PROTOTIPO PARA LA CLASIFICACIÓN DE MOVIMIENTOS CORPORALES Y DETECCIÓN DE CAÍDAS USANDO UNA RNA

PROTOTYPE FOR BODY MOVEMENT CLASSIFICATION AND FALL DETECTION USING AN ANN

José Luis Vázquez Ávila
Universidad Autónoma del Carmen, México
jvazquez@pampano.unacar.mx

Rafael Sánchez Lara
Universidad Autónoma del Carmen, México
rslara@delfin.unacar.mx

Marco Antonio Rodríguez Blanco
Universidad Autónoma del Carmen, México
mrodriguez@pampano.unacar.mx

Tila Trinidad González Grajeda
Universidad Autónoma del Carmen, México
tgonzalez@pampano.unacar.mx

Younes El Hamzaoui
Universidad Autónoma del Carmen, México
eyouness@pampano.unacar.mx

Francisco Méndez Martínez
Universidad Autónoma del Carmen, México
fmendez@pampano.unacar.mx

Recepción: 25/noviembre/2025 **Aceptación:** 24/diciembre/2025

Resumen

Si bien las Redes Neuronales Artificiales (RNAs) suelen desplegarse en equipos con vastos recursos de hardware, las tecnologías recientes han posibilitado su implementación incluso en aplicaciones con capacidades de hardware limitadas. Por lo anterior, este trabajo presenta la propuesta de un prototipo basado en Arduino Nano para clasificar patrones de actividad física elementales (como estar sentado o caminar) y detectar caídas súbitas. Los datos se obtienen mediante el uso de un sensor acelerómetro-giroscopio MPU-6050 colocado en alguna parte del cuerpo.

Para que la red haga una correcta clasificación de los datos se utiliza la técnica de extracción de características. Para el entrenamiento de la red se utilizó el lenguaje Python, además de la librería Tensorflow. La red está construida con una sola capa oculta de 4 neuronas, por lo cual, los recursos de hardware utilizados en el arduino son mínimos. La red RNA resultante logra una exactitud del 96%.

Palabras Clave: Arduino, inteligencia artificial, machine learning, Python, redes neuronales artificiales.

Abstract

While Artificial Neural Networks (ANNs) are typically deployed on computers with extensive hardware resources, recent technologies have enabled their implementation even in applications with limited hardware capabilities. Therefore, this work presents a prototype based on the Arduino Nano to classify basic physical activity patterns (such as sitting or walking) and detect sudden falls. Data is obtained using an MPU-6050 accelerometer-gyroscope sensor placed on a specific part of the body. Feature extraction is used to correctly classify the data. The network was trained using Python and the Tensorflow library. The network is built with a single hidden layer of four neurons, thus, the hardware resources required on the Arduino are minimal. The resulting ANN network achieves an accuracy of 96%.

Keywords: *Arduino, artificial intelligence, artificial neural networks, machine learning, Python.*

1. Introducción

En los últimos años, el campo de la robótica ha visto avances significativos gracias a la aplicación de las Redes Neuronales Artificiales (RNA). Estas redes se basan en el funcionamiento del cerebro humano, imitando su capacidad para pensar, recordar y resolver problemas. Estos avances se pueden observar en dispositivos como relojes inteligentes, dispositivos móviles como los teléfonos inteligentes, entre otros, los cuales se usan para medir la actividad física de los usuarios. Por otra parte, las funcionalidades de estos dispositivos se extienden a los sectores de la salud e industria. La principal fortaleza de las RNAs reside en su

capacidad de aprender y modelar patrones a partir de datos de entrenamiento. Esta habilidad de ajuste predictivo es fundamental para el monitoreo de la actividad física en aplicaciones de salud. Mediante los procesos de entrenamiento, validación y predicción [Kim, 2017], [Paluszek, 2022], las RNA pueden desarrollar modelos para clasificar con precisión los movimientos de una persona (como caminar o estar sentado) y detectar eventos críticos, como caídas súbitas, permitiendo la alerta temprana y una intervención asistencial inmediata.

