Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624 Laboratorio de Microcontroladores

Laboratorio 5: Reconocedor de actividad humana

Pablo Durán Segura B62416

Sergio Garino Vargas B73157

20 de noviembre de 2024

Índice

1. Resumen

En este trabajo, se desarrolló un sistema para el reconocimiento de movimientos humanos utilizando inteligencia artificial. Se creó un modelo de IA con Python y la librería TensorFlow, el cual fue implementado en una placa Arduino BLE 33. El sistema reconoce tres movimientos específicos: mover la mano de arriba a abajo, realizar círculos con la mano y dar un golpe. Para entrenar el modelo, se recopilaron aproximadamente 1200 datos del giroscopio y acelerómetro para cada uno de los movimientos. Posteriormente, se configuró una red neuronal con dos capas principales: una capa de entrada y una capa oculta con varias neuronas, además de una capa de salida con tres neuronas, cada una correspondiente a uno de los movimientos. Se utilizó la función de activación ReLU en las capas ocultas y softmax en la capa de salida, mientras que el modelo se entrenó con el algoritmo de optimización RMSprop y la métrica de pérdida MAE. Finalmente, el modelo entrenado se exportó a un archivo en formato .h y se cargó en la placa para permitir su funcionamiento en tiempo real. En el siguiente enlace se puede acceder al repositorio de github: https://github.com/PabloJoseD/Lab-5

Palabras claves: *Arduino, Python, TensorFlow, giroscopio, comunicación serial, redes neuronales.*

2. Nota teórica

2.1. Características del Arduino Nano 33 BLE

El Arduino Nano 33 BLE tiene las siguientes características.

- Procesador 64 MHz Arm Cortex-M4F
- 1 MB Flash + 256 KB RAM
- Bluetooth 5
- 14 pines digitales y 8 analógicos.
- Periféricos UART, I2C, SPI, y PWM.

- Unidad de Medición Inercial LSM9DS1, utilizando que incluye acelerómetro, giroscopio y magnetómetro.
- Barómetro y sensor de temperatura LPS22HB.
- Sensor de humedad relativa HTS221.
- Micrófono integrado.

A continuación se muestra el diagrama de pines del Arduino.

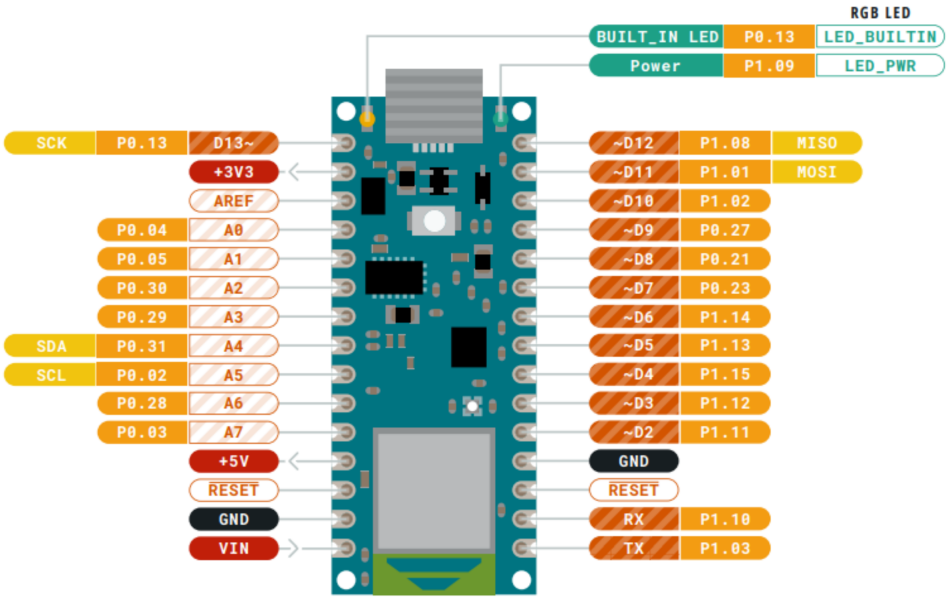

Figura 1: Diagrama de pines del Arduino BLE 33.[1]

2.2. Periféricos

En este caso no se utilizaron periféricos, solo el giroscopio y el acelerómetro que ya están incorporados en el Arduino. A continuación, se detallan algunas características de estos sensores.

