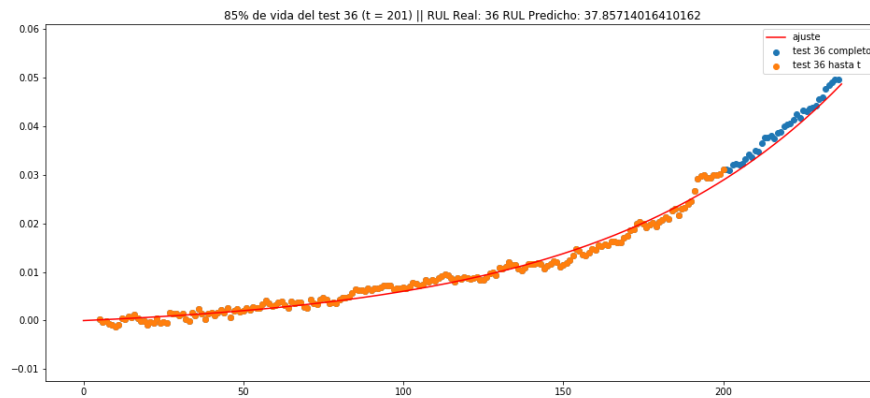


Figura 3.33: **Mínimos cuadrados:** Predicción en el 85 % ($t=201$) de la vida del dispositivo del *Test 36*

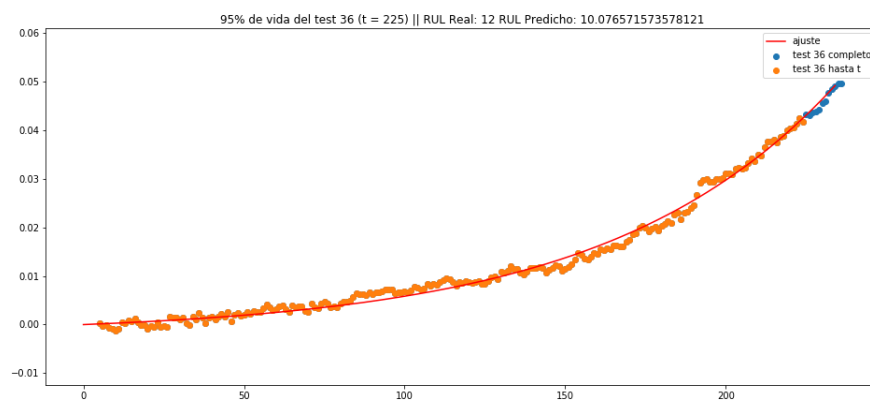


Figura 3.34: **Mínimos cuadrados:** Predicción en el 95 % ($t=225$) de la vida del dispositivo del *Test 36*

min. vivididos	Test 8 141 min.	Test 9 217 min.	Test 11 245 min.	Test 12 243 min.	Test 13 65 min.	Test 14 140 min.	Test 24 189 min.
55 %	-24.50	58.16	73.02	68.14	-87.33	-29.03	-0.97
59 %	-19.37	49.93	55.67	63.83	-83.10	-20.75	-10.04
64 %	-18.68	36.56	-35.01	86.71	-74.26	-28.00	-14.54
68 %	-27.88	26.71	-15.68	81.80	-75.69	-31.37	-34.05
72 %	-31.94	47.47	8.56	20.61	-75.69	-43.43	-37.19
76 %	-37.46	37.06	11.63	16.91	-45.25	-49.40	-15.21
81 %	-44.75	42.28	26.03	14.65	-55.34	-25.99	-52.13
83 %	-30.63	30.93	24.05	15.60	-56.33	-19.08	-54.29
85 %	-22.41	22.26	-68.57	-50.43	-59.37	-17.99	-66.67
87 %	-15.75	5.05	-50.30	-38.27	-66.17	-29.29	-0.20
89 %	-7.61	-5.08	8.34	-90.17	-68.66	-19.75	20.85
90 %	-5.26	-1.04	-0.48	-101.62	-70.70	-12.31	20.31
95 %	7.15	-23.62	-30.47	-3.43	-89.70	3.50	-10.28

Tabla 3.5: Diferencia entre el *RUL* real y predicho. Resultados para el **filtro de Kalman extendido**. Tests 8-24

min. vivididos	Test 26 99 min.	Test 29 136 min.	Test 32 77 min.	Test 35 145 min.	Test 36 237 min.	Test 37 245 min.	Test 38 72 min.
55 %	-64.35	-30,18	-85.01	-42.93	52.48	72,19	-86,92
59 %	-64.81	-33,11	-87.08	-42.62	62.09	75,89	-86,04
64 %	-61.68	-34,37	-88.37	-42.22	34.63	69,30	-85,49
68 %	-61.45	-33,81	-84.51	-38.78	38.64	65,11	-85,49
72 %	-63.08	-30,67	-87.24	-35.22	24.64	35,26	-84,71
76 %	-62.57	-35,34	-82.56	-37.95	40.98	21,61	-80,97
81 %	-52.02	13,50	-85.74	-39.84	23.27	-4,57	-75,26
83 %	-57.08	45,60	-84.16	-17.77	15.85	2,66	-76,97
85 %	-53.52	41,50	-82.26	-13.35	39.29	-20,50	-84,38
87 %	-58.50	26,18	-82.45	-7.70	26.19	-47,98	-80,25
89 %	-59.46	14,81	-80.69	-4.18	11.26	-60,82	-81,52
90 %	-55.81	5,69	-81.22	-7.49	3.82	-84,35	-81,52
95 %	-59.21	-3,94	-86.80	-6.10	-5.08	-92,78	-68,65

Tabla 3.6: Diferencia entre el *RUL* real y predicho. Resultados para el **filtro de Kalman extendido**. Tests 26-38

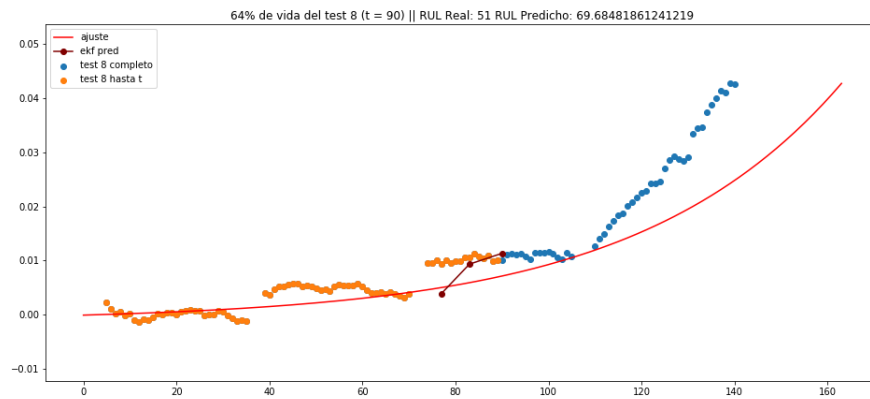


Figura 3.35: **Filtro de Kalman extendido:** Predicción en el 64 % ($t=90$) de la vida del dispositivo del *Test 8*

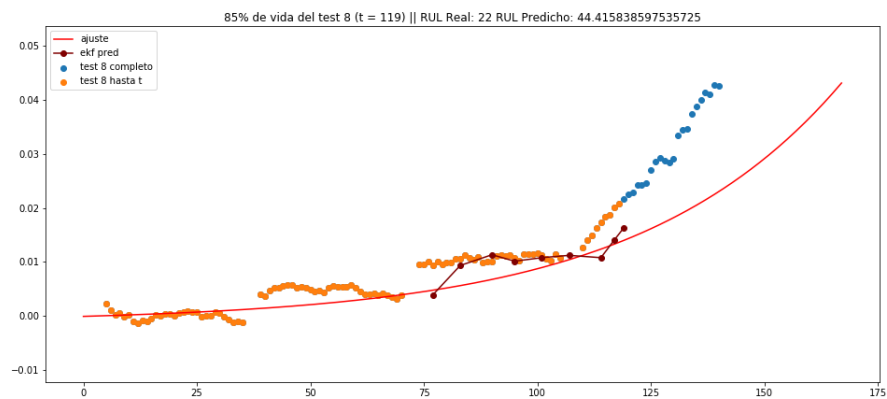


Figura 3.36: **Filtro de Kalman extendido:** Predicción en el 85 % ($t=119$) de la vida del dispositivo del *Test 8*

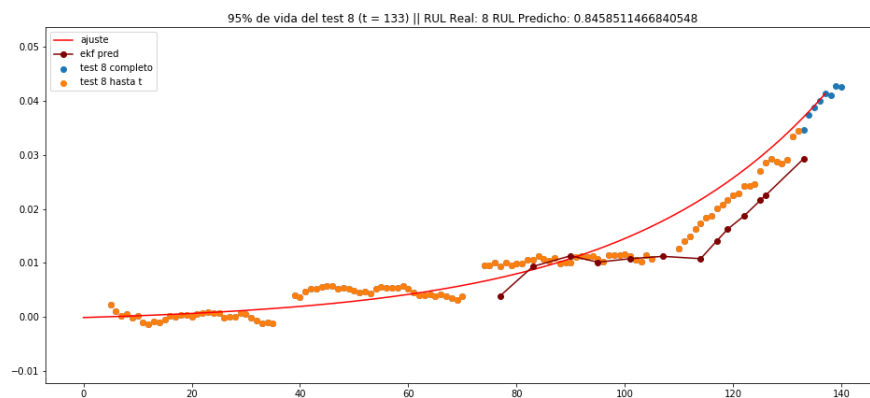


Figura 3.37: **Filtro de Kalman extendido:** Predicción en el 95 % ($t=133$) de la vida del dispositivo del *Test 8*

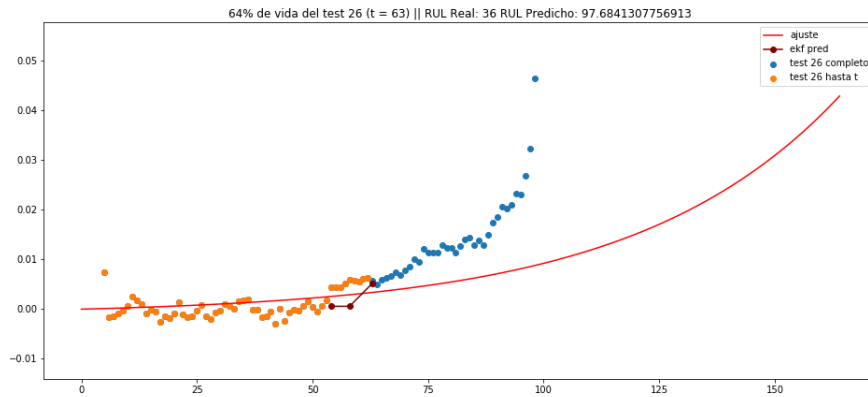


Figura 3.38: **Filtro de Kalman extendido:** Predicción en el 64 % (t=63) de la vida del dispositivo del *Test 26*

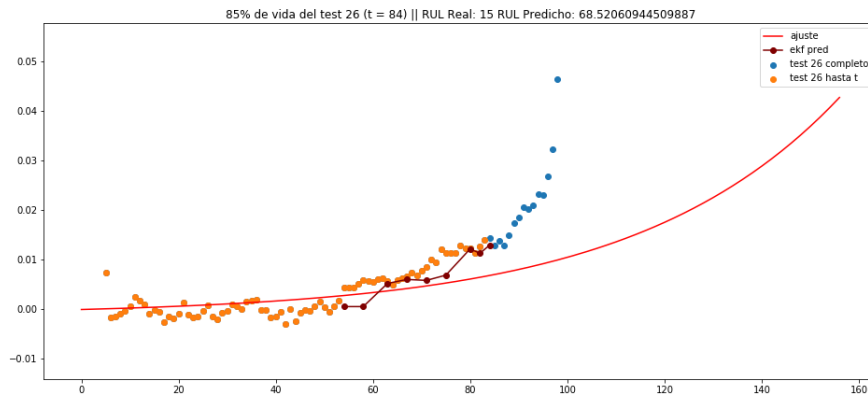


Figura 3.39: **Filtro de Kalman extendido:** Predicción en el 85 % (t=84) de la vida del dispositivo del *Test 26*

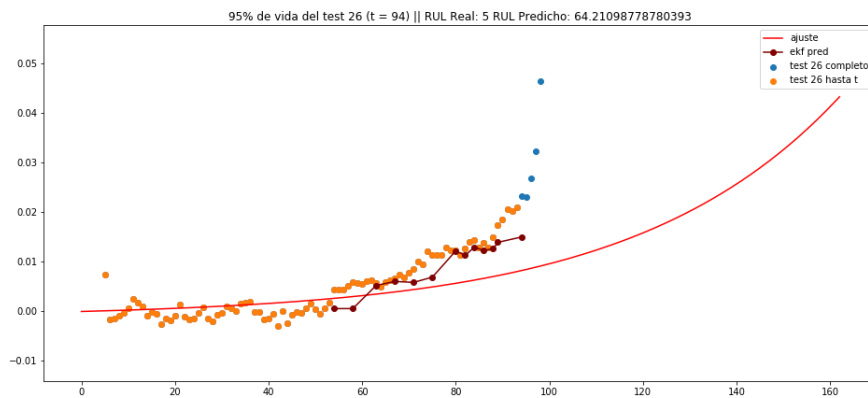


Figura 3.40: **Filtro de Kalman extendido:** Predicción en el 95 % (t=94) de la vida del dispositivo del *Test 26*

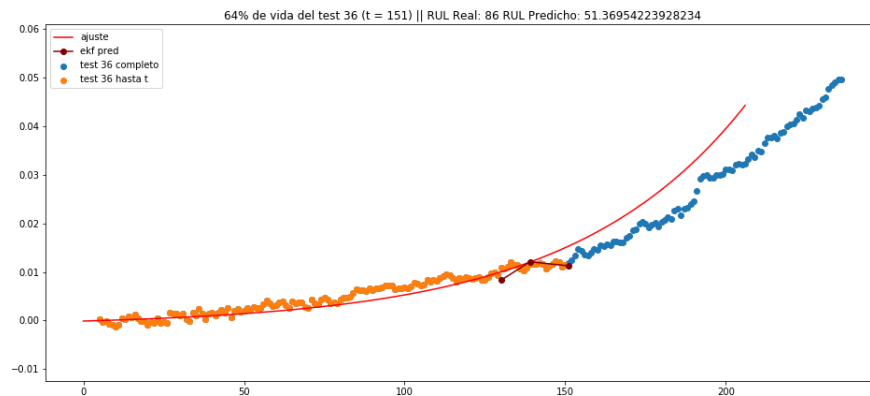


Figura 3.41: **Filtro de Kalman extendido:** Predicción en el 64 % ($t=151$) de la vida del dispositivo del *Test 36*

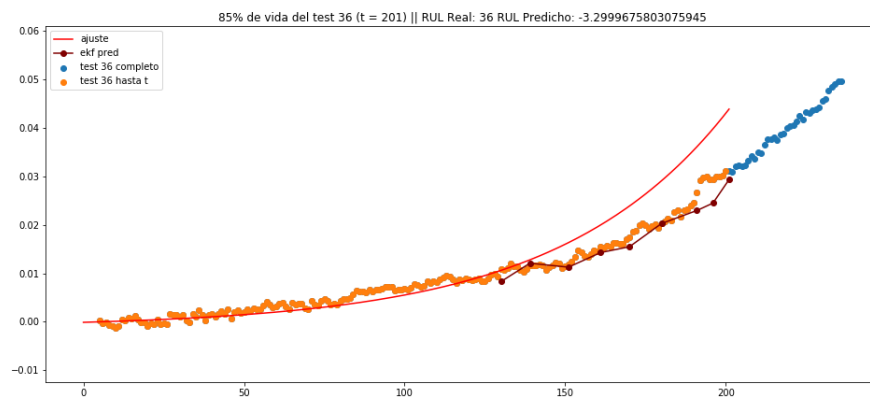


Figura 3.42: **Filtro de Kalman extendido:** Predicción en el 85 % ($t=201$) de la vida del dispositivo del *Test 36*

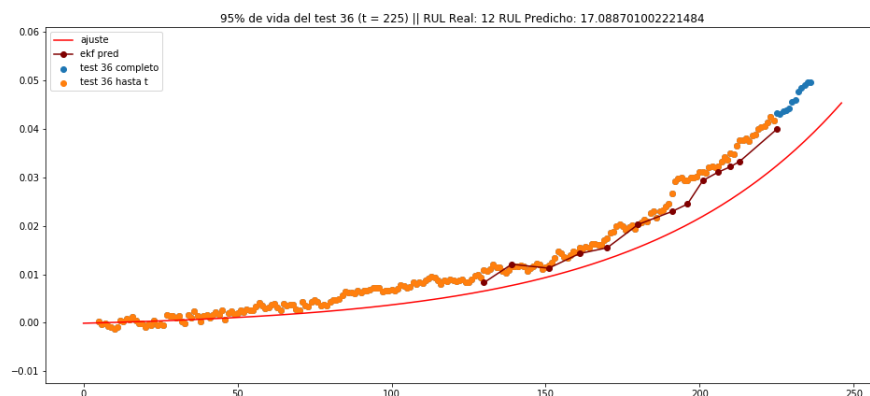


Figura 3.43: **Filtro de Kalman extendido:** Predicción en el 95 % ($t=225$) de la vida del dispositivo del *Test 36*

min. vivididos	Test 8 141 min.	Test 9 217 min.	Test 11 245 min.	Test 12 243 min.	Test 13 65 min.	Test 14 140 min.	Test 24 189 min.
55 %	39.29	49.80	69.59	4.58	26.57	43.74	-1111.64
59 %	35.82	45.59	63.97	16.47	26.32	39.69	-215.75
64 %	30.87	39.84	24.66	31.84	26.32	33.25	-83.10
68 %	26.86	36.89	10.65	37.48	24.45	27.43	-51.57
72 %	21.60	33.89	8.86	34.31	23.25	21.72	-40.91
76 %	16.61	31.49	9.10	29.96	21.46	16.09	-28.90
81 %	12.89	28.70	7.77	24.60	19.66	12.82	-23.03
83 %	10.95	27.31	7.49	21.71	19.05	11.22	-21.69
85 %	9.89	26.13	5.97	18.66	17.79	9.57	-20.11
87 %	8.49	24.95	5.37	15.27	17.15	8.68	-16.41
89 %	7.24	23.45	4.80	12.04	16.52	7.54	-13.60
90 %	6.90	22.78	4.47	10.96	15.89	6.94	-12.50
95 %	4.73	19.38	1.68	5.79	14.03	5.27	-9.15

