Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE-0624 Laboratorio de Microcontroladores

EIE
Escuela de
Ingeniería Eléctrica

Laboratorio #5

UNIVERSIDAD DE
COSTA RICA

Grupo 01

Profesor: MSc. Marco Villalta Fallas

Estudiante: Luis Brenes Campos — **Carné:** C21324
Estudiante: Elizabeth Matamoros — **Carné:** C04652

I SEMESTRE 2025

Índice

1. Introducción

En este laboratorio se implementó un sistema de reconocimiento de actividad humana (HAR) utilizando un microcontrolador Arduino Nano 33 BLE Sense, el cual incluye sensores integrados como acelerómetro y giroscopio. El objetivo fue aplicar un flujo de trabajo completo que integrara adquisición de datos, entrenamiento de un modelo de aprendizaje automático y validación de la inferencia directamente en el microcontrolador.

El proceso inició utilizando un ejemplo base de la librería Arduino_LSM9DS1.h, el cual permitió activar y leer datos del giroscopio del Nano 33 BLE Sense. Con este ejemplo, se generaron lecturas en tiempo real que luego fueron capturadas por un script en Python, encargado de recibir los datos vía puerto serial y almacenarlos en archivos CSV. Esta fase permitió construir un conjunto de datos etiquetado con diferentes movimientos para su posterior análisis.

Posteriormente, se utilizó un cuaderno de Jupyter en Google Colab para realizar el entrenamiento del modelo de clasificación. Esta parte fue desarrollada empleando herramientas como TensorFlow para definir, entrenar y exportar un modelo de red neuronal adecuado para ser ejecutado en microcontroladores. La elección de Colab permitió realizar este entrenamiento de forma eficaz, facilitando pruebas rápidas y ajustes en el diseño del modelo.

Finalmente, con el modelo entrenado en formato compatible con TensorFlow Lite, se procedió a integrarlo en un sketch de Arduino. Esta etapa permitió validar el funcionamiento del modelo en el propio microcontrolador, verificando que la inferencia pudiera realizarse correctamente en tiempo real. Es importante recalcar que el laboratorio permitió comprender cada una de las etapas necesarias para construir un sistema de reconocimiento embebido, desde la adquisición de señales hasta el despliegue final del modelo.

2. Nota teórica

Para sustentar la información técnica presentada en esta sección, se utilizaron como referencia los documentos oficiales del microcontrolador y la placa empleada en el laboratorio. En particular, el *nRF52840 Product Specification* [10], que describe detalladamente las capacidades del microcontrolador incluido en el módulo NINA B306, y el *datasheet* de la placa Arduino Nano 33 BLE Sense [11], el cual especifica las características eléctricas, periféricos integrados y sensores utilizados. Ambos documentos fueron fundamentales para comprender el funcionamiento interno del sistema y justificar el uso de sus componentes durante el desarrollo del laboratorio.

2.1. Información general del Arduino Nano 33 BLE Sense Lite

2.1.1. Microcontrolador nRF52480

El módulo central es un NINA-B306 basado en el SoC Nordic nRF52480, que integra un núcleo Arm Cortex-M4F a 64 MHz con unidad de punto flotante (FPU), 1 MB de Flash y 256 KB de RAM. Además incluye un subsistema de seguridad CryptoCell CC310 que aporta aceleración de algoritmos criptográficos y almacenamiento seguro de claves en hardware.

2.1.2. Periféricos y sensores integrados

Radio y comunicaciones

- Bluetooth 5 multiprotocolo (2 Mbps, Advertising Extensions, Long Range, +8 dBm TX, –95 dBm RX).
- IEEE 802.15.4 (Thread, Zigbee).
- USB Full-speed 12 Mbps (con EasyDMA).
- NFC-A tag.

Interfaces digitales

- QSPI/SPI/TWI (I^2C)/I^2S/PDM/QDEC con EasyDMA.
- SPI de alta velocidad a 32 MHz.
- ADC de 12 bits a 200 ksps.

Sensores de entorno

- IMU LSM9DS1 (9 ejes): acelerómetro (±2/4/8/16 g), giroscopio (±245/500/2000 dps) y magnetómetro (±4/8/12/16 gauss), salida 16 bits.
- Barómetro/temperatura LPS22HB: rango 260–1260 hPa, 24 bits, temp. integrada, 1–75 Hz, interrupciones de dato y FIFO.
- Humedad/temperatura HTS221: 0–100 % rH, ±3.5 % rH, ±0.5 °C, salida 16 bits.
- Proximidad/luz/RGB/gestos APDS-9960: detección de proximidad con rechazo de luz ambiente, color con filtros UV/IR, detección de gestos UP/DOWN/LEFT/RIGHT, FIFO, I^2C.

- Micrófono digital MP34DT05: omnidireccional, SNR 64 dB, AOP 122.5 dB SPL, salida PDM.
- Chip criptográfico ATECC608A: almacenamiento seguro de hasta 16 claves/certificados, co-procesador AES-128, SHA-256, HMAC, ECDH P-256.

2.1.3. Sensor de Movimiento: Giroscopio Integrado en el Arduino Nano 33 BLE Sense

El periférico más importante en este laboratorio es el giroscopio integrado en el módulo LSM9DS1 de la placa Arduino Nano 33 BLE Sense, ya que permite capturar la información necesaria para el reconocimiento de actividad humana (HAR). Este sensor es capaz de medir la velocidad angular en los tres ejes espaciales (X, Y, Z), lo que lo convierte en un componente esencial para detectar y clasificar movimientos físicos del usuario [1].

El LSM9DS1 es un sensor inercial de nueve grados de libertad (9-DoF), que incluye acelerómetro, giroscopio y magnetómetro. En este laboratorio, se hace uso específico del giroscopio, el cual proporciona datos de rotación angular en radianes por segundo. El microcontrolador del Arduino se comunica con este sensor a través del bus I^2C, y mediante el uso de la biblioteca $Arduino\_LSM9DS1$, es posible configurar y obtener lecturas en tiempo real para alimentar tanto el registro de datos como la inferencia del modelo entrenado [1].

