iMuleta: Muletas sensorizadas para la detección del paso

Trabajo de Fin de Máster

Máster en Ingeniería Informática

Autor: Martin Acosta Llano
Director: Manel Frigola Bourlon

FACULTAT D'INFORMATICA DE BARCELONA
UNIVERSITAT POLITÉCNICA DE CATALUNYA
21 de Octubre de 2019

Abstract

Human Activity Recognition (HAR) is a field of study that seeks to identify the movement or action of a person based on sensor data. In this work we will focus on the recognition of human activities when using crutches. For this, a crutch was instrumented through the interface of inertial sensors and a microcontroller. Data was collected by using the instrumented crutch to perform four different activities such as walking, walking upstairs, walking downstairs and standing. A Machine Learning model was trained with the collected data and the results show that it is possible to identify distinctive movement patterns for each activity with high accuracy.

Resumen

El reconocimiento de actividades humanas (HAR, *Human Activity Recognition*) es un área de estudio que busca identificar el movimiento o acción de una persona en base a datos de sensores. En este trabajo nos enfocaremos en el reconocimiento de actividades humanas al utilizar muletas. Para esto, una muleta fue instrumentada mediante la interfaz de unos sensores inerciales y un microcontrolador. Datos fueron recopilados al utilizar la muleta instrumentada para realizar cuatro actividades distintas como caminar, subir escaleras, bajar escaleras, permanecer parado. Se entrenó a un modelo de Machine Learning con los datos recopilados y los resultados muestran que es posible identificar con alta exactitud patrones de movimiento característicos de cada actividad.

Agradecimientos

Quiero dar las gracias a mi esposa Norma por acompañarme a lo largo de este tiempo, sin ella este trabajo no hubiera sido posible. A mis padres y hermanos, por todo el apoyo recibido. También, a mi tutor Manel Frigola por la guía para la realización de este trabajo.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El reconocimiento de actividades humanas (HAR, *Human Activity Recognition*) es un área de estudio que busca identificar el movimiento o acción de una persona en base a datos de sensores. Existen principalmente dos tipos de HAR: los basados en vídeo y los basados en sensores. El HAR basado en vídeo analiza imágenes que contienen movimientos humanos, mientras que el HAR basado en sensores se enfoca en los datos de movimiento de sensores como acelerómetros, giroscopios, etc [23].

Dado el bajo costo y avance en la tecnología de sensores, la mayor parte de la investigación en el campo se concentra en HAR basado en sensores. Las soluciones basadas en sensores pueden dividirse en tres categorías principales en función a su ubicación [7]:

- Fijados sobre el humano: el usuario lleva puesto los sensores mientras realiza las actividades.
- Fijados sobre un objeto: el usuario interactúa con el objeto mientras realiza las actividades.
- Fijados en el ambiente: los sensores son dispuestos en el lugar donde se realiza la actividad y no se requiere de interacción con el usuario.

Recientemente, los algoritmos de aprendizaje profundo o Deep Learning han logrado un rendimiento incomparable en muchas áreas como reconocimiento visual de objetos, procesamiento del lenguaje natural y razonamiento lógico, por lo que también han sido ampliamente implementados para el reconocimiento de actividades humanas basadas en sensores [23].

En este trabajo nos enfocaremos en el desarrollo de un sistema de adquisición de movimiento basado en sensores inerciales y el de un modelo de Deep Learning para el reconocimiento de actividades humanas al utilizar muletas.

Figura 1.1: Esquema conceptual del sistema iMuleta.

1.1 Motivación

En los últimos años, se ha puesto mucha atención en el desarrollo de exoesqueletos robóticos. La principal función de estos exoesqueletos es la de brindar a su usuario capacidades físicas aumentadas. Personas parapléjicas podrían volver a caminar de forma independiente utilizando un exoesqueleto robótico.

Un factor clave de esta tecnología es la interacción entre el humano y el exoesqueleto. Para lograr una experiencia de uso transparente para el humano, el exoesqueleto debe poder ser controlado de la forma mas natural posible. Para este fin, existen tecnologías que permiten detectar la intención del usuario a través de la actividad eléctrica producida por los músculos esqueléticos.

La detección de la intención a través de la actividad eléctrica de los músculos es utilizado en exoesqueletos como HAL (Hybrid Assistive Limb) [3]. El usuario simplemente mueve sus extremidades como lo haría normalmente y este método puede detectar fácilmente las señales neuronales y entender con éxito la intención del usuario.

El método de detección de intención descrito anteriormente no es aplicable en todos los casos. Los personas parapléjicas no pueden mover la parte inferior del cuerpo y las señales neuronales no pueden alcanzar las extremidades inferiores. Nuevos enfoques para la detección de la intención son necesarios para este tipo de usuario.

Cuando se trata de un usuario parapléjico, el exoesqueleto debe ser usado en conjunto con muletas, éstas ayudan al usuario a mantener el equilibrio cuando está parado o caminando. Por tanto, si relacionamos distintos patrones de movimiento de la muleta con distintas acciones a ser llevadas a cabo por el exoesqueleto, las muletas podrían utilizarse como sistema de control del exoesqueleto, como por ejemplo para dar un paso con el exoesqueleto una vez que la muleta se mueva hacia adelante.

1.2 Objetivos

En este proyecto se buscan cumplir con los siguientes objetivos:

1.2.1 Objetivos Generales

- Desarrollar una muleta instrumentada para la monitorización de la marcha humana.
- Entrenar un modelo de Deep Learning capaz de clasificar distintos patrones de movimiento.

1.2.2 Objetivos Específicos

- Desarrollar un sistema de adquisición de movimiento utilizando sensores de bajo costo.
- Integrar el sistema de adquisición a una muleta de forma no invasiva.
- Construir un dataset con las lecturas de los sensores mientras los usuarios realizan actividades diarias.

1.3 Estado del Arte

1.3.1 Reconocimiento de actividades humanas basada en sensores

Técnicas de Machine Learning han sido aplicadas a la tarea de reconocimiento de actividades en muchos trabajos presentes en la literatura. Los primeros trabajos se enfocaban en extraer manualmente las características a ser utilizadas como entradas para los modelos.

Un modelo bayesiano basado en la toma de decisiones para el reconocimiento de 19 actividades de la vida diaria fue propuesto en [1]. Las actividades fueron realizadas por ocho sujetos utilizando acelerómetros ubicados en cinco lugares diferentes del cuerpo del sujeto. Características como la media, la varianza, la asimetría y la transformada discreta de Fourier se extrajeron utilizando una ventana deslizante de 5 segundos.

En [10] se recolectó datos de 29 usuarios mientras éstos realizaban seis actividades diferentes. Los usuarios tenían un teléfono inteligente en el bolsillo delantero mientras realizaban las actividades y los datos del acelerómetro fueron utilizados para extraer seis características: la aceleración media, desviación estándar, tiempo entre picos, etc. Compararon tres modelos distintos al clasificar las características: árbol de decisión, regresión logística y redes neuronales. Los resultados muestran que la red neuronal tuvo el mejor desempeño.

El principal problema con los trabajos mencionados anteriormente es la necesidad de extraer características manualmente. Estas características diseñadas por los investigadores son difíciles de generalizar, es decir, las características adecuadas para un dominio de aplicación pueden no ser adecuadas para otro dominio.

Para combatir este problema, en [19] se utilizó Deep Learning para la extracción automática de características. Un clasificador k-NN (k Nearest Neighbours) fue utilizado con las características extraídas automáticamente así también con unas características extraídas manualmente. Los resultados mostraron que las características aprendidas automáticamente superaron a las características extraídas manualmente.