Tabla 3.7: Diferencia entre el *RUL* real y predicho. Resultados para **mínimos cuadrados con el modelo linealizado**. *Tests 8-24*

min. vivididos	Test 26 99 min.	Test 29 136 min.	Test 32 77 min.	Test 35 145 min.	Test 36 237 min.	Test 37 245 min.	Test 38 72 min.
55 %	-3.32E+20	-42.44	-157.48	13.81	57.73	59.46	-27.58
59 %	-924.42	-40.23	-181.05	14.00	51.39	56.12	-12.17
64 %	-119.55	-29.60	-126.03	13.26	41.52	51.33	-7.54
68 %	-78.49	-22.00	-87.47	12.72	34.65	47.45	-6.54
72 %	-56.74	-21.87	-60.57	11.90	29.56	42.56	-3.76
76 %	-41.05	-19.59	-51.06	10.52	25.41	37.53	-2.17
81 %	-30.71	-6.52	-42.15	9.95	21.24	31.40	-2.78
83 %	-28.97	-4.06	-40.32	9.91	20.52	28.95	-3.04
85 %	-27.26	-1.21	-36.84	9.89	19.54	26.27	-3.04
87 %	-26.13	0.63	-36.52	9.63	18.45	23.57	-3.19
89 %	-25.69	2.01	-35.71	9.39	17.53	20.89	-3.36
90 %	-25.34	2.31	-35.16	9.34	16.90	19.88	-3.36
95 %	-22.42	3.45	-25.40	8.47	14.49	13.73	-3.14

Tabla 3.8: Diferencia entre el *RUL* real y predicho. Resultados para **mínimos cuadrados con el modelo linealizado**. *Tests 26-38*

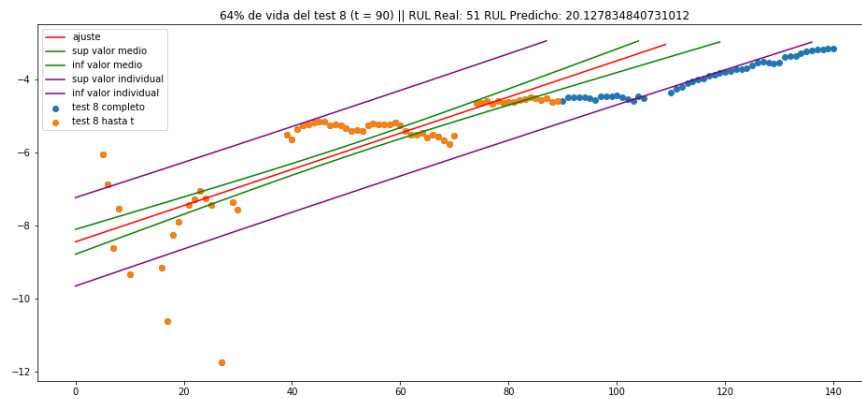


Figura 3.44: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 64 % (t=90) de la vida del dispositivo del *Test 8*

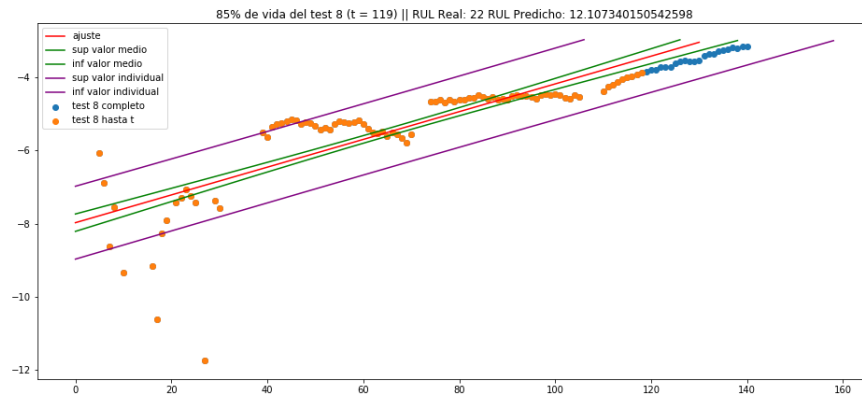


Figura 3.45: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 85 % (t=119) de la vida del dispositivo del *Test 8*

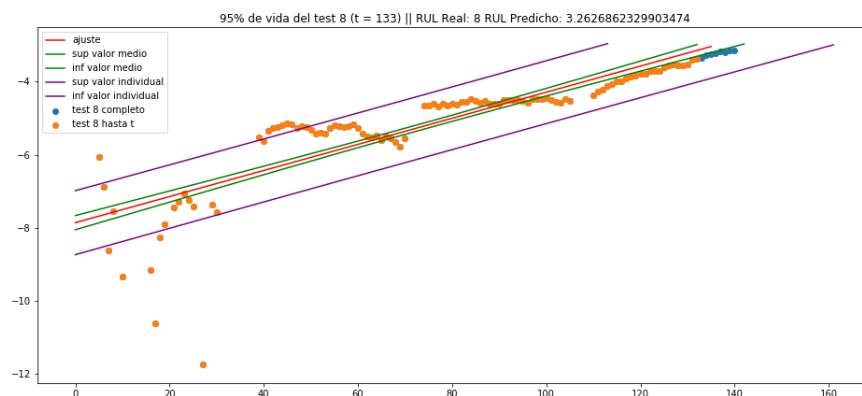


Figura 3.46: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 95 % (t=133) de la vida del dispositivo del *Test 8*

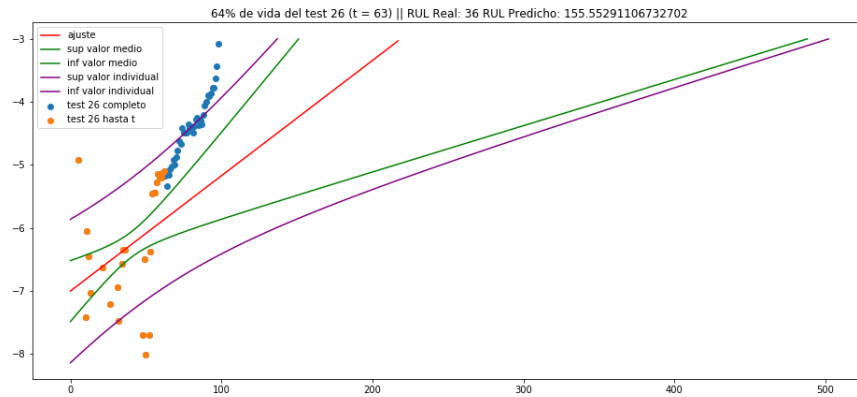


Figura 3.47: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 64 % (t=63) de la vida del dispositivo del *Test 26*

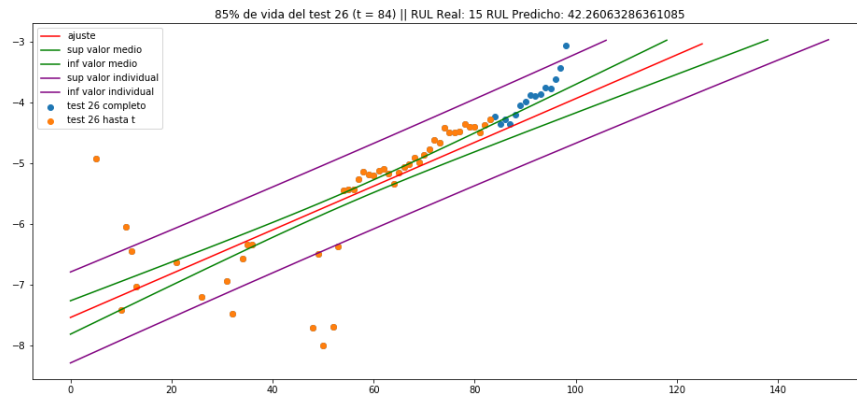


Figura 3.48: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 85 % (t=84) de la vida del dispositivo del *Test 26*

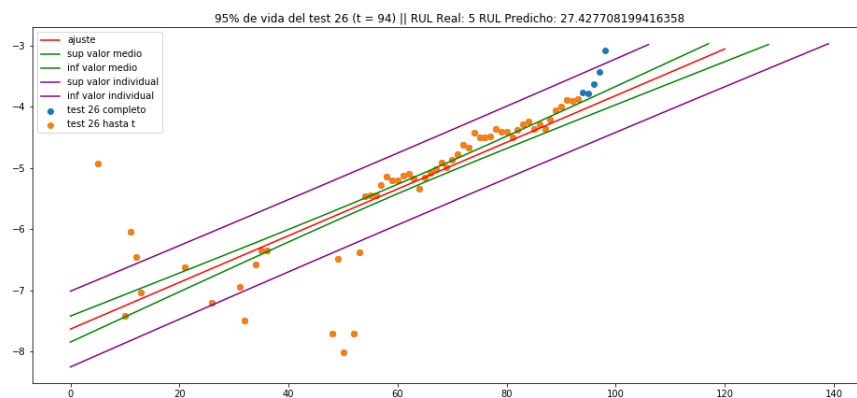


Figura 3.49: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 95 % (t=94) de la vida del dispositivo del *Test 26*

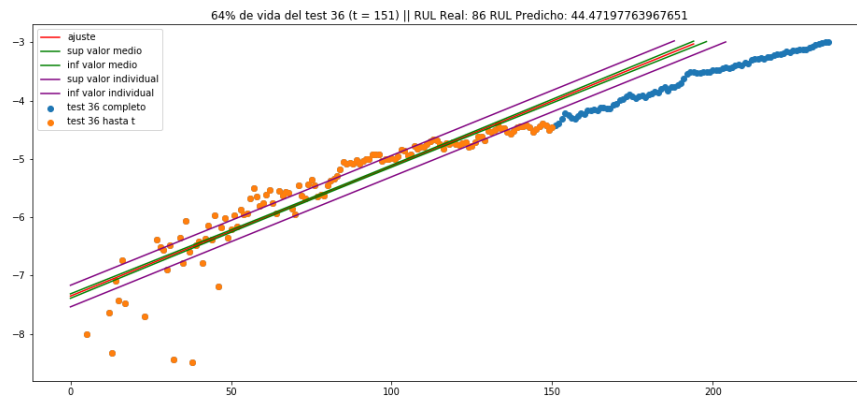


Figura 3.50: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 64 % (t=151) de la vida del dispositivo del *Test 36*

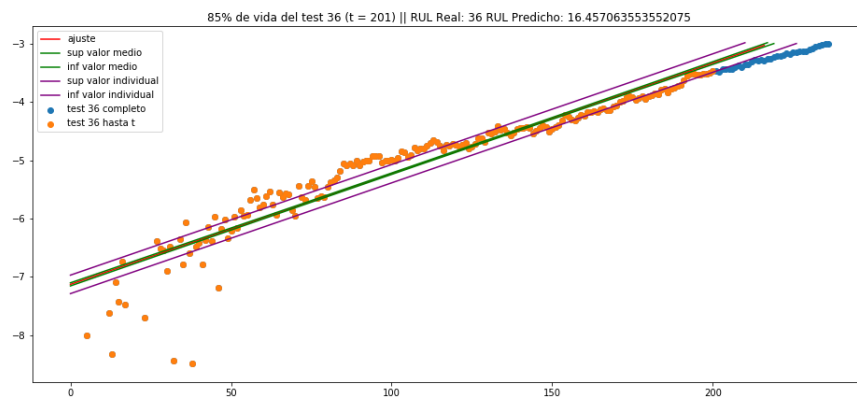


Figura 3.51: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 85 % (t=201) de la vida del dispositivo del *Test 36*

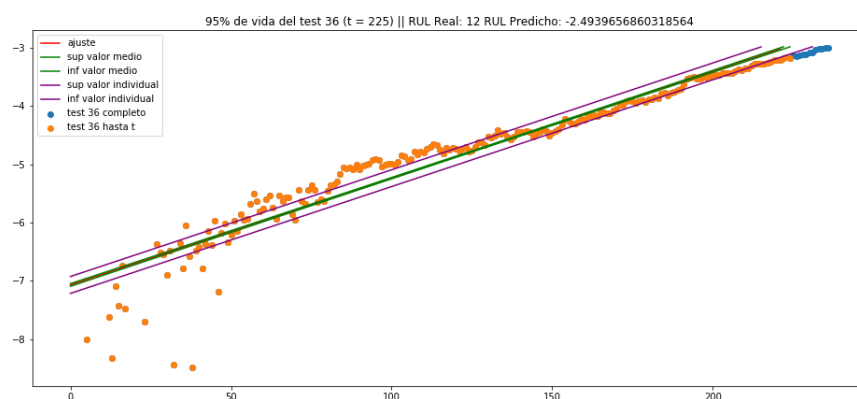


Figura 3.52: **Mínimos cuadrados con el modelo linealizado:** Predicción en el 95 % (t=225) de la vida del dispositivo del *Test 36*

min. vivos	Test 8 141 min.	Test 9 217 min.	Test 11 245 min.	Test 12 243 min.	Test 13 65 min.	Test 14 140 min.	Test 24 189 min.
55 %	4.67	47.74	55.49	40.39	3.03	4.40	22.38
59 %	0.47	37.04	42.68	30.44	-0.32	3.21	28.29
64 %	-7.82	31.70	22.33	45.16	-3.32	-2.76	23.00
68 %	-9.13	32.05	17.04	36.48	12.43	-9.46	12.10
72 %	-13.58	29.53	27.08	23.09	12.35	-17.35	5.82
76 %	-22.52	28.12	21.58	18.17	10.11	-17.88	10.99
81 %	-14.35	23.72	19.96	12.21	8.43	-5.27	3.15
83 %	-12.22	20.02	14.64	7.37	7.01	-8.36	3.50
85 %	-10.38	17.07	7.40	2.31	4.20	-7.20	4.55
87 %	-9.82	16.02	12.10	-4.07	3.11	-6.58	9.86
89 %	-10.09	12.00	7.40	-4.64	2.63	-6.09	5.57
90 %	-7.65	10.77	6.25	-3.27	2.56	-4.02	6.00
95 %	-5.66	6.07	-2.48	-3.47	0.66	-2.35	2.37

Tabla 3.9: Diferencia entre el *RUL* real y predicho. Resultados para la **media horizontal con el modelo linealizado**. Tests 8-24

min. vivos	Test 26 99 min.	Test 29 136 min.	Test 32 77 min.	Test 35 145 min.	Test 36 237 min.	Test 37 245 min.	Test 38 72 min.
55 %	-80.33	-17.17	-37.34	-10.55	48.40	53.41	-45,97
59 %	-43.39	-44.74	-39.65	-10.54	44.77	49.93	-41,87
64 %	-41.73	-19.23	-35.92	-7.11	34.91	48.03	-41,60
68 %	-44.92	-15.14	-27.89	-6.44	32.86	40.42	-42,21
72 %	-41.39	-22.38	-27.99	-6.48	29.41	34.52	-29,17
76 %	-28.56	-10.49	-29.32	-8.72	24.12	26.84	-27,56
81 %	-33.18	13.91	-26.75	0.67	21.14	20.69	-30,55
83 %	-38.43	11.11	-27.20	2.94	22.44	16.44	-31,77
85 %	-32.64	11.99	-25.77	3.98	19.45	11.86	-25,22
87 %	-37.60	8.45	-31.33	1.65	15.56	8.35	-26,36
89 %	-39.32	8.08	-31.08	2.75	13.21	6.15	-23,29
90 %	-34.88	6.06	-29.58	4.51	13.10	4.04	-23,29
95 %	-27.19	2.90	-8.25	1.70	5.75	-5.33	-13,15

Tabla 3.10: Diferencia entre el *RUL* real y predicho. Resultados para la **media horizontal con el modelo linealizado**. Tests 26-38

