COMPONENTE DE INTERNET DE LAS COSAS PARA LA DETECCIÓN DE CAÍDAS EN ADULTOS MAYORES

Tesis que presenta
Rosendo Moncada Sánchez

Como requisito para obtener el grado de
Maestro en Sistemas Computacionales

Director de tesis:
M.C. Verónica Quintero Rosas.

Codirector de tesis:
Dr. Arnoldo Díaz Ramírez.

Mexicali, Baja California, México.
Enero 2022

Mexicali B.C., 08/Diciembre/2021
Oficio DEPI-84/2021
ASUNTO: Autorización de impresión

C. ROSENDO MONCADA SÁNCHEZ
PRESENTE

El que suscribe, Jefe de la División de Estudios de Posgrado e Investigación, comunica a usted que para la obtención del grado de Maestría en Sistemas Computacionales, se ha autorizado la reproducción de su trabajo de Tesis, cuyo título es:

Componente de Internet de las Cosas para la detección de caídas en adultos mayores

La autorización se emite con base a la revisión y dictamen emitido favorablemente y avalado con su rúbrica por los integrantes del Comité Tutorial, integrado por:

VERÓNICA QUINTERO ROSAS
Presidente

ARNOLDO DÍAZ RAMÍREZ
Secretario

JULIA DÍAZ ESCOBAR
Vocal

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
La tecnología para el bien de la humanidad®

ARNOLDO DÍAZ RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

S.E.P.
INSTITUTO TECNOLOGICO DE MEXICALI
DIVISION DE ESTUDIOS DE
POSGRADO E INVESTIGACION
MEXICALI, B.C.

ccp. Archivo

Agradecimientos

Quiero agradecer en gran medida a mis coordinadores de tésis, al maestro Arnoldo Díaz y a la maestra Verónica Quintero por apoyarme en este recorrido en el último par de años, su guía y amistad son muy valiosas para mí. Agradezco de la misma manera al Instituto Tecnológico de Mexicali por proveerme de grandes oportunidades de crecimiento y un espacio para desarrollarme academica y profesionalmente, tanto a nivel maestría como licenciatura, todas las becas y apoyos que recibí de parte de la institución fueron pieza clave para llegar hasta este punto. Agradezco al comité de posgrado por confiar en mi y aceptarme como estudiante de maestría. Agradezco enormemente a mi familia por su comprensión y ayuda. A mi Madre, mi Padre y mis hermanas, Maritza por tu amor y apoyo incondicional.

Índice general

Índice de figuras

Resumen

Las caídas en general resultan un problema de salud para diferentes sectores de la población, especialmente en adultos mayores y niños, repercuten en menor o mayor medida en distintos espectros del problema.

Éstos eventos de caídas representan una carga al sistema de salud pública de los países, ya que las afectaciones que se pudieran llegar a presentar pueden ser tan graves que se necesiten de más recursos para tener la capacidad de solventarlas.

Es por eso que la respuesta ante estos sucesos es de vital importancia para evitar un daño mayor en los accidentados, es importante orientar fuerzas de investigación para crear mecanismos que permitan detectar y/o prevenir estos sucesos no deseados.

En este documento se presentará un sistema no invasivo para la detección de caídas en adultos mayores basado en el paradigma del internet de las cosas.

Utilizando un novedoso sensor térmico infrarrojo, así como también un algoritmo determinístico de fácil implementación y de bajo coste computacional capaz de poder ser ejecutado en entornos embebidos. Obteniendo una tasa de detección que ronda el 89 % de presición incluso con menos sensores y de menor capacidad bajo las condiciones ideales. La solución propuesta cuenta con la característica de tener un dispositivo capaz de comunicarse mediante WiFi con un servicio de internet. Mismo que permite realizar alertas mediante un cliente de chat. Lo que permite generar un estado de alerta para el cuidador del paciente y así poder brindar asistencia lo más pronto posible.

Capítulo 1

Introducción

1.1. Descripción del problema

Las caídas son sucesos repentinos y no deseados que pueden llegar a grandes consecuencias incluso letales para los accidentados. Suponen un gran riesgo a su salud y a su vida. De esta manera la cantidad de dinero que se invierte en consecuencias causadas por caídas puede resultar considerable para el aparato de salud de los países. Países de los cuales, los mas afectados resultan ser los de bajo y medio desarrollo.

Dicho lo anterior podemos considerar a las caídas como un fenomeno que llega a afectar la vida de muchas personas de manera negativa.

Los adultos mayores son un grupo de personas, las cuales son más vulnerables ante las caídas, ya que debido a su edad pueden llegar a presentar dificultades motrices que terminen en accidentes de este tipo.

Por lo general los adultos mayores requieren de una atención especializada y aunque cuentan con cierta automía, es de vital importancia que estén con supervisión de una enfermera o de algún familiar.

Si un adulto mayor vive solo y no recibe asistencia de manera diaria, es posible que en los intervalos sin supervisión, el adulto mayor pueda sufrir de un accidente.

Ya que la asistencia diaria representa una carga de tiempo en los familiares del adulto mayor y/o también una carga económica si se desea contratar a una enfermera, hay que lidiar con la problemática de la supervisión del adulto mayor.

Si el adulto mayor es capaz de realizar sus tareas diarias sin asistencia pues los intervalos sin supervision pueden ser más prolongados. Y el caso contrario si el adulto es más dependiente en sus actividades.

El tiempo en el que el adulto mayor no recibe una atención, es un tiempo de incertidumbre ya que es posible que haya ocurrido un accidente, y de una u otra manera no nos podemos enterar, a menos que estemos físicamente presentes con el.

Es por eso que es de suma importancia poder supervisar el estado físico del

adulto mayor. Utilizando la tecnología a nuestro favor, es posible solventar los inconvenientes que presentan estas problemáticas. Al reducir la dependencia de la asistencia para el adulto mayor.

1.2. Estadísticas del problema

Según los datos de la OMS las caídas representan la segunda causa mundial de muerte por traumatismos involuntarios, justo por detrás de los choques automovilísticos. Se estima que ocurren anualmente 684 000 caídas mortales, de las cuales un 80 % de éstas se producen en los países de medio y bajo ingreso. De esta manera también llegan a ocurrir 37.3 millones de caídas de las cuales se requiere atención médica[26] (veáse figura 1.1).

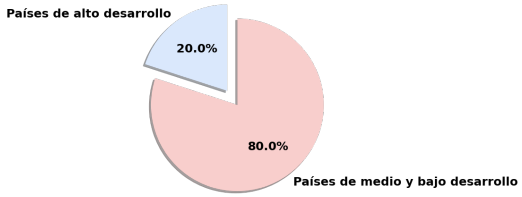

Figura 1.1: Distribución de porcentaje de muertes por caídas

Debido a estas caídas se pierden al año alrededor de 38 millones de años de vida ajustados en función de la discapacidad[1] (AVAD o DALY en inglés). De los cuales se estima que alrededor del 40 % de éstos años corresponde a niños (veáse figura 1.2).

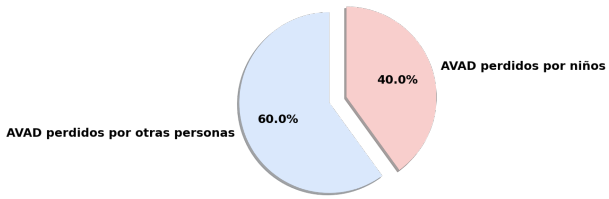

Figura 1.2: Distribución de años de vida ajustados en función de la discapacidad (AVAD) perdidos

Las caídas representan una carga para los sistemas de salud de los países del mundo. Carga la cual representa gastos para solventar los problemas causados por caídas en la población. Según la OMS, el costo promedio por traumatismo producido por caída cuesta alrededor de US$ 3,611 en Finlandia, mientras

[1] AVAD es una medida para años perdidos debido a enfermedades, discapacidades o muertes prematuras

que en Australia ronda los US$ 1,049. Así también la OMS indica que según datos provenientes de Canadá, la implementación de estrategias eficaces permitieron reducir un 20 % la incidencia de caídas en niños menores a 10 años. Esta reducción representó un ahorro neto que supera los US$ 120,000,000 al año.

1.3. Pregunta de investigación

¿De qué manera es posible reducir la complejidad y coste de un sistema de detección determinista de caídas basado en sensores térmicos, manteniendo una buena tasa de detección con respecto a sistemas similares? .

1.4. Objetivo general

Desarrollar un sistema de lazo cerrado determinista para detectar caídas en tiempo real que permita reducir la complejidad y coste de sistemas existentes basados en sensores térmicos.

1.5. Objetivos Específicos

- Elegir el sensor térmico más conveniente para la detección de caídas.
- Detectar posturas con el sensor térmico.
- Detectar parámetros del sensor que indiquen la caída de un paciente.
- Desarrollar el componente de IoT detector de caídas.
- Evaluar la tasa de detección del componente desarrollado.
- Publicar resultados de la investigación.

1.6. Requerimientos del sistema

Definir los requerimientos del sistema puede llegar a ser complejo, demasiados de estos extendería demasiado el proyecto, así que tenemos que plantearlos de la manera mas adecuada y concisa.

La solucion debe cumplir con los siguientes requerimientos:

- Detectar caídas
  - La solución debe contar con un mecanismo que permita conocer si un paciente se ha caído
- Utilizar un sensor NO invasivo
  - De esta manera el paciente no necesita estar al pendiente de equiparlo.

- Respetar la intimidad y privacidad del paciente.
    - Esto restringe la solución a que el sensor o sensores que se utilicen para la detección de caídas no pueda adquirir la suficiente información para invadir su privacidad.
- Generar alertas
    - Ya planteado el requerimiento de detectar caídas, es necesario que cuando ocurra una caída, la solución debe de contar con un mecanismo que le permita generar una alerta o un evento que se pueda notificar a uno o varios servicios y/o personas.

1.7. Organización de la Tesis

La Tesis se organizará de la siguiente manera:

En el Capítulo 2 se verá el trabajo relacionado a la investigación, en donde se mostrarán las diferentes soluciones que se han propuesto para la problemática. Posteriormente en el Capítulo 3 se verá la arquitectura del sistema además de mostrar el modelo matemático y una ejemplificación de su uso. Después se verá en el Capítulo 4 la evaluación de la propuesta y su implementación. Finalmente en el Capítulo 5 se expondran las conclusiones que resultaron de la investigación y de los trabajos futuros a realizar

Capítulo 2

Trabajos relacionados

Diversas investigaciones han sido realizadas y presentan múltiples soluciones que se han propuesto para este fín, estos sistemas se han clasificados en dos grandes vertientes, como se logra apreciar en la figura 2.1 .