1.3.2 Muletas instrumentadas

En los últimos años han sido propuestos varios prototipos de muletas instrumentadas. La mayoría de estos prototipos buscan mejorar el uso de las muletas a través de la monitorización del peso descargado sobre las extremidades inferiores y de los ángulos de inclinación. En [5] se desarrolló un bastón capaz de monitorizar la carga haciendo uso de un resorte compresible dispuesto en el interior del mismo. Este resorte activa unos interruptores cuando se detecta mucha o poca fuerza, y se proporciona retroalimentación a través de vibraciones y luces. En [4] la fuerza ejercida se monitoreó mecánicamente, y para medir la inclinación de la muleta se agregó un interruptor de inclinación de mercurio.

Recientemente, prototipos totalmente inalámbricos fueron propuestos. En [16] se diseño un sistema de muletas capaz transmitir información a una computadora en tiempo real, haciendo uso de un transmisor de radio. Como sensor de fuerza se utilizó una resistencia sensible a la fuerza (FSR) y para medir movimiento un acelerómetro. En [20] se desarrolló otro prototipo similar, con la diferencia que para medir las fuerzas utilizaron galgas extensiométricas.

Mas relacionado con este trabajo se encuentra [18] donde además del diseño de una muleta instrumentada se implementó un modelo de Machine Learning para reconocer los estados de la muleta. Este modelo reconoce los siguientes estados: avanzando, giro a la izquierda, giro a la derecha y estacionario. A diferencia de los otros prototipos, éste no contempla el uso de sensores de fuerza y en cambio incorpora un sensor infrarrojo para detectar cuando la muleta esta en contacto con el suelo.

1.4 Organización del Documento

Para el desarrollo de esta tesis se comienza introduciendo al lector con los conceptos necesarios de Machine Learning. Luego, se describe el prototipo de muleta propuesta, tanto el hardware y software del mismo. Posteriormente, se detallan los experimentos y resultados obtenidos. Por último, se darán las conclusiones y propuestas de trabajo a futuro.

Capítulo 2

Marco Teórico

2.1 Redes Neuronales Artificiales

Las ANNs (Artificial Neural Networks) son sistemas computacionales inspirados por la estructura y funcionamiento de las redes neuronales del cerebro.

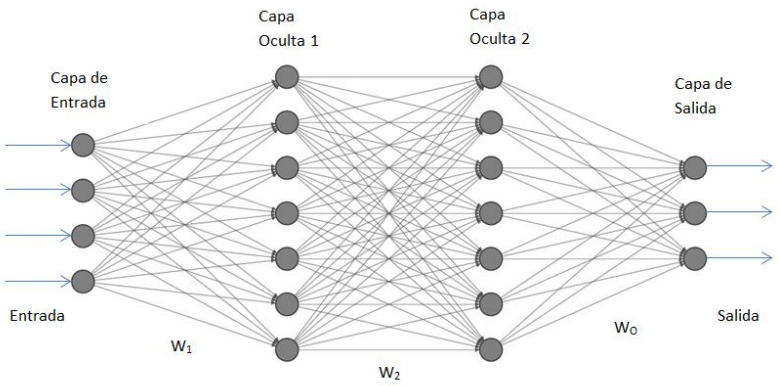
Estas redes se basan en una colección de unidades conectadas llamadas neuronas artificiales o neuronas. Cada conexión entre neuronas transmite una señal de una neurona a otra. La neurona receptora procesa la señal y señala a otras neuronas conectadas a ella.

Las neuronas están organizadas en capas:

- Capa de entrada.
- Capas ocultas.
- Capa de salida.

La estructura se muestra en la Figura 2.1. En general, para una ANN con N capas ocultas hay n_i neuronas en cada capa oculta, n_{in} neuronas de entrada y n_{out} neuronas de salida, donde n_{in} es igual al número de variables de entrada y n_{out} al número de variables de salida.

Figura 2.1: Estructura de ANN completamente conectada.



La fuerza de las conexiones entre neuronas está determinada por una matriz de pesos. Cada neurona en la capa oculta H_i, recibe señales de todas las neuronas de la capa anterior H_{i-1} a través de la matriz W_i y emite señales a todas las neuronas de la siguiente capa H_{i+1} a través de la matriz W_{i+1}.

Entonces, para la *j-ésima* neurona $o_{i,j}$ de la capa oculta H_i tenemos:

$$o_{i,j} = \phi(\sum_{k=1}^{n_i} o_{i-1,k} * W_i^{k,j} + b_{i,j})$$

Donde $b_{i,j}$ es un término de sesgo y $\phi()$ es una función de activación. La función de activación realiza una transformación no lineal a las entradas, lo que hace que el modelo sea capaz de aprender tareas más complejas.

Por último, para la *j-ésima* neurona o_j de la capa de salida tenemos:

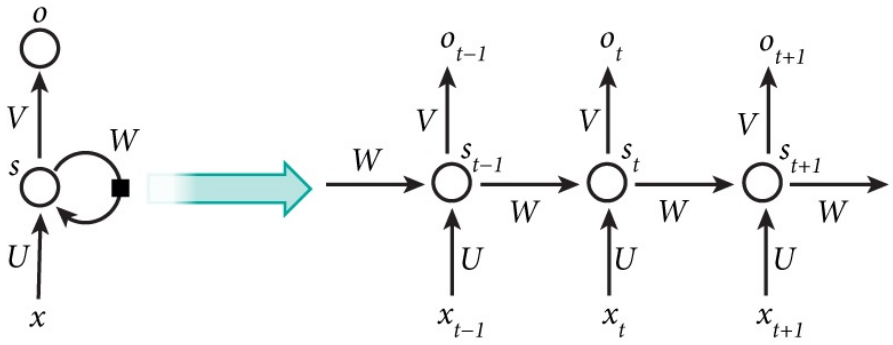
$$o_j = \phi(\sum_{k=1}^{n_N} o_{N,k} * W_o^{k,j} + b_{o,j})$$

2.2 Redes Neuronales Recurrentes

Las RNNs (Recurrent Neural Networks) fueron introducidas para poder trabajar con datos secuenciales, como por ejemplo series temporales. La diferencia con otras arquitecturas es que existen conexiones recurrentes entre neuronas. Lo que se busca con esta arquitectura es retener información a lo largo de la dimensión temporal.

Figura 2.2: Ejemplo de una RNN.

Fuente [12]



La retención de información se logra alimentando la activación de la neurona en el tiempo previo al tiempo actual. Para reducir el número de parámetros del modelo, la matriz de peso es la misma a lo largo de diferentes pasos de tiempo. Como se ve en la Figura 2.2 la matriz de peso W para cada s_t se utiliza en todos los pasos de tiempo t.

En la estructura presentada en la figura 2.2 la ecuación recursiva de la neurona s_t en el tiempo t se define como:

$$s_t = \phi(Ux_t + Ws_{t-1} + b_h^t)$$

donde $\phi()$ es la función de activación y b_h es un término de sesgo.

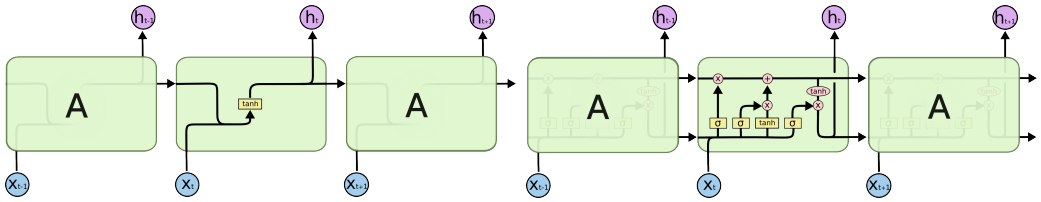
La salida de una capa recurrente puede ser una secuencia de valores o un solo valor. Para el caso de una secuencia, la salida o_t en el tiempo t es:

$$o_t = Vs_t + b_o^t$$

donde b_o se trata de la matriz de sesgo para la salida. Cuando la salida de la capa recurrente es un solo valor, se toma el valor de la capa en la última instancia de tiempo.

