

Universidad de Costa Rica

Laboratorio de Microcontroladores

Proyecto final:
Estación de Monitoreo Inteligente Multisensorial de
Seguridad Basado en Machine Learning

Prof. MSc. Marco Villalta Fallas

Marco Vásquez Ovares - B17032

Junior Ruiz Sánchez - B97026

Grupo: 01

Sub-grupo: 13

I Ciclo 2025

Índice

1. Introducción	5
2. Justificación	6
3. Objetivos	7
3.1. Objetivo General	7
3.2. Objetivos Específicos	7
4. Alcances	8
5. Nota teórica	9
5.1. Arduino	9
5.2. Arduino Tiny Machine Learning Kit	9
5.3. Arduino Nano 33 BLE Sense Lite	9
5.4. Micrófono MP34DT05	11
5.4.1. Diagrama de pines	12
5.4.2. Descripción de pines	12
5.4.3. Características eléctricas	13
5.5. Microcontrolador nRF52840	14
5.5.1. Diagrama de pines	14
5.5.2. Descripción de pines	15
5.5.3. Diagrama de bloques	18
5.5.4. Características eléctricas	19
5.6. TensorFlow Lite	20
5.7. Edge Impulse	20
5.8. Machine Learning (ML)	21
5.9. Diseño del circuito	21
5.9.1. Lista de componentes	22
5.9.2. Información adicional	22
6. Desarrollo	23
6.1. Análisis del firmware	23
6.1.1. Firmware	23
6.2. Análisis electrónico	24
6.3. Análisis de resultados	25
6.3.1. Captura de datos con Arduino y Python	25
6.3.2. Entrenamiento del modelo en Edge Impulse	26
6.3.3. Data Acquisition (Adquisición de datos)	27
6.3.4. Create Impulse	28
6.3.5. MFCC	29
6.3.6. Classifier	30
6.3.7. Model Testing	32
6.3.8. Implementación del modelo en el microcontrolador	33
6.3.9. Validación funcional	34

7. Conclusiones y Recomendaciones	35
7.1. Conclusiones	35
7.2. Recomendaciones	35
Bibliografía	36
8. Apéndices	36
8.1. Repositorio Git	36

Índice de figuras

1.	Arduino® Nano 33 BLE Sense [3]	10
2.	Microfono MP34DT05 [6]	11
3.	Connector Pinouts [3]	12
4.	Descripción de pines [3]	12
5.	Power Consumption [3]	13
6.	Power Tree [3]	13
7.	Diagrama de pines. [4]	14
8.	Diagrama de pines pt1. [4]	15
9.	Diagrama de pines pt2. [4]	16
10.	Diagrama de pines pt3. [4]	17
11.	Diagrama de bloques. [4]	18
12.	Absolute maximum ratings [4]	19
13.	Arduino Tiny Machine Learning Kit [2]	21
14.	Circuito final	24
15.	Captura de datos pt1	25
16.	Captura de datos pt2	26
17.	4 Clases definidas	27
18.	Audios .WAV de 4 segundos a 16 kHz pt1	27
19.	Audios .WAV de 4 segundos a 16 kHz pt2	28
20.	Impulso	28
21.	MFCC	29
22.	Neural network architecture	30
23.	Last training performance	30
24.	Confusion matrix	31
25.	Data explorer	31
26.	Precisión (Accuracy)	32
27.	Resultados del model testing	33
28.	build	34
29.	Resultados	34

Índice de tablas

1.	Lista de componentes	22
----	----------------------	----

1. Introducción

Resumen

El presente proyecto consiste en el desarrollo de una **estación de monitoreo inteligente basada en un sistema embebido** capaz de detectar y clasificar distintos tipos de sonidos en tiempo real, como parte de una solución de seguridad preventiva. Para lograrlo, se utilizó la placa **Arduino Nano 33 BLE Sense Lite**, la cual incorpora un micrófono digital que permite capturar el audio del entorno.

El sistema fue diseñado para reconocer cuatro clases de sonidos relevantes para aplicaciones de seguridad: **ladrido de perro, vidrio rompiéndose, sirena de policía y sonido ambiente normal**. La recolección de datos se realizó utilizando un script en Python que se comunicaba con el microcontrolador vía puerto serial, permitiendo almacenar las grabaciones en formato **.wav** y **.csv** para su posterior procesamiento.

El entrenamiento del modelo se llevó a cabo en la plataforma Edge Impulse, utilizando técnicas de extracción de características mediante **MFCC (Mel Frequency Cepstral Coefficients)** y clasificadores optimizados para dispositivos con recursos limitados. Una vez entrenado y validado, el modelo fue exportado en formato de biblioteca C++ (**.zip**) y cargado nuevamente en el Arduino, el cual ejecuta inferencias localmente para determinar en tiempo real el tipo de sonido detectado.

2. Justificación

La protección de personas y bienes exige un monitoreo que escuche el entorno. Las cámaras y los sensores de movimiento ofrecen datos visuales, pero suelen ignorar eventos sonoros críticos, como cristales rompiéndose o alarmas lejanas. Al integrar la detección de sonidos anómalos, exploramos nuevas capacidades para identificar y responder con rapidez ante riesgos auditivos.

3. Objetivos

3.1. Objetivo General

Clasificar los sonidos del entorno y activar respuestas locales de manera automática, garantizando una reacción inmediata ante eventos relevantes.

3.2. Objetivos Específicos

- Desarrollar el firmware para la captura y el preprocesamiento de audio en la placa Arduino Nano 33 BLE Sense Lite.
- Entrenar y desplegar un modelo TinyML que distinga vidrios rotos, ladridos, sirenas y ruido ambiental con alta fiabilidad.

