UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

DISEÑO DE TÉCNICAS DE ADMINISTRACIÓN DE ENERGÍA PARA NODOS SENSORES DE RED.

TRABAJO DE TESIS

PARA OBTENER EL GRADO DE

MAESTRO EN MECATRÓNICA

PRESENTA

ING. OSCAR EMMANUEL MANZANILLA AVILA

Director

DR. JAVIER VÁZQUEZ CASTILLO

Co-Director

DR. ALEJANDRO ARTURO CASTILLO ATOCHE

Asesores

DR. GUILLERMO BECERRA NUÑEZ

DR. JAIME SILVERIO ORTEGÓN AGUILAR

DRA. EDITH OSORIO DE LA ROSA

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
ÁREA DE TITULACIÓN

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

CONAHCYT
CONSEJO NACIONAL DE HUMANIDADES CIENCIAS Y TECNOLOGÍAS

CHETUMAL QUINTANA ROO, MÉXICO, JUNIO DEL 2023

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

TRABAJO DE TESIS BAJO LA SUPERVISIÓN DEL COMITÉ DEL PROGRAMA DE MAESTRÍA Y APROBADA COMO REQUISITO PARA OBTENER EL GRADO DE:

MAESTRO EN MECATRÓNICA

COMITÉ SUPERVISOR

DIRECTOR: _____

DR. JAVIER VÁZQUEZ CASTILLO

CO-DIRECTOR: _____

DR. ALEJANDRO ARTURO CASTILLO ATOCHE

ASESOR _____

DR. GUILLERMO BECERRA NUÑEZ

ASESOR: _____

DR. JAIME SILVERIO ORTEGÓN AGUILAR

ASESOR: _____

DRA. EDITH OSORIO DE LA ROSA

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

ÁREA DE TITULACIÓN

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

CHETUMAL QUINTANA ROO, MÉXICO, JUNIO DEL 2023

RESUMEN

Las Redes de Nodos de Sensores (WSN por sus siglas en ingles) han tenido un importante implementación en la actualidad, al ser una alternativa para solucionar el alto impacto ambiental que han generado las redes convencionales. Sin embargo, el problema que se genera al implementar las WSNs es el cambio de baterías lo que incrementa su alto consumo e impacto ambiental. La solución que se ha propuesto para reducir el consumo excesivo de baterías es controlar el consumo enérgetico de los WSN. Este proyecto de tesis presenta una alternativa de funcionamiento de WSN basada en el control del consumo energético de las WSN mediante el control de envío de datos basado en el aprendizaje de las IA, tomando como referencia métodos de control de energía como son el algoritmo WOS y el filtrado de Kalman, con el fin de incrementar el tiempo de vida útil de la bateria del nodo sensor, para reducir el consumo de baterias, reducir la cantidad de dispositivos de repuestos utilizados, y reducir el impacto ambiental generado por la WSN.

AGRADECIMIENTOS

A DIOS, por habernos brindado la oportunidad de llevar a cabo esta experiencia, por ayudarme a encontrar la sabiduría necesaria y ponerme en los momentos indicados y a las personas correctas en mi camino.

A mis PADRES, ustedes quienes me apoyaron incondicionalmente durante la licenciatura, me brindaron el mismo apoyo incondicional durante la maestría.

A mis Hermanas, la Lic. Loreli Vianey Manzanilla Avila y la Lic. Dafne Selene Manzanilla Avila, por estar cuando mas las necesité, por apoyarme y alentarme a seguir para concluir mis estudios.

A mi ASESOR, el Dr. Javier Vázquez Castillo, por su gran apoyo, no solo como maestro o asesor, sino como amigo, usted me demostró que puede llegar a cumplir mis metas y que tengo la capacidad de alcanzar lo que me proponga además de brindarme su apoyo para alcanzar mis metas.

A mis compañeros y AMIGOS los Ingenieros Araceli de la Cruz Franco, Eduardo Omar Ríos Arreola, Edwin Omar Pinto Matus, Brisa Argentina Martín Lara, a la Lic. Adriana Guadalupe Días Montero, la Lic. Yamileth Morado Sánchez, la Lic. Dayana Alvarado Peña, la Lic. Arith Alejandra Abreu Hernandez, el Br. José Luis Sala Figueroa, el Lic. Eduardo Mikhail Panting Castillo, el Lic. Christian David Herrera López, el Lic. Cleiver Otep Saucedo López y a mi AMIGO Y HERMANO el Lic. Jose Miguel Martín Aguilar. Gracias por todos los momentos que pasamos, por las rizas y los llantos, por estar conmigo en todo este proceso de la maestría y por entregarme la amistad tan valiosa que tenemos.

A mis maestros de la carrera, gracias por su tolerancia y dedicación al momento de sus enseñanzas, la pandemia nos trajo un nuevo reto a todos, y les estoy agradecido por recorrer esté nuevo camino que nadie estaba preparado para afrontar de esa manera.

DEDICATORIA

A mis padres y a mis hermanas que me apoyaron durante el periodo de licenciatura, y ahora estoy de frente a ustedes obteniendo el grado de maestría, gracias por estar allí apoyándome y por creer en mí y brindarme todo su amor, aun en los momentos mas difíciles, siempre me brindaron el apoyo que necesitaba y a su manera me alentaban a levantarme y seguir. Gracias por ser quienes son y por nunca dejarme solo a pesar de todo.

A mis amigos José Luis, Mikhail, Christian y Cleiver. Ustedes son de las amistades que me trajo esta aventura, las mas inesperadas, y aun así de las que mas aprecio, a pesar de que somos un grupo de videojuegos, fuera de allí me ayudaban, apoyaban y alentaban, los amigos pueden surgir de la manera mas inesperada, pero a ustedes los considero ya mi familia, gracias por todo y en parte esta tesis la avance por sus ánimos que me daban, por eso de igual manera se las dedico.

A mis amigos Aida, Araceli, Omar, Edwin, Miguel, Brisa, Yamileth, Dayana y Adriana. Ustedes siempre me apoyaron siempre estuvieron ayudándome y siempre estuvieron junto a mi incondicionalmente, en las buenas y en las malas. Gracias por todo y gracias por los momentos que pasamos juntos.

A Arith Alejandra Abreu Hernandez, me mostraste lo que es una amiga incondicional, aquella que se preocupa por ti sin importar nada, quien te quiere incondicionalmente, me motivaste cuando mas lo necesitaba y siempre estuviste allí, te volviste alguien importante para mi a pesar de todo lo que vivimos, te agradezco de corazón, todo el cariño y sorprendentemente el amor que me brindaste, sin tu riza y motivación, tal ves no hubiera encontrado el camino de nuevo para terminar mi proyecto ya que siempre me ayudabas a estar feliz y a avanzar, siempre estaré agradecido y siempre estaré contigo.

A mi asesor de tesis y amigo el Dr. Javier Vázquez Castillo, usted creyó en mi incluso antes de entrar a la carrera, me apoyo me alentó y me ayudo, gracias por sus consejos y guías, por motivarme a terminar por ayudarme a plasmar una idea adecuada a mi conocimiento, y por los momentos de alegría que pasamos durante todo este proceso.

Por ultimo a José Miguel Martín Aguilar, Brisa Argentina Martín Lara y Araceli de la Cruz Franco, Al final de la carrera pasaron cosas inesperadas, que me llevaron a salirme del camino, a pesar de no ser mi familia ustedes estuvieron allí, me alentaron, me apoyaron y no me dejaron volver a caer en ese vacío nuevamente, les agradezco su apoyo incondicional y este trabajo de tesis, culminación de mis esfuerzos, se los dedico.

Índice general

Índice de figuras

Índice de tablas

Tabla de Acrónimos

Acrónimo	Significado
GCN	Green Comunicación Networks
LPWA	Low Power Wide Area Network
IoT	Internet of Things (Internet de las cosas)
WSN	Nodos sensores de red
LAN	Red de area local
WAN	Red de area amplia
EH	Circuitos de recoleccion de energia
P-MFC	PLANT-MICROBIAL fuel cell
DPMS	Gestion dinamica de energia
IA	Inteligencia Artificial
DL	Deep Learning o Aprendizaje profundo
LTE	Long Term Evolution
LPWA	Low Power Wide Area Network
WOS	Weighted Order Statistic Algorithm
KDPMS	Kalman Dynamic Power Management Strategy
ANN	Algoritmo de Red Neuronal
MCU	Microcontrolador
K-nn	K-nearest neighbors o Vecino mas cercano a K
GD	Gradiente Descendiente
LM	Levenberg-Marquardt
NNF	Neural Net Fitting
TNB-IOT	Narrowband- Internet of Things
eMTC	enhanced machine-type communication
LoRa	Redes de Largo Alcance
LPWA	Low Power Wide Area Network
TEG	Generador Termoeléctrico
GD	Gradiente Descendiente

Capítulo 1

Introducción

1.1. Antecedentes

1.1.1. Nodos sensores de red

Los nodos sensores de red (WSN) juegan un rol muy importante en el internet de las cosas (IoT) ya que son un conjunto de sensores de bajo costo, y de bajo consumo energético, los cuales entran como una alternativa al crecimiento enorme de las redes convencionales. Los WSN son comúnmente utilizados para el estudio y recopilación de datos y pueden operar con dispositivos y estrategias especializadas con el fin de lograr bajos consumos energéticos. Lo anterior, posibilita que estrategias de cosecha de energía puedan ser utilizadas para alimentar a los WSNs y prolongar la vida de operación de los dispositivos sensores [1].

1.1.2. Aplicaciones de las WSNs

Se tienen antecedentes de trabajos con nodos de sensores de red en implementaciones variadas, desde la implementación de nodos de sensores de red para mejorar el sistema de localización con una mejor cobertura, estableciendo los saltos entre conexiones que hay entre los nodos y su central [2], hasta implementaciones para monitoreo y seguridad en los mismos nodos [3][4][5]. Otros trabajos buscan proveer una mejor cobertura e implementaciones de redes WSN mediante el uso de tecnologías de bajo costo [6]. Así mismo, se tienen antecedentes de trabajos que han implementado nodos de sensores de red para estimar las variaciones de la tasa de información comprometida en la

multiplexación por división de frecuencia ortogonal, típica en redes inalámbricas de área local [7], así como también, implementando herramientas de simulación para emular WSNs [8]. Otros tipos de aplicaciones de importancia para la sociedad basadas en WSNs pueden encontrarse en [9, 10].

1.1.3. Importancia de cuidar la energía/reservas de energía en nodos de sensores WSN

Es necesario tener un plan adecuado para la cosecha y/o consumo de energía en los nodos sensores; es decir, una mala estrategia en cuanto a la operación de los nodos conlleva a impactos negativos en la WSN. Por ejemplo, una WSN con una mala estrategia de operación y de consumo energético impactará en la operación del nodo y a tener un incremento en el mantenimiento de los nodos sensores haciendo frecuente el reemplazo de las baterías utilizadas. Lo anterior incrementaría el deshecho frecuente de pilas o baterías (induciendo a una alta contaminación ambiental por deshecho de baterías) y a un incremento en el costo de mantenimiento de la red WSN. Una técnica de administración energética que adapta los periodos de muestreo de los nodos sensores puede encontrarse en [11].

1.1.4. Estrategias de administración energética

Las estrategias de administración de energía, como su nombre lo dice, son métodos implementados en los dispositivos para mejorar la eficiencia del consumo y/o almacenamiento energético del mismo. Estas técnicas tienen como objetivo extender el tiempo de vida útil de los dispositivos, baterías, etc., además de dar un margen de operación/trabajo mayor al que un dispositivo sin administración estaría sujeto. Las técnicas de administración energética pueden variar desde el definir calendarizaciones de comunicación de los dispositivos (muestreo/trasnmisión de dato sensado), hasta proponer administraciones correctas en el uso de la batería (por ejemplo, poner los nodos en espera o en modo dormir, entre otros.).

La Fig. 1.1, presenta la calendarización de operación de un dispositivo WNS, la cual consta de diversos estados de operación. Por ejemplo, el nodo sensor puede iniciar en un estado

de dormido, el cual puede ser de tiempo fijo/uniforme o puede ser adaptativo (dynamic sleep state) acorde a algún comportamiento de algúna variable de interés; posteriormente, el nodo sensor puede entrar a un estado activo donde podrá adquirir/muestrear las variables de interés (p. ejemp., temperatura, humedad, CO2, entre otros) o implementar algún algoritmo de procesamiento. Una vez que este estado ha finalizado, se podrá transmitir los datos sensados, o la información procesada, para posteriormente entrar a un estado de dormir de nueva cuenta. Lo anterior, permitirá tener al nodo sensor en reposo llevando a cabo así el método de administración de energía.

De forma adicional, la estrategia de administración energética puede hacerse más eficiente a medida de que sea necesario. Por ejemplo, un nodo sensor puede tomar en cuenta el nivel de reserva de energía de la batería con la cual es alimentado, con la finalidad de reducir, o extender, los periodos de operación, buscando siempre prolongar la operación del nodo sensor. Por otra parte, si se tienen técnicas de cosecha de energía, el nodo sensor puede utilizar la información de la energía cosechada para estimar los estados de las baterías en el futuro, y con ello, definir esquemas de calendarización adaptativas para la operación del nodo sensor.

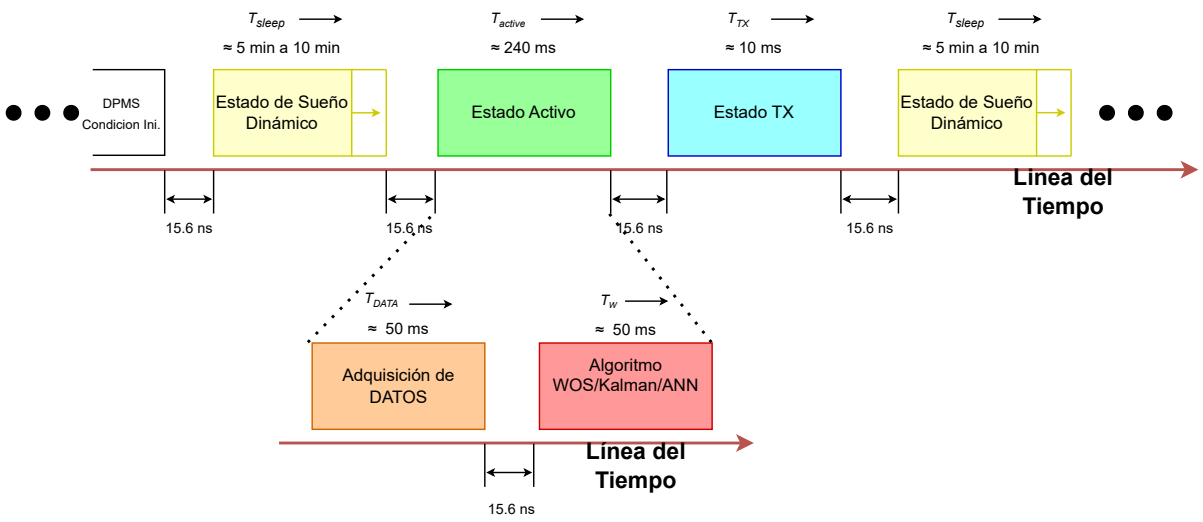

Figura 1.1: Ejemplo de un esquema de calendarización para implementar una estrategia de administración energética.

1.2. Planteamiento del problema

El uso masivo de nodos sensores de red en la era del IoT conlleva a que éstos tengan que ser energizados. En este sentido, se presenta un reto considerable debido a que dichos nodos deben estar en constante mantenimiento. Específicamente, los dispositivos de energía, como las baterías, deben ser reemplazadas de manera frecuente generando un problema de costo, y de forma adicional, un problema de contaminación ambiental. Por otra parte, nodos sensores que son energizados a través de técnicas de cosecha de energía, como aquellas basadas en dispositivos TEGs (term..), páneles fotovoltáicos, rafiofrecuencia, entre otros, necesitan de técnicas de procesamiento, las cuales permitan al nodo sensor operar de forma constante e ininterrumpida.

Dado el reto que implica la energización de nodos sensores de red, será necesario implementar técnicas de administración energética, las cuales cuadyuven a prolongar la vida útil de las baterías (o dispositivos de almacenamiento) y con ello cuidar las reservas de energía en los nodos sensores de red.

1.2.1. Justificación

Actualmente, en la Universidad Autónoma del Estado de Quintana Roo (UAEQROO), se tiene bajo desarrollo nodos sensores de red para su uso en aplicaciones IoT, los cuales son energizados mediante técnicas de cosecha de energía (p. ejemplo [12], basadas en plantas, páneles fotovoltáicos, entre otros). Sin embargo, a pesar de que se realiza un almacenamiento de la energía cosechada, ésta no se presenta de forma constante a lo largo del día. En este sentido será necesario implementar técnicas de administración energética con el objetivo de prolongar la vida útil de las reservas de energía.

1.2.2. Objetivos

Objetivo general

Desarrollar algoritmos de administración energética para mejorar el rendimiento energético en nodos sensores de red.

Objetivos específicos

- Objetivo 1. Investigar sobre modos de operación en microcontroladores de bajo consumo energético.
- Objetivo 2. Investigar sobre estrategias de cosecha y administración de energía para WSN
- Objetivo 3. Implementar técnicas de calendarización de funcionamiento en nodos sensores para extender el tiempo de funcionamiento del nodo sensor.
- Objetivo 4. Redacción del trabajo.

1.3. Metodología

Este trabajo busca proponer soluciones para mitigar el alto consumo energético en nodos sensores de red para implementar WSNs mediante la propuesta de técnicas algorítmicas y lograr una reducción del consumo energético (también puede interpretarse como minimizar la carga computacional o procesamiento) del nodo sensor.

La metodología por seguir en este trabajo tiene como base el conocimiento de forma directa de las reservas de energía existentes en el medio de almacenamiento del nodo. Por ejemplo, sabiendo la energía disponible en las pilas, o en dispositivos de almacenamiento de energía como capacitores, se ejecutarán algoritmos de procesamiento para realizar o establecer la programación de la calendarización a realizar en los eventos posteriores y con ello reajustar los estados de operación de los nodos sensores.

Así, cada vez que el nodo sensor se comunique con los puntos de acceso (dispositivos gateways) para el envío de información recabada, el paquete deberá incorporar información de sus reservas de energía como parte de la información enviada. En la nube o de forma local, los algoritmos de optimización tomarán en cuenta las variables del estado energético, los consumos de los elementos del nodo sensor y con ello reajustará el tiempo de calendarización para alargar la vida útil de los nodos sensores bajo operación.

Las actividades específicas propuestas para alcanzar los objetivos establecidos son los siguiente: Actividad 1: Para alcanzar el objetivo particular 1 se propone lo siguiente:

Realizar una revisión continua del estado del arte de diversos microcontroladores enfocados al bajo consumo energético. Así mismo, se revisará el estado del arte de diferentes técnicas de administración energética. Los microcontroladores estudiados se elegirán tomando en cuenta lo siguiente: disponibilidad en el mercado, documentación existente, herramientas de desarrollo disponibles, entradas, salidas y recursos internos, velocidad, memoria y consumo energético.

Actividad 2: Esta actividad se enfoca en alcanzar el objetivo 2.

El estudio de estrategias de administración de energía en redes de sensores se realizará de manera continua. Se enfatizará la búsqueda en técnicas basadas en inteligencia artificial. En esta actividad se considerará adquirir un conocimiento básico del comportamiento de carga y descarga de las pilas y/o supercapacitores, así como también, sobre el consumo energético del nodo sensor.

Actividad 3: Acciones a implementar para lograr el objetivo 3.

Estudiar las técnicas de calendarización para ahorro de energía en microcontroladores y redes de sensores (estados de manipulación de ahorro de energía en microcontroladores y sensores). Así también, se seleccionarán e implementarán algoritmos que no representen una carga computacional adicional y tengan un impacto negativo en los consumos energéticos del nodo. Por ejemplo, el algoritmo en el nodo sensor pudiera realizar trasmisión de datos sensados si detectara variaciones significativas en la variable a medir.

