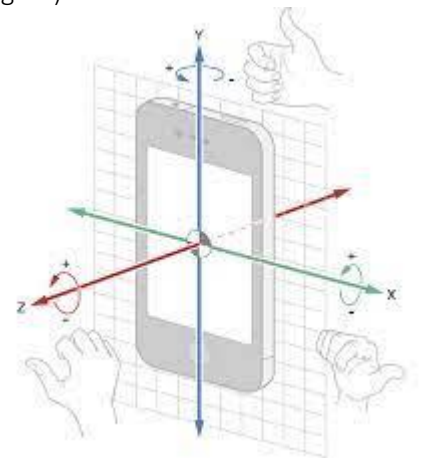
Reconocimiento de movimientos con sensor acelerómetro de teléfono móvil en comunicación con Python

Este proyecto tiene como objetivo el reconocimiento de movimientos realizados con el teléfono móvil mediante un modelo de redes neuronales recurrentes. Seguidamente se describen las etapas del proceso que permitieron desarrollar el modelo antes mencionado:

Obtención de los datos:

Trabajo en Android:

Para la obtención de los datos se desarrolla una aplicación en app inventor (https://appinventor.mit.edu/) software que permite desarrollar aplicaciones para Android. La aplicación desarrollada registra mediciones del sensor denominado acelerómetro que es parte del teléfono móvil, el cual permite medir el movimiento en los tres ejes: X, Y y Z (como se observa en la figura):



Para desarrollar el conjunto de datos de entrenamiento se realizaron cuatro tipos de movimientos con el móvil, el cual es sostenido en la mano derecha del usuario con la pantalla de frente. Los movimientos son:

- Derecha: se gira la muñeca hacia la derecha sosteniendo el móvil mano
- Izquierda: se gira la muñeca hacia la izquierda sosteniendo el móvil con la mano
- Adelante: se gira la muñeca hacia adelante (contrario al usuario) sosteniendo el móvil con la mano
- Atrás: se gira la muñeca hacia atrás (hacia el usuario) sosteniendo el móvil con la mano

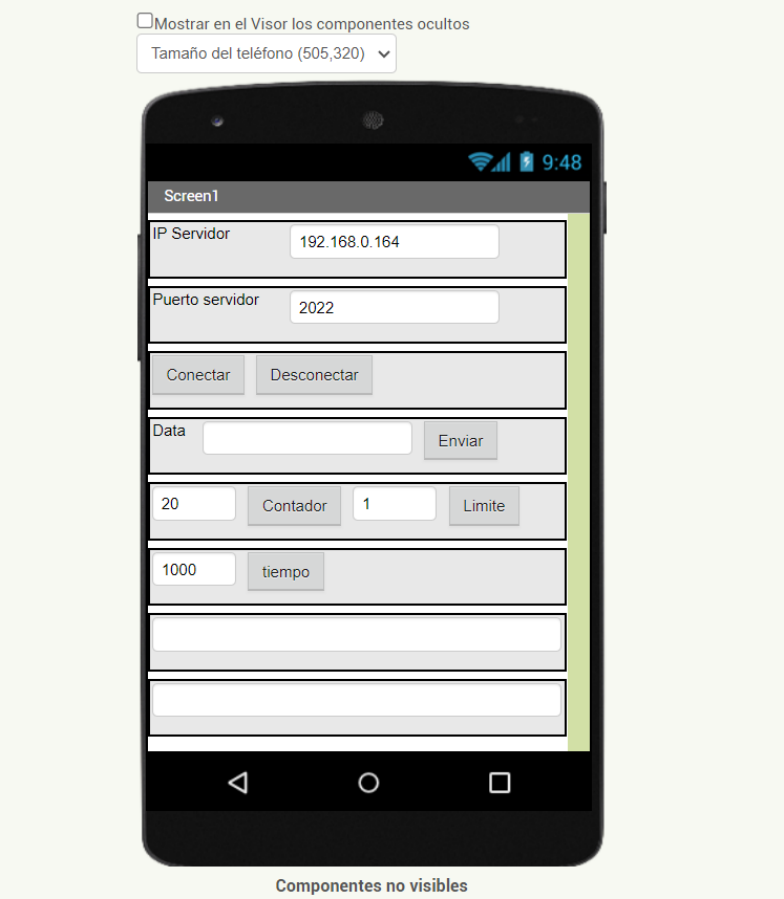
Para cada movimiento se hicieron no menos de 100 repeticiones.