2.3 Long short-term memory (LSTM)

Es una versión mejorada de una RNN. Fue diseñada para mitigar los problemas de dependencia de información de largo plazo. Como se aprecia en las figuras de abajo, la estructura de la LSTM es mucho más compleja que la de una RNN clásica.



(a) RNN (b) LSTM

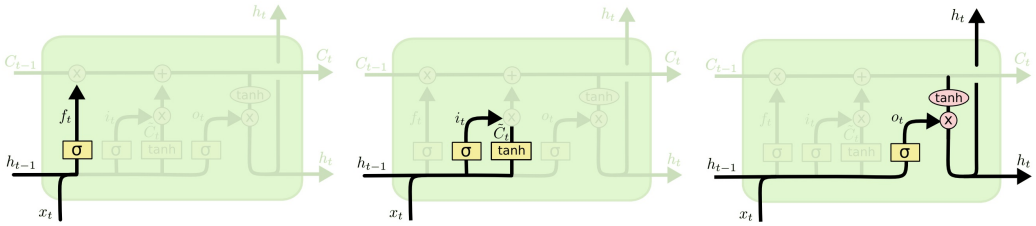
Figura 2.3: Comparación de estructuras entre una RNN y una LSTM.

Fuente [22]

En la Figura 2.3b podemos ver una linea horizontal que atraviesa la cadena de conexiones recurrentes, la cual es conocida como estado de celda. Al estado de celda se lo puede ver como una cinta transportadora que transporta información a lo largo del tiempo.

La información que fluye por el estado de celda es regulada a través de tres puertas conocidas como:

- Puerta de olvido: controla que información se descarta del estado de celda.
- Puerta de entrada: controla que información nueva se añade al estado de celda.
- Puerta de salida: controla cual será la salida de la red. Esta salida se basará en el estado de celda, pero será una versión filtrada.



(a) Puerta de olvido (b) Puerta de entrada (c) Puerta de salida

Figura 2.5: Puertas de una LSTM.

Fuente [22]

De la Figura 2.5a tenemos:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

Esta función produce un valor entre 0 y 1, donde el 0 representa olvidar completamente y el 1 representa retener toda la información.

Luego de la Figura 2.5b tenemos que:

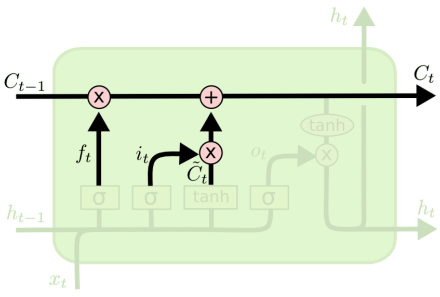
$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C)$$

La función i_t decide que valores serán actualizados y la función $\tilde{C}_t$ determina cuál será el valor actualizado.

Figura 2.7: Estado de celda de una LSTM.

Fuente [22]



Ahora se puede proceder a actualizar el estado de la celda C_t de la siguiente manera:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Por último se genera la salida, de la Figura 2.5c tenemos:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

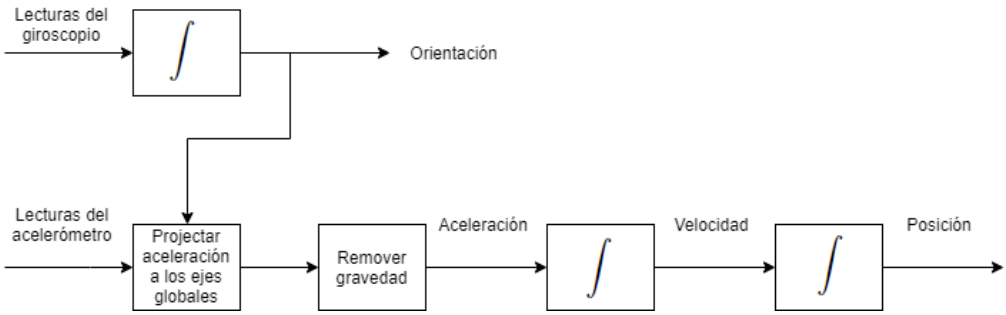
$$h_t = o_t * \tanh(C_t)$$

Con o_t se determina que partes del estado de la celda van a formar parte de la salida y luego al multiplicar con $\tanh(C_t)$ se establece la salida.

2.4 Sensores inerciales para la estimación de posición y orientación

Cuando se requiere conocer la orientación y posición de un dispositivo, típicamente se emplean sensores inerciales. Al integrar las lecturas del giroscopio obtenemos la orientación del dispositivo. Luego, la orientación se utiliza para remover la fuerza de la gravedad de las lecturas del acelerómetro. Sobre este resultado se calcula la integral para obtener velocidad y por último, se vuelve a calcular la integral para obtener la posición del dispositivo. A este método de estimación de posición y orientación se lo conoce generalmente como *navegación por estima* o *dead reckoning*.

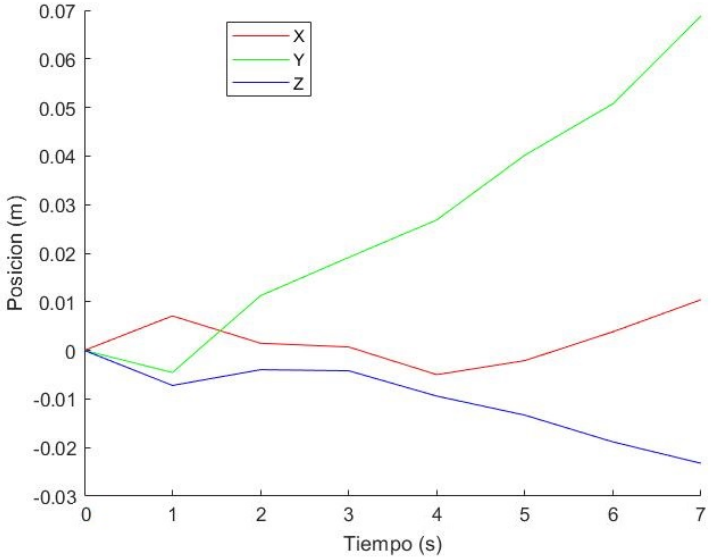
Figura 2.8: Ilustración esquemática del dead reckoning.



En la práctica, los sensores inerciales no son perfectos y sus mediciones sufren de ruido. Esto se traduce en pequeños errores en la estimación de posición y orientación, los cuales, con el tiempo se acumulan y conducen a una diferencia cada vez mayor entre el lugar donde el sistema cree que se encuentra y la ubicación real.

Para ilustrar el problema que esto supone, se recopila datos de una IMU mientras ésta permanece estacionaria. Luego, integramos las medidas de los sensores para obtener la posición del dispositivo. Dado que la IMU se mantiene estacionaria durante la recopilación de datos, se espera que la posición no varíe. Sin embargo, como se muestra en la figura 2.9, la posición deriva a medida que pasa el tiempo.

Figura 2.9: Estimación de la posición de una IMU en estado estacionario basado en la navegación por estima.



Para mejorar las estimaciones, se puede hacer uso de filtros como el de Kalman [8]. También se puede utilizar información de sensores complementarios como magnetómetros para corregir periódicamente la estimación de orientación.

Información sobre posición y orientación podrían ser de utilidad para el problema que busca resolver este trabajo, sin embargo, exploraremos métodos que nos permitan trabajar directamente sobre las lecturas en bruto de los sensores.

Capítulo 3

Diseño del sistema

3.1 Requerimientos

Para cumplir con los objetivos propuestos en este trabajo, primero deben establecerse los requerimientos del sistema a ser diseñado.

Hay que tener en cuenta que las muletas además de ser dispositivos de apoyo para la movilidad, son dispositivos portátiles que acompañan al usuario en su día a día. Por tanto, es importante garantizar que la experiencia de uso de la muleta instrumentada sea la más parecida posible a la de una muleta convencional.

Las propiedades de flexibilidad, resistencia y distribución del peso de la muleta no deben ser modificadas significativamente. La ubicación de la instrumentación añadida debe ser considerada de forma a que no se perturbe el uso normal de la muleta.