2.1.4. Características eléctricas

Alimentación

- Entrada USB o VIN (a través de VUSB y un regulador DC-DC MPM3610).
- Nivel lógico 3.3 V (NO tolera 5 V en I/O).

Regulador MPM3610

- Entrada hasta 21 V, eficiencia $\geq 65\,\%$ al mínimo carga, $\geq 85\,\%$ a 12 V

Consumo

- Modo activo (busy loop): especificación pendiente (TBC).
- Bajo consumo (LP): TBC.

Rango de operación

- Temperatura: –40 °C a +85 °C.

Pinout de alimentación

- +3V3 (salida regulada) y VIN (entrada) en ambos conectores de 15 pines.
- GND distribuido en varias posiciones.

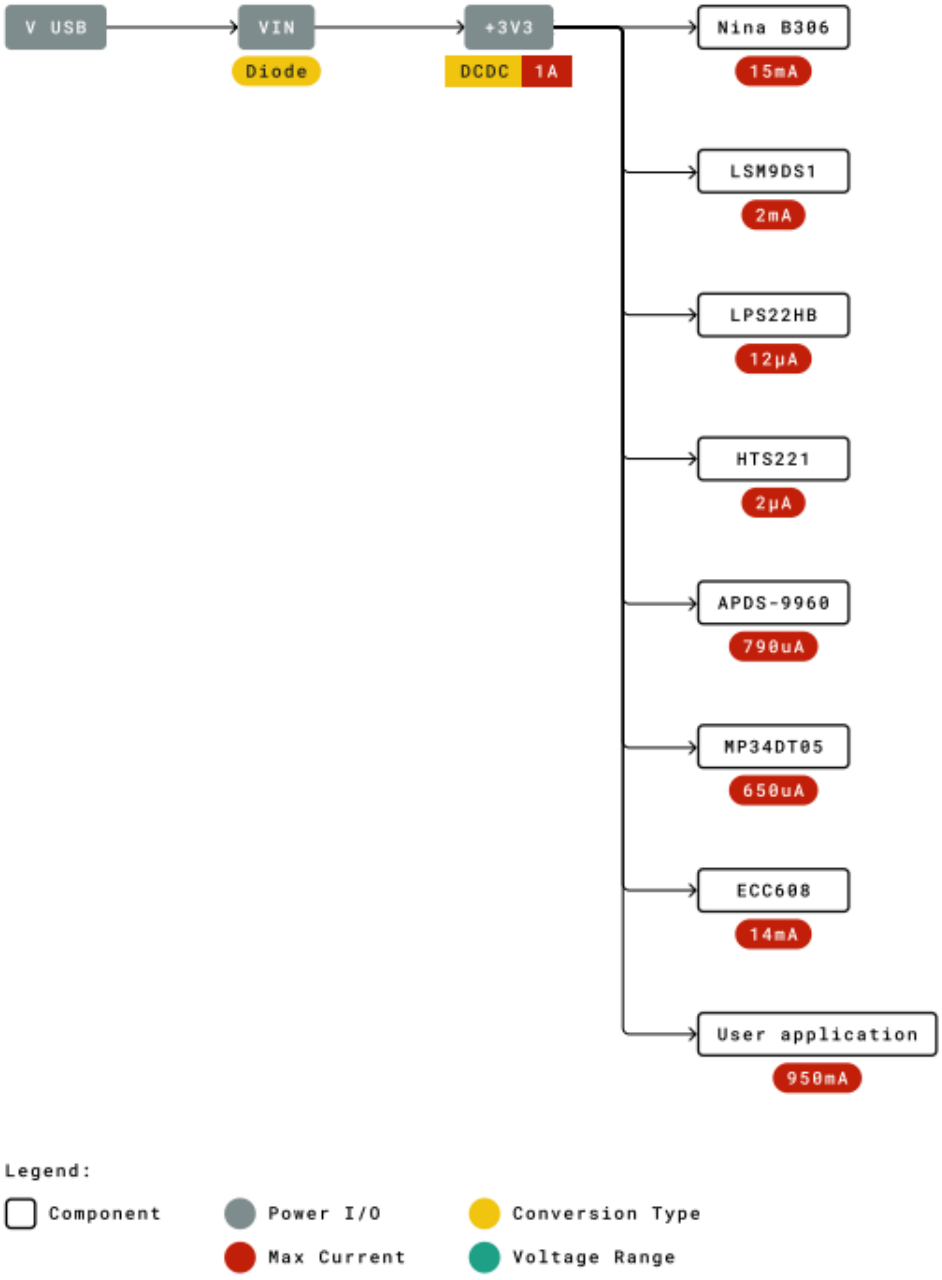

Figura 1: Diagrama de Arbol de Alimentación del Microcontrolador.