Actividad 4: Acciones a implementar para lograr el objetivo 4.
Redacción del trabajo.

Capítulo 2

Estudio del estado del arte.

En los últimos años, la visión del término Internet ha estado en constante expansión, sobre todo en aspectos de la vida cotidiana. Así, con el paso del tiempo, el término de internet se ha asociado con cosas y ahora se identifica como: IoT. Como el nombre lo indica, los IoT asocia a los objetos (cosas) a través de internet mediante nodos sensores inalámbricos (WSN, por sus siglas en inglés Wireless Sensor Networks), haciendo uso de la red de área local (LAN, por sus siglas en inglés Local Area Network), red de área amplia (WAN, por sus siglas en inglés Wide Area Network) ó conexión inalámbrica.

En la actualidad el uso masivo del Internet de las cosas, está permitiendo a la sociedad vivir en un mundo más autónomo, donde los datos recopilados son enviados al usuario, o proveedores de servicio, lo cual frecuentemente ocurre sin ninguna interacción del usuario. Esta autonomía y envío masivo de datos, por la gran cantidad de usuarios y dispositivos IoT conectados a la red, genera un tráfico en el internet volviéndolo inestable y lento. Además, la gran cantidad de dispositivos IoTs generan un exceso de pilas lo cual la convierte en una tecnología contaminante. En este sentido, existen algunas estrategias que se están implementando para volver a esta verde a esta tecnología.

El uso de energías verdes para energizar a estos dispositivos está ayudando a mitigar el uso excesivo de pilas, permitiendo un uso armonioso entre la nueva tecnología y el medio ambiente, dando la oportunidad para desarrollar nuevas técnicas de extracción y almacenaje de energía. Una de las técnicas de cosecha de energía es aquella basada en la implementación de celdas microbiológicas basadas en plantas P-MFC (Plant-Microbial

Fuel Cell) [12].

Una forma adicional de aumentar el rendimiento energético del nodo es mediante la definición de estrategias de cosecha de energía para energizar a los nodos sensores [12], además de, la definición e implementación de estrategias de administración energética.

Por ejemplo, la técnica de administración energética definida como estadísticas de orden ponderado (WOS, por sus siglas en inglés), propone una estrategia de gestión dinámica de la energía (DPMS por sus siglas en inglés), la cual ajusta dinámicamente el período de suspensión (ciclo de trabajo) del nodo sensor con base al comportamiento estadístico de los datos medidos [1].

Mencionando algunas de las técnicas de almacenaje de energía, podemos nombrar como sustituto de una batería convencional a un capacitor, antiguamente los capacitores se quedaron atrás en el avance tecnológico que tuvieron, permitiendo que otras tecnologías fueran priorizadas como almacenes de energía, dejando al capacitor para usos en circuitos de muy baja potencia, con el avance tecnológico y las nuevas tecnologías, el capacitor se ha vuelto un excelente sustituto en estas técnicas, ya que en un capacitor de la actualidad de 3000 Faradios, con un rango de voltaje de 2.7 Volts. almacena un promedio de 11 Kilo-Joules, esto le podría permitir a un dispositivo de análisis de bajo consumo trabajar por 12 horas continuas antes quedar completamente descargado [13]. Tomando esto en consideración se abre el camino a diversas técnicas de implementación de análisis con redes de sensores de bajo consumo energético.

Una de las técnicas utilizadas para el análisis de los nodos es el que utiliza los autores de [14] donde generan mediciones simuladas y de campo sobre nodos de sensores de agua por 3 meses para generar datos para comparaciones y estadísticas, para poder observar sus funcionamientos, este esquema funciona por un sistema de ciclos entre suspendido y análisis, para optimizar el consumo de energía del dispositivo, dando lugar a un ciclo de trabajo generado donde únicamente obtendrá datos de los análisis en determinadas horas.

Otra forma alternativa para mitigar su huella de carbón es disminuir el alto consumo energético de nodos para implementar WSNs mediante técnicas algorítmicas de “computo en la nube” para la reducción de carga computacional en el nodo sensor, es decir, los datos colectados por el dispositivo IoT se envían a la nube, donde se realizará el análisis de la información recabada, toma de decisiones y programación de eventos. Este tipo de procesamiento de datos impacta positivamente en la vida útil de las pilas y/o supercapacitores, disminuyendo así el cambio frecuente de dichos elementos.

Además, con el computo en la nube se busca que las WSN sean inteligentes permitiendo a los usuarios dar comandos básicos para que el sistema tome la mejor decisión de que es lo que debería hacer en base a la funcionalidad del objeto, es allí donde entra la Inteligencia Artificial la cual, enseñándole con ciertas condiciones de parámetros, análisis estadístico, y/o conjunto de datos, que previamente el usuario ha realizado de una toma de muestreo del comportamiento del sistema, en el cual se implementará, podrá decidir el comportamiento de dicho sistema o dispositivo, basado en los datos que obtuvo para su enseñanza, en las cuales no existe un único método, por lo que se debe implementar el aprendizaje al sistema de la AI con las técnicas que se consideren mas apropiadas, observando su comportamiento y en base a los resultados deseados se implementara la mas óptima, tal como se menciona en el libro de MathWorks sobre inteligencia Artificial "*No hay un mejor método o una talla única para todos. Encontrar el algoritmo correcto es en parte solo ensayo y error, incluso los científicos altamente experimentados no pueden saber si un algoritmo funcionará sin probarlo, pero la selección del algoritmo también depende del tamaño y tipo de datos con los que está trabajando, la información de la que desea obtener los datos y cómo se usarán esos conocimientos*" [15].

Avanzando con el sistema de aprendizaje, tomando en cuenta los múltiples métodos de programación que el usuario puede utilizar, se puede profundizar mas en el modelo de la arquitectura del sistema de la IA, dando lugar a un aprendizaje profundo (Deep Learning o DL por sus siglas en inglés) aquí la estructura de procesos dependerá completamente de la programación del usuario, ya que se asemejara a lo que vendría siendo una red neuronal para el aprendizaje y comportamiento del sistema. Una red neuronal

profunda combina múltiples capas de procesamientos no lineales, utilizando elementos simples que operan en paralelo e inspirados por el sistemas nerviosos biológicos. Consiste en una capa de entrada, varias capas ocultas y una capa de salida. Las capas están interconectadas a través de nodos, o neuronas, con cada capa oculta utilizando la salida de la capa anterior como entrada [16].

Se debe tener en cuenta para el aprendizaje de las IA independientemente del sistema que se elija para el aprendizaje, no sera únicamente un conjunto pequeño de datos, para in refinamiento mas fino sobre las decisiones que tomara por si misma, se necesitan cientos incluso miles de datos de muestra para el sistema, desde Megabytes, hasta Terabytes de datos, con el objetivo de darle un mayor panorama del objetivo que queremos obtener, por lo que la minería de datos (Data Minning o DT por sus siglas en ingles) se ha vuelto parte vital del sistema de la IA ya que por consiguiente los usuarios se han adaptado a la obtención masiva de datos para su funcionamiento y en conjunto con ello han ido perfeccionando el algoritmo que se maneja en el sistema, no hay que olvidar, que al final es un algoritmo el cual estamos optimizando para que pueda aprender y tomar decisiones por si mismo.

La complejidad de la programación de la IA a menudo se divide en dos partes: la complejidad de la formación y la complejidad de aplicar un algoritmo de entrenamiento. El entrenamiento generalmente no es rápido y no aplica en tiempo critico, por lo que puede tomar más tiempo, factor en el que influye todos los elementos obtenidos para la implementación, sin embargo, a menudo queremos una decisión sobre un punto de prueba rápidamente, y hay potencialmente un montón de puntos de prueba cuando un algoritmo está en uso, por lo que este necesita tener un bajo costo computacional [17].

Los dispositivos se estarán comunicando entre si cada determinado tiempo según indique la evaluación de los datos obtenidos a lo largo del análisis, la IA determinara a cada dispositivo, que pueden ir desde cientos hasta miles de nodos de sensores, en que momento sera necesario la comunicación entre ellos.

Por otro lado, para dar soporte a la gran densidad de dispositivos que se comunican entre

sí, es necesario usar una tecnología para ayudar a que el tráfico de datos sea eficiente, ya que las tecnologías tradicionales como LAN, WLAN y red celular no son suficientes para este tipo de casos. Por lo que varias tecnologías se han desarrollado para ayudar a resolver este problema. Algunas de las cuales utilizan la infraestructura ya existente de tercera generación como es Evolución a largo plazo (LTE, por sus siglas en inglés Long Term Evolution), mientras que otros dependen de tecnologías patentadas. Las tecnologías que utilizan la infraestructura LTE incluye red del internet de las cosas de banda estrecha (NB-IOT, por sus siglas en inglés Narrowband- Internet of Things) y comunicación mejorada de tipo máquina (eMTC, por sus siglas en inglés enhanced machine-type communication), mientras que las tecnologías que se basan en patentes se encuentran SigFox y las redes de largo alcance (LoRa, por sus siglas en ingles). Todas estas tecnologías están categorizadas bajo la etiqueta de redes de baja potencia y área amplia (LPWA, por sus siglas en ingles Low Power Wide Area Network).

Las redes SigFox y LoRa son conveniente para el IoT ya que se pueden mantener al usuario comunicado con su red privada, reduciendo el alto consumo que tienen los objetos en la actualidad, optimizando el funcionamiento de la red.

Por otro lado, la red IoT se considera una red de Internet de las cosas de banda estrecha o NB-IoT cuando se envían tasas pequeñas de datos en periodos de tiempos largos, esto ahorra energía, consumo, entre otros factores.

En la Tabla 2.1 se muestra las características importantes de las diferentes tecnologías de conectividad.

Tabla 2.1: Características de diferentes tecnologías de la comunicación.

Tecnology	Wireless Technologies (Short Range)(Operates in License Exempted Band)	LPWAN Technologíes (Low Range)	
		Non 3GPP Unlicensed	3GPP Licensed

Continued on next page

Tabla 2.1: Características de diferentes tecnologías de la comunicación. (Continued)

	ZigBee	Wi-Fi	BLE	Z-Wave	LoRa	SIGFOX	INGENU	Weightless-P	LTE-M	ECGSM-IoT	NBIoT
Range	75-100 m	70-250 meters (802.11n) (Outdoor)	100 m (class 1)	~100 m	2-5 km (URBAN), 15 km (suburban)	3-10 km (URBAN) 30-50 km (RURAL)	15 km (URBAN)	2 km (URBAN)	~11 kms	~15 kms	10-15 kms
Bandwidth	2 MHz	22 MHz	2.4 GHz ISM band Each channel has 1 MHz	-	500 KHz (channel band-width of 125 KHz)	100 Hz	1MHz	12.5 KHz	1.08 MHz(1.4 MHz carrier bandwidth)	200 KHz	180 KHz (200 KHz carrier bandwidth)
Frecuency Band	868 MHz, 915 MHz, 2.4 GHz	2.4 GHz/ 5 GHZ	2.4 GHz ISM band Each channel has 1 MHz	908.42 MHz	868 MHz and 915 MHz Sub 1 GHz	915-928 MHz Sub 1 GHz band	2.4 GHz	sub GHz ISM Band	Cellular Band	2.4 GHz	700, 800, 900 MHz
Data Rate	250 kbps	600 Mbps (802.11n)	1Mbps	40Kbs-100Kbps	50Kbps with FSK	Less than 100 bps	624Kbps-DL 156 Kbps UL	0.625 Kbps to 100 Kbps	300/375 Kbps variable	~350 bps to 700 Kbps	200 Kbps

Continued on next page

Tabla 2.1: Características de diferentes tecnologías de la comunicación. (Continued)

Latency	~15 ms	Less than 20 ms	~3 ms less than 10 ms	1s (Wake up)	1-10 ms	High, 1-30 ms	~10 s	-	10s-15 ms	10 segs	10 segs
Throughput	250 Kbps	802.11n 600 Mbps	0.27 Mbps	40 Kbps	~50Kbps	Ultra-Low ~100 bps	30 Mbps	-	1Mbps	10 Kbps	~150 Kbps
Modulation scheme	O-QPSK	QPSK	GFSK	GFSK	FSK techniques	GFSK(DL), DBPSK(UL)	Patient RPMA	GMSK, offset-QPSK	QPSK, QAM	GSM-based	BPSK,QPSK
Requirement	Require Zig-Bee kit to operate	An access point to connect	BLE station (avaia-ble in smart phone)	Continuous internet connection and Z-Wave appliance module to operate	Require gate-way for connectivity	SIGFOZ modem and SIGFOX network	Private Network	Require kit to and access point to operate	Uses Existing LTE network Software upgradation	Software upgradation	Software upgrada-tion and a sim to operate

Continued on next page

Tabla 2.1: Características de diferentes tecnologías de la comunicación. (Continued)

Batery lifeteime	for years	-	-	Claim 10 years life of a coined size battery	10 years	10 years if 1 message us sent 10 years if 6 messages is sent	10-20+ years (target to achieve by ingenu)	3-8 years	10 years	~10 years	10 years with a battery capacity of 5Wh
Cost	Low	Moderate	Less cost	Low cost	Low cost	Low cost	Low cost	Moderate	Low	Low	Low
Application	Smart home, heart-care, Smart lighthing, retail	Use to con-nect to network	Car access, smart watches, smart phones	Smart Home	Air pollution Monito-ring, Fire detection	Smart meter, pet trackung, smoke detector, agriculture	oil and gas field automation		Fleet tracking, Smart Home, tele-matics, Wearables	Water and Gas meter ap-plication, machinery control, smart grid	street lightning, pulse meter, agriculture, trach canes
Mobility	Yes, but low	Yes	Limited mobility	Yes	Yes	Yes	no	Yes	Yes	Nomadic	Nomadic

Continued on next page

Tabla 2.1: Características de diferentes tecnologías de la comunicación. (Continued)

Advantages	Hight Reliable and Salable	Easy to deploy and access	Provide more privacy Optimal for sending small chunks of data	Low cost, less vulnerable, signal remain strong up tp 100 feet claimed by z wave	highly immune to interference adaptive data rate, longer battery life	High Reliability, Device Complexy is low as, it can opt random frecuency to transmit the channel	High Reliability, extreme coverage, Transmit power control to extended battery lifetime, High Capacity Offer High Reliability, Supports hig number of devices	Provide support for multicasting positioning, Higher data Rates, Support VoLTE	Improved GSM/EDGE security, Low complexy device, all the key mobile equipment, chipset supports EC-GSM-IoT	USE PSM, edrx(almost upto 3 hrs), Obsolete the requeriment of Gateway Better range and coverage	
Limitations	Short range communication Security in vulnerable as keys are exchanges over air	Limited Range, vulnerable, Speed is much slower	Not very flexible, cavesdropping is possible, No accepted std. for generic gateway	Interference signal cant propagate over long distance due large wavelenght of signals, Four hops are essental for transmission of signal to device	Supports limited size data packets Longer latenci time, not acknowledges all packets	Low security, No Forward Error Corrections, BS could not support multiple sectors, Suffer from Interference	No analityc study avaiable to specific this	Scarcity of hardware and infrequient update specification	Coupling losses occur due to indoor environment, Repetitions slowdowns the transmission		a)No Handoff support, b)Interference immunity is low, c) Lacks in acknowledging
Standard body	ZigBee Alliance	IEEE	IEEE	Z wave	LoRa Alliance	SIGFOX	INGENU	WEIGHTLESS SIG	3GPP	3GPP	3GPP

Estas técnicas que se han implementado para el control de estos dispositivos, con el

avance de la tecnología, han implementado algoritmos de programación cada vez más novedosos, integrando técnicas como el aprendizaje automático e inteligencia artificial, sin embargo dichos algoritmos de igual manera han tenido un crecimiento en consumo exponencial por lo que requieren de aplicaciones de optimización de algoritmos con el objetivo de reducir los tiempos de ejecución que los trabajos que ira ejecutando el sistema por tareas, para así aumentar la calidad del producto final y tener una mejora en funcionalidad y eficiencia de los dispositivos que estaban conformados por la red de sensores.

La gran mayoría de los problemas de optimización con implicaciones en ciencias, ingeniería, economía y negocios son muy complejos y difíciles de resolver, bajo estas circunstancias los métodos de cómputo evolutivo se han vuelto una solución alternativa, los cuales son considerados herramientas genéricas de optimización que pueden resolver problemas muy complejos caracterizados por contar un espacio de búsqueda demasiado grande, dichos métodos permiten resolver problemas de una manera rápida y robusta, además, en comparación a otros algoritmos heurísticos, las técnicas de cómputo evolutivo son más simples de diseñar e implementar [18].

Partiendo de todo lo anterior, nace la necesidad de la implementación de técnicas para control, administración y optimización de la energía en los WSN, para ello nacen diferentes técnicas de extracción de la misma, con el objetivo de mantener el equilibrio entre energía verde e innovación, esto en conjunto con la búsqueda de diferentes formas de almacenamiento de la misma, estas diferentes técnicas de administración energética pueden variar según la situación a la cual los WSN estén expuestos. Un claro método alternativo seria el caso de la bioelectricidad generada por las plantas, la cual representa una solución adecuada para sistemas SRI-WSN, y anteriores, los estudios han analizado el uso potencial de plantas en su lugar de baterías para el funcionamiento sostenible de los auto-alimentados IoT-WSNs. Además, el desarrollo de técnicas de gestión de ahorro de energía son necesarias para mejorar el rendimiento energético de los nodos de sensores basados en EH para garantizar que haya suficiente energía almacenada para la adquisición y transmisión de datos [1].

Hoy en día, la sostenibilidad energética sigue siendo uno de los principales retos para operar estructuras de red basadas en IoT en el largo término. En el caso particular de la contaminación acústica urbana en las ciudades, los nodos WSN se implementan en entornos hostiles y debe trabajar el mayor tiempo posible sin interrupción. Para resolver el problema de la capacidad limitada de dispositivos de almacenamiento de energía de uso común (por ejemplo, baterías), se pueden aprovechar diferentes fuentes de energía renovable (solar, viento y vibraciones electromagnéticas y mecánicas).

Otra opción de fuente de energía, relativamente inexplorada en la literatura; es la pila de combustible microbiana vegetal (P-MFC), que genera bioelectricidad a través de la actividades de rizo-deposición y foto-sintética en la región radicular de una planta. Estudios recientes demuestran que los P-MFC puede generar suficiente energía para sostener el funcionamiento de largo alcance Dispositivos WSN basados en LoRa. Por lo tanto, la vida útil de WSN puede extenderse utilizando sistemas de recolección de energía (EH), logrando una operación sostenible de la red IoT. Sin embargo, aunque los sistemas EH han aumentado la vida útil de operación de los WSN, todavía hay oportunidades abiertas en la implementación de mecanismos de ahorro de energía frente a la generación de energía intermitente o para ampliar los recursos energéticos de los sistemas desarrollados. Algunas técnicas de ahorro de energía para aplicaciones de IoT se han abordado en distintos proyectos realizados: por ejemplo, varios algoritmos de muestreo adaptativo se han utilizado para ajustar el nodo del sensor y la frecuencia de muestreo según la energía disponible en el sistema [19].

Una vez que las problemáticas existentes en la administración energética de los nodos WSN han sido presentadas, en este trabajo se desarrollan tres técnicas de administración de energía como alternativas de uso en nodos sensores de red, éstas son: 1) Algoritmo de orden ponderado (WOS por sus siglas en ingles, Weighted Order Statistic Algorithm), 2) Estrategia de gestión dinámica de energía de Kalman y 3) Técnica de administración energética basadas en Redes Neuronales.