Otro factor a tener en cuenta es el costo. Un elevado costo en la instrumentación se vería reflejado en un alto costo del producto final. Esto desmotivaría a la implementación del sistema en productos comerciales para el mercado general.

Como la muleta es un dispositivo portátil, la misma no puede depender de conexiones alámbricas para su funcionamiento. Por lo cual, el sistema instrumentado debe ser alimentado por una batería pequeña y ligera. Para conseguir una larga duración de la batería los componentes que integren el sistema instrumentado deben ser de bajo consumo.

Las variables a ser medidas son la velocidad angular y aceleración de la muleta, además de la fuerza aplicada en el mango. Las mediciones deben transmitirse de forma inalámbrica a un dispositivo de almacenamiento para su posterior análisis.

Figura 3.1: Diseño del sistema en esquema de bloques.

Adaptado de [11]

3.2 Componentes

Para satisfacer los requerimientos establecidos, se deben determinar que componentes utilizar. Se eligen componentes comerciales para minimizar el costo potencial. En la tabla 3.1 se listan las elecciones de los mismos.

Funcionalidad	Componente
Microcontrolador	mbed NXP LPC1768
Sensor inercial	LSM9DS1
Sensor de fuerza	Galgas extensiométricas
Comunicación Inalámbrica	Módulo BLE CC41-A

Cuadro 3.1: Componentes del sistema instrumentado.

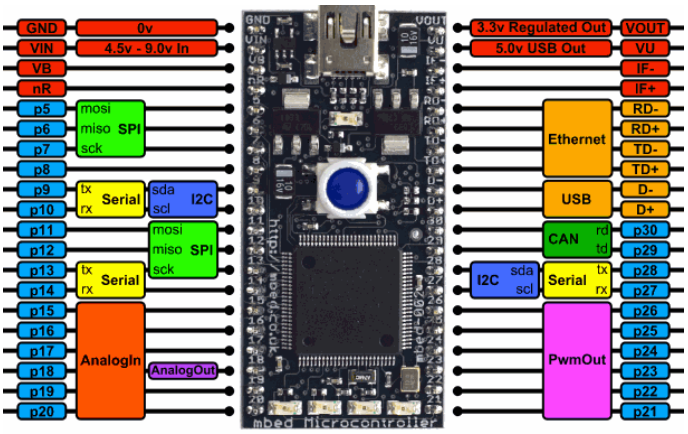
3.2.1 Microcontrolador

Es el componente principal del sistema, encargado de adquirir las mediciones de los sensores y transmitirlas para el posterior procesado de las mismas. En particular se trata de un mbed NXP LPC1768, el cual es capaz de trabajar tanto con señales digitales como con señales analógicas y además, tiene un diseño compacto.

El microcontrolador se programa fácilmente a través de un entorno de desarrollo integrado (IDE) [13] disponible online y utilizando los lenguajes de programación C y C ++. El microcontrolador se comunica con una computadora a través de una interfaz serial USB.

Figura 3.2: Esquema de patillaje de un mbed NXP LPC1768.

Fuente [14].

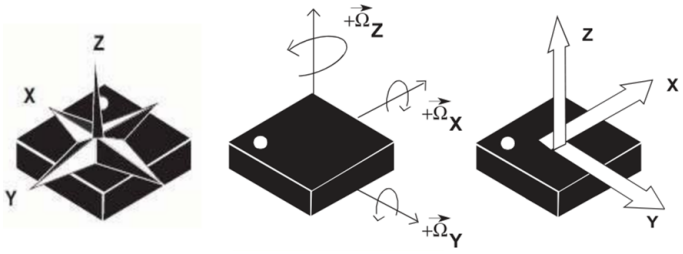


3.2.2 Unidad de medición inercial

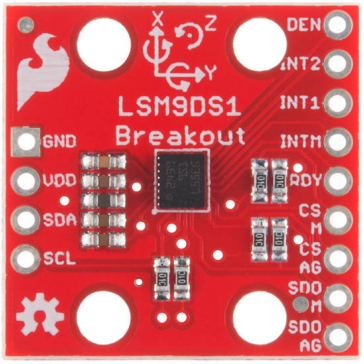
Para medir las aceleraciones y velocidades angulares en los tres ejes se utiliza una unidad de medición inercial (IMU) LSM9DS1 de STMicroelectronics, la cual esta compuesta por un acelerómetro tri-axial, un giroscopio tri-axial y un magnetómetro tri-axial.

Se conecta con el microcontrolador a través de un bus de comunicación I2C, y cada sensor en el LSM9DS1 admite una amplia gama de rangos: la escala del acelerómetro se puede configurar en ± 2, 4, 8 o 16 g, el giroscopio admite ± 245, 500 y 2000 °/s, y el magnetómetro rangos de escala de ± 2, 4, 12 o 16 gauss.

La IMU está integrada en una placa adaptadora, la cual es ideal para la creación rápida de prototipos. La placa se puede enchufar a un protoboard y proporciona el pinout LSM9DS1 completo.



(a) Ejes del magnetómetro, giroscopio y acelerómetro respectivamente .



(b) Placa adaptadora Sparkfun.

Figura 3.3: Unidad de medición inercial LSM9DS1.

Fuente [21]

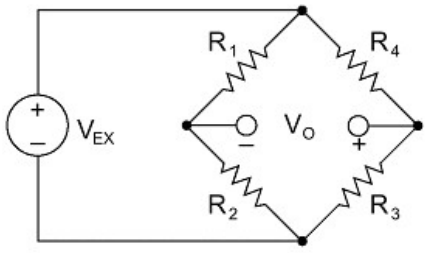
3.2.3 Sensor de fuerza

La fuerza aplicada a la muleta se mide a través de galgas extensiométricas dispuestas en el interior del mango. Estos sensores convierten la tensión mecánica en un cambio de la resistencia eléctrica el cual puede ser medido.

Para medir la deformación de la galga, ésta debe estar conectada a un circuito eléctrico capaz de medir los cambios en la resistencia correspondientes a la tensión. Normalmente se emplean 4 galgas extensiométricas conectadas eléctricamente en lo que se conoce como circuito de puente de Wheatstone.

Figura 3.5: Puente de Wheatstone.

Fuente [15]



El voltaje V_o está definido con la siguiente ecuación:

$$V_o = (\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_2 + R_1}) * V_{EX}$$

De esta ecuación se ve que cuando $\frac{R_1}{R_2} = \frac{R_4}{R_3}$, V_o es igual a cero, cualquier cambio en la resistencia en cualquier brazo del puente da como resultado un voltaje de salida distinto de cero. Por tanto, si se reemplaza a una resistencia con una galga extensiométrica, cualquier cambio en la resistencia de la galga extensiométrica produce un voltaje de salida distinto de cero, de esta forma el voltaje se vuelve una función de la tensión mecánica [15].

3.2.4 Comunicación inalámbrica

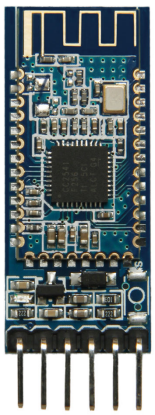
Los datos son transmitidos utilizando un módulo Bluetooth Low Energy (BLE) CC41-A.

El módulo nos permite conectar el microcontrolador a un smartphone o PC de forma inalámbrica, con la facilidad de operación de un puerto serial a través de los pines RX y TX.

En comparación con el Bluetooth clásico, Bluetooth Low Energy está diseñado para proporcionar un consumo de energía y un costo considerablemente reducido, manteniendo un rango de comunicación similar.

Figura 3.6: Módulo Bluetooth.

