FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DISEÑO DE UN PROTOTIPO PARA DETECCIÓN DE CAÍDAS BASADO EN SENSORES INERCIALES

AUTORES

DAMARIS SOLANGE BUSTOS GUEVARA

MARÍA JOSÉ FLORES ANCHATUÑA

AÑO

2020

FACULTAD DE INGENIERIA Y CIENCIAS APLICADAS

DISEÑO DE UN PROTOTIPO PARA DETECCIÓN DE CAÍDAS BASADO EN SENSORES INERCIALES

Trabajo de titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de información

Profesor Guía

MSc. David Fernando Pozo Espín

Autores

Damaris Solange Bustos Guevara

María José Flores Anchatuña

Año

2020

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido el trabajo, Diseño de un prototipo para detección de caídas basado en sensores inerciales, a través de reuniones periódicas con las estudiantes Damaris Solange Bustos Guevara y María José Flores Anchatuña, en el semestre 202020, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación"

David Fernando Pozo Espín

Máster en Automática y Robótica

C.I: 1717340143

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Diseño de un prototipo para detección de caídas basado en sensores inerciales, de las estudiantes Damaris Solange Bustos Guevara y María José Flores Anchatuña, en el semestre 202020, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

Jorge Luis Rosero Beltrán

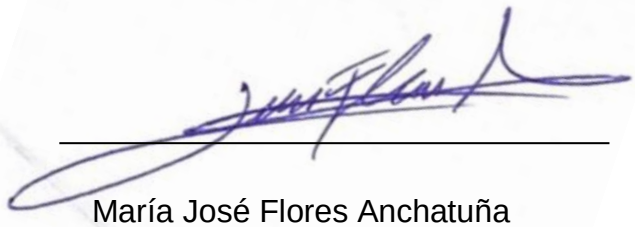
Máster en Ciencias con Especialidad en Automatización

C.I: 1803610185

DECLARACIÓN DE AUTORIA DEL ESTUDIANTE

“Declaro que este trabajo es original, de nuestra autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

Damaris Solange Bustos Guevara
C.I: 1726089764



María José Flores Anchatuña
C.I: 1726518341

AGRADECIMIENTOS

Agradezco en primer lugar a Dios por brindarme la fuerza para poder alcanzar grandes logros en la vida.

Gracias a mi familia por su apoyo incondicional. A mis padres todo lo que soy se los debo a ellos, por todos los esfuerzos, sacrificios y apoyo incondicional, a mi hermana quien es la mejor compañera de vida que tengo ya que sin su apoyo, consejos y confianza no habría podido llegar a donde estoy ahora.

A mi amiga María José Flores por haber estado siempre junto a mí y haber compartido durante tanto tiempo muchas experiencias inolvidables

A mis amigos Estéfano, Elvis, Santiago S, Kevin, quienes me han brindado su apoyo incondicional y han hecho que este camino sea más fácil.

Solange

AGRADECIMIENTOS

Al culminar este trabajo de titulación quiero agradecer a Dios por todas sus bendiciones. A mis padres quienes a lo largo de mi vida me han brindado las herramientas y oportunidades necesarias para crecer personal como profesionalmente, a mi hermano por sus consejos y motivación.

A mi amiga Solange Bustos, por ser mi compañera durante todo este proceso, por compartir alegrías y fracasos, por haber logrado nuestro objetivo.

A las personas importantes que fui encontrando en el camino Elvis, Santiago S y Ronald por haber sido parte de todo este proceso y haberme compartido su conocimiento de la manera más desinteresada. A todos quienes han formado parte de este proceso.

María José.

DEDICATORIA

Este proyecto de tesis se lo dedico a Dios por haberme guiado y cuidado para poder cumplir mis metas, a mis padres quienes son mi pilar fundamental y quienes me han acompañado en todo este camino, a mis hermanos Mishell y Mathías que siempre han sido mi más grande apoyo y fortaleza para poder seguir adelante y no rendirme nunca.

Solange.

DEDICATORIA

El presente proyecto se lo dedico a mis padres por ser mi luz en todo momento, por siempre confiar en mí y motivarme a ser mejor, son el pilar fundamental de mi vida y mi ejemplo a seguir, a mi hermano por ser mi fuente de motivación en todo momento, y siempre estar dispuesto a escucharme y aconsejarme en las distintas etapas de mi vida y mi carrera.

María José.

RESUMEN

De acuerdo con investigaciones realizadas se determinó que las caídas son una de las principales razones de mortalidad a nivel mundial, a lo largo del tiempo se han buscado maneras de poder brindar un mejor estilo de vida al ser humano. Es por eso que se ha buscado diseñar e implementar un prototipo detector de caídas.

En el presente trabajo de titulación se desarrolla un sistema embebido de adquisición de señales en los ejes x, y, z para la detección de tres distintas clases: caídas, caminar e inmóvil, el prototipo será colocado en la cadera y la información enviada será procesada para que el algoritmo pueda predecir cuál es la actividad realizada.

Esto es posible mediante la utilización del sensor inercial MPU-6050 el cual envía señales de aceleración en los tres ejes y mediante la comunicación Bluetooth poder realizar la transmisión de datos. Después que la señal sea procesada se procede a extraer las características (Media, Mediana, Varianza, Percentil 25 y Asimetría), las cuales son aplicadas en el algoritmo clasificador Naive Bayes previamente entrenado con una base de datos. Luego se predice el evento realizado y se envía el resultado a la interfaz gráfica para poder visualizarlo.

Finalmente, se realizan las pruebas del sistema a seis personas tanto en la clase de caídas como caminar y tres personas en la clase inmóvil para poder verificar el correcto funcionamiento del algoritmo, Las pruebas son realizadas tomando varios conjuntos de datos de diferentes tamaños para evaluar el comportamiento frente a la variación de información obtenida, se determina la exactitud del sistema que se encuentra entre 60% al 82% con tiempos de procesamiento del algoritmo de 0.50 milisegundos a 1.35 milisegundos. Lo que permite concluir que el número de datos afectará a los resultados y el tiempo de procesamiento.

ABSTRACT

According to research, it was determined that falls are one of the main reasons for mortality worldwide, over time, ways have been found to provide a better lifestyle for humans. That is why we have sought to design and implement a prototype fall detector.

In the present degree work, an embedded signal acquisition system is developed in the x, y and z axes for the detection of three different classes: falls, walking and immobile, the prototype will be placed on the hip and the information sent will be processed so that the algorithm can predict what the activity is.

This is possible by using the MPU-6050 inertial sensor which sends acceleration signals on all three axes and by means of Bluetooth communication to carry out data transmission. After the signal is processed, the characteristics (Mean, Median, Variance, 25th Percentile and Asymmetry) are extracted, which are applied in the Naive Bayes classifier algorithm previously trained with a database. Then the event performed is predicted and the result is sent to the graphical interface to be able to visualize it.

Finally, the system tests will be carried out on six people in both the falling and walking classes and three people in the immobile class to verify the correct operation of the algorithm. The tests are carried out by taking several data sets of different sizes to evaluate the behavior faced with the variation of information obtained, the precision of the system is determined, which is between 60% to 82% with algorithm processing times from 0.50 milliseconds to 1.35 milliseconds. This allows us to conclude that the number of data affects the results and the processing time.

ÍNDICE

1. INTRODUCCIÓN

La Organización Mundial de la Salud (OMS) define las caídas como “acontecimientos involuntarios que hacen perder el equilibrio y dar con el cuerpo en tierra u otra superficie firme que lo detenga”. De acuerdo con los datos recopilados por la OMS se conoce que aproximadamente se producen al año alrededor de 646.000 caídas mortales. Colocando a las caídas como la segunda causa mundial de muerte por lesiones no intencionales (Salud, 2018).

Anualmente se producen 37,3 millones de caídas que, a pesar de no ser mortales, requieren atención médica. La mayor morbilidad corresponde a las personas mayores de 65 años, jóvenes de 15 a 29 años y a los menores de 15 años (Salud, 2018).

Actualmente, el uso de dispositivos inteligentes y aplicaciones móviles son muy comunes para poder brindar seguridad a niños como a jóvenes y adultos. Por ello, se desarrollan distintas tecnologías con el fin de que las personas estén más seguras.

Estas se pueden dividir en cuatro grandes grupos: dispositivos de detección inmediata, dispositivos de comportamiento inusual, otro método de detección de caídas y la monitorización ambiental en comportamientos inusuales(González et al., 2015).

Dispositivos de detección inmediata son aquellos dispositivos portables, los cuales son capaces de detectar una caída cuando ocurre y enviar una alarma inmediata. Las tecnologías que suelen ser utilizadas son dispositivos de choque, posición, sensores de inclinación y acelerómetros junto con algoritmos de control adecuados en un microcontrolador (González et al., 2015) (Solórzano et al., 2018).

Dispositivos de comportamiento inusual son pequeños dispositivos llevados por los usuarios, pero no detectan caídas. Estos solo controlan la actividad de la persona y detectan cuando ocurre algún comportamiento inusual comparado con el patrón típico de una persona. Estos dispositivos alertan al detectar acciones anormales lo que no diferencia si es caída (González et al., 2015).

Otro método de detección de caídas es con la tecnología de monitorización ambiental de detección inmediata, la cual, consiste en la instalación de sensores en el entorno de la persona para detectar una caída. Las tecnologías utilizadas son: grabación de video (María et al., n.d.), análisis de imagen (María et al., n.d.), análisis de sonido, instalación de sensores de choque en el suelo y alfombras (González et al., 2015).

En la monitorización ambiental para comportamientos inusuales, los entornos se equipan con sensores para monitorizar a las personas como en puertas, ventanas, barreras infrarrojo, detectores IR pasivos. Se procede a analizar la información recopilada por los sensores y poder detectar comportamientos que conllevan a una caída. Estos dispositivos requieren una gran infraestructura y un buen sistema de análisis (González et al., 2015).

Así también, existen sistemas de detección de caídas de alta precisión que principalmente se basan en un sistema llamado “Motion History Imafin” (MHI). Este sistema se centra en pixeles de color blanco y gris, lo cual ayuda a estimar cuan rápido es el movimiento considerando todo fondo negro innecesario. Tomando en cuenta que el blanco es la persona y lo gris su estela al moverse, para poder detectar caída utilizan una formula C= (número de pixeles grises)/(número de pixeles grises + número de pixeles blancos (Persona-Ordenador, 2018).

Resumiendo lo mencionado anteriormente, los sistemas portátiles detectores de caídas tienen la facilidad de que pueden ser utilizados fuera de la casa, siempre y

cuando no sean incomodos dependiendo del usuario (Persona-Ordenador, 2018) .

Como ejemplo se puede mencionar a Angel4, que es un dispositivo que detecta una caída usando un acelerómetro triaxial y un algoritmo en específico. Este dispositivo consta de un pequeño sensor colocado en la cintura, este sensor es conectado al sistema de teleasistencia, al centro de salud o al teléfono móvil mediante Bluetooth, y es controlado a través de una aplicación fácil de usar como de personalizar (SENSE4CARE, n.d.)(Solórzano et al., 2018).

Otro dispositivo que detecta caídas es “Vigi’Fall”, el cual envía una alerta en el caso de que sea una caída seguida de la imposibilidad de levantarse por sí mismo. Está formado por un pequeño biosensor en forma de parche triangular. Este dispositivo va adherido y se lo puede usar incluso en la ducha. Su función es emitir señales que son vitales y son recibidas por sensores inalámbricos colocados en las paredes de la casa, en el caso de existir una caída el biosensor emite una alerta además los sensores de las paredes detectan la ausencia de movimiento y de forma inalámbrica emiten una señal a la caja de control que se encuentra dentro de la casa (Vigi’Fall, n.d.) (Barreras, n.d.)

También se puede encontrar “Vibby”, otro dispositivo detector de caídas, el cual se lo lleva en la muñeca, este dispositivo emitirá una alarma en caso de detectar una caída y además permite llamar fácilmente para poder pedir ayuda de forma manual. Este dispositivo funciona con un sensor de presión y un algoritmo innovador que mide la pérdida de altitud y velocidad, además de poder contar con vibraciones para confirmar que las alertas han sido enviadas (Asistencias, n.d.).

Bodymedia es un dispositivo utilizado como brazalete colocado en el antebrazo del usuario, este dispositivo tiene un peso aproximado de 82 gr. Este dispositivo consta de sensores de movimiento, flujo calórico, temperatura en la piel. Además, guarda toda la información en una memoria interna, la cual es transferida vía USB para

poder tener un análisis de los datos guardados (Otero, 2007).

Así como los dispositivos han ido evolucionando y volviéndose más útiles, se han realizado varias investigaciones previas para lograr tener equipos funcionales. Es así como en el 2016 se realizó una investigación en la cual se propone varios elementos textiles que pueden ayudar a la detección de caídas; entre estos elementos se puede encontrar

El diseño de un calcetín sensorizado, que está compuesto de sensores de presión, los cuales serán ubicados de manera óptima obteniendo así la información del dedo gordo que es la última parte que tiene contacto con el suelo y la zona del talón que es la primera zona en perder el contacto con el suelo (Aitex, 2017).

También se realizó el diseño de los elementos de protección de pasivos que es una tecnología que busca el desarrollo de sistemas amortiguadores de golpes mediante elementos pasivos, mediante la impresión 3D. El diseño específicamente se plantea para protecciones para caderas, rodillas, codos, espaldas y hombros (Aitex, 2017).

Otro diseño propuesto es crear un dispositivo de monitorización de posturas que consiste en la monitorización de la aceleración del cuerpo humano, una camiseta, un pantalón y una faja. Para este traje se utilizarán 10 sensores IMU en total de forma en que los movimientos de los brazos, piernas y tronco queden totalmente monitorizados (Aitex, 2017).

“Automatic Detection of Falls and Fainting” es otro de los dispositivos que surgió a partir de las investigaciones en el campo, el cual consiste en detectar posibles caídas o desmayos dentro de un área en específico por un dispositivo Kinect situado en esta área y conectado a un computador del personal de vigilancia. En el caso de que exista una caída, una alerta se activará avisando de que ha sucedido una caída o desmayo (Persona-Ordenador, 2018).

Varios de los dispositivos previamente mencionados han podido ayudar a la detección de caídas gracias a la aplicación del Machine Learning, este es un método analítico que permite que un sistema pueda aprender ciertos patrones sin la necesidad del ser humano (SAS, s.f.), esto se lo realiza a través de un entrenamiento con grandes volúmenes de datos. Para lograr esto se utiliza algoritmos, los cuales se los puede clasificar en tres grandes grupos:
Algoritmos supervisados, se los utiliza con una recopilación de datos precalificados, los cuales son procesados para poder realizar predicciones sobre los mismos. El proceso de entrenamiento terminará cuando el modelo alcanza la precisión deseada de los datos (Russo et al., 2016) (Perspective, 2015).

Algoritmos semi supervisados, en este algoritmo se toman datos preclasificados como no preclasificados para poder crear una función clasificadora. Este tipo de algoritmo se utiliza para poder realizar predicciones (Russo et al., 2016)(Perspective, 2015).

Algoritmos no supervisados, en este algoritmo se utiliza datos no preclasificados además de que no se tiene un resultado conocido. Este resultado se lo puede obtener a través de procesos matemáticos para reducir la redundancia (Russo et al., 2016)(Perspective, 2015).

Además, también se puede encontrar algoritmos específicos con diferentes características para el tratamiento de datos, entre estos tenemos:

Deep Learning, es un algoritmo de aprendizaje automático con múltiples niveles de abstracción de la información, este algoritmo consta de una estructura compleja en grandes conjuntos de datos. Cuyo fin es poder crear modelos capaces de crear de una manera automática una jerarquía desde los conceptos simples hasta llegar a crear conceptos más complejos. (Perspective, 2015)(Russo et al., 2016)(Casas Roma et al., 2019).

Active Learning, es un algoritmo semi supervisado el cual puede lograr una mayor precisión con menos instancias de entrenamiento, este algoritmo interactúa con un usuario para poder obtener la precisión deseada (Settles, 2009)(Russo et al., 2016).

Existen muchos más algoritmos, los cuales ayudan al desarrollo de distintas tecnologías dependiendo del problema o dispositivo a implementar; se deberá escoger uno para poder obtener los resultados esperados.
Uno de los algoritmos más eficientes y efectivos es Naive Bayes, el cual infiere que una acción pertenece a alguna clase basándose en el supuesto de que todos sus atributos son independientes entre sí (Chen et al., 2020).

Naive Bayes es un algoritmo muy eficaz ya que se ha probado en muchas aplicaciones incluyendo, la clasificación de texto, diagnóstico médico y la gestión de rendimiento de sistemas (Rish, 2020).

La ventaja que tiene este algoritmo es la independencia de las características, lo que quiere decir que no están relacionadas dando como resultado que las características coinciden en la clase más probable. Esto permite que cada característica contribuya a la decisión final (Rish, 2020)(Ting et al., 2011).

En el presente proyecto de titulación se procederá a diseñar e implementar un prototipo embebido para la detección de caídas con el uso de una placa microcontroladora y sensores inerciales; lo que permitirá la recolección de los datos y su procedimiento aplicando un algoritmo clasificador.

1.1. Objetivo General

Implementar un dispositivo embebido que permita la detección de caídas a través de sensores inerciales.

1.2. Objetivos Específicos

- Realizar un estudio acerca de los dispositivos existentes disponibles para la detección de caídas e investigaciones en curso.
- Revisar el funcionamiento y aplicación del algoritmo de Naive Bayes como algoritmo clasificador.
- Implementar un sistema microcontrolado que permita detectar caídas.
- Realizar pruebas de funcionamiento del prototipo de detección de caídas.

1.3. Alcance

En una primera instancia el trabajo contempla una amplia revisión bibliográfica sobre los distintos tipos de dispositivos comerciales existentes e investigaciones que se hayan realizado sobre el tema.

Además, el trabajo de titulación está enfocado en el diseño e implementación de un prototipo que permita detectar caídas mediante el uso del algoritmo de Naive Bayes como clasificador que utilizará los datos de aceleración obtenidos de un sensor inercial en los ejes X-Y-Z. Los datos obtenidos de aceleración triaxial serán adquiridos y procesados en un sistema microcontrolado. Para la visualización y almacenamiento de los datos obtenidos en las pruebas del prototipo se desarrolla una interfaz gráfica.

Con el fin de comprobar el correcto funcionamiento del sistema implementado se realizarán las pruebas pertinentes en un ambiente controlado.

1.4. Justificación

Basándose en los antecedentes presentados anteriormente, la tecnología ha sido un pilar importante en avances médicos y de cuidado de las personas para poder brindar una mayor seguridad a las mismas.

Surge una necesidad de ayudar a personas vulnerables a este tipo de accidentes involuntarios diseñando un dispositivo que proporcione ayuda en el caso de sufrir alguna caída, lo que lleva a las personas a investigar nuevas tecnologías y adaptarlas a distintas necesidades.

Este proyecto pretende detectar caídas para mejorar el cuidado de las personas, lo cual a futuro se podría utilizar como base para ayudar a las personas de la tercera edad. Actualmente, en ecuador no se puede encontrar un dispositivo completo para poder brindar ayuda en el caso de una caída.

2. MARCO TEÓRICO

En el presente capítulo se explica de forma adecuada los conceptos teóricos relacionados con los dispositivos detectores de caídas, algoritmos clasificadores y otros temas importantes para una mejor comprensión del desarrollo del presente trabajo de titulación.

2.1 Sensor inercial

Los sensores inerciales son aquellos dispositivos que permiten medir la aceleración lineal o velocidad angular, estos dispositivos son pequeños, tienen rangos de medición y sensibilidad totalmente adecuados para ser implementados en aplicaciones donde los equipos son ligeros para no impedir o alterar los movimientos naturales de los sujetos bajo medición.

MPU-6050

Este sensor consta de un giroscopio de 3 ejes y un acelerómetro 3 ejes con un procesador digital de movimiento, además que tiene puertos de I2C. Debido a su costo y su disponibilidad en el mercado nacional, ha sido escogido para el proyecto de titulación Figura 1.