4. Alcances

Este sistema se enfoca en la detección y clasificación en tiempo real de cuatro clases de sonido críticas para la seguridad: vidrios rotos, ladridos de perro, ambiente normal y sirenas de policía. El procesamiento de señal y la inferencia TinyML se ejecutan íntegramente en la placa Arduino Nano 33 BLE Sense Lite, sin depender de conexiones de red ni servicios en la nube.

Quedan fuera del alcance de este proyecto:

- Integración de sensores adicionales, como cámaras o detectores de presencia.
- Almacenamiento o análisis de datos en plataformas remotas.
- Clasificación de eventos acústicos distintos a las cuatro categorías definidas.

5. Nota teórica

5.1. Arduino

Arduino constituye una plataforma de hardware y software de código abierto, pensada para simplificar la creación de prototipos electrónicos y sistemas embebidos. Sus placas programables, dotadas de microcontroladores, se combinan con un entorno de desarrollo integrado (*IDE*) multiplataforma que permite compilar y cargar firmware en cuestión de segundos [1].

Gracias a su diseño modular, es posible conectar sensores, actuadores y módulos de comunicación sin configuraciones complejas. Por ello, resulta habitual emplearla en proyectos de automatización, robótica e Internet de las Cosas (*IoT*). Su comunidad global comparte librerías, ejemplos y documentación que aceleran el aprendizaje y fomentan la innovación.

El lenguaje de programación de Arduino se basa en Wiring y C/C++, lo que asegura compatibilidad con bibliotecas estándar y favorece un ciclo de desarrollo ágil. Además, la naturaleza de “hardware abierto” de sus placas facilita su adaptación y expansión, manteniendo bajos los costos para estudiantes e investigadores.

5.2. Arduino Tiny Machine Learning Kit

El *Arduino Tiny Machine Learning Kit* ofrece un entorno práctico y compacto para explorar cómo funciona el aprendizaje automático en dispositivos pequeños. Permite crear prototipos que reaccionan a gestos, sonidos o patrones visuales sin complicaciones, combinando hardware específico y cursos interactivos en línea que guían paso a paso cada experimento [2].

Este kit resulta útil para quienes buscan iniciarse en TinyML, ya que combina herramientas accesibles con ejemplos claros que simplifican la puesta en marcha de modelos de inferencia en tiempo real. Este se puede observar en la figura 13

Contenido del kit

- 1 × Placa **Arduino Nano 33 BLE Sense**
- 1 × Cámara **OV7675**
- 1 × **Arduino Tiny Machine Learning Shield**
- 1 × Cable USB A–Micro USB

5.3. Arduino Nano 33 BLE Sense Lite

El *Arduino Nano 33 BLE Sense Lite* acerca el aprendizaje automático al borde en un formato mínimo: basta conectar la placa por USB, cargar un *sketch* y comenzar a probar modelos TinyML [3]. Incluye los siguientes sensores, todos listados en la documentación oficial:

Sensores integrados

- IMU **LSM9DS1**: acelerómetro, giroscopio y magnetómetro.
- Micrófono MEMS **MP34DT05**: captura audio omnidireccional.

- Barómetro **LPS22HB**: mide presión y temperatura ambientales.
- Sensor **APDS9960**: detecta luz, color, proximidad y gestos.

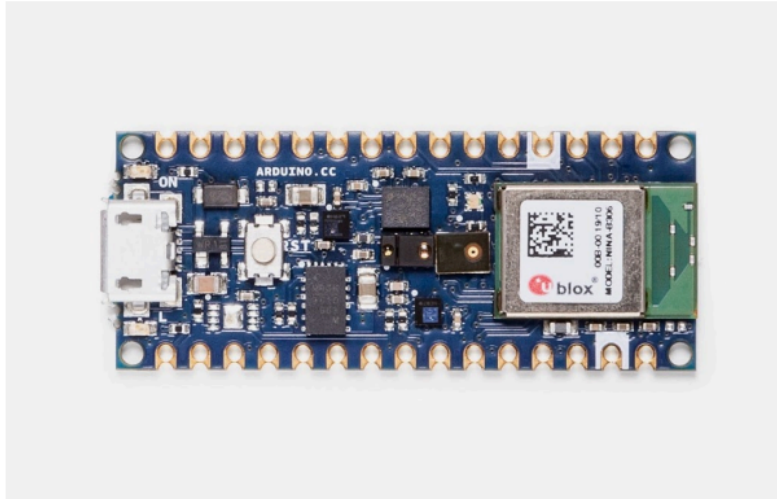


Figura 1: Arduino® Nano 33 BLE Sense [3]

Características generales – Arduino Nano 33 BLE Sense Lite [5]

- **Microcontrolador**: nRF52840 (Arm Cortex-M4F @ 64 MHz)
- **Memoria Flash**: 1 MB
- **RAM**: 256 KB
- **Voltaje de operación**: lógica a 3,3 V; las E/S no son tolerantes a 5 V
- **Conectividad inalámbrica**: Bluetooth Low Energy (BLE) 5.0
- **IMU**: LSM9DS1 (acelerómetro, giroscopio y magnetómetro de 9 ejes)
- **Interfaces de comunicación**: UART, SPI, I²C (TWI), QSPI, I²S, PDM, entre otras
- **E/S digitales**: 14 pines GPIO disponibles
- **Entradas analógicas**: 8 canales ADC de 12 bits (A0 – A7)
- **Peso**: el documento técnico no especifica el peso de la placa.