Aunque existe mucha investigación y muchas implementaciones de RNAs en la literatura, pocas se centran en su implementación en dispositivos portátiles. Algunas investigaciones se centran en el reconocimiento de gestos y el reconocimiento de patrones. Hay trabajos del área de la salud relacionados con el reconocimiento de gestos para determinar el alfabeto en lenguaje de señas [Naosekpam, 2019], o el control de prótesis ortopédicas [Arvetti, 2007], [Serrezuela, 2019]. También, existen trabajos que hacen uso de la interacción fisiológica del cuerpo para controlar videojuegos [Nacke, 2011]. Otros trabajos se basan en la visión artificial, empleando cámaras para capturar y procesar imágenes. Por ejemplo, [Nope, 2008] compara técnicas como las redes neuronales artificiales y los clasificadores bayesianos para el reconocimiento de gestos visuales, logrando una precisión de entre 87% y 97%. [Ortega, 2017] propone un sistema similar con redes neuronales convolucionales en MATLAB, y [Vargas, 2010] desarrolla un sistema para reconocer el lenguaje de señas con una precisión cercana al 99%. Aunque estas soluciones son efectivas, dependen del uso de cámaras y computadoras, lo que implica un alto costo computacional para el procesamiento de imágenes.

Una alternativa a la visión artificial es el uso de señales extraídas de los músculos. Estudios como los de [Lee, 2021], [Hristov, 2022] y [Parajuli, 2019] investigan la detección de gestos mediante señales nerviosas, mientras que [Fatayerji, 2022] y [Ozdemir, 2022] utilizan RNA y redes neuronales convolucionales (RNC) para clasificar gestos a partir de señales musculares. En [Artemiadis, 2010] se usan sensores electromiográficos para el control de brazos robóticos para personas con alguna discapacidad. Sin embargo, estos enfoques a menudo implican costosos sistemas de adquisición de datos y algoritmos computacionalmente intensivos.

Existen varias publicaciones relacionadas con el estudio de la actividad física de las personas, de las cuales destacamos a [Cavita, 2024] y [López, 2024]. En [Cavita, 2024] se propone un modelo de redes neuronales que clasifica 12 actividades físicas a partir de señales de acelerometría. Este enfoque se basa en la extracción de características temporales y utiliza tres acelerómetros por usuario. Aunque el sistema alcanza una exactitud máxima del 90%, los dispositivos empleados no son de recursos limitados. Por otro lado, en [López, 2024] el estudio se centró en el monitoreo de la actividad diaria para promover un estilo de vida saludable, con el objetivo de identificar seis movimientos: caminar, subir y bajar escaleras, sentarse, ponerse de pie y acostarse. El conjunto de datos provino de repositorios. Se evaluaron tres algoritmos de clasificación: K-NN, Regresión Logística y Redes Neuronales Convolucionales (RNC). Las RNC resultaron ser más efectivas con una exactitud del 99%, sin embargo, los algoritmos son complejos y no aptos para dispositivos de recursos limitados.

En contraste con las metodologías mencionadas, este trabajo propone un prototipo para la clasificación de patrones de actividad física. El sistema se basa en la adquisición de datos a través de un Arduino Nano y un acelerómetro. Los datos pasan por un proceso de extracción de características temporales. Los datos obtenidos se usan para entrenar una RNA, que luego se implementa en otro Arduino Nano. El objetivo es ofrecer una solución de bajo costo, que, aunque solo considera dos actividades físicas (sentado y caminando), se agrega una de detección de caídas. Esta última es muy importante en aplicaciones de hospitales, vigilancia de pacientes enfermos o de la tercera edad o en aplicaciones dentro de la industria.

2. Métodos

Para resolver el problema, se utilizó una red neuronal artificial. El proceso se dividió en 4 fases (Figura 1):

- **Adquisición de Datos**: Se generó una base de datos a partir de las señales captadas por un sensor acelerómetro-giroscopio MPU-6050 que se colocó en el brazo.
- **Extracción de Características**: Se extraen 3 características temporales.

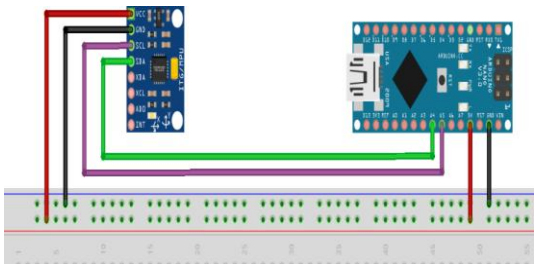
- **Entrenamiento de la red**: Se entrenó la red neuronal artificial utilizando TensorFlow en Python. Se creó un modelo de red multicapa de tipo secuencial para procesar los datos.
- **Implementación en Arduino Nano**: La RNA resultante se implementa en un Arduino Nano.

Fuente: elaboración propia

Figura 1 Diagrama de Bloques del Sistema.