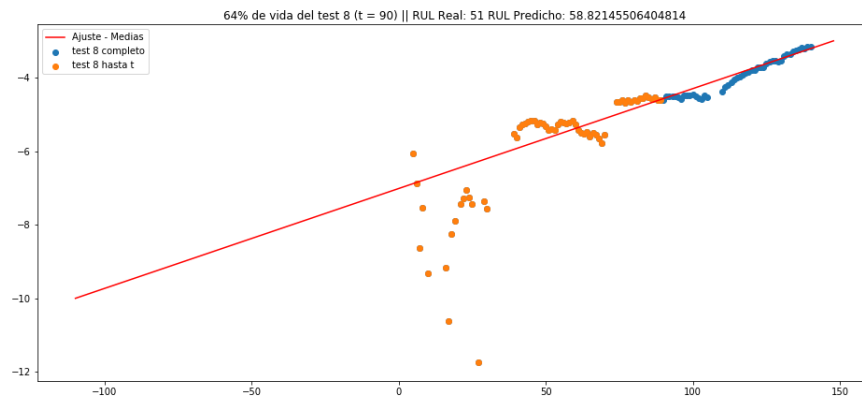


Figura 3.53: **Media horizontal con el modelo linealizado:** Predicción en el 64 % (t=90) de la vida del dispositivo del *Test 8*

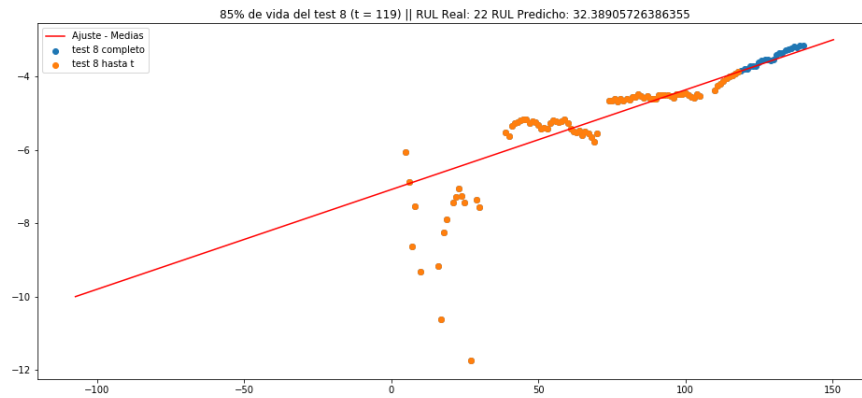


Figura 3.54: **Media horizontal con el modelo linealizado:** Predicción en el 85 % (t=119) de la vida del dispositivo del *Test 8*

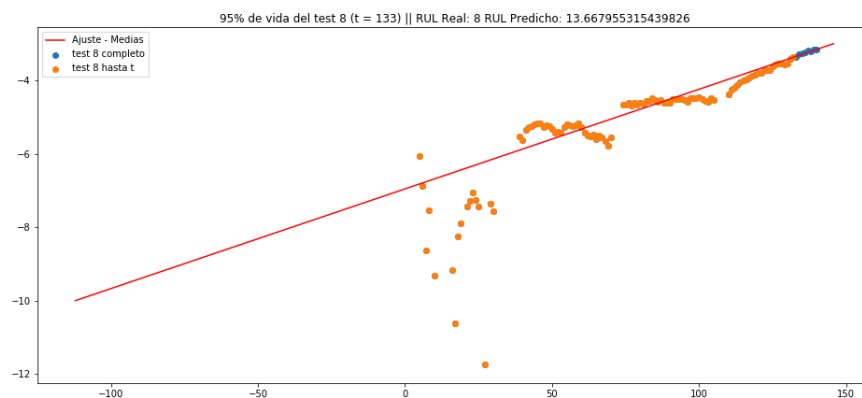


Figura 3.55: **Media horizontal con el modelo linealizado:** Predicción en el 95 % (t=133) de la vida del dispositivo del *Test 8*

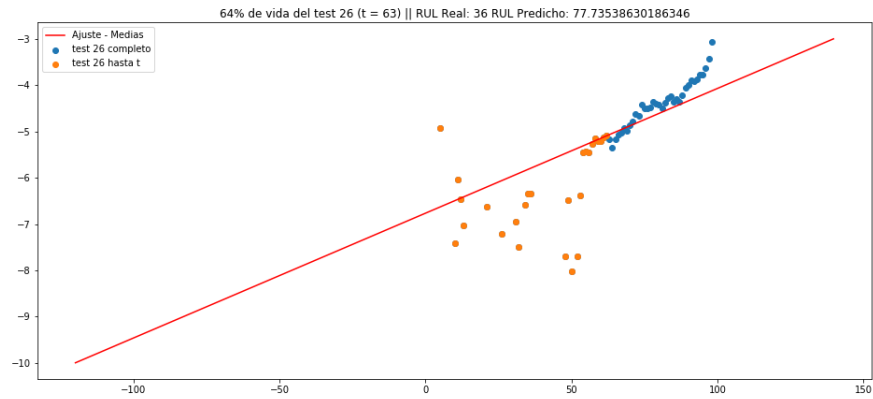


Figura 3.56: **Media horizontal con el modelo linealizado:** Predicción en el 64 % (t=63) de la vida del dispositivo del *Test 26*

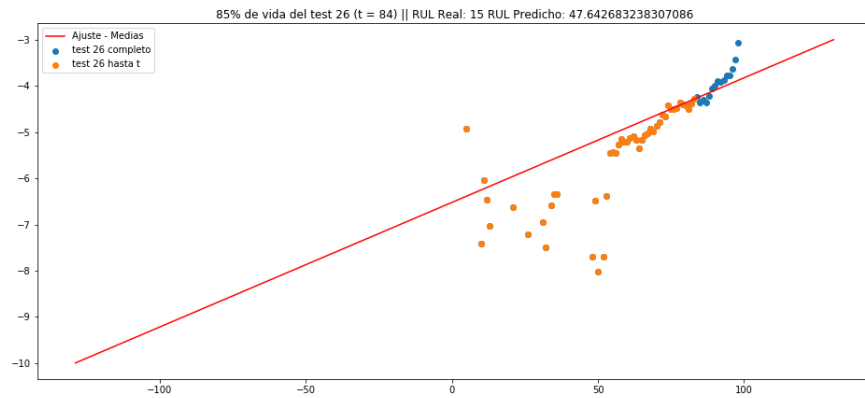


Figura 3.57: **Media horizontal con el modelo linealizado:** Predicción en el 85 % (t=84) de la vida del dispositivo del *Test 26*

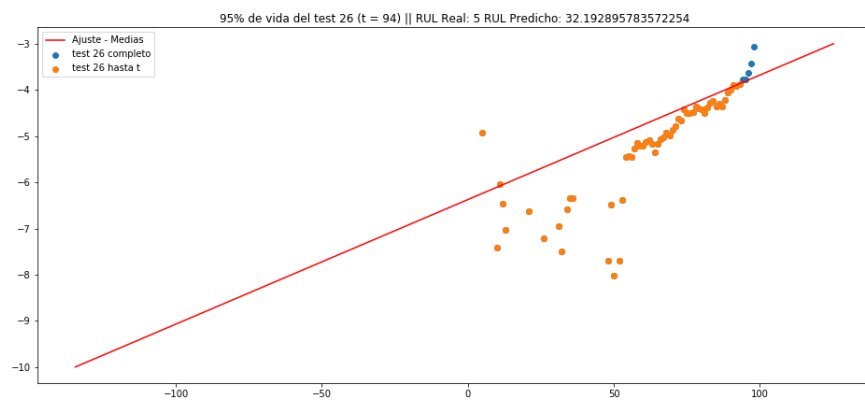


Figura 3.58: **Media horizontal con el modelo linealizado:** Predicción en el 95 % (t=94) de la vida del dispositivo del *Test 26*

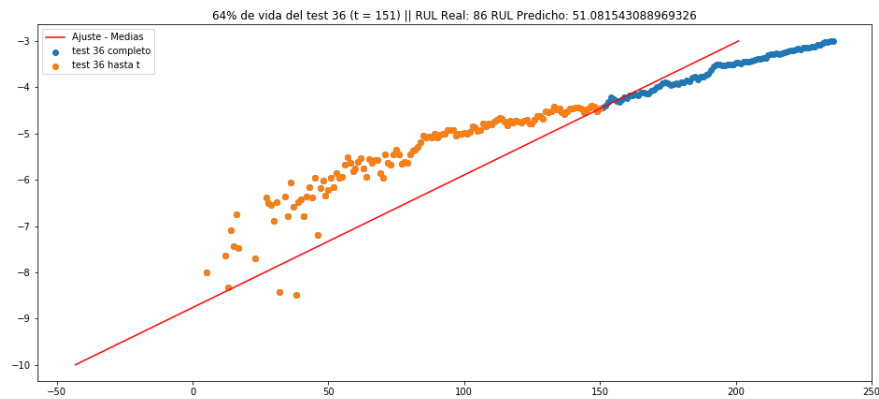


Figura 3.59: **Media horizontal con el modelo linealizado:** Predicción en el 64 % (t=151) de la vida del dispositivo del *Test 36*

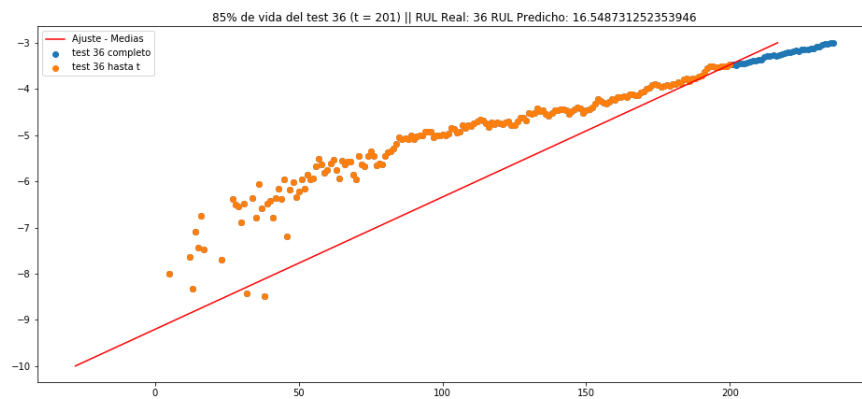


Figura 3.60: **Media horizontal con el modelo linealizado:** Predicción en el 85 % (t=201) de la vida del dispositivo del *Test 36*

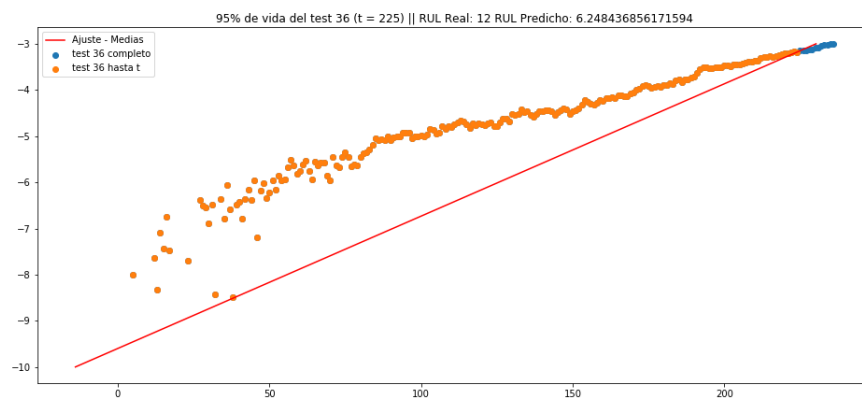


Figura 3.61: **Media horizontal con el modelo linealizado:** Predicción en el 95 % (t=225) de la vida del dispositivo del *Test 36*

3.7. Discusión de resultados

Dada la variabilidad de los datos observados en las *gráficas 3.14 y 3.15* para ilustrar la discusión se ha elegido el *test 36* para centrar la discusión, que es el que se discute en el trabajo de referencia [3], que puede simbolizar el tipo de test ideal, y se ha contrastado con otros dos tests que consideramos como alejados de ese comportamiento ideal: los *tests 8 y 26*. En la *tabla 3.11* muestra los resultados obtenidos para el *test 36* en el trabajo de referencia [3] aplicando diferentes técnicas (*Extended Kalman Filter, Gaussian Process Regressor y Particle Filter*).

	RUL	EKF RUL*	GPR RUL*	PF RUL*
t=140 61.40 %	88	64.98 (23.02)	N/A	77.65 (10.35)
t=150 67.79 %	78	80.22 (-2.22)	N/A	65.85 (12.15)
t=160 70.18 %	68	56.64 (11.36)	N/A	58.33 (9.67)
t=170 74.56 %	58	50.15 (7.85)	N/A	49.47 (8.53)
t=180 78.95 %	48	42.75 (5.25)	73.2 (-25.2)	38.68 (9.32)
t=190 83.33 %	38	30.35 (7.65)	33.4 (4.6)	27.14 (10.86)
t=195 85.53 %	33	18.57 (14.43)	17.6 (15.4)	24.76 (8.24)
t=200 87.72 %	28	17.24 (10.76)	14.6 (13.4)	21.09 (6.91)
t=205 89.91 %	23	18.28 (4.72)	13.8 (9.2)	16.66 (6.34)
t=210 92.11 %	18	13.46 (4.54)	11.8 (6.2)	14.68 (3.32)

Tabla 3.11: Resultados obtenidos en el trabajo de referencia [3]

Las *tablas 3.1 y 3.2* muestran los resultados del método más sencillo, la estimación de la media horizontal para los tests seleccionados. Los resultados de estimación del *RUL* para tres instantes de vida del dispositivo (en el 64 % de vida, en el 85 % y en el 95 %) se observan en las *gráficas desde 3.17 hasta 3.25*, para los *test 8, 26 y 36*, respectivamente. En todos los casos el método ajusta mejor según avanzan los experimentos y se acerca al final de vida útil. Incluso en el *test 8* los resultados son buenos. En el caso del *test 26* que termina de forma muy rápida el ajuste tiende a retrasar mucho el tiempo de estimación.

Los resultados de este método son muy buenos en general, obteniendo resultados equivalentes o mejores a los del *EKF* de la referencia [3] (*tabla 3.11*) y a todos los métodos que hemos utilizado, con la excepción de la media horizontal sobre el modelo linealizado. La *tabla 3.12* muestra cómo los métodos basados en la media horizontal obtienen el

menor error medio en la estimación de la vida útil cerca de su final (95 %). Una posible explicación de este hecho es que la ley de envejecimiento propuesta en [3] no describe correctamente todos los test. El método de la media horizontal, aunque también se basa en esta ley, promedia el comportamiento de los restantes tests, proporciona una función que no representa a ningún dispositivo concreto pero que, en promedio, describe mejor la evolución futura del test actual.

Las *tablas 3.3 y 3.4* muestran los resultados del método de regresión mediante mínimos cuadrados. Los resultados de estimación del *RUL* para tres instantes de vida del dispositivo (en el 64 % de vida, en el 85 % y en el 95 %) se observan en las gráficas desde 3.26 hasta 3.34, para los *test 8, 26 y 36*, respectivamente. El método obtiene resultados que ajustan bastante bien el final de vida del dispositivo, siempre acercándose a la predicción a medida que se acerca el fin de su vida. Se observa el excelente ajuste que hace para el 36 y un comportamiento aceptable para el 8. En el caso del 26, como ocurrió con el anterior, tiene problemas para ajustar en las fases iniciales, haciendo predicciones alejadas.

Las *tablas 3.5 y 3.6* muestran los resultados del filtro extendido de Kalman. Los resultados de estimación del *RUL* para tres instantes de vida del dispositivo (en el 64 % de vida, en el 85 % y en el 95 %) se observan en las gráficas desde 3.35 hasta 3.43, para los *test 8, 26 y 36*, respectivamente. En estas gráficas se ha añadido también la predicción hasta el instante de vida *actual* del *EKF*. Se observa que la estimación está algo más alejada que en los dos métodos anteriores y la predicción está muy alejada cuando la vida del dispositivo es muy corta y se aleja del modelo utilizado. En este caso los resultados de las tablas nos indican que para comportamientos similares a lo esperado, el ajuste es razonablemente bueno (*tests 8, 12, 14, 24, 29, 35 y 36*), pero para el resto hace estimaciones peores que los métodos anteriores. Su comportamiento promedio, al final de la vida útil es el peor, como indica la *tabla 3.12*.

Las *tablas 3.7 y 3.8* muestran los resultados del método de regresión por mínimos cuadrados en la versión linealizada del modelo. Los resultados de estimación del *RUL* para tres instantes de vida del dispositivo (en el 64 % de vida, en el 85 % y en el 95 %) se observan en las gráficas desde 3.44 hasta 3.52, para los *test 8, 26 y 36*, respectivamente. En este caso además se añaden las bandas inferior y superior de las predicciones tanto puntuales como con el valor medio. Se puede observar que el comportamiento del método no mejora los resultados anteriores de forma general. Puntualmente mejora pero también empeora resultados previos.

Las *tablas 3.9 y 3.10* muestran los resultados del método de regresión por mínimos cuadrados en la versión linealizada del modelo. Los resultados de estimación del *RUL* para tres instantes de vida del dispositivo (en el 64 % de vida, en el 85 % y en el 95 %) se observan en las gráficas desde 3.53 hasta 3.61, para los *test 8, 26 y 36*, respectivamente. En este caso obtenemos resultados similares a la técnica anterior, sin conseguir una mejora generalizada.