5.4. Micrófono MP34DT05

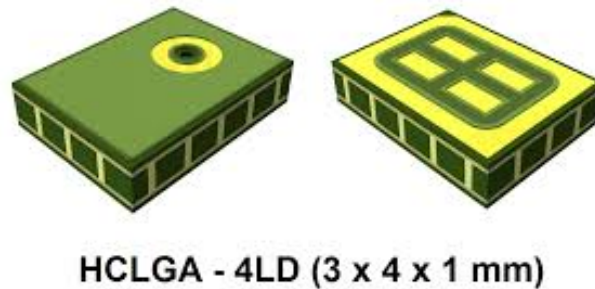


Figura 2: Microfono MP34DT05 [6]

El *MP34DT05* es un micrófono MEMS digital y omnidireccional que combina un elemento sensor capacitivo con una interfaz CMOS capaz de entregar audio en formato PDM. Su diseño top-port de $3 \times 4 \times 1$ mm facilita la integración en dispositivos compactos, mientras que el encapsulado HCLGA incluye blindaje EMI y cumple con normas RoHS. [6]

Aspectos destacados [6]

- **Rango de alimentación:** 1.6 V – 3.6 V, típico 1.8 V.
- **Consumo:** 650 μ A en modo activo; 5 μ A en apagado.
- **AOP:** 122.5 dBSPL (punto de sobrecarga acústica).
- **Sensibilidad:** -26 dBFS \pm 3 dB.
- **Relación señal-ruido:** 64 dB(A).
- **Salida:** audio digital en formato PDM.
- **Aplicaciones típicas:** reconocimiento de voz, sensores ambientales y dispositivos portátiles.

5.4.1. Diagrama de pines

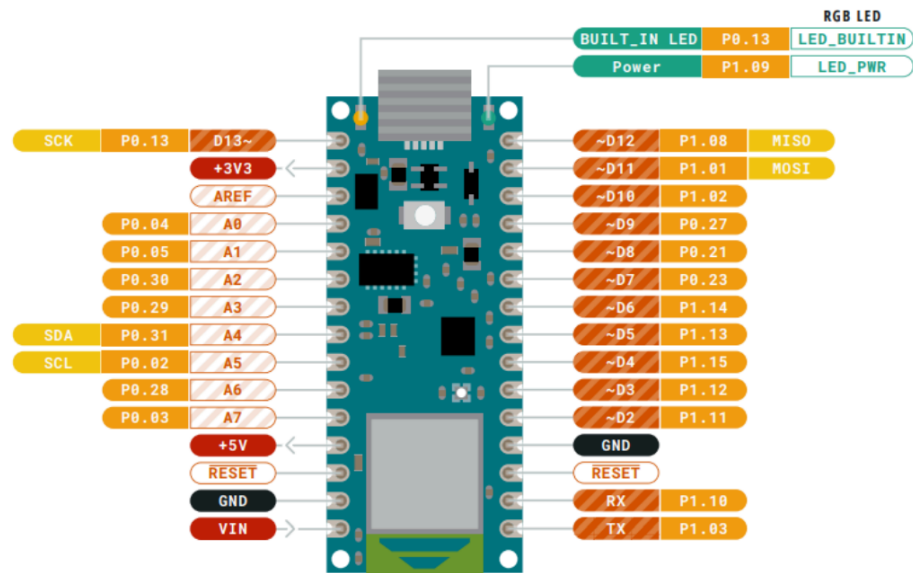


Figura 3: Connector Pinouts [3]

5.4.2. Descripción de pines

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 4: Descripción de pines [3]

5.4.3. Características eléctricas

Symbol	Description	Min	Typ	Max	Unit
PBL	Power consumption with busy loop		TBC		mW
PLP	Power consumption in low power mode		TBC		mW
PMAX	Maximum Power Consumption		TBC		mW

Figura 5: Power Consumption [3]

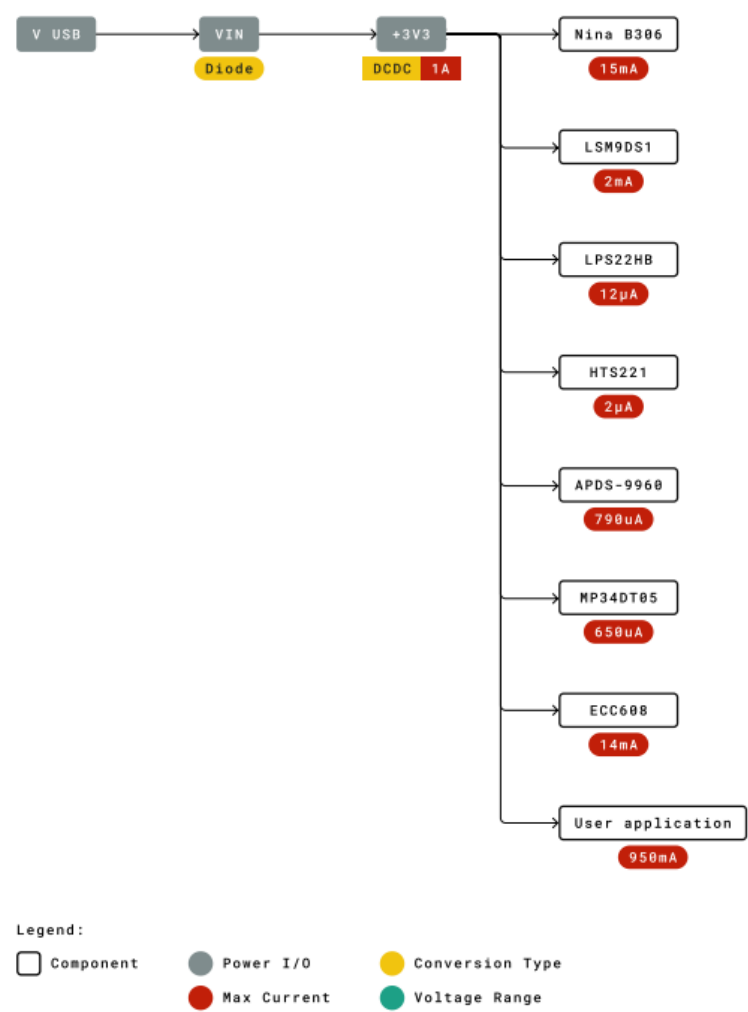


Figura 6: Power Tree [3]