2.1.5. Diagrama del Microcontrolador

A continuación, se muestra el diagrama de bloques del microcontrolador

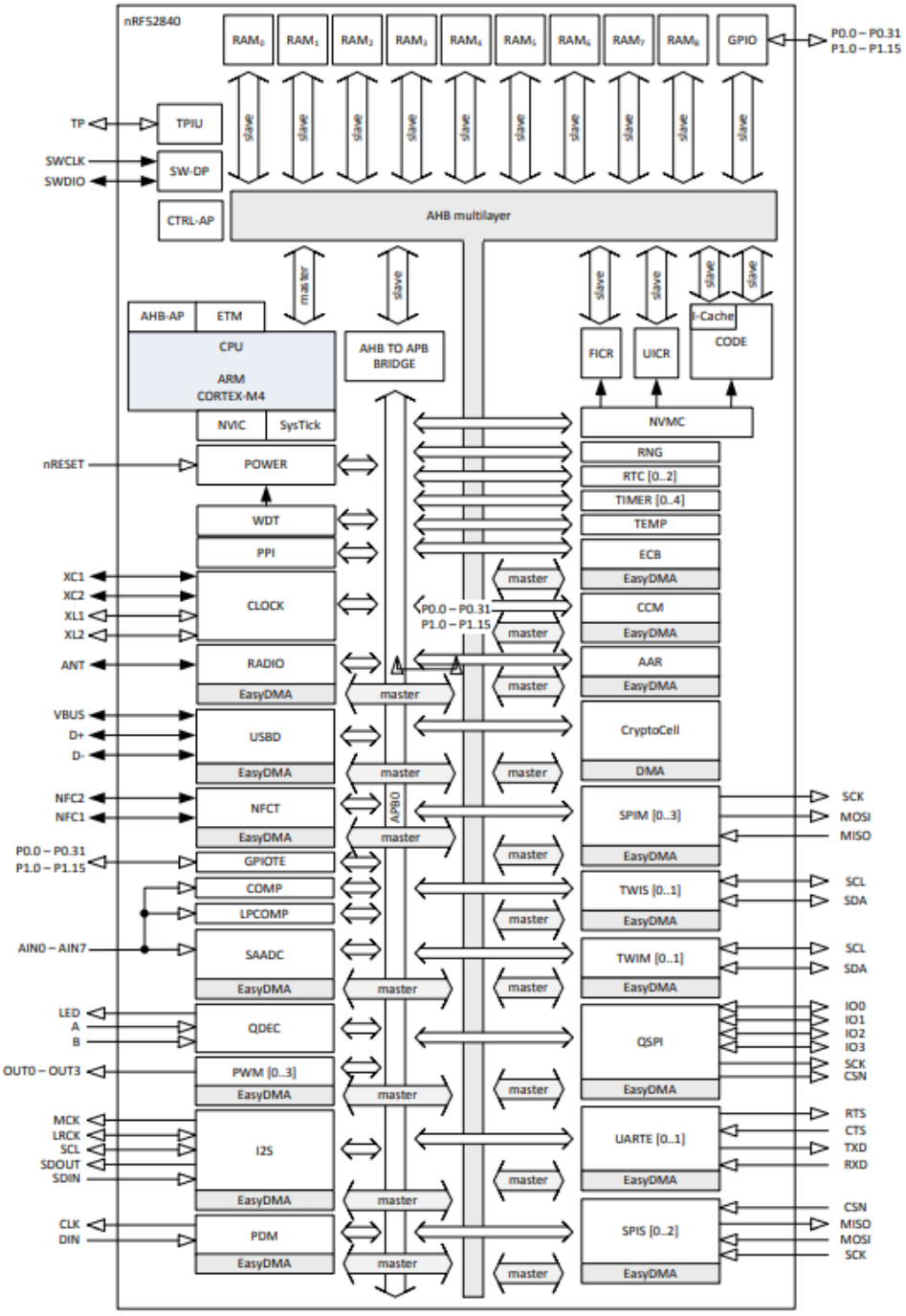

Figura 2: Diagrama de Bloques del Microcontrolador

2.1.6. Diagrama de pines

A continuación, se muestra el diagrama de pines del Arduino Nano 33 BLE Sense:

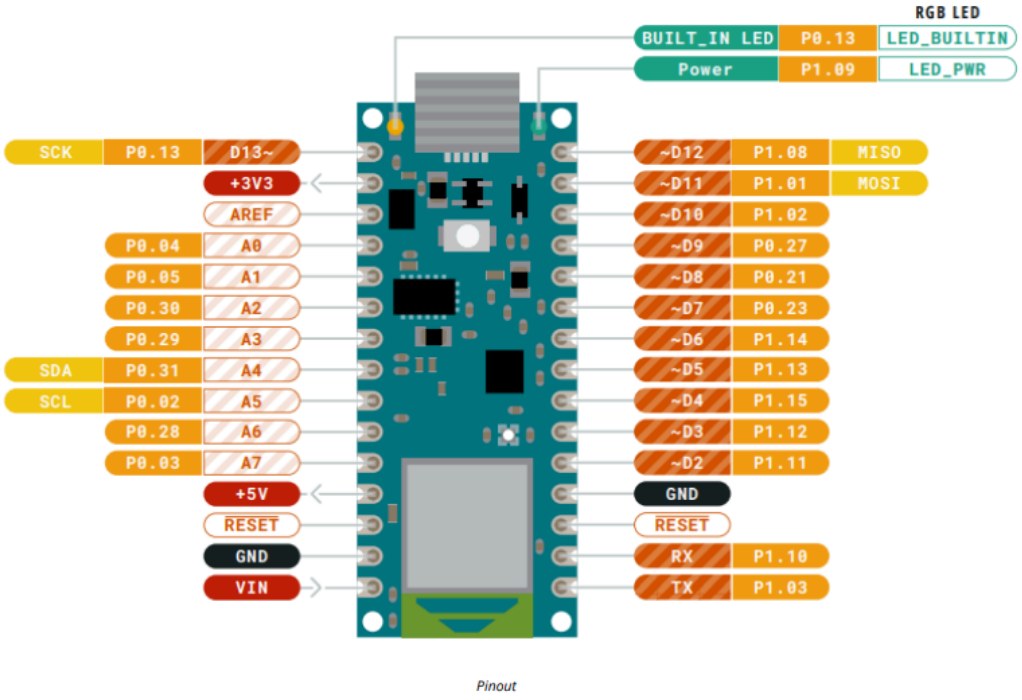

Figura 3: Diagrama de Pines

Ahora bien, en las siguientes tablas se especifican las funcionalidades de los pines:

Pin	Function	Type	Description
1	VUSB	Power	Power Supply Input. If board is powered via VUSB from header this is an Output **(1)**
2	D-	Differential	USB differential data -
3	D+	Differential	USB differential data +
4	ID	Analog	Selects Host/Device functionality
5	GND	Power	Power Ground

Figura 4: Tabla Especificaciones Pines USB

Pin	Function	Type	Description
1	+3V3	Power Out	Internally generated power output to be used as voltage reference
2	SWD	Digital	nRF52480 Single Wire Debug Data
3	SWCLK	Digital In	nRF52480 Single Wire Debug Clock
5	GND	Power	Power Ground
6	RST	Digital In	Active low reset input

Figura 5: Tabla Especificaciones Pines Debug

Pin	Function	Type	Description
1	D13	Digital	GPIO
2	+3V3	Power Out	Internally generated power output to external devices
3	AREF	Analog	Analog Reference; can be used as GPIO
4	A0/DAC0	Analog	ADC in/DAC out; can be used as GPIO
5	A1	Analog	ADC in; can be used as GPIO
6	A2	Analog	ADC in; can be used as GPIO
7	A3	Analog	ADC in; can be used as GPIO
8	A4/SDA	Analog	ADC in; I2C SDA; Can be used as GPIO **(1)**
9	A5/SCL	Analog	ADC in; I2C SCL; Can be used as GPIO **(1)**
10	A6	Analog	ADC in; can be used as GPIO
11	A7	Analog	ADC in; can be used as GPIO
12	VUSB	Power In/Out	Normally NC; can be connected to VUSB pin of the USB connector by shorting a jumper
13	RST	Digital In	Active low reset input (duplicate of pin 18)
14	GND	Power	Power Ground
15	VIN	Power In	Vin Power input
16	TX	Digital	USART TX; can be used as GPIO
17	RX	Digital	USART RX; can be used as GPIO
18	RST	Digital	Active low reset input (duplicate of pin 13)
19	GND	Power	Power Ground
20	D2	Digital	GPIO
21	D3/PWM	Digital	GPIO; can be used as PWM
22	D4	Digital	GPIO
23	D5/PWM	Digital	GPIO; can be used as PWM
24	D6/PWM	Digital	GPIO, can be used as PWM
25	D7	Digital	GPIO
26	D8	Digital	GPIO
27	D9/PWM	Digital	GPIO; can be used as PWM
28	D10/PWM	Digital	GPIO; can be used as PWM
29	D11/MOSI	Digital	SPI MOSI; can be used as GPIO
30	D12/MISO	Digital	SPI MISO; can be used as GPIO

Figura 6: Tabla Especificaciones Pines Headers

2.2. TensorFlow y TensorFlow Lite

2.2.1. TensorFlow

TensorFlow es una biblioteca de software de código abierto desarrollada por Google que permite realizar operaciones de computación numérica intensiva mediante grafos de flujo de datos. Su arquitectura está optimizada para el desarrollo y entrenamiento de modelos de aprendizaje automático, particularmente redes neuronales profundas. TensorFlow ofrece herramientas de alto nivel para definir modelos, gestionar conjuntos de datos, realizar operaciones matemáticas sobre tensores, y ejecutar procesos de entrenamiento y validación. Gracias a su flexibilidad, se puede ejecutar sobre diferentes tipos de hardware, como CPU, GPU, TPU, e incluso dispositivos distribuidos en la nube [2].

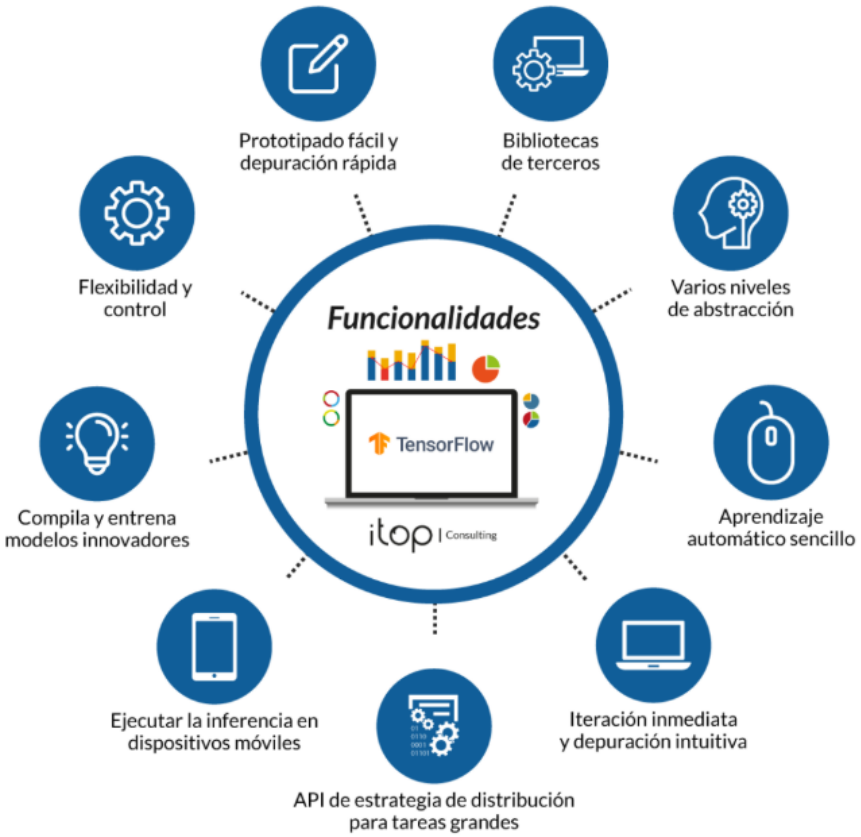

Figura 7: TensorFlow

2.2.2. TensorFlow Lite

TensorFlow Lite (TFLite) es una versión ligera de TensorFlow diseñada para permitir la ejecución de modelos previamente entrenados en dispositivos con recursos computacionales limitados, como microcontroladores, teléfonos móviles o sistemas embebidos. TFLite emplea un formato de modelo optimizado que reduce el tamaño del archivo y mejora la velocidad de inferencia. Para ello, aplica técnicas como la cuantización de pesos, eliminación de operaciones innecesarias, y reestructuración del grafo computacional. Esto lo convierte en una solución ideal para aplicaciones de inferencia en tiempo real que deben ejecutarse localmente, sin dependencia de servidores externos [3].