Las conclusiones que se pueden obtener en términos generales es que los dos métodos más sencillos son los que producen estimaciones genéricas más precisas, realizando siempre el análisis post-mortem. Los modelos linealizados funcionan también mejor de lo esperado porque el ajuste realizado es mejor que en el caso no lineal. Finalmente indicar que sería necesario mejorar el tipo de filtro empleado (en lugar de utilizar *EKF*, utilizar un filtro de partículas o un observador de estados y parámetros) para conseguir un mejor

	Error absoluto medio
Media horizontal	7.78
Ajuste por mínimos cuadrados	10.16
Filtro de Kalman extendido	36.00
Ajuste por mínimos cuadrados (modelo linealizado)	10.78
Media horizontal (modelo linealizado)	6.27

Tabla 3.12: Error absoluto medio de cada método al 95 % de la vida del dispositivo sobre todo el conjunto de tests

ajuste. De todas formas, las dificultades que tiene *EKF* para ajustar la serie hace sospechar que el modelo elegido de envejecimiento no funciona en toda la gama de dispositivos.

En las *tablas 3.13, 3.14 y 3.15* que se presentan a continuación, muestran el *RUL* estimado por cada uno de los métodos en distintos instantes de tiempo para los *test 8, 26 y 36*, respectivamente. Para facilitar la comparación se incluye una columna con el *RUL* verdadero y, entre paréntesis, la diferencia entre el *RUL* estimado y el *RUL* verdadero. Las *figuras 3.62, 3.63 y 3.64* muestran gráficamente la información de las tablas. La línea gris representa el *RUL* verdadero y las dos líneas negras corresponden a un error de $\pm 30\%$.

	RUL	MH RUL*	MMCC RUL*	EKF RUL*	MMCC_LN RUL*	MH_LN RUL*
t=77 55 %	64	69.77 (-5.77)	72.39 (-8.39)	88.50 (-24.50)	24.70 (39.30)	59.33 (4.67)
t=83 59 %	58	67.77 (-9.77)	53.51 (4.49)	77.38 (-19.38)	22.18 (35.82)	57.53 (0.47)
t=90 64 %	51	69.21 (-18.21)	54.67 (-3.67)	69.68 (-18.68)	20.13 (30.87)	58.82 (-7.82)
t=95 68 %	46	65.10 (-19.10)	59.46 (-13.46)	73.89 (-27.89)	19.13 (26.87)	55.14 (-9.14)
t=101 72 %	40	63.36 (-23.36)	69.03 (-29.03)	71.94 (-31.94)	18.40 (21.60)	53.58 (-13.58)
t=107 76 %	34	66.66 (-32.66)	82.98 (-48.98)	71.47 (-37.47)	17.39 (16.61)	56.53 (-22.53)
t=114 81 %	27	49.38 (-22.38)	76.90 (-49.90)	71.75 (-44.75)	14.11 (12.89)	41.35 (-14.35)
t=117 83 %	24	43.41 (-19.41)	64.11 (-40.11)	54.63 (-30.63)	13.05 (10.95)	36.22 (-12.22)
t=119 85 %	22	38.91 (-16.91)	55.38 (-33.38)	44.42 (-22.42)	12.11 (9.89)	32.39 (-10.39)
t=122 87 %	19	34.70 (-15.70)	44.47 (-25.47)	34.75 (-15.75)	10.50 (8.50)	28.82 (-9.82)
t=125 89 %	16	31.46 (-15.46)	36.16 (-20.16)	23.62 (-7.62)	8.76 (7.24)	26.09 (-10.09)
t=126 90 %	15	27.37 (-12.37)	33.22 (-18.22)	20.27 (-5.27)	8.10 (6.90)	22.66 (-7.66)
t=133 95 %	8	16.58 (-8.58)	18.38 (-10.38)	0.85 (7.15)	3.26 (4.74)	13.67 (-5.67)

Tabla 3.13: Resultados para el *Tests 8*

	RUL	MH RUL*	MMCC RUL*	EKF RUL*	MMCC_LN RUL*	MH_LN RUL*
t=54 55 %	45	133.72 (-88.72)	48963.05 (-48918.05)	109.35 (-64.35)	3.32E+20 (-3.32E+20)	125.34 (-80.34)
t=58 59 %	41	98.12 (-57.12)	5.69 (35.31)	105.82 (-64.82)	965.43 (-924.43)	84.39 (-43.39)
t=63 64 %	36	91.27 (-55.27)	10.55 (25.45)	97.68 (-61.68)	155.55 (-119.55)	77.74 (-41.74)
t=67 68 %	32	90.43 (-58.43)	15.88 (16.12)	93.46 (-61.46)	110.49 (-78.49)	76.93 (-44.93)
t=71 72 %	28	82.37 (-54.37)	18.21 (9.79)	91.08 (-63.08)	84.75 (-56.75)	69.40 (-41.40)
t=75 76 %	24	63.52 (-39.52)	16.52 (7.48)	86.58 (-62.58)	65.05 (-41.05)	52.57 (-28.57)
t=80 81 %	19	63.08 (-44.08)	16.88 (2.12)	71.02 (-52.02)	49.71 (-30.71)	52.19 (-33.19)
t=82 83 %	17	66.81 (-49.81)	18.37 (-1.37)	74.09 (-57.09)	45.98 (-28.98)	55.44 (-38.44)
t=84 85 %	15	57.81 (-42.81)	18.74 (-3.74)	68.52 (-53.52)	42.26 (-27.26)	47.64 (-32.64)
t=86 87 %	13	61.25 (-48.25)	19.44 (-6.44)	71.50 (-58.50)	39.14 (-26.14)	50.60 (-37.60)
t=88 89 %	11	60.93 (-49.93)	20.75 (-9.75)	70.46 (-59.46)	36.69 (-25.69)	50.32 (-39.32)
t=89 90 %	10	54.59 (-44.59)	20.82 (-10.82)	65.82 (-55.82)	35.34 (-25.34)	44.89 (-34.89)
t=94 95 %	5	39.51 (-34.51)	16.30 (-11.30)	64.21 (-59.21)	27.43 (-22.43)	32.19 (-27.19)

Tabla 3.14: Resultados para el Tests 26

	RUL	MH RUL*	MMCC RUL*	EKF RUL*	MMCC_LN RUL*	MH_LN RUL*
t=130 55 %	107	68.80 (38.19)	124.85 (-17.85)	54.51 (52.48)	49.26 (57.73)	58.59 (48.40)
t=139 59 %	98	62.82 (35.17)	119.08 (-21.08)	35.90 (62.09)	46.60 (51.39)	53.22 (44.77)
t=151 64 %	86	60.41 (25.58)	129.28 (-43.28)	51.36 (34.63)	44.47 (41.52)	51.08 (34.91)
t=161 68 %	76	51.34 (24.65)	120.33 (-44.33)	37.35 (38.64)	41.34 (34.65)	43.13 (32.86)
t=170 72 %	67	44.93 (22.06)	104.95 (-37.95)	42.35 (24.64)	37.43 (29.56)	37.58 (29.41)
t=180 76 %	57	39.41 (17.58)	81.04 (-24.04)	16.01 (40.98)	31.58 (25.41)	32.87 (24.12)
t=191 81 %	46	29.94 (16.05)	62.24 (-16.24)	22.72 (23.27)	24.75 (21.24)	24.85 (21.14)
t=196 83 %	41	22.43 (18.56)	47.16 (-6.16)	25.14 (15.85)	20.47 (20.52)	18.55 (22.44)
t=201 85 %	36	20.02 (15.97)	37.85 (-1.85)	-3.29 (39.29)	16.45 (19.54)	16.54 (19.45)
t=206 87 %	31	18.68 (12.31)	31 (0)	4.80 (26.19)	12.54 (18.45)	15.43 (15.56)
t=210 89 %	27	16.69 (10.30)	26.42 (0.57)	15.73 (11.26)	9.46 (17.53)	13.78 (13.21)
t=213 90 %	24	13.21 (10.78)	22.93 (1.06)	20.17 (3.82)	7.09 (16.90)	10.89 (13.10)
t=225 95 %	12	7.59 (4.40)	10.07 (1.92)	17.08 (-5.08)	-2.49 (14.49)	6.24 (5.75)

Tabla 3.15: Resultados para el *Tests 36*

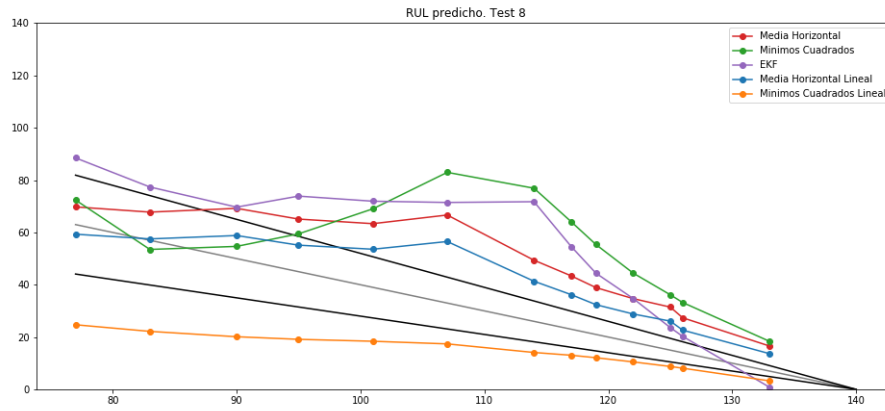


Figura 3.62: Predicción del *RUL* a lo largo del tiempo para el *Test 8*

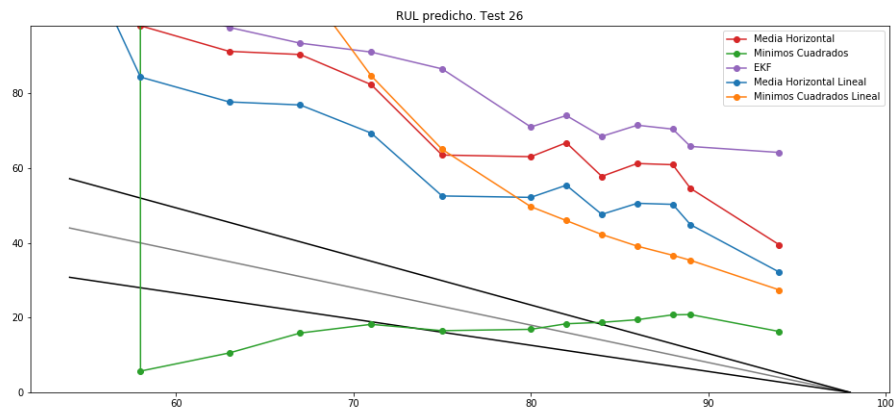


Figura 3.63: Predicción del *RUL* a lo largo del tiempo para el *Test 26*

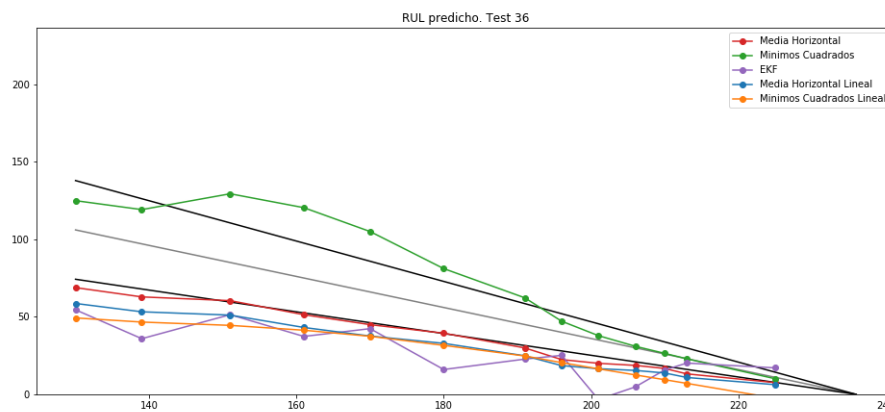


Figura 3.64: Predicción del *RUL* a lo largo del tiempo para el *Test 36*

Capítulo 4

Estudio de una posible plataforma Big Data

En este capítulo se va a plantear el diseño de una plataforma Big Data para la empresa *BnL*, fabricante de dispositivos electrónicos MOSFET.

La arquitectura que se va a diseñar está planteada para la realización de tests de calidad sobre los distintos dispositivos fabricados. Un ejemplo de análisis de los datos generados en este tipo de empresa es el realizado con el conjunto de datos de los MOSFET donde se estima la vida útil restante de cada dispositivo. El testeo de estos dispositivos, a altas frecuencias, genera una gran cantidad de datos, normalmente inmanejables con los sistemas de procesamiento tradicional. En este marco, para facilitar el almacenamiento, gestión y procesamiento de los datos generados en los experimentos, es donde se quiere diseñar una arquitectura Big Data. Para hacerse una idea de la cantidad de datos que pueden llegar a manejar, para el caso concreto del conjunto de datos analizados en el capítulo anterior, se han generado en 50 GB de datos en formato CSV para 42 tests realizados sobre transistores. Si se supone que este número de tests son los que realiza la empresa en un día, en un mes habrá generado 1.5TB de datos, solo para un tipo de dispositivo concreto.

La empresa *BnL* es una empresa ficticia fabricante de semiconductores. Esta empresa realiza pruebas de calidad para todos los productos que fabrica. Actualmente esta empresa se ha centrado en los transistores MOSFET, es uno de los productos mas importantes dentro de su catálogo de productos. En las pruebas a las que se someten los transistores se realizan mediciones a altas frecuencias, lo que implica una gran cantidad de datos en un período de tiempo relativamente pequeña. Estas pruebas se realizan de forma diaria sobre multitud de componentes. La metodología actual en la realización de los experimentos es la siguiente:

- La empresa dispone de varias máquinas o PCs en las que se tiene instalado un software (*LabVIEW*) con el que recoge los datos y controla los aparatos de medición, un sistema de adquisición de datos fabricado por *National Instruments* y un osciloscopio fabricado por *Agilent Technologies*.
- Los datos generados se almacenan dentro de un directorio en red compartido para toda la organización. En ese directorio en red se encuentran todos los experimentos realizados en la empresa. El equipo de experimentación los vuelca manualmente ahí de forma diaria.

- El equipo que quiera analizar los experimentos tiene que ir al directorio de experimentos, coger los que le interesen, procesarlos, con todo el esfuerzo computacional que conlleva y analizarlos.
- La empresa tiene una herramienta de visualización de *Business Intelligence* para la visualización de los datos, y de los análisis y de los informes que realizan los equipos de análisis. Todos estos datos se cargan manualmente en la herramienta de visualización.

La empresa *BnL* se ha dado cuenta de que la metodología con la que trabaja, aunque en el inicio de los tests de los componentes pudo funcionar al ser pocos experimentos, actualmente no es eficiente. Por tanto, plantea dos necesidades: trabajar con la gran cantidad de datos generados a causa de la experimentación y automatizar las tareas de almacenamiento, transformación y carga de los datos en la herramienta de visualización. Estas dos necesidades ha llevado a la empresa *BnL* a plantearse implantar una plataforma Big Data que las de respuesta.

4.1. Especificación de los requisitos del sistema

En esta sección se van a especificar las necesidades de la empresa a las que la plataforma tiene que hacer frente. Se especifican los usuarios y requisitos (funcionales y no funcionales) de la misma adaptados a las necesidades de la empresa.

4.1.1. Roles en el sistema

- **Administrador:** Encargado de administrar permisos y el mantenimiento del sistema.
- **Ingeniero de datos:** Encargado de trabajar en las tareas de transformación de los datos. Programa y crea los flujos de datos y sus transformaciones, desde que los datos están en bruto hasta que los datos están refinados.
- **Científico de datos:** Encargado de realizar análisis de los datos y creación de los modelos predictivos.

4.1.2. Objetivos

- Almacenar los datos generados de los experimentos, manteniendo un único histórico de los mismos.
- Acceder a los datos generados en los experimentos.
- Transformar la gran cantidad de datos generados durante la experimentación para que sean abordables por herramientas analíticas no Big Data.
- Realizar distintos tipos de análisis sobre los datos almacenados. Un ejemplo de análisis puede ser la estimación de la vida útil de un dispositivo. Los análisis pueden realizarse en tiempo real o sobre un experimento previamente almacenado.

- Exportar los datos a herramientas de visualización tanto en tiempo real como atemporal.

4.1.3. Requisitos funcionales

Referencia	RF-1
Nombre	Recoger los datos creados por los sistemas de adquisición de datos (<i>LabVIEW</i>)
Descripción	El sistema debe ser capaz de recoger los datos creados por los sistemas de adquisición de datos conectados a máquinas que transmitirán los datos a la plataforma a través de <i>LabVIEW</i> .

Referencia	RF-2
Nombre	Histórico de experimentos
Descripción	El sistema debe ser capaz de almacenar los datos de todos los experimentos realizados hasta la fecha.

Referencia	RF-3
Nombre	Datos accesibles
Descripción	El sistema debe permitir consultar los datos que tenga almacenados.

Referencia	RF-4
Nombre	Procesamiento de experimentos
Descripción	El sistema debe poder realizar muestreos y transformaciones sobre los datos para que puedan ser tratados de forma analítica.

Referencia	RF-5
Nombre	Modelos predictivos
Descripción	El sistema debe permitir crear y almacenar modelos predictivos para poder realizar clasificaciones y predicciones de los datos

Referencia	RF-6
Nombre	Análisis predictivo
Descripción	El sistema debe realizar predicciones en tiempo real para calcular el tiempo útil de vida (<i>RUL</i> o <i>Remaining Useful Life</i>) de dispositivos

Referencia	RF-7
Nombre	Integración con herramientas de <i>Business Intelligence</i>
Descripción	El sistema debe procesar, transformar y servir los datos para que sean integrables con herramientas de <i>Business Intelligence</i>

4.1.4. Requisitos no funcionales

Referencia	RNF-1
Nombre	Herramientas de recolección de datos integrables
Descripción	El sistema debe permitir integrar herramientas de recolección de datos como <i>LabVIEW</i> .