Tabla 1.

Especificaciones técnicas

Ejes	**x y z**
Protocolo	i2c
Voltaje entrada	2.3V-3.4V
Sensibilidad	± 0.02
Rango aceleración	$\pm 2g, 4g, 8g, 16g$

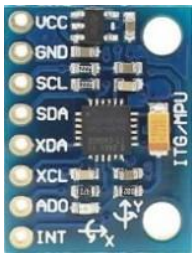


Figura 1. sensor MPU-60-50

2.2 Microcontrolador

Los microcontroladores tienen un circuito integrado el cual contiene un microprocesador, memoria de datos, memoria de programa y periféricos tanto de entrada como de salida haciéndolo pequeño y fácil al utilizarlo. (Molina, 2019)

Teensy 4.0

La placa contiene un microcontrolador de ARM Cortex-M7 de 32 bits a 600Hz, esta placa consume alrededor de 100mA de corriente.

Posee 40 entradas y salidas digitales de las cuales, 31 pueden ser usados con PWM, 14 entradas analógicas con 13 bits de resolución, 7 UARTs, 2 DAC, posee un clock de 600 MHz, también cuenta conexión USB para alimentación y programación.

Además se tiene pines para un puerto USB, el cual permite conectar memorias USB o mouse (Store, 2020).

En cuanto a velocidades, se tiene que el Teensy 4.0 es 5 veces más rápido al Teensy 3.6, 22 veces más rápido que el Arduino Due y hasta 330 veces mucho más rápido que el Arduino Uno/Mega/Nano (Store, 2020).

*Figura 2.*Teensy 4.0

Una de las características principales de esta placa es el proceso de recuperación de la placa, lo que quiere decir que se restaurará sin necesidad de una computadora. El proceso se lo realiza presionando el botón de reinicio durante aproximadamente 15 segundos, el Led rojo comenzará a parpadear indicando que se ha ingresado en modo de restauración. Una vez que el botón ya sea soltado la memoria flash se borrará y se reestablecerá a la configuración inicial (*New Teensy 4.0 Blows Away Benchmarks, Implements Self-Recovery, Returns To Smaller Form* | *Hackaday*, 2019).

A continuación, se podrá visualizar las diferencias entre varias placas microcontroladoras Tabla 2.

Tabla 2.

Comparaciones entre microcontroladores

Placa	Teensy 4.0	Teensy 3.2	Arduino Due	STM 32	Arduino Mega	Arduino Nano	Arduino Uno
Memoria RAM	1024 KB	64 KB	64 KB	20 KB	8 KB	1 KB	2 KB
Memoria Flash	2048 KB	256 KB	512 KB	32 KB	256 KB	32 KB	32 KB

Clock	600 MHz	120 MHz	84 MHz	16 MHZ	16 MHz	16 MHz	16 MHz
Voltaje operación	3.3 V	3.3 V	3.3 V	3.3 V	5V	5V	5V
Voltaje alimentación	3.6 - 6 V	3.6 - 6 V	7-12V	5 V	5V	5 V	5V
Microcontrolador	ARM cortex M7	ARM cortex M4	AT91SAM3X8E	ARM cortex M3	ATmega2560	ATmega328P	ATmega328P
Dimensiones	3.5x1.8 cm	3.5x1.8 cm	10.16x 5.3 cm	5.3x2.2 cm	10.15x5.33 cm	4.5x1.8 cm	8x5.5x2.5 cm

2.3 K-Fold

Es un procedimiento de muestreo utilizado para poder estimar modelos de aprendizaje automático dado una muestra de datos limitada (*3.1. Cross-Validation: Evaluating Estimator Performance — Scikit-Learn 0.23.1 Documentation*, n.d.).

El conjunto de datos se divide en K conjuntos más pequeños, los cuales serán utilizados para entrenar un modelo y el porcentaje que queda son utilizados para evaluar el desempeño del modelo. Este proceso se lo llama validación cruzada, el cual se utiliza principalmente para el aprendizaje automático, en la figura_3 tomado de (Anguita et al., n.d.) se puede observar la división que se realiza en el K-fold (Anguita et al., n.d.)(3.1. Cross-Validation: Evaluating Estimator Performance — Scikit-Learn 0.23.1 Documentation, n.d.).

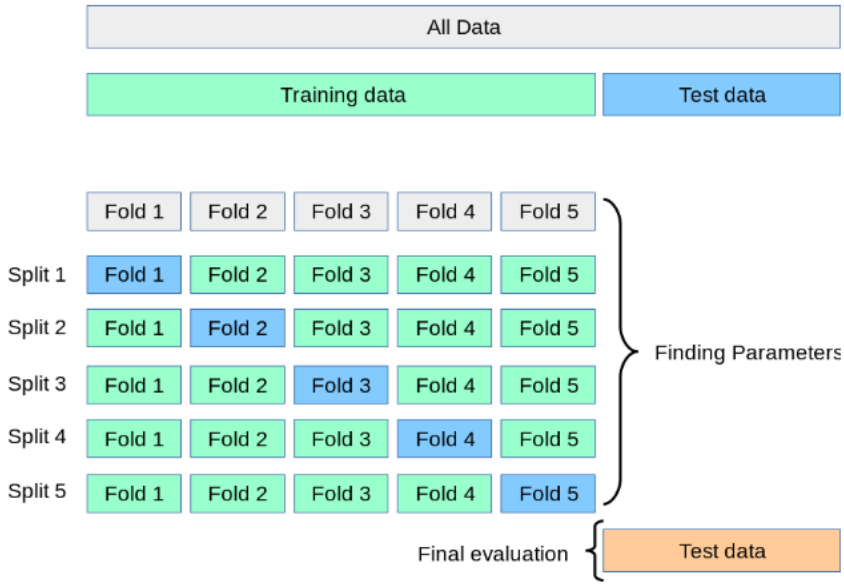


Figura 3. Tomado de (Anguita et al., n.d.) K-Fold cross validation

2.4 Clasificador Naïve Bayes

El clasificador de Naive Bayes es un clasificador estadístico. También es llamado algoritmo ingenuo ya que supone que todas las características contribuyen independientemente a la probabilidad de que sea una clase u otra.

Este clasificador determina la presencia (o ausencia) de una característica particular de una clase no está relacionada con la presencia (o ausencia) de cualquier otra característica cuando la clase es dada.

Este algoritmo está basando en el Teorema Bayesiano y es utilizado cuando la dimensionalidad de las entradas es alta. (Jadhav, 2016)

Ventajas

- Requiere un corto tiempo de cálculo para el entrenamiento.
- Mejora el rendimiento de clasificación al eliminar características irrelevantes.
- Buen rendimiento.

Desventajas

- Requiere un gran número de registros para un mejor resultado

- Menos preciso en comparación con otros clasificadores en algunos conjuntos de datos.

Algoritmo Naive Bayes

Este clasificador nace a partir del teorema de Bayes, el cual se indica en la Ecuación 1 (Pozo, 2017)(Abel et al., 1996)

$$P(C_j|X) = \frac{P(X|C_j)P(AC_j)}{P(X)} \qquad \text{(Ecuación 1)}$$

Donde:

$\mathbf{P(C_j|X)}$: Probabilidad posterior de una clase

$\mathbf{P(X|C_j)}$: Probabilidad condicional acumulada

$\mathbf{P(X)}$: Evidencia

$\mathbf{P(X|C_j)}$: Es detallada en la Ecuación 2, la cual consiste en la multiplicación de las probabilidades condicionales, la cual es detallada en la Ecuación 3 (Pozo, 2017)

$$P(X|C_j) = \prod_{i=1}^{n} P(X_i|C_j) \qquad \text{(Ecuación 2)}$$

Donde:

$\mathbf{P(X|C_j)}$: Probabilidad condicional acumulada.

$\mathbf{X}$: Vector de características de clase j

$\mathbf{P(X_i|C_j)}$: Probabilidad condicional de i características con la clase j.

$$P(X_i|C_j) = \frac{1}{\sqrt{2\pi\delta_{ic}^2}} * e^{\frac{(x_i-u_{ic})^2}{2\delta_{ic}^2}} \qquad \text{(Ecuación 3)}$$

Donde:

$\mathbf{P(X_i|C_j)}$: Probabilidad condicional.

$\mathbf{x_i}$: Características del vector X

$\mathbf{C_j}$: Clase a ser analizada

$\boldsymbol{\delta_{ic}}$: Desviación estándar de la característica de entrenamiento i de la clase j.

$\boldsymbol{u_{ic}}$: Valor medio de la característica de entrenamiento i de la clase j

$\boldsymbol{P(C_j)}$**:** Es un valor que tiende a variar dependiendo de las clases que se tengan entrenadas. (Pozo, 2017)

$$P(C_j) = 0.33 \qquad \text{(Ecuación 4)}$$

$\boldsymbol{P(X)}$: Es la sumatoria de los productos entre la probabilidad condicional acumulada y la probabilidad previa de las clases abierto y cerrado. Ecuación 5 (Pozo, 2017)

$$P(X) = \sum_{j=1}^{m}\left(\prod_{i=1}^{n}\left(P\left(\frac{X_i}{C_j}\right) * P(C_j)\right)\right) \qquad \text{(Ecuación 5)}$$

Donde:

$\boldsymbol{P(X)}$: Evidencia

$\boldsymbol{P(X_i|C_j)}$: Probabilidad condicional acumulada de la clase i.

$\boldsymbol{P(C_j)}$: Probabilidad previa de la clase i

2.5 Características utilizadas por el clasificador

Con lo mencionado anteriormente, los clasificadores requieren de características para su funcionamiento, las cuales se calculan con matemáticas estadísticas.

A continuación, se detallarán las características que utilizará el clasificador Naïve Bayes.

2.5.1 Varianza (VAR)

La varianza se la representa con la sumatoria del cuadrado de las diferencias entre un dato y su valor medio del grupo al que pertenece. Y ser dividido para el número total elementos, como se muestra en la Ecuación 6 (Pozo, 2017)(Nazmi et al., 2015)

$$VAR = \frac{1}{n}\sum_{i}^{n}(x_i - \bar{x})^2 \qquad \text{(Ecuación 6)}$$

Donde:

VAR: Varianza

$\bar{x}$: Valor medio del grupo de datos.

n: Número total de elementos

2.5.2. Mediana (Me)

Para el cálculo de la mediana se tiene que tomar en cuenta dos procesos:

En el caso de que el número total de valores n sea impar, entonces: el valor de la mediana ocupará la posición de la Ecuación 7, siempre y cuando los valores este ordenados (Santiago Solórzano, 2019)

$$Me = x_{\frac{n+1}{2}} \qquad \text{(Ecuación 7)}$$

En el caso de que el número total de valores n sea par, entonces el valor de la mediana ocupara la posición de la Ecuación 8.

$$Me = \frac{\left(X_{\frac{n}{2}} + X_{\frac{n}{2}+1}\right)}{2} \qquad \text{(Ecuación 8)}$$

Donde:

Me: Mediana

n: Número total de elementos

2.5.3. Media (Me)

Se define como la sumatoria de los números de la muestra dividido para el número total de la muestra. Ecuación 9 (Santiago Solórzano, 2019)

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n} \qquad \text{(Ecuación 9)}$$

Donde:

$\bar{x}$: Valor medio del grupo de datos.

n: Número total de elemento

x_n: Valores de la muestra

2.5.4. Asimetría (CA)

Se define como la resta de los datos con la media elevada al cubo, todo esto dividido para la desviación típica. (Santiago Solórzano, 2019)

$$CA_F = \frac{\sum_{i=1}^{N}(X_i - \bar{x})^3}{N * S_x^3} \qquad \text{(Ecuación 10)}$$

Donde:

$\bar{x}$: Valor medio del grupo de datos.

$\mathbf{X_i}$: Valor dado

$\mathbf{S_x^3}$= desviación típica

2.5.5 Percentil 25 (x)

Se define como el número de la muestra por el percentil dividido para cien. (Santiago Solórzano, 2019)

$$x = \frac{n * i}{100} \qquad \text{(Ecuación 11)}$$

Donde:

x: Percentil

n: Número de elementos de la muestra

i: Percentil

3. DESARROLLO E IMPLEMENTACIÓN

En el presente capítulo, se detalla el desarrollo del sistema microcontrolado y la captura de una señal de aceleración en los ejes x, y, z. También se presenta la implementación del algoritmo clasificador de Naive Bayes para la detección de caídas.

3.1. Esquema del general de funcionamiento del sistema

La implementación del prototipo detector de caídas se divide en dos partes esenciales, la parte de entrenamiento fuera de línea, donde se realiza la preparación de datos y pruebas para que el algoritmo de Naive Bayes tenga un buen desempeño la cual se la puede ver en el diagrama de flujo en el primer cuadro entrecortado. La parte de implementación del algoritmo para las actividades de vida diaria ADLs, donde se toma los datos en tiempo real y se procesa la información para conocer si es caída o no la cual se puede observar en el segundo cuadro entrecortado. A continuación, se puede observar lo antes mencionado Figura4.

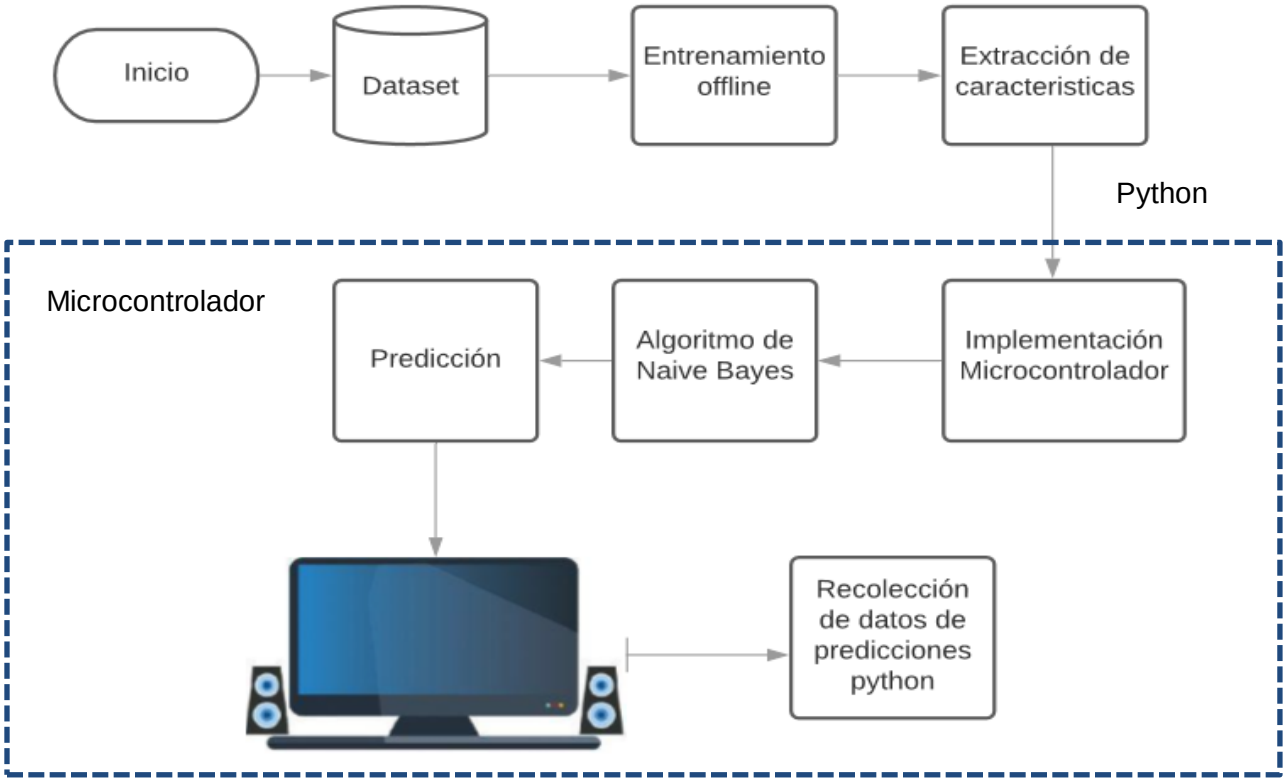


Figura 4. Esquema del Sistema.

3.2. Adaptación del módulo Bluetooth HC-05

Para la adaptación del módulo HC-05 se procede a cambiar la configuración inicial del Bluetooth, el cual viene a una velocidad de 9600 Baudios a 115200 Baudios; ya que esa velocidad es la requerida para la abstracción de información en tiempo real (Configuración Del Módulo Bluetooth HC-05 Usando Comandos AT, n.d.).

El bluetooth es conectado a un Arduino uno y se envía el siguiente comando para poder realizar el cambio de velocidad mencionado previamente.

"AT+UART=<Baud> ,< StopBit>,< Parity>"

Donde:

Baud: es la velocidad a la que se requiere cambiar donde los valores pueden ser: 4800, 9600, 19200, 38400, 57600, 115200, 23400, 460800, 921600 o 138240.

StopBit: representa al bit de parada, este bit puede ser 0 o 1, para 1 o 2 bits de parada respectivamente.

Parity: Equivale a la paridad estos valores puede ser 0,1, 2 en donde 0 representa si tiene paridad, 1 si se tiene paridad impar y 2 si la paridad es par.

Al enviar el comando se recibirá un "OK" para confirmar el cambio de la velocidad.

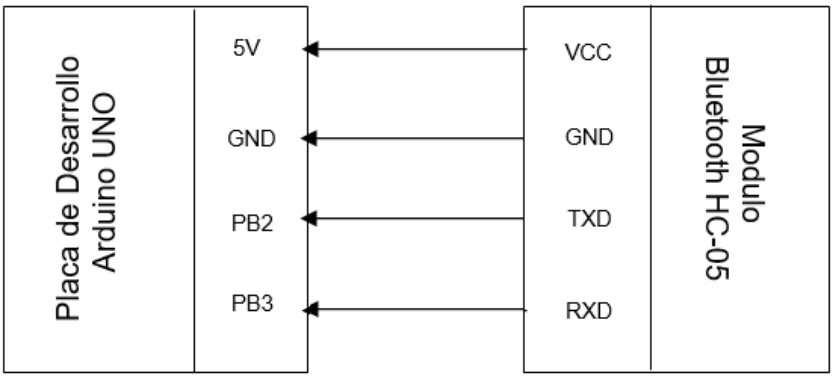


Figura 5. Conexión con Arduino Uno 4.0

3.3 Conexiones del Teensy 4.0

Antes de comenzar con la programación, es importante mostrar el diagrama de conexión entre los elementos fundamentales que son, el módulo HC-05, el sensor inercial MPU-6050 y el microcontrolador.

El diagrama respectivo se lo podrá ver en la figura6.

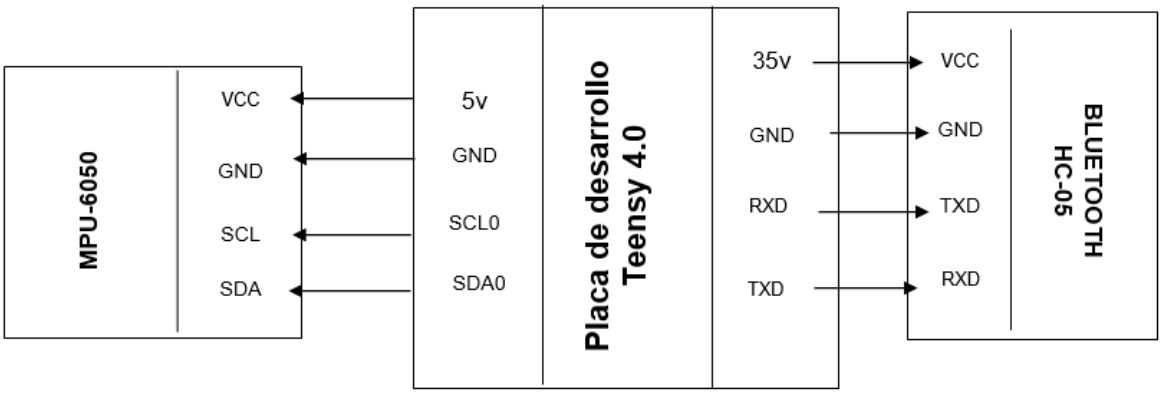


Figura 6. Conexiones Teensy

Tabla 3.