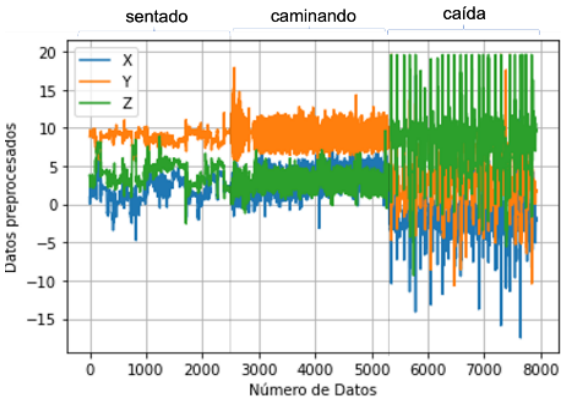
Adquisición de datos

La adquisición de los datos se realizó a través de un acelerómetro-giroscopio MPU-6050 conectado a un arduino Nano, en una configuración como la mostrada en la Figura 2. Los parámetros para la adquisición de los datos son los siguientes: Periodo de muestreo de $10\ ms$; Se leyeron alrededor de $2{,}500$ datos para las clases sentado, caminando y caída, para los ejes x, y y z del acelerómetro, Figura 3. Los datos extraídos representan series temporales que a simple vista parecen tener el mismo comportamiento. Una práctica muy conveniente en el preprocesamiento de los datos es la extracción de características, las cuales ofrecen más información sobre la señal. En este trabajo se utiliza la extracción de algunas características temporales a las señales x, y y z.



Fuente: elaboración propia.

Figura 2 Adquisición de datos.



Fuente: elaboración propia.

Figura 3 Datos Adquiridos.

Extracción de características temporales

La extracción de características tiene por objeto obtener la representación y la información que contienen las señales, lo que es necesario para minimizar la complejidad de la aplicación y reducir el coste de procesamiento de la información. Existen varias extracciones de características, sin embargo, utilizaremos solo tres:

- **Valor absoluto medio (MAV).** Es el promedio del valor absoluto de los datos adquiridos, se calcula a través de la Ecuación 1, dónde X_K son los datos, $K = \{1,2, \dots, N\}$ y N es el número de muestras.

$$MAV = \frac{1}{N}\sum_{K=1}^{N} |X_K| \tag{1}$$

- **Longitud de onda (WL)**. Es la variación acumulativa que puede indicar el grado de variación de la señal y se calcula a través de la Ecuación 2.

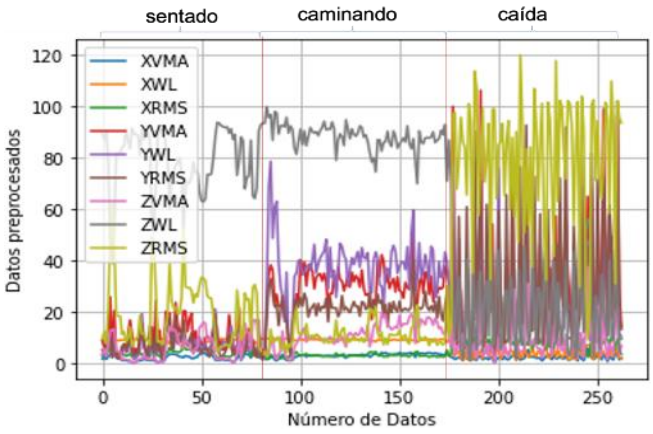
$$WL = \sum_{K=1}^{N} |X_{K+1} - X_K| \tag{2}$$

- **Valor Cuadrático Medio (RMS)**. Este valor indica la magnitud efectiva de una señal, independientemente de si sus valores son positivos o negativos, la cual viene dada por la Ecuación 3.

$$RMS = \sqrt{\sum_{K=1}^{N} \frac{X_K^2}{N}} \tag{3}$$

Para este trabajo se consideró $N = 30$, como el tamaño de ventana de segmentación, además se utiliza segmentación disjunta [Spiewak, 2018]. Por lo anterior, al aplicar la extracción de características se tendrán 3 características para cada eje del acelerómetro lo que resulta en un total de 9 entradas a la RNA. La Figura 4 muestra el comportamiento de los datos al aplicar la extracción de características. La extracción de características ofrece una representación diferente del comportamiento de las señales, mostrando otro tipo de información que no se ve en los datos brutos como los de la Figura 3. También, se observa de la Figura 4 que el número de datos se ha reducido en un orden de 30, con respecto a los datos de la Figura 3. Para que número de datos sea mayor se pueden utilizar técnicas de

segmentación con traslapes, donde se traslapa un porcentaje de los datos [Cavita, 2024], [Spiewak, 2018].