Fuente [6]

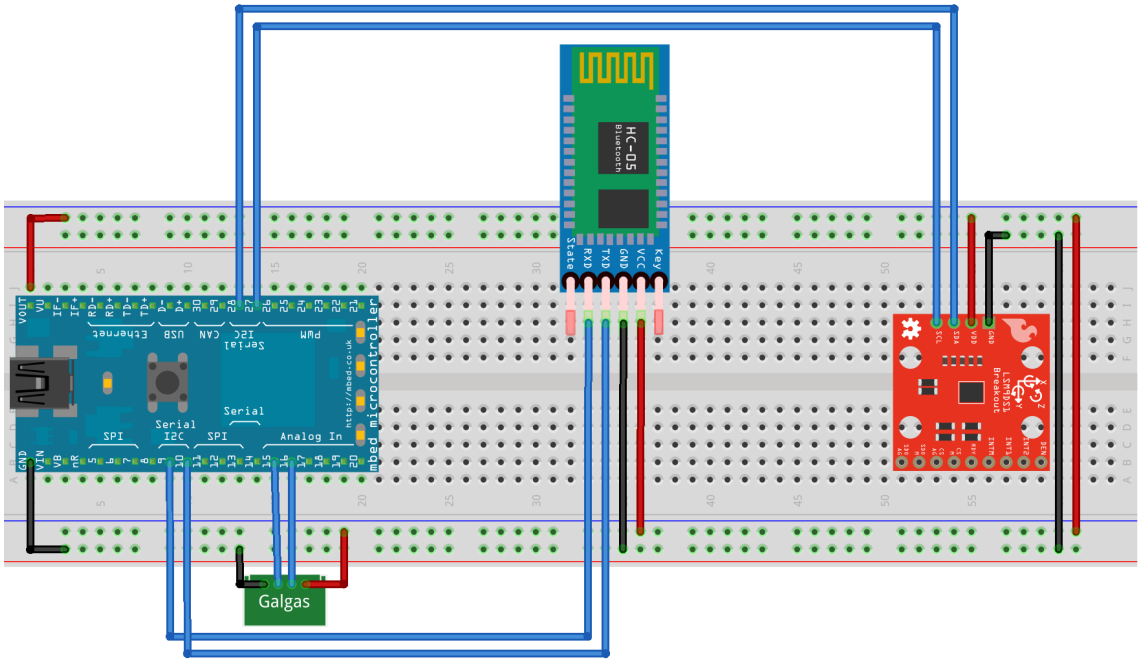


3.2.5 Suministro de energía

El microcontrolador puede ser alimentado con una tensión entre 4.5V y 9V. El suministro de energía para el sistema se obtiene de una batería recargable portátil con una capacidad de 2800 mAh y una tensión de 5 V.

En la Figura 3.7 podemos ver la conexión de todos los componentes del sistema.

Figura 3.7: Conexión de los componentes.

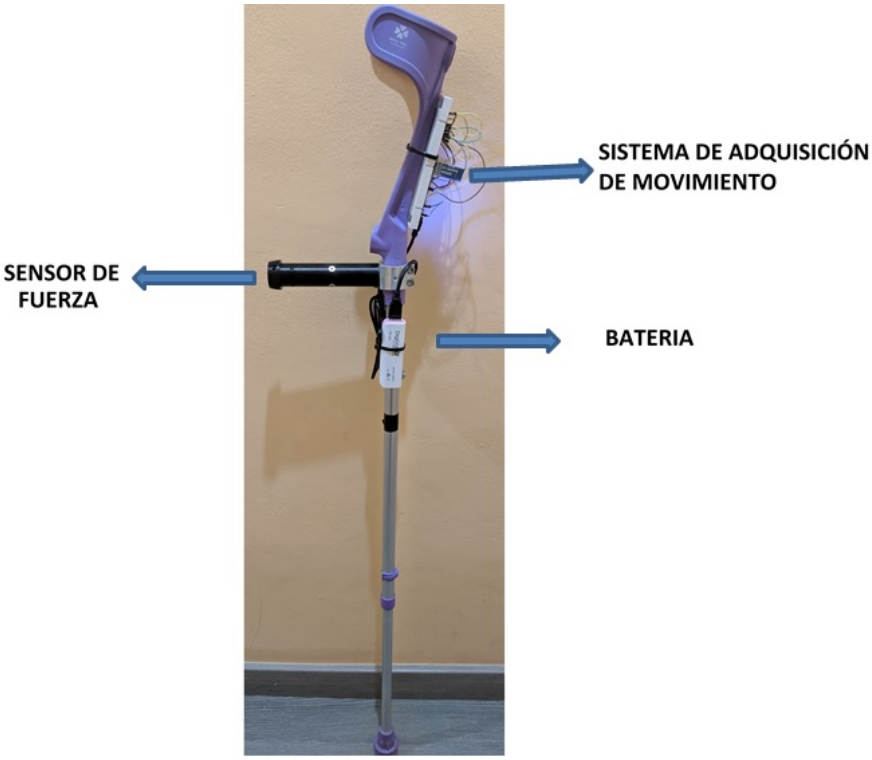


3.3 Prototipo

Como se muestra en la Figura 3.8 el prototipo desarrollado esta compuesto de una muleta de antebrazo al cual se le ha agregado el sistema de adquisición de movimiento y una batería, además el mango fue modificado para incluir sensores de fuerza.

La modificación del mango y la incorporación de los sensores de fuerza a la misma fue realizada por un grupo de alumnos de la *Facultad de Informática de Barcelona* como proyecto para la materia *Proyecto de Ingeniería de Computadores*.

Figura 3.8: Perspectiva del sistema completo.



3.4 Diseño de software

Para poder capturar y almacenar las mediciones de los sensores se requieren de dos piezas de software. Un programa corriendo en el microcontrolador se encarga de la captura y un programa corriendo en un smartphone Android se encarga del control y almacenamiento.

3.4.1 Microcontrolador

El programa mbed esta compuesto por un bucle de control y un bucle temporizado de tiempo real. El bucle de control se ejecuta cada 0.5 s y cuenta con una maquina de estados que consta de 3 estados: inicialización, espera y captura.

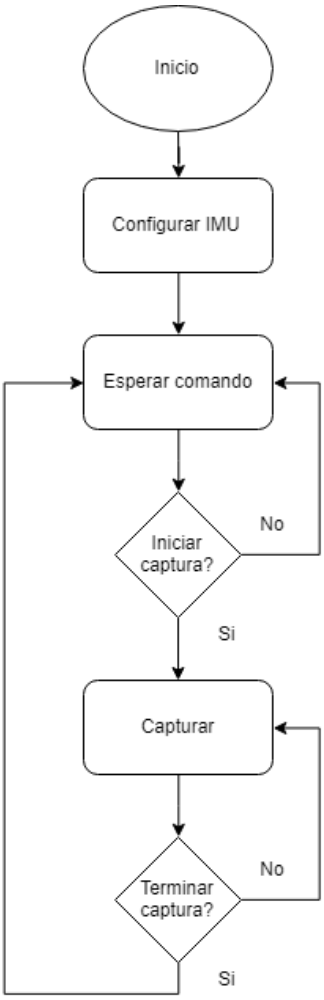
- En el estado de inicialización se configuran los sensores de la IMU para que estén en la escala correcta. El rango de medida del giroscopio es ±245 °/s, el del acelerómetro ±2 g y del magnetómetro ±4 gauss.

- En el estado de espera, el programa espera a que el usuario envíe el comando para comenzar la captura. Una vez recibido el comando, se pasa al estado de captura.

- En el estado de captura, el microcontrolador registra las medidas de los sensores mediante el bucle de tiempo real y las envía a través de Bluetooth para que sean almacenadas. Una vez el usuario envíe el comando para terminar la captura, se vuelve al estado de espera.

La frecuencia de ejecución del bucle de tiempo real es de 30 Hz. En cada iteración, las medidas de los sensores se convierten a la magnitud física correspondiente y éstas son transmitidas al smartphone, siempre que el programa se encuentre en el estado de captura.

Figura 3.9: Diagrama de flujo del programa de captura.