2.1. Algoritmo de WOS

El algoritmo estadístico de orden ponderado es un algoritmo que trabaja con ecuaciones estadísticas el cual recibe un conjunto de datos a analizar (en este trabajo, datos del análisis de la medición de la variable temperatura a utilizar en la fase de diseño y los datos del análisis del consumo energético del sistema para la fase de pruebas). Así mismo, es necesario calcular una distancia euclidiana de los datos recibidos con la finalidad de realizar una clasificación acorde a las clases que se tengan definidas en el algoritmo WOS. Este algoritmo trabaja con vectores de datos de tamaño M, y que es necesario calcular su mediana acorde a Ec. (2.1)

$$\zeta_i = mediana(W_i), \tag{2.1}$$

donde W_i representa el vector ya ordenado de mayor a menor, del cual se obtendrá la mediana; posterior a ello, se deberá obtener el valor absoluto de la diferencia de cada uno de los valores del vector menos la mediana resultante. Los valores resultantes deberán ser almacenados y sumados cuadráticamente como es expresado en Ec. (2.2) y Ec. (2.3).

$$\Delta_i = |(W_i - \zeta_i)| \tag{2.2}$$

$$\epsilon_i = \Sigma(\Delta_i)^2 \tag{2.3}$$

Como último paso del algoritmo WOS se deberá calcular el valor absoluto de la diferencia entre los datos a priori y la sumatoria obtenida en (2.3), esto se expresa en la Ec.(2.4).

$$\rho_i = |W_i - \epsilon_i| \tag{2.4}$$

El vector resultante ρ_i contendrá las distancias con las que el algoritmo tomará una decisión (cerrando como un esquema de optimización) con la finalidad de realizar una clasificación de tiempo de trabajo de los dispositivos.

2.2. Estrategia de gestión dinámica de energía de Kalman

La estrategia de gestión dinámica de energía de Kalman o algoritmo de Kalman (DPMS por sus siglas en ingles, Kalman Dynamic Power Management Strategy) estima el ruido que genera los datos entrantes de un dispositivo inalámbrico, posteriormente genera una clasificación de la variación del mismo, y con base a los resultados de este mismo, se puede determinar los ciclos de trabajo que tendrá los sistemas inalámbricos o dispositivos WSN ya que en muchos casos se utilizan nodos sensores de red para recolección de datos. El algoritmo DPMS basado en Kalman puede ser implementado directamente en el MCU del dispositivo, con el objetivo de administrar correctamente los tiempos de trabajo, optimizando la precisición de los resultados de clasificación.

Con esto en mente, el MCU le pueden ordenar con mayor eficacia a los dispositivos LoRa o dispositivos de radio, los cambios que tendrá entre sus fases de standby, transmisión y sleep, ahorrando la mayor cantidad de batería posible, y optimizando el tiempo de vida útil de la batería del dispositivo o nodo de sensor de red. Cabe señalar que el algoritmo de Kalman trabaja con datos matriciales, a diferencia del algoritmo WOS que trabaja con vectores. El algoritmo de Kalman es capaz de analizar matrices para obtener una mayor precisión en la variación de la clasificación de los tiempos de sueño (sleep) con la finalidad de que varias variables sensadas (p. ejemp., temperatura, humedad, CO2, entre otros.) puedan ser tomadas en cuenta por el algoritmo a la hora de realizar la clasificación. Es decir, este algoritmo puede considerar realizar la clasificación de sueño utilizando múltiples variables de entrada.

El algoritmo DPMS basado en Kalman considera la Ec. (2.5) y (2.6):

$$\widehat{x}_k^- = F\widehat{x}_{k-1} + q_{k-1} \tag{2.5}$$

$$y_k = H\widehat{x}_k + r_k \tag{2.6}$$

donde $\widehat{x}_k^-$ es el vector de estado en el paso k, y es el vector de salida (medición), F, H son el estado y las matrices de transferencia de observación, respectivamente, q_k es la dinámica de ruido de proceso, y rk es el ruido de medición producido por sensores y condiciones externas. Así, los datos de Kalman a priori se aprecian en la Ec. (2.7).

$$Eq_k = Er_k = 0 \tag{2.7}$$

Debido a que las matrices necesarias sean de covarianza, qk y rk quedan tal cual se aprecia en la Ec. (2.8):

$$E\{q_k q_k^T\} = Q_k,\ E\{r_k r_k^T\} = R_k \tag{2.8}$$

A partir de este punto, Kalman estima el estado del proceso en algún momento k y luego obtiene la retroalimentación a través de mediciones.

Las ecuaciones específicas para la actualización de estados se pueden apreciar en la Ec. (2.9) y la Eq. (2.10).

$$\widehat{x}_k^- = F\widehat{x}_{k-1} \tag{2.9}$$

$$P_k^- = FP_{k-1}F + Q_{k-1} \tag{2.10}$$

Además, la estrategia de Kalman utiliza una serie de ecuaciones, para modificar la ganancia de forma activa y actualiza los valores de estimación que utiliza para su predicción, las cuales se pueden observar en Ec. (2.11), Ec. (2.12) y Ec. (2.13).

$$G_k = P_k^- H^T [HP_k^- H_T + R_k]^{-1} \tag{2.11}$$

$$\widehat{x}_k = F\widehat{x}_{k-1}^- + G_k(yk - H_{\widehat{x}_k^-}) \tag{2.12}$$

$$P_k = P_k^- - P_k^- G_k H \tag{2.13}$$

Donde $= P_k^-$ es el error de estimación de la covarianza a priori, y G_k es la ganancia al minimizar el valor P_k posterior al cálculo del error de covarianza, cabe mencionar que la Ec. (2.11), Ec.(2.12) y Ec. (2.13) trabajan en un ciclo el cual optimiza conforme a la matriz entrante.

2.3. Algoritmo de red neuronal (ANN)

En el mundo, con la tecnología actual, es posible entrenar sistemas que, con el aprendizaje adecuado, puedan generar hipótesis con los datos entrantes, para posteriormente predecir los posibles resultados. La bondad que ofrece el sistema de una red neuronal es que con un costo computacional considerado, puede dar resultados más precisos con base a los datos entrantes en el tiempo; para ello, se debe tener una base de guía o entrenamiento.

Existe un gran número de algoritmos de aprendizaje para una red neuronal, los cuales se pueden clasificar entre lineales o no lineales. Un sistema lineal es conocido por ser un sistema supervisado, es decir, se conoce tanto sus datos de entrada, y es preciso en la relación con sus datos de salida en el aprendizaje, por lo que siempre va seguir el camino indicado, los sistemas no lineales a diferencia, son conocidos como sistemas no supervisados, ya que pueden conocer los parámetros a los cuales esta asociado, pero no conocen los posibles resultados, por lo cual, el sistema tiene que realizar una estimación al punto ideal.

Las aplicaciones de las redes neuronales empezaron desde hace mas de 50 años, con la motivación para los científicos de usar las maquinas de manera eficiente, delegándole tareas que usualmente los humanos tienen que completar y resolver, utilizando reglas y procedimientos que usualmente los humanos no aplicarían.

Actualmente los conceptos de aprendizaje a través de una red neuronal se han aplicado en diferentes campos, el mas utilizado es el de machine learning donde aplica una serie de algoritmos iterativos para predecir un posible resultado a partir de un valor entrante, esto con el objetivo de ahorrar complejidad en el algoritmo bajo desarrollo, tiempo de trabajo y optimización de resultados.

Las redes neuronales trabajan utilizando neuronas, donde, la neurona mas simple recibe durante su entrenamiento uno o mas datos (p. ejemp., datos de variables de entrada y datos de salida del sistema), y el resultado obtenido a partir del mismo, para que al

recibir datos de prueba, éste entregue una predicción/estimación de resultados.

Estos modelos usualmente son conocidos como nodos. Estos nodos reciben entradas de datos con un peso w asociado, que se puede modificar para modelar el aprendizaje sináptico. La unidad calcula la función f de la suma ponderada de sus entradas, tal como se aprecia en la Ec.(2.14) :

$$y_i = f(\sum_j w_{ij} y_j) \tag{2.14}$$

La sumatoria que se aprecia en la Eq. (2.14) se le conoce usualmente como net input a la unidad i, usualmente se escribe como net_i. En la misma ecuación la variable w_{ij} hace referencia al peso de la unidad j a la unidad i, y la función f es la función de identidad, donde sus salidas, dependiendo de sus entradas pueden ser simples, hasta datos matemáticos enviados por paquetes a través de la red. (ver Fig. 2.1).

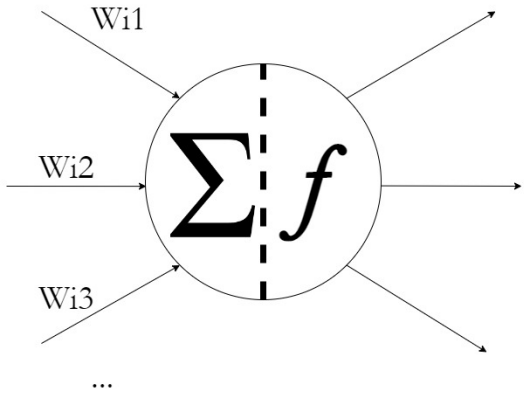

Figura 2.1: Ilustración de funcionamiento de una neurona.

Debido a que la red neuronal es un predictor, puede tener un cierto margen de error en la precisión que tendrá. Para que las redes neuronales sean buenas predictoras, se elaboró una función la cual se define como función de perdida, o de error (comúnmente definida con la letra E), a partir de los parámetros del modelo. Una de las opciones mas populares es la función “sum squared-error” (o suma de los errores cuadráticos) la cual

se puede apreciar en la Eq. (2.15):

$$E = \frac{1}{2}\sum_{p}(t_p - y_p)^2 \tag{2.15}$$

Eq. (2.15) es la suma sobre todos los puntos i en nuestro conjunto de datos de la diferencia al cuadrado entre el valor objetivo y la predicción del modelo y_i, calculada a partir del valor de entrada x_i. Para un modelo lineal, el “num scuared-error” es una función cuadrática de los parámetros del modelo. La función de pérdida E nos proporciona una medida objetiva del error predictivo para una elección específica de los parámetros del modelo. Por lo tanto, podemos reafirmar nuestro objetivo de encontrar el mejor modelo (lineal) como encontrar los valores para los parámetros del modelo que minimizan E.

En los modelos lineales, la regresión lineal les proporciona una forma directa de calcular los parámetros óptimos del modelo, pero hay que tener en cuenta que es únicamente para modelos lineales. En caso de que sea un modelo no lineal se puede utilizar el método de gradiente descendente.

A partir de este punto, el método utilizado para calcular el peso variará dependiendo del tipo de red neuronal que se tenga, puede ser un método simple para una red de una entrada y una salida, pero para redes con n entradas con m salidas, el método varía. Así mismo, puede variar de 1 hasta n capas, y las estructuras y resultados variarían según el método. En las redes neuronales no existe un método perfecto, se selecciona a partir de prueba y error el mejor método posible para la situación en la que se encuentre; tal vez se tenga una red de 3 entradas con 2 salidas y una capa oculta, el método de gradiente daría un error de 1 % pero con el método de backpropagation es posible alcanzar un 0.8 %. Estas diferencias pueden parecer no significativas, pero para redes neuronales que analizan millones de datos para su entrenamiento, esa diferencia puede significar alcanzar mejores resultados en cuanto a precisión.

2.3.1. Clasificador de vecino más cercano a fit k

El método de algoritmo de vecinos mas cercanos a K (Abreviado K-nn por sus siglas en ingles “K-nearest neighbors”) es un algoritmo diseñado para machine learning basado en instancias. K-nn es de tipo supervisado, lo que significa que requiere de datos para realizar el proceso de aprendizaje y prueba del mismo. Puede ser usado para evaluar nuevas muestras de los datos relacionados, o para realizar predicciones de los datos entrantes, similar a las funciones que se tienen en los aprendizajes de regresión lineal.

Como su nombre lo dice, el algoritmo K-nn trabaja buscando similitudes con base a las cercanías que tuvo durante su aprendizaje; es decir, que asociará las muestras entrantes en relación a la clasificación de orden de las muestras que obtuvo durante su entrenamiento. Debido a esta característica, con forme el algoritmo trabaja con mayor cantidad de datos, es capaz de mejorar su propio aprendizaje, realizando relaciones con mayor precisión.

A diferencia de su algoritmo hermano “K-means” el cual es un algoritmo no supervisado, que trabaja con grupos que va creando en su aprendizaje, el algoritmo "K-nn"va a trabajar únicamente con los puntos o "vecinos"que el usuario define al inicio del entrenamiento con los datos de muestras entrantes.

Gracias a su método de aprendizaje, el cual no se basa explícitamente en un modelo, sino en la memorización de los datos obtenidos con las muestras previamente como base para su conocimiento, el algoritmo K-nn se vuelve una herramienta muy práctica y la mejor para el uso de detección de anomalías y para sistemas de recomendaciones. Así mismo, gracias a sus características, se convierte en un algoritmo de fácil implementación y de un fácil y rápido entendimiento para sus aplicaciones; pero, como el mismo algoritmo no usa los sistemas (formas de entrenamiento) que aplican para los aprendizajes como el de regresión lineal", sino que considera directamente el dataset para su entrenamiento, el costo computacional se eleva a diferencia de sus semejantes. Sin embargo, aumenta su precisión (mayor complejidad computacional, pero mayor precisión).

El principal problema del algoritmo k-NN es encontrar el valor de k con el que obten-

gamos un mayor rendimiento al clasificar. Generalmente se utiliza una técnica conocida como cross validation. Esta técnica consiste en dividir el conjunto de entrenamiento (train) en distintas partes. Por ejemplo en 5 partes, y con éstas, clasificar a cada una de ellas mediante k-NN con los valores de k que se requieran y utilizando las otras 4 partes como conjunto de train. De este modo tendremos el rendimiento de k-NN con todos los valores de k para cada una de las 5 particiones que hemos hecho del conjunto de train original. Haciendo la media de esos rendimientos obtendríamos el valor de k con el que mejor rendimiento se obtiene y lo utilizaríamos para clasificar cualquier nuevo ejemplo. Sin embargo, existe otra técnica que no clasifica con el mismo valor de k cada nuevo ejemplo, sino que la k con la que clasificaremos a un nuevo ejemplo vendrá determinada en función de distintos parámetros como por ejemplo cuáles son sus vecinos más cercanos [20].

2.3.2. Retropropagación de Levenberg-Marquardt

Las ANN utilizan los algoritmos de retropropagacion para realizar entrenamientos en paralelo y mejorar la eficiencia de una red de multicapas. Las técnicas de retropropagacion son las más populares, fáciles de aprender y muy efectivas de utilizar en ANN de múltiples capas.

Una retropropagación es una técnica de aprendizaje supervisado que se basa en el método de gradiente Descendiente (GD) que busca minimizar el error de la red al bajar el gradiente de la curva de error.

Una técnica ampliamente utilizada de aprendizaje para redes neuronales es la retropropagación de Levenberg-Marquardt (Levenberg-Marquardt backpropagation), la cual es una técnica que actualiza los valores de peso y sesgo de la red neuronal de acuerdo con la optimización requerida. Al igual que los métodos cuasi-Newton, el algoritmo de Levenberg-Marquardt fue diseñado para acercarse a la velocidad de entrenamiento de segundo orden sin tener que calcular la matriz de Hesse o dicho de otra manera, que puede igualar la velocidad de entrenamiento unicamente usando el vector gradiente, si-

milar a la velocidad de una red entrenada utilizando la gradiente y la matriz de Hesse. Cuando la función de rendimiento tiene la forma de una suma de cuadrados (como es típico en el entrenamiento de redes feedforward), entonces la matriz de Hesse se puede aproximar como se aprecia en la Ec. (2.16).

$$H = J^T * J \tag{2.16}$$

Así mismo, es posible calcular la gradiente como se aprecia en la eq.(2.17) .

$$g = J^T * e \tag{2.17}$$

Donde J es la matriz jacobiana que contiene las primeras derivadas de los errores de red con respecto a los pesos y sesgos, y e es un vector de errores de red. La matriz jacobiana se puede calcular a través de una técnica estándar de retropropagación que es mucho menos compleja que calcular la matriz de Hesse [21].

El algoritmo de Levenberg-Marquardt utiliza esta aproximación a la matriz de Hesse como se aprecia en la eq. (2.18) de forma similar a la de Newton.

$$X_{k+1} = X_k - [J^T J + \mu I]^{-1} J^T e \tag{2.18}$$

Cuando el μ escalar es cero, esto es solo el método de Newton, utilizando la matriz hessiana aproximada. Cuando μ es grande, esto se convierte en gradiente descendiente con un tamaño de paso pequeño. El método de Newton es más rápido y más preciso cerca de un mínimo de error, por lo que el objetivo es cambiar hacia el método de Newton lo más rápido posible. Por lo tanto, μ disminuye después de cada paso exitoso (reducción en la función de error). De esta manera, la función de que calcula el error siempre se reduce en cada iteración del algoritmo [21].

Capítulo 3

Etapa de diseño

Para la etapa de diseño de los algoritmos de estrategias de administración de energía, tres algoritmos fueron considerados: 1) Algoritmo WOS, 2) Algoritmo de filtrado de Kalman, y 3) Estrategia basada en redes neuronales. Ésta última se diseñó utilizando 2 métodos de aprendizaje (K-NN y LM-Backpropagation).

Como actualización de los algoritmos de WOS y Kalman previamente diseñados, éstos fueron modificaron para considerar además de los valores entrantes como temperatura, humedad, ruido, etc. consideren el porcentaje de carga de la batería para realizar la clasificación.

Matlab versión “2018a” ha sido utilizado para realizar la implementación de los algoritmos WOS, filtrado de Kalman y realizar el aprendizaje de la red neuronal.

3.1. Diseño del algoritmo DPMS basado en WOS

Para el diseño e implementación del algoritmos WOS se propuso el diagrama de flujo que se aprecia en la Fig. 3.1. Sin embargo, para poder realizar la descripción del funcionamiento del algoritmo de forma detallada, éste se dividió en secciones, las cuales serán descritas con la ayuda de la Figura 3.2 a la Figura 3.7.

Ahora podemos iniciar explicando el diagrama de flujo utilizando la sección que se aprecia en la Fig.3.2, la cual es la encargada de configurar o establecer las clases que se tomarán en cuenta para realizar la clasificación a través de WOS; es decir, los ciclos de sleep o sueño del microcontrolador o dispositivo a implementar el DPMS mediante WOS.