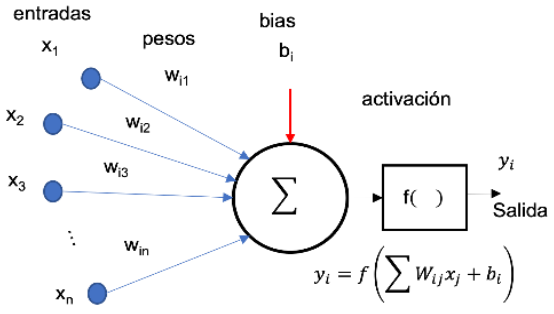
Fuente: elaboración propia

Figura 4 Datos con extracción de características.

Red Neuronal Artificial (RNA)

Las RNAs se representan mediante el modelo de perceptrón multicapa, Figura 5. El elemento fundamental de una red neuronal es la neurona. La red neuronal artificial se compone por los siguientes elementos [Nacelle, 2009], [Paluszek, 2022]:

- Entradas (x): Es la información que recibe la red.
- Pesos (w): son los valores numéricos que se encargan de establecer la influencia de una entrada en la salida deseada, los cuales se ajustarán en cada iteración.
- Bias (b): Es parecido al peso solo que este está asociado a una entrada constante. Lo que hará al modelo más flexible.



Fuente: elaboración propia

Figura 5 Estructura de una neurona artificial simple, también llamada Perceptrón.

- Función de activación o transferencia $f()$: Las funciones de activación se utilizan para introducir la no linealidad en las RNA.

Procesamiento de la base de datos

Una vez que se adquieren los datos y se extraen las características el siguiente paso es el preprocesamiento de los datos. Previo a la etapa de entrenamiento se realiza un preprocesamiento de los datos, cuyo objetivo principal es facilitar el aprendizaje de la red. El preprocesamiento de datos puede constar de diferentes métodos, tales como [Paluszek, 2022]:

- Normalización.
- Transformaciones no lineales.
- Extracción de características.
- Codificación de entradas y salidas.
- Manejo de datos faltantes.

Para este trabajo se utilizó la normalización, ya que se recomienda normalizar los datos que usan escalas e intervalos diferentes. La normalización normal estándar requiere de la media y la desviación estándar del conjunto de datos por clase. El resultado de la normalización produce otro conjunto de datos con media 0 y desviación estándar 1 [Paluszek, 2022]. La Tabla 1 muestra la codificación usada .

Tabla 1 Codificación One-Hot.

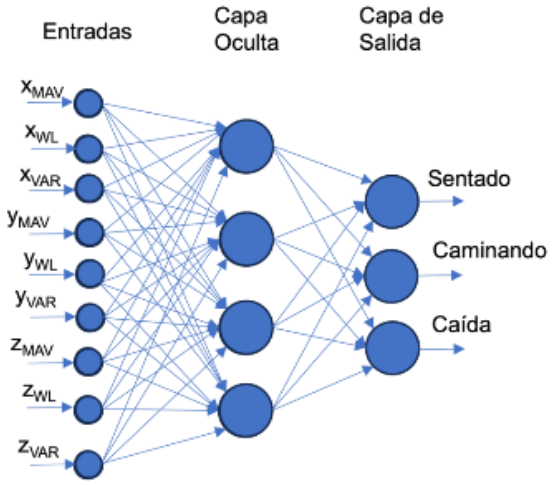
Clase	sentado	caminando	caída
sentado	1	0	0
caminando	0	1	0
caída	0	0	1

Fuente: elaboración propia

Creación del modelo de Red Neuronal Artificial

Para definir la RNA se implementó en Python utilizando TensorFlow y su librería Keras como una secuencia de capas [Campesato, 2019]. La red que se define en este trabajo consta de nueve entradas, definidas por las características de cada uno de los ejes del acelerómetro y de una sola capa oculta con cuatro neuronas, las

cuales se implementaron con la función de activación “ReLu”. La capa de salida está compuesta de tres neuronas que definen las 3 clases de actividad física a reconocer, dicha capa tiene la función de activación “Softmax”, ésta es la utilizada para problemas multiclase. La Figura 6 muestra la arquitectura de la RNA a implementar.



Fuente: elaboración propia

Figura 6 Red neuronal implementada.

3. Resultados

Una vez entrenada la RNA en TensorFlow se obtienen los pesos, los bias y las estadísticas necesarias para la implementación en el Arduino Nano. Para una mejor comprensión de la necesidad del uso de la extracción de características, se implementó la red tanto con los datos brutos del sin extracción de características y con extracción de características. Adicionalmente, para fines de comparación y con los mismos parámetros se utilizó la regresión logística multiclase [Dong, 2019].