5.5. Microcontrolador nRF52840

El *nRF52840* combina un núcleo Arm Cortex-M4F a 64 MHz con un transceptor de 2.4 GHz que admite Bluetooth LE 5 y protocolos 802.15.4. Su bajo consumo, la memoria integrada (1 MB de flash y 256 kB de RAM) y un motor criptográfico dedicado lo convierten en una opción sólida para dispositivos IoT y ropa electrónica que requieren conexión inalámbrica segura [4].

Características generales [4]

- Núcleo Arm Cortex-M4F @ 64 MHz.
- 1 MB de memoria flash y 256 kB de RAM.
- Radio multiprotocolo 2.4 GHz: Bluetooth LE 5, IEEE 802.15.4 y modos propietarios.
- Potencia de transmisión ajustable de -20 dBm a $+8$ dBm; sensibilidad de -95 dBm (1 Mbps).
- Rango de alimentación de 1.7 V a 5.5 V con regulador DC/DC interno.
- Interfaces: USB 2.0 FS, NFC-A, 48 GPIO, SPI, I²C, UART, QSPI, PDM/I²S, PWM y ADC de 12 bits.
- Motor de seguridad CryptoCell-310 y generador de números aleatorios verdaderos (TRNG).

5.5.1. Diagrama de pines

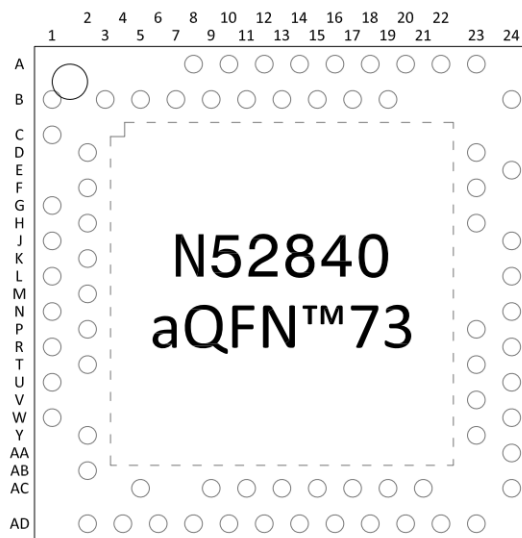


Figura 7: Diagrama de pines. [4]

5.5.2. Descripción de pines

Pin	Name	Function	Description	Recommended usage
A8	P0.31	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	AIN7	Analog input	Analog input	
A10	P0.29	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	AIN5	Analog input	Analog input	
A12	P0.02	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	AIN0	Analog input	Analog input	
A14	P1.15	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
A16	P1.13	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
A18	DEC2	Power	1.3 V regulator supply decoupling	
A20	P1.10	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
A22	VDD	Power	Power supply	
A23	XC2	Analog input	Connection for 32 MHz crystal	
B1	VDD	Power	Power supply	
B3	DCC	Power	DC/DC converter output	
B5	DEC4	Power	1.3 V regulator supply decoupling	Must be connected to DEC6 (pin E24)
B7	VSS	Power	Ground	
B9	P0.30	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	AIN6	Analog input	Analog input	
B11	P0.28	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	AIN4	Analog input	Analog input	
B13	P0.03	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	AIN1	Analog input	Analog input	
B15	P1.14	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
B17	P1.12	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
B19	P1.11	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
B24	XC1	Analog input	Connection for 32 MHz crystal	
C1	DEC1	Power	1.1 V regulator supply decoupling	
D2	P0.00	Digital I/O	General purpose I/O	
	XL1	Analog input	Connection for 32.768 kHz crystal	
D23	DEC3	Power	Power supply, decoupling	
E24	DEC6	Power	1.3 V regulator supply decoupling	Must be connected to DEC4 (pin B5)
F2	P0.01	Digital I/O	General purpose I/O	
	XL2	Analog input	Connection for 32.768 kHz crystal	
F23	VSS_PA	Power	Ground (radio supply)	
G1	P0.26	Digital I/O	General purpose I/O	
H2	P0.27	Digital I/O	General purpose I/O	
H23	ANT	RF	Single-ended radio antenna connection	See Reference circuitry on page 937 for guidelines on how to ensure good RF performance
J1	P0.04	Digital I/O	General purpose I/O	
	AIN2	Analog input	Analog input	
J24	P0.10	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only

Figura 8: Diagrama de pines pt1. [4]