Así mismo, en esta sección se recibe la configuración del vector de datos W a analizar

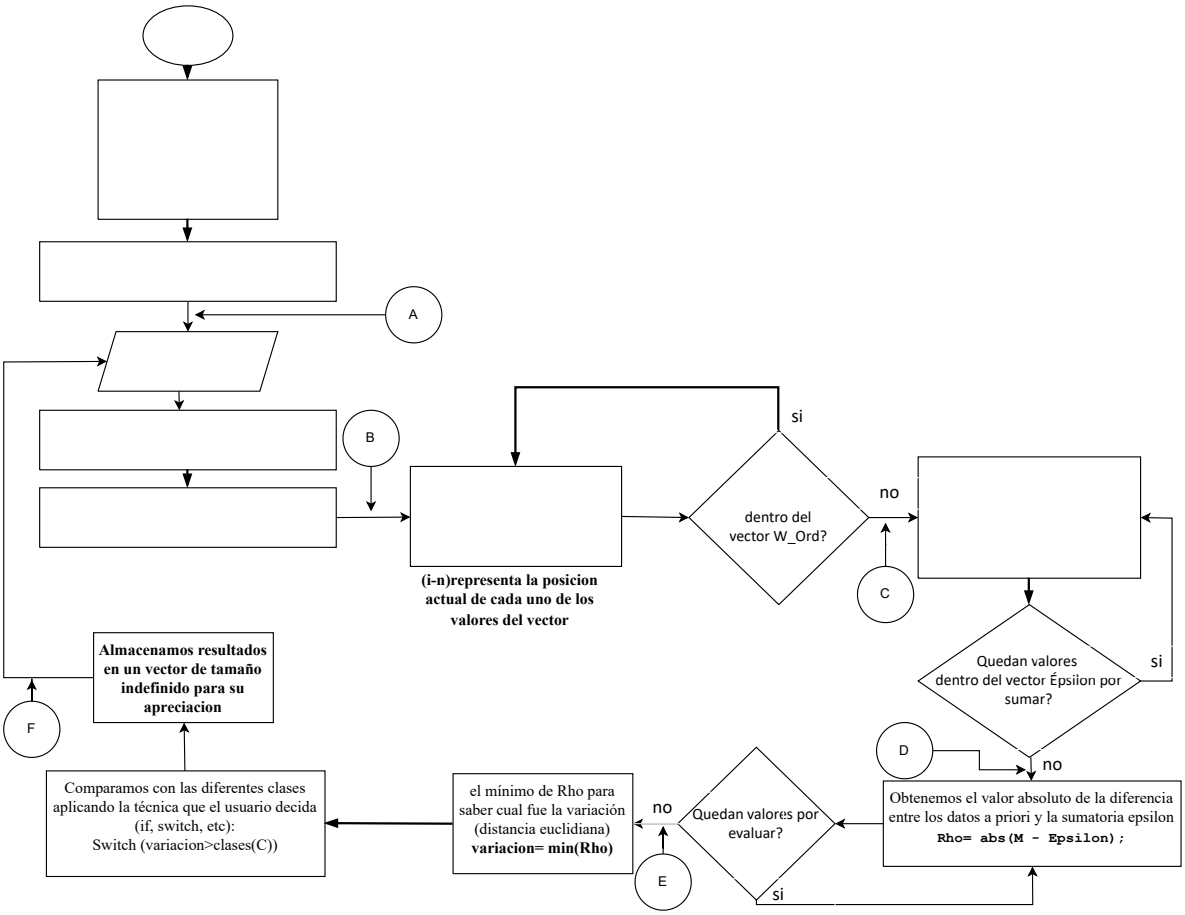

Figura 3.1: Diagrama de flujo de algoritmo WOS implementado.

y clasificar (por ejemplo, datos de temperatura, humedad relativa, ruido, entre otros.). Se tiene para ser configurado con valores de 5 o 10.

Posterior a ésto, como se puede observar en la Fig. 3.3 se recibe el vector de datos de entrada a analizar. Los datos se almacena en un vector designado **“W”**; sin embargo, el nombre del vector puede ser definido como el usuario desee. Posterior a ello, se ordena de mayor a menor y se almacena el resultante en un nuevo vector, en este caso llamado **“W_ord"** acorde a lo explicado en la sección 2.3.

Una vez que se tiene el vector ordenado, se calcula la mediana del vector **“W_ord"** resultante. La mediana es almacenada en la variable definida con el nombre **“Zetha”**. En cuanto a la implementación de este proceso mediante Matlab, ésto es logrado utilizando el comando “median(vector a obtener la mediana)”.

Se comenta que los nombres utilizados en las variables hasta el momento fueron elegidos con la finalidad de mantener una mayor relación posible con los nombres simbólicos

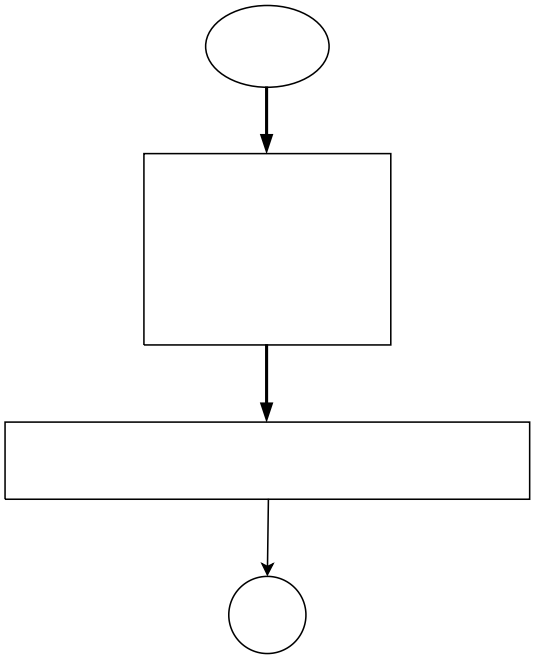

Figura 3.2: Sección 1 del diagrama WOS donde se define los valores fijos.

que se expresan en la Eq. (2.1).

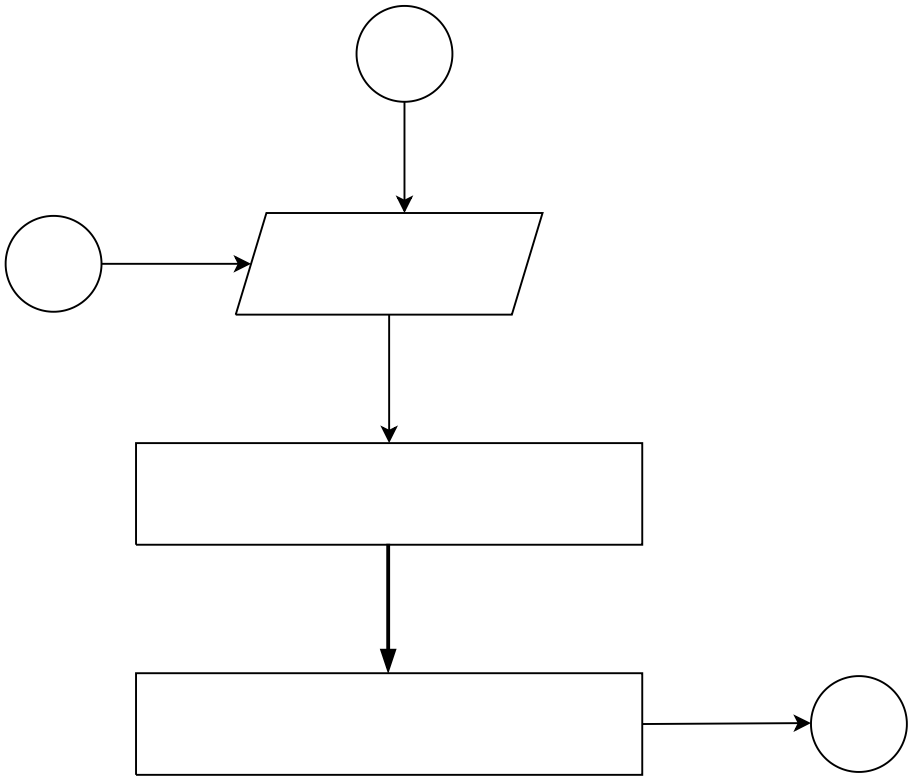

Figura 3.3: Sección 2 del diagrama WOS donde se cumple la Ec. (2.1).

Posterior al cálculo de la mediana, se procede a la realización de Eq.(2.2) como se aprecia en la Fig. 3.4. En la sección 2.2 fue explicado que se debe obtener el valor absoluto de la diferencia de cada uno de los valores del vector menos la mediana resultante (la cual almacenamos en la variable “Zetha”). Lo anterior se implementa mediante un ciclo encargado de recorrer las posiciones del vector ordenado **“W_ord"** y restando el valor de la mediana definida previamente como **“Zetha”**. El resultado es almacenado en un nuevo vector definido como **“Delta”** acorde a lo definido en la Eq. (2.2). Finalmente, **“Delta”** es actualizado para contener solamente los valores absolutos; en Matlab se traduce a la ejecución de un comando definido como “Delta = abs(Delta)” .

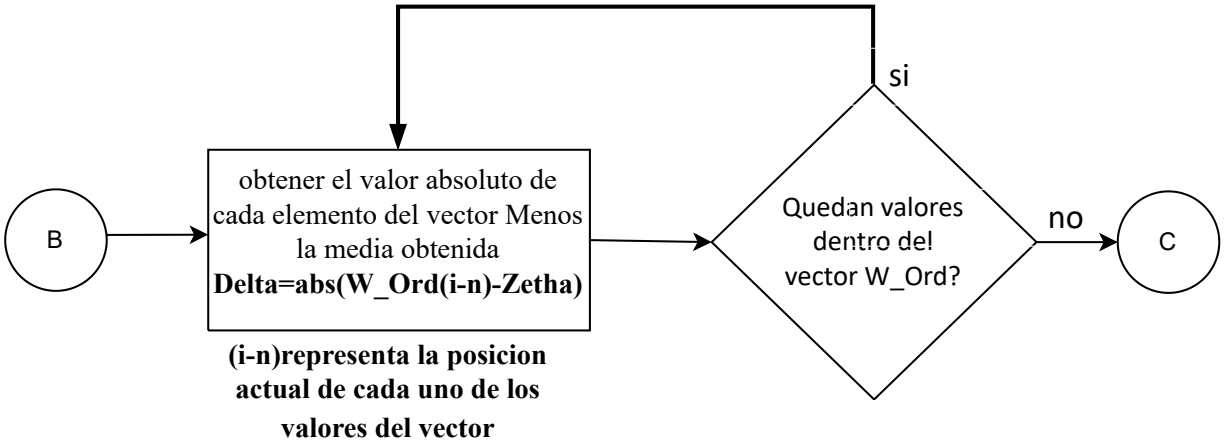

Figura 3.4: Sección 3 del diagrama WOS donde se cumple la Ec. (2.2) .

Con referencia a implementación de la Eq.(2.3) se aplica la misma lógica que fue considerada para la Eq. (2.2), ya que ambas deben recorrer su respectivo vector; sin embargo, ahora en esta ecuación es necesario recorrer el vector “Delta” con la finalidad de poder realizar una sumatoria de cuadrados de cada uno de sus valores almacenados. El resultado de la sumatoria es almacenado en la variable “Epsilon”. Como puede observarse, los nombres elegidos para alojar los resultados computados están asociados a las etiquetas, o nombres, definidos en las diversas ecuaciones.

En la Fig. 3.5 se presenta el procedimiento asociado a la implementación de la Eq. (2.4) y que está relacionado al cálculo del valor absoluto de la diferencia entre los datos a priori almacenados en el vector “M” y la sumatoria resultante alojada en la variable “Epsilon”. El resultado de este cálculo es almacenado en la variable “Rho”.

Hasta ahora, queda el obtener la distancia euclidiana del vector “Rho”. En este punto

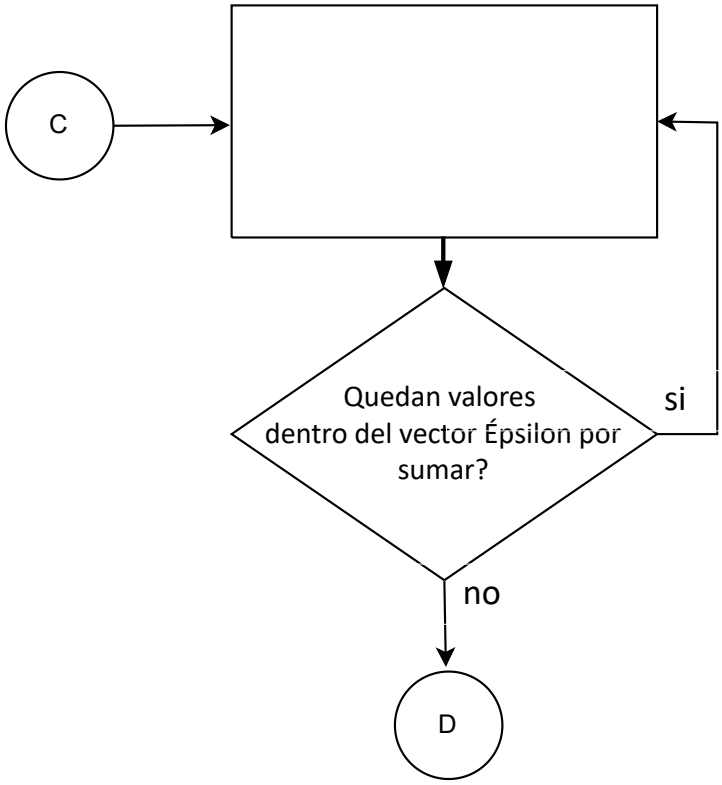

Figura 3.5: Sección 4 del diagrama WOS donde se cumple la Ec. (2.3).

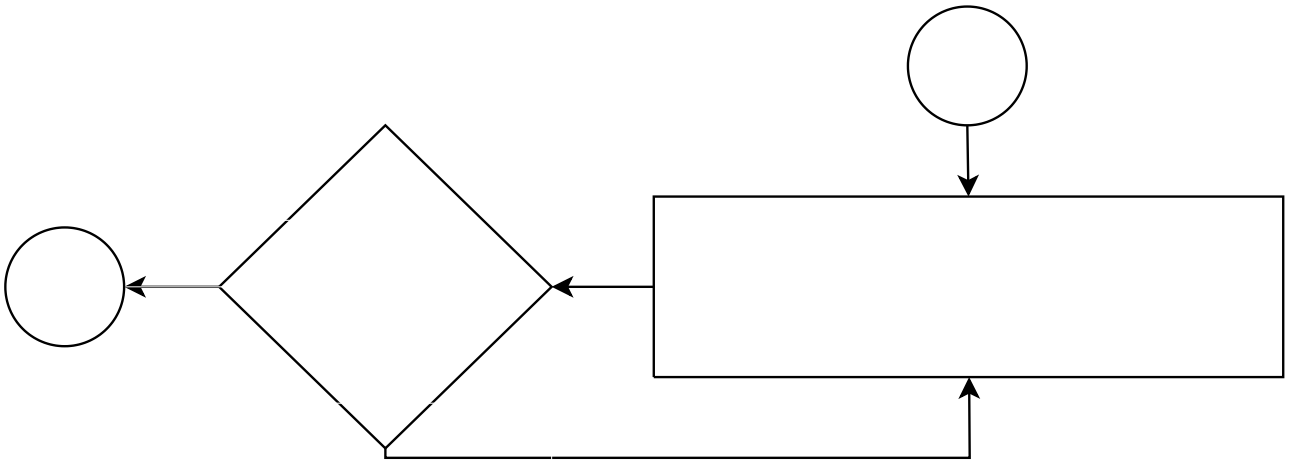

Figura 3.6: Sección 5 del diagrama WOS donde se cumple la Ec. (2.4).

se debe ser precavido, ya que el resultado dependerá completamente del criterio considerado por el usuario, o mejor dicho, la configuración del dispositivo. El paso siguiente es presentado en la Fig. 3.7, el cual utiliza una lógica para obtener los valores mínimos. Para ello es necesario considerar el vector del paso previo y compararlo con las clases,

con la finalidad de obtener la posición del valor mínimo del vector “Rho”.

Para ello se utiliza el comando “min (vector a evaluar)”, el cual obtiene el valor mínimo en un vector, posterior a ello es necesario utilizar el comando de comparación switch con la finalidad de evaluar y comparar cada uno de ellos con los datos a priori. Así, el que esté más cercano al valor mínimo almacenado en “Rho” sera la posición que obtendrá en la clasificación. Este resultado puede ser almacenado en un vector para poder ser graficado y visualizado. Como ejemplo de este método de programación se puede analizar el **Anexo A** de este documento el cual es un ejemplo de código aplicado en Matlab siguiendo las instrucciones del diagrama de flujo que se observa en la Fig. 3.1.

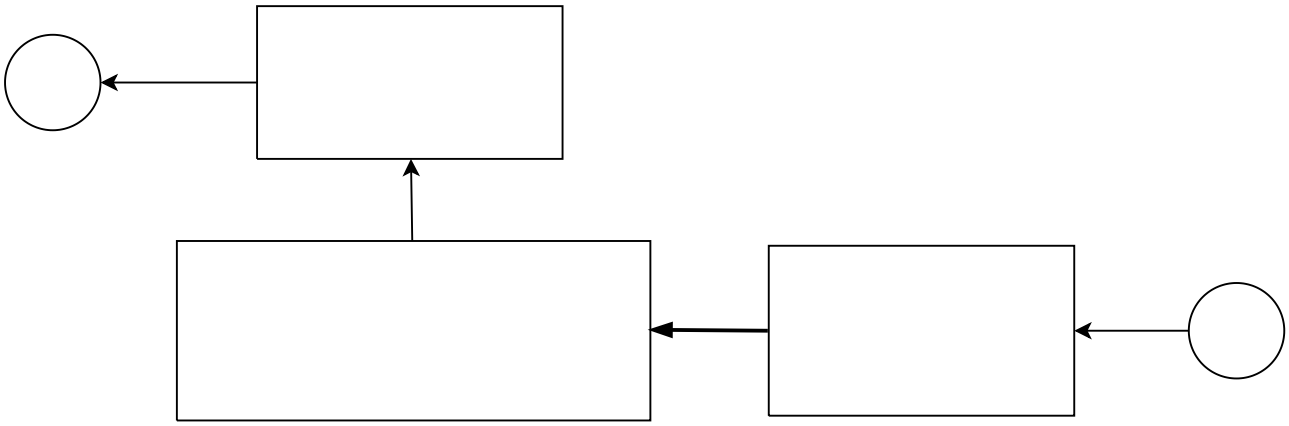

Figura 3.7: Sección 6 donde podemos observar un método de como llevar a cabo la realización de la clasificación.

3.2. Diseño del algoritmo DPMS basado en filtrado de Kalman

Para el diseño del algoritmo se ha propuesto el diagrama de flujo que se presenta en la Fig. (3.8), el cual ha sido dividido en secciones para una mejor explicación.

La sección 1 del algoritmo se aprecia en la Fig. 3.9, el cual representa el inicio del algoritmo. Como puede observarse, en esta sección también se definen los valores de entrada.

Posterior a ello, se continúa con la sección 2 que se presenta en la Fig. 3.10. Esta sección recibe los datos a analizar y obtiene las métricas necesarias para poder operar.

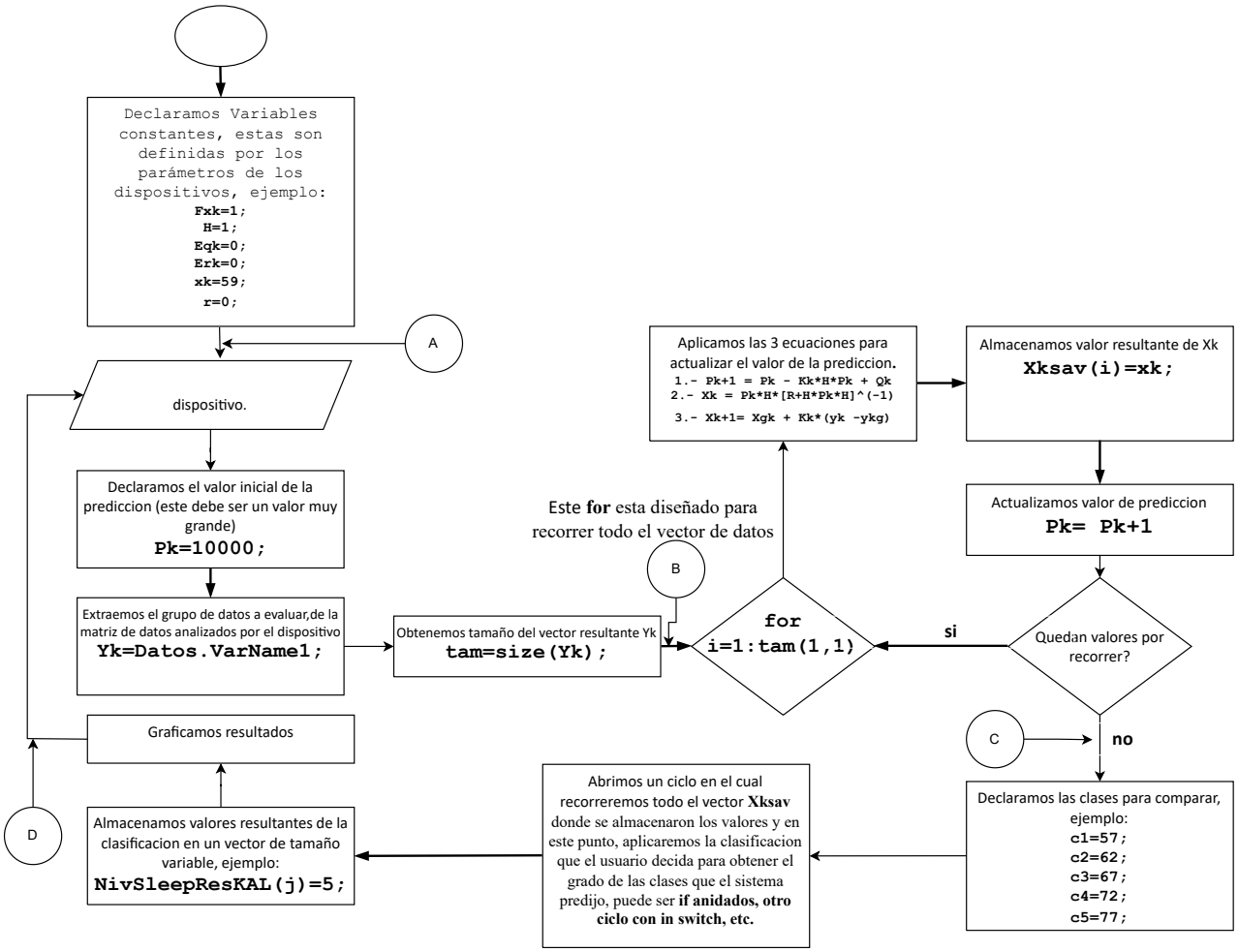

Figura 3.8: Diagrama de flujo de algoritmo Kalman implementado.