3.4.2 Android

Esta aplicación se empareja con el microcontrolador para controlar los tiempos de inicio y fin de captura, y se encarga del etiquetado de las grabaciones. Tiene dos modos de funcionamiento:

- Modo de registro: a los datos leídos se añade el tiempo de recepción en formato UNIX (timestamp) y son almacenados en memoria de teléfono en formato de valores separados por comas (CSV) para el posterior análisis. Cada fichero CSV es etiquetado según la actividad realizada.
- Modo de inferencia: los datos recibidos no son almacenados en memoria y en cambio son utilizados por el modelo de Machine Learning para realizar predicciones en tiempo real.

Capítulo 4

Experimentos y Resultados

4.1 Dataset

4.1.1 Recolección de datos

El dataset esta compuesto por las lecturas de los sensores de movimiento y fuerza registrados mientras los usuarios realizaban actividades diarias típicas. Las actividades incluyen:

- Caminando.
- Subiendo escalera.
- Bajando escalera.
- Parado.

Se contó con la participación de 3 voluntarios, donde cada participante contribuyó con la misma cantidad de tiempo de grabación. Cada participante realizo cada actividad por un total de 10 minutos, totalizando así 120 minutos de grabación.

Los datos fueron etiquetados utilizando la funcionalidad implementada en la aplicación Android desarrollada.

Figura 4.1: Voluntario realizando la actividad *bajando escalera* durante recolección de datos.

En la Figura 4.2 podemos ver de la estructura del dataset y las variables leídas. Donde *gx, gy, gz* son las mediciones del giroscopio en los ejes X,Y,Z respectivamente. Luego *ax, ay, az* son las mediciones del acelerómetro en los ejes X,Y,Z. Además tenemos que *mx, my, mz* son las mediciones del magnetómetro en los ejes X,Y,Z. Por último *force* es la fuerza medida en el mango de la muleta y *timestamp* corresponde al tiempo de captura.

Figura 4.2: Estructura del dataset.

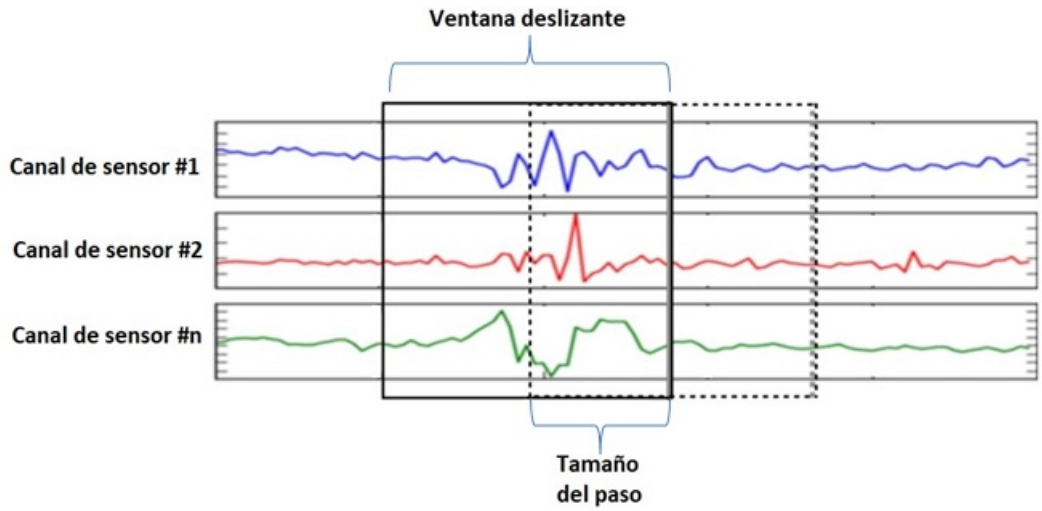
	gx	gy	gz	ax	ay	az	mx	my	mz	force	timestamp
0	-4.688	-2.527	0.052	0.951	0.138	-0.264	0.177	-0.128	0.090	4	1563896661590
1	-1.346	-2.086	-0.950	0.984	0.100	-0.275	0.177	-0.128	0.090	2	1563896661631
2	-0.628	-1.914	-0.778	0.978	0.099	-0.253	0.177	-0.128	0.090	4	1563896661663
3	0.344	-0.935	-0.890	0.999	0.115	-0.225	0.181	-0.129	0.087	0	1563896661696
4	1.877	0.142	-1.211	1.002	0.074	-0.222	0.181	-0.129	0.087	2	1563896661726

4.1.2 Pre-procesado de los datos

Las secuencias de entrada fueron segmentadas a lo largo de la dimensión temporal mediante un mecanismo de ventana deslizante. Mientras se realiza la segmentación, el tamaño del paso es siempre la mitad del tamaño de la ventana de tiempo, ésta relación constante se basa en los experimentos realizados en [17]. En la sección 4.4 se analiza cual es el valor mas apropiado para la longitud de la ventana de tiempo.

Figura 4.3: Mecanismo de ventana deslizante.

Adaptado de [17].



En cuánto a las etiquetas, como se tratan de variables categóricas, éstas deben ser convertidas a un formato con el cual el modelo pueda trabajar. Para lograr esto se aplica la técnica conocida como One Hot Encoding. Ésta técnica transforma la variable categórica de n valores posibles en un vector de n características binarias, con solo una activa a la vez.

Etiqueta	Codificación
Caminando	[1, 0, 0, 0]
Subiendo escalera	[0, 1, 0, 0]
Bajando escalera	[0, 0, 1, 0]
Parado	[0, 0, 0, 1]

Cuadro 4.1: Codificación de etiquetas.

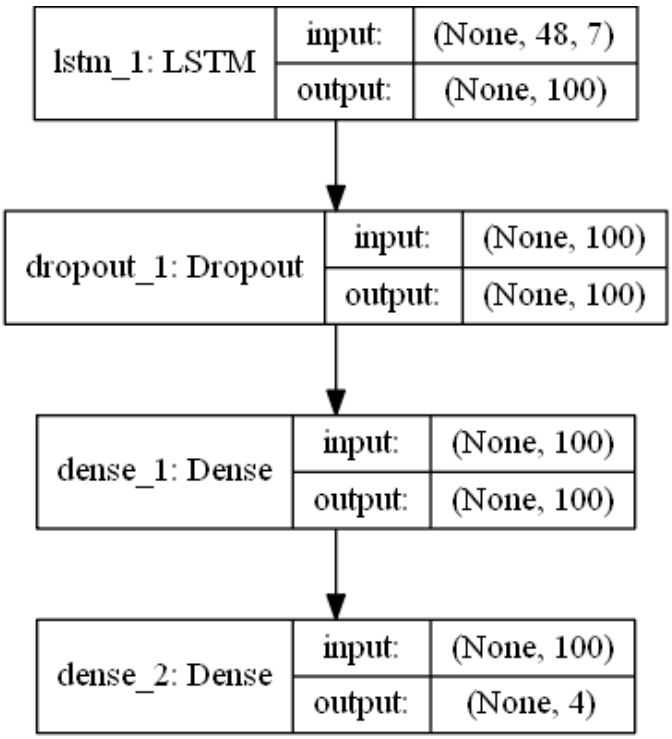
4.2 Modelo

Se entrenó a un modelo con los datos recolectados utilizando la librería de Machine Learning Tensorflow[2] y su API denominado Keras [9]. En la Figura 4.4 se ve la arquitectura del modelo.

El modelo está compuesto de una combinación de capas recurrentes y capas completamente conectadas. Entre la primera capa completamente conectada y la recurrente se dispuso una capa Dropout.

Dropout es una técnica utilizada para evitar que un modelo se sobreajuste a los datos. Funciona desactivando aleatoriamente un número determinado de neuronas en cada actualización de la fase de entrenamiento, lo cual dificulta el aprendizaje al modelo y por ende el modelo es obligado a generalizar mejor.

Figura 4.4: Arquitectura del modelo.