Puertos Teensy

Teensy 4.0	**Puerto**
A5	SCL0
A4	SDA0
0	RDX
1	TDX

3.4. Entrenamiento fuera de línea (offline)

En la parte offline se utiliza la base de datos del “Institute of Communications and Navigations” (*DLR - Institute of Communications and Navigation - Data Set*, n.d.) que contiene 17 clases. de las cuales se ha decidido usar dos clases: caídas y caminar. Cada una de las clases contiene las variables de la aceleración en los ejes x, y, z.

Para el propósito de este trabajo los datos se han transformado en módulos donde el módulo es igual a $\sqrt{x^2 + y^2 + z^2}$, además es importante mencionar que cada clase contiene distintos números de eventos que son:

La clase caídas consta de 32 eventos, donde cada uno de los eventos tienen un número determinado de muestras que varían de 92 a 280.

La clase caminar consta de 42 eventos y el número de muestras varía desde 661 hasta 22084.

Adicional a las dos clases obtenidas de la base de datos antes mencionada se trabajó con una tercera clase denominada inmóvil, la cual fue creada tomando los datos desde el sensor en tiempo real. Esta clase cuenta con 23 eventos con un número de muestras que varía desde 133 hasta 192.

Posteriormente, se realiza la lectura de los módulos, los cuales son utilizados en función a lo detallado en (Solórzano et al., 2018), donde se establece que para hacer una

detección de caídas efectiva se debe realizar el cálculo de los módulos, y a partir de estos realizar los cálculos de las características a utilizar: media, mediana, varianza, percentil 25 y asimetría. Estas características fueron tomadas de una investigación previa realizada (Solórzano et al., 2018).

Una vez realizados los cálculos previamente nombrados se procede a sacar la media y desviación estándar de cada una de las características teniendo, así como resultado las constantes que serán utilizadas en la fórmula de Naive Bayes para poder predecir si un evento es caída o no.

Etapa de entrenamiento

El algoritmo para la etapa de entrenamiento fue desarrollado en el IDE de Python, en donde su objetivo principal es, calcular las constantes de Naive Bayes, las cuales serán utilizadas para realizar las predicciones. En la figura 7 se puede observar el procedimiento seguido.

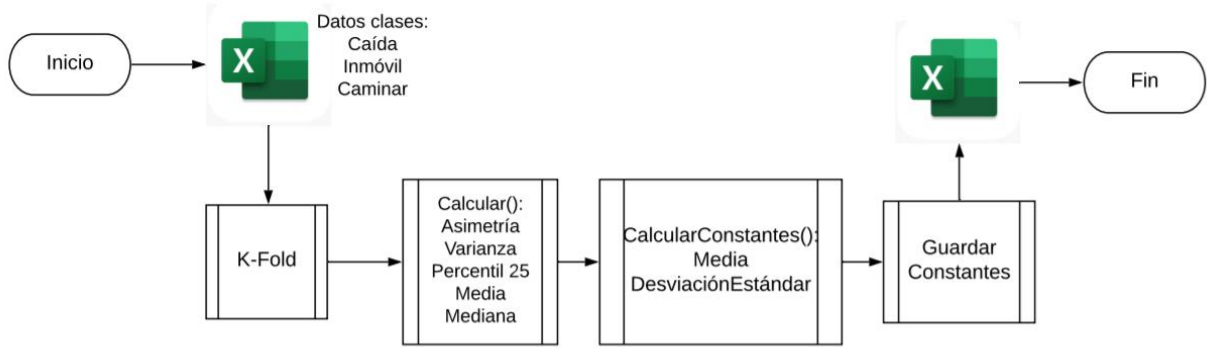


Figura 7. Diagrama de flujo Test

Como se puede observar en el diagrama de flujo se recibe como entrada un archivo Excel únicamente con dos columnas, la primera es de los módulos que se los obtiene del programa en tiempo real que permite construir datasets, el cual será explicado más a detalle a lo largo del capítulo y el número de evento que permite identificar la cantidad de módulos obtenidos en un lapso de, tiempo al realizar una actividad.

Posteriormente, los módulos son procesados por un algoritmo de K-Fold, el cual consiste dividir los dados en k bloques, los cuales sirven para poder particionar la información y dividir el 20% en test y el 80% en entrenamiento.

En K-Fold se optó por utilizar cross - validation ya que este permite tomar valores de diferentes partes e ir haciendo pruebas para poder escoger el que mejor rendimiento tenga dentro del algoritmo y de esta manera el algoritmo tenga las mejores predicciones.

Una vez lista la división de los datos se toma el 80% y se envía a calcular las características de cada uno de los eventos. Posteriormente se realiza el cálculo de las constantes teniendo como resultado 10 constantes por cada clase, dadas por la media y desviación estándar de cada característica, las cuales serán aplicadas en el algoritmo de Naive Bayes para las predicciones de los eventos.

3.5. Implementación del algoritmo para las actividades de la vida diaria ADLs

La implementación del algoritmo tiene dos componentes: la sección que es implementada en el microcontrolador en el IDE de Arduino y la interfaz gráfica en el IDE de Python en donde se puede observar la predicción del algoritmo. A continuación, se detallará el funcionamiento de los programas.

3.5.1 Algoritmo de predicción implementado en el microcontrolador.

El objetivo principal de este programa es poder implementar el algoritmo de Naive Bayes y realizar el análisis de los resultados, para ello se recibe la información en tiempo real del sensor y se calcula las características para ser enviadas al algoritmo de Naive Bayes. Este procesa los datos para obtener la predicción del evento

A continuación, en la figura 8 se explica el funcionamiento del programa.

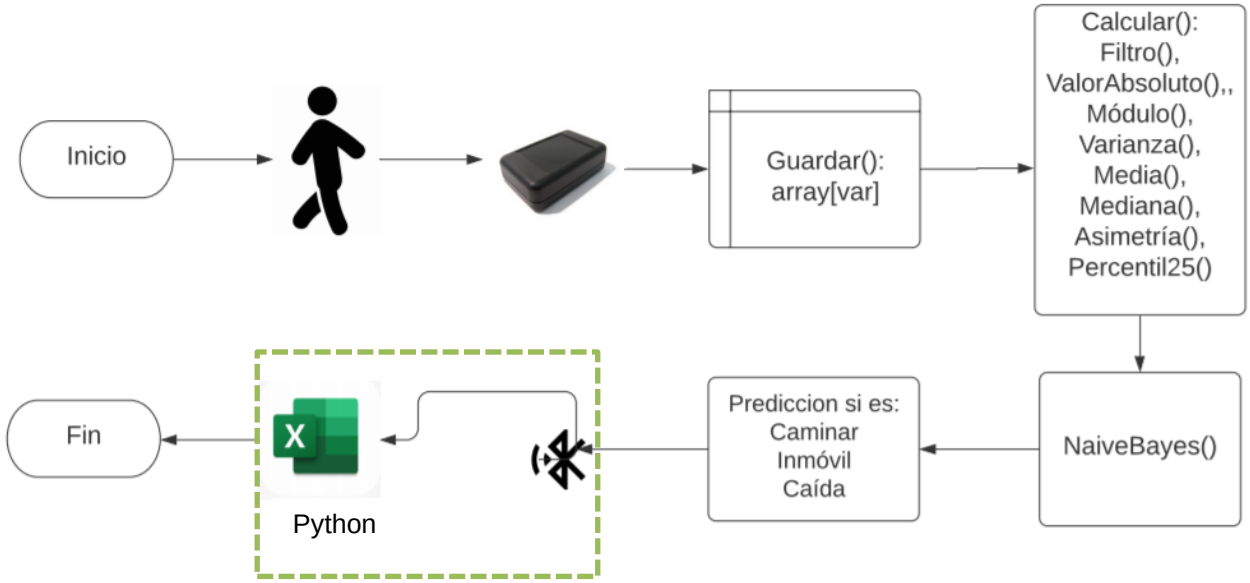


Figura 8. Diagrama de flujo análisis de datos

En el diagrama de flujo se puede observar que, el programa recibe la información de las señales de aceleración en los ejes x, y, z, estos datos son almacenados en arreglos, los cuales son un tipo de dato que almacena una secuencia de datos. El arreglo es de dimensión variable, posteriormente los datos almacenados son enviados a la función calcular, una vez que se tiene los resultados de las características entra al algoritmo de Naive Bayes para conocer la predicción y finalmente por bluetooth se envían los datos a Python donde se los guarda en un archivo Excel. A continuación, se explica as a detalle las funciones:

3.5.1.1 Función Calcular ():

Esta función es utilizada para realizar todos los cálculos de las características. Dentro de esta función se llama a 6 funciones adicionales las cuales son:

- Función CalcularModulo():

  Dentro de esta función se calcula el módulo dado por $\sqrt{x^2 + y^2 + z^2}$ y el resultado lo almacena en otro arreglo par los demás cálculos.

- Función CalcularMedia (dato):
  En esta función recibe como parámetro un arreglo de datos y en ella se realiza el cálculo de la media dada por la suma de los módulos obtenidos en la función anterior dividido para el número total de módulos.
- Función CalcularMediana(dato):
  En esta función recibe como parámetro un arreglo de datos y realiza el cálculo de la mediana dada por dos fórmulas importantes dependiendo si el número total de módulos es par o impar.
  En el caso de ser par se toma el número que este en la mitad del arreglo y en el caso de ser impar lo que se hace es sumar los números de la posición que este en la mitad y ese número menos uno y dividir para 2.
- Función CalcularVarianza (dato):
  Recibe como parámetro un arreglo de datos y calcula la varianza dada por la raíz cuadrada de la resta entre el módulo y la media calculada previamente.
- Función CalcularPercentil(dato):

Recibe como parámetro un arreglo de datos y para este cálculo se tiene el módulo por el percentil dividido para 100

- Función CalcularAsimetria (dato):

Recibe como parámetro un arreglo de datos y para el cálculo se utiliza los módulos elevados al cubo dividido para la desviación.

3.5.1.2 Naive Bayes:

Para la implementación de Naive Bayes se tiene dos principales funciones la primera que es:

- Función probabilidadFeature(dato,med,des):

Esta función recibe como parámetros una característica de las antes mencionadas (media, mediana, varianza, asimetría, percentil25) ya calculada de los datos en tiempo real, y las constantes previamente obtenidas en la etapa de entrenamiento. Para el cálculo se aplica la fórmula de la probabilidad dada por:

$$probabilidad = \frac{1}{\sqrt{2\pi * desviacion^2}} * exp\left(\frac{muestra - media}{2 * varianza}\right)$$

Donde:

Desviación: constante realizada en el programa offline

Media: constante realizada en el programa offline

Muestra: dato recibido en tiempo real

Varianza: cuadrado de la desviación.

- Función probabilidadClase(datos):

Esta función recibe como parámetro un arreglo de datos que contiene las características (media, mediana, varianza, asimetría, percentil25) previamente calculadas de los datos obtenidos en tiempo real. Dentro de esta función se realiza el cálculo de la probabilidad de cada una de las clases dada por la fórmula:

$$P(C_x) = \frac{Pbx * P(F1) * P(F2) * P(F3) * P(F4) * P(F5)}{evidencia}$$

$$evidencia = P(C_1 + C_2 + C_3)$$

Donde:

P(Cx) :es la probabilidad de cada clase.

Pbx: es la probabilidad de que suceda ese evento en este caso dado que se tiene 3 clases es de 0.33

P(F_x): es la probabilidad de cada una de las características (media, mediana, varianza, asimetría, percentil25), calculada con la fórmula anterior.

Evidencia :es la suma de las probabilidades de cada clase

Una vez calculadas las probabilidades de cada clase se realiza la comparación de entre las tres para determinar cuál de ellas es la mayor y así dar un resultado de predicción del evento obtenido.

3.5.2 Interfaz en Python para análisis de resultados

Este programa permite adquirir los resultados de la predicción calculados en el microcontrolador para el análisis por parte del usuario.

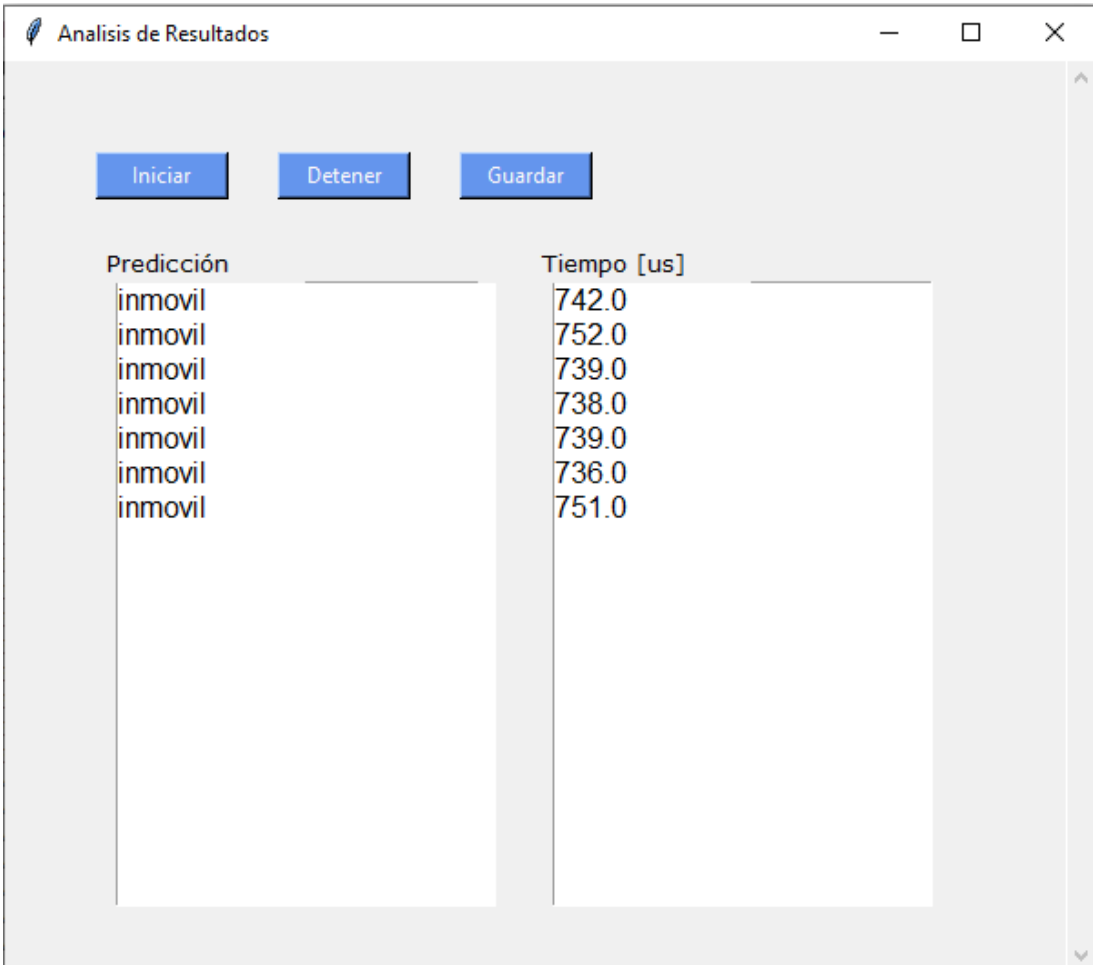


Figura 9. Interfaz análisis de resultados

3.6 Interfaz para la creación de base de datos (data sets)

Finalmente, se establece la interfaz para poder crear la base de datos, la cual consiste en recolectar los datos e irlos almacenando y graficando en tiempo real. Una vez que se detenga la lectura de datos estos se podrán guardar en un archivo “.csv” en donde se almacenarán los datos de los ejes x, y, z además de también guardar el cálculo del

módulo; de esta manera también se puede leer el dataset guardado y volverlo a graficar. Esta interfaz nos ayuda a que se pueda utilizar la información guardada de los módulos en el programa offline para sacar las constantes de Naive Bayes.

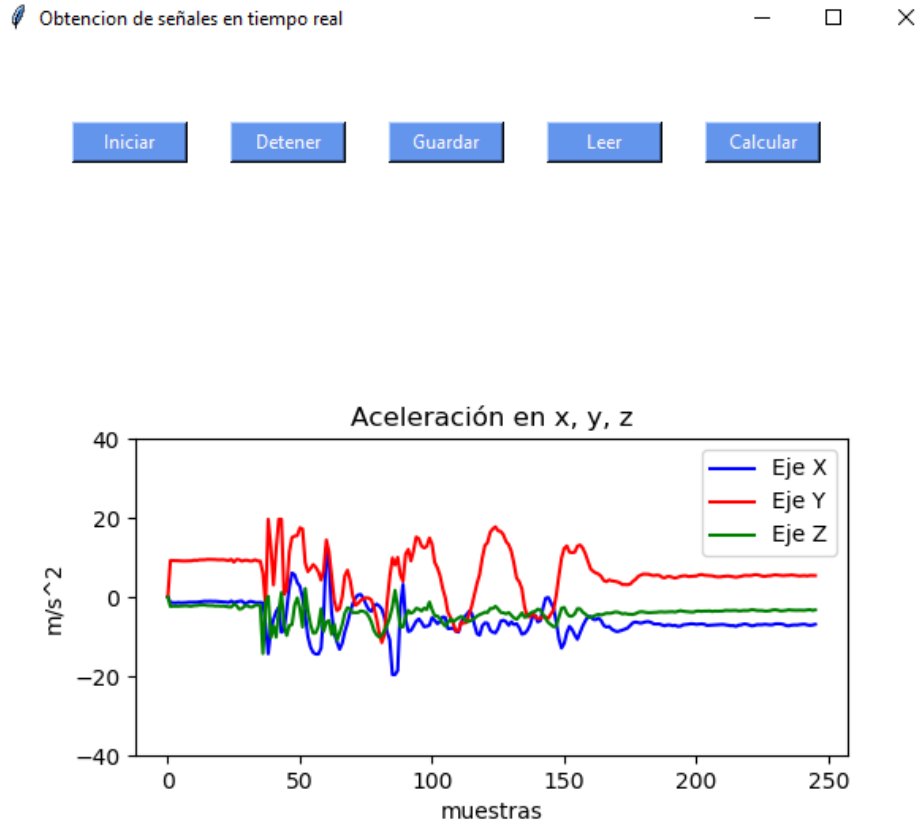


Figura 10. Interfaz para la creación de base de datos

3.6.1 Algoritmo para la obtención de señales en tiempo real en Arduino.

Para la obtención de las señales en tiempo real, se tiene dos programas: el primero que se encuentra en el IDE Arduino en donde se tiene las funciones que se muestran en la figura 11.