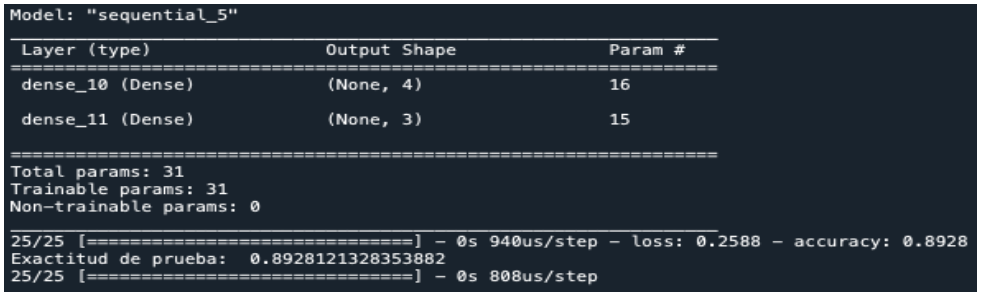
Parámetros de entrenamiento

Se seleccionó “Adam” como el algoritmo de optimización para el entrenamiento de la red. La “entropía cruzada categórica” se eligió como la función de pérdida ya que es la opción estándar para problemas de clasificación con múltiples clases [Campesato, 2019]. Se utilizó la función de activación *SoftMax* para la capa de salida. Durante el proceso de entrenamiento del modelo, solo se ajustó el número

de épocas. Esto se debe a que el optimizador Adam combina las ventajas de otros optimizadores como AdaGrad y RMSProp al usar tasas de aprendizaje adaptativas para cada parámetro, mejorando la eficiencia y la velocidad de convergencia [Páez, 2020]. Las configuraciones consistieron en: 500 épocas, los pesos de la red se actualizan después de cada época. Una capa oculta, compuesta por 4 neuronas.
La evaluación del modelo se realiza utilizando dos métricas principales: el valor de pérdida y la exactitud de la red.

Resultados de RNA sin extracción de características

Para este escenario se utilizó una RNA con los parámetros antes mencionados. La Figura 7 muestra los resultados de la ejecución de la RNA con una exactitud de aproximadamente el 90%. El total de parámetros está formado por los pesos y bias de la red, donde se tienen 3 entradas y 4 nodos en la capa oculta, por lo que hay 12 pesos más los 4 bias de cada uno de los nodos de la capa oculta. Para la capa de salida se tienen los 4 nodos de la capa oculta y 3 nodos de la capa de salida, por lo que se obtienen 12 pesos, más los 3 bias de la capa de salida se obtienen 15 parámetros; total 31 parámetros. Al ejecutar la regresión logística se obtuvo una exactitud del 87%, por lo que la RNA presenta mejores condiciones en la clasificación.



Fuente: elaboración propia

Figura 7 Resultado de la ejecución de la RNA sin características.

Resultados de RNA con extracción de características

Como se está considerando la extracción de características el número de entradas ahora es igual a 9. Al igual que en la sección anterior, podemos determinar

el número de parámetros de la RNA, con un total de 40 parámetros en la capa oculta y 15 en la capa de salida, por lo que se tiene un total de 55 parámetros. Nótese que el número de parámetros se incrementa debido a que el número de entradas es mayor. Sin embargo, la exactitud se incrementa a aproximadamente 96%, tal como se muestra en la Figura 8. Lo anterior representa una ganancia significativa en cuestiones de la exactitud de la RNA, por lo que la implementación de extracción de características en series temporales es de gran utilidad. Con respecto a la implementación en el algoritmo de regresión logística, se obtuvo una exactitud del 92%. Para aplicaciones donde se utilizarán procesadores con capacidades limitadas es importante tener en consideración el número total de parámetros debido a que esto se convertirá en una relación de la cantidad de memoria requerida para la implementación. La Figura 9 muestra el resultado de la matriz de confusión. La matriz de confusión muestra la exactitud con la que la RNA hace las predicciones y se genera a través de los datos de validación.

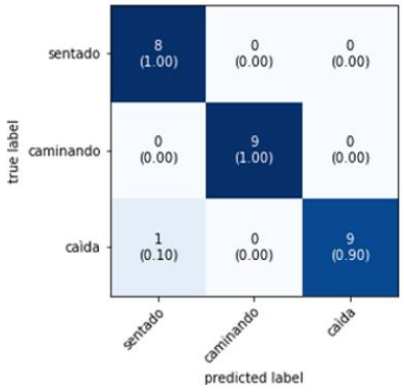
```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 4)                 40

 dense_15 (Dense)            (None, 3)                 15

=================================================================
Total params: 55
Trainable params: 55
Non-trainable params: 0
_________________________________________________________________
1/1 [==============================] - 0s 16ms/step - loss: 0.1877 - accuracy: 0.9630
Exactitud de prueba:  0.9629629850387573
1/1 [==============================] - 0s 39ms/step
```

Fuente: elaboración propia

Figura 8 Resultado de la ejecución de la RNA con características.