- Módulo LSM9DS1
- Mide la posición angular y aceleración lineal en los tres ejes (X, Y, Z).
- ±245/±500/±2000 dps rango completo de velocidad angular.

- ±2/±4/±8/±16 g rango completo de aceleración lineal.
- Salida de datos de 16 bits.

2.3. Registros

Para este laboratorio no se utilizó ningún registro del Arduino. Todos los programas se desarrollaron por medio de bibliotecas que contienen funciones útiles para manejar los sensores y la comunicación serial.

2.4. Diseño del circuito

El circuito de este laboratorio es básicamente el Arduino conectado a la computadora mediante un cable USB. Los datos de los sensores se envían a la computadora por un puerto serial y con un programa de Python se capturan esos datos.

2.5. Componentes

Componente	Precio/unidad	Cantidad
Arduino BLE 33	15200	1

Tabla 1: Tabla de componentes

3. Desarrollo y análisis de resultados

Como ha sido costumbre en todos los laboratorios la metodología para resolver la tarea planteada fue partir desde lo más básico hasta llegar a la solución final, se parte con el objetivo en mente de desarrollar una aplicación con un Arduino Nano BLE 33 Sense de manera que este sea capaz de reconocer tres movimientos hechos con la mano, utilizando las medidas del giroscopio incorporado.

Tal como se recomendó, lo primero que se hizo fue realizar ejemplos que se encuentran en línea para familiarizarnos con el dispositivo, el primer tutorial fue cargar al arduino un programa que corre una algoritmo de ML capaz distinguir entre las palabras “yes” y “no”. Este ejemplo nos permitió

comprender cómo cargar el programa al microcontrolador.

El segundo tutorial fue similar al objetivo de este proyecto, este consistió en entrenar un modelo capaz de detectar dos movimientos un puñetazo y una flexión de brazo, al desarrollar este programa pudimos comprender cómo se capturan los datos para entrenar el modelo, aprovechamos este ejemplo para desarrollar nuestro propio script de Python capaz de capturar y guardar en un archivo los datos generados por el microcontrolador, ya que en este tutorial se propone copiar y pegar desde el monitor serial.

También se pudo entrenar un modelo de ML capaz de reconocer los movimientos, para ellos se tomó como base este notebook de Google Colab, desarrollado por Sandeep Mistry y Don Coleman como parte de este mismo ejemplo, el principal reto fue hacer funcionar el script ya que está un poco desactualizado y algunas secciones fallaban, una vez solucionado se logró obtener un archivo `model.h` que contiene el modelo entrenado, es interesante ya que es solo un archivo con números hexadecimales. Los mismo autores facilitan un script de arduino capaz de utilizar dicho modelo para reconocer los movimientos, así fue como se llevó a la placa el programa y se logró detectar ambos movimientos con muy buena precisión.

Luego se siguió un tutorial que aprovecha Edge Impulse para detectar objetos como plantas o lámparas, decidimos seguirlo pero adaptando los objetivos a nuestros intereses, es decir, se entrenó el modelo para detectar los movimientos en vez de objetos, desafortunadamente esta ya que luego de hacer todo el procedimiento no se logró cargar el modelo a la placa, a continuación se explica todo lo que se hizo en Edge Impulse.

Se inicia construyendo las muestras con las que se entrenará el modelo, se usaron los archivos CSV del ejemplo anterior. Se cargaron los datos en la plataforma, se seleccionó la opción de usar 80 % de los datos para entrenamiento y 20 % para pruebas. Lo siguiente es **diseñar un impulso**.