Cabe mencionar que los nombres seleccionados para las variables y las constantes fueron pensadas con base a las variables que se utilizan en la sección 2.3 de este documento. En la sección 2 se define en la variable “tam” la cual es la encargada de almacenar el tamaño total del vector o matriz entrante.

La sección 3 es la encargada de realizar el procedimiento mas importante del algoritmo. En esta sección se implementan las Eqs. (2.11), (2.12) y (2.13) (en ese mismo orden). Así, este ciclo del algoritmo utiliza las 3 ecuaciones para generar, o realizar una predicción, del siguiente valor entrante tomando en cuenta el valor actual comparándolo con su predicción. Posterior a ello, se realiza un reajuste de sus valores para generar una nueva predicción del siguiente valor a ingresar. El resultado de esta predicción se almacena en el vector “Xksav”, el cual es de tamaño dinámico, para posteriormente actualizar

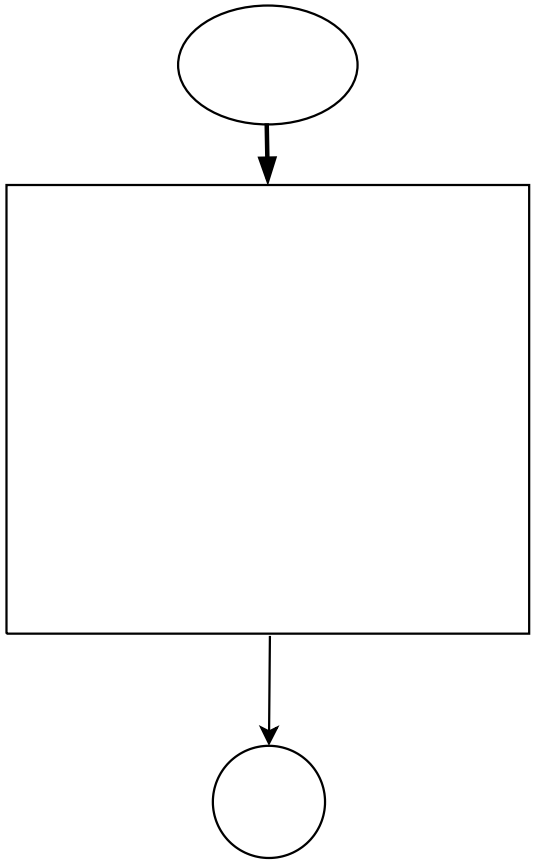

Figura 3.9: Sección 1 del diagrama de Kalman donde se define los valores constantes.

el valor de la predicción del siguiente numero (variable “Pk”) y repitiendo este mismo ciclo. Recorriendo todos los datos ingresados, se podrá ir obteniendo las predicciones de todos los valores de entrada. Todo este procedimiento se puede apreciar en la Fig. 3.11.

La sección 4 definida para este algoritmo puede apreciarse en la Fig. 3.12. En esta sección se definen las clases para evaluar el estado errático de los valores analizados. Estas clases como se mencionó en el algoritmo de WOS, se definen completamente por el programador, de allí se realiza una comparación de cercanía de los datos, utilizando comparadores de ciclo FOR, SWITCH, IF, etc.

El resultado generados serán los que el nodo sensor recibirá para saber cuando debería enviar su siguiente dato de análisis. Los resultados de las clasificaciones (o clases como se define en algunos algoritmos) obtenidas se almacenan en un vector de tamaño dinámico llamado “NivSleepResKAL”, con la cual se puede generar, u obtener, una gráfica con el objetivo de poder apreciar la predicción del comportamiento errático de los datos tomando en cuenta las clasificaciones definidas para el análisis.

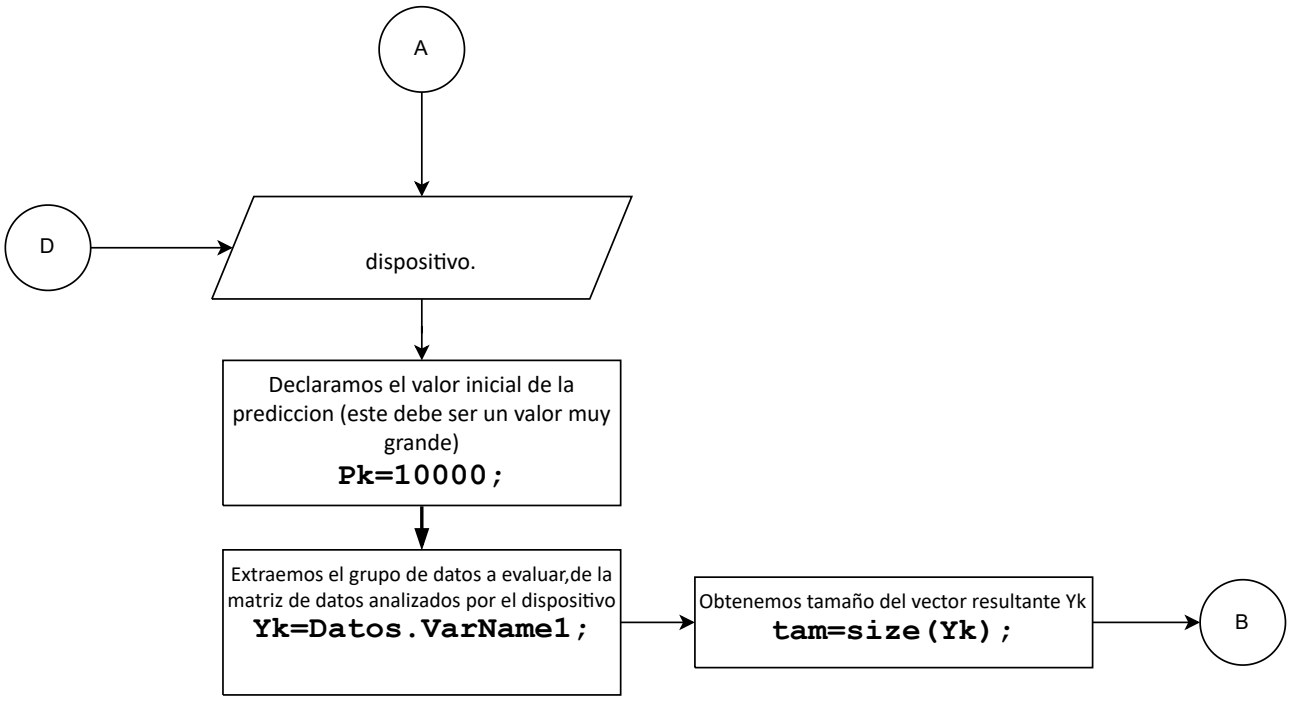

Figura 3.10: Sección 2 del diagrama de Kalman para la implementación del DPMS

Al igual que con el Algoritmo WOS, el algoritmo DPMS basado en Kalman fue diseñando en la herramienta de cómputo científico Matlab. El código propuesto tiene como base los cálculos matemáticos de algoritmo de Kalman explicados en la Sección 2.3. Los datos utilizados para probar la funcionalidad de dicho algoritmo son los mismos datos de medición de temperatura utilizados en la elaboración del primer algoritmo (WOS). Adicionalmente, el algoritmo de Kalman propuesto toma en cuenta el porcentaje de carga de la batería (considerando 0 % al 100 % de carga). El código resultante se puede encontrar en el **Anexo B** al final de este documento.

3.3. Diseño del algoritmo DPMS basado en redes neuronales

Para el diseño del algoritmo de la red neuronal se utilizó de nueva cuenta la herramienta Matlab (versión 2018a), la cual cuenta con múltiples herramientas que permiten

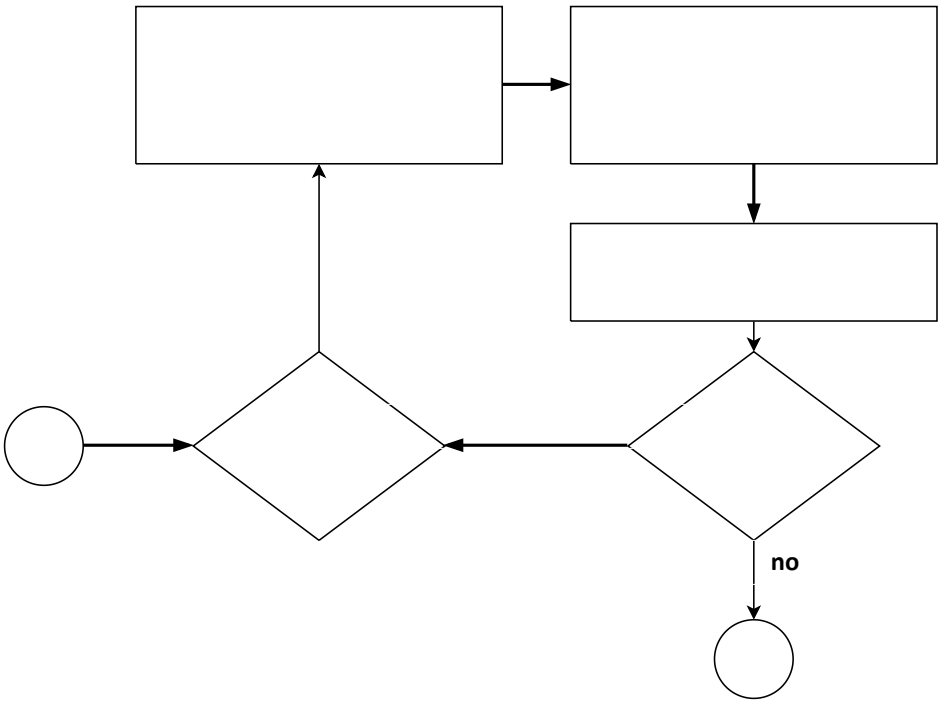

Figura 3.11: Sección 3 del diagrama de Kalman.

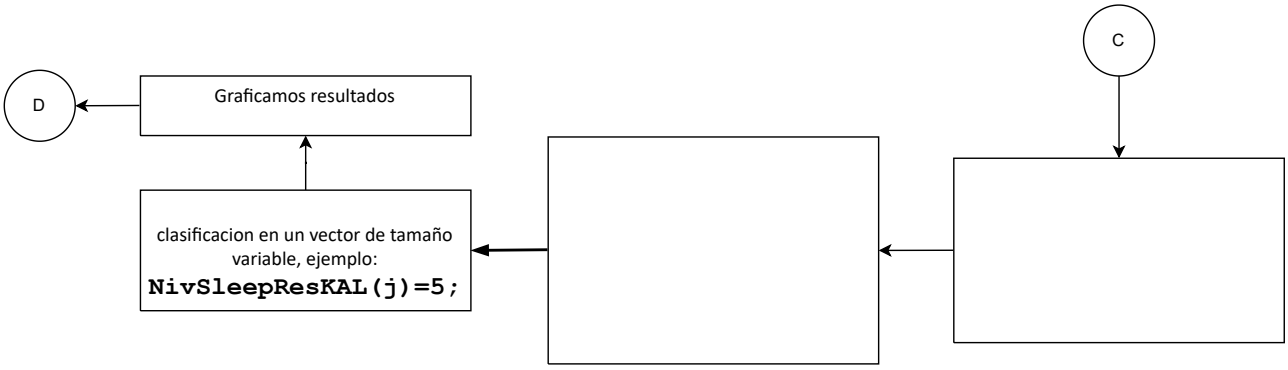

Figura 3.12: Sección 4 del diagrama de Kalman.

al usuario diseñar redes neuronales. Matlab también permite tener acceso al código para modificarlas y crear redes mas complejas. Una de estas herramientas disponibles es Neural Net Fitting (NNF) (ver Fig.3.13).

La aplicación de redes neuronales NFF permite introducir datos de entrada y salida a relacionar en la red neuronal. Se debe tomaren cuenta que ambos vectores de datos deben tener las mismas dimensiones. NFF permite elegir el porcentaje de datos que utilizará del total de datos para las fases de training, validation y test de la red neuronal. También, permite elegir el numero de neuronas que tendrá la capa de entrada donde se realizará

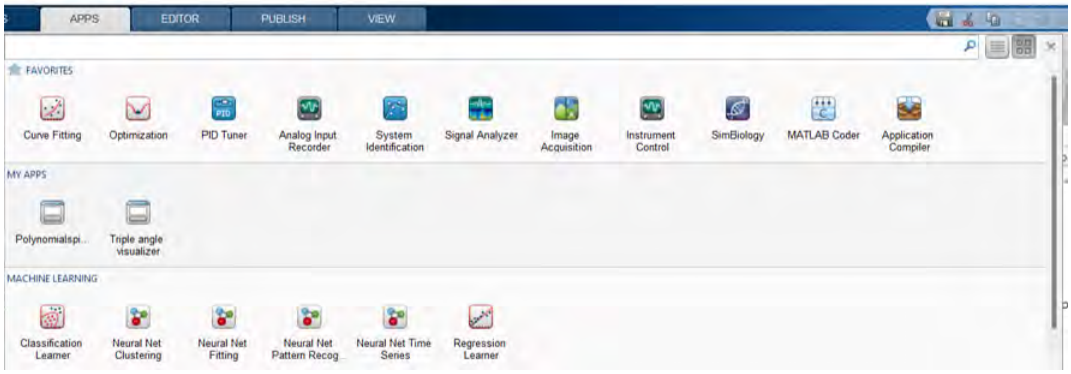

Figura 3.13: Vista de herramientas de Matlab disponibles para su uso en redes neuronales.

la función de transferencia. Además, es posible elegir el método de entrenamiento que, para nuestro caso ha sido ha sido la retro propagación de Levenberg-Marquardt. Una vez realizado lo anterior, es posible dar paso al entrenamiento de la red neuronal. Un ejemplo de la pantalla de entrenamiento de la herramienta NFF se puede apreciar en la fig. 3.14 .

Figura 3.14: Ventana de entrenamiento de la aplicacion Neural Net Fitting.

Una vez realizado el entrenamiento, se tendrá una pestaña donde será posible visualizar los resultados, y a través de ella, elegir las gráficas a analizar (ver el recuadro plots de la fig. 3.15).

También es posible generar un código programable (utilizable en linea de comandos

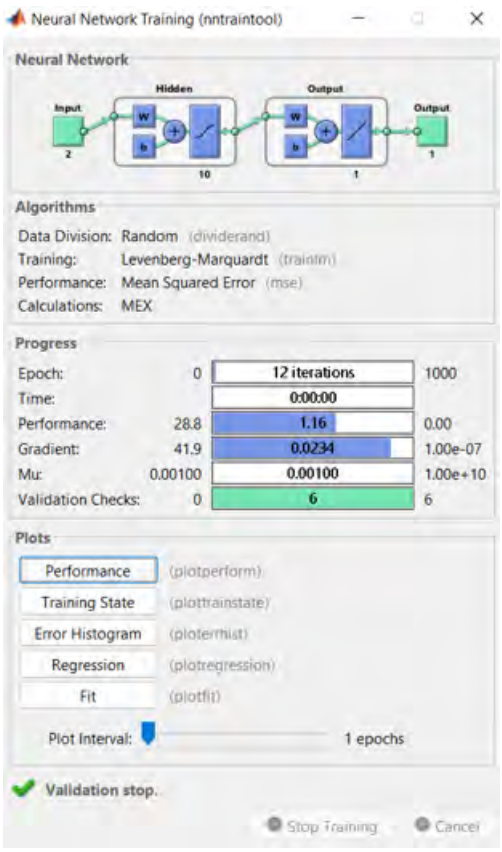

Figura 3.15: Ventana de control de la red ya entrenada.

de Matlab u otro dispositivo) relacionado a la red neuronal entrenada y ser utilizada en un sistema-microcontrolador con capacidad para ejecutar una versión de Matlab portable; por ejemplo, dispositivos o sistemas Raspberry).

Para nuestro caso, es necesario generar tanto el archivo script como el de la función de la red neuronal ya entrenada. Lo anterior, genera una ultima ventana en la herramienta NNF la cual tiene 4 opciones. Sin embargo, solo se seleccionarán las primeras 2 (ver recuadro rojo de la fig. 3.16).

Ya que la red neuronal ha sido entrenada, es posible realizar un diagrama de nuestra red propuesta (ver la fig. 3.17). Para ello, debemos reflejar los datos de entrada, neuronas, dimensión de los pesos calculados.

Para que la red neuronal pueda ser programada en un microprocesador, la ecuación lineal ec. (3.1) debe ser desarrollada en el sistema microprocesador utilizando el resultado de las métricas obtenidas tras el entrenamiento.

En este sentido, ec. (3.1) representa la evaluación de la primera capa de la red neu-

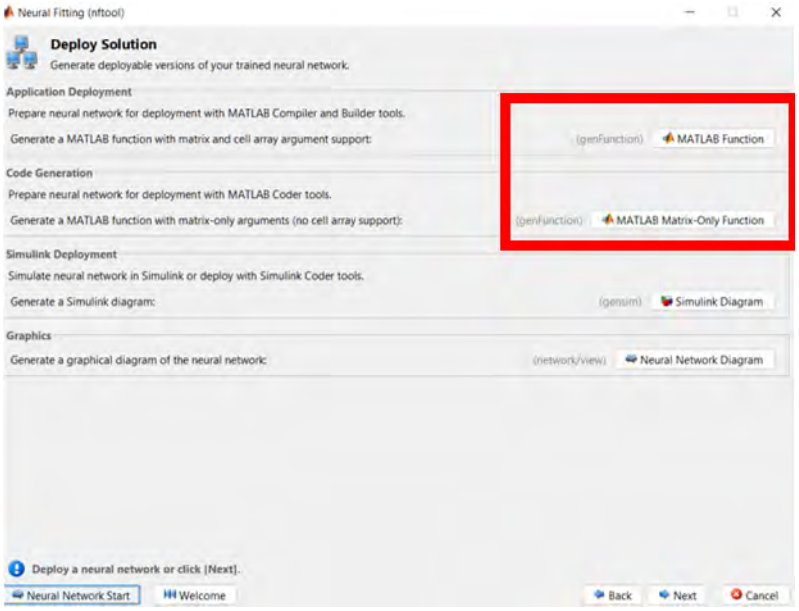

Figura 3.16: Ventana final de la herramienta de NNF.