En cuánto al optimizador se utiliza Adam con sus valores predeterminados. Como nuestro problema se trata de clasificación multiclase la opción natural para la función de costo es la entropía cruzada categórica.

En la tabla 4.2 se indican la selección de hiper-parámetros utilizados.

Parámetros	Valor
Tamaño de ventana	48
Ritmo de aprendizaje	0.0001
Tamaño de lote	64
Repeticiones	300
Tasa de Dropout	0.5

Cuadro 4.2: Hiper-parámetros del modelo.

4.3 Resultados

El dataset fue dividido en 2 partes: una para entrenamiento y otra para prueba. El conjunto de datos para entrenamiento comprende el 67 % del total de los datos y el resto de los datos comprende el conjunto para prueba.

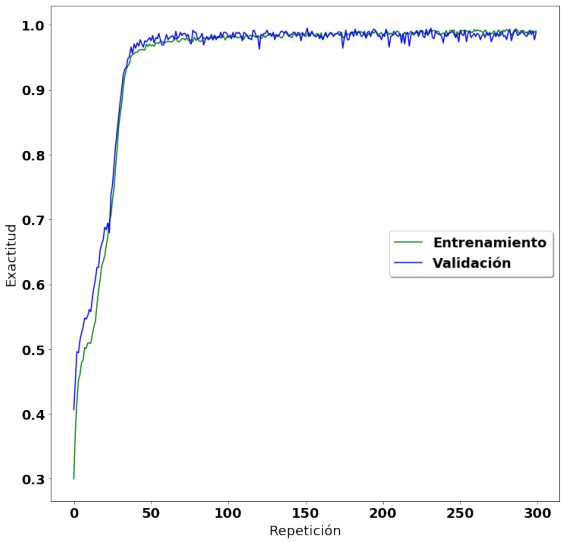
Durante entrenamiento se reservó 10 % de los datos de entrenamiento para validación del modelo.

La métrica utilizada para evaluar el rendimiento del modelo es la exactitud. Formalmente, la exactitud tiene la siguiente definición:

$$\text{Exactitud} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$$

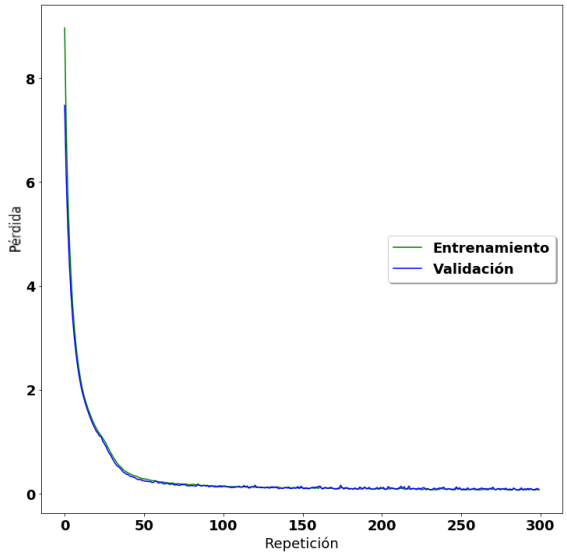
En la Figura 4.5 se puede ver el progreso de la exactitud durante la fase de entrenamiento. Las exactitudes sobre el conjunto de entrenamiento y validación aumentan a un ritmo estable hasta establecerse alrededor del 97 % luego de las primeras 50 repeticiones.

Figura 4.5: Exactitud durante entrenamiento.



En la Figura 4.6 se ve como las pérdidas de entrenamiento y validación decrecen a la par hasta asentarse luego de las 50 repeticiones, por tanto, se puede decir que no existe sobreajuste a los datos.

Figura 4.6: Pérdida durante entrenamiento.

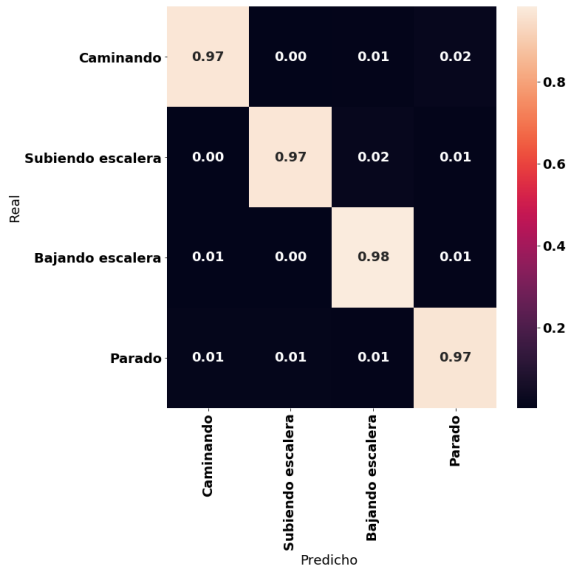


Luego del entrenamiento, se evaluó al modelo sobre el conjunto de datos de prueba. El modelo logró un 97.25 % de exactitud. Si nos fijamos en la Figura 4.7 podemos notar que el modelo logra casi la misma exactitud en todas las categorías. Además, se ve los errores de clasificación mas frecuentes:

- Predice la categoría *bajando escalera* y en realidad se trata de la categoría *subiendo escalera*.
- Predice la categoría *parado* y en realidad se trata de la categoría *caminando*.

.

Figura 4.7: Matriz de confusión normalizada.



4.4 Estudio del tamaño de la ventana de tiempo

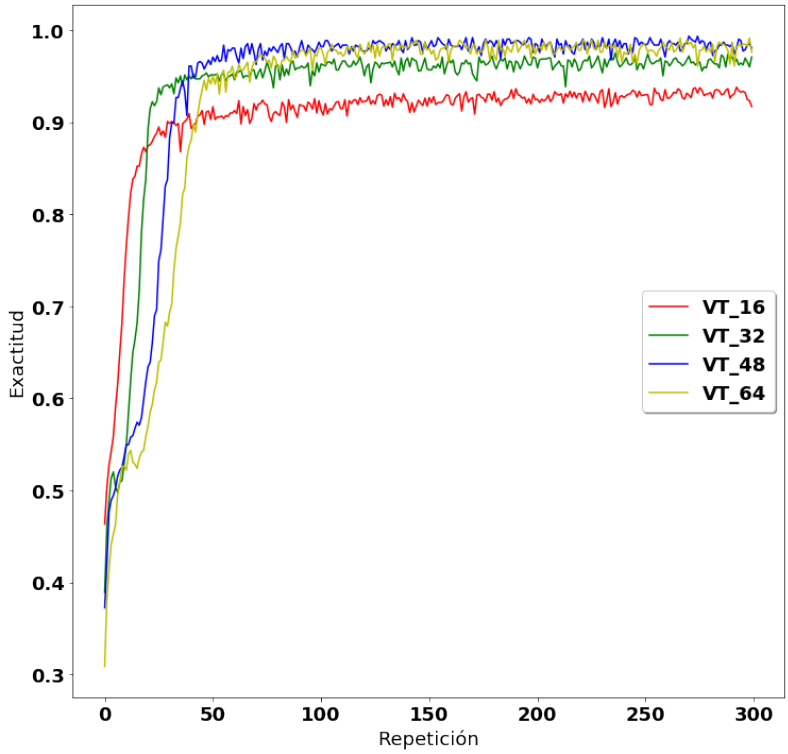
Para determinar el tamaño de la ventana de tiempo más apropiado, comparamos la exactitud lograda por cuatro modelos entrenados con distintos valores de tamaño de ventana. Los modelos utilizan la arquitectura e hiper-parámetros descritos en la sección 4.2. En la tabla 4.3 vemos los valores de ventana adoptados.

Nombre del modelo	Tamaño de ventana (muestras)	Tamaño de ventana (segundos)
VT_16	16	0.5
VT_32	32	1
VT_48	48	1.5
VT_64	64	2

Cuadro 4.3: Tamaños de ventana.