Referencia	RNF-2
Nombre	Disponibilidad de los datos
Descripción	El sistema debe tener disponibles en todo momento los datos almacenados.

Referencia	RNF-3
Nombre	Procesamiento en tiempo real
Descripción	El sistema debe permitir recoger, procesar y servir flujos de datos en tiempo real.

Referencia	RNF-4
Nombre	Seguridad y tolerancia a fallos
Descripción	El sistema debe ser seguro y tolerante a fallos.

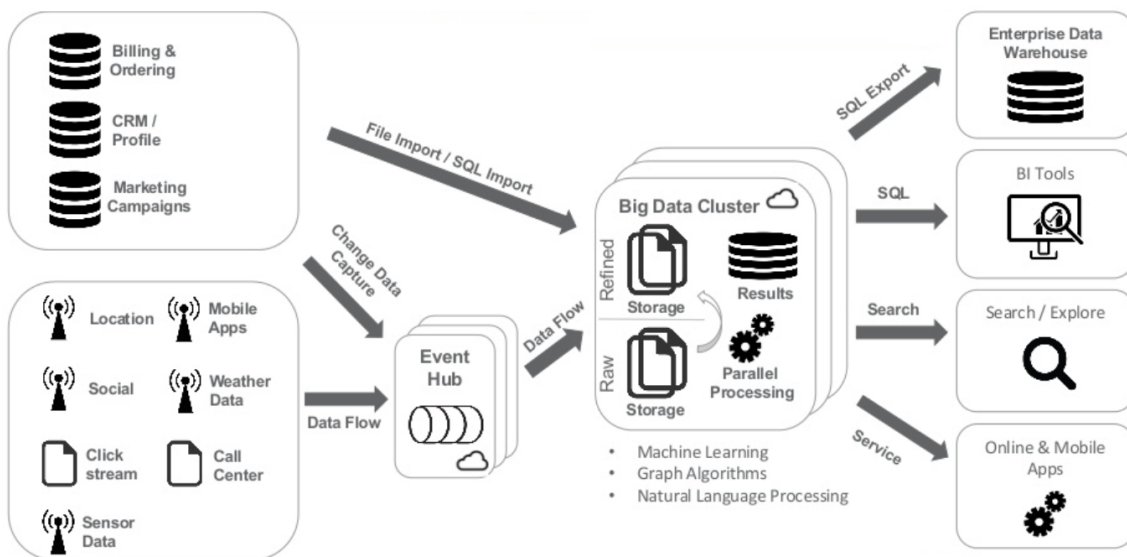
Referencia	RNF-5
Nombre	Escalabilidad
Descripción	El sistema debe ser escalable horizontalmente.

4.2. Arquitecturas Big Data

En esta sección se va a realizar una exposición de las arquitecturas Big Data existentes para ver cuál de todas ellas se adapta mejor al problema que se pretende resolver.

La estructura y flujo tradicional de una plataforma Big Data se puede ver en la *Figura 4.1*. Se tienen unas fuentes de datos que se importan en la plataforma Big Data, se almacenan y procesan, y finalmente son consultadas por diferentes consumidores de datos. El problema existente es, que para los consumidores de datos, no es abordable recoger y transformar todos los datos de las fuentes. Situar la plataforma entre las fuentes de los datos y los consumidores de datos ayuda a recolectar los datos, mezclarlos y refinarlos, para que los consumidores de datos sólo tengan que consultarlos.

En una arquitectura Big Data, no sólo hay que tener en cuenta las fuentes y los consumidores de datos, también hay que tener en cuenta cómo son los datos dentro del flujo de la plataforma. Hay dos tipos de datos: *data at rest* y *data in motion*. *Data at rest* o datos en reposo, hace referencia información recopilada de varias fuentes, y que se encuentra

Figura 4.1: Arquitectura *general* [9]

después de que los eventos de creación de esos datos ocurrieran. *Data in motion* o datos en movimiento, hace referencia a información que se está recogiendo en el momento que los eventos de creación de los datos ocurren. Otro factor que hay que tener en cuenta es la frecuencia con la que los consumidores acceden a los datos.

En este sentido se puede definir tres tipos de datos: *cold data*, *warm data* y *hot data*. *Cold data* o datos fríos son datos estructurados o no estructurados a los que no se suele acceder o solicitar frecuentemente. *Warm data* o datos cálidos (no calientes) son datos estructurados a los que se accede con una frecuencia moderada. *Hot data* o datos calientes, son datos estructurados a los que se accede o se solicitan frecuentemente.

En resumen, cómo sean los datos y la frecuencia de acceso a los mismos son factores que afectan a la hora de elegir la arquitectura de la plataforma Big Data.

4.2.1. Arquitectura tradicional

Este tipo de arquitectura Big Data es la primera aproximación a una solución que se planteó a la hora de diseñar una plataforma Big Data. Está enfocada al almacenamiento y procesamiento de datos estáticos. Sobre esta arquitectura tradicional, se construyeron o han evolucionado, el resto de arquitecturas que se van a comentar más adelante. Este tipo de arquitectura tradicional lleva a la construcción de un *data lake*.

Un *data lake* es un repositorio donde se almacenan datos en bruto (estructurados y no estructurados) a cualquier escala. No se les dota de estructura alguna hasta que sea necesaria su explotación. La idea principal de un *data lake* es tener un único almacén para todos los datos que cualquiera dentro de la organización o empresa pueda necesitar. En este contexto de este trabajo, se va a hablar de *data lake* asociado al almacenamiento de objetos orientado a Hadoop, aunque el concepto sea mucho más amplio.

Aunque los datos se almacenan en bruto, no es habitual su uso como tal. La mayoría de los *data lakes* tienden a usar métodos de extracción, carga y transformación (ETL) para

coleccionar e integrar los datos. Normalmente estos datos son procesados y refinados, dotándolos de estructura, combinándolos, enriqueciéndolos, etc. Este proceso de refinado sólo se puede dar debido al principal beneficio del *data lake*, que es la centralización de las fuentes de datos de contenidos dispares. Si se hubiese construido de otro modo, todo ese proceso hubiese sido imposible. En la arquitectura tradicional, la mayoría de las operaciones que se realizan para todo el proceso *ETL* que se ha comentado antes, y para el análisis y modelado de los datos, se realizan sobre disco con trabajos *MapReduce*. Es decir, se hace uso de herramientas que trabajan sobre *HDFS* como *Hive* o *Pig*, que realmente lo que hacen es traducir su lenguaje de consultas (*Hive*) o Scripting (*Pig*) a trabajos *MapReduce*. *HDFS* o *Hadoop Distributed File System* es un sistema de archivos distribuido, escalable y portátil escrito en Java para el framework Hadoop, ha sido creado especialmente para trabajar con ficheros de gran tamaño.

En este tipo de procesos se trabaja con bloques de datos que se han ido recogiendo a lo largo del tiempo, lo que habitualmente se llama procesamiento *batch*. El procesamiento *batch*, o procesamiento por lotes, es útil cuando la prioridad es trabajar sobre grandes volúmenes de información más que obtener rápidos análisis de resultados. El flujo de trabajo habitual de esta arquitectura se puede ver en la *Figura 4.2*

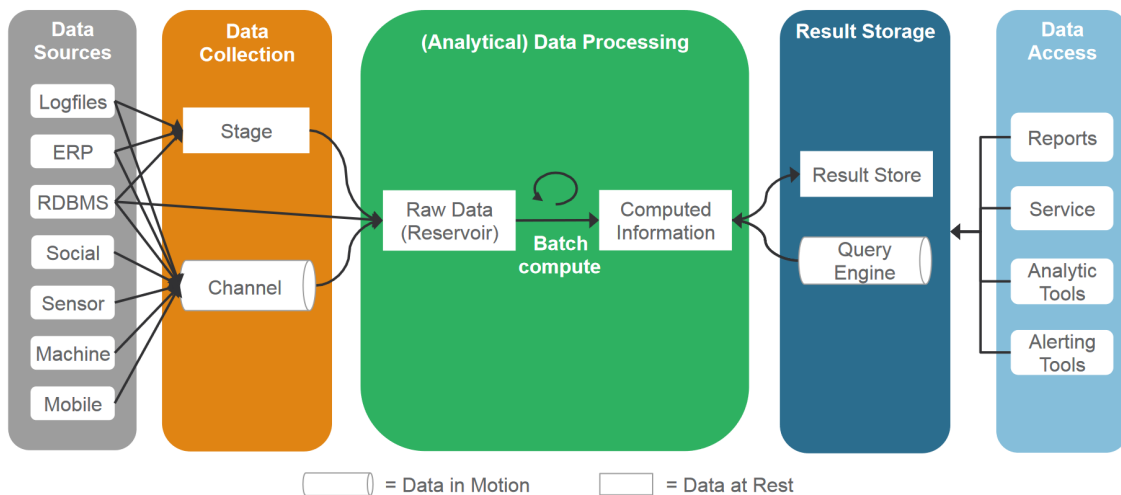


Figura 4.2: Arquitectura *Tradicional* [8]

Los trabajos *MapReduce* funcionan bien para grandes cantidades de datos, pero se pueden volver algo lentos, sobre todo porque escriben en disco en las diferentes fases del trabajo. Para solventar este problema, y como evolución natural, aparecen tecnologías que resuelven el problema de la lentitud de los trabajos en disco trabajando en memoria (*Spark*).

Las herramientas que se suelen utilizar en en la arquitectura tradicional, agrupadas según la fase del flujo de datos dentro de la arquitectura son las siguientes:

- Ingestión: Para la ingestión de *data at rest* alojados en base de datos se suele emplear *Sqoop* y para la ingestión de *data in motion* se suele usar *Flume* o *Kafka*. Ambos tipos de datos se guardan en *HDFS*.

- **Procesamiento:** El procesamiento en disco se realiza mediante operaciones *MapReduce*. Para facilitar en procesamiento con operaciones *MapReduce* se suele emplear *Hive* o *Pig*. El procesamiento en memoria se realiza con *Spark*.
- **Almacén de resultados:** El resultado de los procesamientos se pueden almacenar y posteriormente consultar a través de una base de datos como *HBase* o directamente consultar con un motor de consultas como *Hive*, *Cloudera Impala* o *Spark SQL*. También se puede exportar el resultado del procesamiento volcando los datos a una base de datos externa con *Sqoop*.

4.2.2. Arquitectura CEP (Complex Event Processing)

La arquitectura *CEP* o arquitectura de analítica *streaming*, es completamente opuesta a la arquitectura tradicional. En esta arquitectura no es necesario crear un *data lake* de datos estáticos, todo lo contrario, se encarga de manejar los datos en movimiento que llegan a la plataforma. Se puede traducir como una técnica de rastreo, análisis y procesamiento de datos en el momento que ocurren. Normalmente está orientado a la recopilación de información de sensores, relacionado por ejemplo, con el internet de las cosas. Es decir, es capaz de analizar datos que ocurren con alta frecuencia y baja latencia. Con este tipo de arquitectura se puede analizar y crear alertas de comportamiento en tiempo real. El flujo de la información se puede ver en la *Figura 4.3*. La idea detrás de este tipo de arquitectura es la de establecer correlaciones entre flujos de información y encontrar un patrón común que permita reconocer comportamientos. Está basado en el procesamiento de eventos simples, recopilando y combinando datos de diferentes fuentes para descubrir eventos y patrones que puedan dar como resultado acciones. Este tipo de arquitectura intenta dar respuesta a varios desafíos:

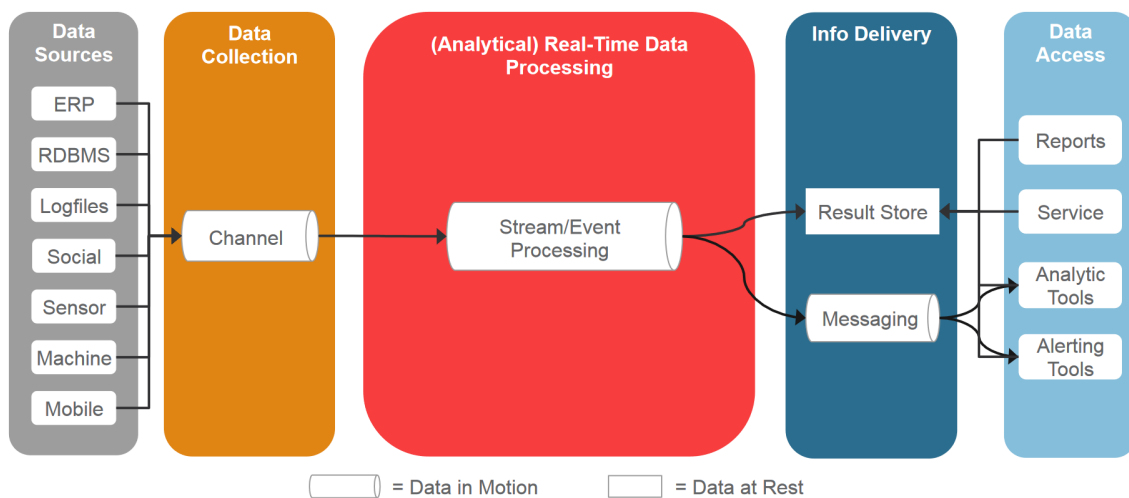


Figura 4.3: Arquitectura CEP [8]

- Posibilidad de producir resultados tan pronto como la secuencia de eventos de entrada esté disponible.

- Posibilidad de proporcionar cálculos como la agregación en el tiempo o tiempo de espera entre eventos de interés.
- Posibilidad de proporcionar alertas y notificaciones en tiempo real o casi en tiempo real sobre la detección de patrones de eventos complejos.
- Capacidad para conectar y correlacionar fuentes heterogéneas y analizar patrones con ellas.
- Capacidad de lograr alto rendimiento con un procesamiento de baja latencia.

Las herramientas que se suelen utilizar en la arquitectura *CEP*, agrupadas según la fase del flujo de datos dentro de la arquitectura son las siguientes:

- Ingestión: Como únicamente es *data in motion*, se suele usar *Kafka*.
- Procesamiento: Para el procesamiento de datos y eventos en *streaming* se suele usar *Storm*, *Flink*, *Spark Streaming* o *Samza*.
- Almacén de resultados: Para gestionar las consultas en tiempo real se suele usar una base de datos como *Cassandra*. También se puede exportar el resultado del procesamiento en forma de mensajes con *Kafka*.

4.2.3. Arquitectura lambda

Como se ha visto hasta ahora, existen dos tipos de arquitecturas completamente diferentes u opuestas. Por un lado, está la arquitectura tradicional, que es la que más tiempo de existencia tiene, enfocada al procesamiento por lotes, y por otro, la arquitectura *CEP*, enfocada en el procesamiento (casi) en tiempo real o *streaming*. La arquitectura lambda plantea una solución híbrida entre las dos arquitecturas anteriores, proponiendo que una no debe ser excluyente de la otra. Se podría decir, que este tipo de arquitectura combina lo mejor de las dos arquitecturas, resultados rápidos y procesado profundo a gran escala.

La arquitectura lambda ha sido la primera arquitectura en definir cómo el procesado *batch* y *stream* pueden trabajar en conjunto para resolver un amplio abanico de casos de uso. La premisa detrás de la arquitectura lambda es que debe permitir realizar consultas contra todo el conjunto de datos para obtener resultados. Esta premisa es cara en términos de recursos y eficiente, por lo tanto, la idea es precomputar los datos en conjuntos de vistas para realizar consultas contra las vistas. Este tipo de datos se le puede llamar: *Conjuntos de Datos Orientados a Consultas (QFD o Question Focused Datasets)*. La arquitectura lambda se puede dividir en tres capas (*layers*):

- **Batch Layer**: Esta capa es responsable de primero, almacenar el conjunto de datos maestro, que se encuentra en constante crecimiento, y segundo, computar arbitrariamente vistas de este dataset maestro. El proceso de computación de las vistas es continuo, así que cada vez que llegan nuevos datos al conjunto de datos maestro, se integran en la siguiente iteración de (re)computación. Este tipo de iteraciones de recomputación no se realizan frecuentemente. Corresponde a la sección verde de la *Figura 4.4*

- **Serving Layer:** La salida de la *batch layer* es un conjunto de ficheros planos que contienen las vistas precomputadas. Esta capa es la encargada de indexar estas vistas y ponerlas a disposición de las consultas que puedan llegar. Esta capa normalmente está constituida por una base de datos que se utiliza para consultar las vistas precomputadas. Por lo tanto la *batch layer* se encarga de almacenar y computar petabytes de datos y la *serving layer* se encarga de satisfacer las consultas a los datos de manera rápida e interactiva. Estas dos capas no satisfacen ningún requisito que esté relacionado con tiempo real, ya que *MapReduce* por definición está diseñado para alta latencia, pudiendo llevar un par de horas computar y propagar a la *serving layer* todo el conjunto de vistas. Corresponde a la sección azul de la *Figura 4.4*
- **Speed Layer:** Esta capa está diseñada para compensar la alta latencia de la capa *batch*, esto se consigue computando las vistas en tiempo real. Estas vistas calculadas en tiempo real sólo contienen la diferencia en datos con respecto a la capa *batch*. Es decir, que las vistas calculadas en tiempo real sólo se mantienen hasta la recomputación de la capa *batch*, una vez se propagan los datos de la capa *batch* a la capa *serving*, las vistas en tiempo real se pueden descartar. Corresponde a la sección roja de la *Figura 4.4*

En resumen, mientras que la capa *batch* está diseñada para ir recalculando en las vistas *batch* desde cero, la *speed layer* utiliza un modelo incremental, sus vistas aumentan a medida que se van recibiendo nuevos datos. Lo inteligente de la *speed layer* es que las vistas están diseñadas para ser transitorias, por lo tanto, una vez propagados los datos a través de la *batch layer* y la *serving layer*, los datos alojados en las vistas en tiempo real pueden descartarse. Esto se conoce como *aislamiento de complejidad* o *complexity isolation*. La parte más compleja de la arquitectura se inserta en una capa cuyos resultados son solo temporales.