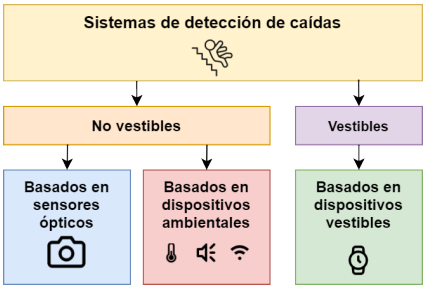

Figura 2.1: Clasificación de sistemas de detección de caídas

En la tabla 2.1 se encuentran trabajos relacionados a la detección de caídas en donde se pueden diferenciar en alg.

Estas dos grandes ramas son vestibles y no vestibles, los vestibles constan de dispositivos que se equipan en el paciente, y los no vestibles, aquellos dispositivos que no son equipados en el paciente y que se montan en un espacio a sensar. En esta última división se subdivide en dos tipos, la primera, que se basa en el uso de sensores ópticos para el análisis de imágenes y la restante en el que se utilizan uno o varios dispositivos sensores capaces de medir magnitudes del ambiente.

En los sistemas vestibles uno de los más utilizados es el de añadir un conjunto de sensores que nos ayuden a detectar la posición y aceleración de los pacientes.Los dispositivos basados en estos sensores de posición y aceleración son fáciles de miniaturizar dado que los sensores son muy pequeños, además de que el algoritmo para detectar caídas utilizando acelerómetros y/o giroscopios es de baja complejidad y es que hasta pueden utilizarse dispositivos embebidos

Artículo	**Detección**	**Exactitud**	**Presición**	**Sensibilidad**	**Especificidad**
Wickramasinghe et al	Tag(s) RFID		90 %		
Chen et al[2]	Tag(s) RFID				
Pierleoni et al[17]	Acelerómetro	90.37 %		80.74 %	100 %
Lee et al[10]	Acelerómetro		93.2 %		
Saadeh et al [19]	Acelerómetro			98.6 %	99.3 %
Santos et al [20]	Acelerómetro	99.92 %	100 %	99.45 %	100 %
Wu et al[28]	Acelerómetro			97.1 %	98.3 %
Ye et al [29]	Acelerómetro	90-100 %			
Rougier et al [18]	Cámara(s)			88 %	87.5 %
Mecocci et al [13]	Cámara(s)			62.4 - 80.3 %	92.5 - 97.7 %
Mashiyama et al [12]	Sensores térmicos	94.3-95.8 %			
Taramasco et al [23]	Sensores térmicos	93 %		93 %	93 %
Taniguchi et al [22]	Sensores térmicos	95.5 %			
Fan et al [7]	Sensores térmicos	95 - 100 %			
Sixsmith et al [21]	Sensores térmicos	30 %			
Wang et al [25]	Señales de WiFi		78 - 94 %		
Li et al [11]	Micrófono (s)	98 %		100 %	97 %
Díaz-Ramirez et al[3]	Micrófono (s)	83 - 90 %			
Khan et al[9]	Micrófono (s)			82 - 96 %	

Cuadro 2.1: Comparativa de trabajos relacionados

para el procesamiento de este algoritmo.

El inconveniente con este tipo de dispositivos está relacionado con que su uso no sea adecuado o incluso que no se use, el buen funcionamiento de los mismos depende de que el paciente esté dispuesto a vestirlos y de la manera correcta. Esto aunado a que los adultos mayores son propensos a olvidar equipar el dispositivo en su día o incluso por causa de enfermedades mentales, ejemplos estos sistemas se pueden encontrar en estas referencias: [5, 10, 14, 15, 17, 19, 28, 29].

También se han propuesto sistemas menos convencionales que incluyen sensores en las suelas de los zapatos, que mediante tecnología RFID se comunican con tags que están en el suelo. Este tipo de sistema proporciona una gran información al brindar un mapa por donde el paciente ha transitado. Pero un gran inconveniente con este tipo de dispositivos es que se necesita una infraestructura compleja para que se trace el piso de la vivienda en un programa que reconozca la posición del paciente. Aunado a esto también existe el inconveniente de utilizar zapatos que están especialmente diseñados para ser detectados por este suelo. Ejemplos para estos sistemas se han trabajado en: [2, 27].

El uso de sensores ópticos en conjunto con herramientas de procesamiento de imágenes es posible detectar con una gran precisión eventos de caídas, de esta manera es posible saber mucha más información que la caída en cuestión. El uso de cámaras puede resultar contraproducente para la detección de caídas, invadiendo la intimidad de los individuos que estén en el foco del sensor, incluso si de antemano están conscientes de su propósito.

El análisis de imágenes representa un gran costo computacional, además son necesarios algoritmos mas complejos, a causa de esto se requiere de procesadores más potentes y/o específicos para realizar este análisis de imágenes. Esto eleva el costo en general al requerir de procesadores más potentes además de requerir más energía y posiblemente más tiempo de procesamiento, o incluso elevar la complejidad al poder requerirse de estaciones externas de procesamiento o de computación en la nube. Este tipo de sistemas se puede ver con más detalle en: [1, 4, 18, 13, 16].

El uso de sensores para medir el ambiente puede llegar a ser una opción viable ya que de esta manera no es necesario equiparse en el paciente, además que muchos de estos no alcanzan a obtener la suficiente información como para invadir la privacidad de los pacientes. La capacidad de los sensores de percibir la realidad y transformarla en información procesable es amplia y diversa, en estos casos se han explorado el uso de micrófonos [3, 9, 11], arreglos de sensores térmicos [7, 12, 21, 22, 23] y en otras instancias el análisis de redes de Wi-Fi [24, 25]. En toda la diversidad de sensores se han logrado detectar los eventos de caídas y su precisión se basa en la capacidad de los sensores mismos así como su uso y de la implementación de software que se encarga de procesar esta información.

Capítulo 3

Arquitectura del sistema propuesto

3.1. Arquitectura general

Se propone una arquitectura en capas que se puede apreciar en la figura 3.1 que pueda facilitar el diseño e implementación de estos sistemas. Esta arquitectura se propone con el fin de establecer un patrón que pueda ser seguido y a su vez en segmentar y definir las tareas que conforman cada una de las capas de un sistema para la elaboracion detección de caídas y que se detallarán adelante de forma ascendente.

Capa Monitorización

En esta conforman aquellos dispositivos que puedan medir magnitudes físicas y transformarlas en información legible y procesable por sistemas digitales, en esta capa pueden o no estar incluidos microcontroladores que colecten estos

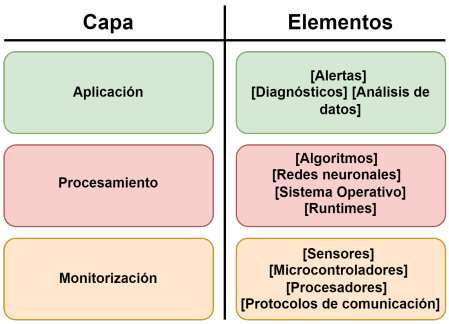

Figura 3.1: Arquitectura propuesta

datos de uno o varios dispositivos, a su vez en este conforman los protocolos de comunicación que emplean estos dispositivos.

Capa procesamiento

En esta capa se realiza todo el procesamiento de las señales provenientes de los dispositivos de sensado, y es el responsable de determinar el estado de la persona en base a las mediciones de la capa de monitorización. Esta capa comprende al dispositivo en cuestión que realiza el procesamiento, el algoritmo o inteligencia artificial a implementar, así como otros componentes importantes como el sistema operativo, librerías de ejecución, controladores, entre otros.

Capa aplicación

Consta del manejo y uso de los datos resultantes obtenidos a partir de la información procesada en la capa de procesamiento, aquí es donde la información se puede utilizar. En primera instancia para poder hacer alertas mediante servicios de internet y/o telefonía a parientes cercanos al paciente y/o a su cuidador. En segundo lugar para análisis de los eventos de caídas del paciente, cuántos falsos positivos resultaron. Para que de esta manera poder generar un histórico para los pacientes.

De esta manera se puede generar información útil que defina los aspectos a mejorar en la creación de los algoritmos, el uso y acomodo de los dispositivos.

3.2. Modelo matemático

El objetivo del modelo es determinar si una persona se ha caído, mediante una serie de matrices de dos dimensiones cuyos elementos representan temperaturas y representan un mapa de calor, de este modo se definen los elementos que conforman el modelo matemático:

Muestra

Una muestra consiste en una matriz de mediciones de temperatura y el orden de la matriz pertenece al orden de:

$$S \in R^{m,n}$$

Cada elemento de la matriz representa una temperatura en un sector particular del sensor térmico en un tiempo determinado.

Número de muestra

Consiste en una sucesión infinita.

$$\tau_n = 1, 2, 3, 4, ..., n$$

Esta sucesión representa la secuencia de las muestras que se toman cada cierto intervalo.

Ventana de muestras

Consta del número de muestras que se tomarán para determinar en ese espacio de muestras si alguien se ha caído, será definido como W.

Temperatura de elemento en una muestra

Consiste en la temperatura de una muestra un sector determinado, tenemos:

$$T_\tau(x, y) \tag{3.1}$$

En el cual:

$$(x, y) \in R^{m,n}$$

x e y representan la ubicación del sector, τ representa el número de muestra. El resultado de esta función consiste en la temperatura de la muestra τ en la posición (x,y).

Diferencia de temperatura

Teniendo una colección de muestras en la cual cada uno de estas muestras también consta de una colección de sectores en un tiempo τ podemos decir que delta de la temperatura consiste en la diferencia entre la temperatura de hace W muestras contra la muestra más reciente de un sector en particular.

$$\Delta T_\tau(x, y) = T_\tau(x, y) - T_{\tau-w}(x, y) \tag{3.2}$$

Zona

Una zona consiste en un conjunto de tuplas que pertenecen a $\mathrm{R}^{\mathrm{m,n}}$ y describen la unión de multiples sectores tal que una zona Z:

$$Z = z_i \in R^{m,n}$$

Temperatura promedio de zona

Consiste en el promedio de temperaturas registradas en un tiempo τ en una determinada zona utilizando la ecuación 3.1. Está definido como:

$$AT_\tau(Z) = \frac{\sum_{i=1}^{|Z|} T_\tau(Z_i)}{|Z|} \tag{3.3}$$

Esta definición esta inspirada en las ecuaciones de Taniguchi [22].