Pin	Name	Function	Description	Recommended usage
K2	NFC2	NFC input	NFC antenna connection	
	P0.05	Digital I/O	General purpose I/O	
	AIN3	Analog input	Analog input	
L1	P0.06	Digital I/O	General purpose I/O	
L24	P0.09	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
	NFC1	NFC input	NFC antenna connection	
M2	P0.07	Digital I/O	General purpose I/O	
	TRACECLK	Trace clock	Trace buffer clock	
N1	P0.08	Digital I/O	General purpose I/O	
N24	DEC5	Power	1.3 V regulator supply decoupling for build codes Dxx and earlier.	
	Not connected		Not connected for build codes Fxx and later.	
P2	P1.08	Digital I/O	General purpose I/O	
P23	P1.07	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
R1	P1.09	Digital I/O	General purpose I/O	
	TRACEDATA3	Trace data	Trace buffer TRACEDATA[3]	
R24	P1.06	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
T2	P0.11	Digital I/O	General purpose I/O	
	TRACEDATA2	Trace data	Trace buffer TRACEDATA[2]	
T23	P1.05	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
U1	P0.12	Digital I/O	General purpose I/O	
	TRACEDATA1	Trace data	Trace buffer TRACEDATA[1]	
U24	P1.04	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
V23	P1.03	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
W1	VDD	Power	Power supply	
W24	P1.02	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
Y2	VDDH	Power	High voltage power supply	
Y23	P1.01	Digital I/O	General purpose I/O	Standard drive, low frequency I/O only
AA24	SWDCLK	Debug	Serial wire debug clock input for debug and programming	
AB2	DCCH	Power	DC/DC converter output	
AC5	DECI5B	Power	USB 3.3 V regulator supply decoupling	
AC9	P0.14	Digital I/O	General purpose I/O	
AC11	P0.16	Digital I/O	General purpose I/O	
AC13	P0.18	Digital I/O	General purpose I/O	QSPI/CSN
	nRESET		Configurable as pin RESET	
AC15	P0.19	Digital I/O	General purpose I/O	QSPI/SCK
AC17	P0.21	Digital I/O	General purpose I/O	QSPI
AC19	P0.23	Digital I/O	General purpose I/O	QSPI
AC21	P0.25	Digital I/O	General purpose I/O	
AC24	SWDIO	Debug	Serial wire debug I/O for debug and programming	
AD2	VBUS	Power	5 V input for USB 3.3 V regulator	
AD4	D-	USB	USB D-	
AD6	D+	USB	USB D+	

Figura 9: Diagrama de pines pt2. [4]

Pin	Name	Function	Description	Recommended usage
AD8	P0.13	Digital I/O	General purpose I/O	
AD10	P0.15	Digital I/O	General purpose I/O	
AD12	P0.17	Digital I/O	General purpose I/O	
AD14	VDD	Power	Power supply	
AD16	P0.20	Digital I/O	General purpose I/O	
AD18	P0.22	Digital I/O	General purpose I/O	QSPI
AD20	P0.24	Digital I/O	General purpose I/O	
AD22	P1.00	Digital I/O	General purpose I/O	QSPI
	TRACEDATA0	Trace data	Trace buffer TRACEDATA[0]	
			Serial wire output (SWO)	
AD23	VDD	Power	Power supply	
Die pad	VSS	Power	Ground pad	Exposed die pad must be connected to ground (VSS) for proper device operation

Figura 10: Diagrama de pines pt3. [4]