Finalmente se utiliza una herramienta (como por ejemplo *Apache Impala*) para mezclar las vistas en tiempo real y por lotes para dar respuesta a las consultas de los datos.

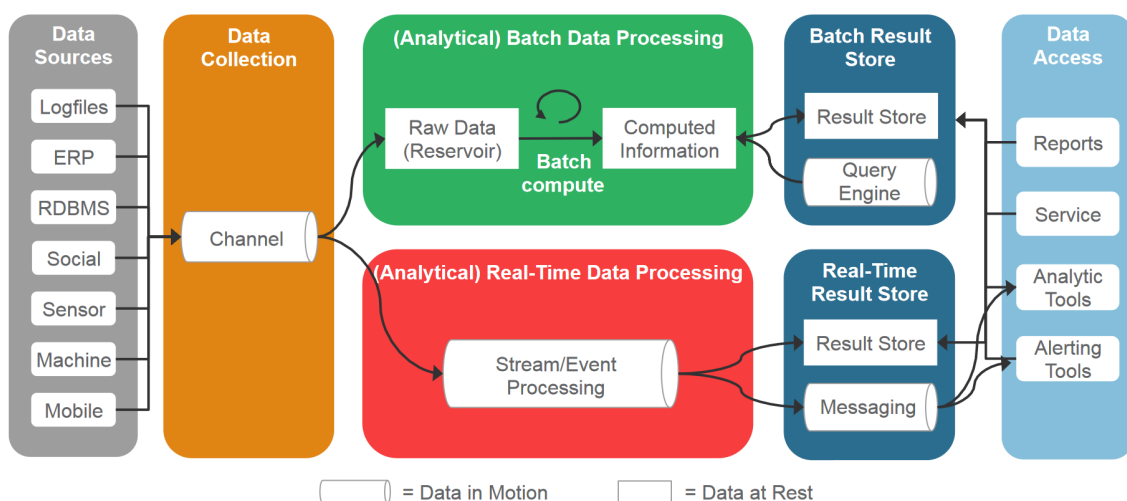


Figura 4.4: Arquitectura *Lambda* [8]

Las herramientas que se suelen utilizar en la arquitectura *Lambda*, agrupadas según la fase del flujo de datos dentro de la arquitectura son las siguientes:

- Ingestión: Al igual que en la arquitectura tradicional, para la ingestión de *data at rest* alojados en base de datos se suele emplear *Sqoop* y para la ingestión de *data in motion* se suele usar *Flume* o *Kafka*. Desde aquí los datos se guardan en *HDFS* o se envían a las herramientas que los procesan en tiempo real.
- Procesamiento: Para la parte de procesamiento *batch* se utilizan las mismas herramientas que en la arquitectura tradicional, trabajos *MapReduce* con *Hive* o *Pig*, o procesamiento en memoria con *Spark*. Para la parte de procesamiento *streaming* se usan las mismas herramientas que en la arquitectura *CEP*, *Storm*, *Flink Spark Streaming* o *Samza*.
- Almacén de resultados: Los resultados se pueden guardar en una base de datos como *HBase* o *Cassandra*, o servir en un motor de consultas como *Hive* o *Spark SQL*.

4.2.4. Arquitectura kappa

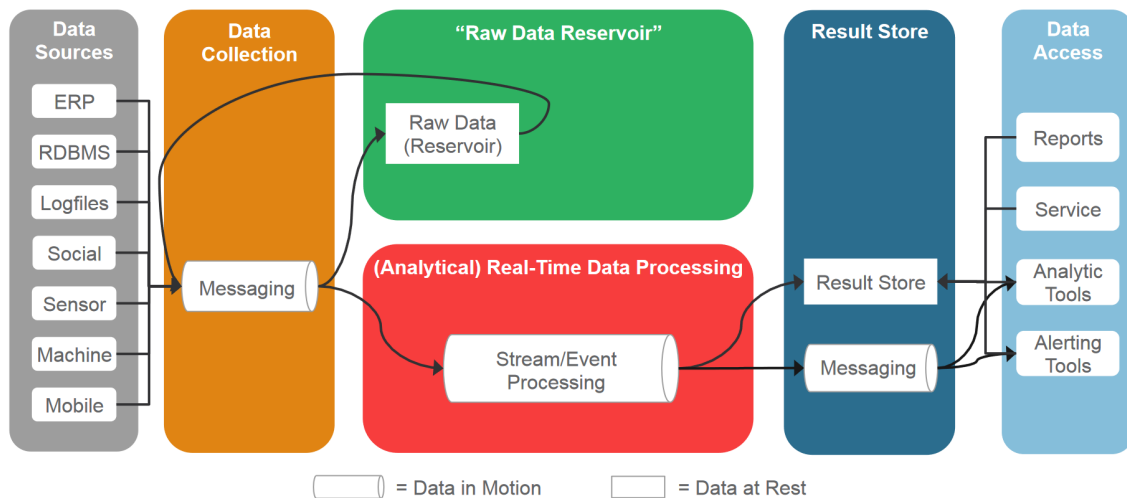
La arquitectura *kappa* surge de la simplificación de la arquitectura *lambda*. Esta arquitectura fue descrita por primera vez en el artículo [19] en el que se cuestiona la arquitectura *lambda* y plantean alternativas.

La arquitectura *kappa* propone eliminar la capa *batch* de la arquitectura *lambda* y dejar únicamente la capa *streaming*. Esta solución cobra sentido cuando lo que se calcula en la capa *batch* y en la capa *streaming* es lo mismo, lo que implica dos cosas: se está manteniendo dos códigos diferentes que hacen exactamente lo mismo, y durante todo el procesamiento se mantiene la sincronización entre ambas capas.

Actualmente se puede usar el procesamiento *batch* si la latencia no es un factor importante y el procesamiento *streaming* si la latencia sí es importante, pero solamente es recomendable usar ambas si es totalmente necesario. ¿Por qué la arquitectura *lambda* combina ambos procesamientos? Porque cada vez se necesitan sistemas mas complejos con un procesamiento de baja latencia. Se tienen dos tipos de procesamiento (*batch* y *streaming*) que no solucionan el problema, el primero es capaz de procesar grandes historicos de datos por lotes pero a una alta latencia, y el segundo es capaz de procesar a baja latencia pero no puede reprocesar los resultados. Esta combinación de ambos procesamientos aporta la solución que se estaba buscando (ser capaz de realizar consultas sobre todo el conjunto de datos a baja latencia), pero que parece una solución temporal (según [19]) y que no se prevee que se mantenga como paradigma en un futuro. Más bien parece una solución temporal limitada por las herramientas estándar.

La arquitectura *kappa* plantea que no es necesario reprocesar los datos en la capa *batch*, pudiéndose calcular en la capa *streaming* cuando los datos cambien. Los sistemas *streaming* ya conciben la noción de paralelismo, por lo tanto, sólo habría que aumentar el paralelismo para realizar el reprocesamiento de los datos.

Por otro lado, el procesamiento *streaming* no significa que no se pueda almacenar los datos en *HDFS*, sino que el reprocesamiento de los mismos no se ejecuta sobre *HDFS*. Si los datos se almacenan en *HDFS*, solo hay que usar una herramienta que transforme los

Figura 4.5: Arquitectura *Kappa* [8]

datos almacenados en *HDFS* a un *stream* de datos para su reprocesado, siempre manteniendo el orden en el que se generaron. El flujo de datos en esta arquitectura se puede ver en la *Figura 4.5*

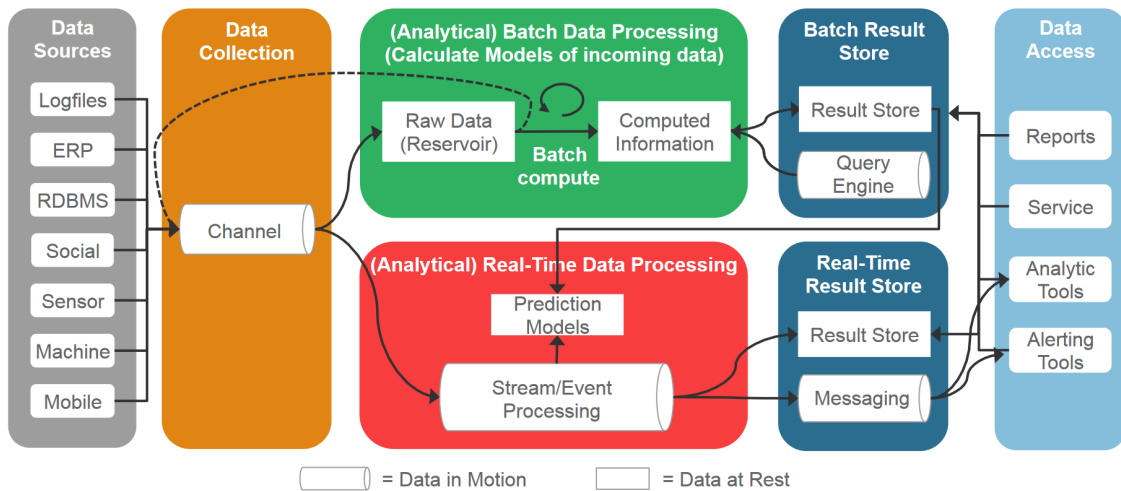
Las herramientas que se suelen utilizar en la arquitectura *Kappa*, agrupadas según la fase del flujo de datos dentro de la arquitectura son las siguientes:

- **Ingestión:** Como para la ingestión los datos siempre tienen que ser *data in motion* se suele usar *Flume* o *Kafka*. Los datos que no estén en *HDFS* o se guardan en *HDFS*.
- **Procesamiento:** Se usan herramientas de procesamiento *streaming* como son *Storm*, *Flink Spark Streaming* o *Samza*.
- **Almacén de resultados:** Los resultados se suelen guardar en una base de datos de alto rendimiento como es *Cassandra*, aunque también se puede utilizar un motor de consultas como *Apache Drill*.

4.2.5. Arquitectura unificada

En esta arquitectura se mantienen ambos planteamientos comentados en la *Sección 4.2.3, Arquitectura lambda*, y en *Sección 4.2.4, Arquitectura Kappa*. Aunque no exista mucha literatura sobre este tipo de arquitectura, parece una opción para arquitecturas grandes (o muy específicas) en las que es necesario realizar procesamiento *batch* sobre *HDFS* y un procesamiento *streaming* extrayendo los datos de *HDFS* en forma de *stream de datos*, mezclándolos con los datos en *streaming* que llegan a la plataforma. Es decir que hace uso de datos en movimiento y datos en reposo independientemente.

Como se ve en la *Figura 4.6*, una forma de relacionar las dos capas de la plataforma es utilizar los datos refinados por la capa *batch* para entrenar un modelo predictivo en la capa *streaming* y de ese modo realizar análisis predictivos sobre los datos que pasan por ella.

Figura 4.6: Arquitectura *Unificada* [8]

En este tipo de arquitectura tiene cabida cualquier herramienta del ecosistema Big Data. Un ejemplo de uso de herramientas dentro de esta arquitectura sería, usar *Kafka* para la gestión de mensajes y datos *streaming* tanto de entrada como de salida, usar *HDFS* para el almacenamiento de los datos tanto refinados como en bruto, utilizar *Spark* para el procesamiento de los datos en *batch* y *Spark SQL* para servir los datos refinados, a partir de esos datos refinados usar *Spark MLlib* para crear modelos predictivos, usar *Spark Streaming* para usar los modelos predictivos y para procesar los datos que se recogen en tiempo real.

4.3. Propuesta de plataforma Big Data

Tradicionalmente la monitorización y recopilación de experimentos se realiza de forma local, conectando el hardware de medición a un PC local y almacenando los datos de forma local mediante el software que acompaña al sistema de medición. El hardware y software habitualmente lo distribuyen marcas de productos de medición y *testing*. Un ejemplo de estas marcas son: *National Instruments*, *Teradyne*, *Tektronix*, *Agilent* o *Dewesoft*, entre otras. Este tipo de hardware y software se convertiría en la fuente principal de datos para el sistema Big Data que se va a diseñar. Por lo tanto, es importante encontrar una forma de conectar el software de medición con la plataforma Big Data, tanto si se quiere hacer procesamiento en *streaming* como si se quiere llevar el resultado de la medición a la plataforma para su almacenamiento y posterior procesamiento (formato *batch*).

Una aplicación del procesamiento *streaming* puede ser que, mientras se van tomando las mediciones para los dispositivos, la plataforma, a través de un modelo predictivo, procese la información que recibe y devuelva, en tiempo real, el tiempo de vida estimado que le queda al dispositivo. El procesamiento *batch*, a parte de la creación y reentrenamiento de modelos predictivos, puede ser aplicada para el control de calidad de los dispositivos, por ejemplo, clasificar y predecir los dispositivos defectuosos, adaptándose (por ejemplo) al modelo de calidad 6σ .

En una primera aproximación, la plataforma estaría estructurada como se ve en la

Figura 4.7. Se tienen unas fuentes de datos y unos consumidores de datos, y entre medias de los dos, la plataforma que almacena y procesa los datos.

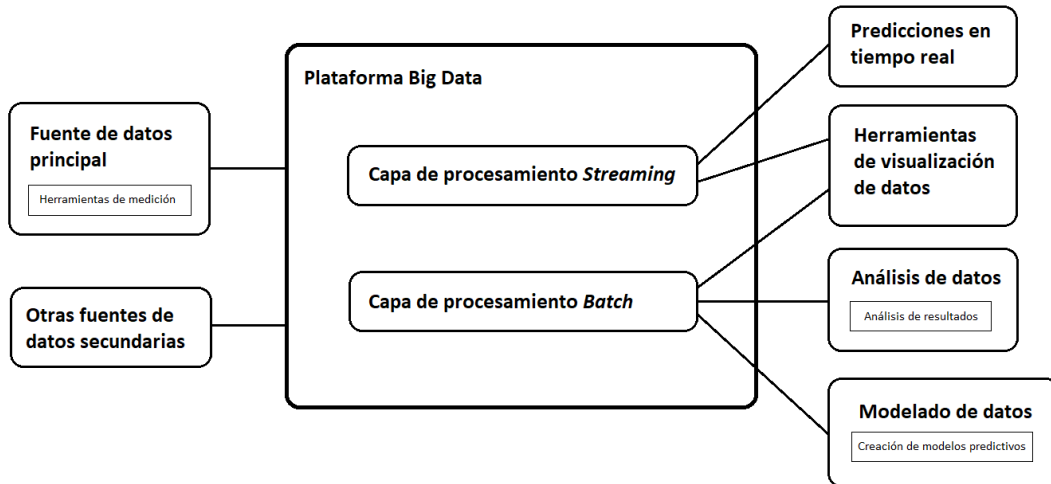


Figura 4.7: Primera aproximación a la estructura de la plataforma

Como ya se ha comentado, la plataforma se va a nutrir principalmente de los datos recogidos por los experimentos sobre los dispositivos (*Fuente de datos principal* en la Figura 4.7), además se tienen que contemplar *Otras fuentes de datos secundarias* si se quiere enriquecer los datos de alguna manera, o en algún momento se contempla la posibilidad de añadir otra fuente de datos.

Tal y como se ha planteado el uso de la plataforma, abarca el problema del procesamiento *batch* y el problema del procesamiento *streaming*, por lo tanto en este escenario se descartan la arquitectura *tradicional* y la arquitectura *CEP*. Como se van a tener dos escenarios distintos, uno de procesamiento por lotes para el análisis y la creación de modelos en el que no es importante la latencia, y otro en el que se realizan predicciones en tiempo real del *RUL* en el que sí que importa la latencia, ya que cambia a lo largo del tiempo, y además en cada escenario se realizan acciones diferentes, se opta por la arquitectura unificada.

4.4. Diseño de la plataforma

4.4.1. Fuente de los datos

La principal fuente de datos de la plataforma Big Data va a ser los datos generados durante la experimentación con los MOSFET. En este sentido se pueden encontrar los datos de dos formas, en movimiento, es decir, se recogen mientras se generan, o estáticos, es decir, los datos se generaron sin tener conexión con la plataforma y es necesario

volcarlos en la plataforma para incluirlos en las transformaciones y en el análisis de los datos tras dichas transformaciones. Los datos se generan con sensores de medida, estos sensores de medida están conectados a un sistema de recopilación de datos, que a su vez está conectado a una máquina. El sistema de recopilación de datos se comunica con la máquina a través de un software especial para este tipo de sistemas de recopilación, que se llama *LabVIEW*. *LabVIEW* es el encargado de transmitir los datos en tiempo real o almacenarlos para su posterior transmisión a la plataforma. *LabVIEW* proporciona distintos protocolos de comunicación en función de las necesidades del usuario.