Diferencia promedio de zona

Consiste en el promedio de las diferencias de temperatura. Para esta definición se utiliza la ecuación 3.2

$$A\Delta_\tau(Z) = \frac{\sum_{i=1}^{|Z|} \Delta T_\tau(Z_i)}{|Z|} \tag{3.4}$$

Esta definición esta inspirada en las ecuaciones de Taniguchi [22].

Transición

Teniendo un conjunto de muestras, llamaremos a una transición al evento en el cual, la temperatura de una zona en un tiempo determinado pasa a estar en otra zona en otro tiempo, es decir que la fuente de calor (persona) paso de estar en una zona y paso a estar en otra.

Cabe destacar que esto por si solo no detecta si una persona se cayó o no, el concepto general es la transición que hace un objeto que emite calor a traves de los sectores del sensor, para esto se describirán elementos adicionales que hacen uso de los elementos ya descritos.

Caída

Ya mencionamos que una transición por si sola no representa una caída, de manera sencilla una caída es una es una transición con contexto, podemos definir la zona de inicio como el lugar donde estará una persona antes de caerse, y una zona final en donde estara el sujeto al final de la caída.

3.2.1. Determinar una caída

Con los conceptos anteriormente mencionados podemos describir el como se detectará una caída, a continuación se enlistarán los parámetros de entrada, los cálculos de unas variables y de las condiciones para determinar si ha ocurrido un evento de caída.

Parámetros de entrada

Los parámetros de entrada constan de las entradas que debe de recibir el modelo para poder determinar el estado de caída, se enlistan a continuación:

- **Zona inicial (IZ):**

  Consiste en un conjunto de elementos que conforman la zona de la cual sale la persona, es decir cuando esta erguida o de pie.

- **Zona final (FZ):**

  Consiste en el conjunto de elementos que conforman la zona en donde terminará la persona reposando al final de la caída.

- **Zona referencia (RZ):**

  Consiste en un conjunto de elementos que conforma la zona que se utilizará para determinar la referencia para el modelo.

- **Tamaño de ventana (W):**

  Es la cantidad de muestras que se utilizarán para determinar la caída.

- **Límite promedio calor (AHT):**

  Utilizado para determinar una condición (Ver condición 1). Este parámetro nos permite comenzar a detectar la transición, filtrando aquellas muestras en las que la zona final no es más caliente que la zona de referencia por ésta cantidad de grados especificada.

- **Límite porcentaje transferencia calor (HPT):**

  Utilizada para determinar una condición (ver condición 4). Este parámetro dicta dentro de que rango de porcentaje de calor se transfiere de la zona inicial a la zona final.

- **Límite diferencia promedio calor (AHD):**

  Utilizada para determinar una condición (ver condición 2). Este parámetro indica el mínimo de transferencia de calor que se tiene que detectar, este parámetro existe porque puede ocurrir que la zona final se caliente muy poco y a su vez la zona inicial se enfrie un poco también, estamos hablando del orden de las décimas de grados.

Cálculo de valores para determinar el estado de la caída

Estos valores se calculan a partir de las entradas anteriormente descritas, se enumeran a continuación:

- **Promedio zona inicial (IZA):**

  Es el promedio de temperatura de la zona inicial en τ siendo de esta manera. Utilizando la ecuación 3.3, queda de la siguiente manera:

$$IZA = AT_\tau(IZ) = \frac{\sum_{i=1}^{|IZ|} T_\tau(IZ_i)}{|IZ|}$$

- **Promedio delta zona inicial (IZDA):**

  Es el promedio de diferencia de la temperatura en la zona inicial en la muestra τ siendo así. Utilizando la ecuación 3.4, se define de la siguiente manera:

$$IZDA = A\Delta_\tau(IZ) = \frac{\sum_{i=1}^{|IZ|} \Delta T_\tau(IZ_i)}{|IZ|}$$

- **Promedio zona final (FZA):**

  Es el promedio de temperatura de la zona final en τ siendo de esta manera. Utilizando la ecuación 3.3, queda de la siguiente manera:

$$FZA = AT_\tau(FZ) = \frac{\sum_{i=1}^{|FZ|} T_\tau(FZ)}{|FZ|}$$

- **Promedio delta zona inicial (FZDA):**

  Es el promedio de diferencia de la temperatura en la zona final en la muestra τ siendo así. Utilizando la ecuación 3.4, de esta manera:

$$FZDA = A\Delta_\tau(FZ) = \frac{\sum_{i=1}^{|FZ|} \Delta T_\tau(FZ_i)}{|FZ|}$$

- **Promedio zona referencia (RZA):**

  Es el promedio de temperatura de la zona inicial en τ siendo de esta manera. Utilizando la ecuación 3.3, se define de la siguiente manera:

$$RZA = AT_\tau(RZ) = \frac{\sum_{i=1}^{|RZ|} T_\tau(RZ_i)}{|RZ|}$$

- **Promedio delta zona referencia (RZDA):**

  Es el promedio de diferencia de la temperatura en la zona inicial en la muestra τ siendo así .Utilizando la ecuación 3.4, queda de la siguiente manera:

$$RZDA = A\Delta_\tau(RZ) = \frac{\sum_{i=1}^{|RZ|} \Delta T_\tau(RZ_i)}{|RZ|}$$

- **Porcentaje de transferencia (HTP) :**

  Es el porcentaje de calor que se paso de una zona hacia otra, siendo así:

$$HTP = \frac{IZDA}{RZDA}$$

- **Porcentaje de transferencia absoluto (AHTP):**

  Es el valor absoluto de HTP, siendo así:

$$AHTP = |HTP|$$

Condiciones para determinar el estado de la caída

Ya una vez con los parámetros y los valores numéricos definidos podemos determinar una caída en base a un conjunto de condiciones, se muestran separadas ya que juntas serían muy complicadas de explicar así que las describiremos en cuatro partes. Cada condición permite evaluar diferentes cualidades de los datos de entrada para determinar si hubo o no una transición. A continuación se enlistan:

- **Condición 1 (T1):**

  Esta condición evalúa si la zona final tiene más temperatura que la zona inicial, utilizando la temperatura de la zona de referencia, así de esta manera queda la condición:

$$T1 = IZA < (RZA + AHT) \bigwedge (RZA + AHT) < FZA$$

- **Condición 2 (T2):**

  Esta condición evalúa si la diferencia de temperatura es mayor a lo especificado en el parámetro AHD, este parámetro se utiliza para filtrar lecturas cuando se ha transferido menos que el parámetro AHD, el cual queda de la siguiente manera:

$$T2 = (IZDA <= -AHD) \bigwedge (AHD <= FZDA)$$

- **Condición 3 (T3):**

  Esta condición evalúa si el porcentaje de transferencia de calor (HTP) es negativo, ya que evalúa si la zona final esta relativamente más caliente y la zona de inicio que este relativamente más fria, ya que si las dos zonas son relativamente calientes HTP sera positivo de esta manera, queda de la siguiente manera:

$$T3 = HTP < 0$$

- **Condición (T4):**

  Esta condición evalúa, después de haber comprobado que la transferencia de temperatura se hizo desde la zona inicial hacia la zona final, lo siguiente a saber es que tanta temperatura se paso de un lado hacia el otro.

  Teniendo AHTP, si este ronda en la unidad podemos decir que el 100 % de la temperatura que estaba en la zona inicial esta ahora en la zona final, de este modo utilizando el parámetro HPT podemos determinar que tanto sera la holgura hacia arriba y hacia abajo a partir del 100 %.

  Es decir que si tenemos una olgura del 10 % se evaluara si entre el 90 % y 110 % de la temperatura se transfirió de la zona inicial hacia la zona final, y así sustantivamente, quedando de la siguiente manera:

$$T4 = (1 - HPT) <= AHTP \bigwedge AHTP <= (1 + HPT)$$

De esta manera ya definidos todos los elementos que conforman el modelo matemático podemos determinar el estado de la caída en un tiempo τ de la siguiente manera

$$F = T1 \bigwedge T2 \bigwedge T3 \bigwedge T4$$

Así que si todas las condiciones anteriormente mencionadas se cumplen entonces ocurrio una transición de la zona de inicio hacia la zona final.

3.2.2. Ejemplificación del modelo en un caso

Para ver de una manera más sencilla en qué consta el algoritmo, se creó un notebook en jupyter notebook que nos permitió analizar de manera gráfica un evento de caída.

Figura 3.2: Zonas de inicio y fin utilizadas

En la figura 3.2 se logra apreciar que se utilizan 4 columnas del sensor como parámetro de zonas iniciales, con 3 renglones cada una y el renglón restante se utiliza como zona final, abarcando las 4 columnas, de esta manera se analiza desde una perspectiva lateral una caída.

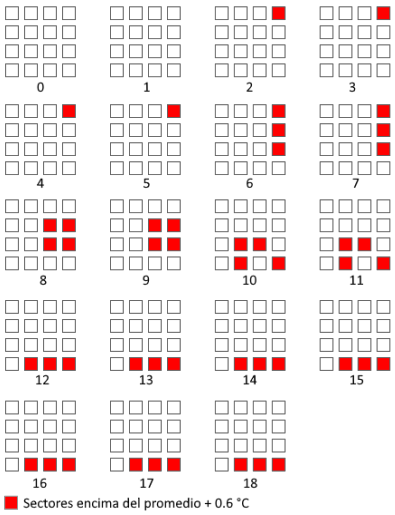

Figura 3.3: Contenido de un archivo de caída

En la figura 3.3 podemos apreciar el contenido de un archivo de muestra en donde se encuentra una caída, esta simplificado con el fin de que se aprecie de mejor manera. En la que los sectores en rojo son más calientes que el promedio de todos más 0.6°C. De esta manera podemos ubicar bien donde se encuentra una persona. Podemos ver cuadro a cuadro como sucede un evento de caída, en los primeros dos cuadros aun la persona no es detectada.