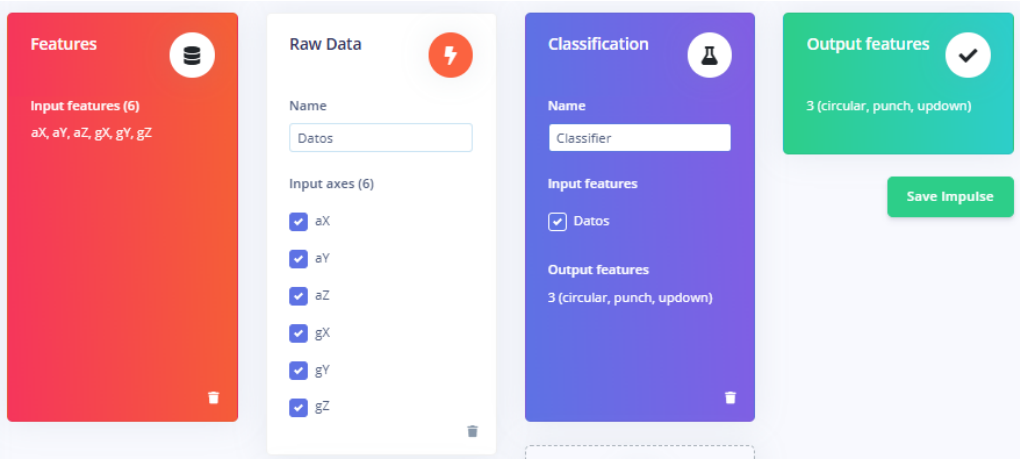

Figura 2: Diseño del impulso

De izquierda a derecha se tiene un bloque de Features donde se indica que las seis entradas son las coordenadas XYZ de las medidas del giroscopio y acelerómetro, el bloque color blanco es de procesamiento, la opción recomendada por Edge Impulse fue RAW DATA (útil para Deep Learning), el bloque color morado es de aprendizaje, la opción recomendada fue usar un algoritmo de procesamiento. A la derecha, el bloque color verde define las tres posibles salidas del modelo, movimiento circular, un golpe o arriba-abajo.

Se configuró el bloque de procesamiento.

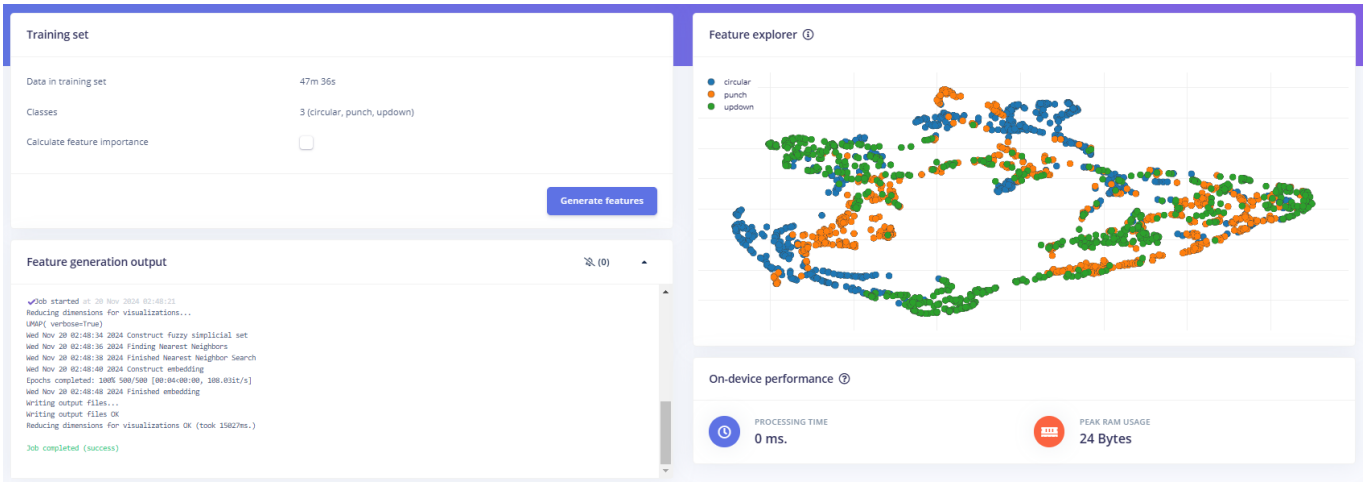

Figura 3: Bloque de procesamiento

Los datos no son están esparcidos a lo largo de todo el mapa lo que quiere decir que no son fácilmente identificables.

Se entrenó el modelo.

Figura 4: Entrenamiento del modelo

Se hicieron pruebas en el modelo.