A parte de los datos de los experimentos, se tiene en cuenta que se debe poder incluir en cualquier momento otra fuente secundaria de información que pueda enriquecer la fuente principal de los datos.

4.4.2. Ingesta de datos

Una vez definida la fuente, se necesita recolectar esos datos producidos e introducirlos dentro de la plataforma, ya sea como datos en tiempo real o datos estáticos por lotes. Para este proceso de ingesta se va a utilizar dos herramientas: *Apache Flume* y *Apache Kafka*. Primero se va a hablar sobre ellas y después cómo interactúan entre ellas y con el resto de herramientas de la plataforma que se está diseñando.

Apache Kafka es un proyecto de intermediación de mensajes. Es una plataforma unificada de manipulación de datos en tiempo real de alto rendimiento y baja latencia. Permite publicar y suscribirse a *streams* de datos, almacenar los datos de forma duradera y tolerante a fallos y procesar esos datos que llegan inmediatamente.

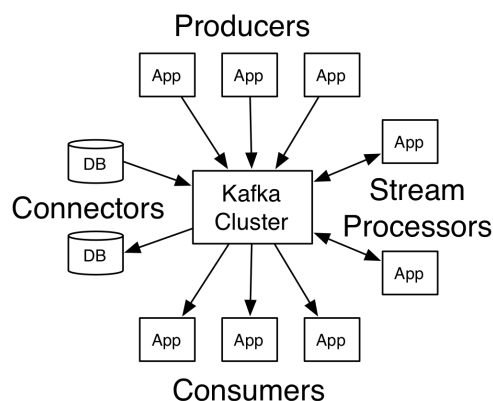


Figura 4.8: *Apache Kafka*

Apache Kafka interactúa con cuatro tipos de elementos:

- Productores o *Producers*: Aplicaciones que publican *streams* de datos.
- Consumidores o *Consumers*: Aplicaciones que se suscriben a los diferentes canales de datos.
- Procesadores de flujo o *Stream Processors*: Aplicaciones que consumen y publican datos indiferentemente.

- Conectores o Connectors: Consumidores o productores reusables que permiten conectarse a aplicaciones o a sistemas existentes a *Kafka*.

Apache Kafka realiza la comunicación entre los clientes y servidores con un protocolo TCP simple, de alto rendimiento, independientemente del idioma.

Apache Flume es un servicio distribuido, fiable y de alta disponibilidad capaz de recolectar, agregar y mover eficientemente grandes cantidades de datos.

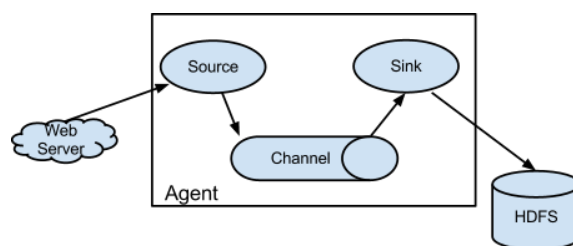


Figura 4.9: *Apache Flume*

Un agente *Flume* está formado por una *fuentes* o *source*, un *canal* o *channel* y un *sumidero* o *sink*. A través del agente fluyen eventos, que funcionan como unidad de datos. En resumen, *Flume* aloja los componentes que hacen que los eventos fluyan desde una fuente externa a un destino externo, como se muestra en la *Figura 4.9*. De esta manera se pueden configurar el sumidero y la fuente a medida de las necesidades.

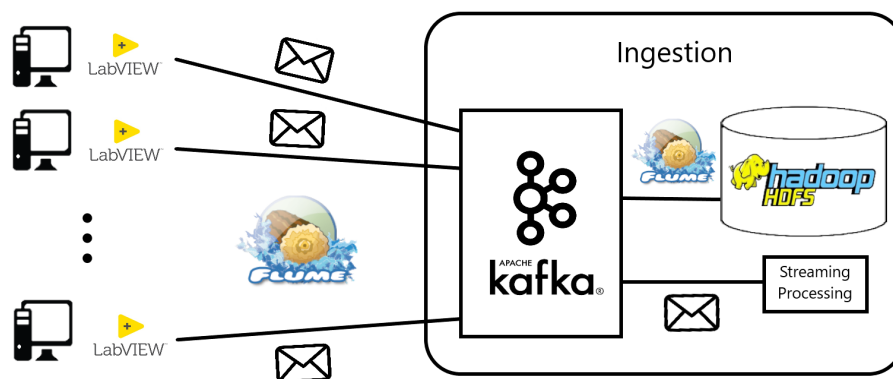


Figura 4.10: Diseño de la ingestión de datos

Dentro de la plataforma que se está diseñando, *Apache Kafka*, en esta sección de ingesta de datos, toma un rol de orquestador entre los flujos de datos de entrada y las herramientas que van a almacenar y procesar los datos. *Apache Flume* va a ser el encargado de transportar los datos, tanto desde la fuente definida hasta *Kafka*, como desde *Kafka* a *HDFS*.

4.4.3. Diseño del *Data Lake*

Como ya se ha comentado en la sección 4.2.1 (*arquitectura tradicional*), el *data lake* es el repositorio donde se van a almacenar los datos en bruto, en este caso, el resultado de

la experimentación con los MOSFET. En escenario también se van a almacenar distintas fases de procesado o refinado de los datos. En esta plataforma, el *data lake* está orientado al almacenamiento de los datos en *HDFS*, esto obliga a diseñar una organización de directorios dentro del sistema de archivos distribuido del sistema. La organización de directorios que se plantea es la siguiente:

```
/ <fuente_dato> / <tipo_dispositivo> / <nivel_procesado> / <año> / <mes> / <día> / <test> /
```

- *fuente_dato*: Nombre explicativo del lugar de donde provienen los datos ingeridos en la plataforma. En el caso de esta solución sólo se tendría los datos de las mediciones, el nombre del directorio en este caso podría ser *testing*.
- *tipo_dispositivo*: En el caso de esta solución solo se tienen MOSFET, pero se pueden realizar mediciones sobre otros tipos de dispositivos o incluso indicar el modelo del dispositivo, un ejemplo de nombre de directorio en este nivel podría ser *mosfet_model_m*.
- *nivel_procesado*: Este nivel de directorio contiene el nivel de procesado de los datos, por ejemplo muestreos y submuestreos de los datos. El directorio que está obligatoriamente en este nivel es *raw_data* que contiene los datos en bruto.
- *año*: Nivel de directorio para clasificar los experimentos por años.
- *mes*: Nivel de directorio para clasificar los experimentos por meses.
- *día*: Nivel de directorio para clasificar los experimentos por días.
- *test*: En este nivel se encuentran los directorios de los tests con todos los datos que genera cada test dentro. El nombre de los directorios de este nivel debe ser un identificador único, puede ser simplemente el número del test en ese día.

De esta manera los datos de un test podrían estar por ejemplo en esta ruta:

```
/testing/mosfet_model_m/raw_data/2018/03/21/test_8/
```

Para el problema que se plantea en este trabajo se tiene solo una fuente de datos y una sola línea de procesado de datos, pero en el momento que el sistema empieza a crecer son necesarias herramientas de gobierno de datos. Estas herramientas permiten manejar los metadatos, identificar el flujo de procesamiento que han tenido los datos o bajo qué ubicaciones se encuentran qué datos, además de controlar qué herramientas tienen permiso de manejar qué datos. En este contexto entran en juego *Apache Atlas* y *Apache Ranger*.

4.4.4. Procesado y modelado de datos

Como ya se había comentado en secciones anteriores, existen dos tipos de datos dentro de la plataforma que se está diseñando: datos estáticos y datos en movimiento. Estos dos tipos de datos plantean dos escenarios de cara al procesado:

- Procesado de datos almacenados en *HDFS*: muestreo, análisis y creación de modelos predictivos.
- Procesado del flujo de datos aplicando los modelos predictivos creados.

Para el procesamiento de los datos almacenados se podría usar herramientas que realizan operaciones *MapReduce* sobre disco como son *Apache Hive* o *Apache Pig*, pero en este caso encaja mucho mejor *Apache Spark* que va a permitir dar solución a ambos escenarios y relacionarlos entre sí.

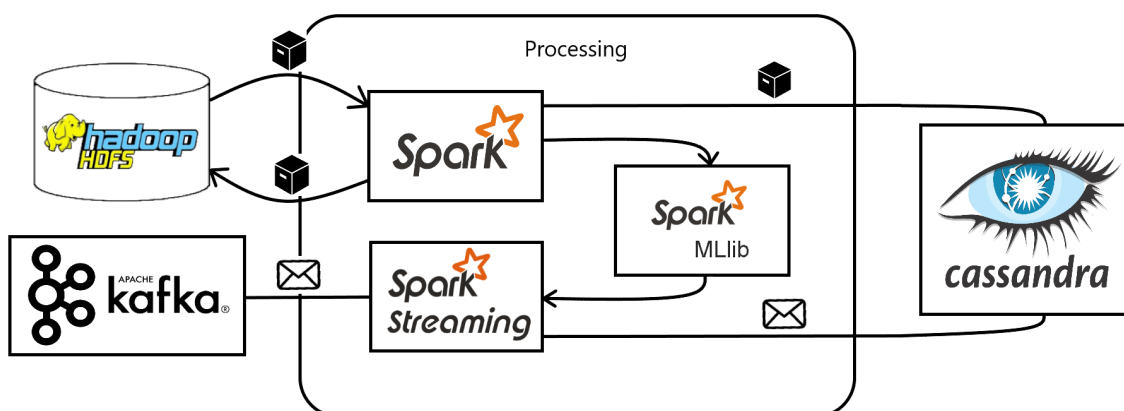


Figura 4.11: Diseño del procesamiento y modelado de datos

En el primer escenario, el procesamiento de los datos almacenados en *HDFS*, lo que se va a realizar es ciertos procesados por lotes a los datos en bruto para que finalmente sean fácilmente añadidos a un modelo predictivo o que sean fácilmente analizables, por ejemplo, realizar distintos tipos y niveles de muestreo en los datos para reducir su volumen e intentar reducir también el ruido del muestreo. Para la creación de modelos predictivos se puede usar *Apache Spark MLlib* y llevarlos al segundo escenario.

En el segundo escenario, procesamiento de flujo de datos, se va a coger los datos de la experimentación, procesarlos para que sea posible su paso por el modelo predictivo creado en el primer escenario. En este segundo escenario, para el procesamiento de flujos de datos se puede usar *Spark Streaming*. En nuestra solución, el ejemplo de aplicación es para el cálculo del tiempo de vida (*RUL* o *Remaining Useful Life*) de los MOSFET.

Tanto para el escenario primero como para el segundo, el resultado del procesamiento de datos se almacena en *Apache Cassandra*. Esta base de datos va a permitir que los datos procesados sean accesibles desde el exterior incluso para el procesamiento en tiempo real.

4.4.5. Automatización de las tareas

Existen herramientas en el ecosistema Big Data que permiten controlar y diseñar los flujos de datos (no en el sentido de tiempo real, sino por qué herramientas pasan y qué procesamiento se realiza en cada una de ellas) dentro de la plataforma. *Apache Oozie* es la herramienta que se encarga de este tipo de tareas. El caso de uso más frecuente de *Apache Oozie* es el diseño de un flujo de datos para el procesamiento por lotes de los datos, este

flujo se interpreta un grafo acíclico en el que se especifica la secuencia de acciones a ejecutar (*Oozie Workflow*), estos flujos de trabajo se suelen disparar por tiempo o por disponibilidad de los datos.

Esta herramienta, dentro de la solución que se está diseñando, encaja en la parte de procesado por lotes de los datos con *Apache Spark* desde *HDFS*. Diseñando primero el grafo de transformaciones y segundo cada cuánto tiempo se quiere ejecutar el procesado de los datos alojados en *HDFS*.

4.4.6. Capa de servicio de datos

Habitualmente todas las arquitecturas Big Data tienen una capa de servicio en la que exponen los datos procesados y refinados, tanto por la capa de procesamiento por lotes como por la capa de procesamiento en tiempo real. Normalmente la herramienta que se utiliza en esta capa es una base de datos de alta disponibilidad y baja latencia como son *Apache HBase* o *Apache Cassandra*, aunque en esta capa de servicio también se pueden encontrar casos en los que se realizan consultas directamente sobre *Apache Hive* o *Spark SQL*.

La herramienta elegida en el diseño de plataforma que se está realizando es *Apache Cassandra*. *Apache Cassandra* es una base de datos *NoSQL* distribuida de código abierto. Entre sus principales características se encuentra la escalabilidad y la disponibilidad. Trabaja con una arquitectura multinodo en la que los nodos se comunican entre si mediante un protocolo *P2P*, por lo tanto no tiene nodo maestro. Su modelo de datos está orientado a tablas donde se almacenan los datos en forma clave-valor. Las tablas se pueden gestionar en tiempo de ejecución permitiendo operaciones de creación, borrado o alteración sin detener las actualizaciones o las consultas.

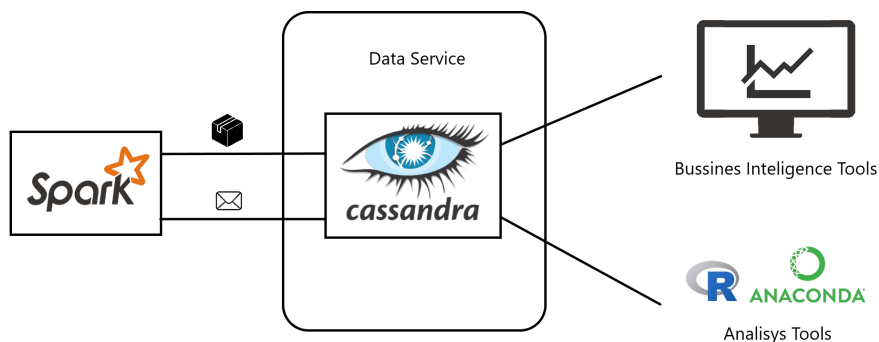


Figura 4.12: Diseño del servicio de datos

En este caso en *Apache Cassandra* se van a almacenar los datos de los muestreos que se realizan sobre los datos recogidos en los experimentos, además de los resultados de las predicciones del *RUL* de cada MOSFET. Los posibles consumidores de estos datos pueden ser herramientas de B.I. en las que se visualizan tanto los muestreos (datos estáticos) como las predicciones del *RUL* (datos en movimiento) en tiempo real. También pueden ser consumidores de estos datos los científicos de datos que trabajan con herramientas de análisis como son *R* o *Anaconda (Python)*, ya tienen los datos procesados y solo se tienen que centrar en realizar análisis descriptivos o predictivos.

4.5. Implementación de la plataforma

Una vez establecidas las necesidades que se tienen y cómo se da respuesta a través del diseño y las herramientas descritas en las secciones anteriores, es necesario plantear qué infraestructura va a soportar el diseño planteado.

A continuación se plantean las diferentes opciones que existen a la hora de elegir el tipo de infraestructura sobre la que se quiere montar la plataforma:

- Montado manual: Esta opción es la menos recomendable. Consiste en comprar cierto número de máquinas, conectarlas en red, e ir montando una a una todas las herramientas que se han descrito anteriormente. Este trabajo de construcción y configuración manual es muy tedioso e incluso puede dar muchos problemas de configuración y compatibilidad entre herramientas debido a compatibilidad entre versiones etc.
- Uso de una distribución: Esta opción consiste en seguir manteniendo la estructura física pero a diferencia de la anterior, no es necesario configurar una a una todas las herramientas en todas las máquinas manualmente, si no que se hace uso de una distribución *Apache Hadoop* que contiene todas las herramientas necesarias y compatibles entre sí para poder montar un clúster Big Data. Un ejemplo de estas distribuciones son *Cloudera*, *Hortonworks* o *MapR*.
- Recursos en la nube: Esta opción consiste en prescindir de máquinas físicas y alquilar los recursos en la nube, es decir, contratar un clúster Big Data a través de una compañía como puede ser *Amazon*, *Azure* o *Google*.
- Abstracción de los recursos: Esta opción consiste en introducir una capa de abstracción entre los recursos físicos y las herramientas que se integran en la plataforma (*Mesosphere*).

Que opción elegir de todas las expuestas depende de la empresa y sus recursos económicos. La opción *montado manual* sólo sería recomendable en el caso de que se quisiese construir algo muy concreto. La opción de los *recursos en la nube* puede ser una opción muy cómoda pero puede resultar muy costosa en términos económicos. La opción de abstracción de recursos parece una buena opción pero a cambio se debe conocer bien como funcionan las herramientas de abstracción. La opción más rápida y la que parece que se usa de forma mayoritaria es montar un clúster *Hadoop* con una de las distribuciones preconstruidas sobre un hardware propio.