En los siguientes 4 cuadros apenas la parte superior se alcanza a detectar, en los siguientes dos cuadros, el 6 y 7 se detecta a la persona la cual esta

erguida. Desde el cuadro 8 hasta el 11 se puede apreciar como ya la persona no esta erguida y esta en transición de caerse, para finalmente quedar reposando a partir del cuadro 12 hasta el final del archivo.

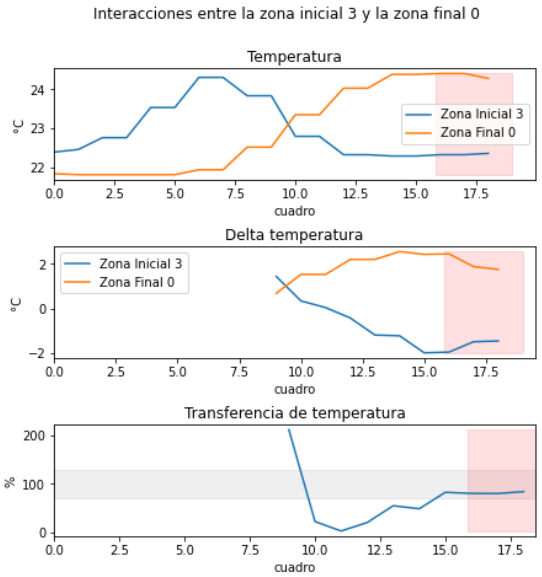

Figura 3.4: Comportamiento de las variables en un archivo de caída

En la figura 3.4 podemos apreciar tres componentes en los cuales podemos ver un comportamiento, esto es en relación al archivo de la figura 3.3. En esta figura se puede apreciar las interacciones entre la zona inicial 3 y en la zona final 0, en la primera gráfica se puede apreciar la temperatura de ambas zonas, la cual es similar, a partir del cuadro 3 empezamos a notar un incremento en la zona inicial, y la zona final permanece con su temperatura promedio.

Para el cuadro 6 y 7 la zona inicial registra su mayor temperatura durante todo el archivo y la temperatura de la zona final aumenta prácticamente nada. A partir de este cuadro es cuando comienza la caída, se puede observar que la temperatura de la zona inicial esta bajando a lo largo de los cuadros 8 - 12 y permanece baja hasta el final de la muestra, por el otro lado la zona final incrementa su temperatura al mismo nivel al que estaba la zona inicial apenas algunos cuadros atras.

También se puede apreciar en la figura 3.4 el comportamiento de delta de temperatura, que es la diferencia de temperatura con W cuadros hacia atrás, en este caso son 9. Esta medición solo puede existir a partir del cuadro 9, el cual al principio tanto para la zona inicial como para la zona final ronda la unidad, dado que eso aumentaron de temperatura.

Llegando al cuadro 15 que es cuando se detecta la caída, se puede apreciar que la zona final se calentó casi en la misma proporción en la que la zona inicial

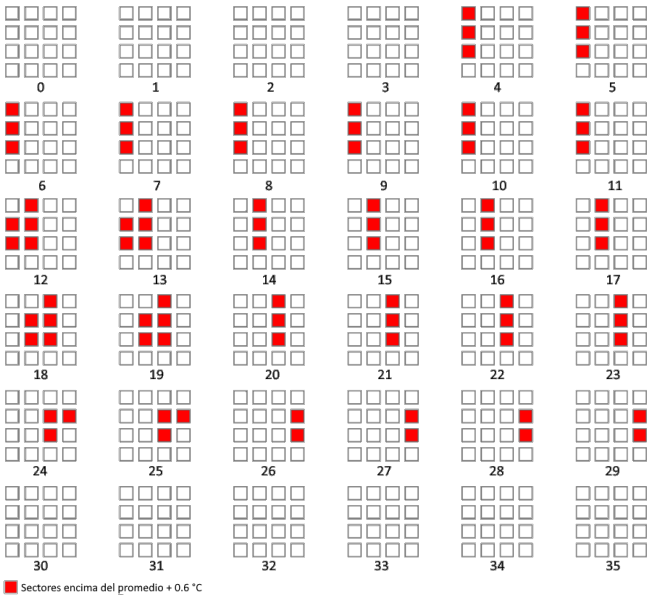

Figura 3.5: Contenido de un archivo de no caída

se enfrió. En la tercera gráfica de la figura 3.4 se muestra la proporción que existe entre el delta de temperatura de la zona inicial contra el delta de temperatura de la zona final, en terminos absolutos. El área gris de la gráfica representa el umbral del rango de transferencia de temperatura, el cual para este caso se uso el 35 % , de esta manera si la proporción que existe entre los delta de temperatura de estas dos zonas está entre el 65 % y el 135 % se considera que ocurrió una caída (junto con las demás condiciones).

El umbral de rango de detección es importante ya que si se incrementa mucho podrían generarse muchos falsos positivos, por el contrario si es muy pequeño podrían no detectarse eventos de caídas que realmente sucedieron. Dado a que en la práctica la temperatura que existe en una zona no se transfiere exactamente a la otra, esta es la razón de este parámetro.

Por el otro lado se puede apreciar en la figura 3.5 el contenido de un archivo que contiene un evento de no caída en la que una persona camina de izquierda a derecha, como se muestra, en ningún momento la zona final que se estaba analizando no calentó de manera considerable.

Analizando la figura 3.6, la cual contiene las interacciones de todas las zonas iniciales y la zona final para no ocupar mucho espacio. No hay que perder de vista la linea de color morado, que es la que representa la temperatura y delta de la temperatura de la zona final.

Como pudimos apreciar en la figura 3.4 hay una clara relación de las temperaturas y los delta de temperatura de las zonas que presentan una transición. En el caso de no caída podemos ver que la zona final mantiene una temperatura

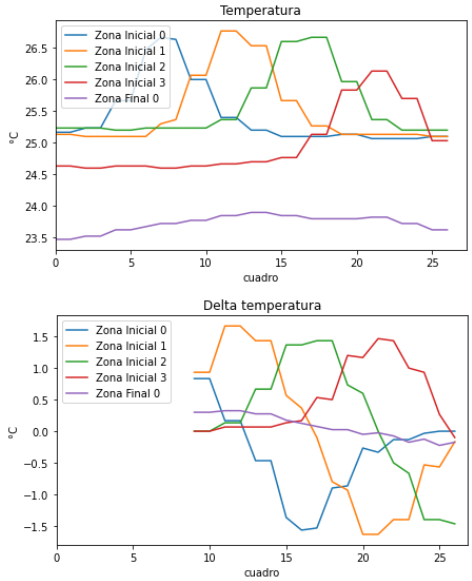

Figura 3.6: Comportamiento del evento de no caída

relativamente estable con respecto a los demás, así como el delta de temperatura, el cual no excede de la unidad ni positiva ni negativa. De esta manera podemos ver de una manera más gráfica el comportamiento que se trata de clasificar mediante la medición de las temperaturas.

Capítulo 4

Evaluación y resultados

4.1. Evaluación de la propuesta

Para la muestra de resultados se realizaron dos prototipos cuyo proposito es demostrar la tasa de detección del algoritmo y también demostrar la viabilidad del sistema de detección, los cuales se veran a continuación en las siguientes dos subsecciones

4.1.1. Implementación del modelo en un algoritmo

En el algoritmo 4.1 se puede apreciar la implementación que se realizó en el lenguaje de programación C#, mismo que se implementó en el programa para seccionar archivos. Dado a que es más simple solo esta implementación se agregará al escrito. El microcontrolador cuenta con la implementación de este mismo algoritmo pero en lenguaje C. Dadas las características del lenguaje se vuelve muy complicado de leer y explicar. Esta es la razón por la cual se presenta esta implementación del modelo.

4.1.2. Recolección de datos y análisis de la tasa de detección

Para la recolección de datos se utilizó un microcontrolador tipo Arduino que nos permite mediante el puerto i2c[1] con el que cuenta, leer las temperaturas del sensor Omron D6T-44L-06, este se conecta a una fuente de voltaje y se conecta al puerto i2c del microcontrolador.

[1] i2c Es un bus de comunicación serial que permite comunicar dos circuitos en distancias cortas

Algoritmo 4.1 Implementación del modelo matemático en lenguaje C#

```
private bool AnalizarLectura
(
  //ZONA INICIAL
  int[] IZ,
  //ZONA FINAL
  int[] FZ,
  //ZONA REFERENCIA
  int[] RZ,
  //LIMITE PORCENTAJE TRANSFERENCIA CALOR
  float HPT = 0.20f,
  //LIMITE PROMEDIO CALOR
  float AHT = 1.0f,
  //LIMITE DIFERENCIA PROMEDIO CALOR
  float AHD = 0.4f
){
  /*
  EN LOS METODOS 'PromedioDeltaZona' y 'PromedioTemperaturaZona'
  está incluido el parametro W de tamaño de ventana, por eso no
  se encuentra en los parametros de entrada de este algoritmo
  */

  //PROMEDIO DELTA ZONA REFERENCIA
  float RZDA = PromedioDeltaZona(RZ);

  //PROMEDIO DELTA ZONA INICIAL
  float IZDA = PromedioDeltaZona(IZ);

  //PROMEDIO DELTA ZONA FINAL
  float FZDA = PromedioDeltaZona(FZ);

  //PROMEDIO TEMPERATURA ZONA REFERENCIA
  float RZA = PromedioTemperaturaZona(RZ);

    //PROMEDIO TEMPERATURA ZONA INICIAL
  float IZA = PromedioTemperaturaZona(IZ);

  //PROMEDIO TEMPERATURA ZONA FINAL
  float FZA = PromedioTemperaturaZona(FZ);

  //PORCENTAJE DE TRANSFERENCIA DE CALOR
  float HTP = IZDA / (FZDA == 0 ? 0.0001f : FZDA);

  //PORCENTAJE DE TRANSFERENCIA DE CALOR ABSOLUTO
  float AHTP = Math.Abs(HTP);

  //CONDICION COMPUESTA
  return
    (IZA < (RZA + AHT) && (RZA + AHT) < FZA) &&   //CONDICION 1
    (IZDA <= (-AHD) && FZDA >= AHD) &&            //CONDICION 2
    (HTP < 0) &&                                  //CONDICION 3
    (1.0f - HPT) <= AHTP && AHTP <= (1.0f + HPT));//CONDICION 4
}
```

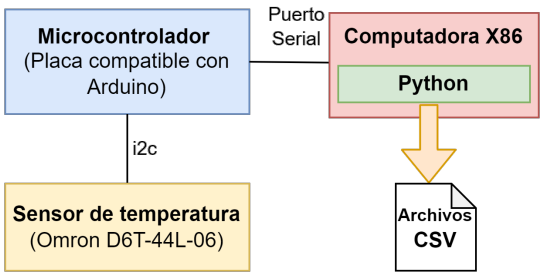

Figura 4.1: Diagrama de prototipo para realizar lecturas

El sensor ya cuenta con un mecanismo que filtra lo que se ve a traves del lente del sensor, lo que nos ahorra procesamiento. El sensor manda un conjunto de bytes que se transforman en una serie de enteros que representan la temperatura, dependiendo del sensor la lectura sera más o menos extensa y el método de conversión podría variar.