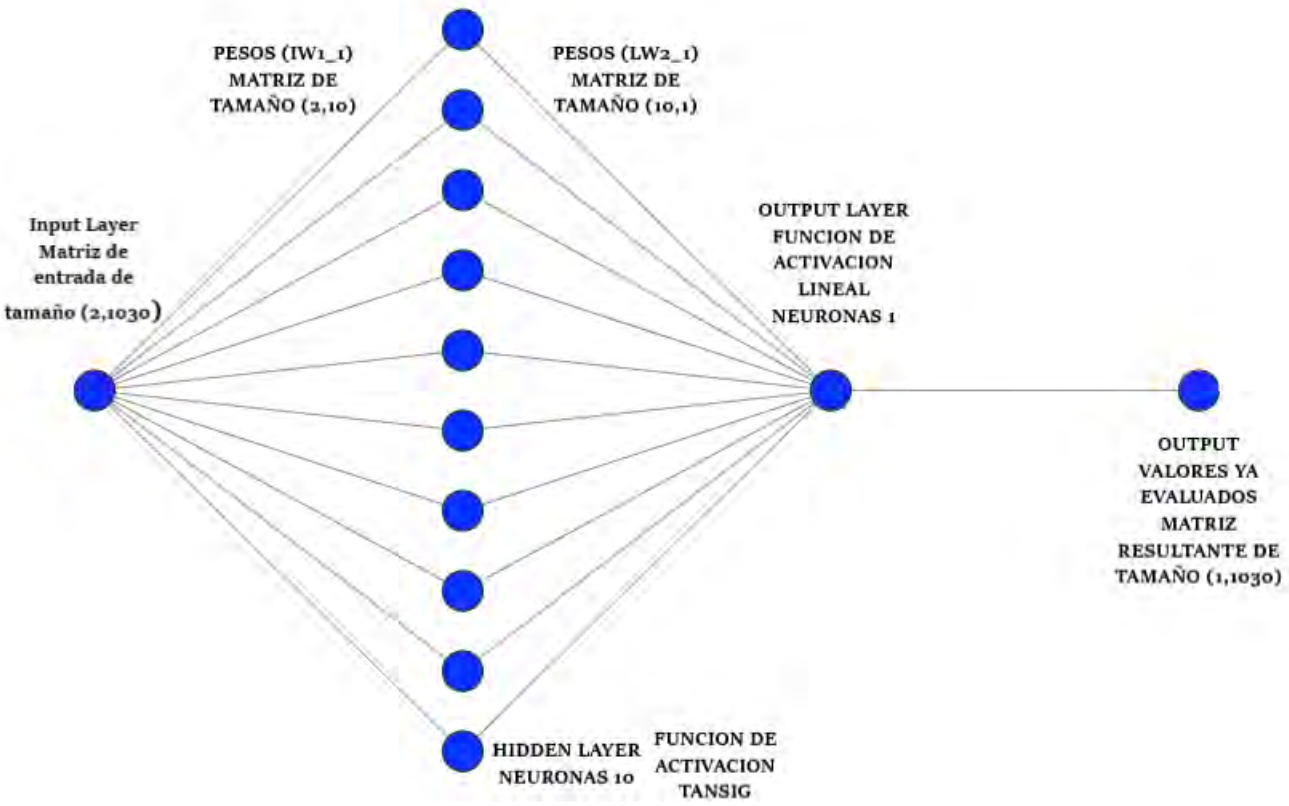

Figura 3.17: Arquitectura de la red neuronal diseñada.

ronal, y la ec. (3.2) representa el segundo nivel de la red neuronal (salida):

$$x = lin(w_1 * h + 1 * b_1) \tag{3.1}$$

$$y = w_2 * x + b_2 \tag{3.2}$$

donde W representa los costos, o pesos, que la red ha calculado para la predicción del valor de salida por capa, b son los bias, h representa los datos de entrada y lin representa la función de activación definida para la capa en particular. En este caso, Matlab tiene de forma predefinida operar la red neuronal con la función de activación tansigmoidal (Hyperbolic tangent sigmoid activation function). Esta función (ver ec. (3.3)) toma una matriz de vectores de entrada netos, y devuelve la matriz S-by-Q de los elementos de de entrada con valores entre -1 y 1 [21]. La función tansigmoidal es definida como sigue:

$$tansig(b) = \frac{1}{1 + e^{-2*b}} - 1 \tag{3.3}$$

Para los datos de aprendizaje, se modificó el código de la propuesta del DPMS basado en el filtrado de Kalman para convertirlo en una función, y con ello, realizar pruebas de automatización con el código general de la red neuronal.

Posterior a ello, se realizó un cambio en cuanto automatización del código de la red neuronal con el objetivo de hacer pruebas de carga rápida de datos. Lo anterior, con la finalidad de evitar procesos innecesarios y optimizar el tiempo de trabajo del algoritmo, el resultado de la primera prueba de aprendizaje puede apreciarse en la fig. 3.18.

Para extraer los valores de los pesos de cada neurona de la red neuronal mediante Matlab, se utilizó la línea de comandos mediante el uso del comando, o función, "net".

Así mismo, se definieron las etiquetas o nombres ”IW1_1"para estar asociados los pesos de las neuronas de la capa 1 y ”LW2_1"para los pesos de la capa de salida. Para acceder a los pesos es necesario complementar el comando "netçon las etiquetas asociados a los pesos. Por ejemplo, net.IW1_1, para el caso de la capa 1.

Una segunda manera de extraer los pesos es a partir de la misma herramienta NNF utilizada para diseñar la red neuronal. Como se pudo observar en la Fig. 3.16, se puede generar una función en Matlab, la cual es modificable por el diseñador de la red neuronal una vez que conocemos los nombres de las variables en la forma que los genera el código

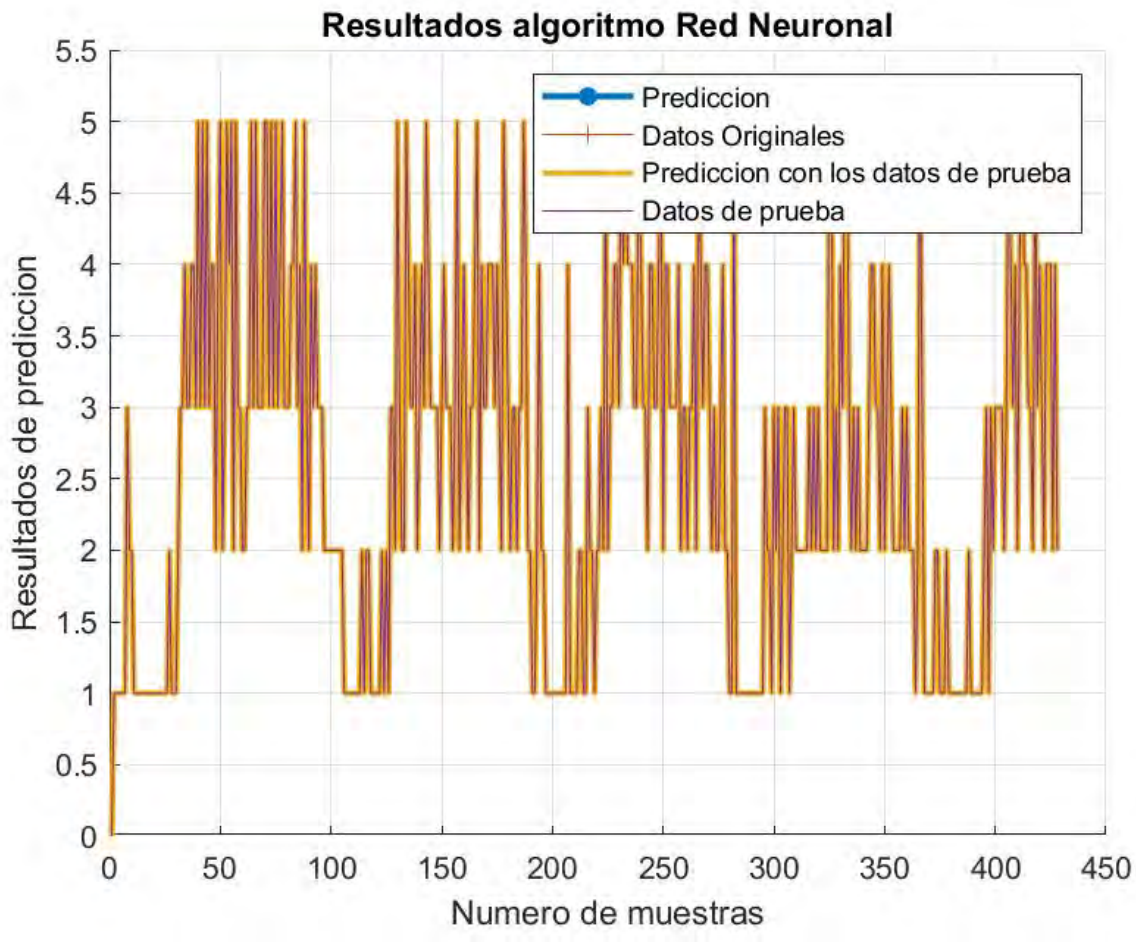

Figura 3.18: Resultados de la clasificación de la prueba de la red neuronal.

proporcionado por Matlab (etiquetas de los pesos), ver Fig. 3.19 para observar los pesos y bias definidos entre las lineas 26 y 32.

```
26    % Layer 1
27 -  b1 = [-4.3664958078553564036;3.1893777306327701915;2.2963827945724104573;-1.8160988501554733432;0.499110921(
28 -  IW1_1 = [4.439724387613637191 0.58821704821779652317;-2.8062496330280546353 3.677008636934473 44;-4.43683806(
29
30    % Layer 2
31 -  b2 = 0.51606918985349159446;
32 -  LW2_1 = [-0.15971414557586494087 -0.17875308419085308986 -0.22236474316081827896 0.13387912986504171831 0.1:
33
```

Figura 3.19: Ejemplo de programación de los pesos y bias obtenidos de la herramienta NNF una vez entrenada la red neuronal.

Por otra parte se diseñó una red neuronal utilizando el método de aprendizaje K-nn [20]. Para ello fue necesario realizar el entrenamiento de la red mediante la linea de

comandos de Matlab; es decir, actualmente Matlab no se dispone de una herramienta flexible como NFF para K-nn. Los códigos asociados al diseño de la red neuronal basada en K-nn se pueden consultar en el Anexo C de este documento. La razón del diseño de esta nueva red fue con la finalidad de tener una red neuronal de referencia para ser comparada con la red neuronal obtenida a través de la herramienta NNF.

Capítulo 4

Etapa de Prueba

4.1. Resultados de prueba del algoritmo WOS

Para el testeo del algoritmo WOS, se utilizaron una serie de datos de medición de ruido ambiental obtenidas de [19]. Cabe mencionar que el algoritmo diseñado guarda automáticamente las gráficas generadas en imágenes en formato “.jpeg”. Las gráficas resultantes de la evaluación del algoritmo DPMS basado en WOS se pueden observar en la Fig. 4.1 y Fig. 4.2 :

4.2. Resultados de pruebas del algoritmo DPMS basado en filtrado de Kalman.

Para el algoritmo de Kalman se utilizaron los mismos datos de medición de ruido, al igual que en el algoritmo de WOS, éstas se almacenaron en formato .jpeg para poder ser observadas con mas detalle, los resultados se pueden apreciar en las Fig. 4.3 y Fig. 4.4.

4.3. Resultados de pruebas del algoritmo DPMS basado en redes neuronales

En cuanto al entrenamiento de la red neuronal se obtuvieron resultados satisfactorios, ya que como se aprecia en la Fig. 4.5 el entrenamiento arroja solamente pequeños errores de aproximación por décimas. De las magnitudes de los resultados, se comenta que la

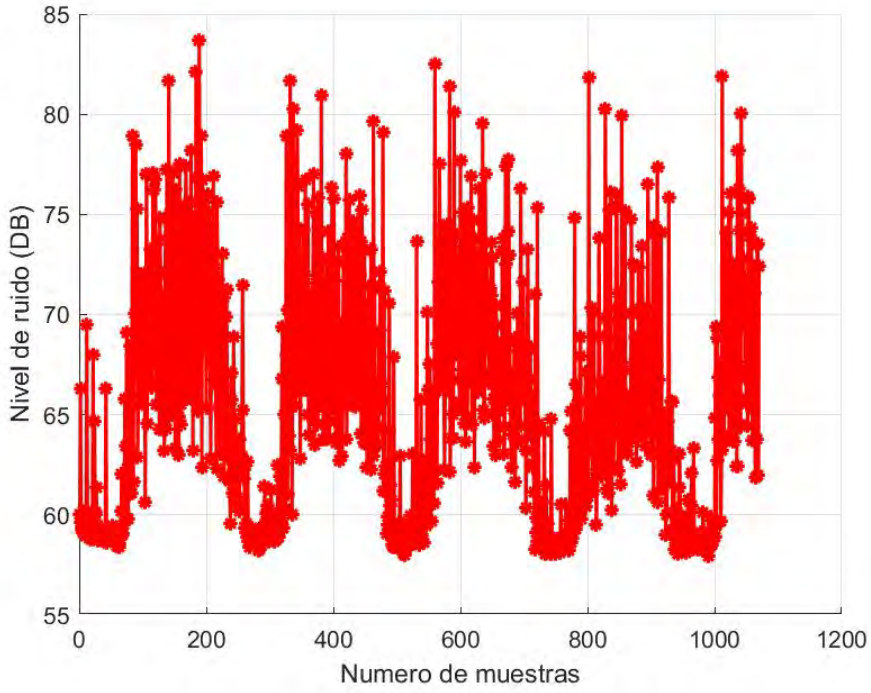

Figura 4.1: Datos de entrada de medición de calor.

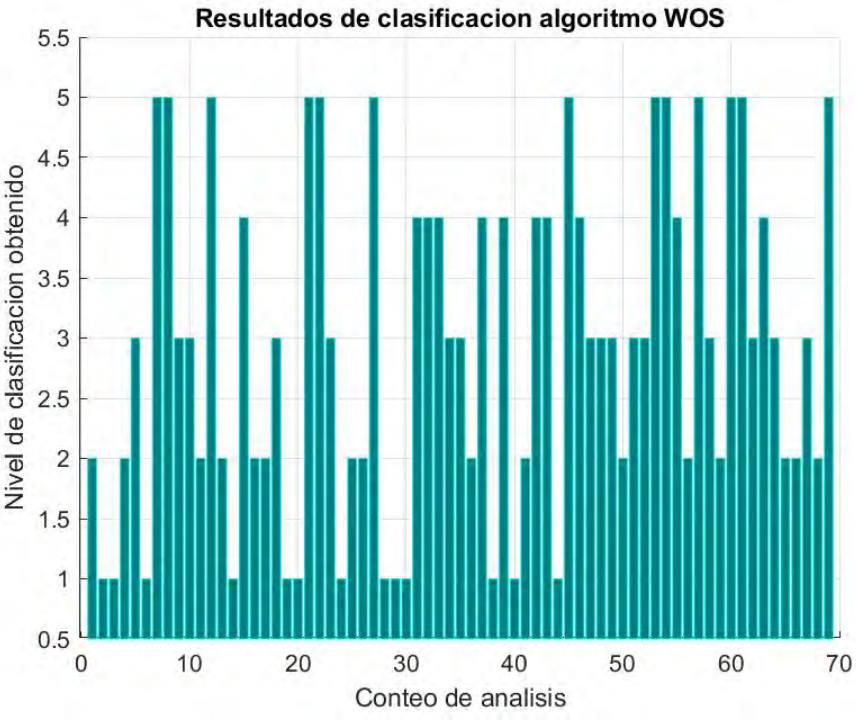

Figura 4.2: Clasificación realizada del algoritmo DPMS basada en WOS.

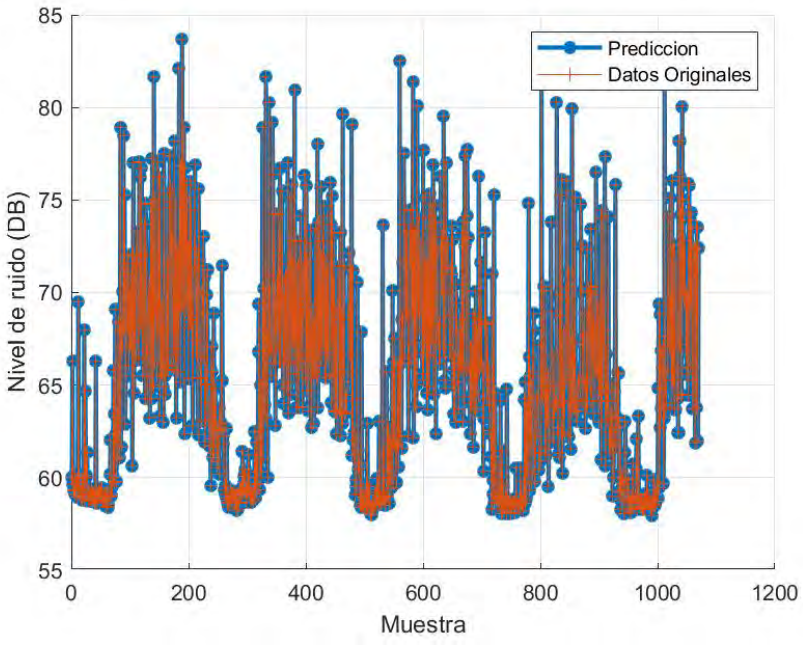

Figura 4.3: Comparación entre los datos de entrada y la predicción de comportamiento realizada por el algoritmo de Kalman.

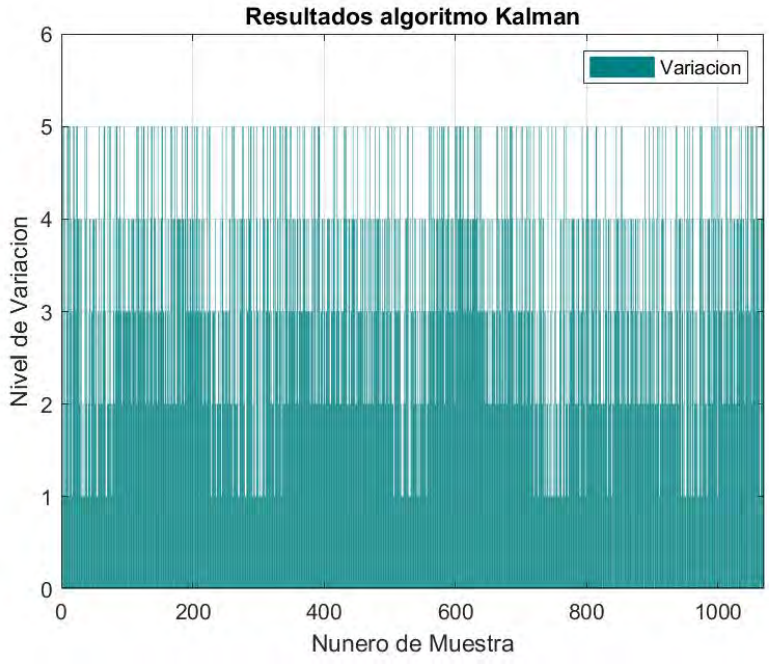

Figura 4.4: Resultados de la clasificación de variación de tiempos calculada por el algoritmo.

parte entera es las más importante, por lo que aplicando un filtro el cual elimina las decimales sobrantes en los valores resultantes, se pueden alcanzar resultados con mejor nivel de aproximación (ver Fig. 4.6). En dicha figura, se puede observar que el error resultante es mínimo.