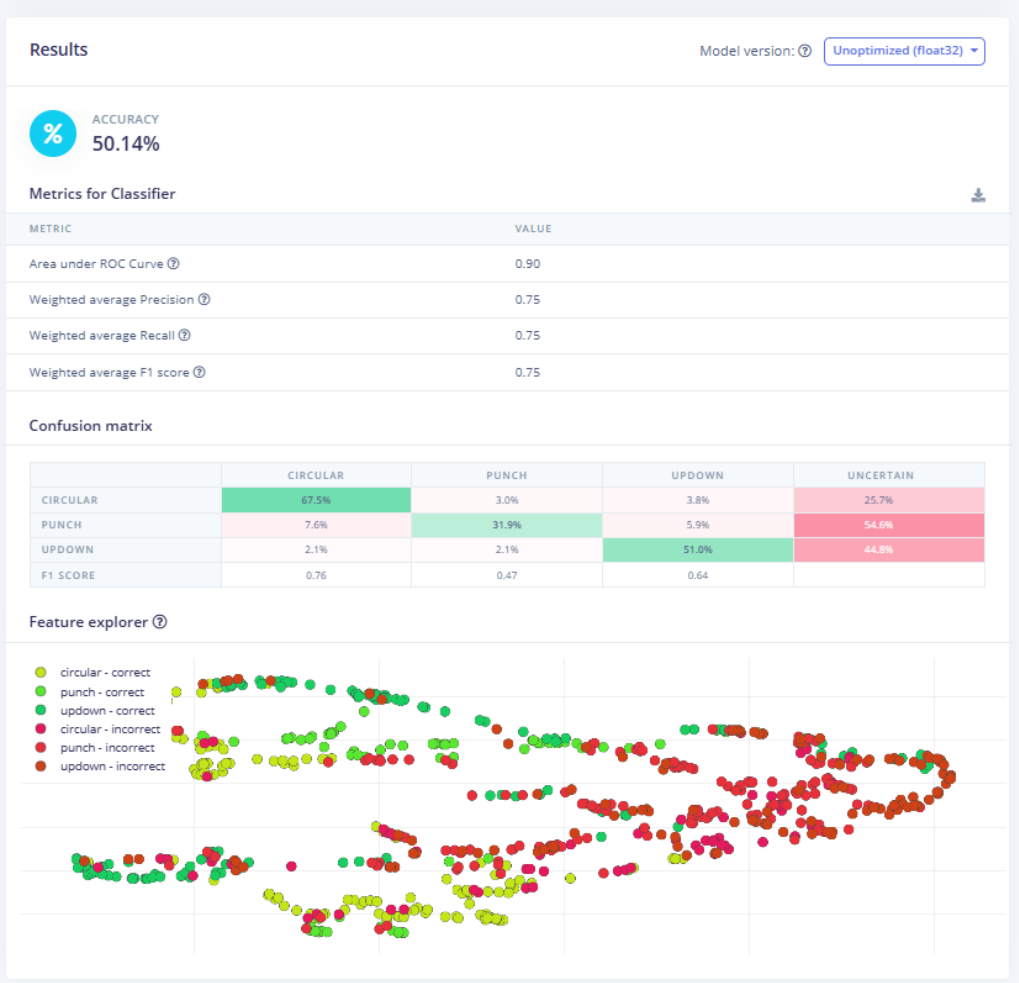

Figura 5: Enter Caption

Se intentó desplegar el modelo en la placa tal como lo indica la documentación, se seleccionó el despliegue como una librería de arduino, se descargó el archivo .zip y se instaló como una librería de arduino, se seleccionó uno de los ejemplos incluidos pero al momento de cargar el archivo a la nano 33 no se pudo, se quedó pegado en “uploading file”.

La opción más viable fue adaptar el segundo tutorial a tres movimientos, ya que este cumple con todas las características que se piden en el Laboratorio, la red neuronal tiene su capa de entrada, una capa intermedia que usa ReLu como función de activación y a la capa de salida se modificó para que sean tres salida y usa softmax. También utiliza el algoritmo de optimización rmsprop y métrica de perdida mae.

Hay varias consideraciones a tomar en cuenta, primero a pesar de tomar que se recomienda que uno de los movimientos sea la posición estática este no se incluye ya que la captura de datos y el modelo tienen un umbral de 2.5G para detectar movimiento, es decir, si el microcontrolador esta quieto, no lee ningún dato ya que necesita acelerarse a 2.5G para medir, se intentó eliminar este umbral pero la detección es errática ya que el microcontrolador siempre está analizando datos. Segundo, el enunciado pide hacer la toma de datos solo con el giroscopio, a diferencia del ejemplo que usa giroscopio y acelerómetro, se intentó usar solo el giroscopio, pero la detección de movimientos fue imprecisa, por esta razón es que también se hizo entrenó al modelo con los datos de ambos sensores.

Se probaron cinco movimientos distintos, rotar el micro sobre su eje horizontal, flexiones, círculos verticales, puñetazos y arriba-abajo. los que mejor funcionaron son los últimos tres.