5.5.3. Diagrama de bloques

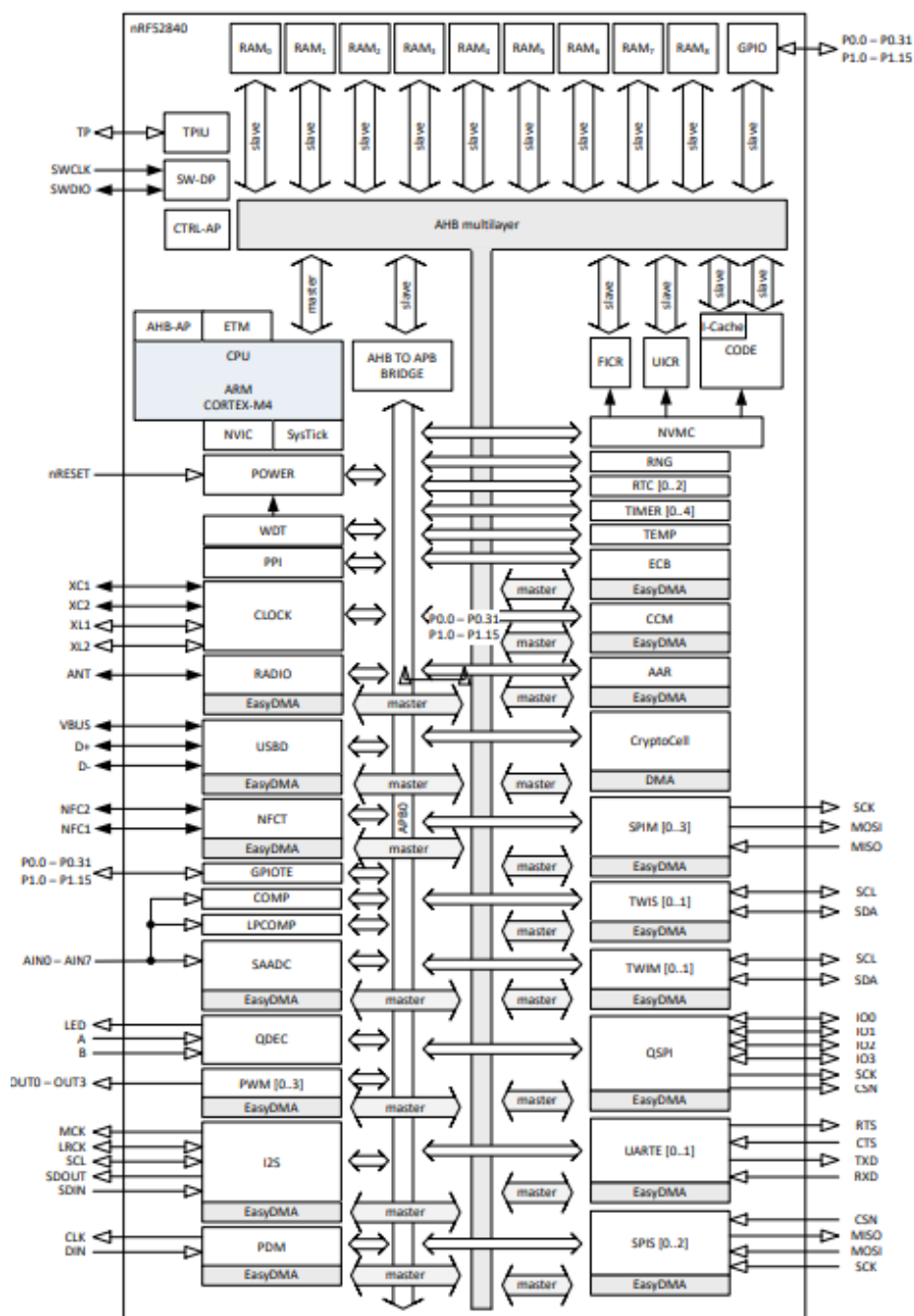


Figura 11: Diagrama de bloques. [4]

5.5.4. Características eléctricas

	Note	Min.	Max.	Unit
Supply voltages				
VDD		-0.3	+3.9	V
VDDH		-0.3	+5.8	V
VBUS		-0.3	+5.8	V
VSS			0	V
I/O pin voltage				
$V_{I/O}$, VDD \leq 3.6 V		-0.3	VDD + 0.3	V
$V_{I/O}$, VDD $>$ 3.6 V		-0.3	3.9	V
NFC antenna pin current				
$I_{NFC1/2}$			80	mA
Radio				
RF input level			10	dBm
Environmental aQFN73 package				
Storage temperature		-40	+125	°C
MSL	Moisture Sensitivity Level		2	
ESD HBM	Human Body Model		2	kV
ESD HBM Class	Human Body Model Class		2	
ESD CDM	Charged Device Model		450	V
Environmental QFN48 package				
Storage temperature		-40	+125	°C
MSL	Moisture Sensitivity Level		2	
ESD HBM	Human Body Model		4	kV
ESD HBM Class	Human Body Model Class		3A	
ESD CDM	Charged Device Model		1	kV
Environmental WLCSP 3.544 x 3.607 mm package				
Storage temperature		-40	+125	°C
MSL	Moisture Sensitivity Level		1	
ESD HBM	Human Body Model		1	kV
ESD HBM Class	Human Body Model Class		1C	
ESD CDM	Charged Device Model		500	V
Flash memory				
Endurance		10 000		write/erase cycles
Retention at 85 °C		10		years

Figura 12: Absolute maximum ratings [4]

5.6. TensorFlow Lite

TensorFlow Lite es un conjunto de herramientas pensado para llevar el aprendizaje automático directamente al dispositivo. Permite ejecutar modelos en móviles, sistemas embebidos o microcontroladores sin depender de un servidor externo, lo que reduce la latencia y protege los datos del usuario [7].

Características clave [7].

- Optimiza la inferencia en el borde y aborda cinco límites habituales: latencia, privacidad, conectividad, tamaño y consumo de energía
- Funciona en varias plataformas, incluidas iOS, Android, Linux embebido y microcontroladores.
- Ofrece API en Java, Swift, Objective-C, C++ y Python.
- Integra aceleración de hardware y herramientas de optimización que aumentan el rendimiento.
- Proporciona ejemplos completos para tareas como clasificación de imágenes, detección de objetos, estimación de poses y análisis de texto.
- El intérprete puede ocupar 1 MB con todos los operadores o menos de 300 KB si se compilan solo los necesarios.

5.7. Edge Impulse

Edge Impulse es una plataforma de aprendizaje automático pensada para el borde que cubre todo el flujo de trabajo: captura de datos, procesamiento de señales, entrenamiento, prueba y creación de un modelo listo para ejecutarse en el dispositivo. Su interfaz unifica herramientas que, de otro modo, exigirían varias soluciones distintas, y automatiza tareas repetitivas para acelerar la puesta en producción. [8]

Aspectos destacados [8]

- Un único entorno MLOps integrado; escala sin aprender otras herramientas.
- Exporta el modelo optimizado como biblioteca C++ con un clic para el hardware de destino.
- **EON Tuner**: ayuda a elegir la mejor combinación de preprocesado y modelo según las restricciones del dispositivo.
- **EON Compiler**: reduce hasta un 55 % de RAM y un 35 % de ROM sin perder precisión.
- **FOMO**: algoritmo que lleva la detección y cuenta de objetos en tiempo real a los microcontroladores.
- Despliegue agnóstico: funciona con cualquier sensor, procesador o flujo de trabajo existente, manteniendo la propiedad intelectual del usuario.

5.8. Machine Learning (ML)

Machine learning es una técnica de inteligencia artificial que enseña a los sistemas a aprender directamente de los datos en vez de basarse en ecuaciones fijas [9]. A medida que aumentan las muestras, los algoritmos ajustan sus parámetros y mejoran sus predicciones.

Modalidades principales

- **Aprendizaje supervisado:** el modelo parte de pares entrada-salida conocidos para inferir etiquetas (clasificación) o valores continuos (regresión).
- **Aprendizaje no supervisado:** identifica patrones ocultos sin necesidad de datos etiquetados; la técnica más habitual es la agrupación en *clusters*.