En una computadora convencional x86[2] se programó un script en python que se encarga de comunicarse con la placa de desarrollo mediante puerto serial y guardar las temperaturas que se obtuvieron en multiples archivos CSV[3], grabando una muestra por renglón en el cual estaba escrito el timestamp en el que se capturó la muestra y las 16 temperaturas correspondientes a la lectura del sensor.

En la figura 4.1 se puede apreciar los elementos que conforman el prototipo que permitió capturar las muestras para el análisis. Se crearon múltiples archivos dependiendo de la forma de la caída (de lado, de frente, en una silla, etc.) y eventos de no caídas (caminando, recogiendo, sentandose, etc.) en archivos con centenas de renglones cada uno (veáse figura 4.2).

Para analizar de manera más efectiva los datos, era necesario poder clasificarlas de tal manera que se seccionaron en archivos más pequeños que constaban de un solo evento de caída o no caída. Esto se realizo con ayuda de un programa que se escribió en C# en la plataforma Windows Forms en .NetFramework, con ayuda de Visual Studio IDE[4] para su desarrollo (vease figura 4.3) .

En la sección de Anexos se puede acceder al repositorio donde se encuentra el programa que se utilizó para la visualización y seccionado de archivos.

El propósito de este programa fue el de poder visualizar los grandes archivos de manera cómoda y realizando una inspección manual se separaron todos los archivos de muestras. Se plasmó un archivo CSV adicional el cual contiene una tabla con los nombres de los archivos y las etiquetas que le correspondían, las etiquetas fueron dos, si existía una caída o no en el archivo, además del nombre del escenario que ocurrio en dicho archivo.

[2] x86 es una familia de set de instrucciones, la cual es actualmente la mas utilizada en computadoras de escritorio y laptops

[3] Un archivo CSV es un archivo plano en el cual: imitando a una tabla cada renglon representa un registro y cada columna esta delimitada por una coma

[4] IDE Significa entorno de desarrollo integrado

Al final de todo este proceso resultaron 237 archivos que describen caídas en diferentes escenarios, así como 246 archivos que contienen eventos de no caída, también en diferentes escenarios. Una vez separados y clasificados todas las muestras será más fácil determinar la tasa de detección del algoritmo. Para el análisis de estos nuevos archivos se utilizó jupyter notebook[5], en dónde se implementó el algoritmo de detección, de tal manera que se puede recorrer cada archivo de manera individual para el analisis.

De este análisis se obtuvieron las medidas de evaluación del algoritmo que se discutirán mas adelante en la sección de resultados. Además gracias a Jupyter Notebook es que se pudieron analizar y graficar las figuras que se encuentran en la Subsección **Ejemplificación del modelo en un caso.**

En la sección de Anexos se encuentra la dirección web del repositorio en donde se puede encontrar tanto los archivos muestra, así como los notebooks que se utilizaron para esta investigación.

Figura 4.2: Ejemplo de captura de muestra por el sensor infrarrojo

[5]Jupyter Notebook es un entorno interactivo web para python en donde se pueden desarrollar scripts de este mismo lenguaje

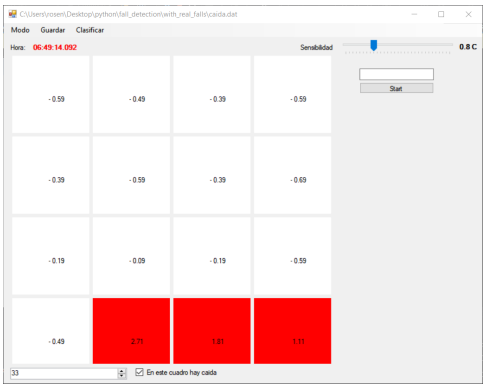

Figura 4.3: Programa para seccionar archivos en muestras

4.1.3. Implementación de prototipo de sistema basado en IoT

Para la demostración del funcionamiento del sistema se realizó otro prototipo con la función de leer los datos del sensor de temperatura y este se conectará a internet a través del router. En la figura 4.4 es donde podemos apreciar el esquema general de la solución propuesta.

El microcontrolador tiene la capacidad para procesar el algoritmo, de esta manera manda una alerta a un servicio de internet mediante el uso de web-socket[6]. El servicio de alertas podría ser desplegado en una computadora local así como en una computadora remota en un servicio de la nube. Este servicio podría interactuar a su vez con otros servicios web que permitan enviar SMS, mensajes de chat, Correo, entre otros. Para este caso en particular se utilizo un servicio que nos permite mandar mensajes por Telegram[7].

Dado a que la cantidad de información es muy poca, se puede mandar en tiempo real la información sobre las temperaturas que se estan leyendo, el web-socket persiste mientras el dispositivo este encendido.

[6] Websocket es un protocolo de comunicación que permite comunicación de doble sentido entre el cliente y el servidor con una sola conexion TCP

[7] Telegram es un cliente de chat en linea muy similar a otros como Whatsapp o Facebook Messenger

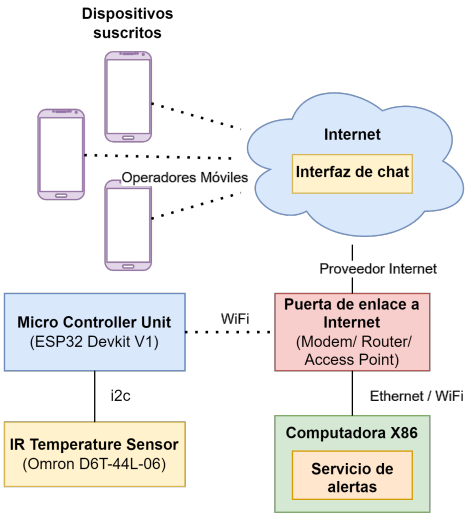

Figura 4.4: Esquema del sistema de detección de caídas

Entrando más en detalle se utilizó un sensor Omron D6T-44L es un sensor térmico infrarrojo con una resolución de 4x4 y un área de detección de hasta 2.5x2.5 metros a una distancia de 3 metros. Este sensor es conectado a un microcontrolador mediante i2c (se uso un logic level shifter para diferencia de voltaje entre el sensor y el microcontrolador).

La placa de desarrollo empleada es un ESP32 Devkit V1, es una placa de desarrollo muy popular y el chip que incorpora es uno muy robusto para aplicaciones de IoT. Este microcontrolador incluso se utiliza en cientos de aparatos ya comercializados, incorpora WiFi y Bluetooth lo que lo hace ideal para el caso de uso, además de soportar todos o la mayoría de protocolos para el intercambio de datos.

Se utilizaron las herramientas que provee espressif, la compañía que diseña la serie de microcontroladores ESP, la cual provee un conjunto de herramientas para el desarrollo de aplicaciones, llamado ESP-IDF (ESP IoT Development Framework).

ESP-IDF es un entorno completo que permite compilar archivos binarios que contienen el sistema operativo así como la aplicación desarrollada, para posteriormente inyectarlos en el microcontrolador. El sistema operativo que se incluye en la compilación del binario es una variante FreeRTOS, un sistema operativo de tiempo real, especialmente diseñado para microcontroladores ya que ocupa muy poco espacio.

Para reafirmar la viabilidad del proyecto como un componente de IoT, se diseñó un dispositivo en base a módulos comerciales que cumplen con las diferentes funciones del dispositivo. A continuación se muestra un diagrama en la figura 4.5 que muestra los elementos del prototipo y de como estan conectados, para facilidad se simplifican algunas conexiones.

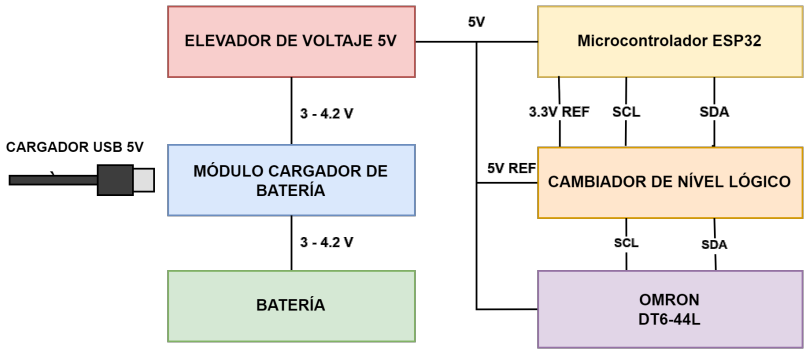

Figura 4.5: Diagrama simplificado de la conexión de los módulos (con batería)

Cada uno de los elementos que conforman el prototipo son elementales para el funcionamiento, aunque si se desea que no funcione con baterías se pueden prescindir de algunos elementos. Comenzando con la batería, es una batería de litio, con un voltaje nominal de 3.7 voltios, dependiendo de la carga, puede suplir desde 3.0 hasta 4.2 voltios.

El dispositivo podría funcionar con 4.2 voltios pero no con 3.0 voltios, así que necesitamos mantener el voltaje constante, es cuando utilizamos un módulo elevador de voltaje. Este elevador de voltaje es el que incrementa el voltaje de la batería hasta 5 voltios, independientemente de la carga de la batería.

Hablando de la carga de la batería, tenemos un módulo que se encarga de cargar y proteger la batería, basado en el circuito integrado TP4056 este módulo se encarga de recibir un voltaje de carga de 5 voltios para poder recargar la energía de la batería. Además este módulo se encarga de que proteger a la batería de sobrecargas, así como de cortar el suministro de la batería hacia el resto del circuito si el voltaje de ésta se encuentra por debajo del límite inferior de carga.