La aplicación toma las mediciones del sensor acelerómetro y las envíe mediante un socket al computador. Las mediciones se realizan de manera periódica, el tiempo de muestreo puede ser definido por el usuario mediante la interfaz gráfica, para la obtención de los datos se utilizó un periodo de 150 ms.

Cada secuencia de datos asociada a un movimiento contiene 15 muestras de datos de cada eje (X, Y y Z). La cantidad de muestras puede ser cambiada por el usuario en la interfaz gráfica de la aplicación.

Otros datos que pueden ser modificados por el usuario mediante la interfaz grafica son:

- Dirección IP a conectarse por el socket
- Puerto a utilizar por el socket
- Botón para conectarse
- Botón para desconectarse



La aplicación envía las mediciones de movimiento de los tres ejes en el siguiente formato: “dd-mm-yyyy HH:MM:SS;medida_eje_x;medida_eje_y;medida_eje_z;validador”.

- dd-mm-yyyy HH:MM:SS: fecha y hora en que se realiza la medida
- medida_eje_x: medición en el eje x
- medida_eje_y: medición en el eje y
- medida_eje_z: medición en el eje z
- validador: esta variable se configura a un valor 0 si la medida es invalida o a 1 si la medida es valida

Las mediciones se transmiten cuando los movimientos superan un umbral de 1, ósea el valor medido en el tiempo actual, en alguno de los ejes, se compara con el valor medido en el instante

inmediatamente anterior (150 ms antes) y si la variación (valor absoluto de la diferencia) es igual o mayor a 1 (valor que puede ser ajustado por el usuario) se envían los datos medidos desde los ejes y la variable “validador” se configura a un valor de 1. En cambio, si la variación antes descrita es menor que 1, se envían valores 0 para cada una de las medidas de los ejes y la variable “validador” se envió con un valor de 0. Una vez iniciado el envío de datos (variación igual o mayor a 1) se envía una secuencia de datos de 15 valores, cantidad que puede ser ajustada por el usuario.

Los valores umbrales para la cantidad de datos enviados (15 datos), para el umbral de variación (valor de 1) y para el periodo de muestreo se obtuvieron mediante prueba y error, repitiendo el proceso en múltiples oportunidades hasta conseguir valores que permitieran obtener datos adecuados para entrenar el modelo.

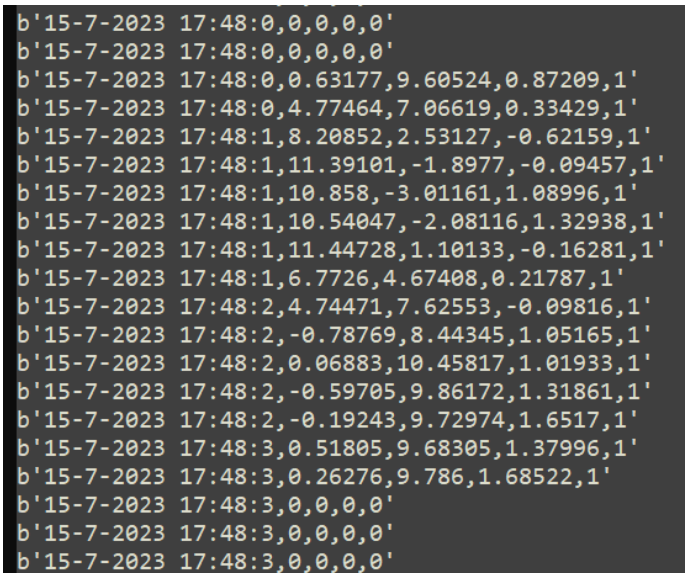
Trabajo en computador:

Como se indicó anteriormente la aplicación en Android se conectaba al computador mediante un socket. Un socket es un punto de ingreso en los extremos de un canal de comunicación bidireccional. Los sockets se pueden comunicar dentro de un proceso, entre procesos dentro de la misma máquina o entre procesos de máquinas diferentes. En particular el socket que se utilizo en este caso esta implementado en la red LAN (wifi) en donde están conectado el computador y el teléfono móvil con Android. Los sockets se pueden implementar en diferentes lenguajes de programación, en este caso se utilizó Python en el extremo del computador y app inventor con una extensión en el teléfono móvil con Android. En la carpeta “captura_datos” del proyecto se puede encontrar la aplicación en Python denominada socket.py que permitió recopilar los datos. Dicha aplicación abre el socket servidor que se encuentra a la espera o a la “escucha” de los clientes. Al conectar la aplicación en Android (que se encuentra en la carpeta antes mencionada) se genera un servidor en el script socket.py, dicho servidor cuenta con un método que le permite recibir datos en un formato binario y como esta dentro de un ciclo while True puede recibir de forma constante información desde el cliente (aplicación en Android). La información recibida se transforma del formato binario a un tipo de dato string con la siguiente forma: “dd-mm-yyyy HH:MM:SS;medida_eje_x;medida_eje_y;medida_eje_z;validador”, mediante el método split de los objetos string de Python se separa en una lista los datos en el formato antes mostrado, separando por el carácter punto y coma (;). En cada iteración del ciclo whie True se agrega la fila de datos a un archivo csv. Finalmente se crean 4 archivos para los 4 movimientos, cada archivo contiene por lo menos 100 secuencias de movimientos.