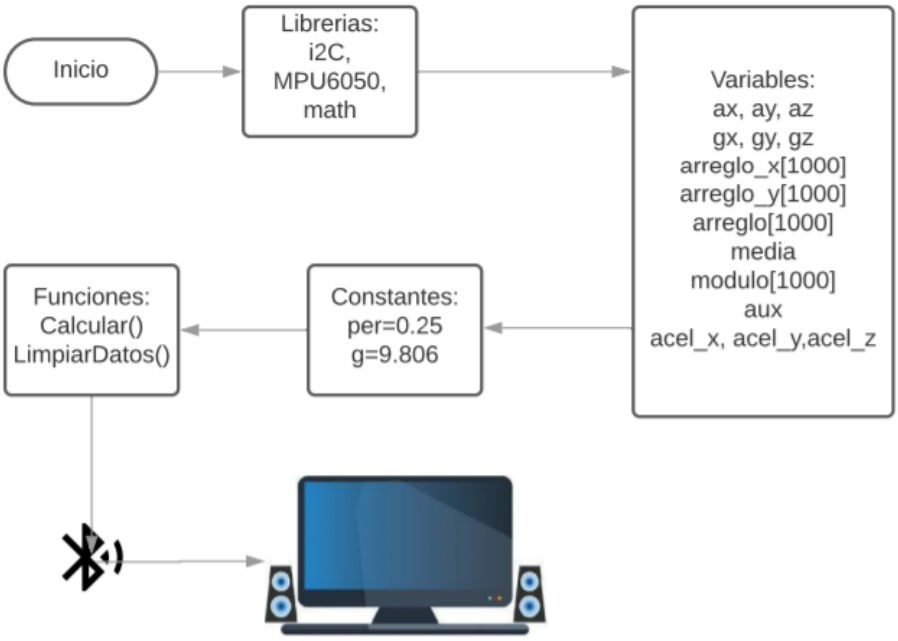


Figura 11. Diagrama de flujo del programa de señales en tiempo real

Como se muestra en la figura previa, el diagrama muestra claramente las librerías a utilizar como las variables y constantes, además de las principales funciones que serán utilizadas para poder obtener los resultados deseados.

- Función: Setup

  Esta es la función que se ejecuta primero al correr el programa, aquí se realiza la inicialización de la comunicación serial además de tener la verificación de la conexión e inicialización del sensor. A continuación, se explica de manera breve los comandos dentro de la función.

  “Serial.begin(115200)”

  “accelgyro.initialize()”

Donde:

Serial: Puerto serial (Arduino, 2019)

Begin: Método para inicializar el enlace (Arduino, 2019)

Accelgyro.initialize: inicialización del sensor inercial.

- Función: Loop ()

Esta función permite obtener las señales del sensor en tiempo real, en donde comenzará a guardar los datos obtenidos siempre y cuando se inicialice con el botón de la interfaz en Python. Se lo puede ver en figura 12 a continuación.

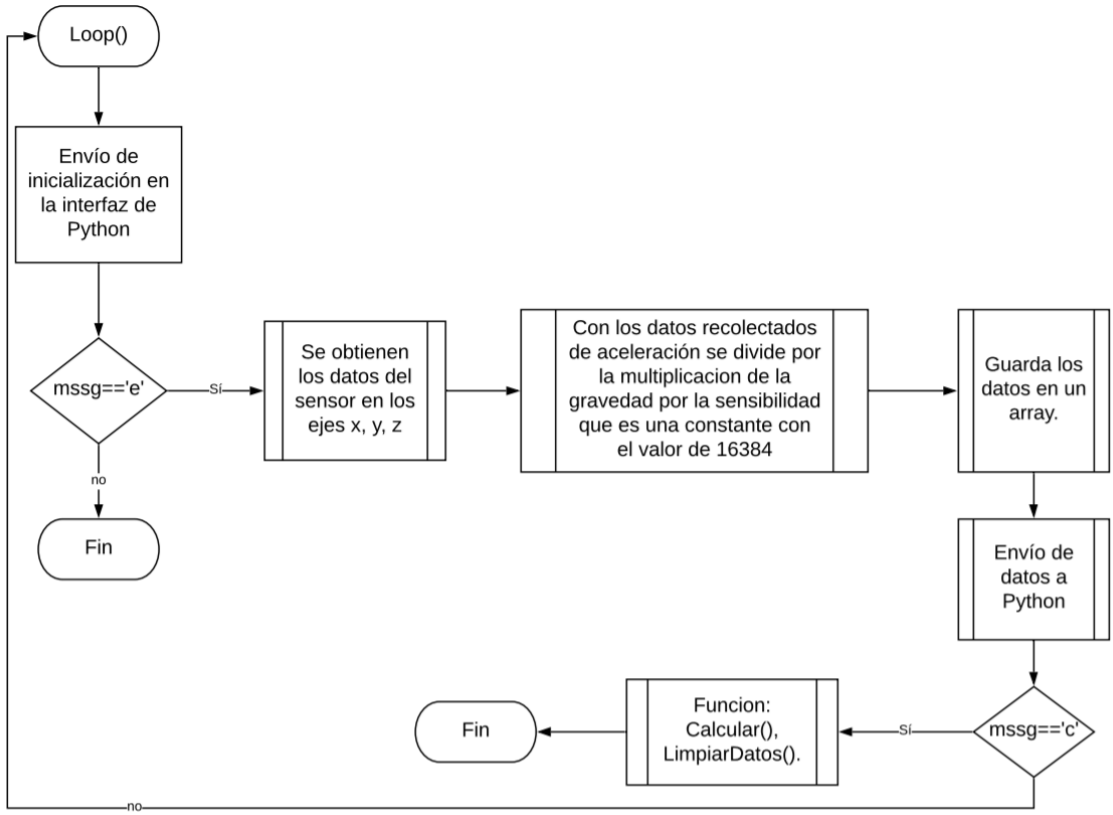


Figura 12. Función Loop ()

- Función calcular ():

Dentro de esta función se realiza el cálculo del módulo y de las características de las clases cuyo cálculo se encuentra en diferentes funciones las cuales se explicarán a continuación.

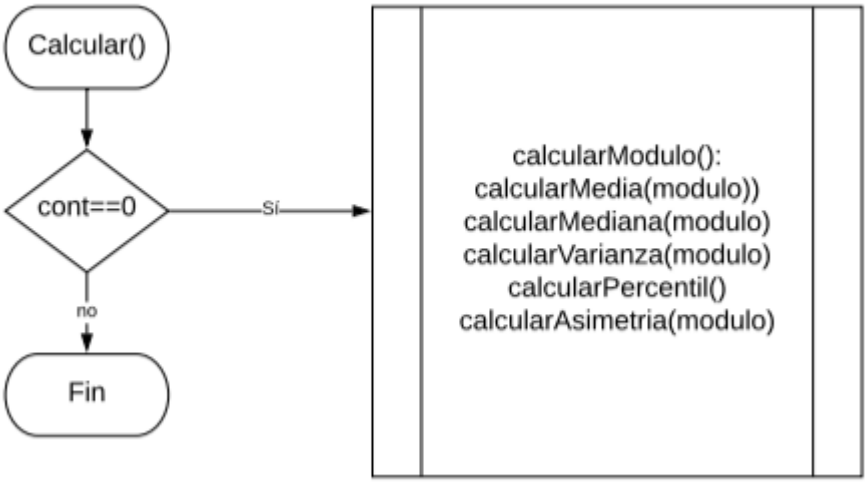


Figura 13. Función Calcular

3.6.2 Algoritmo para la obtención de señales en tiempo real en Python.

Se desarrolló la interfaz gráfica dentro del IDE de Python en donde se muestra la gráfica de en tiempo real de los ejes x, y, z. Además, en esta interfaz se puede guardar la información tanto de los ejes como el módulo que se calculó previamente en Arduino. A continuación, se puede observar brevemente en el diagrama de flujo la composición del código.

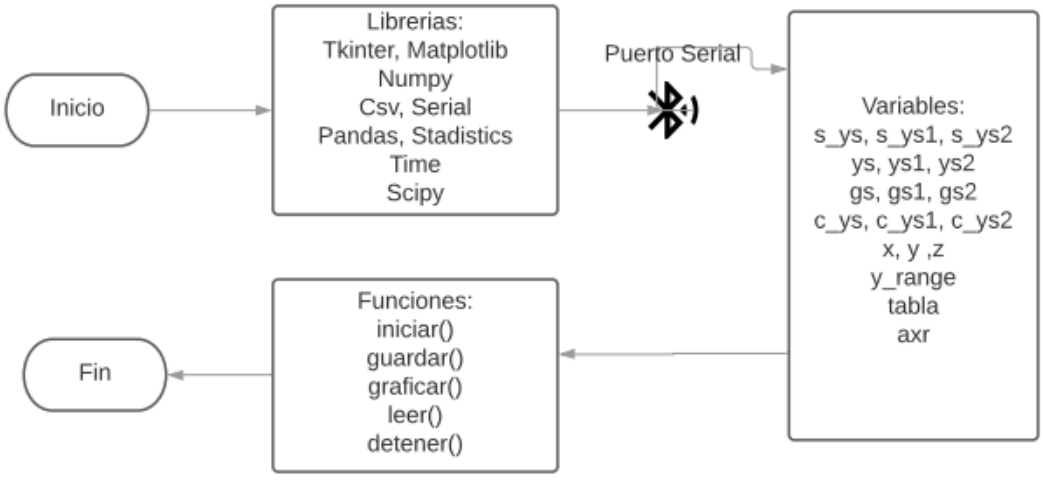


Figura 14. Diagrama de flujo de las señales en tiempo real.

- Función Iniciar() :

  En la función iniciar se envía la letra “e” a Arduino por medio del puerto serial para que comience la recolección de datos y pueda ir graficando en tiempo real.

- Función guardar():

  El objetivo de esta función es almacenar los datos dentro de un archivo .csv el cual tendrá los datos en los ejes x, y z y el módulo para poder sacar las constantes dentro del ambiente de test.

4. ANÁLISIS DE RESULTADOS

En este capítulo se analiza los resultados obtenidos tras la implementación del prototipo detector de caídas. Para ello se tiene un grupo de seis personas las cuales realizarán las actividades solicitadas (caminar, caerse) mientras utilizan el prototipo en la cadera.

Para las pruebas, se configurará al sistema microcontrolado en seis ventanas de 130, 150, 170,190, 210, 230 tomando en cuenta las siguientes variables

- **Caminar:** Número de predicciones caminando
- **Inmóvil:** Número de predicciones inmóvil
- **Caer:** Número de predicciones al caer

4.1 Entrenamiento del clasificador

El proceso de entrenamiento de Naive Bases implementado en Arduino, consiste en tomar el 80% de la base de datos la información de cada evento para posteriormente sacar las características.

Finalmente, se calcula los valores de media y desviación estándar de los resultados de las características sacadas previamente para poder obtener las características del algoritmo y posteriormente se envía el 20% de la base restante para comprobar que el algoritmo para comprobar su funcionamiento.

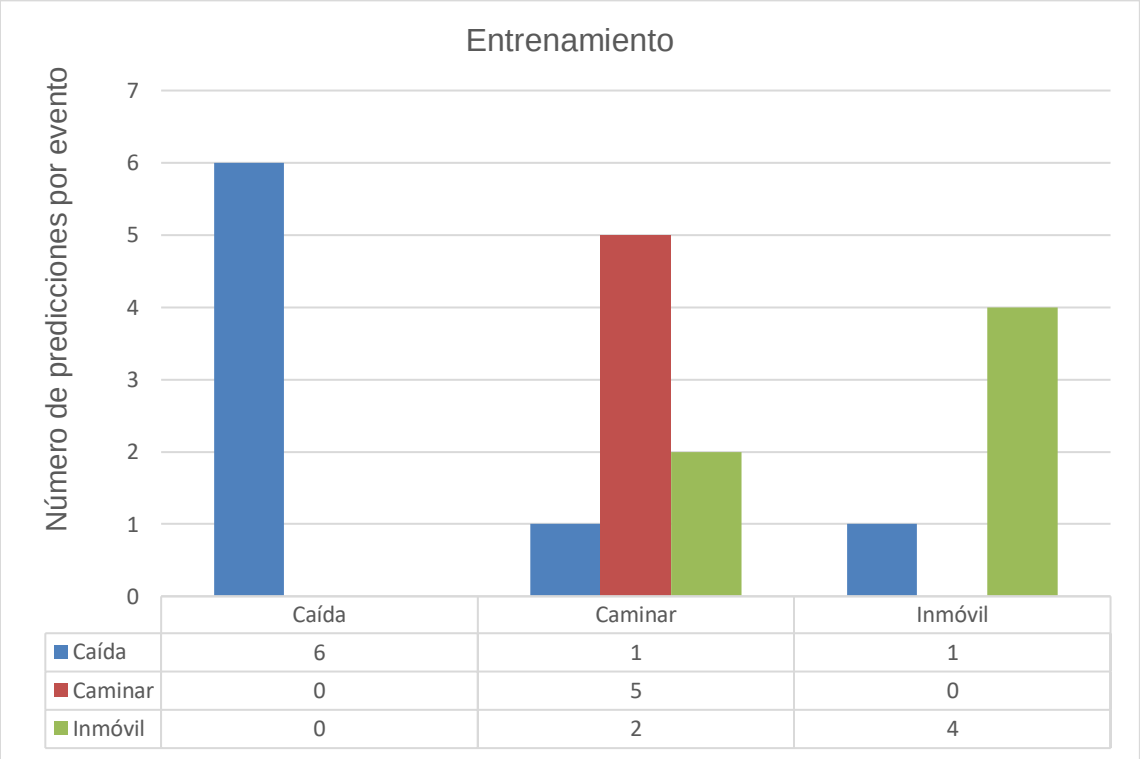


Figura 15. Entrenamiento test

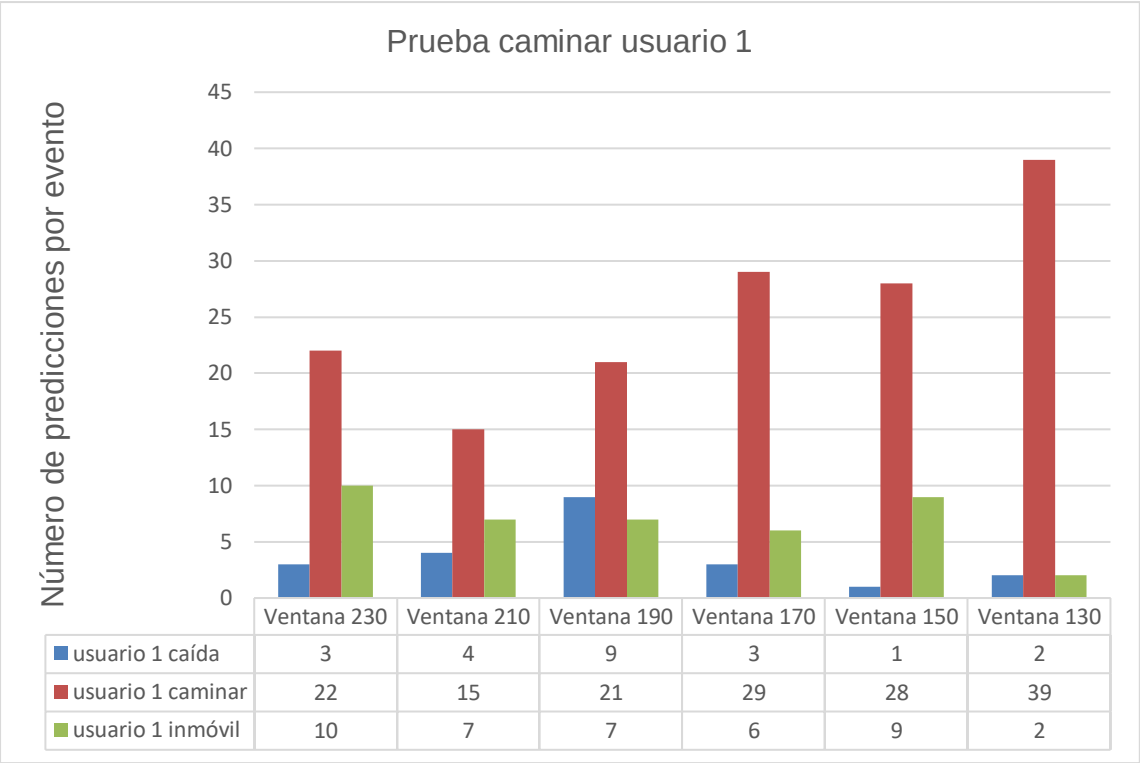
4.2 Pruebas con usuarios.

Las pruebas consistieron en recolectar seis ventanas mientras se realizaba la actividad a predecir. Al final de la recolección de datos, el microcontrolador procede a enviar la predicción a la interfaz de mostrar datos y estos se guardarán en un documento Excel en donde se tiene la predicción y el tiempo en el que el algoritmo se demoró en procesar.

Las evaluaciones fueran aplicadas a seis personas en caminar, seis personas en caídas y tres personas en inmóvil, con el objetivo de poder comprobar el rendimiento dependiendo de la ventana escogida.

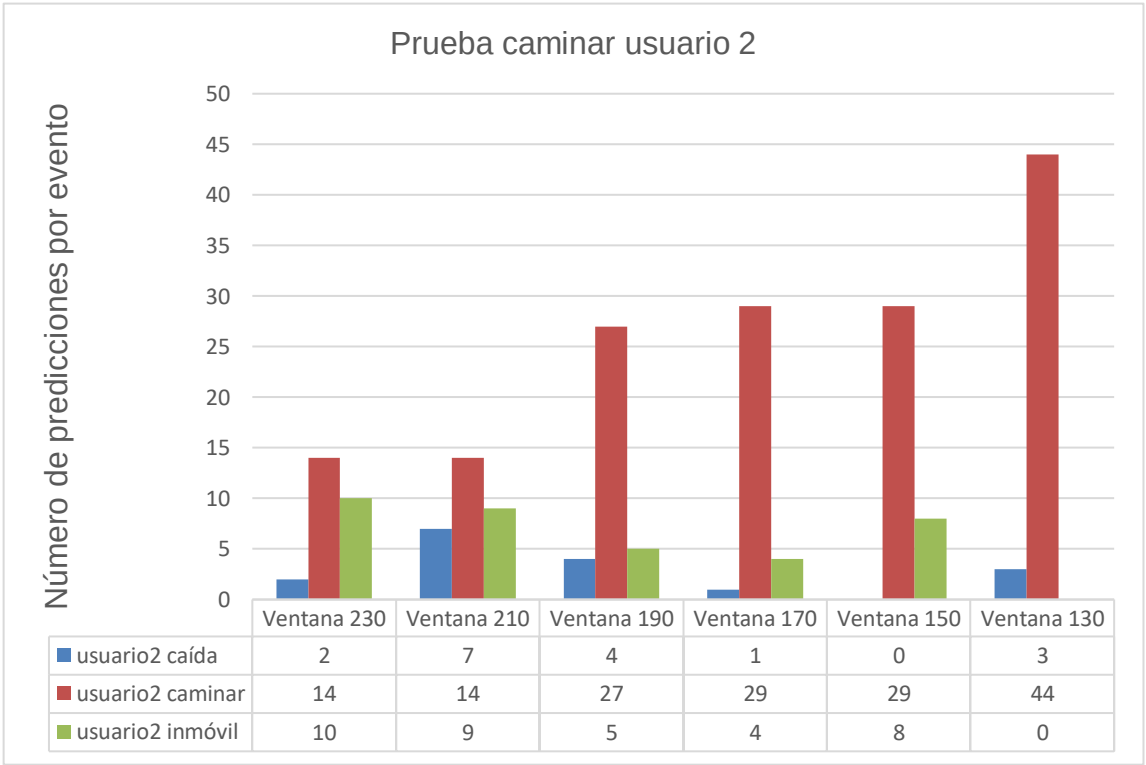
4.2.1 Caminar

En las Figuras 16,17,18,19,20,21 y 22 se puede observar la representación gráfica de las pruebas realizadas con el evento de caminar en ventanas de 130, 150, 170, 190, 210, 230. En este caso se toma como acierto el evento de caminar y como fallido el evento inmóvil y caer. Las tablas que permitieron realizar las figuras se las puede encontrar en el ANEXO 3.



	Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
usuario 1 caída	3	4	9	3	1	2
usuario 1 caminar	22	15	21	29	28	39
usuario 1 inmóvil	10	7	7	6	9	2

Figura 16. Prueba Caminar Usuario 1



	Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
usuario2 caída	2	7	4	1	0	3
usuario2 caminar	14	14	27	29	29	44
usuario2 inmóvil	10	9	5	4	8	0

Figura 17. Prueba Caminar Usuario 2

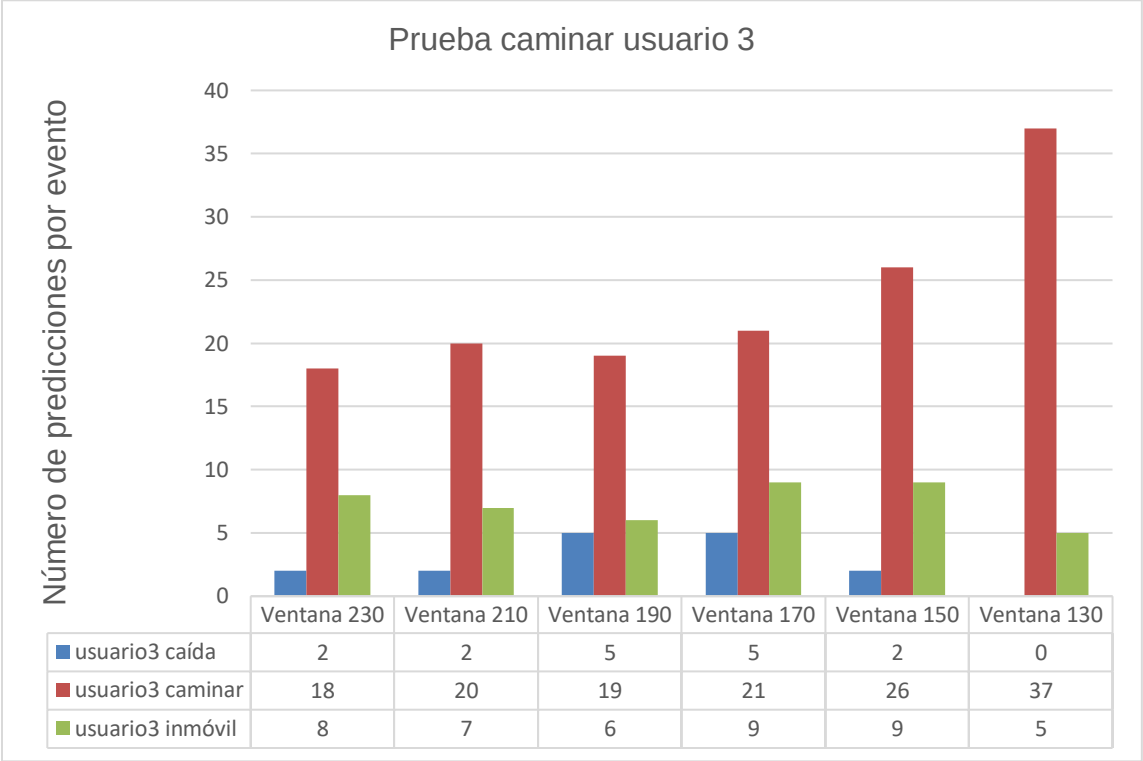


Figura 18. Prueba Caminar Usuario 3

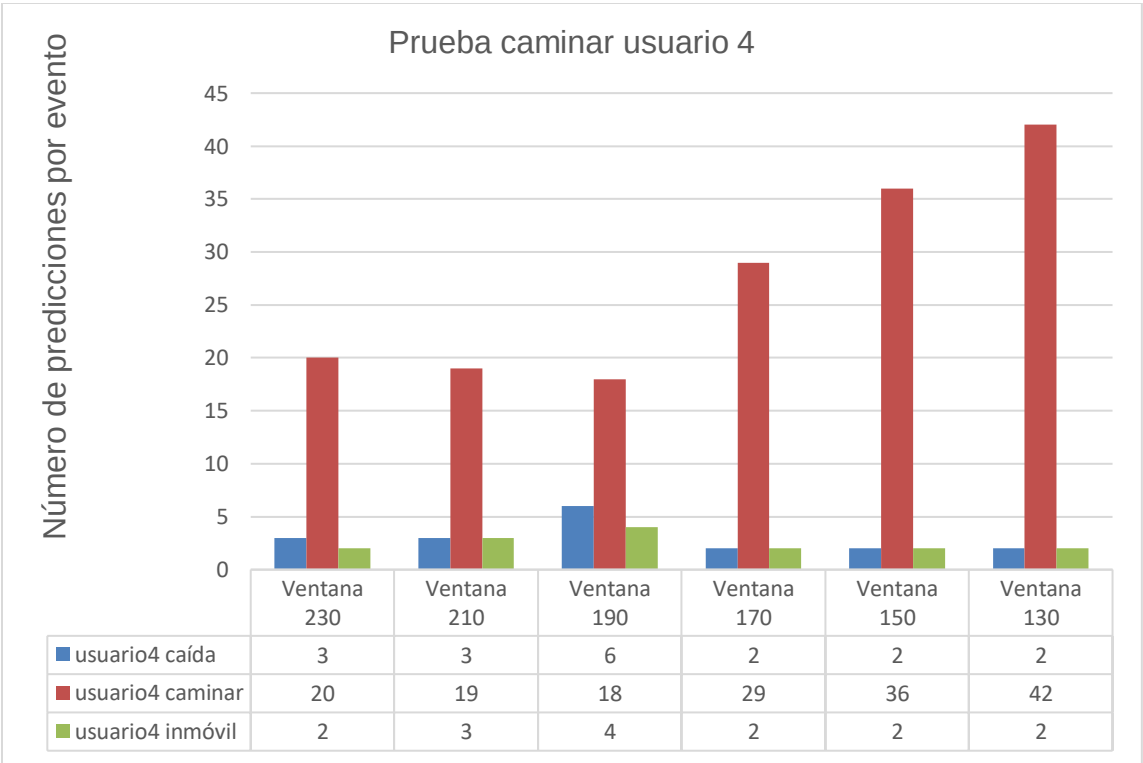


Figura 19. Prueba Caminar Usuario 4

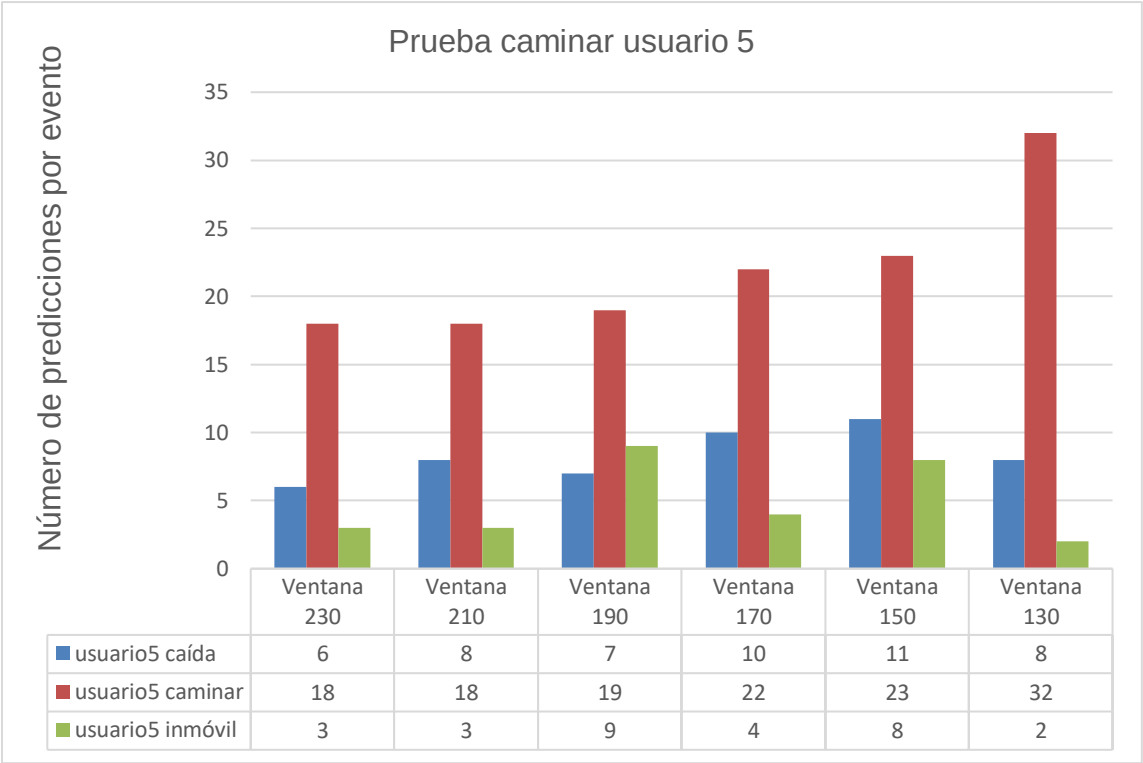


Figura 20. Prueba Caminar Usuario 5

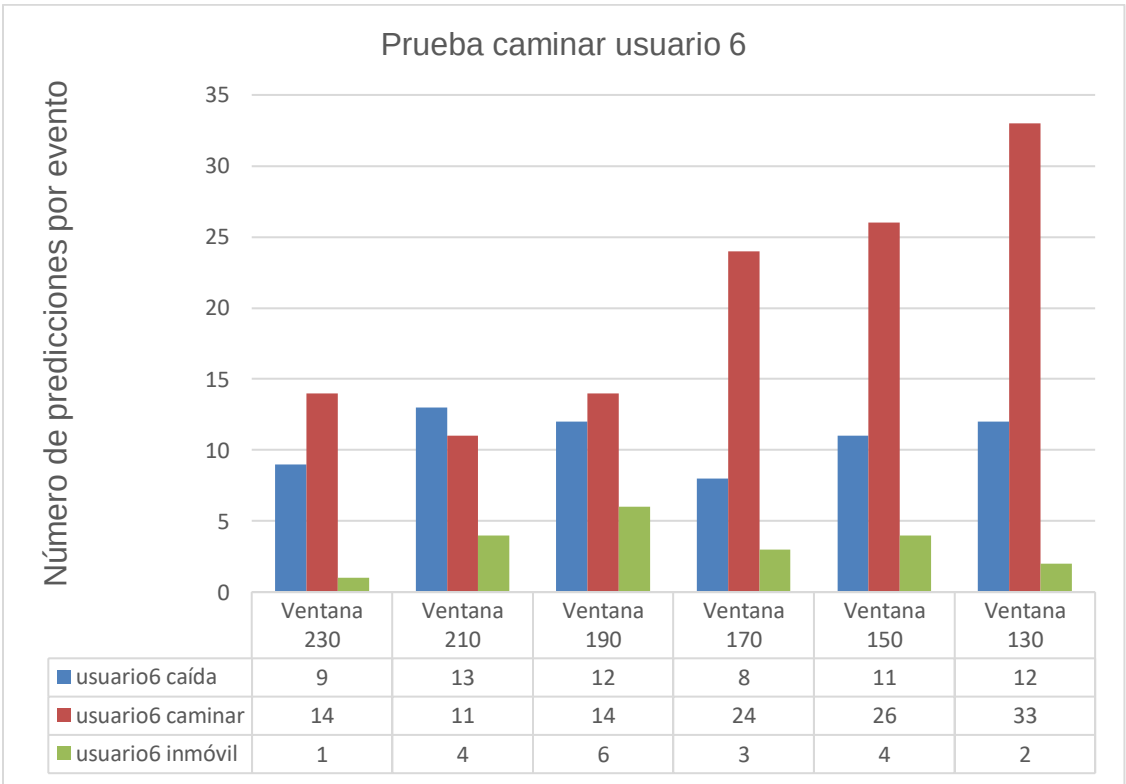


Figura 21. Prueba Caminar Usuario 6

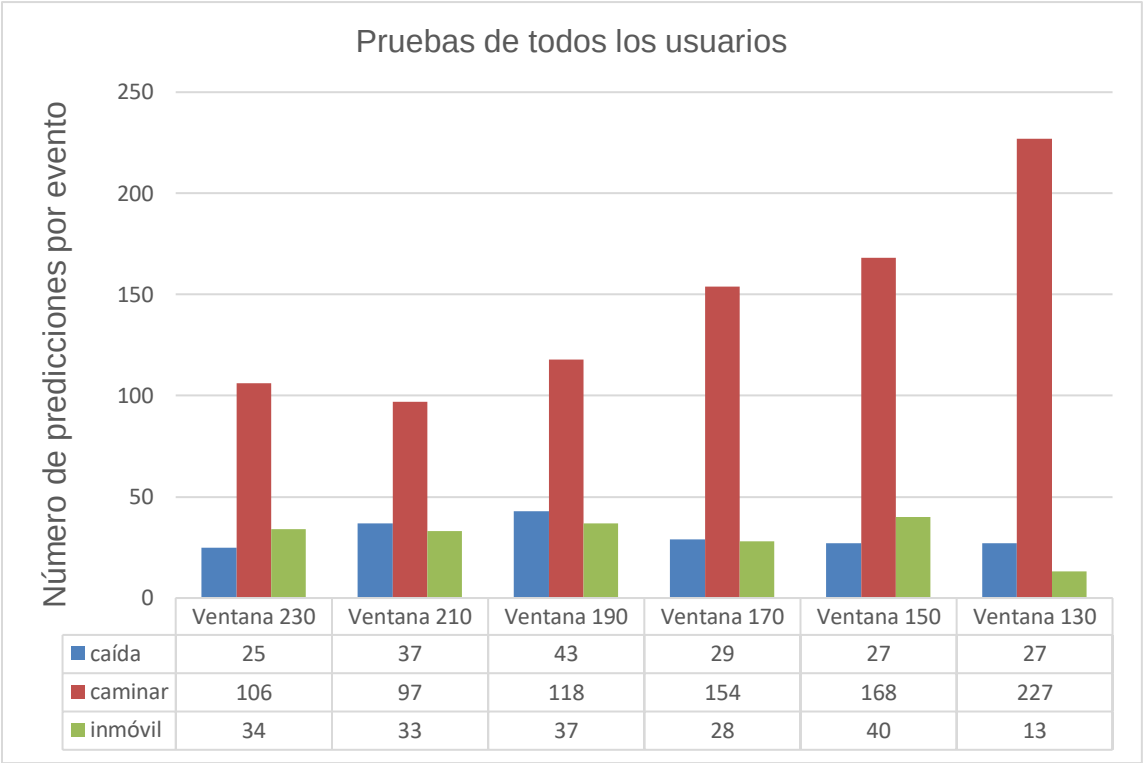


Figura 22. Pruebas total usuarios

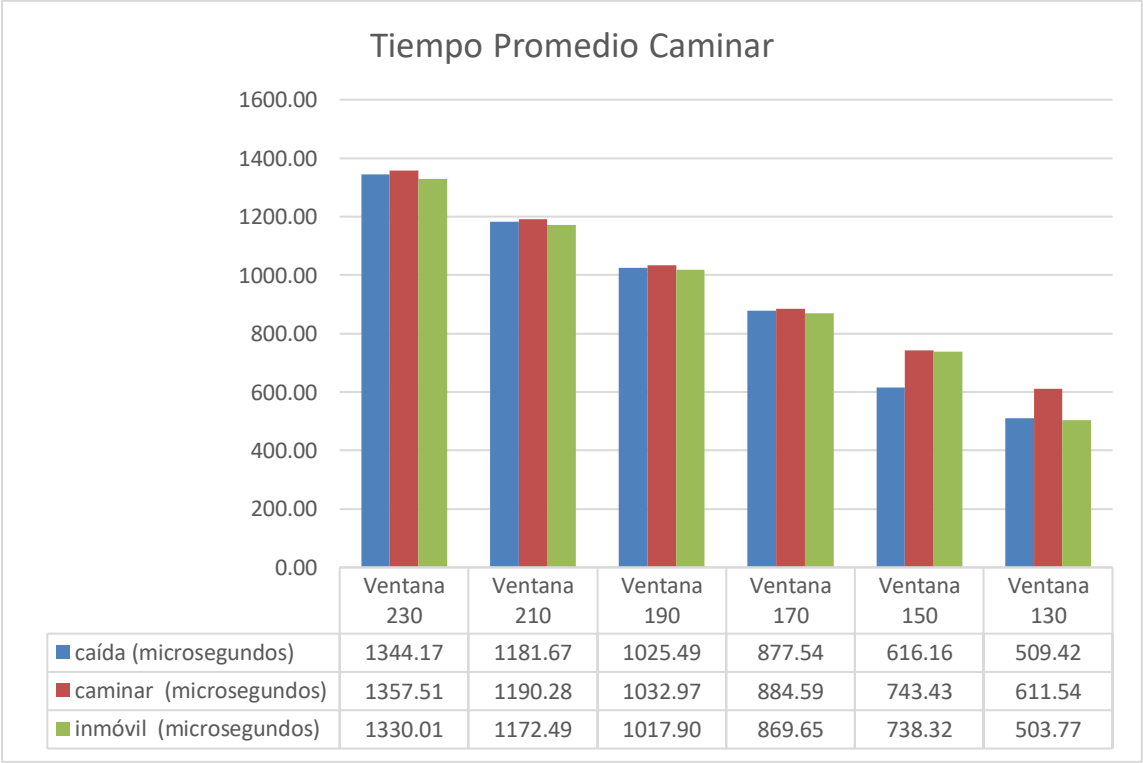
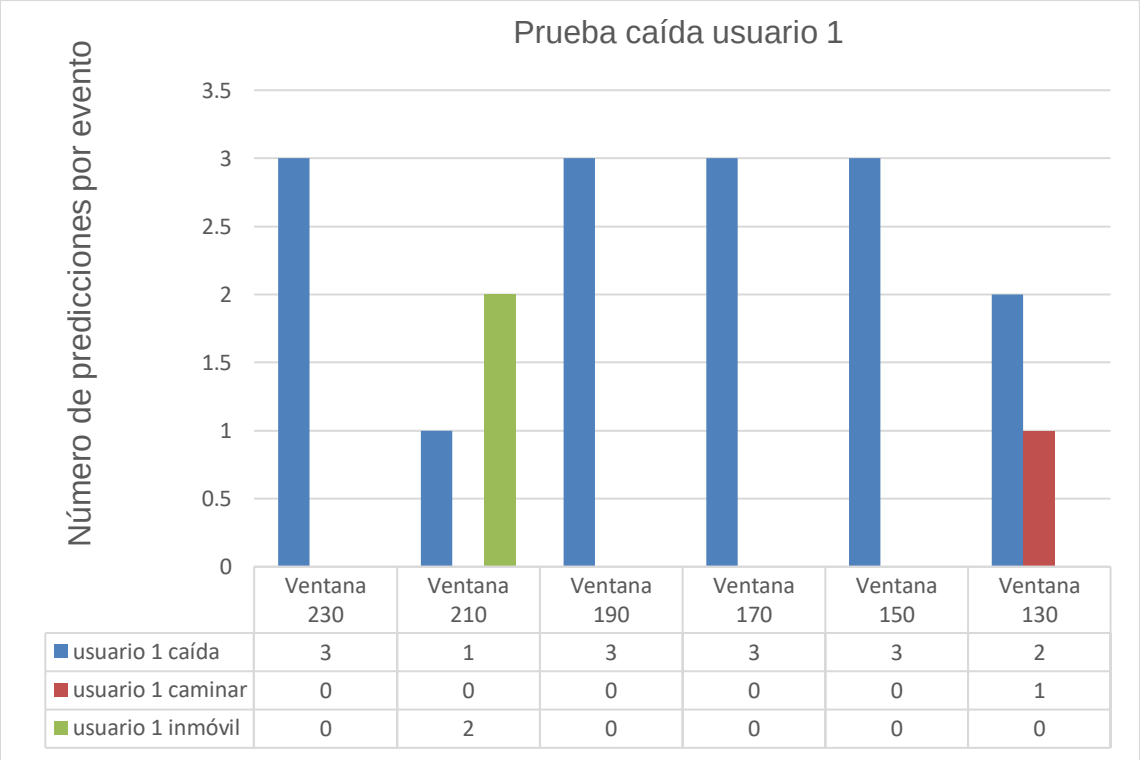


Figura 23. Tiempo promedio en las pruebas del evento caminar.