Adicionalmente, el ecosistema de TensorFlow Lite incluye herramientas para la conversión de modelos desde TensorFlow estándar, APIs específicas para diversas plataformas (como Android, iOS, y microcontroladores con soporte de C++ o Arduino), y documentación para facilitar su integración en proyectos de inteligencia artificial embebida.

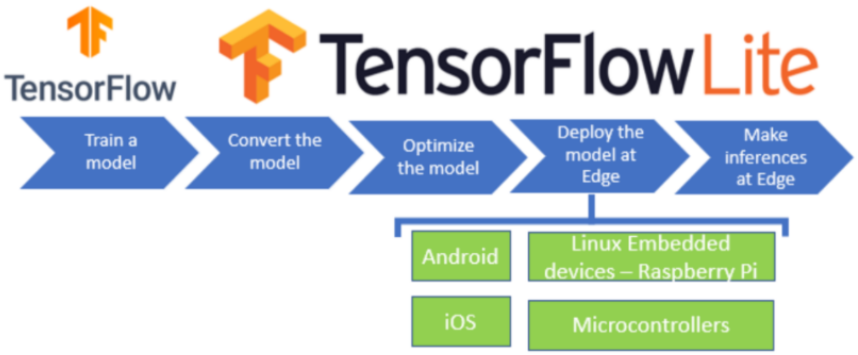

Figura 8: TensorFlow Lite

2.3. Tiny Machine Learning

Tiny Machine Learning (TinyML) es una rama del aprendizaje automático enfocada en la implementación de modelos de inteligencia artificial en dispositivos de muy bajos recursos computacionales, como microcontroladores, sensores inteligentes o sistemas embebidos. Su objetivo principal es llevar la capacidad de inferencia de modelos entrenados como redes neuronales directamente al borde (edge) del sistema, donde se recolectan los datos, eliminando la necesidad de enviar información a servidores externos para su procesamiento [4].

Este paradigma es posible gracias a la combinación de técnicas como la cuantización, el pruning (poda de pesos irrelevantes) y el uso de bibliotecas especializadas como TensorFlow Lite for Microcontrollers, que permiten ejecutar modelos de tamaño reducido utilizando apenas unos pocos kilobytes de memoria y ciclos de CPU. TinyML abre la puerta al desarrollo de aplicaciones autónomas, de bajo consumo energético y con alta capacidad de respuesta en tiempo real, lo que lo hace ideal para sectores como el Internet de las Cosas (IoT), la robótica, la medicina portátil, y la agricultura inteligente [5].

Uno de los principales desafíos de TinyML es lograr un equilibrio entre la precisión del modelo y las limitaciones de memoria y procesamiento del hardware destino. Por esto, el diseño y entrenamiento del modelo se realiza generalmente en entornos con alta capacidad de cómputo, y luego se exporta una versión optimizada lista para ser desplegada en el dispositivo embebido.

2.4. Reconocimiento de Actividad Humana

El Reconocimiento de Actividad Humana (Human Activity Recognition, HAR) es un área del aprendizaje automático que se enfoca en identificar y clasificar acciones o movimientos realizados por personas a partir de datos sensoriales. Estos datos suelen provenir de sensores inerciales como acelerómetros, giroscopios o magnetómetros, que están integrados en dispositivos móviles, wearables o microcontroladores [6].

El proceso de HAR puede dividirse en varias etapas: adquisición de datos, preprocesamiento, extracción de características, entrenamiento de un modelo de clasificación y ejecución de inferencias. En la actualidad, es común el uso de redes neuronales para aprender directamente patrones en los datos crudos, lo que permite una mayor precisión

en tareas como detectar movimientos del brazo, reconocer gestos, contar pasos o incluso identificar actividades complejas como correr, caminar o estar en reposo [7].

HAR tiene aplicaciones en áreas como salud, seguridad, domótica, deportes, y tecnologías asistivas. En el contexto de sistemas embebidos, se combina frecuentemente con TinyML para permitir el reconocimiento en tiempo real directamente en el dispositivo, sin depender de conectividad externa ni procesamiento en la nube, optimizando así el consumo energético y la latencia [8].

2.5. Google Colaboratory

Google Colaboratory, comúnmente conocido como Google Colab, es una plataforma gratuita desarrollada por Google que permite ejecutar código Python en un entorno basado en la nube, directamente desde el navegador. Está diseñada principalmente para tareas de análisis de datos, aprendizaje automático, visualización y computación científica. Uno de sus componentes más importantes es el soporte nativo para cuadernos de Jupyter, los cuales combinan código ejecutable, visualizaciones y texto descriptivo en un mismo documento interactivo [9].

Google Colab proporciona acceso gratuito a recursos computacionales como CPUs, GPUs e incluso TPUs, lo que lo convierte en una herramienta accesible y poderosa para entrenar modelos de machine learning sin requerir hardware especializado por parte del usuario. Además, permite integrar fácilmente bibliotecas populares como TensorFlow, NumPy, Pandas, Matplotlib, entre muchas otras.

Otra ventaja relevante es su integración con Google Drive, lo que facilita el almacenamiento y compartición de proyectos.