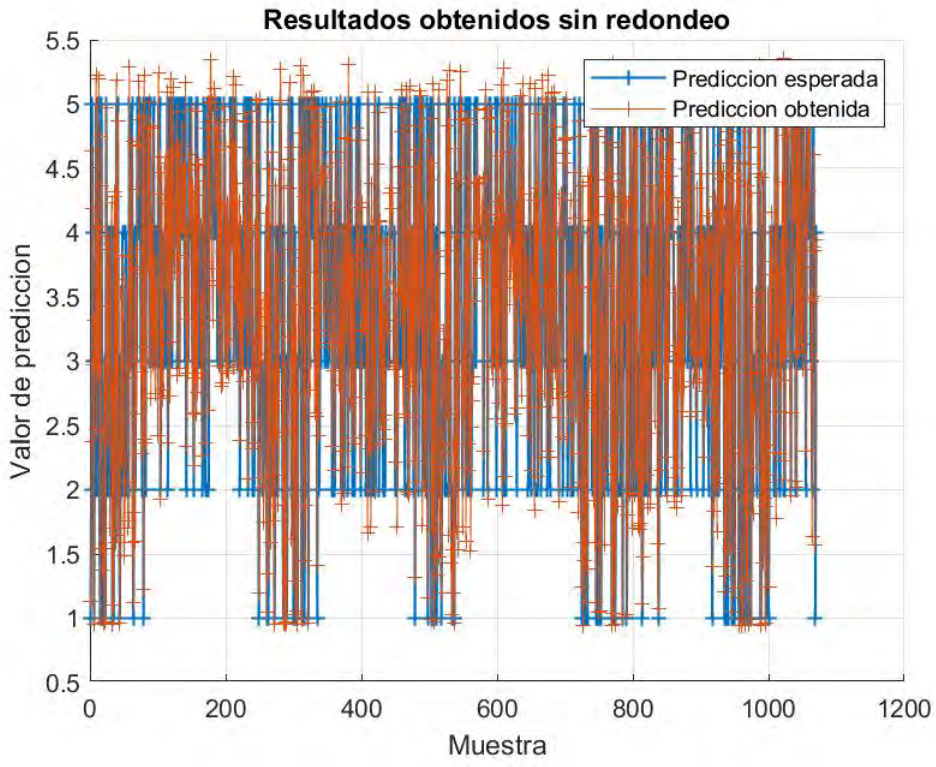

Figura 4.5: Resultado de la predicción tomando en cuenta las decimales.

Con respecto al código diseñado utilizando la herramienta NNF (para la cual se eligió el método de entrenamiento como la retro propagación de Levenberg-Marquardt, LM), los resultados de los entrenamientos fueron muy favorables resultando tasas de error menores al 1 %. Los resultados se pueden apreciar en las Figs. 4.7 y 4.8.

Para poder apreciar mejor la precisión de los valores, se puede observar la Fig. 4.9. El resultado de la predicción de la red neuronal tomando solo los valores enteros con una vista mas ampliada.

A partir de lo anterior, se pudieron obtener los valores de costos (o pesos) para programar en el microcontrolador, para la capa 1 se se los pesos presentaos en la Tabla 4.2, en la cual se ve que los costos de las neuronas coinciden con el numero de neuronas de la capa y con la cantidad de valores de entrada. De igual manera, fue posible obtener los

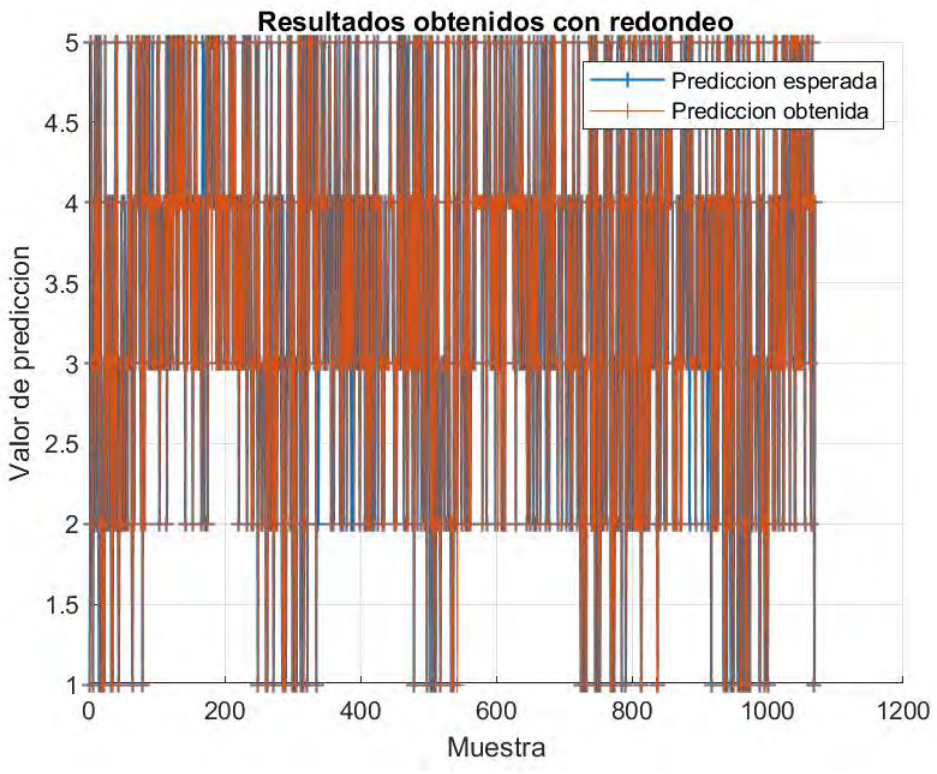

Figura 4.6: Resultado de la predicción solo tomando valores enteros necesarios.

valores de la matriz de bias para la capa 1, las cuales se pueden observar en la Tabla 4.1.

Neurona	valor de la Bia
1	-4.6485565971597013757
2	-2.2699133412895724504
3	-1.8438692519015336124
4	1.6706727530765699274
5	2.0698770025404251705
6	-0.022571128997700737329
7	-1.8364442120050032958
8	-1.0802860344625719868
9	3.9723175581858565231
10	-3.78338656059072731

Tabla 4.1: Bias de la capa 1 de la red neuronal.

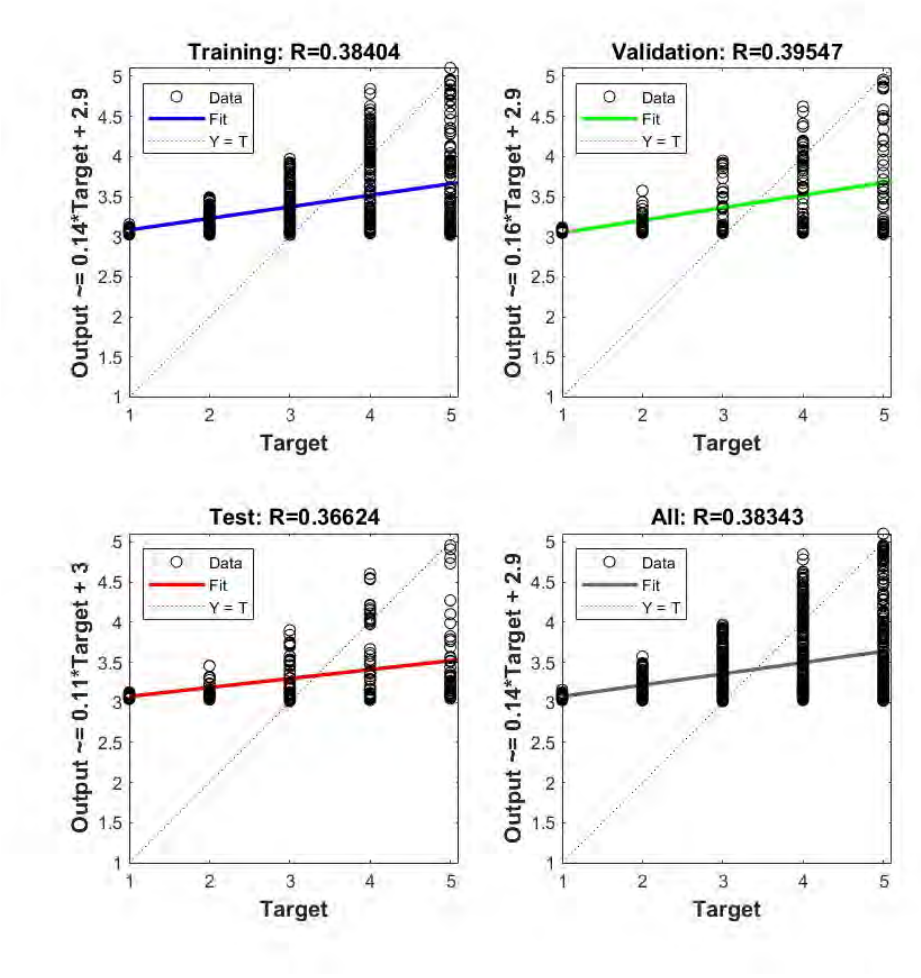

Figura 4.7: Resultados de la gradiente calculada por el algoritmo implementando LM.

Al igual que con la capa 1, en la capa 2 podemos encontrar valores de costos y bias, los resultados arrojados por el entrenamiento de la red neuronal para dicha capa se pueden apreciar en las Tablas 4.3 y 4.4, hay que mencionar, que en la capa 2 únicamente se cuenta con 1 neurona, por lo que los datos de costos y bias en tamaño comparado con la capa 1 se reducen. Así mismo, para que la multiplicación entre matrices sea posible, la forma de almacenar los valores de la capa 2 cambio a un vector de tipo fila.

Ahora tomando en cuenta lo mencionado en las ecuaciones (3.1), (3.2) y (3.3), éstas se puede programar en el microcontrolador únicamente 2 lineas de código que representan las 2 capas de la red neuronal para poder generar una predicción rápida y con el menor costo computacional posible, las ecuaciones resultantes son Eq. (4.1) y Eq. (4.2).

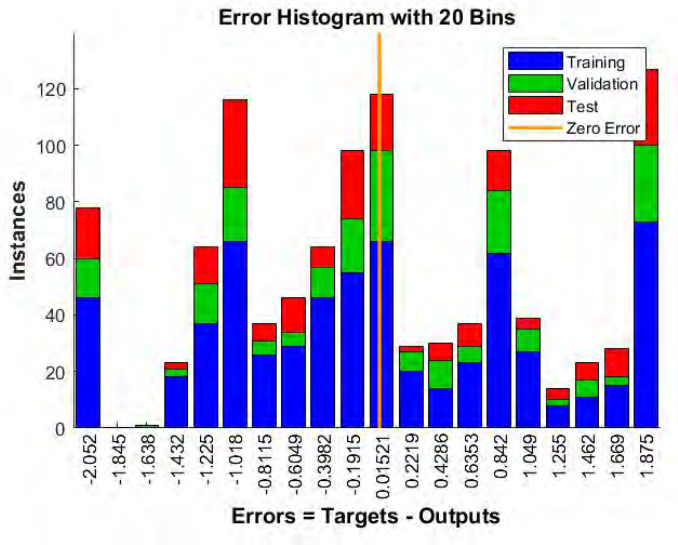

Figura 4.8: Histograma de errores evaluados en el algoritmo utilizando LM.

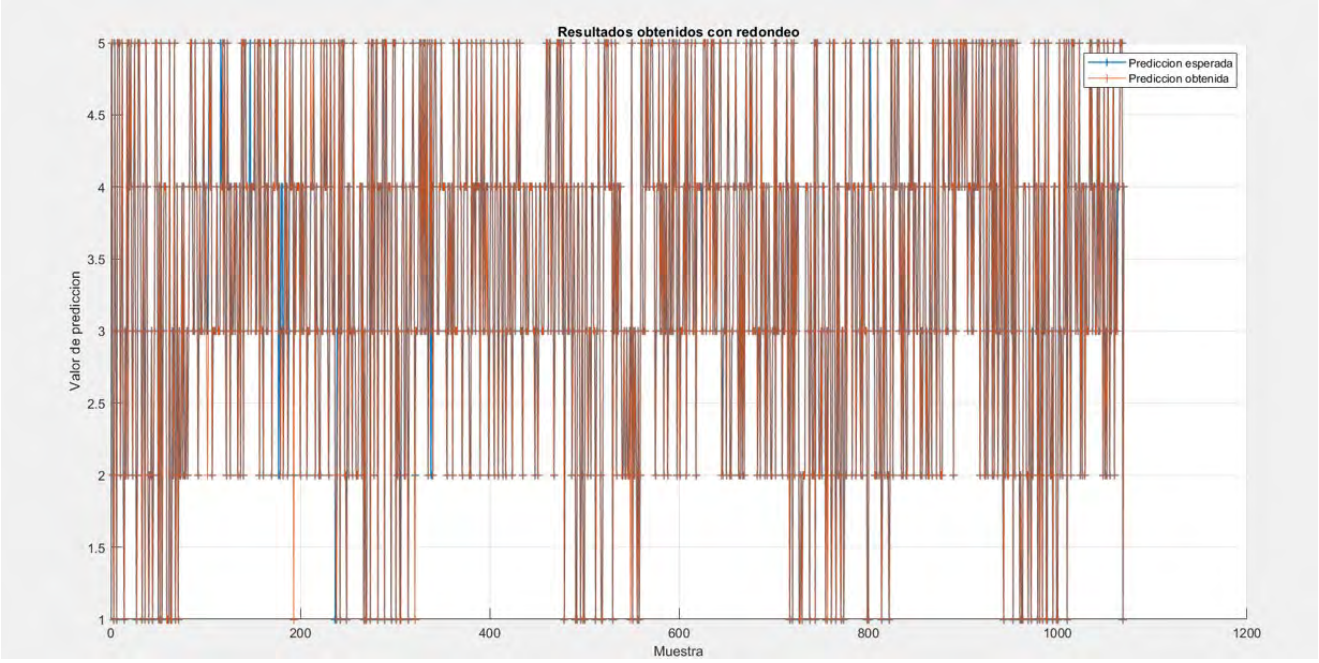

Figura 4.9: Vista ampliada de los resultados únicamente tomando los valores enteros necesarios.

$$x = tansig(w_1 * h + 1 * b_1) = \frac{1}{1 + e^{-2*(w_1*h+1*b_1)}} - 1 \tag{4.1}$$

$$y = w_2 * x + b_2 \tag{4.2}$$

Neurona	Columna 1	Columna 2
1	4.484034202874497	0.456078755642162
2	4.511963828131733	-3.11351147551946
3	2.827837323011379	1.27729794191363
4	-1.39024611262362	2.64893302302475
5	-2.32144492394955	2.92444449475657
6	-2.53425161634025	-1.97702972373176
7	-2.78443505414407	-1.59278517819633
8	-2.13596960433773	1.48813912293402
9	4.32389503506352	-0.100497847976193
10	-3.36856339911641	-1.20970281580107

Tabla 4.2: Pesos (o costos) de la capa 1 de la red neuronal.

Neurona	Valores de las Bias
1	-0.20527161931227747016

Tabla 4.3: Bias de la capa 2 de la red neuronal.

En el caso de la Eq. (4.2), la cual representa la función de activación de la capa 2, dicha función es únicamente lineal, por lo que solo toma los valores resultantes de la eq. (4.1). Además, los valores de w_1 son tomados de la Tabla 4.2 y los valores de b_1 son tomados de la Tabla 4.1, por lo que la forma de programarlas en el microcontrolador, ya sea en formato de matrices o de Tablas, dependerá del lenguaje del microprocesador a utilizar. Por otra parte, los valores de w_2 y b_2 correspondientes a la capa 2 son tomados de las Tablas 4.4 y 4.3 respectivamente; y al igual que sus semejantes de la capa 1.

Fila	Neurona 1
1	-0.802214568145753
2	0.286939579534737
3	0.417403930097405
4	-0.819690256770354
5	0.707967552709661
6	-0.275710964869178
7	-0.375530146002106
8	0.452894829257385
9	-0.412098234354104
10	-0.730845186065644

Tabla 4.4: Pesos (o costos) de la capa 2 de la red neuronal.

Capítulo 5

Conclusiones

En la elaboración de la tesis se abordó el problema del alto consumo energético en nodos sensores de red cuando no disponen de estrategias de administración de energía en dichos nodos. Así mismo, el no contar con técnicas DPMS optimizadas en los nodos de red ocasionará de forma constante, que las fuentes de almacenamiento de energía sean cambiadas (o desechadas), impactando de forma negativa en el ambiente. Una forma de mitigar el impacto ambiental es utilizar técnicas novedosas de cosecha de energía para energizar los nodos sensores de red como. Las técnicas de cosecha de energía más comunes están basadas en el uso de paneles fotovoltaicos, dispositivos de radiofrecuencia, dispositivos TEG, entre otros. Sin embargo, tales técnicas necesitan apoyarse de estrategias DPMS que contribuyan al uso eficiente de la energía por parte del nodo sensor, y con ello, garantizar la operación ininterrumpida de los mismos.

Durante el transcurso de este trabajo se elaboraron 3 propuestas de algoritmos para su uso como estrategias DPMS: a) algoritmo WOS, b) algoritmo basado en el Filtrado de Kalman, y finalmente, c) algoritmo basado en Redes neuronales (ANN).

Los 3 algoritmos implementados han demostrado ser capaces de realizar las clasificaciones para el control de tiempo de sueño de los dispositivos de manera eficiente, por lo que gracias a eso pueden ser implementados en WSN, o dispositivos IoT, para optimizar el consumo energético de los mismos reduciendo el impacto que genere un desgaste en las baterías de los nodos sensores de red.

Así, el implementar DPMS eficientes impactará de manera positiva en los costos por mantenimiento, reemplazo de pilas, y desecho de pilas desgastadas en los nodos sensores de red. Lo anterior contribuirá a mitigar el impacto ambiental, ya que al prolongar la

vida útil de las fuentes de almacenamiento de energía, el mantenimiento y reemplazo de las mismas también se prolongará.

Finalmente comento que, de forma muy personal, que este proyecto me ha dejado un gran aprendizaje en el área de estrategias de administración energética en nodos sensores de red. Específicamente, me permitió contribuir en la implementación de tres técnicas DPMS, las cuales fueron presentadas analíticamente y probadas a través de herramientas de cómputo científico como MATLAB. De manera adicional, en cuanto habilidades adquiridas, este trabajo permitió que mis conocimientos sobre programación se expandieran, al abarcar implementaciones basadas en funciones objetivo, así como, comprender y entender el impacto de las métricas elegidas para la mejora de los DPMS a desarrollar. También, se observó acorde al estado del arte estudiado, una tendencia al uso de métodos de predicción basados en el aprendizaje de las IA's (incluso este trabajo se presenta una técnica basada en IA); sin embargo, este tipo de propuestas deben ser estudiadas a detalle en cuanto a complejidad computacional, dado que, pueden impactar en el consumo energético perdiéndose el objetivo del por que fueron propuestos.