Fuente: elaboración propia

Figura 9 Matriz de Confusión de la RNA.

En este trabajo se consideró un porcentaje de validación del 10%. De la figura se puede observar que la red confunde en una ocasión “caída” con “sentado”, pero hace buenas predicciones en los escenarios de “sentado” y “caminando”.

Implementación en Arduino Nano

Para implementar la red neuronal artificial (RNA) en la tarjeta Arduino, primero se necesita acceder a los pesos y los bias que se obtuvieron durante el entrenamiento. Los resultados específicos de estos parámetros, que corresponden a las conexiones entre las capas de entrada y oculta (pesos y bias) y las conexiones entre la capa oculta y de salida (pesos y bias), se detallan en las Tablas 2, 3 y 4, respectivamente. La red neuronal artificial (RNA), una vez entrenada, se integra en Arduino usando los pesos y *bias*. La Figura 10 muestra el segmento de código, en lenguaje C.
Los datos de las Tablas 2, 3 y 4 son usados para que la RNA realice las predicciones con nuevos datos adquiridos desde el acelerómetro. El proceso sigue estos pasos:

- Preparación de datos: Los datos del acelerómetro se adquieren y se les extraen características. Luego se normalizan con la media y la desviación estándar del conjunto de entrenamiento para formar el vector de entrada ($a0$).

Tabla 2 Pesos de las conexiones de la capa oculta y las entradas.

$W1ij$	0	1	2	3	4	5	6	7	8
0	0.303	−1.284	−1.263	−0.137	0.161	0.62	−0.502	−1.453	−0.105
1	0.885	0.776	−1.37	0.054	0.728	0.011	−0.501	1.225	−0.076
2	0.37	−1.033	0.342	0.729	0.19	0.898	−0.071	−0.085	−0.068
3	0.3	1.122	−0.4	0.145	2.069	0.33	0.071	0.049	−0.259

Fuente: Elaboración propia.

Tabla 3 Pesos de las conexiones de la capa oculta y la capa de salida.

$W2ij$	0	1	2	3
0	−1.74	−0.746	−0.238	−1.154
1	−0.426	0.32	−1.258	1.205
2	1.945	−1.839	1.04	0.217

Fuente: Elaboración propia.

Tabla 4 Bias correspondientes a las neuronas de la capa oculta y capa de salida.

bi	0	1	2	3
$b1i$	0.841	−0.041	0.32	0.559
$b2i$	1.81	−1.625	−1.292	N/A

Fuente: Elaboración propia

```
//////////Entradas de la RNA///////////////////
  a0[0] = dataNormalized(mavX,mean[0],dstd[0]);
  a0[1] = dataNormalized(mavY,mean[1],dstd[1]);
  a0[2] = dataNormalized(mavZ,mean[2],dstd[2]);

  a0[3] = dataNormalized(wlX,mean[3],dstd[3]);
  a0[4] = dataNormalized(wlY,mean[4],dstd[4]);
  a0[5] = dataNormalized(wlZ,mean[5],dstd[5]);

  a0[6] = dataNormalized(varX,mean[6],dstd[6]);
  a0[7] = dataNormalized(varY,mean[7],dstd[7]);
  a0[8] = dataNormalized(varZ,mean[8],dstd[8]);

//////////////////////////// Estructura Red Neuronal ////////////////////////////////
for(int i = 0 ; i<8; i++ ) {aux=0.0;for(int j = 0 ; j <9 ; j++ ) { aux=aux+W1[i][j]*a0[j];} a1[i]=relu(aux+b1[i]);}
double aux1 = 0;
for(int i = 0 ; i<4; i++ ) {aux=0.0;for(int j = 0 ; j <8 ; j++ ){ aux=aux+W2[i][j]*a1[j];} a2[i]=(aux+b2[i]);aux1=aux1+exp(a2[i]);}
double minimo = 0.0;
int classes = 0;
for(int i = 0;  i<4; i++){a2[i] = exp(a2[i])/aux1;if(a2[i]>minimo){minimo=a2[i];classes=i;}}
//////////////////////////////////////////////////////////////////////////////////////
```

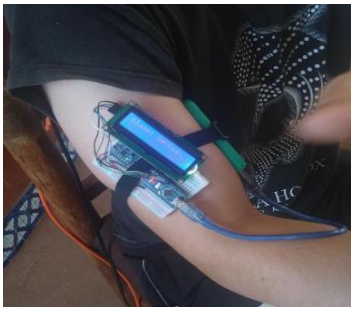
Fuente: elaboración propia

Figura 10 Código de la RNA en Arduino IDE.