Estos enfoques resultan valiosos cuando los problemas implican grandes volúmenes de datos y relaciones complejas para las que no existe una fórmula explícita. El aprendizaje profundo constituye una variante especializada que automatiza la extracción de características, si bien exige conjuntos de datos más extensos y mayor potencia de cálculo [9].

5.9. Diseño del circuito

A continuación se presenta el diseño final del circuito, y en este caso el circuito corresponde únicamente al **kit Arduino Nano 33 BLE Sense Lite**, y para efectos de implementación solo se utiliza el microcontrolador como tal, y el uso de uno de sus periféricos el cual corresponde al micrófono MEMS.



Figura 13: Arduino Tiny Machine Learning Kit [2]
imagen con fines representativos

Tabla 1: Lista de componentes

Componente	Cantidad	Precio
Arduino Tiny Machine Learning Kit [2]	1	€61,50

5.9.1. Lista de componentes

5.9.2. Información adicional

Red(es) Neuronal(es)

Las redes neuronales artificiales son un tipo de algoritmo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Están formadas por nodos (neuronas) organizados en capas (entrada, ocultas y salida), que transforman datos de entrada mediante conexiones con pesos ajustables. Estas redes son capaces de aprender patrones complejos a partir de datos, siendo utilizadas en tareas como clasificación, reconocimiento de voz, procesamiento de imágenes y más. Su poder radica en la capacidad de generalizar a partir de ejemplos, lo que las hace útiles en problemas donde las reglas explícitas no son evidentes. [\[10\]](#)

6. Desarrollo

6.1. Análisis del firmware

6.1.1. Firmware

Para facilitar el proceso de prueba del modelo y su integración con el microcontrolador, se utiliza un firmware de ejemplo provisto por la plataforma Edge Impulse, específicamente adaptado para la placa Arduino Nano 33 BLE Sense Lite. Este firmware ya incluye la estructura necesaria para capturar datos del micrófono integrado, ejecutar inferencias con el modelo previamente entrenado en formato TensorFlow Lite, y mostrar los resultados a través del puerto serial.

El uso de este software de ejemplo permite una implementación rápida y confiable del modelo de clasificación de sonidos, sirviendo como base para validar el funcionamiento del sistema en tiempo real, mediante la identificación de eventos acústicos como ladridos, vidrios rotos, sirenas o sonidos ambientales.

6.2. Análisis electrónico

Como se menciona previamente, el análisis electrónico del sistema se centra exclusivamente en la utilización del microcontrolador Arduino Nano 33 BLE Sense Lite, el cual se conecta a la computadora mediante un cable USB. Esta conexión permite establecer la comunicación bidireccional entre el microcontrolador y la PC, facilitando tanto la programación del dispositivo como la visualización en tiempo real de los resultados obtenidos durante las inferencias.

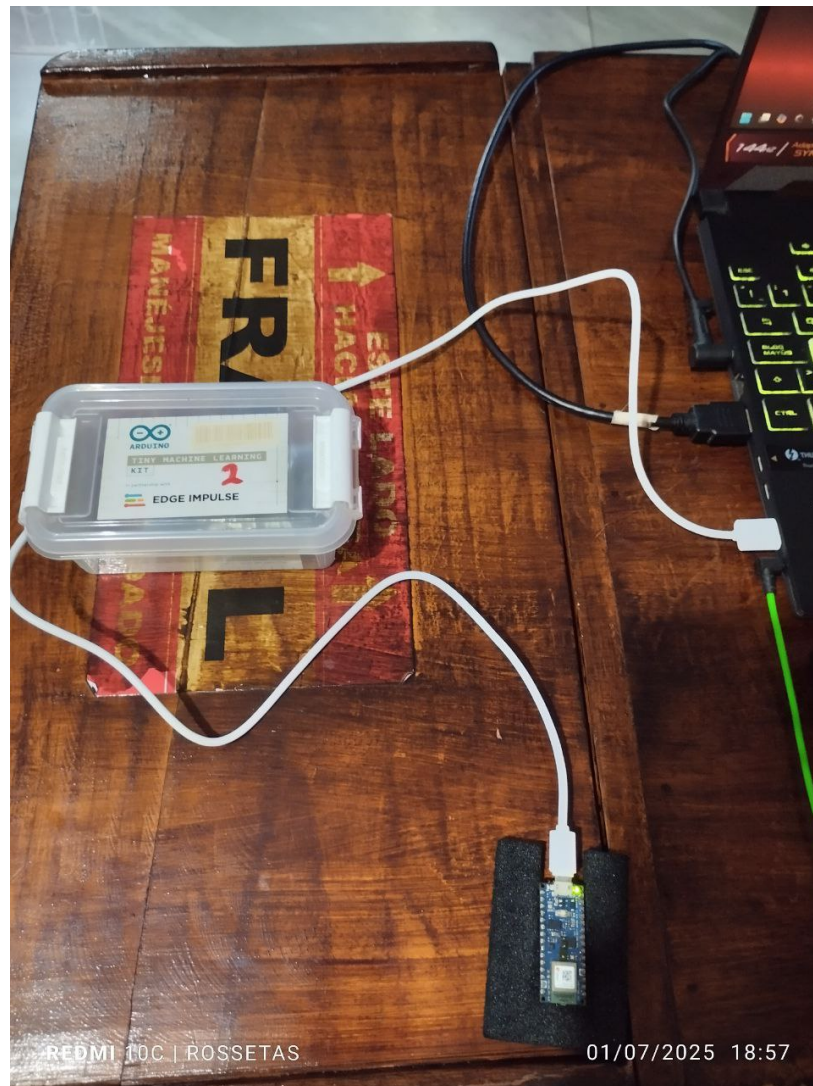


Figura 14: Circuito final
Elaboración propia

6.3. Análisis de resultados

El desarrollo del proyecto se centró en construir un sistema embebido capaz de reconocer sonidos específicos del entorno en tiempo real, utilizando técnicas de aprendizaje automático. Para lograr esto, se siguió una serie de etapas bien definidas que abarcan desde la captura de los datos hasta la inferencia embebida del modelo, con apoyo de herramientas como **Arduino** y la plataforma **Edge Impulse**.