Apéndice A

Anexos

A.0.1. Anexo A: Códigos generado en Matlab para el algoritmo WOS.

```
clc;
clear all;
close all;
%ejemplo de algoritmo de kallman
%declaramos variables de tiempo de sue o Tsleep
C1=5;
C2=10;
C3=15;
C4=20;
C5=25;
%Para este ejemplo, se utilizaran los valores proporcionados por los
%docentees, los cuales son nombrados como Dat1 y Dat2, los cuales se
   cargan
%a continuacion
Datos=load('Dat1.mat');
tamMuestra=size(Datos.VarName1);
%Declaramos informacion a priori
M=[min(Datos.VarName1) min(Datos.VarName1)+(6.4275*1) min(Datos.VarName1)
   +(6.4275*2) min(Datos.VarName1)+(6.4275*3) max(Datos.VarName1)];
%aplicamos algoritmo de kallman
```

```
%%
%calculamos la media de los tramos de 5 valores
cont=1;
band=1;
dat=1;
postslep=1;
while band == 1
    RecMuestra=dat;
    if RecMuestra>(tamMuestra(1,1)-10)
        break
    end
    for i=1:5
        %Calculamos MCU
        %Extraemos valores de muestra
        vectmediana(i)=Datos.VarName1(RecMuestra);
        %ordenamos los valores de muestra
        vectmediana=sort(vectmediana);
        %obtenemos la mediana
        Mediana=median(vectmediana);
        %ste contador nos permite extraer el siguiente valor del conjunto
        %de muestras
        RecMuestra=RecMuestra+1;
    end
    %una ves teniendo la mediana se procede a obtener el valor absoluto de
    %cada elemento del vector Menos la media obtenida
    for j=1:5
        Deltai(j)=abs(vectmediana(j)-Mediana);
    end
    %ahora elevamos cada resultado obtenido de la resta al cuadrado y
    %realizamos la sumatoria de los mismos
    Epsiloni=0;
    for j=1:5
```

```
    Epsiloni=Epsiloni+Deltai(j)^2;
end
%Ahora Obtenemos el valor absoluto de la diferencia entre los datos a
   priori y la sumatoria
%obtenida previamente
for k=1:5
    Rho(k)= abs(M(k) - Epsiloni);
end
%obtenemos el minimo de Rho para saber cual fue la variacion que se
%tiene de minima media o maxima y obtenemos su posicion en el vector
Minimo=min(Rho);
%declaramos una bandera para un ciclo comparador
bandcomp=0;
contrec=0;
while bandcomp == 0
    contrec=contrec+1;
    if(Minimo)==Rho(contrec)
        bandcomp=1;
    end
end
%ahora definimos en base al minimo el tiempo de sue o que tendra
switch contrec
    case 1 %este define minima variacion
        dat=dat+C5;
    case 2 %este define media variacion
        dat=dat+C4;
    case 3 %este define Alta variacion
        dat=dat+C3;
    case 4 %este define Alta variacion
        dat=dat+C2;
    case 5 %este define Alta variacion
        dat=dat+C1;
```

```
    end
    %creamos un vector que almacene las variaciones
    tslepsWOS(postslep)=contrec;
    postslep=postslep+1;
end
fig=figure (1);
hold on
%plot(tsleps,'LineWidth',1.5,'Marker','*')
title('Resultados algoritmo WOS')
bar(tslepsWOS,'FaceColor',[0 .5 .5],'EdgeColor',[0 .9 .9])
ylim([0.5 5.5]);
xlabel('Conteo de analisis')
ylabel('Nivel de variacion de datos')
hold off
grid on;
saveas(fig,'ResultadoVariacionWOS.jpeg');
fig= figure (2);
hold on
plot(Datos.VarName1,'color','r','LineWidth',1.5,'Marker','*')
hold off
xlabel('Numero de muestras')
ylabel('Nivel de ruido (DB)')
grid on;
saveas(fig,'DatosdeAnalisisWOS.jpeg');
```

A.0.2. Anexo B: Códigos generado en Matlab para el algoritmo de Kalman.

```
clc;
clear all;
close all;
%ejemplo de algoritmo de kallman
%Declaramos Variables iniciales
Q=1;
R=1;
q=0;
r=0;
%Para este ejemplo, se utilizaran los valores proporcionados por los
%docentees, los cuales son nombrados como Dat1 y Dat2, los cuales se
   cargan
%a continuacion
Datos=load('Dat1.mat');
tamMuestra=size(Datos.VarName1);
%% Ejecuta Seccion de ciclo de Algoritmo
%declaramos variables de datos
%F y H valen 1 porque unicamente medimos una se al , si fueran ams
   se ales el
%valor cambia, convirtiendose en una matriz
Fxk=1;
H=1;
Eqk=0;
Erk=0;
xk=59;
%declaramos las ecuaciones iniciales
%yk=H*xk+r;
%declaramos los valores de la ganancia;
%EkqkT=R;
```

```
%yk es el crudo de los datos de medicion
%xk es el valor inicial con el que recalculara el ciclo
%Asignamos el vector de datos a analizar
Pk=10000;
Yk=Datos.VarName1;
%iniciamos el ciclo;
tam=size(Yk);
for i=1:tam(1,1)
%P k+1 =  Pk - Kk*H*Pk + Qk
%Kk = Pk*H*[R+H*Pk*H]^(-1)
%Xg k+1= Xgk + Kk*(yk -ykg)
    %Gk=Pk*H*(H*Pk*H+R);
    Gk=Pk*H*(H*Pk*H+R)^(-1);% error Oscar
    %almacenamos el resultado del valor para el tiempo de sue?o
    pGK(i)=Gk;
    xk=xk+Gk*(Yk(i)-H*xk);
    Xksav(i)=xk;
    pxk(i)=xk;
    Pk=Pk;%-Pk*Gk*H; %%% Conflico con ecs (7,10)
    % Para valores de Pk grandes estima mejor
end
fig=figure (1);
hold on;
plot(pxk,'LineWidth',2,'Marker','*','DisplayName','Prediccion')
plot (Yk,'DisplayName','Valor original','Marker','+','DisplayName','Datos
   Originales');
xlabel('Muestra');
ylabel('Nivel de ruido (DB)');
legend
grid on;
saveas(fig,'DatosComparativoKalman.jpeg');
```

A.0.3. Anexo C: Códigos generado en Matlab para el algoritmo de la Red Neuronal utilizando K-nn.

Algoritmo Kalman modificado para trabajar con la red neuronal.

```
function [w,pwm] = FNCKalman(Datos)
ym = [Datos];;
dym = length(ym);
y1 = zeros(dym,1); y1(1)=10; x = zeros(dym,1);
xg = zeros(dym,1); xg(1)=59; xg(2)=59; xgm = zeros(dym,1); xgm(1)=0;

Pk = zeros(dym,1); Pk(1)=0; Pm = zeros(dym,1);
Pm(1)=1000;
Gk = zeros(dym,1); Gk(1)=0; w = zeros(dym,1); w(1)=0;

%%%% Salida digital
pwm = zeros(dym,1); pwm(1)=0;
%%%%%%%%%%%%
G1=1; F=1; H = 1; R=1;
for j=2:1:dym
x(j+1) = x(j) + G1.* ym(j) ;
xgm(j) = F*xgm(j-1);
y1(j) = H*ym(j);
Pm(j) = F*Pm(j-1)*F + 0;
Gk(j) = Pm(j)*H*(H*Pm(j)*H+R)^(-1);
xg(j+1) = xgm(j)+Gk(j)*(y1(j)-H*xgm(j));
%%% %%% El segundo integra la condicion de la entrada...
w(j+1) = w(j)+Gk(j)*(y1(j)-w(j));
%%%%% Genera PWM a partir de la variable estimada w
if (w(j) >= 0) & (w(j) <= 50)
pwm(j) = 0;
elseif (w(j) > 50) & (w(j) <= 60)
pwm(j) = 1;
```

```
elseif (w(j) > 60) & (w(j) <= 65)
pwm(j) = 2;
elseif (w(j) > 65) & (w(j) <= 70)
pwm(j) = 3;
elseif (w(j)> 70) & (w(j)<= 75)
pwm(j) = 4;
else
pwm(j) = 5;
end
end
end
```

Codigo de red neuronal aplicando entrenamiento K-nn utilizando los datos modificados del codigo del algoritmo Kalman mencionado en el Anexo C.

```
% Ej NN Usa valores de NN8GBjul21 w (salida del Filtro de
% Kalman) y pwm (selector)
%clear all; clc; close all;

%[ax,ay] = bodyfat_dataset;
Datos=load('Dat1.mat');
%%
x=(Datos.VarName1);
[ax,ay] = FNCKalman(x);

p = cvpartition(1070,'HoldOut',0.4);
itrain = p.training; %%% indice dataset
itest = p.test; %%% indice prueba
Xtrain = ax(itrain,:); %%%% solo los de la particion para entrenar
ytrain = ay(itrain); %%%% solo los de la particion para entrenar
Xtest = ax(itest,:); %%%% %%%% solo los de la particion para prueba
ytest = ay(itest); %%%%  solo los de la particion para prueba

%model1 = fitcnb(Xtrain,ytrain) %%% entrenamiento de red, con datos de
   entrenamiento
model2 = fitcknn(Xtrain,ytrain);
% Fit k-nearest neighbor classifier

yhat = model2.predict(Xtest); %%% predicci n de salida con los datos de
   preueba
%plotconfusionchart(categorical(ytest),categorical(yhat)) % muestra los
   porcentajes de exactitud de clasificaci n y error.
%%% confusionchart
fig=figure (1);
```

```
plot(yhat,'DisplayName','Prediccion con los datos de prueba','LineWidth'
   ,2)
hold on
title('Resultados algoritmo Red Neuronal')
plot(ytest,'DisplayName','Datos de prueba');
legend
xlabel('Numero de muestras');
ylabel('Resultados de prediccion');
ylim([0 5.5])
saveas(fig,'ResultadosRedNeuronalTest.jpeg');
```

A.0.4. Anexo D: Códigos generado en Matlab para el algoritmo de la Red Neuronal utilizando NNF.

```
%% Fase 1 evaliuacion de datos de prueba
clc;
clear all;
close all;
%ejemplo de algoritmo de kallman
%Declaramos Variables iniciales
Q=1;
R=1;
q=0;
r=0;
%Para este ejemplo, se utilizaran los valores proporcionados por los
%docentees, los cuales son nombrados como Dat1 y Dat2, los cuales se
   cargan
%a continuacion
Datos=load('Dat1.mat');
tamMuestra=size(Datos.VarName1);
%% Ejecuta Seccion de ciclo de Algoritmo
%declaramos variables de datos
%F y H valen 1 porque unicamente medimos una se al , si fueran ams
   se ales el
%valor cambia, convirtiendose en una matriz
Fxk=1;
H=1;
Eqk=0;
Erk=0;
xk=59;
%declaramos las ecuaciones iniciales
%yk=H*xk+r;
%declaramos los valores de la ganancia;
```

```
%EkqkT=R;
%yk es el crudo de los datos de medicion
%xk es el valor inicial con el que recalculara el ciclo
%Asignamos el vector de datos a analizar
Pk=10000;
Yk=Datos.VarName1;
%iniciamos el ciclo;
tam=size(Yk);
for i=1:tam(1,1)
%P k+1 =  Pk - Kk*H*Pk + Qk
%Kk = Pk*H*[R+H*Pk*H]^(-1)
%Xg k+1= Xgk + Kk*(yk -ykg)
    %Gk=Pk*H*(H*Pk*H+R);
    Gk=Pk*H*(H*Pk*H+R)^(-1);% error Oscar
    %almacenamos el resultado del valor para el tiempo de sue?o
    pGK(i)=Gk;
    xk=xk+Gk*(Yk(i)-H*xk);
    Xksav(i)=xk;
    pxk(i)=xk;
    Pk=Pk;%-Pk*Gk*H; %%% Conflico con ecs (7,10)
    % Para valores de Pk grandes estima mejor
end
fig=figure (1);
hold on;
plot(pxk,'LineWidth',2,'Marker','*','DisplayName','Prediccion')
plot (Yk,'DisplayName','Valor original','Marker','+','DisplayName','Datos
   Originales');
xlabel('Muestra');
ylabel('Nivel de ruido (DB)');
legend
grid on;
saveas(fig,'DatosComparativoKalman.jpeg');
```

```
%% Seccion para evaluar los datos
%Definimos clases
c1=57;
c2=62;
c3=67;
c4=72;
c5=77;
%obtenemos clasificacion para los tiempos
ykres2(:,1)=Yk;
ykres2(:,2)=randi([0 100],1070,1);
for j=1:tam(1,1)
    if (((Xksav(j)>= c1) && (Xksav(j) < c2)) || ((ykres2(j,2)>=0))&&(
       ykres2(j,2)<=20))%Clasifica en nivel 1 (minimo)
        NivSleepResKAL(j)=1;
    end
    if (((Xksav(j)>= c2) && (Xksav(j) < c3)) || ((ykres2(j,2)>20))&&(
       ykres2(j,2)<=40))%Clasifica en nivel 2
        NivSleepResKAL(j)=2;
    end
    if (((Xksav(j)>= c3) && (Xksav(j) < c4)) || ((ykres2(j,2)>40))&&(
       ykres2(j,2)<=60))%Clasifica en nivel 3
        NivSleepResKAL(j)=3;
    end
    if (((Xksav(j)>= c4) && (Xksav(j) < c5)) || ((ykres2(j,2)>60))&&(
       ykres2(j,2)<=80))%Clasifica en nivel 4
        NivSleepResKAL(j)=4;
    end
    if ((Xksav(j)>= c5) || ((ykres2(j,2)>80))&&(ykres2(j,2)<=100))%
       Clasifica en nivel 5 (maximo)
        NivSleepResKAL(j)=5;
    end
```

```
end
fig=figure (2);
bar(NivSleepResKAL,'FaceColor',[0 .5 .5],'DisplayName','Variacion')
title('Resultados algoritmo Kalman')
ylim([0 6])
xlabel('Nunero de Muestra');
ylabel('Nivel de Variacion');
legend
grid on;
saveas(fig,'ResultadosKalman.jpeg');
ykres=transpose(NivSleepResKAL);

ykres2(:,1)=Yk;
ykres2(:,2)=randi([0 100],1070,1);
%% Ejemplo red neuronal basica
%Utilizamos para empezar el comando network, dicho comando funciona en
%matlab para crear la estructura de la red como tu la definas

%La informacion de funcionamiento de comandos en este ejemplo es
%directamente extraida de MathWorks

%net = network(numInputs,numLayers,biasConnect,inputConnect,layerConnect,
   outputConnect)
%NumImputs=Canales de entrada a la red neuronal (Se debe tomar en cuenta
   que pueden entrar multiples datos por un mismo canal)
%NumLayers=Numero de capas de la red neuronal
%biasConnect=Es una matriz la cual define en que capa va estar conectada
   el
%bias
%inputConnect=Define en cual capa estara conectada la entrada de datos
%layerConnect=Define como van a estar las conexiones de las capas, esto se
%hace con una matriz, se hace con la matriz de 0 y 1 donde cada columna
```

```
%y fila representan las capas y tu defines las conexiones entre ellas
%outputConnect=Defines en cual capa va estar conectada la salida
red=network(1,2,[1;0],[1;0],[0 0;1 0],[0 1]);
%tenemos que definir los parametros de la red para eso podemos mandar a
%llamar los comandos a la red especifica usando su nombre
%Para empezar debemos definir las funciones de transferencia de cada capa
red.layers{1}.transferFcn = 'logsig';%Esto define la funcion como una
   sigmoidal
red.layers{2}.transferFcn = 'purelin';%Esto define la funcion de
   transferencia como una recta pura
red.layers{1}.size=1;%Definimos los pesos por capas
red.layers{2}.size=2;
red.inputs{1}.size=1;
red.trainFcn= 'trainlm';
%Nota: El peso de la ultima capa es el mismo que el peso del output
view(red);
```

Bibliografía

[1] A. Castillo Atoche, J. Vazquez Castillo, E. Osorio de la Rosa, J. C. Heredia Lozano, J. Aviles Vinas, R. Quijano Cetina, and J. J. Estrada Lopez, "An energy saving data statistics drivenmanagement technique for bio powered indoor wireless sensor nodes," *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, vol. 1, p. 10, Mar. 2021.

[2] W. Liu, X. Luo, G. Wei, and H. Liu, "Node localization algorithm for wireless sensor networks based on static anchor node location selection strategy," *Computer Communications*, vol. 192, pp. 289–298, 2022.

[3] U. Panahi and C. Bayılmıs, "Enabling secure data transmission for wireless sensor networks based iot applications," *Ain Shams Engineering Journal*, June 2022.

[4] A. Espinoza-Ruiz, J. A. López-Chaides, E. Ruiz-Ibarra, J. Cortez-González, A. García-Berúmen, and J. Ruiz-Ibarra, "Plataforma autoconfigurable de monitoreo remoto para aplicaciones móviles en wsan," *Ingeniería Investigación y Tecnología*, vol. XVI, pp. 369–389, July 2015.

[5] J. García-Espinosa, I. Ortigosa, and A. Fernández, "Desarrollo de un sistema de apoyo a la decisión para optimizar el comportamiento de barcos de vela," *Revista Internacional de Métodos Numéricos paraCálculo y Diseño en Ingeniería*, vol. 31, pp. 146–153, Apr. 2015.

[6] Rubio, J. A. Hernández-Aguilar, F. J. Ávila Camacho, J. M. Stein-Carrillo, and A. Meléndez-Ramírez, "Sistema sensor para el monitoreo ambiental basado en redes neuronales," *Ingeniería Investigación y Tecnología*, vol. XVII, pp. 211–222, Apr. 2016.

[7] P. van Rhyn and G. P. Hancke, “Simplified performance estimation of ism-band, ofdm-based wsns according to the sensitivity/sinad parameters,” *Journal of Applied Researchand Technology*, vol. 15, pp. 1–13, Jan. 2017.

[8] T. Eswaran and V. Suresh Kumar, “Particle swarm optimization (pso)-based tuning technique for pi controller for management of a distributed static synchronous compensator (dstatcom) for improved dynamic response and power quality.,” *Journal of Applied Researchand Technology*, vol. 15, pp. 173–189, July 2016.

[9] A. Z. Vizuete, I. Pérez Llopis, C. P. Salvador, and M. E. Domingo, “Sistema distribuido de detección de sismos usando una red de sensores inalámbrica para alerta temprana.,” *ScienceDirect*, vol. 12, pp. 260–269, June 2015.

[10] M. Pule and J. C. Abid Yahya, “Wireless sensor networks: A survey on monitoring water quality,” *Journal of Applied Researchand Technology*, vol. 15, pp. 562–570, July 2017.

[11] C. A. Fellow, IEEE, M. D. F. Giuseppe Anastasi, and M. Roveri, “An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors,” *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, vol. 59, Feb. 2010.

[12] E. Osorio de la Rosa, J. Vazquez Castillo, A. Castillo Atoche, J. Heredia Lozano, A. Castillo Atoche, G. Becerra Nunez, and R. Barbosa, “Arrays of plant microbial fuel cells forimplementing self sustainable wireless sensor networks,” *IEEE SENSORS JOURNAL*, vol. 21, no. 2, p. 10, 2021.

[13] M. Zhu, M. Hassanalieragh, C. Zhuan, S. Fahad, AmalKai, and T. Soyata, “Energy-aware sensing in data-intensive fieldsystems using supercapacitor energy buffer,” *IEEE SENSORS JOURNAL*, vol. 18, p. 12, Apr. 2018.

[14] T. Shu, M. Xia, J. Chen, and C. de Silva, “An energy efficient adaptive sampling algorithm in a sensor network for automated water quality monitoring,” *MDPI*, vol. 1, p. 14, 2017.

[15] M. Inc., *Machine Learning with MATLAB*, vol. 1. MathWorks, 2020. Obtenido de https://la.mathworks.com/campaigns/offers/machine-learning-with-matlab.html.

[16] M. Inc., *Introducing Deep Learning with MATLAB*, vol. 1. MathWorks, 2021. Obtenido de https://la.mathworks.com/campaigns/offers/deep-learning-with-matlab.html.

[17] R. V. Zicari, *Explorations in Artificial Intelligence and Machine Learning*, vol. 1 of *1*. Taylor and Francis Group, 2021.

[18] E. V. Cuevas Jimenés, J. V. Osuna Enciso, D. A. Olivia Navarro, and M. A. Díaz Cortés, *Optimización. Algoritmos Programados con MATLAB*, vol. 11. primera ed., 2016. Decimá primera Reimpresion 2021.

[19] J. Vázquez-Castillo, A. Castillo-Atoche, J. Estrada-López, E. Osorio-de-la Rosa, G. Becerra-Nuñez, J. Heredia-Lozano, R. Atoche-Enseñat, and V. Sandoval-Curmina, "Energy-saving techniques for urban noise wsn with kalman-based state estimation and green facade energy harvester," *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, vol. 71, 2022.

[20] G. Berástegui Arbeloa and M. Galar Idoate, *Implementación del algoritmo de los k vecinos más cercanos (k-NN) y estimación del mejor valor local de k para su cálculo*. PhD thesis, Universidad Publica de Navarra, Pamplona, Mar. 2018.

[21] MathWorks, "Matlab." 2021, 1994-2021. shorturl.at/fAO78.