2.6. Lista de componentes y precios

Tabla 1: Lista de componentes y precios estimados en Costa Rica

Componente	**Descripción**	**Precio unitario (CRC)**
Arduino Nano 33 BLE Sense Lite	Arduino Tiny Machine Learning Kit	₡34 800

2.7. Flujo de Trabajo implementado

La siguiente figura muestra el flujo de trabajo seguido para la implementación del sistema de reconocimiento de actividad humana. Este proceso se dividió en cuatro etapas principales: captura de datos, almacenamiento, entrenamiento del modelo y pruebas de inferencia.

Inicialmente, se activó el sensor giroscopio mediante un sketch cargado en el Arduino Nano 33 BLE Sense. Luego, los datos recogidos fueron enviados por puerto serial a una computadora, donde se almacenaron en archivos CSV utilizando un script de Python. Posteriormente, los datos fueron utilizados para entrenar una red neuronal en un cuaderno de Google Colab, empleando la biblioteca TensorFlow. Finalmente, el modelo entrenado fue convertido a formato (.h) e integrado en un nuevo programa de Arduino, el cual fue cargado en la placa para realizar inferencias en tiempo real sobre los movimientos ejecutados por el usuario.

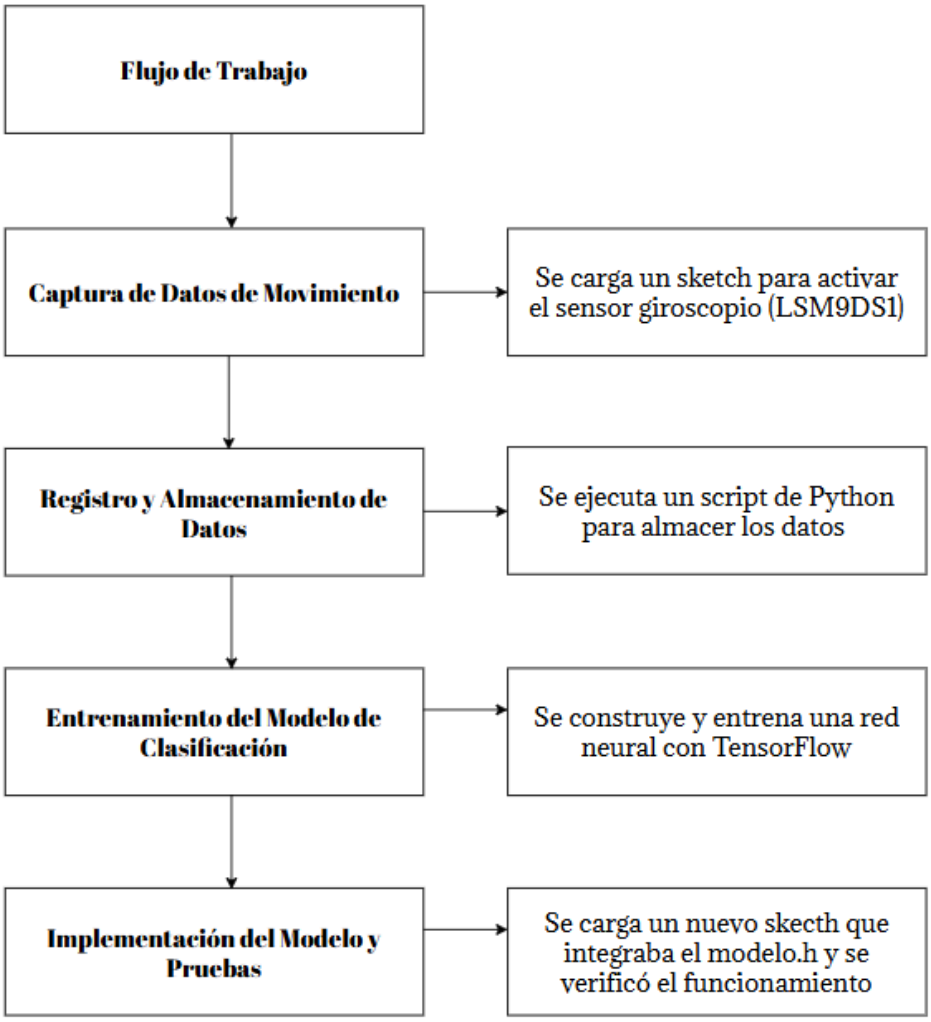

Figura 9: Flujo de trabajo implementado

3. Desarrollo/análisis

En esta sección se describe detalladamente el proceso de desarrollo que se llevó a cabo durante la realización del laboratorio, así como el análisis de los resultados obtenidos. Se presenta el flujo de trabajo implementado, desde la captura de datos hasta la ejecución del modelo de clasificación de gestos en el microcontrolador. Además, se analizan los programas desarrollados, la lógica utilizada, el rendimiento del sistema y la precisión alcanzada en las pruebas experimentales.

3.1. Captura y Almacenamiento de Datos

3.1.1. Configuración y lectura del sensor (Arduino)

Para la captura de datos del movimiento se utilizó el sensor inercial **LSM9DS1** integrado en la placa *Arduino Nano 33 BLE Sense*. La inicialización del sensor se realizó mediante la biblioteca `Arduino_LSM9DS1`, la cual permite acceder directamente a los datos del acelerómetro y giroscopio de manera sencilla y eficiente.

En la función `setup()`, se configura el puerto serial a 9600 baudios para establecer la comunicación con la computadora. Además, se inicializa el sensor IMU mediante el método `IMU.begin()`. En caso de error durante la inicialización, se muestra un mensaje por el monitor serial y se detiene la ejecución del programa mediante un ciclo infinito.

El bloque principal en `loop()` está dividido en dos fases:

- **Detección de movimiento:** se monitorea continuamente la aceleración total ($|a_x| + |a_y| + |a_z|$) y se compara contra un umbral definido (2.5 unidades). Cuando se detecta un movimiento significativo, se inicia la captura de datos.