4.2.1 Caída

En las Figuras 24, 25, 26,27,28,29, 30, 31 se puede observar la representación gráfica de las pruebas realizadas con el evento de caída en ventanas de 130, 150, 170,190, 210, 230. En este caso se toma como acierto el evento de caída y como fallido el evento inmóvil y caminar. Las tablas que permitieron realizar las figuras se las puede encontrar en el ANEXO 4



	Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
usuario 1 caída	3	1	3	3	3	2
usuario 1 caminar	0	0	0	0	0	1
usuario 1 inmóvil	0	2	0	0	0	0

Figura 24. Prueba caída Usuario 1

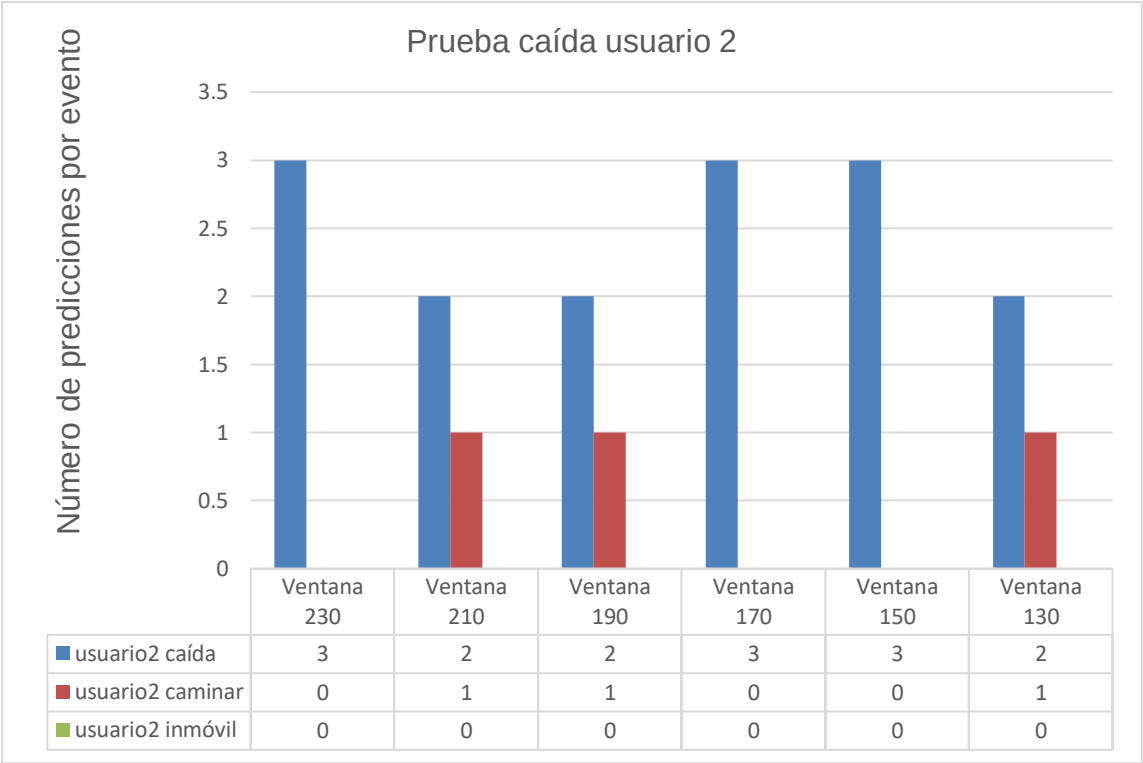


Figura 25. Prueba caída Usuario 2

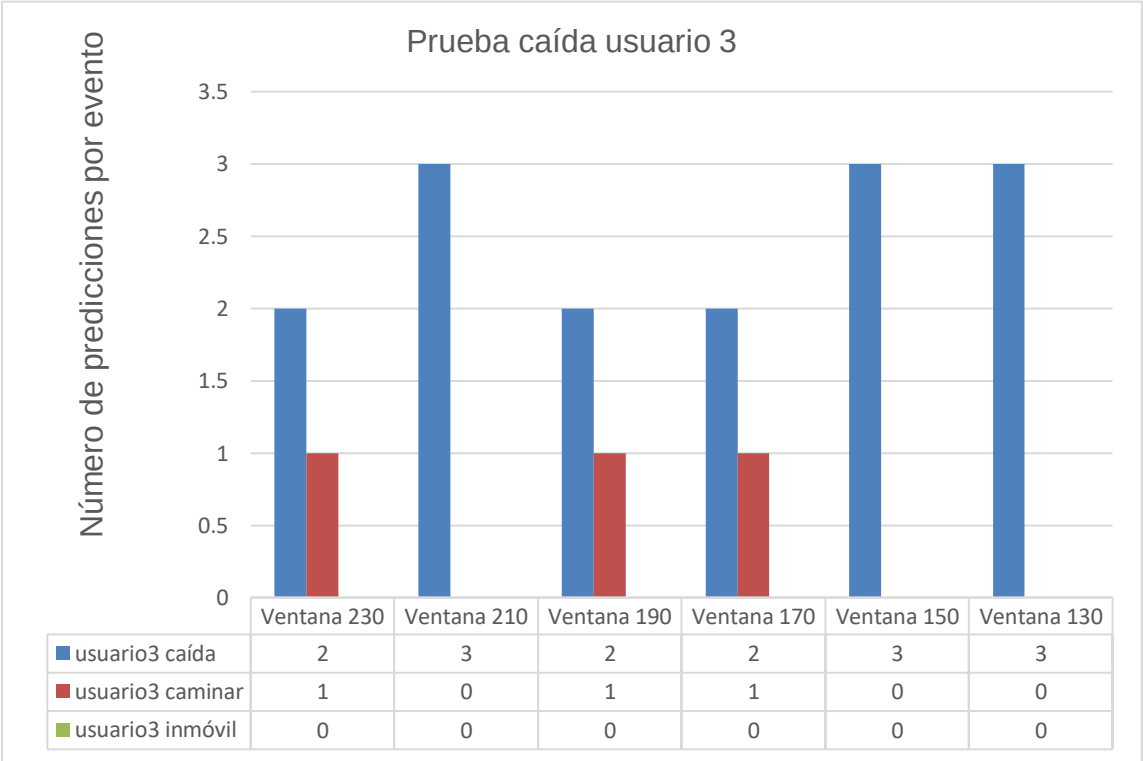


Figura 26. Prueba caída Usuario 3

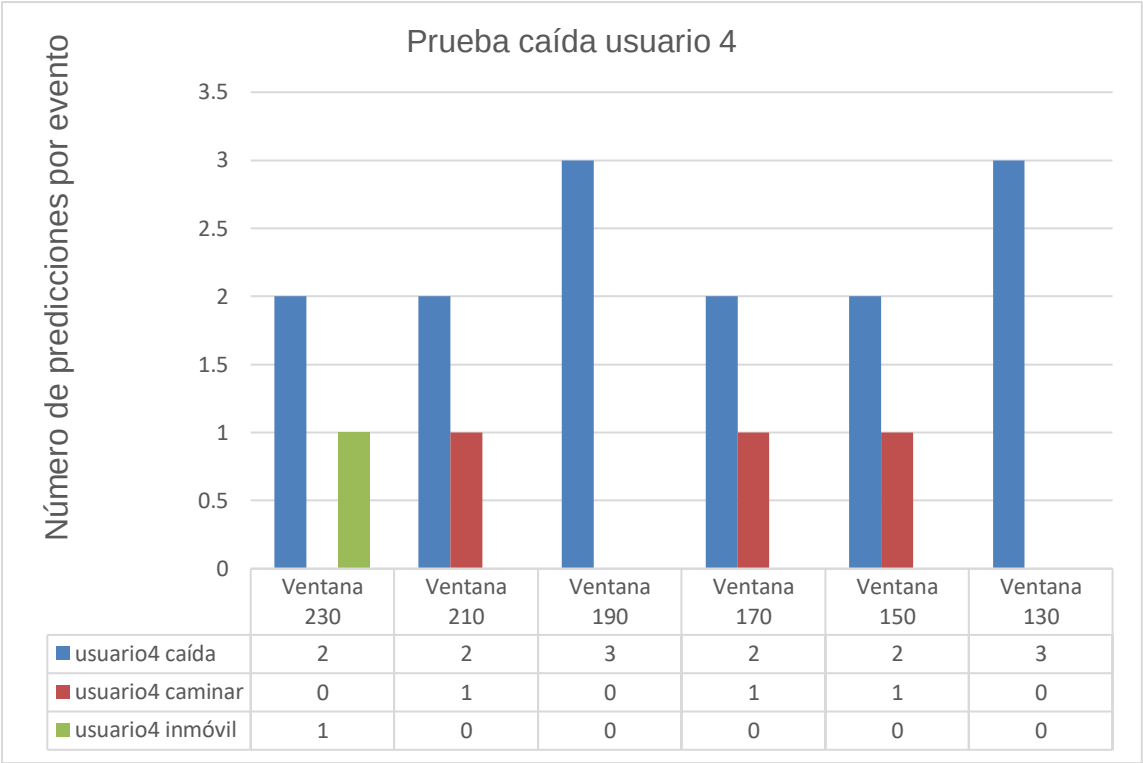


Figura 27. Prueba caída Usuario 4

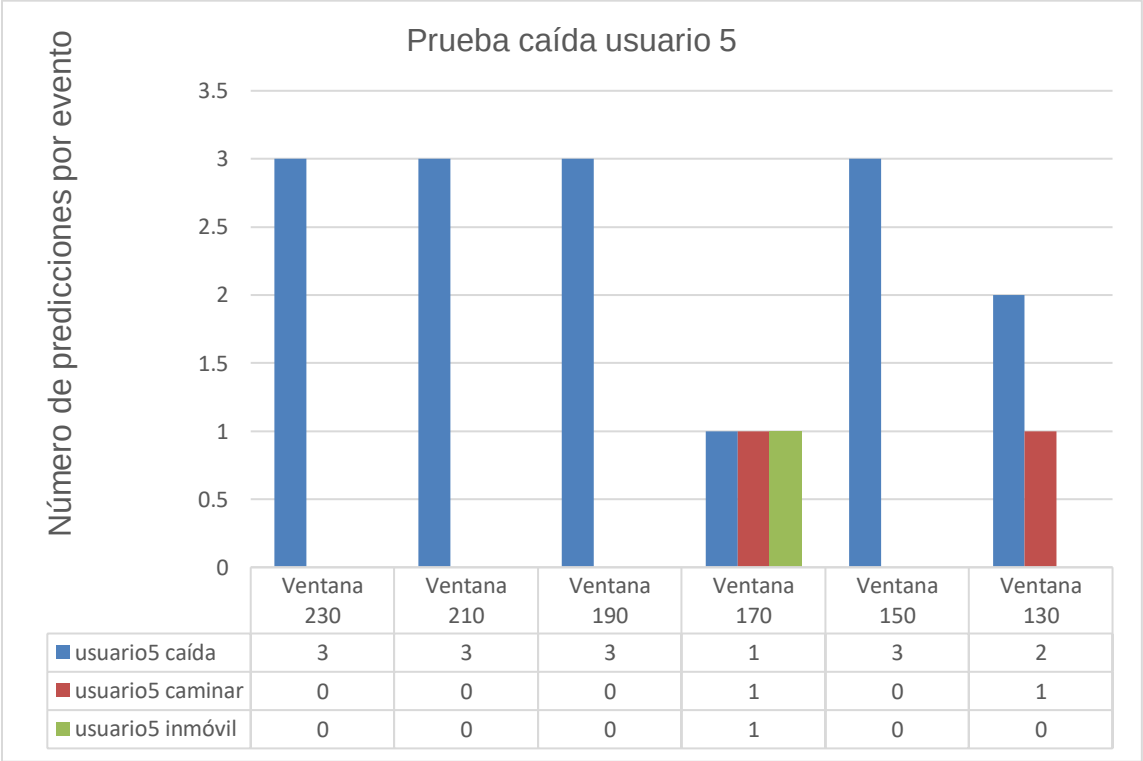


Figura 28. Prueba caída Usuario 5

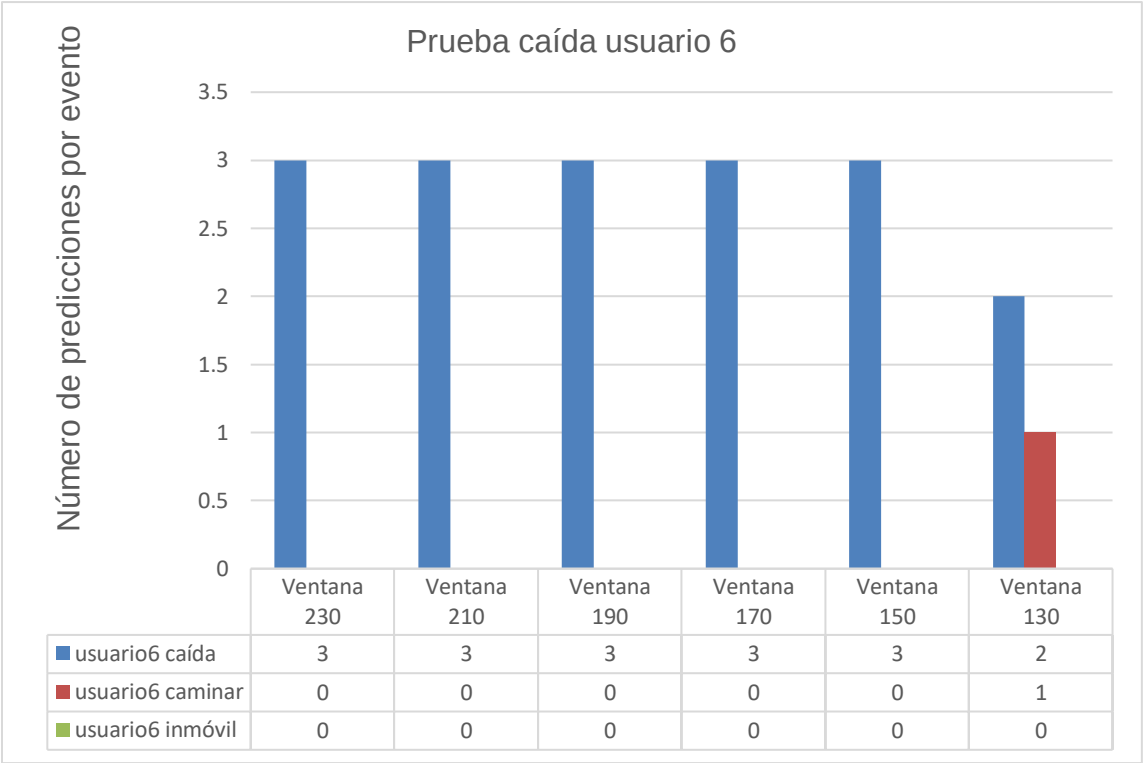


Figura 29. Prueba caída Usuario 6

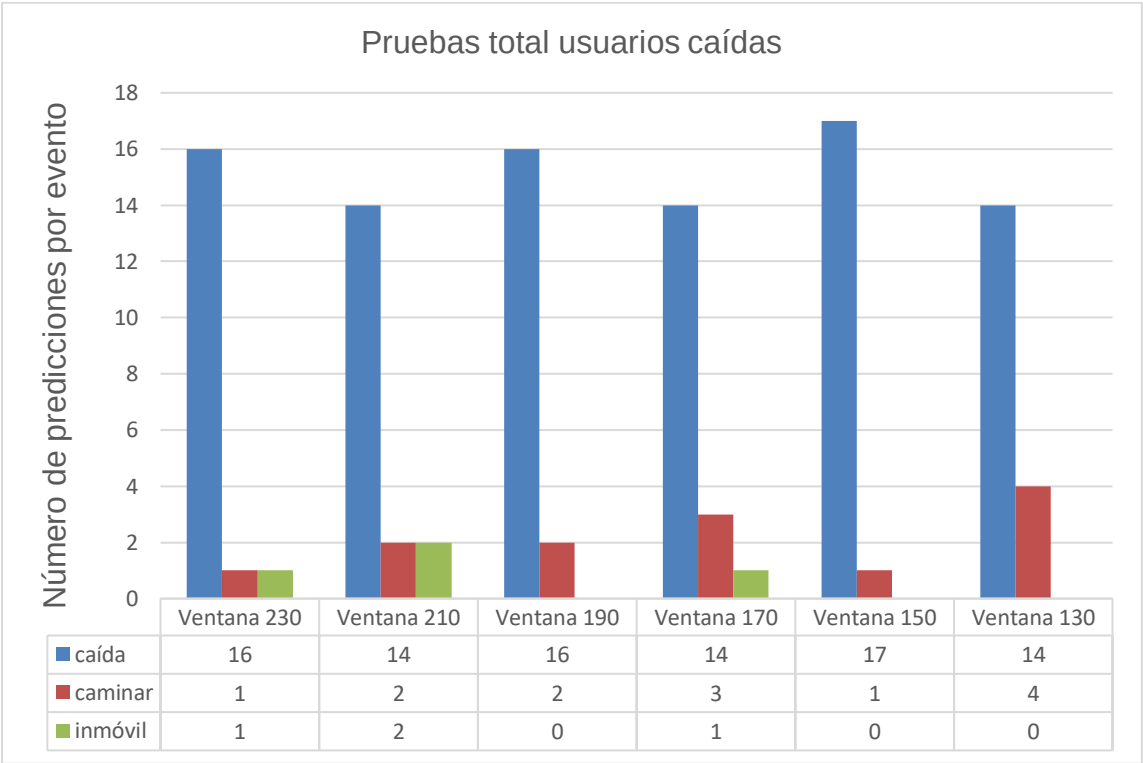
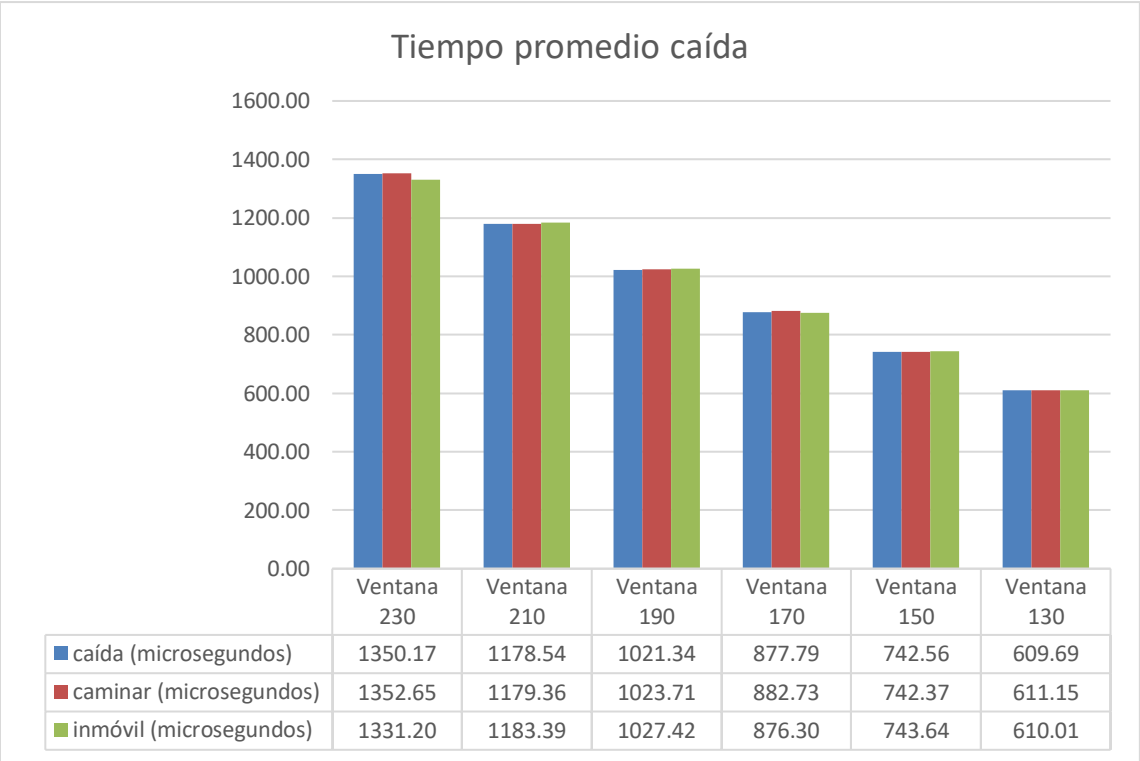