4.6. Arquitectura final

El resultado final de la plataforma Big Data propuesta es el de la *Figura 4.13*. En resumen, los datos se recogen a través de *LabVIEW* que está conectado con la plataforma. *Apache Kafka* es el encargado de orquestrar todos los puntos de medición de los experimentos, es decir, se encarga de recopilar toda la información enviada por las distintas instancias de *LabVIEW*. *Apache Kafka* se encarga de enviar los datos a *HDFS* y al canal

de procesamiento *streaming*. Esta parte corresponde con la ingestión de datos en la plataforma. Una vez ingeridos los datos, se procesan de dos formas *streaming* y *batch*. Tal y como se ha planteado en las secciones anteriores, el procesamiento *batch* o por lotes va a ser el encargado de transformar los datos y construir modelos de predicción, y el procesamiento *streaming* va a ser el encargado de realizar predicciones en tiempo real sobre los datos ingeridos en la plataforma. Finalmente se implementa una capa de servicio de datos en la que se integran tanto los datos resultantes en el procesamiento *batch*, como los resultantes en el procesamiento *streaming*. Esta capa de servicio nutrirá de datos herramientas de *B.I* o aportará datos para el análisis de datos con herramientas no Big Data.

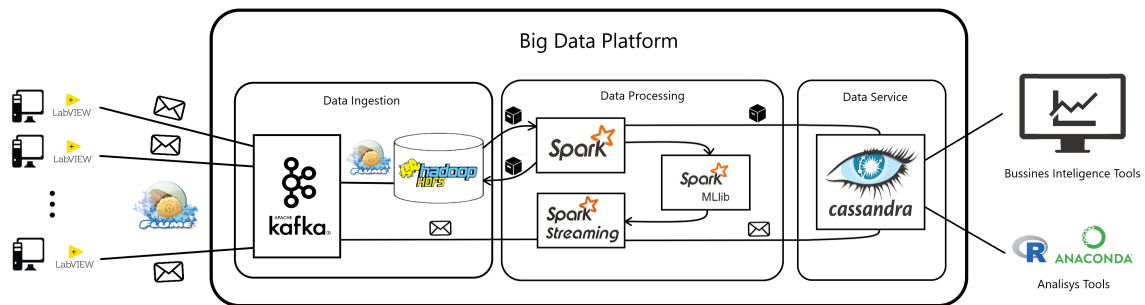


Figura 4.13: Arquitectura final del sistema

Finalmente la infraestructura sobre la que se construye la plataforma depende en parte de los recursos económicos de la empresa. Una implementación habitual es la construcción de un clúster con hardware propio con una distribución Big Data preconstruida como pueden ser *Cloudera* o *Hortonworks*.

Capítulo 5

Conclusiones

En este Trabajo Fin de Máster se ha abordado, en una primera parte, el problema de la prognosis de transistores MOSFET en la que, como punto de partida, se tenían las referencias [3], [4] y [5], y los datos [1] sobre los que trabajan dichas referencias. En una segunda parte se ha descrito un supuesto teórico en el que una empresa ficticia tiene que abordar el problema de la experimentación con transistores MOSFET y el problema de la prognosis a gran escala. En este contexto se ha planteado una hipotética plataforma Big Data que pretende dar solución al problema.

Para abordar el problema de la prognosis de transistores MOSFET ha sido necesario primeramente aprender sobre la teoría de este tipo de dispositivos y que fenómenos físicos se dan en ellos. Una vez realizado esto, se ha trabajado en dos líneas, por un lado, se ha trabajado en la comprensión, exploración y procesamiento de los datos, y por otro lado se ha buscado información sobre las técnicas utilizadas en los artículos de partida.

En relación con la primera línea de trabajo, gran parte del esfuerzo se ha empleado en la extracción y filtrado de los datos. En el ámbito de la ciencia de datos se habla de que el 80 % del esfuerzo se suele destinar a la extracción y filtrado de datos y sólo el 20 % restante está destinado al modelado de los datos. En este trabajo se ha cumplido prácticamente esa distribución del esfuerzo, ocupando gran parte del tiempo el obtener unos datos que sean modelables. En relación con la segunda línea de trabajo, se ha optado por aplicar en una fase inicial unos métodos de modelado más sencillos, como pueden ser la media horizontal o el ajuste por mínimos cuadrados, y en una segunda fase métodos más complejos, aunque finalmente (por falta de tiempo) sólo se ha podido aplicar el filtro de Kalman extendido. Curiosamente los métodos iniciales más sencillos obtienen mejores resultados que el filtro de Kalman extendido.

Respecto a los resultados obtenidos y el conjunto de datos analizados se quiere resaltar varios aspectos. El primer aspecto es que el conjunto de datos proporcionado era muy heterogéneo y ha sido necesario realizar una selección de aquellos tests que tuviesen al menos una hora de vida del dispositivo y unos rangos de temperatura razonables (entre 0°C y 260°C aproximadamente), teniendo que rechazar 25 de 42 tests. La suposición inicial que se tenía de que sería complicado que un modelo como el elegido en [3] pudiese ajustar todos los comportamientos se ha visto confirmado por los resultados. El segundo aspecto es que, incluso restringiéndonos a los datos de los tests que aparecen en las figuras 3.12 y 3.13, se siguen observando comportamientos con dinámicas muy distintas. Unos experimentos van a concluir mucho antes que otros, con lo cual conseguir valores de β en

el modelo de la ecuación (2.2) que sirvan en todos los casos sería complicado, como se ha constatado en los resultados.

Los resultados obtenidos indican que un método sencillo, que simplemente trata de caracterizar el comportamiento promedio de los dispositivos a lo largo de su vida, se comporta bien al final de la vida útil, donde se aprecian dinámicas más diferentes: algunas se ajustan al modelo esperado pero otras se asemejan más a un cambio abrupto. La técnica de la media horizontal realza, en promedio, las mejores predicciones al final de la vida útil. También hay que indicar que la técnica de ajuste por mínimos cuadrados se comporta bien en promedio, tanto para el modelo no lineal como para el linealizado.

Aún con todas estas limitaciones de desconocimiento del entorno de la toma de datos y de algunos parámetros de configuración de los modelos, los resultados que hemos obtenido usando nuestra versión del *EKF* son similares a los que se obtienen en el trabajo de referencia [3] y en general se comporta peor que los métodos anteriores.

Una última conclusión respecto a la prognosis de estos dispositivos MOSFET es que la ley de envejecimiento propuesta no es adecuada para describir la degradación de estos dispositivos, al menos en condiciones de envejecimiento acelerado.

Como líneas futuras de trabajo en esta primera parte del trabajo se plantean varias propuestas. La primera propuesta es continuar el trabajo realizado calculando los intervalos de confianza para los instantes de tiempo de las predicciones, en este trabajo sólo se ha abordado el problema muy superficialmente. La segunda propuesta es aprovechar los datos procesados y filtrados para ampliar por un lado los métodos predictivos y, por otro lado, las funciones que modelan el comportamiento de los transistores. En particular, los resultados sugieren que el proceso de envejecimiento se podría modelar mejor considerando que hay un punto de ruptura en los datos experimentales, sugiriendo la posibilidad de modelar el comportamiento con dos modelos diferentes, uno anterior, y otro posterior al punto de ruptura.

En la segunda parte del trabajo se ha realizado una recopilación de todos los tipos de arquitecturas que existen en la literatura para plataformas Big Data, acompañado de una descripción de cada una de las herramientas que se incluyen en ellas. A partir de una de estas arquitecturas se ha definido una plataforma Big Data con las herramientas que mejor encajan en el problema de la experimentación y prognosis de transistores MOSFET. Por limitaciones en el tiempo sólo se ha podido hacer una especificación de requisitos y no se ha podido hacer una definición de casos de uso asociados a estos requisitos.

Un aspecto a analizar en el futuro a la hora de elegir la plataforma es que habría que tener en cuenta la disponibilidad de métodos de regresión o disponer de bibliotecas que permitan integrar los métodos de predicción aquí tratados, además de otros similares a los que se discuten en [3],[4] y [5]. También sería interesante analizar la posibilidad de utilizar algoritmos en tiempo real, asumiendo que pudiese haber una inyección de datos muestreados a altas frecuencias directamente al sistema de almacenamiento y se necesitase su procesamiento inmediato.

Como otras líneas futuras de trabajo en esta segunda parte del trabajo se propone trabajar en una prueba de concepto para la arquitectura descrita. Para esto no es necesario montar un clúster Big Data, se puede probar en un entorno virtual, ya sean máquinas virtuales o contenedores *Docker* que alojen las imágenes (normalmente *sandbox*) de las distribuciones Big Data preconstruidas.

Bibliografía

- [1] Celaya, J., Saxena, A., Saha, S., and Goebel, K.. *MOSFET Thermal Overstress Aging Data Set*, NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA
<https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#mosfet>
- [2] International Rectifier. *IRF520NPbF HEXFET Power MOSFET*. Características del MOSFET modelo IRF520NPbF. Última visita 12/07/2018.
<http://ctms.engin.umich.edu/CTMS/Content/Activities/MOS-irf520npbf.pdf>
- [3] Celaya, J., Saxena, A., Saha, S., & Goebel, K.. *Prognostics of Power MOSFETs under Thermal Stress Accelerated Aging using Data-Driven And Model-Based Methodologies*. NASA Aviation Safety Program, projects IVHM and SSAT.
- [4] Celaya, J., Saxena, A., Wysocki, P., Saha, S., & Goebel, K.. (2010a) *Towards Prognostics of Power MOSFETs: Accelerated Aging and Precursors of Failure*. Paper presented at the Annual Conference of the Prognostics and Health Management Society 2010.
- [5] Celaya, J., Patil, N., Saha, S., Wysocki, P., & Goebel, K.. (2009) *Towards Accelerated Aging Methodologies and Health Management of Power MOSFETs (Technical Brief)*. Paper presented at the Annual Conference of the Prognostics and Health Management Society 2009.
- [6] Welch, G., Bishop, G.. *An Introduction to the Kalman Filter*. Última actualización 24 Julio de 2006. Última visita 12/07/2018.
http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
- [7] *Intervalo de confianza para la predicción de un valor en el modelo*. Última visita 18/07/2018.
<http://www.ub.edu/stat/GrupsInnovacio/Statmedia/demo/Temas/Capitulo13/B0C13m1t9.htm>
- [8] Schmutz, G.. *Big Data Architecture*. Slideshare. 26/09/2015,
<https://www.slideshare.net/gschmutz/big-data-architecture-53231252>

- [9] Schmutz, G.. *Architecture of Big Data Solutions*. Slideshare. 18/12/2017,
<https://www.slideshare.net/gschmutz/architecture-of-big-data-solutions-84394952>
- [10] Parscale, L.. *Big Data Infrastructure Decisions: Data in Motion vs. Data at Rest*. Virtualization & Cloud Review. 11/05/2014,
<https://virtualizationreview.com/articles/2014/11/05/data-in-motion-and-at-rest.aspx>
- [11] *Data lake: definición, conceptos clave y mejores prácticas*. Power Data. Última visita 18/06/2018,
<https://www.powerdata.es/data-lake>
- [12] DATAVERSITY. *Data Lake Architecture*. Slideshare. 10/10/2017. Última visita 19/07/2018.
<https://es.slideshare.net/Dataversity/data-lake-architecture>
- [13] Fowler, M.. *DataLake*. Sito web de Martin Fowler. ThoughtWorks. 05/02/2015. Última visita 19/07/2018.
<https://martinfowler.com/bliki/DataLake.html>
- [14] *How to Guide: Architecture Patterns to Consider When Designing an Enterprise Data Lake*. Cloud Technology Partners, Inc., a Hewlett Packard Enterprise company. Última visita 19/07/2018.
<https://www.cloudtp.com/doppler/how-to-guide-architecture-patterns-to-consider-when-designing-an-enterprise-data-lake/>
- [15] Hausenblas, M.,Bijnens, N.. *Lambda Architecture*. A repository dedicated to the Lambda Architecture (LA). Última visita 19/07/2018.
<http://lambda-architecture.net/>
- [16] Kinley, J.. *The Lambda architecture: principles for architecting realtime Big Data systems*. Última visita 19/07/2018.
<http://jameskinley.tumblr.com/post/37398560534/the-lambda-architecture-principles-for>
- [17] Hausenblas, M.. *Applying the Big Data Lambda Architecture*. Dr.Dobb's. 12/11/2013. Última visita 19/07/2018.
<http://www.drdobbs.com/database/applying-the-big-data-lambda-architecture/240162604>
- [18] Prokopp, C.. *Lambda Architecture: Achieving Velocity and Volume with Big Data*. Big Data Science and Cloud Computing. 07/05/2014. Última visita 19/07/2018.
<http://www.semantikoz.com/blog/lambda-architecture-velocity-volume-big-data-hadoop-storm/>

- [19] Kreps, J.. *Questioning the Lambda Architecture*. O'Really, 2 Julio de 2014. Última visita 13/07/2018,
<https://www.oreilly.com/ideas/questioning-the-lambda-architecture>
- [20] Uesugi, S.. *kappa-architecture.com*. Repository dedicated to Kappa Architecture. Última visita 19/07/2018.
<http://milinda.pathirage.org/kappa-architecture.com/>
- [21] Samizadeh, I.. *A brief introduction to two data processing architectures - Lambda and Kappa for Big Data*. Towards Data Science. Última visita 19/07/2018.
<https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb>
- [22] *LabVIEW*. National Instruments. Última visita 19/07/2018.
<http://www.ni.com/es-mx/shop/labview.html>
- [23] *Soluciones de Big Analog Data*. National Instruments. 09/06/2016. Última visita 19/07/2018.
<http://www.ni.com/white-paper/14667/es/>
- [24] *Guía Práctica para Conectar LabVIEW al Internet Industrial de las Cosas*. National Instruments. 18/07/2017. Última visita 19/07/2018.
<http://www.ni.com/white-paper/53954/es/>
- [25] *Apache Flume*. The Apache Software Foundation. Última visita 19/07/2018.
<https://flume.apache.org/index.html>
- [26] *Apache Kafka*. The Apache Software Foundation. Última visita 19/07/2018.
<https://kafka.apache.org/intro>
- [27] Shapira, G., Holoman, J.. *Flafka: Apache Flume Meets Apache Kafka for Event Processing*. Cloudera Engineering Blog. 06/11/2014. Última visita 19/07/2018.
<http://blog.cloudera.com/blog/2014/11/flafka-apache-flume-meets-apache-kafka-for-event-processing/>
- [28] *Using Apache Kafka with Apache Flume*. Cloudera Documentation. Última visita 19/07/2018.
https://www.cloudera.com/documentation/kafka/latest/topics/kafka_flume.html
- [29] *Apache Spark*. The Apache Software Foundation. Última visita 19/07/2018.
<https://spark.apache.org/>
- [30] *Spark Streaming + Kafka Integration Guide (Kafka broker version 0.10.0 or higher)*. The Apache Software Foundation. Última visita 19/07/2018.
<https://spark.apache.org/docs/latest/streaming-kafka-0-10-integration.html>

- [31] *Apache Oozie*. The Apache Software Foundation. Última visita 19/07/2018.
<http://oozie.apache.org/>
- [32] *Apache Cassandra*. The Apache Software Foundation. Última visita 19/07/2018.
<http://cassandra.apache.org/>
- [33] Schmutz, G.. *Real-Time Analytics with Apache Cassandra and Apache Spark*. SlideShare. 25/10/2017. Última visita 19/07/2018.
<https://es.slideshare.net/gschmutz/realtime-analytics-with-apache-cassandra-and-apache-spark>
- [34] Aldo. *Primeros pasos con Apache Cassandra*. Geeky Theory. Última visita 19/07/2018.
<https://geekytheory.com/primeros-pasos-con-apache-cassandra>
- [35] *Cloudera*. Cloudera, Inc. Última visita 19/07/2018.
<https://www.cloudera.com/>
- [36] *Hortonworks*. Hortonworks Inc. Última visita 19/07/2018.
<https://es.hortonworks.com/>
- [37] *MapR*. MapR Technologies, Inc. Última visita 19/07/2018.
<https://mapr.com/>
- [38] *Mesosphere*. Mesosphere, Inc. Última visita 19/07/2018.
<https://mesosphere.com/dcos>

Apéndice A

Apéndice: Contenido del CD

El contenido del CD entregado se estructura de la siguiente manera:

- *doc*: directorio que contiene esta memoria y los resultados de los modelos aplicados en un subdirectorio llamado *anexos*. El directorio *anexos* contiene:
 - *1-gráficas_seleccion_tests*: este directorio contiene las gráficas utilizadas para determinar que tests eran válidos para aplicar métodos de estimación del *RUL*.
 - *2-gráficas_resultados*: este directorio contiene las gráficas en las que se describe como ajusta la curva del método propuesto con los datos para distintos instantes de tiempo, y en que instante el ajuste corta el umbral de ruptura.
 - *3-tablas_resultados*: este directorio contiene los resultados de todos los tests seleccionados para todos los métodos propuestos.
- *src*: directorio que contiene todo el código desarrollado en el transcurso del proyecto. Este directorio contiene a su vez:
 - *conversor_matlab*: contiene el código *MATLAB* desarrollado para transformar los ficheros *.mat* a formato *CSV*. Este directorio contiene el instalador de la aplicación y los códigos fuente.

Una vez instalada la aplicación solo es necesario indicar el directorio donde se encuentran los ficheros *.mat*, seleccionar que fichero se quiere convertir y el directorio de salida para la conversión (se recomienda usar el mismo que el de la entrada).
 - *notebooks*: contiene los *notebooks* de *Jupyter Notebook* desarrollados.
 - *script_procesado.py*: *Script* que coge los datos en formato *CSV* de la conversión y realiza las tres fases de procesamiento: muestreo de los *transient*, normalización por temperatura y media por minuto.
- *data*: contiene los datos generados en cada uno de los procesados descritos en la memoria.