- **Lectura y transmisión de datos:** una vez activado el proceso, se capturan 119 muestras consecutivas que incluyen lecturas de aceleración y velocidad angular en los tres ejes (x, y, z). Estas lecturas se envían por el puerto serial en formato `CSV`, con seis columnas por línea: `aX, aY, aZ, gX, gY, gZ`.

Este mecanismo asegura que las muestras correspondan a una ventana temporal específica asociada a un gesto, y permite que la computadora las reciba mediante un script de Python para su posterior etiquetado y almacenamiento.

3.1.2. Recepción y almacenamiento de datos (Python)

Una vez que el Arduino comienza a transmitir datos del sensor inercial, estos deben ser recibidos, etiquetados y almacenados en la computadora. Para esto, se desarrolló un script en `Python` que se comunica con la placa a través del puerto serial y guarda las lecturas en un archivo `CSV`.

El script comienza cargando las bibliotecas necesarias, como `serial` para la comunicación, `csv` para la escritura del archivo y `os/sys` para manejo de argumentos y rutas. Al ejecutarlo, el usuario debe proporcionar como argumento el nombre del archivo de salida, lo que permite etiquetar fácilmente el gesto que se está registrando.

Se establece la conexión serial con el puerto correspondiente (`COM10`) a 9600 baudios y se espera un breve período de tiempo para que el Arduino se reinicie correctamente. Una vez establecida la conexión, se inicia el proceso de captura de datos.

Durante la ejecución, el script lee continuamente las líneas enviadas por el Arduino, las decodifica y verifica que tengan el formato esperado (seis valores separados por comas). Si la línea es válida, se escribe en el archivo CSV, el cual se almacena en una subcarpeta llamada `data`. El proceso se repite hasta alcanzar una cantidad fija de muestras (3000 en este caso), tras lo cual se cierra el archivo y se finaliza la sesión serial.

3.2. Inferencia y Clasificación en Tiempo Real

3.2.1. Integración del modelo en Arduino por medio de entrenamiento en Google Colab

Librerías y carga de datos:

Como primera parte del modelo se realizó la instalación de las diferentes librerías necesarias para el entrenamiento del modelo. Entre ellas encontramos:

- xxd: convserión a un .h.
- pandas numpy matplotlib: modificación y graficación de los dstos.
- tensorflow: entrenamienro del modelo.

A su vez, se cargaron los 3 .csv correspondientes a cada uno de los movimientos que identificará el arduino.

Acomodo de datos:

Antes de realizar el entrenamiento, debemos separar los datos en subgrupos de entrenamiento. En este casi cada subgrupo corresponderá a 150 datos. Aunado a ello, los datos son mezclados de manera aleatoria para ser separados en conjuntos de entrenamiento, validación y testeo.

Entrenamiento del modelo:

Como fue especificado por el profesor, se utilizó ReLU como función de activación, algoritmo de optimización rmsprop y métrica de pérdida mae.

Aunado a ello, se decidió utilizar 2 capas para el entrenamiento, siendo una de 50 neuronas y otra de 15.

Testeo de resultados:

Teniendo el modelo entrenado, se procede con la verificación del modelo. Para ello, se testea utilizando el 20 % para dichas pruebas. Además, se grafican las pérdidas y el MAE para visualizar de mejor manera los resultados.

Conversión del modelo:

Con del modelo verificado lo convertimos a un archivo del tipo .tflite. Este posteriormente con ayuda de xxd, lo convertimos a un .h, el cual, ya puede ser utilizado en Arduino IDE.

3.2.2. Procesamiento de datos e inferencia en tiempo real

Para ejecutar la inferencia directamente en el microcontrolador, se desarrolló un programa en Arduino que integra el modelo entrenado (.h) y realiza predicciones en tiempo real a partir de los datos capturados por el sensor inercial. El programa se estructura en tres fases principales: inicialización del modelo, captura y preprocesamiento de datos, e inferencia con visualización de resultados. 3mm

En la primera parte, se incluye el archivo model.h, que contiene el modelo exportado en formato de arreglo binario. Este modelo es cargado mediante la función tflite::GetModel(). Para ejecutarlo, se crea un intérprete de TensorFlow Lite Micro, que requiere un bloque de memoria para almacenar los tensores de entrada, salida y temporales. Se utiliza un resolver de operaciones que habilita todas las funciones necesarias para correr el modelo.

En la segunda parte, el sistema permanece a la espera de un movimiento significativo, detectado mediante la suma absoluta de las aceleraciones en los tres ejes. Al detectar este umbral, se capturan 119 muestras de aceleración y velocidad angular, las cuales son normalizadas y escritas directamente en el tensor de entrada. Esta normalización garantiza que los valores se mantengan dentro del rango de entrenamiento del modelo, lo cual es clave para obtener inferencias confiables.

Finalmente, al completar la ventana de captura, se invoca el modelo mediante Invoke(). El tensor de salida contiene una probabilidad por clase, y estas probabilidades se imprimen en el monitor serial. Las etiquetas correspondientes se definen en el arreglo GESTURES[], que coincide con el orden utilizado durante el entrenamiento del modelo.