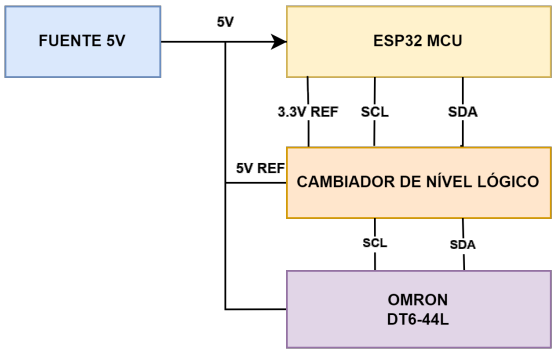

Figura 4.6: Diagrama simplificado de la conexión de los módulos (sin batería)

A partir de aquí es donde se muestra lo realmente escencial del dispositivo, si se desea usar sin batería se puede optar por omitirlo, en la figura 4.6 se puede apreciar el diagrama de conexión sin batería. Cabe destacar que de esta manera el dispositivo estará limitado a que siempre esté conectado a una fuente de energía que provea 5 voltios.

Comenzando por el componente principal que es el sensor Omron D6T-44L, que ya fue descrito anteriormente, este se encarga de sensar el ambiente y de transformarlo en información digital y transmitirla mediante i2c al microcontrolador. El ESP32 que también fue descrito anteriormente, es un microcontrolador orientado a IoT y que se encarga del procesamiento de la información recibida del sensor térmico.

El microcontrolador también tiene la tarea de comunicarse inalámbricamente con el servicio de alertas, ya sea de manera local o en la nube. El sensor térmico y el microcontrolador operan a diferentes niveles lógicos, el ESP32 funciona principalmente con 3.3 voltios y el D6T-44L Opera a 5 voltios.

Esta diferencia podría dejar inutilizable al microcontrolador, ya que no esta preparado para recibir ese voltaje. El cambiador de nivel lógico se encarga de ser un intermediario entre estos dos elementos, de esta manera ambos pueden comunicarse como si estuvieran operando con el mismo nivel lógico.

Figura 4.7: Placa del prototipo (vista frontal)

Recorriendo la figura 4.7 de izquierda a derecha, tenemos en la parte superior el cambiador de nivel lógico, en la parte inferior el microcontrolador ESP32. En la parte central tenemos el sensor térmico, y detrás de éste la batería de litio, y por último en la parte derecha tenemos el elevador de voltaje.

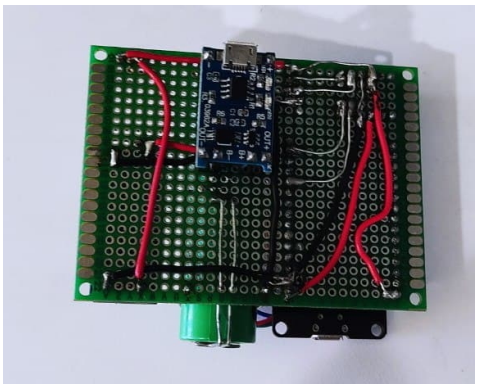

Figura 4.8: Placa del prototipo (vista trasera)

En la figura 4.8 se puede apreciar parte trasera del prototipo, en la que se encuentra la mayoría de las conexiones que estan en la parte frontal del dispositivo, además se encuentra el módulo de carga para la batería. Dentro del microcontrolador es donde se encuentra una buena parte del sistema que consiste en el sistema operativo y la aplicación que permite que se puedan detectar las caídas.

El sistema operativo que ejecuta el dispositivo es FreeRTOS, el cual esta escrito en C y es altamente robusto, tiene un gran respaldo de multiples compañías de gran renombre como ARM, Mediatek, Texas Instruments entre muchos otros incluyendo la misma espressif[8].

Esto nos asegura un buen soporte al sistema operativo a lo largo del tiempo además de que la documentación [6] esta muy bien elaborada y facilita el diseño de aplicaciones en gran medida , además de contar con una gran comunidad de desarrolladores. Un sistema operativo de tiempo real es un sistema operativo cuyo propósito es ejecutar aplicaciones de tiempo real, las aplicaciones de este tipo son aquellas que estan restringidas por límites de tiempo. De no cumplir con las restricciones de tiempo se podría provocar una falla en el sistema, para nuestro caso, la toma de muestras debe ser periodica y constante.

El procesador del ESP32 cuenta con dos núcleos, por lo cual podemos tener dos hilos de ejecución concurrentes, para la aplicación se utilizó un modelo productor-consumidor. En la figura 4.9 se puede apreciar un esquema general de como interactuan los hilos de ejecución y de como se procesa la informacion en el microcontrolador.

En la sección de Anexos se puede acceder al repositorio donde se encuentra el programa que se creó para ejecutarse en el microcontrolador.

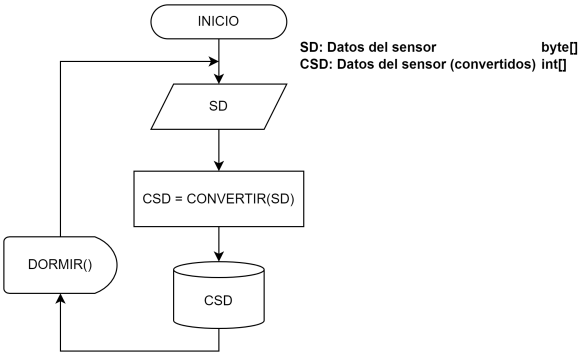

Figura 4.10: Diagrama de flujo del hilo productor

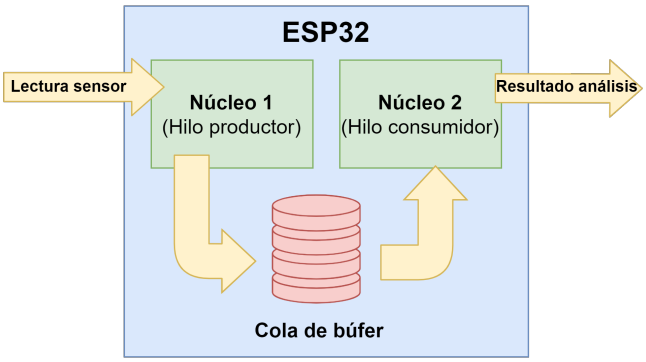

Figura 4.9: Distribución del uso de hilos del procesador

En el que el hilo productor (veáse figura 4.10) se encargará de leer los datos del sensor, convertirlos a un formato decimal para despues almacenar estos datos en una cola que servira como buffer.

En la sección de Anexos se encuentra una guía para la lectura de los diagramas de flujo.

Una vez depositada la información, el hilo se detiene por un tiempo y se repite la operación, esta cola contendrá un elemento por cada muestra tomada. El Hilo consumidor (veáse figura 4.11) estará revisando si existen elementos que esten en la cola, en caso de encontrar un elemento, este pasara a procesarse, de otro modo, esperara un tiempo y repetirá la operación.

La lectura es una operación que no se puede interrumpir y que debe ser cumplida en tiempo para poder realizar la siguiente lectura. Por el otro lado, el hilo consumidor precisamente tiene una cola de procesamiento ya que este podría tardar más o menos tiempo dependiendo del caso.

De esta manera aseguramos que no haya pérdida de información porque el

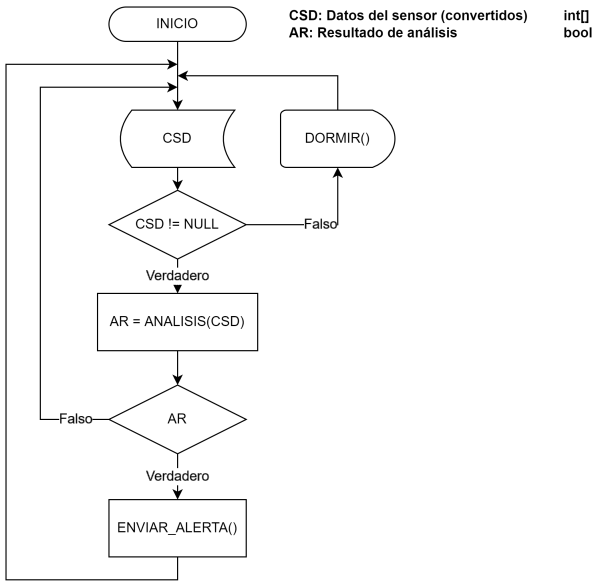

Figura 4.11: Diagrama de flujo del hilo consumidor

consumidor no consuma a la misma velocidad a la que el productor produce, o que el consumidor ocupe tiempo de procesamiento porque no hay nada que consumir.

El hilo consumidor (veáse figura 4.11) lo que hará sera sustraer el siguiente elemento de la cola, es decir la lectura del sensor y determinará si hay una caída en base a la lectura actual y las pasadas.

Es en este mismo hilo también se ejecuta la instrucción de mandar la alerta. Dado que este proceso esto no se ejecuta todo el tiempo, la duración de la tarea puede variar.Aún cuando la tarea pueda ocupar más tiempo de procesamiento, la tarea del productor no se retrasa. De esta manera la lectura se mantiene consistente.

El hilo principal de ejecución (veáse figura 4.12) se encargá de inicializar los parámetros para el algoritmo y la cola del buffer así como la comunicación por WiFi e inicializar el Cliente Websocket. Una vez inicializado el sistema, se procede a iniciar la tarea del hilo productor e inmediatamente después se comienza la tarea del hilo consumidor. En este punto es donde termina la ejecución del hilo principal. Los hilos de producción y consumo aún están en ejecución de manera indefinida, hasta que el dispositivo se quede sin energía.

El microcontrolador se programó de tal modo que implementa el algoritmo descrito en este artículo. Con el añadido de utilizar multiples zonas de detección. Se utilizó el algoritmo múltiples veces para detectar no solamente una pareja de zonas. De este modo se detectan multiples parejas de zonas, lo que hace que se cubra una mayor cantidad de espacio para detectar las caídas.