En un inicio a precisión no fue del 100 % pero era aceptable, por ejemplo, los círculos que se detectan son en sentido horario y despacio, si se hace anti-horario o muy rápido lo señala como arriba-abajo, a pesar de que se entrenó para detectar ambos sentidos del círculo. En algunas ocasiones los puñetazos los detectaba como arriba-abajo. Se notó que el modelo tenía una especie de memoria y favorecía hacer el mismo movimiento varias veces.

Por esta razón se hizo una nueva captura de datos, esta vez solo se hicieron diez repeticiones por movimiento, lo cual resultó en un modelo más preciso. A continuación, se muestran los resultados obtenidos.

Movimiento arriba y abajo

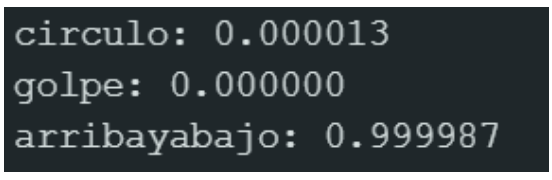

Figura 6: Arriba y abajo

Movimiento circular

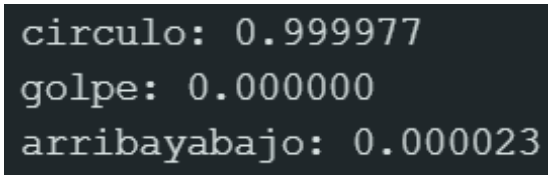

Figura 7: Círculo

Golpe

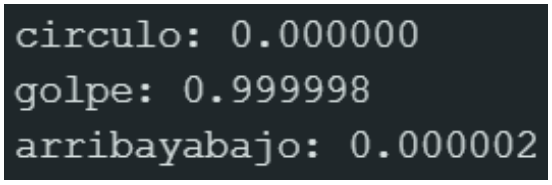

Figura 8: Golpe

La detección no es 100 % precisa, notamos que la principal falla es que a veces detecta arriba y abajo para los otros dos movimientos.

Queda como tarea pendiente entrenar el modelo con más movimientos a detectar y que sea capaz de decir que no reconoce el movimiento.

4. Conclusiones

- Es posible crear un dispositivo de HAR con un Arduino Nano BLE 33 Sense que pueda detectar movimientos.
- La aplicación más óptima de este laboratorio sería entrenar al modelo para que sea capaz de detectar caídas y mande una alerta, esto puede ser de gran ayuda a personas adultas mayores. La primera limitante es la alimentación ya que debe estar conectado a la computadora y que el microcontrolador no nos pertenece y una caída podría dañarlo.
- La herramienta TensorFlow Lite es muy útil y facilita el trabajo de crear modelos de inteligencia

artificial. Además permite desplegar estos modelos en sistemas empotrados para resolver tareas de diferentes tipos.

- La precisión del modelo depende de la cantidad de datos que se utilicen para entrenarlo, aunque no es el único parámetro que define que tan bueno será el mismo. También depende de otros factores como la cantidad de neuronas, capas, algoritmos de optimización, etc.
- Para entrenar una red neuronal es mejor hacerlo directamente con un programa en Python que haciéndolo con EdgeImpulse. Esto le da más libertad para configurar los parámetros de la red.
- Al momento de capturar datos, más muestras no implicar que la red sea mejor, es más importante que los datos sean tomados bien.

5. Recomendaciones

- Se recomienda entrenar al modelo con la misma cantidad de datos para cada tipo de movimiento.
- Es importante comprender bien los conceptos de IA para lograr entrenar un buen modelo que se ajuste a la aplicación que se esté desarrollando.
- Para tomar datos de gestos se recomienda hacerlos de diferentes formas. Por ejemplo para un gesto de golpe de brazo se podrían tomar varias mediciones de datos a diferentes velocidades, primero lento, después intermedio y luego más rápido. Esto mejoraría la respuesta del modelo ya que en una situación real un golpe no siempre se da de una misma forma.
- Se recomienda leer con detenimiento los tutoriales de Arduino para Machine Learning ya que si nunca se ha trabajado con inteligencia artificial, puede resultar complicado saber cómo empezar y que pasos seguir. La documentación de Arduino contiene muchos recursos útiles para guiarse en esta área.

Referencias

[1] Arduino. Nano 33 ble. [Online]. Available: https://docs.arduino.cc/hardware/nano-33-ble/