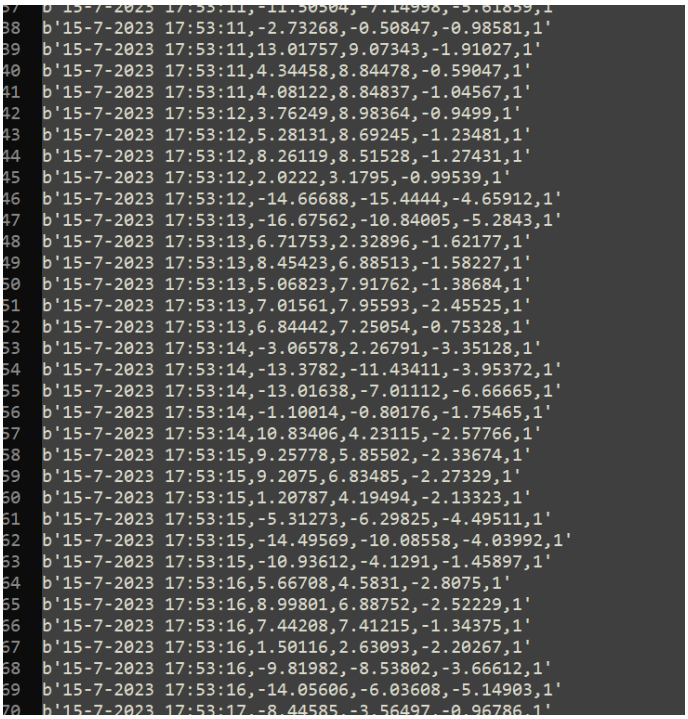
Limpieza de los datos:

Mientras se recopilaban los datos se produjeron problemas de escritura en los archivos csv por lo cual quedaban mal alineadas algunas líneas de datos. Otro problema se producía cuando los movimientos se realizaban con poco tiempo entre si, generándose una secuencia de datos de mas de 15 muestras la cual no servía para entrenar el modelo. Estos problemas se resolvieron mediante inspección de los distintos conjuntos de datos con un editor de texto plano, borrando aquellos conjuntos de datos que tenían mas de 15 muestras o alineando las líneas de datos con problemas.

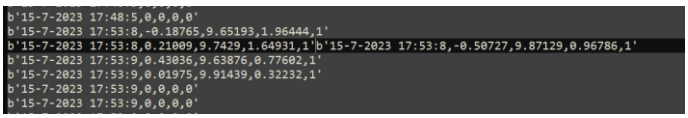
Seguidamente se muestran conjuntos de datos registrados con ejemplos de datos incorrectos y correctos:



Conjunto de datos con 15 muestras y valores en 0 arriba y abajo (correcto)



Conjunto de datos con más de 15 muestras (incorrecto)



Línea de datos con problema de alineación (incorrecto)

Adicionalmente, para asegurar que los datos estaban correctos se procedió a revisar mediante las funciones de inspección de datos que integra la librería panda, funciones que permiten determinar si existen datos NAN (Not a number), determinar si los datos tienen un tipo adecuado, etc.

Visualización de los datos

Una vez se revisaron los datos, se realizó un filtro para solo considera aquellos datos validos mediante considerar solo los datos que tenían un valor de 1 en la cuarta columna ("validador"). Seguidamente se realizo un reformateo de los datos con la función reshape de numpy, para explicar esta acción se utilizará el ejemplo de los datos de movimientos a la derecha:

```
data_derecha.shape
```

```
(1545, 3)
```

En este caso luego de la limpieza y el filtrado de datos queda una secuencia de 1545 registros con 3 canales (los 3 ejes), no obstante nosotros necesitamos secuencias de 15 registros por ello utilizamos la herramienta reshape para dividir en secuencias de 15 registros:

```
data_derecha = data_derecha.reshape(int(data_derecha.shape[0]/15), 15, data_derecha.shape[1])
```

```
data_derecha.shape
```

```
(103, 15, 3)
```

Se observa que luego de aplicar el método de reformateo quedan 103 secuencia de 15 registros con 3 canales, este proceso se repite para cada conjunto de datos (derecha, izquierda, arriba, abajo) y nos permite darles el formato a los datos para poder entrenar la red neuronal recurrente.