- Cálculos de la primera capa: Un ciclo for implementa la conexión de las 9 entradas a las 4 neuronas de la capa oculta. Se usan los pesos de la Tabla 2 ($W1_{ij}$) y la función de activación ReLU.
- Cálculos de la segunda capa: Un segundo ciclo for se encarga de la conexión de las 4 neuronas ocultas a las 3 de salida. Aquí entran en juego los pesos de la Tabla 3 ($W2_{ij}$), los bias (Tabla 4) y las activaciones de ambas capas ($a1$ y $a2$).
- Determinación del resultado: Un tercer y último ciclo for utiliza la función de activación SoftMax para encontrar la neurona de salida con el valor de probabilidad más alto (entre 0 y 1). Esta es la clase final detectada por la red.

Implementación del prototipo

La Figura 11 muestra el resultado del prototipo implementado., se presenta a un individuo sentado con el prototipo colocado en su brazo derecho.



Fuente: Elaboración propia

Figura 11 Prueba del prototipo.

El display del prototipo muestra el tipo de actividad física que está realizando, que en este caso es “sentado”. Cabe mencionar que este prototipo es incómodo debido al tamaño, sin embargo, en aplicaciones prácticas se buscaría realizar implementaciones más compactas como las bandas inteligentes.

4. Discusión

Las RNA juegan un papel muy importante en las aplicaciones actuales. Las tecnologías emergentes y la habilidad de los sistemas para capturar datos a través de sensores y dispositivos pertinentes generan un caudal de información aparentemente ilimitado. En algunas aplicaciones, como la que se muestra en este trabajo, los datos provienen de señales que se comportan como series temporales. Dichas señales presentan diferentes comportamientos y en algunos casos son indistinguibles entre ellas a simple vista. Por lo anterior, es muy común que se haga uso de la extracción de características para extraer información adicional a las señales. La extracción de características, como se vio en secciones anteriores, incrementa la exactitud de la RNA propuesta, a la hora de la clasificación, lo cual incrementa el número de parámetros de la red. Para aplicaciones donde los recursos de hardware son limitados, esto representa un reto. Adicionalmente, se presentaron resultados de exactitud al entrenar los datos usando regresión logística multiclase y se observó que, a pesar de la simplicidad del algoritmo, de esta última, la RNA ofrece una mayor exactitud (92% contra 96%).

Es importante mencionar que este trabajo está enfocado al uso de una tarjeta donde los recursos de hardware son limitados, como es el caso del Arduino Nano. Sin embargo, actualmente existen en el mercado tarjetas como el “*Arduino Nano BLE sence”* que se caracteriza por manejar algoritmos de “*Machine Learning”* con mayor poder de procesamiento [Barrett, 2022], pero económicamente es más costoso.

5. Conclusiones

El estudio de los patrones de actividad física de los individuos ha sido considerado ampliamente, a diferentes niveles, debido a su importancia en la calidad de vida de las personas. Existen bandas inteligentes o aplicaciones en

dispositivos móviles que realizan estas tareas. Sin embargo, pocos consideran las caídas como una clase de actividad ya que va más orientada a aplicaciones dentro del cuidado de personas que padecen alguna enfermedad, personas de la tercera edad o en aplicaciones en la industria donde se opera con riesgos de sufrir alguna caída. La RNA que se presentó requiere de pocos recursos de hardware, con apenas 4 neuronas en la capa oculta y tres en la capa de salida, generando 55 parámetros y una exactitud de aproximadamente 96%. La regresión logística multiclase, a pesar de ser menos compleja que la RNA ofrece una menor exactitud. En resumen, una red neuronal artificial sencilla como la que se presenta puede integrarse en procesadores limitados en hardware, como el Arduino Nano, debido a sus mínimos requisitos computacionales.

6. Referencias y Bibliografía

[1] Artemiadis, P. K., Kyriakopoulos, K. J. An EMG-based robot control scheme robust to time-varying EMG signal features. IEEE Transactions on Information Technology in Biomedicine, Volume 14, Issue 3, pp. 582-588, 2010.

[2] Arvetti, M., Gini, G., Folgheraiter, M. Classification of EMG signals through wavelet analysis and neural networks for controlling an active hand prosthesis. IEEE 10th international conference on Rehabilitation Robotics, IEEE, pp. 531-536, June 2007.