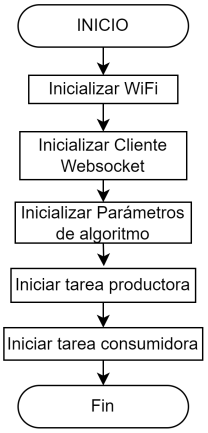

Figura 4.12: Diagrama de flujo del hilo principal de ejecución

El microcontrolador se conecta mediante el uso de protocolo de Websocket a un servicio programado en .NETCore. Este servicio recibe información cada vez que el microcontrolador detecte con el algoritmo una caída. A su vez también el servicio de alertas se conecta a otro servicio de chat en la nube, el cual es una interfaz de chat que provee Telegram. De esta manera nos permite enviar mensajes mediante un chatbot. El cual se creó desde la misma aplicación de Telegram. Cuando un usuario de Telegram manda un mensaje al Chatbot, esta interfaz puede reenviar el texto mediante un HTTP POST a la dirección que nosotros indiquemos.

Teniendo el servicio local expuesto a internet mediante ngrok[8] (o en su defecto exponer un puerto hacia internet).

Mediante ngrok es como llegarán los mensajes, de esta manera recibimos en nuestro servicio programado los mensajes de las personas que hablan con nuestro chatbot. La información que incluye es el nombre de usuario, el texto que nos mando, entre otros datos. De esta manera podemos almacenar el nombre de usuario y posteriormente mandarle mensajes.

Dicho así podemos seguir un flujo en el cual nos registramos con el chatbot y este nos notificará cuando el servicio le indique que recibió del dispositivo la alerta de que ocurrió un evento de caída.

Como se puede apreciar en la figura 4.13 se puede ver el orden en el que algunos elementos interactúan en el sistema. Este diagrama está simplificado y algunos elementos se omitieron por conveniencia, pero en general se trata de un conjunto de elementos que interactúan entre sí y que están ejecutandose en diferentes lugares.

En la sección de Anexos se puede acceder al repositorio donde se encuentra el programa que se creó para el servicio web.

[8]Ngrok es un servicio de internet que nos permite exponer de manera segura un puerto de nuestra computadora sin tener que lidiar con abrir puertos de nuestro router.

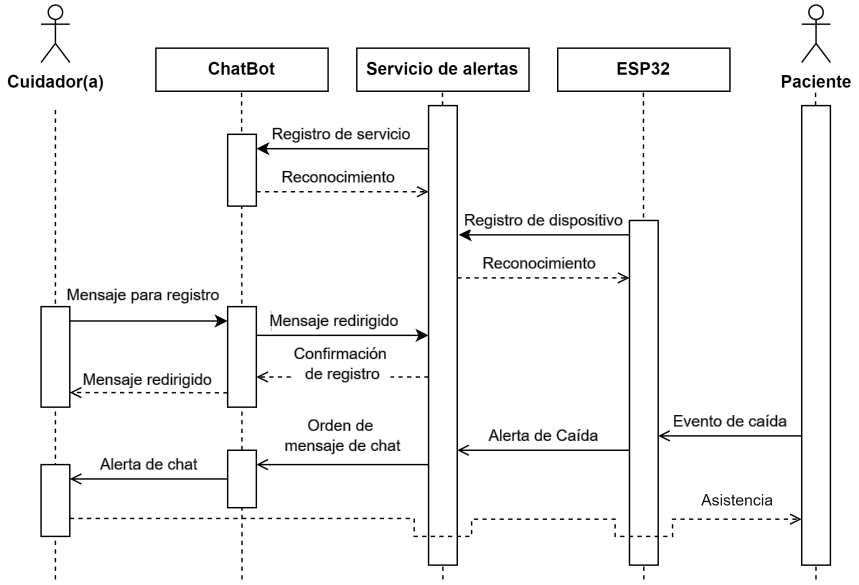

Figura 4.13: Diagrama de secuencia del sistema

4.2. Resultados

En esta sección se discutirán los resultados que se obtuvieron de la evaluación de la propuesta. Se obtuvo una serie de informaciones que detallaremos un poco más adelante y que, en base a esta información obtenida pudimos desarrollar las conclusiones en el siguiente capítulo.

4.2.1. Desempeño del algoritmo para la detección de caídas

Utilizando el algoritmo y los datos recolectados que se clasificaron, se tomaron archivos de caídas convencionales tomadas desde un costado. Para los archivos de no caída se consideraron en los que se caminaba de costado, de esta manera contrastando las muestras de caídas contra la clasificación del algoritmo podemos diferenciar cuales eventos se clasificaron correctamente y de esta manera se consigue una matriz de confusión, y a partir de esta matriz podemos evaluar la efectividad del algoritmo.

Como se puede apreciar en la figura 4.14 a partir del análisis de los archivos con la detección lateral se consiguieron estos resultados. Obteniendo 94 casos que el algoritmo detectó correctamente que no existía caída en el archivo (**TN**[9]). 80 casos que el algoritmo detectó correctamente que si había caída (**TP**[10]).

[9] True Negative: Verdadero Negativo
[10] True Positive: Verdadero Positivo

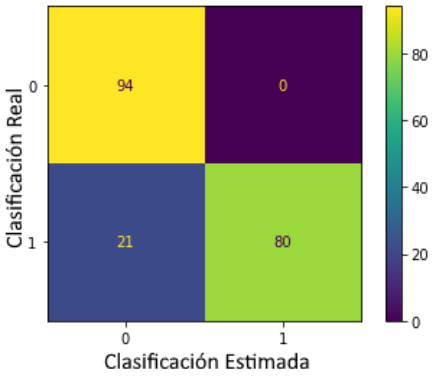

Figura 4.14: Matriz de confusión de la evaluación del algoritmo

Resultaron 21 casos en los que a pesar de que existía una caída se detectó como no caída **(FN**[11]**)**. Y en 0 casos en los que se haya clasificado como caída cuando realmente no lo era **(FP**[12]**)**. Con estos resultados se calcularon los valores para el desempeño del algoritmo. De los cuales se muestran en la tabla 4.1, podemos observar las medidas y los valores que se obtuvieron a partir de la matriz de confusión.

A partir de los resultados obtenidos podemos apreciar en la tabla 4.2 que llegamos a tener un buen desempeño en comparación con los demás trabajos relacionados. A pesar de contar con un único sensor, y de que es de menor resolucion que los demás. La propuesta tiene la capacidad de enviar alertas, así como de ejecutarse en entornos embebidos gracias al algoritmo de baja complejidad computacional.

Después del análisis de los resultados podemos concluir que el algoritmo bajo las circunstancias ideales y los parámetros adecuados puede llegar a tener una buena tasa de detección de caídas. El hecho de solo usar un solo sensor limita la capacidad de obtención de información de tal manera que se complica la tarea de determinar si hay una caída.

De la misma manera la colocación del sensor juega un rol muy importante en la tasa de detección, ya que dependiendo de la posición en la que el sensor es colocado, es más o menos díficil saber con precisión si ha ocurrido una caída. Por ejemplo si el sensor se usa de manera diagonal las temperaturas varían en función de la proximidad hacia el sensor. Y si la distancia varía demasiado puede no detectarse una caída.

De este modo el escenario ideal es cuando el paciente permanece a la misma distancia a lo largo de una caída. Los escenarios no favorables podrían mejorar con los parámetros de entrada. Así pues de esta manera si en una escena en particular la perspectiva cambia mucho con la distancia también la detección del algoritmo puede verse afectada.

[11] False Negative: Falso Negativo
[12] False Positive: Falso Positivo

Medida	**Valor**	**Cálculo**
Sensibilidad	0.7921	`TPR = TP / (TP + FN)`
Especificidad	1	`SPC = TN / (FP + TN)`
Valor Predictivo Negativo	0.8174	`PPV = TP / (TP + FP)`
Tasa de falsos positivos	0	`FPR = FP / (FP + TN)`
Tasa de Descubrimientos Falsos	0	`FDR = FP / (FP + TP)`
Tasa de falsos negativos	0.2079	`FNR = FN / (FN + TP)`
Precisión	**0.8923**	`ACC = (TP + TN) / (P + N)`
Valor F	0.884	`F1 = 2TP / (2TP + FP + FN)`
Coeficiente de Correlación Matthews	0.884	`TP*TN - FP*FN / sqrt((TP+FP)* (TP+FN)*(TN+FP)*(TN+FN))`

Cuadro 4.1: Resultados de la evaluación de desempeño del algoritmo

Característica	**Taniguchi et**	**Taramasco et al**	**Mashiyama et al**	**Moncada et al**
Precisión	76.0-95.5 %	93 %	94.3-95.8 %	**89.23**
Numero de sensores	2	4	1	**1**
Tipo de algoritmo	Determinista	Redes Neuronales	K vecinos mas cercanos (k-NN)	**Determinista**
Resolución sensor	16x16	1x8	8x8	**4x4**
Computo de información de caida	PC(x86)	ARM 32 Bits	PC(x86)	**ESP32 Microcontrolador**
Generación de alertas	No	Si	No	**Si**
Dataset		96 Caídas 112 No caídas	30 Caídas 172 No caídas	**101 Caídas 94 No caídas**

Cuadro 4.2: Tabla comparativa de resultados obtenidos en comparación con trabajos similares

Figura 4.15: Interacción con el chatbot y mensaje de alerta de caída usando Telegram

4.2.2. Implementación del sistema

Una vez que se ejecuta el servicio se puede mandar un mensaje para poder registrarse como receptor de la alerta. Este proceso se ha simplificado en gran medida con el objetivo de demostrar su funcionamiento. Este proceso de registro se puede apreciar en la figura 4.15.

En la sección de Anexos se puede acceder a un vídeo demostrativo en el que se muestra el funcionamiento de la solución.

Ya en la práctica se requieren de más elementos de seguridad tales como emparejar ciertos dispositivos con ciertos usuarios, realizar un mecanismo de respuesta que permita reiniciar una alerta, un mecanismo que permita alertar por otro medio, encriptación de información, entre otros. Estas son algunas faltas en la implementación que se omitieron, precisamente con el fin de realizar una demostración que sustente la viabilidad del desarrollo de este tipo de dispositivos.

Capítulo 5

Conclusiones y trabajo futuro

En la investigación se desarrolló una solución no invasiva, inspirada en el paradigma del IoT que nos permite detectar caídas, muy útil para escenarios en los que el adulto mayor no puede ser supervisado la mayoría del tiempo.