Más abajo se muestra una visualización de dos datos en donde podemos observar las secuencias de 15 datos obtenidas. A simple vista se observa que los datos tienen un patrón evidente.

Separación de los datos

Mediante la librería de scikit-learn se dividen los datos en datos de entrenamiento y de prueba. Este ultimo conjunto se utiliza para validar el modelo y comparar las métricas entre datos reales y datos a predecir.

Modelo

Como se mencionó anteriormente se desarrolla un modelo de redes neuronales recurrentes, mediante redes neuronales del tipo LSTM (Long short-term memory), agregando capas de redes neuronales densas y otras capas de dropout utilizando la librería tensorflow. La ultima capa tiene 4 neuronas densas y se utiliza la función de activación softmax para poder lograr la clasificación de las cuatro clases o categorías (movimientos adelante, atrás, derecha e izquierda).

Luego el modelo se evaluo mediante compara las curvas de accuracy y de los con datos de entrenamiento y datos de prueba, también se desarrollo la matriz de confusión y los scores de R2 y accuracy obteniendo valores de 92% y 98% respectivamente, en el jupyter se puede observar las métricas utilizadas. Por ultimo el modelo es guardado en un formato h5.

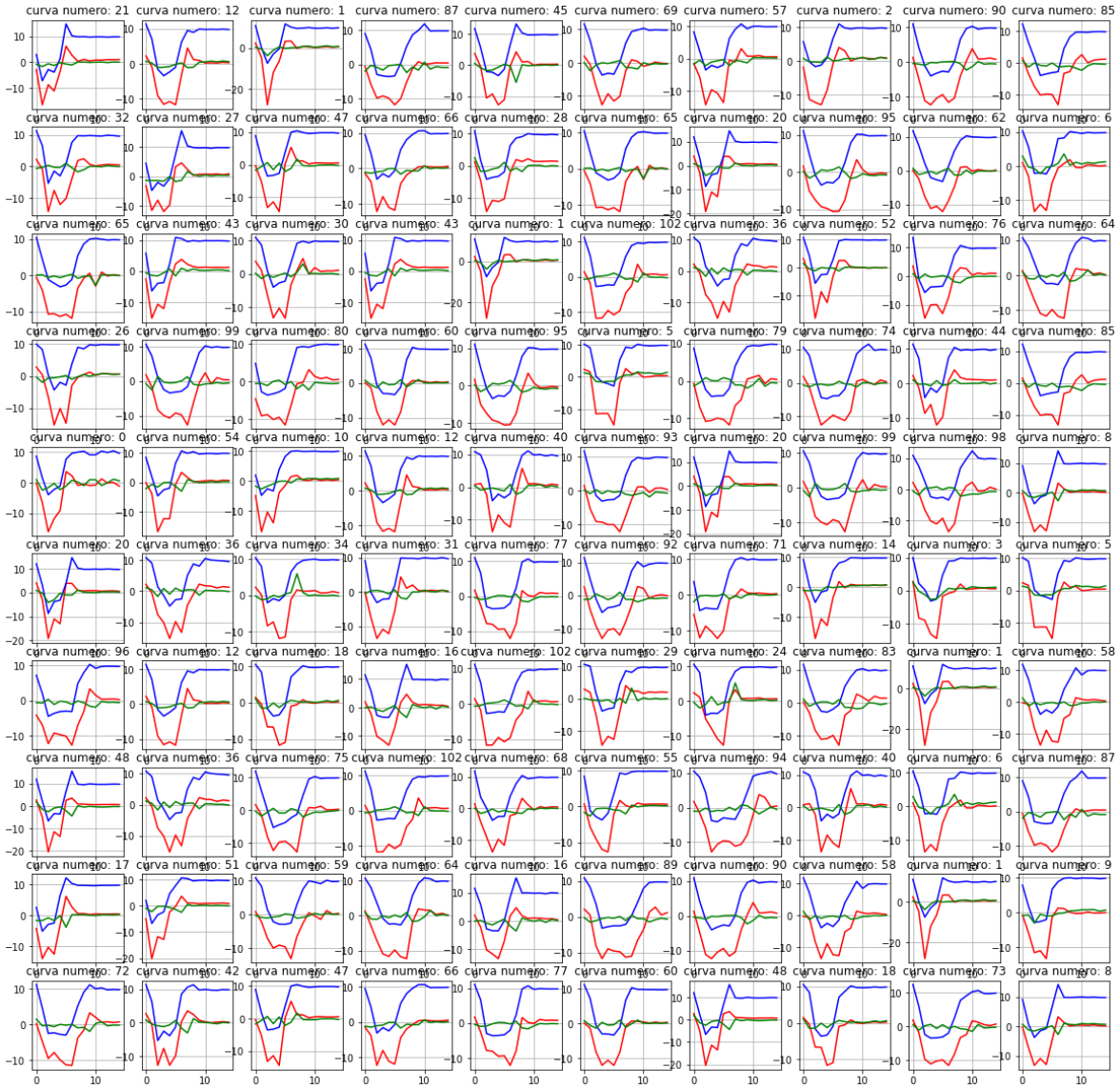
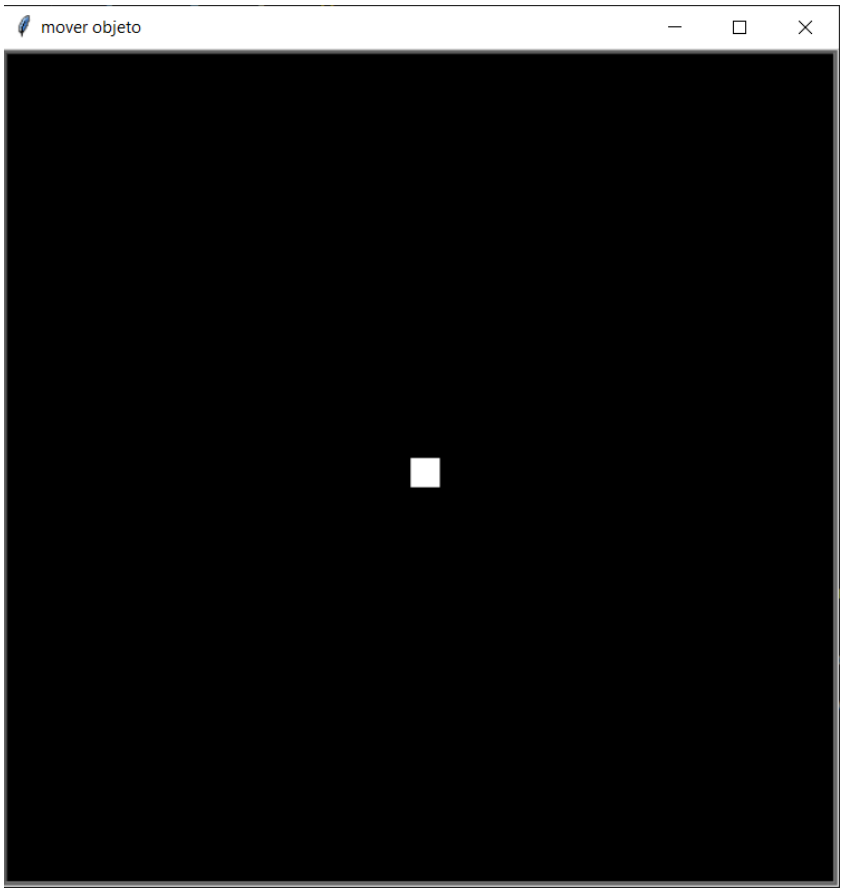


Figura con secuencias de 15 datos para los movimientos hacia la derecha.

APP para prueba

Para probar el modelo en producción se utilizo la misma app desarrollada para el teléfono móvil, la cual se comunico vía socket a una nueva aplicación creada en Python que se encuentra en la carpeta “app”. Esta nueva aplicación muy similar a la utilizada para registrar los datos. La nueva app registra o escucha los datos que vienen con los 3 ejes y el validador, cuando registra un cambio en el validador de 0 a 1 empieza a almacenar los datos hasta alcanzar un total de 15 registros, luego procesa esos 15 registros con sus 3 canales y los envía al modelo previamente entrenado y previamente cargado a esta app y por ultimo el modelo realiza su predicción. La app tiene un segundo hilo o thread para implementar una interfaz gráfica con la librería turtle de Python, una interfaz muy sencilla con un cuadrado en el centro de color blanco y rodeada por un fondo negro (ver la figura más abajo), la idea de este segundo hilo es recibir la predicción del modelo mediante una variable global y luego generar el movimiento del cuadrado (se adjunta un video de demostración en el repositorio github).



Interfaz grafica implementada con la librería turtle de Python para mover el cuadrado con la predicción del modelo previamente entrenado.