En la figura 4.8 vemos la exactitud de validación de los modelos durante entrenamiento. Se nota que los modelos que logran mejor rendimiento son el VT_48 y el VT_64, seguidos por el VT_32 y por último el VT_16.

Figura 4.8: Comparación de exactitud de validación entre modelos.



Todos los modelos logran una alta exactitud de prueba como se muestra en la tabla 4.4. El modelo que logra mayor exactitud de prueba es el VT_48, por lo cual determinamos que el tamaño de la ventana de tiempo más apropiado para nuestro problema es de 1.5 segundos o 48 muestras por ventana.

Nombre del modelo	Exactitud de prueba
VT_48	**97.25 %**
VT_64	96.81 %
VT_32	96.31 %
VT_16	92.52 %

Cuadro 4.4: Exactitud de prueba.

Capítulo 5

Conclusiones

5.1 Conclusiones

Una muleta instrumentada fue desarrollada con éxito mediante la interfaz de una unidad de medición inercial (IMU) y un microcontrolador. Datos son obtenidos al medir la velocidad angular y aceleración de la muleta, además de la fuerza aplicada al mango. Las mediciones son transmitidas de forma inalámbrica a un teléfono inteligente a través de Bluetooth Low Energy.

La muleta instrumentada fue probada con tres sujetos sanos, a quienes se les pidió utilizar el dispositivo mientras realizaban cuatro actividades distintas. Los datos recolectados durante las pruebas fueron procesados de forma que constituyan un conjunto de datos para el entrenamiento de algoritmos de Machine Learning.

Se entrenó a un modelo de Machine Learning con el conjunto de datos y los resultados muestran que es posible identificar con alta exactitud patrones de movimiento característicos de cada actividad.

Se demostró que el dispositivo tiene el potencial de representar una herramienta no invasiva, barata y portátil para el monitoreo de la marcha humana así como para el control de exoesqueletos robóticos.

5.2 Trabajo a futuro

Se deben realizar pruebas con pacientes reales que utilizan exoesqueletos como asistencia para la movilidad. El objetivo de las pruebas será obtener una evaluación de la potencialidad de la muleta para el control del exoesqueleto y retroalimentación de los usuarios.

También se podría investigar como mejorar la exactitud lograda por el sistema de reconocimiento de actividad. Esto puede conseguirse de distintas maneras:

- Probar distintos hiperparámetros al entrenar el modelo.
- Aumentar el conjunto de datos de entrenamiento.
- Implementar arquitecturas más complejas como DeepConvLSTM [17].

Bibliografía

[1] Kerem Altun, Billur Barshan y Orkun Tunçel. "Comparative Study on Classifying Human Activities with Miniature Inertial and Magnetic Sensors". En: *Pattern Recogn.* 43.10 (oct. de 2010), págs. 3605-3620. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2010.04.019. URL: http://dx.doi.org/10.1016/j.patcog.2010.04.019.

[2] *An Open Source Machine Learning Framework for Everyone*. https://www.tensorflow.org/.

[3] Alex Ansari y col. "A Survey of Current Exoskeletons and Their Control Architectures and Algorithms (Draft 4 . 0)". En: 2015.

[4] Gustav Bergmann y col. "[Walking with walking aids. III. Control and training of partial weightbearing by means of instrumented crutches (author's transl)]." En: *Zeitschrift fur Orthopadie und ihre Grenzgebiete* 117 3 (1979), págs. 293-300.

[5] Jeffery Engel y col. "Walking cane designed to assist partial weight bearing." En: *Archives of physical medicine and rehabilitation* 64 8 (1983), págs. 386-8.

[6] *HM-10 or CC41-A module? Automatic Arduino BLE module identification*. https://blog.yavilevich.com/2016/12/hm-10-or-cc41-a-module-automatic-arduino-ble-module-identification/. Accessed: 2019-08-30.

[7] Zawar Hussain, Michael Sheng y Wei Emma Zhang. "Different Approaches for Human Activity Recognition: A Survey". En: *CoRR* abs/1906.05074 (2019). arXiv: 1906.05074. URL: http://arxiv.org/abs/1906.05074.

[8] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". En: *Journal of Fluids Engineering* 82 (mar. de 1960). ISSN: 0098-2202. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/4756308/35_1.pdf. URL: https://doi.org/10.1115/1.3662552.

[9] *Keras: The Python Deep Learning library*. https://keras.io/. Accessed: 2019-08-30.

[10] Jennifer R. Kwapisz, Gary M. Weiss y Samuel A. Moore. "Activity Recognition Using Cell Phone Accelerometers". En: *SIGKDD Explor. Newsl.* 12.2 (mar. de 2011), págs. 74-82. ISSN: 1931-0145. DOI: 10.1145/1964897.1964918. URL: http://doi.acm.org/10.1145/1964897.1964918.

[11] Matteo Lancini y col. "Upper limb loads during robotic assited gait: a measuring system to guide training". En: oct. de 2017, págs. 613-620. DOI: 10.1142/9789813231047_0074.

[12] Yann LeCun, Y Bengio y Geoffrey Hinton. "Deep Learning". En: *Nature* 521 (mayo de 2015), págs. 436-44. DOI: 10.1038/nature14539.

[13] *Mbed Compiler*. https://ide.mbed.com/compiler/. Accessed: 2019-08-30.

[14] *mbed NXP LPC1768*. https://os.mbed.com/platforms/mbed-LPC1768/. Accessed: 2019-08-30.

[15] *Measuring Strain with Strain Gages*. http://www.ni.com/es-es/innovations/white-papers/07/measuring-strain-with-strain-gages.html. Accessed: 2019-08-30.

[16] Geoff V. Merrett y col. "An instrumented crutch for monitoring patients' weight distribution during orthopaedic rehabilitation". En: *Procedia Chemistry* 1.1 (2009). Proceedings of the Eurosensors XXIII conference, págs. 714-717. ISSN: 1876-6196. DOI: https://doi.org/10.1016/j.proche.2009.07.178. URL: http://www.sciencedirect.com/science/article/pii/S187661960900179X.

[17] Francisco Javier Ordóñez Morales y Daniel Roggen. "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition". En: *Sensors*. 2016.

[18] Youji Ochi. "Development of Crutch-Motion Recognition System Using RNN". En: *Proceedings of the International MultiConference of Engineers and Computer Scientists 2019, IMECS '19, March 13 - 15, 2019, Hong Kong*. Ed. por S. I. Ao y col. Lecture Notes in Engineering and Computer Science. International Association of Engineers. Newswood Limited, 2019, págs. 51-54. ISBN: 978-988-14048-5-5. URL: http://www.iaeng.org/publication/IMECS2019/IMECS2019_pp51-54.pdf.

[19] Thomas Plötz, Nils Y. Hammerla y Patrick Olivier. "Feature Learning for Activity Recognition in Ubiquitous Computing". En: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011. ISBN: 978-1-57735-514-4. DOI: 10.5591/978-1-57735-516-8/IJCAI11-290. URL: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-290.

[20] E. Sardini y col. "Wireless Instrumented Crutches for Force and Tilt Monitoring in Lower Limb Rehabilitation". En: *Procedia Engineering* 87 (2014). EUROSENSORS 2014, the 28th European Conference on Solid-State Transducers, págs. 348-351. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2014.11.745. URL: http://www.sciencedirect.com/science/article/pii/S1877705814028677.

[21] *Sparkfun LSM9DS1 Breakout*. https://learn.sparkfun.com/tutorials/lsm9ds1-breakout-hookup-guide/all. Accessed: 2019-08-30.

[22] *Understanding LSTM Networks*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2019-08-25.

[23] Jindong Wang y col. "Deep learning for sensor-based activity recognition: A survey". En: *Pattern Recognition Letters* 119 (2019). Deep Learning for Pattern Recognition, págs. 3-11. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2018.02.010. URL: http://www.sciencedirect.com/science/article/pii/S016786551830045X.