El objetivo se cumple mediante el uso de un algoritmo de fácil implementación y de bajo coste computacional. Algoritmo el cual cuenta con una desventaja ya que resulta ser muy estricto en cuanto a la diversidad de posibles casos de caída.

Que por el contrario, esta baja complejidad computacional es la que nos permite integrar soluciones más sencillas.

Utilizando nuevos microcontroladores que facilitan la interacción con internet. De esta manera es posible desarrollar sistemas que sean más baratos de producir y menos complejos de implementar.|

Una vez teniendo los resultados de la investigación podemos constatar que se puede obtener una buena tasa de detección utilizando un solo sensor infrarrojo. Como pudimos apreciar, utilizando el algoritmo para la detección de caídas se logro detectar con una precisión de alrededor del 89 % bajo condiciones ideales. Tasa la cual mejoraría considerablemente si se añade algún(os) sensor(es) del mismo tipo o incluso diferentes (Como micrófonos, otros sensores de temperatura, etcetera). Utilizando los elementos que conforman el modelo para la detección de caídas es posible añadir condiciones que puedan mejorar la tasa de detección o incluso la velocidad del algoritmo.

El algoritmo puede ser personalizado para que se validen más situaciones o incluso para omitir algunas características para que se ejecute en menos tiempo. Cabe destacar que el algoritmo al ser ligero se puede implementar en un sistema embebido. Se tiene previsto un desarrollo que contemple inteligencia artificial con la finalidad de poder detectar caídas en diferentes escenarios y angulos, en terminos de dispositivos embebidos es algo limitado utilizar estos aparatos como unidades de procesamiento.

Las restricciones de capacidad de computación, de energía y robustez puede afectar la viabilidad de aparatos embebidos con inteligencia artificial. Con el tiempo, dado a las mejoras en chips dedicados a inteligencia artificial y su uso

extensivo provocaría que más aparatos embebidos utilizen inteligencia artificial. En el desarrollo del prototipo se planteó la idea de utilizar computación en la nube para extender las capacidades del sistema embebido. De esta manera el sistema es escalable al no depender enteramente de la potencia computacional del dispositivo, sino que se releva a la computación en la nube que puede ser escalada de manera fácil y transparente para el sistema embebido.

De este modo también el algoritmo en la nube podría ser mejorado con el tiempo y aún con el mismo dispositivo, este seguiría operando de la misma manera conforme pase el tiempo. El uso del algoritmo ligero que se planteó en este artículo podría utilizarse en situaciones circunstanciales en las que no se cuente con acceso a la computación en la nube.

Bibliografía

[1] Ann Borda, Cecily Gilbert, Cathy Said, Frank Smolenaers, Michael McGrath, and Kathleen Gray. Non-contact sensor-based falls detection in residential aged care facilities: developing a real-life picture. *Studies in health technology and informatics*, 252:33–38, 2018.

[2] Yung-Chin Chen and Yi-Wen Lin. Indoor rfid gait monitoring system for fall detection. In *2010 2nd International Symposium on Aware Computing*, pages 207–212. IEEE, 2010.

[3] Amoldo Díaz-Ramírez, Edgar Domínguez, and Luis Martínez-Alvarado. A falls detection system for the elderly based on a wsn. In *2015 IEEE International Symposium on Technology and Society (ISTAS)*, pages 1–6. IEEE, 2015.

[4] Giovanni Diraco, Alessandro Leone, and Pietro Siciliano. An active vision system for fall detection and posture recognition in elderly healthcare. In *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pages 1536–1541. IEEE, 2010.

[5] Charalampos Doukas and Ilias Maglogiannis. Advanced patient or elder fall detection based on movement and sound data. In *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*, pages 103–107. IEEE, 2008.

[6] Espressif. Api reference esp32.

[7] Xiuyi Fan, Huiguo Zhang, Cyril Leung, and Zhiqi Shen. Robust unobtrusive fall detection using infrared array sensors. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 194–199. IEEE, 2017.

[8] FreeRTOS. Freertos - market leading rtos.

[9] Muhammad Salman Khan, Miao Yu, Pengming Feng, Liang Wang, and Jonathon Chambers. An unsupervised acoustic fall detection system using source separation for sound interference suppression. *Signal processing*, 110:199–210, 2015.

[10] Youngbum Lee, Jinkwon Kim, Muntak Son, and Myoungho Lee. Implementation of accelerometer sensor module and fall detection monitoring system based on wireless sensor network. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2315–2318. IEEE, 2007.

[11] Yun Li, KC Ho, and Mihail Popescu. A microphone array system for automatic fall detection. *IEEE Transactions on Biomedical Engineering*, 59(5):1291–1301, 2012.

[12] Shota Mashiyama, Jihoon Hong, and Tomoaki Ohtsuki. A fall detection system using low resolution infrared array sensor. In *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pages 2109–2113. IEEE, 2014.

[13] Alessandro Mecocci, Francesco Micheli, Claudia Zoppetti, and Andrea Baghini. Automatic falls detection in hospital-room context. In *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 000127–000132. IEEE, 2016.

[14] Hoa Nguyen, Farhaan Mirza, M Asif Naeem, and Mirza Mansoor Baig. Detecting falls using a wearable accelerometer motion sensor. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 422–431, 2017.

[15] Thuy-Trang Nguyen, Myeong-Chan Cho, and Tae-Soo Lee. Automatic fall detection using wearable biomedical signal measurement terminal. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5203–5206. IEEE, 2009.

[16] Jacob Nogas, Shehroz S Khan, and Alex Mihailidis. Deepfall: Non-invasive fall detection with deep spatio-temporal convolutional autoencoders. *Journal of Healthcare Informatics Research*, 4(1):50–70, 2020.

[17] Paola Pierleoni, Alberto Belli, Lorenzo Palma, Marco Pellegrini, Luca Pernini, and Simone Valenti. A high reliability wearable device for elderly fall detection. *IEEE Sensors Journal*, 15(8):4544–4553, 2015.

[18] Caroline Rougier, Jean Meunier, Alain St-Arnaud, and Jacqueline Rousseau. Fall detection from human shape and motion history using video surveillance. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, volume 2, pages 875–880. IEEE, 2007.

[19] Wala Saadeh, Saad Adnan Butt, and Muhammad Awais Bin Altaf. A patient-specific single sensor iot-based wearable fall prediction and detection system. *IEEE transactions on neural systems and rehabilitation engineering*, 27(5):995–1003, 2019.

[20] Guto Leoni Santos, Patricia Takako Endo, Kayo Henrique de Carvalho Monteiro, Elisson da Silva Rocha, Ivanovitch Silva, and Theo Lynn. Accelerometer-based human fall detection using convolutional neural networks. *Sensors*, 19(7):1644, 2019.

[21] A Sixsmith, N Johnson, and R Whatmore. Pyroelectric ir sensor arrays for fall detection in the older population. In *Journal de Physique IV (Proceedings)*, volume 128, pages 153–160. EDP sciences, 2005.

[22] Yusuke Taniguchi, Hiroshi Nakajima, Naoki Tsuchiya, Junichi Tanaka, Fumiji Aita, and Yutaka Hata. A falling detection system with plural thermal array sensors. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 673–678. IEEE, 2014.

[23] Carla Taramasco, Tomas Rodenas, Felipe Martinez, Paola Fuentes, Roberto Munoz, Rodrigo Olivares, Victor Hugo C De Albuquerque, and Jacques Demongeot. A novel monitoring system for fall detection in older people. *IEEE Access*, 6:43563–43574, 2018.

[24] Hao Wang, Daqing Zhang, Yasha Wang, Junyi Ma, Yuxiang Wang, and Shengjie Li. Rt-fall: A real-time and contactless fall detection system with commodity wifi devices. *IEEE Transactions on Mobile Computing*, 16(2):511–526, 2016.

[25] Yuxi Wang, Kaishun Wu, and Lionel M Ni. Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):581–594, 2016.

[26] WHO. Falls.

[27] Asanga Wickramasinghe, Roberto Luis Shinmoto Torres, and Damith C Ranasinghe. Recognition of falls using dense sensing in an ambient assisted living environment. *Pervasive and mobile computing*, 34:14–24, 2017.

[28] Falin Wu, Hengyang Zhao, Yan Zhao, and Haibo Zhong. Development of a wearable-sensor-based fall detection system. *International journal of telemedicine and applications*, 2015, 2015.

[29] Zhenhe Ye, Ying Li, Qiaoxiang Zhao, and Xue Liu. A falling detection system with wireless sensor for the elderly people based on ergnomics. *International Journal of Smart Home*, 8(1):187–196, 2014.

Anexos

En esta sección de anexos se encontrará el contenido adicional que se empleo durante la investigación, en la mayoría de los casos son direcciones a repositorios donde se encuentran los programas que se desarrollaron para la elaboración de esta tesis, para mayor .

1. Notebooks para el análisis de las muestras:

   En este repositorio se encuentran los archivos que se capturaron utilizando el sensor térmico y también los notebooks utilizados para realizar el analisis de estos archivos implementando el algoritmo de detección.

   Dirección: `https://github.com/rosendo655/fall_data_analysis`

2. Programa para visualización en tiempo real y de archivos:

   Este programa cumple con varios propósitos, el poder ver en tiempo real las temperaturas que muestra el sensor infrarrojo, así como de visualizar los archivos generados y también para seccionarlos.

   Dirección: `https://github.com/rosendo655/D6TReader`

3. Servicio Web para las alertas:

   Este es el servicio Web que se programó para recibir las alertas provenientes del sensor y reenviarlas al servicio de chat, esta aplicación no cuenta con el algoritmo de detección.

   Dirección: `https://github.com/rosendo655/FallNotificationService`

4. Aplicación embebida para microcontrolador:

   Esta es la dirección hacia el repositorio donde se encuentra la aplicación en C y C++ que se escribió para programarse en el microcontrolador junto con el sistema operativo

   Dirección: `https://github.com/rosendo655/esp32_d6t_reader`

5. Video demostrativo del sistema de detección de caídas:

   Aquí se muestra un video en youtube en el que se puede apreciar el funcionamiento de todos los elementos que se describieron en este documento y se realiza una prueba demostrativa del funcionamiento de la propuesta

   Dirección: `https://youtu.be/_ihN2xeNsgo`

6. Guía para la lectura de diagramas de flujo