Figura 30. Pruebas total usuarios caídas

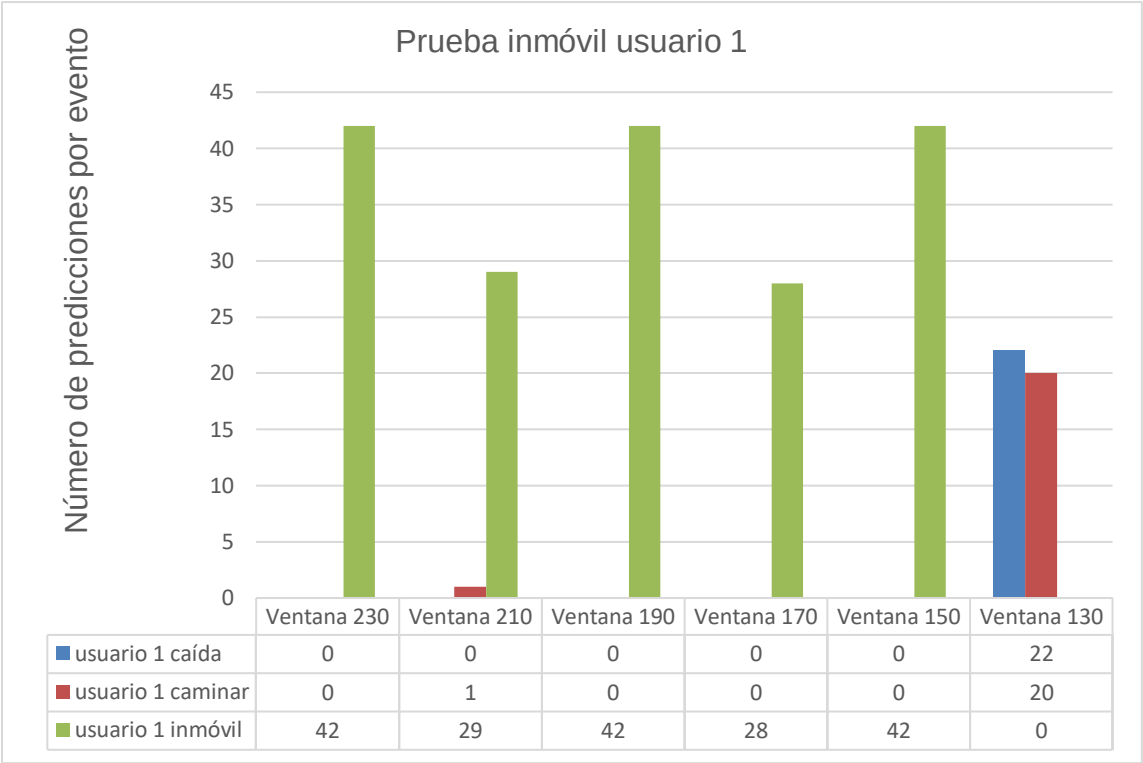


	Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
caída (microsegundos)	1350.17	1178.54	1021.34	877.79	742.56	609.69
caminar (microsegundos)	1352.65	1179.36	1023.71	882.73	742.37	611.15
inmóvil (microsegundos)	1331.20	1183.39	1027.42	876.30	743.64	610.01

Figura 31. Tiempo promedio en las pruebas del evento caída

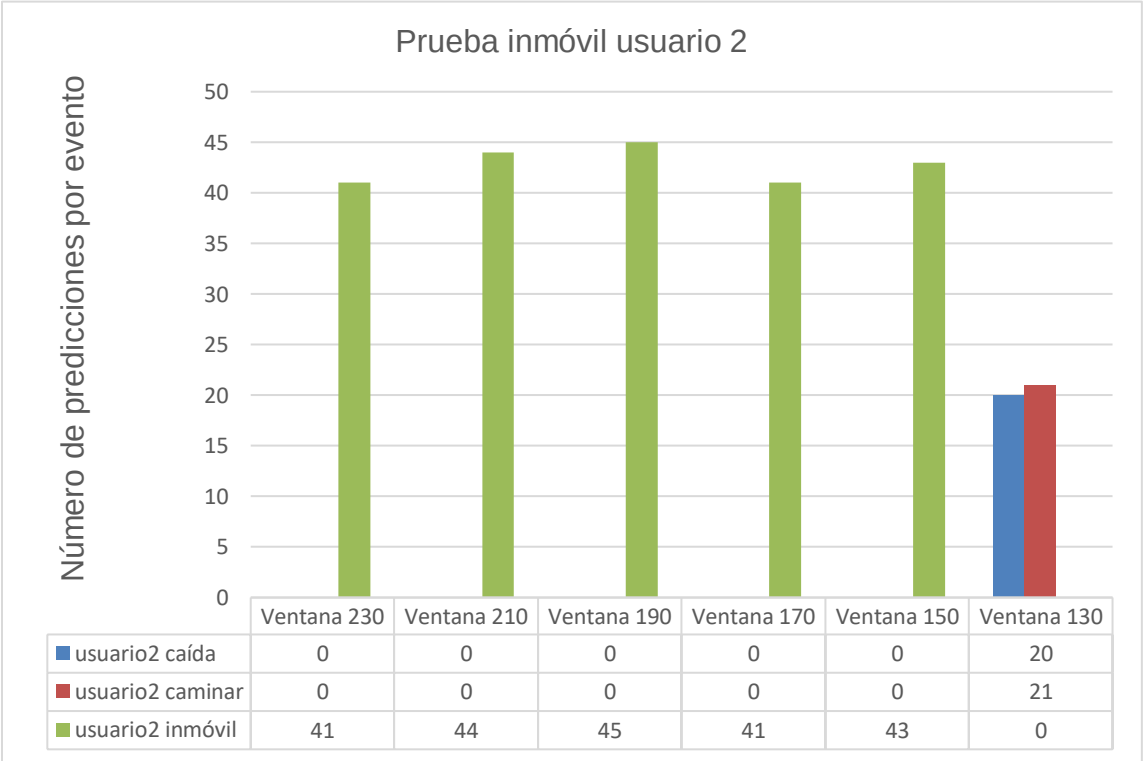
4.2.2 Inmóvil

En las Figuras 31, 32, 33, 34, 35 se puede observar la representación gráfica de las pruebas realizadas con el evento inmóvil con las seis ventanas previamente planteadas. Del mismo modo, como evento acierto se tomará el evento de inmóvil y como evento fallido se tomarán los eventos caminar y caída. ANEXO 5



	Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
usuario 1 caída	0	0	0	0	0	22
usuario 1 caminar	0	1	0	0	0	20
usuario 1 inmóvil	42	29	42	28	42	0

Figura 32. Prueba inmóvil Usuario 1



	Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
usuario2 caída	0	0	0	0	0	20
usuario2 caminar	0	0	0	0	0	21
usuario2 inmóvil	41	44	45	41	43	0

Figura 33. Prueba inmóvil Usuario 2

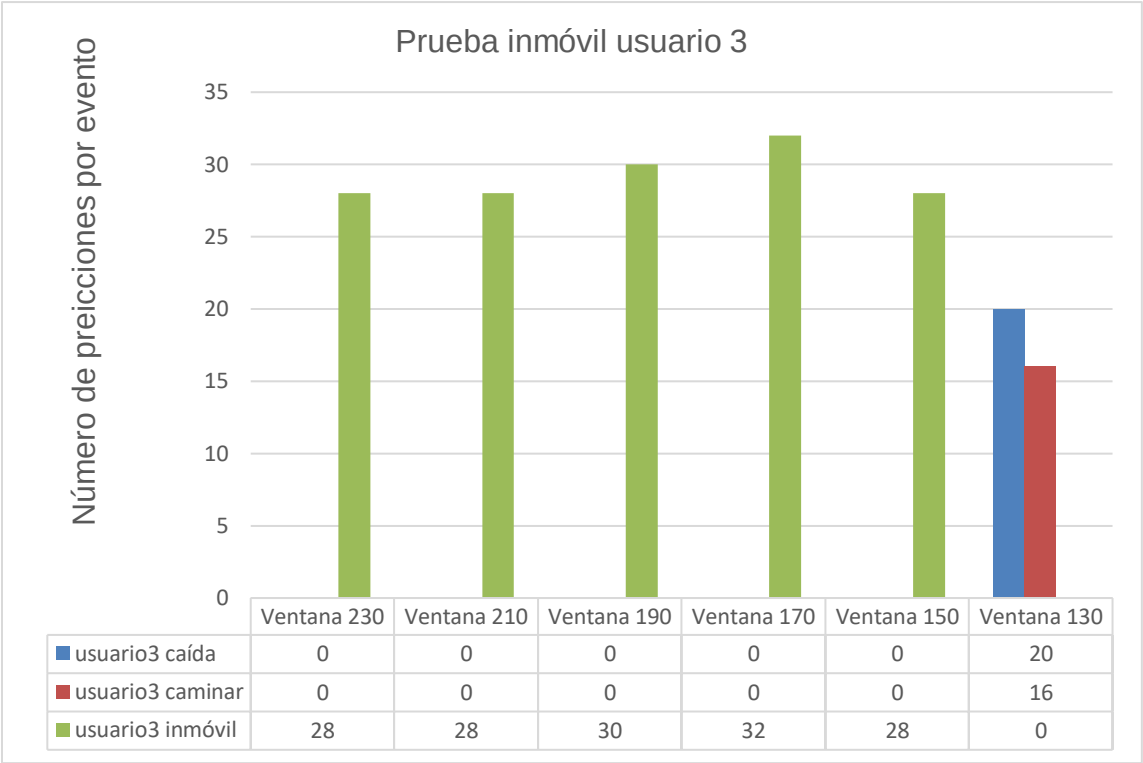


Figura 34. Prueba inmóvil Usuario 3

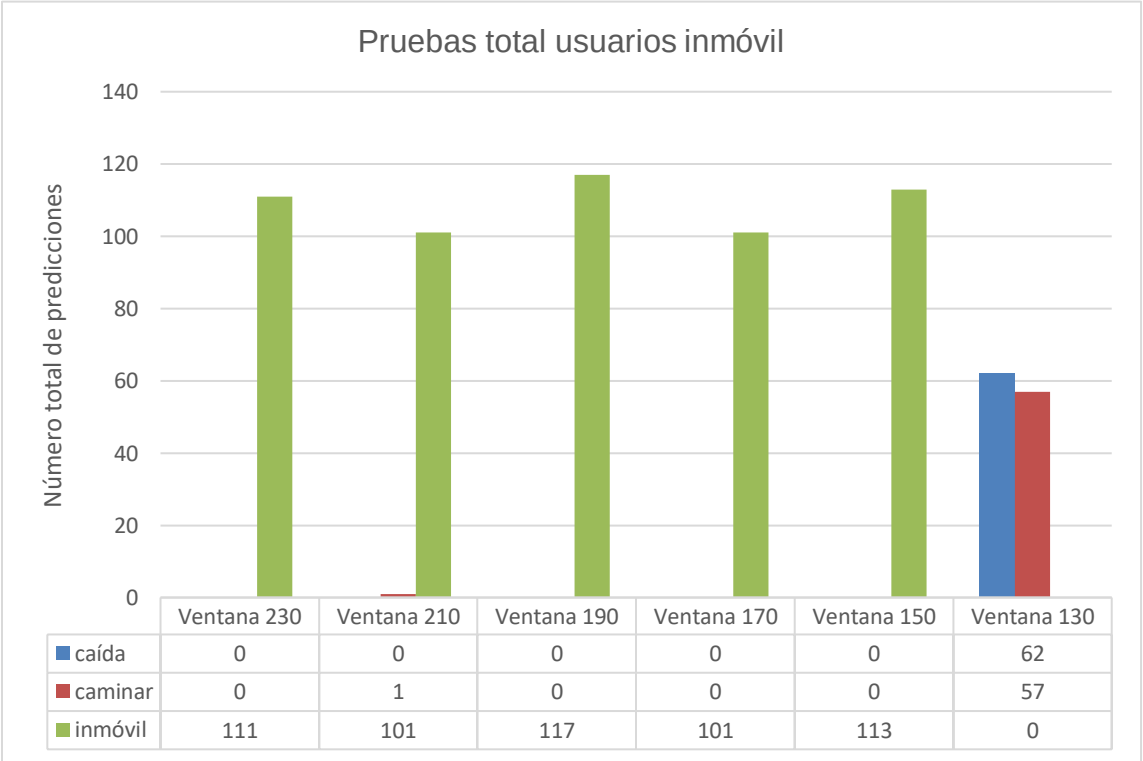


Figura 35. Pruebas total usuarios inmóviles.

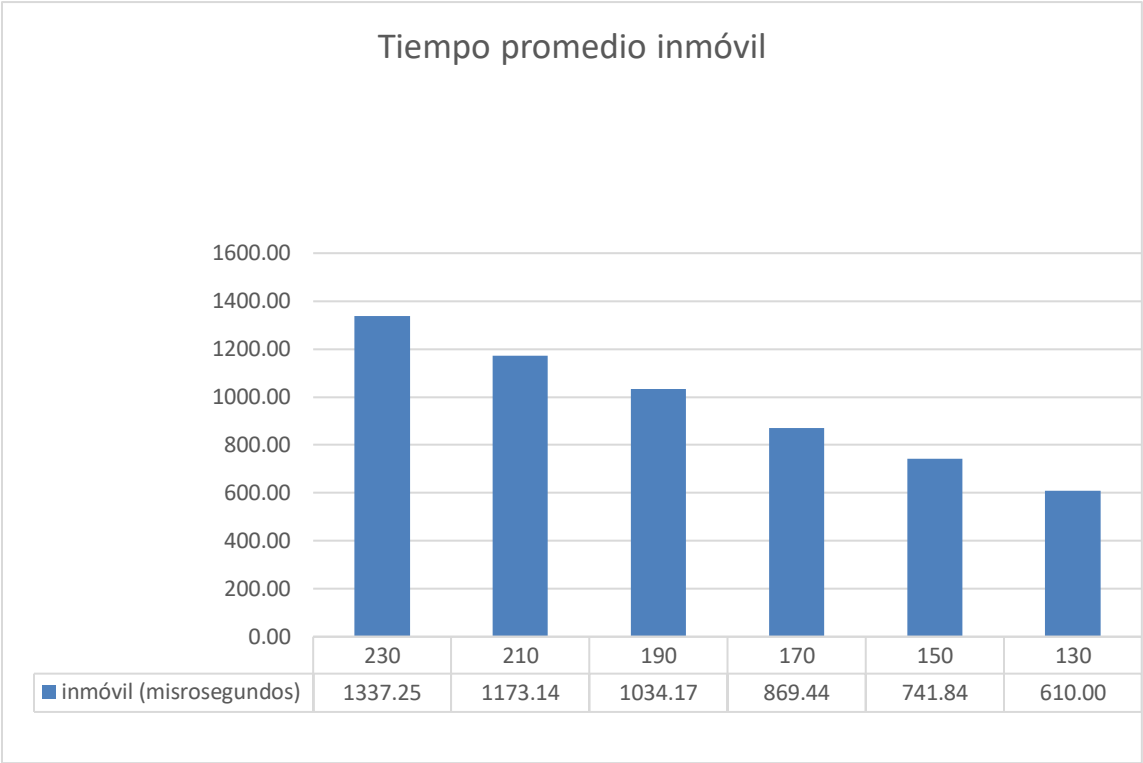


Figura 36. Tiempo promedio en las pruebas del evento inmóvil.

4.3 Evaluación de los resultados

En este punto se procede a utilizar las matrices de confusión, las cuales son el punto de partida para poder evaluar que tan fiable y que exactitud tiene el algoritmo clasificador. Estas tablas muestran la predicción obtenida vs el evento realizado en las pruebas.

En este caso se utilizó letras para poder representar los valores de la tabla, a continuación, se explica su representación:

- A: Predicciones correctas de caminar
- B: Predicciones incorrectas de caídas.
- C: Predicciones incorrectas de inmóvil
- D: Predicciones incorrectas de caminar
- E: Predicciones correctas de caídas.
- F: Predicciones incorrectas de inmóvil
- G: Predicciones incorrectas de caminar.
- H: Predicciones incorrectas de caídas

- E: Predicciones correctas de inmóvil.

Posterior a la creación de la matriz de confusión, se procede a calcular la exactitud del sistema por cada ventana, para esto se divide la suma de la diagonal dividido para la suma total de la matriz de confusión. En la Ecuación 12 se puede apreciar el ejemplo de lo previamente explicado.

Tabla 4

Matriz de confusión

	Caminar	**Caída**	**Inmóvil**
Caminar	A	B	C
Caída	D	E	F
Inmóvil	G	H	I

$$Exactitud = \frac{A + E + I}{A + B + C + D + E + F + G + H + I}$$

Una vez entendido el proceso de evaluación se aplica en los resultados de las pruebas realizadas previamente, a continuación, en la Figura 36 se puede observar la exactitud por ventana. En las tablas 4,5,6,7,8,9 se puede observar las matrices de confusión que permitieron llegar a dichas exactitudes

Tabla 5

Matriz de confusión ventana 130

	Caminar	**Caída**	**Inmóvil**
Caminar	227	4	57

Caída	27	14	62
Inmóvil	13	0	0

Tabla 6

Matriz de confusión ventana 150

	Caminar	**Caída**	**Inmóvil**
Caminar	168	1	0
Caída	27	17	0
Inmóvil	40	0	113

Tabla 7

Matriz de confusión ventana 170

	Caminar	**Caída**	**Inmóvil**
Caminar	154	3	0
Caída	29	14	0
Inmóvil	28	1	101

Tabla 8

Matriz de confusión ventana 190

	Caminar	**Caída**	**Inmóvil**
Caminar	118	2	0
Caída	43	16	0
Inmóvil	37	0	117

Tabla 9

Matriz de confusión ventana 210

	Caminar	**Caída**	**Inmóvil**
Caminar	97	2	1
Caída	37	14	0
Inmóvil	33	2	101

Tabla 10

Matriz de confusión ventana 230

	Caminar	**Caída**	**Inmóvil**
Caminar	106	1	0
Caída	25	16	0
Inmóvil	34	1	111

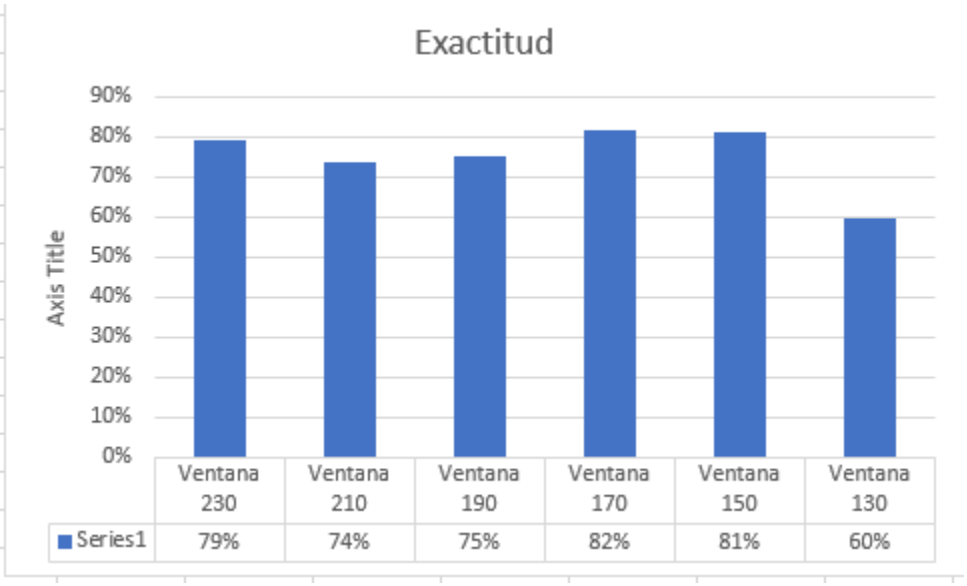


Figura 37. Exactitud

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones.