3.3. Análisis de Resultados

Para verificar el funcionamiento del sistema de clasificación en tiempo real, se realizaron pruebas ejecutando los tres gestos definidos en el modelo: *circles*, *clap* y *updown*. A continuación, se presentan algunos ejemplos representativos de las salidas generadas por el microcontrolador, donde se puede observar la probabilidad asignada por el modelo a cada clase:

```
20:26:01.485 ->
20:26:02.516 -> circles: 0.966276
20:26:02.516 -> clap: 0.010870
20:26:02.553 -> updown: 0.022854
20:26:02.553 ->
```

Figura 10: Resultado para circles

```
20:26:02.553 ->
20:26:03.578 -> circles: 0.023571
20:26:03.578 -> clap: 0.973821
20:26:03.578 -> updown: 0.002608
20:26:03.578 ->
```

Figura 11: Resultado para clap

```
20:28:04.388 ->
20:28:05.846 -> circles: 0.072066
20:28:05.846 -> clap: 0.000783
20:28:05.846 -> updown: 0.927150
20:28:05.846 ->
```

Figura 12: Resultado para updown

Gesto Realizado	circles	clap	updown	Predicción
circles	0.966	0.010	0.023	circles
clap	0.024	0.974	0.003	clap
updown	0.072	0.001	0.927	updown

Tabla 2: Resultados de inferencia para cada gesto probado

Los resultados muestran que el modelo fue capaz de asignar una probabilidad significativamente mayor a la clase correspondiente al gesto ejecutado, demostrando un buen rendimiento en condiciones reales.

Cabe destacar que, aunque los resultados obtenidos fueron correctos en las pruebas mostradas, el desempeño del modelo puede verse afectado por variaciones en la ejecución del gesto, condiciones del entorno o interferencias en los sensores. Por ejemplo, gestos incompletos, movimientos lentos o interrupciones durante la captura podrían reducir la precisión. Además, la cantidad de muestras utilizadas para el entrenamiento y la ventana de captura influyen directamente en la capacidad del modelo para generalizar. A pesar de estas limitaciones inherentes al entorno, el sistema logró clasificar correctamente los gestos, lo que evidencia una adecuada configuración del flujo de trabajo y del entrenamiento.

4. Git y enlace a video de demostración de funcionamiento

El desarrollo completo del laboratorio, que incluye el código fuente, las simulaciones y la documentación del laboratorio, se encuentra disponible en el repositorio oficial del curso en Git UCR, en la carpeta correspondiente al Laboratorio #5 `https://git.ucr.ac.cr/laboratorio_microcontroladores_i2025/laboratorio_5.git`.

Además, en el siguiente enlace se encuentra un video que comprueba el funcionamiento del sistena de reconocimienro de actividad humana `https://youtu.be/0LZBiBfXoxE?si=f1Ea5ouhFWgsJ0nN`.

5. Conclusiones y recomendaciones

5.1. Conclusiones

- Por la parte de la obtención de los datos a un archivo .csv, se logró de manera eficaz, tomando cada uno de los distintos movimientos con una cantidad igualitaria para cada archivo.
- Se logró entrenar un modelo capaz de identificar cada uno de los movimientos con una precisión mayor al 90 %. Esto mientras se cumplían las especificaciones dadas por el profesor sobre el modelo a utilizar.
- El Arduino fue capaz de cumplir con la obtención e identificación de un conjunto de datos y determinar el movimiento que representan. Esto a una velocidad alta y sin errores en el espacio para el ML.

5.2. Recomendaciones

- Mayor toma de datos: un punto que mejoraría el modelo es utilizar una mayor cantidad de datos de entrenamiento. De esta manera, es posible disminuir la cantidad de neuronas necesarias por el modelo y seguir manteniendo una gran precisión.

- Movimientos variados: para aumentar las capacidades del arduino, es recomendable aumentar la cantidad de movimientos identificados. Aunque esto podría aumentar la memoria del modelo, si lo unimos con la recomendación, se podría seguir manteniendo la misma cantidad de neuronas sin la necesidad de utilizar más espacio.
- Mayor cantidad de pruebas del modelo: aunque se realizaron bastantes gráficas para verificar el buen funcionamiento del modelo, es comendable aumentar la cantidad de verificaciones. Por ejemplo, realizar testeos donde se observen los movimientos calificados de manera errónea y por cuál movimiento se confundió, podría ayudar a determinar qué puntos pueden fortalecerse en el modelo.
- Recolección de datos mejor administrada: a la hora de adquirir los .csv con los datos, se podría intentar guardar los datos inmediatamente en el formato necesario para el entrenamiento y separado en los 3 grupos (testeo, validación y entrenamiento). De esta manera se mejora la eficacia a la hora de crear el modelo y se eliminaría un paso intermedio en el proceso del ML.

Referencias

[1] STMicroelectronics, "LSM9DS1: iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer," *Datasheet*, rev. 2, July 2015. Disponible en: `https://www.st.com/resource/en/datasheet/lsm9ds1.pdf`

[2] Martín Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," Google Research, 2015. Disponible en: `https://www.tensorflow.org/`

[3] TensorFlow Lite, "TensorFlow Lite Guide," Google Developers, 2024. Disponible en: `https://www.tensorflow.org/lite/guide`

[4] Pete Warden and Daniel Situnayake, "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers," O'Reilly Media, 1st edition, 2019.

[5] J. Redmon et al., "A Survey on Tiny Machine Learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10841–10860, 2022. Disponible en: `https://ieeexplore.ieee.org/document/9705486`

[6] O.D. Lara and M.A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[7] C.A. Ronao and S.B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016.

[8] D. Singh et al., "Human Activity Recognition Using Deep Learning: A Review," *IEEE Access*, vol. 9, pp. 103455–103469, 2021. Disponible en: `https://ieeexplore.ieee.org/document/9474166`

[9] Google Research, "Welcome to Colaboratory," Google Colab Documentation, 2024. Disponible en: `https://colab.research.google.com/notebooks/intro.ipynb`

[10] Nordic Semiconductor, *nRF52840 Product Specification v1.1*, Rev. 1.1, Mar. 2021. Disponible en: `https://www.mouser.co.cr/datasheet/2/297/nRF52840_PS_v1.1-1623672.pdf`

[11] Arduino S.r.l., *Arduino Nano 33 BLE Sense Datasheet*, Rev. 3, 25 de abril de 2024. Disponible en: `https://store.arduino.cc/products/arduino-nano-33-ble-sense`