[3] Barrett, S. F. Arduino Nano 33 BLE Sense. Arduino V: Machine Learning, Springer International Publishing, pp. 17-65, 2022.

[4] Campesato, O. TensorFlow 2 Pocket Primer. Boston: Mercury Learning and Information, Berlin, 2019.

[5] Cavita-Huerta, E., Reyes-Reyes, J. Clasificación de actividad física mediante señales de acelerometría. Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI, Volumen 12, pp. 50-56, 2024.

[6] Dong, Q., Zhu, X., Gong, S. Single-label multi-class image classification by deep logistic regression. Proceedings of the AAAI conference on artificial intelligence, Volume 33, Issue 01, pp. 3486-3493, July 2019.

[7] Fatayerji, H., Al Talib, R., Alqurashi, A., Qaisar, S. M. sEMG Signal Features Extraction and Machine Learning Based Gesture Recognition for Prosthesis Hand. Fifth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), IEEE, pp. 166-171, March 2022.

[8] Hristov, B., Nadzinski, G., Latkoska, V. O., & Zlatinov, S. Classification of Individual and Combined Finger Flexions Using Machine Learning Approaches. IEEE 17th International Conference on Control & Automation (ICCA), IEEE, pp. 986-991, June 2022.

[9] Kim, P. Matlab deep learning: With machine learning, neural networks and artificial intelligence. Apress, 2017.

[10] Lee, K. H., Min, J. Y., Byun, S. Electromyogram-based classification of hand and finger gestures using artificial neural networks. MDPI, Sensors, Volume 22, Issue 1, 2021.

[11] López-Reynaga, B. A., Acevedo-Mosqueda, M. E., Acevedo-Mosqueda, M. A., Gómez-Coronel, S. L. Clasificación de actividades humanas aplicando Inteligencia Computacional. Ingeniería, investigación y tecnología, Volumen 25, Número 2, 2024.

[12] Nacelle, A., Mizraji, E. Redes neuronales artificiales: Núcleo de ingeniería biomédica. Universidad de la República Uruguay, 2009.

[13] Nacke, L. E., Kalyn, M., Lough, C., Mandryk, R. L. Biofeedback game design: using direct and indirect physiological control to enhance game interaction. Proceedings of the SIGCHI conference on human factors in computing systems, pp. 103-112, May 2011.

[14] Naosekpam, V., Sharma, R. K. Machine learning in 3D space gesture recognition. Jurnal Kejuruteraan, Volume 31, Issue 2, pp. 243-248, 2019.

[15] Nope, S. E., Loaiza, H., Caicedo, E. Estudio Comparativo de Técnicas para el Reconocimiento de gestos por Visión Artificial. Avances en Sistemas e Informática, Volumen 5, Número 3, pp. 127-134, 2008.

[16] Ortega-Asensio, E. Sistema de reconocimiento de gestos de la mano basado en procesamiento de imagen y redes neuronales convolucionales. UPCT, Tesis, 2017.

[17] Ozdemir, M. A., Kisa, D. H., Guren, O., Akan, A. Hand gesture classification using time–frequency images and transfer learning based on CNN. Elsevier, Biomedical Signal Processing and Control, Volume 77, 2022.

[18] Paluszek, M., Thomas, S. Practical Matlab deep learning: A Project-Based Approach, Apress, 2022.

[19] Parajuli, N., Sreenivasan, N., Bifulco, P., Cesarelli, M., Savino, S., Niola, V., Gargiulo, G. D. Real-time EMG based pattern recognition control for hand prostheses: A review on existing methods, challenges and future implementation. MDPI, Sensors, Volume 19, Issue 20, 2019.

[20] Páez, R. F. F., Medina, A. D. R. D. Desarrollo, entrenamiento y evaluación de modelos de machine learning para clasificación de imágenes. Revista Científica Estudios e Investigaciones, Volumen 9, pp. 175-176, 2020.

[21] Serrezuela, R. R., Cardozo, M. Á. T., Montiel, J. J. G., Zamora, R. S., Reyes, E. M. Análisis comparativo entre de MAE y RNA en señales de EMG obtenidas para control de una prótesis mano robótica. Memorias de Congresos UTP, pp. 107-112, agosto 2019.

[22] Spiewak, C. A comprehensive study on EMG feature extraction and classifiers. Open Access Journal of Biomedical Engineering and Biosciences, Volume 1, Issue 1, 2018.

[23] Vargas, L. P., Barba Jiménez, L., Mattos, L. Sistema de identificación de lenguaje de señas usando redes neuronales artificiales. Revista Colombiana de Física, Volumen 42, Número 5, 2010.