Durante la realización de las pruebas de funcionamiento del prototipo se pudo evidenciar que al realizar cualquier actividad que este fuera de las propuestas en la etapa de entrenamiento del algoritmo, al momento de la predicción el clasificador puede confundir esta actividad con alguna otra de las previamente entrenadas.

Después de realizar varias pruebas con diferentes usuarios y analizar las mismas se concluye que el tamaño de la ventana para la recolección de datos que permite obtener buenos resultados en la predicción es de 170 datos. Manejar ventanas de más o menos datos ocasiona que los porcentajes de exactitud al predecir un evento sean bajos y poco precisos

La utilización del microcontrolador Teensy 4.0 en el presente trabajo de titulación ha permitido que los tiempos de predicción sean de hasta 1.35 milisegundos gracias a su procesador ya que es capaz de procesar en un tiempo menor al tiempo de espera que se tiene entre la recolección de los datos del sensor, por lo que no se consideró necesario implementar un método de solapamiento de ventanas, para el proceso de predicción del algoritmo.

Tomando en cuenta que los algoritmos clasificadores no son precisos al 100%, la implementación de estos en el área de machinen learning es fundamental para poder alcanzar varios objeticos teniendo así resultados con éxito. Al utilizar Naive Bayes en el presente trabajo de titulación ha permitido obtener exactitud desde 60% hasta el 82% en tiempos de procesamiento hasta 1.35 milisegundos.

Se podría experimentar el uso de nuevos algoritmos de inteligencia artificial no supervisados para poder probar la eficiencia del mismo y la predicción en comparación al algoritmo actualmente utilizado.

Una parte fundamental del dispositivo es el dimensionamiento de la batería para que se pueda funcionar por un tiempo adecuado mientras se lo utiliza y de esta manera que pueda tener el desempeño correcto.

Actualmente el dispositivo está entrenado para que pueda detectar tres actividades, pero se podría aumentar el número de actividades de la vida diaria, lo cual serviría para la creación de bases de datos para poder ver las variaciones que se podrían obtener y poder aplicarlo exclusivamente para personas de la tercera edad.

5.2 Recomendaciones

Es recomendable que al desarrollar un sistema de machine learning, se lo pueda construir a un alto nivel de escalabilidad tanto en hardware como en software para que a futuro se pueda aumentar funciones para tener mejores resultados.

Es recomendable al momento de realizar las pruebas no alejar el prototipo más allá de 6 metros del ordenador donde se receptan los datos, ya que, pasada esta distancia, la conexión bluetooth puede cortarse ocasionando perdida de información del sensor, lo que afectaría en la obtención de las predicciones y la tabulación de los resultados obtenidos.

Se debe tomar mucho en cuenta que el microcontrolador Teensy 4.0 no se tiene como elemento en el software “Proteus” para el diseño de la placa, para lo cual, se tiene que hacer coincidir los pines para que este pueda tener las conexiones correctas con los elementos del dispositivo.

Para obtener una placa con una distribución optima los componentes que son de formas similares deben ser orientados de manera similar, ya que así el proceso de soldadura

será más eficiente. Es importante también identificar los componentes que tengan mayor o menor resistencia térmica y no soldarlos directamente, en su lugar utilizar espadines hembra o macho según se necesite para dicho componente.

Se debe tomar muy en cuenta que al momento de rutear las pistas en la placa PCB estas no tengan ángulos de 90° ya que esto ocasionaría perdidas de la señal entre un 5 y 9%, de igual manera es importante considerar que para las pistas de voltaje y tierra el ancho debe ser un poco mayor que el resto de pistas, esto de acuerdo al rango de corriente que circulará por las pistas.

Es recomendable realizar las pruebas en ambientes controlados o utilizar un maniquí para la obtención de datos en las pruebas del evento caída, realizar estas pruebas podría ocasionar lesiones en los sujetos de prueba, además que el realizar las caídas a voluntad puede ocasionar en algunos casos que los resultados no sean exactos.

REFERENCIAS

Abel, E. W., Zacharia, P. C., Forster, A., & Farrow, T. L. (1996). Neural network analysis of the EMG interference pattern. *Medical Engineering and Physics*, *18*(1), 12–17. https://doi.org/10.1016/1350-4533(95)00032-1

Aitex. (2017). Investigación y desarrollo de un sistema inteligente basado en sensores y actuadores integrados en textiles de aplicación a la prevención, detección y protección frente a caídas de personas mayores.

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (n.d.). *The "K" in K-fold Cross Validation*. Recuperado el 18 de Junio del 2020 de: http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Arduino. (2019). Arduino Recuperado el 20 de Marzo del 2020 de: https://www.arduino.cc/reference/en/language/functions/communication/serial/write

Asistencias, T. (n.d.). Detector de caídas. Recuperado el 3 de Mayo del 2020 de https://topasistencia.es/detector-de-caidas/

Barreras, V. S. (n.d.). *El detector de caídas "Vigi'fall."* Recuperado el 3 de Mayo del 2020 de https://es.validasinbarreras.com/blog/post/el-detector-de-caidas-vigi-fall/

Casas Roma, J., Bosch Rué, A., & Lozano Bagén, T. (2019). *Deep learning : principios y fundamentos*. UOC.

Chen, S., Webb, G. I., Liu, L., & Ma, X. (2020). A novel selective naïve Bayes algorithm. *Knowledge-Based Systems*, *192*, 105361. Recuperado el 3 de Mayo del 2020 de https://doi.org/10.1016/j.knosys.2019.105361

Configuración del módulo bluetooth HC-05 usando comandos AT. (n.d.). Recuperado el 4 de Junio del 2020, de: https://naylampmechatronics.com/blog/24_configuracion-

del-modulo-bluetooth-hc-05-usa.html

Cross-validation: evaluating estimator performance — scikit-learn 0.23.1 documentation. (n.d.). Recuperado el 17 de Junio del 2020 de https://scikit-learn.org/stable/modules/cross_validation.html

DLR - Institute of Communications and Navigation - Data Set. (n.d.). Recuperado el 17 de Junio 17 del 2020 de: https://www.dlr.de/kn/en/desktopdefault.aspx/tabid-12705/22182_read-50785/

González, R., Raúl, V., Bretones, H., Jiménez, B., Dirigido, D. O., Jesús, J., & Sanz, G. (2015). Desarrollo de un sistema de detección de caídas basado en acelerómetros.

María, J., Plaza, C., Agüero, C. E., & González, T. (n.d.). CHAPTER X Detección visual de caídas para ambientes inteligentes. Recuperado el 3 de Mayo del 2020 de www.teleasistencia.com

Nazmi, N., Rahman, M. A. A., Mazlan, S. A., Zamzuri, H., & Mizukawa, M. (2015, September 1). Electromyography (EMG) based signal analysis for physiological device application in lower limb rehabilitation. *Proceedings - 2015 2nd International Conference on Biomedical Engineering, ICoBE 2015*. https://doi.org/10.1109/ICoBE.2015.7235878

New Teensy 4.0 Blows Away Benchmarks, Implements Self-Recovery, Returns To Smaller Form | Hackaday. (2019). https://hackaday.com/2019/08/07/new-teensy-4-0-blows-away-benchmarks-implements-self-recovery-returns-to-smaller-form/

Persona-Ordenador, I. (2018). Interacción 2018 XIX Congreso Internacional de.

Perspective, A. A. (2015). *Chapman & Hall/CRC Machine Learning & Pattern Recognition Series Chapman & Hall/CRC Machine Learning & Pattern Recognition Series Machine Learning M AC H I N E LEARNING*.

Rish, I. (2020). *An empirical study of the naive Bayes classifier*.

Russo, C., Ramón, H., Alonso, N., Cicerchia, B., Esnaola, L., Ardo, & Tessore, J. P. (2016). Tratamiento Masivo de Datos Utilizando Técnicas de Machine Learning: Vol. WICC 2016.

Salud, O. M. de la. (2018). Recuperado el 17 de Mayo del 2020 *Caídas de:* https://www.who.int/es/news-room/fact-sheets/detail/falls

SENSE4CARE. (n.d.). *The best fall detector in the market. Reliable, easy to use and economical*. Recuperado el 3 de Mayo del 2020 de https://www.sense4care.com/fall-detection/

Settles, B. (2009). *Computer Sciences Department Active Learning Literature Survey*.

Solórzano, S., Pozo, D., Morales, L., & Villalonga, C. (2018). Análisis comparativo de algoritmos de aprendizaje supervisado para la detección de caídas.

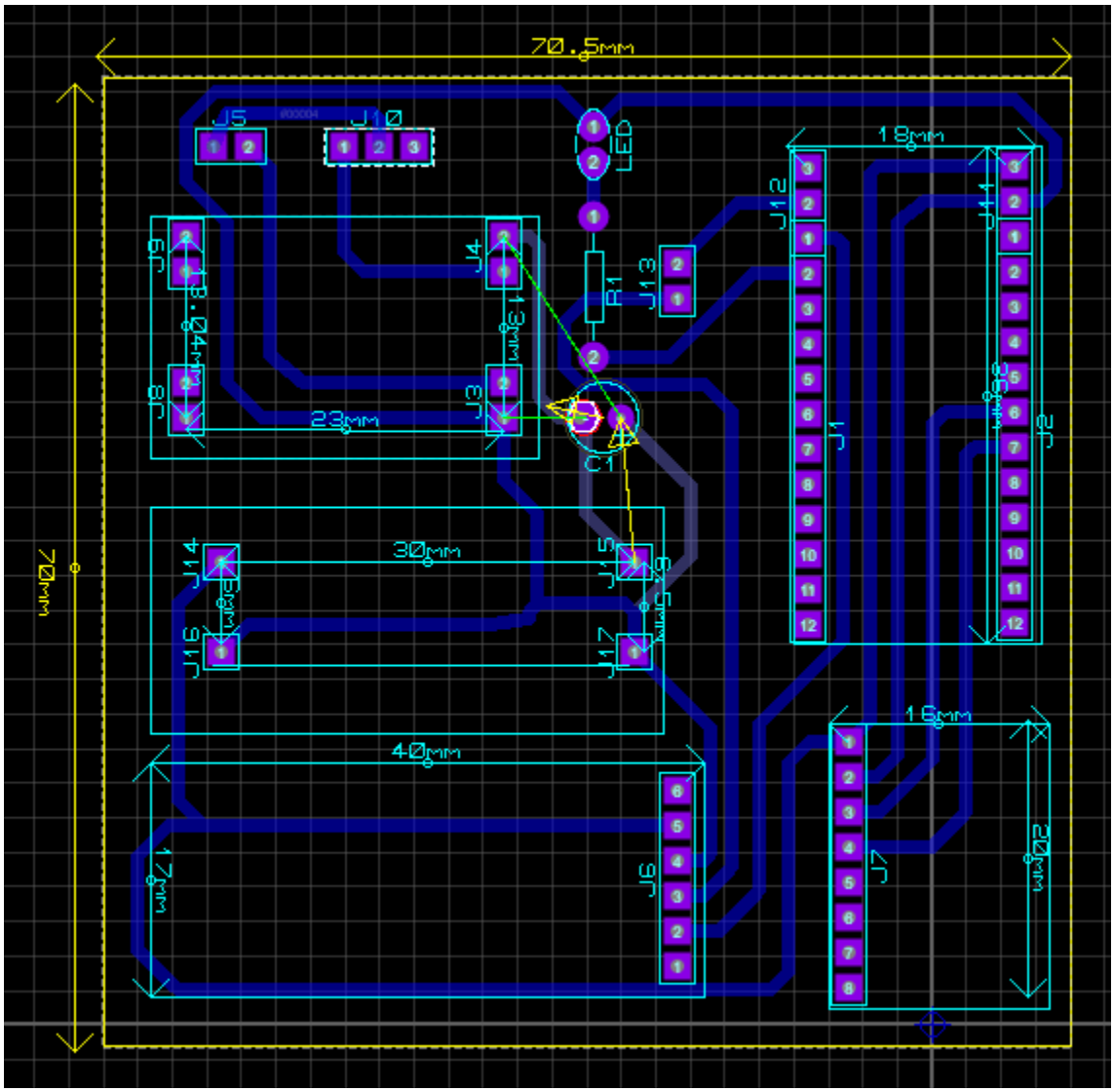
Store, P. (2020). Recuperado el 3 de Mayo del 2020 de *PJRC Store*. https://www.pjrc.com/store/teensy40.html

Ting, S. L., Ip, W. H., & Tsang, A. H. C. (2011). Is Naïve Bayes a Good Classifier for Document Classification? In *International Journal of Software Engineering and Its Applications* (Vol. 5, Issue 3).

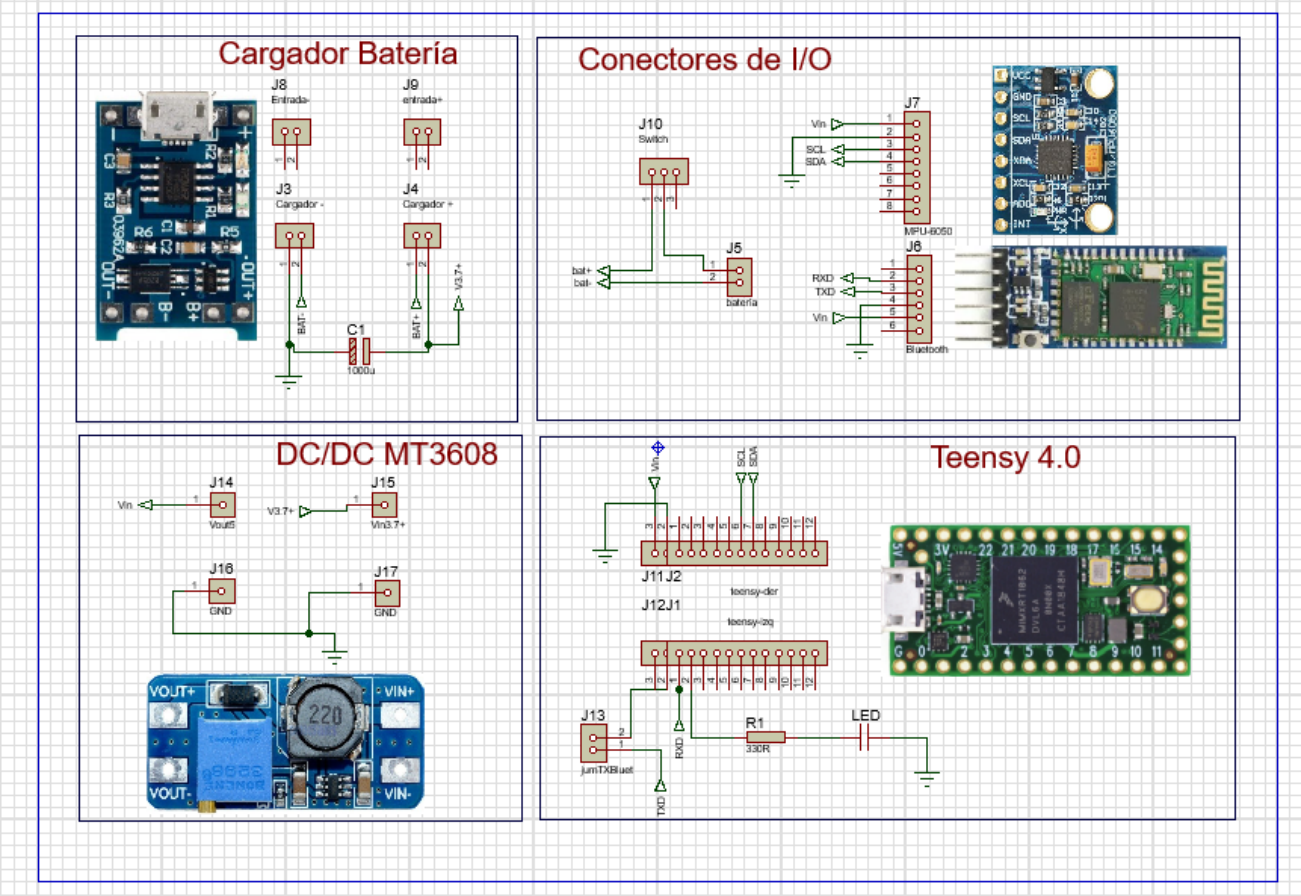
Vigi'Fall. (n.d.). *Detector de caída a distancia - Vigi'Fall ™ - Vigilio Telemedical*. Recuperado el 3 de Mayo del 2020 de https://www.medicalexpo.es/prod/vigilio-telemedical/product-75024-462746.html

ANEXOS

ANEXO 1 : Placa del sistema microcontrolado



ANEXO 2: Circuito del sistema microcontrolado



ANEXO 3. Tablas de las pruebas realizadas a los usuarios con el evento caminar.

		Ventana 230	**Ventana 210**	**Ventana 190**	**Ventana 170**	**Ventana 150**	**Ventana 130**
usuario 1	**caída**	3	4	9	3	1	2
	caminar	22	15	21	29	28	39
	inmóvil	10	7	7	6	9	2
usuario2	**caída**	2	7	4	1	0	3
	caminar	14	14	27	29	29	44
	inmóvil	10	9	5	4	8	0
usuario3	**caída**	2	2	5	5	2	0
	caminar	18	20	19	21	26	37
	inmóvil	8	7	6	9	9	5
usuario4	**caída**	3	3	6	2	2	2
	caminar	20	19	18	29	36	42
	inmóvil	2	3	4	2	2	2
usuario5	**caída**	6	8	7	10	11	8
	caminar	18	18	19	22	23	32
	inmóvil	3	3	9	4	8	2
usuario6	**caída**	9	13	12	8	11	12
	caminar	14	11	14	24	26	33
	inmóvil	1	4	6	3	4	2

ANEXO 4. Tablas de las pruebas realizadas a los usuarios con el evento caída.

		Ventana 230	Ventana 210	Ventana 190	Ventana 170	Ventana 150	Ventana 130
usuario 1	**caída**	3	1	3	3	3	2
	caminar	0	0	0	0	0	1
	inmóvil	0	2	0	0	0	0
usuario2	**caída**	3	2	2	3	3	2
	caminar	0	1	1	0	0	1
	inmóvil	0	0	0	0	0	0
usuario3	**caída**	2	3	2	2	3	3
	caminar	1	0	1	1	0	0
	inmóvil	0	0	0	0	0	0
usuario4	**caída**	2	2	3	2	2	3
	caminar	0	1	0	1	1	0
	inmóvil	1	0	0	0	0	0
usuario5	**caída**	3	3	3	1	3	2
	caminar	0	0	0	1	0	1
	inmóvil	0	0	0	1	0	0
usuario6	**caída**	3	3	3	3	3	2
	caminar	0	0	0	0	0	1
	inmóvil	0	0	0	0	0	0

ANEXO 5. Tablas de las pruebas realizadas a los usuarios con el evento inmóvil

.

		Ventana 230	**Ventana 210**	**Ventana 190**	**Ventana 170**	**Ventana 150**	**Ventana 130**
usuario 1	**caída**	0	0	0	0	0	1
	caminar	0	0	0	0	0	0
	inmóvil	42	30	42	28	42	42
usuario 2	**caída**	0	0	0	0	0	0
	caminar	0	0	0	0	0	0
	inmóvil	41	44	45	41	43	41
usuario 3	**caída**	0	0	0	0	0	0
	caminar	0	0	0	0	0	1
	inmóvil	28	28	30	32	28	36