6.3.1. Captura de datos con Arduino y Python

La primera etapa consistió en la recolección de datos de audio. Para esto, se programó la placa **Arduino Nano 33 BLE Sense Lite**, la cual integra un micrófono **PDM (Pulse Density Modulation)**, permitiendo la captura de señales acústicas del ambiente. Se desarrolló un programa en lenguaje C++ (Arduino IDE) mediante el Arduino IDE, que configura el micrófono para operar a una frecuencia de muestreo de **16 kHz** (16,000 muestras por segundo), enviando los datos crudos a través del puerto serial.

Simultáneamente, se implementó un script en **Python** que recibe las muestras en tiempo real mediante la comunicación serial, y realiza dos funciones principales:

- Guarda los datos numéricos en un archivo **.csv** para análisis posterior.
- Genera archivos **.wav**, formato estándar de audio, que permiten escuchar y verificar la calidad de las grabaciones.

Este proceso se repitió múltiples veces para capturar muestras representativas de distintas clases de sonidos, tales como **ladridos de perro**, **vidrios rotos**, **sirenas de policía** y **sonido ambiente**. Cada clase fue etiquetada correctamente para su posterior uso en el entrenamiento del modelo.

```
Presiona ENTER para grabar muestra 2/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 2 guardada: ladrido_02.wav

Presiona ENTER para grabar muestra 3/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 3 guardada: ladrido_03.wav

Presiona ENTER para grabar muestra 4/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 4 guardada: ladrido_04.wav

Presiona ENTER para grabar muestra 5/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 5 guardada: ladrido_05.wav

Presiona ENTER para grabar muestra 6/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 6 guardada: ladrido_06.wav

Presiona ENTER para grabar muestra 7/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 7 guardada: ladrido_07.wav

Presiona ENTER para grabar muestra 8/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 8 guardada: ladrido_08.wav

Presiona ENTER para grabar muestra 9/50 de 'ladrido'...
-> Grabando...
64000 muestras capturadas.
Muestra 9 guardada: ladrido_09.wav
```

Figura 15: Captura de datos pt1

Elaboración propia

Es importante destacar que durante el proceso de recolección de datos se capturó un número significativo de muestras para asegurar la calidad del entrenamiento. En total, se recopilaron **50 archivos de audio** por cada clase, cada uno con una duración promedio de **4 segundos**. Dado que la frecuencia de muestreo utilizada fue de **16,000 Hz**, esto equivale a un total aproximado de **64,000 muestras por archivo**, lo que proporciona una base sólida de datos para el posterior entrenamiento del modelo.

```
-> Grabando...
64000 muestras capturadas.
Muestra 46 guardada: vidrios_46.wav

Presiona ENTER para grabar muestra 47/50 de 'vidrios'...
-> Grabando...
64000 muestras capturadas.
Muestra 47 guardada: vidrios_47.wav

Presiona ENTER para grabar muestra 48/50 de 'vidrios'...
-> Grabando...
64000 muestras capturadas.
Muestra 48 guardada: vidrios_48.wav

Presiona ENTER para grabar muestra 49/50 de 'vidrios'...
-> Grabando...
64000 muestras capturadas.
Muestra 49 guardada: vidrios_49.wav

Presiona ENTER para grabar muestra 50/50 de 'vidrios'...
-> Grabando...
64000 muestras capturadas.
Muestra 50 guardada: vidrios_50.wav

Grabacion finalizada.
└─ audio [main] > python3 data audio.py
Nombre de la clase (ej: ladrido): sirena_polic
¿Cuántas muestras a grabar?: 50
Conectado a /dev/ttyACM0 a 230400 baudios.

Presiona ENTER para grabar muestra 1/50 de 'sirena_polic'...
-> Grabando...
64000 muestras capturadas.
Muestra 1 guardada: sirena_polic_01.wav

Presiona ENTER para grabar muestra 2/50 de 'sirena_polic'...
-> Grabando...
64000 muestras capturadas.
Muestra 2 guardada: sirena_polic_02.wav

Presiona ENTER para grabar muestra 3/50 de 'sirena_polic'...
```

Figura 16: Captura de datos pt2
Elaboración propia

6.3.2. Entrenamiento del modelo en Edge Impulse

Con las muestras capturadas, se procedió a utilizar la plataforma **Edge Impulse**, una herramienta especializada para crear, entrenar e implementar modelos de machine learning en dispositivos embebidos.

El flujo seguido dentro de Edge Impulse incluyó las siguientes fases:

6.3.3. Data Acquisition (Adquisición de datos)

En esta etapa se cargaron manualmente los archivos `.wav` previamente generados, asegurándose de asignar la etiqueta correcta a cada clase de sonido. Edge Impulse permite visualizar la forma de onda del audio para verificar su integridad. Además, la plataforma separa automáticamente los datos en subconjuntos de **entrenamiento** y **prueba**, lo cual es fundamental para evaluar objetivamente el rendimiento del modelo.

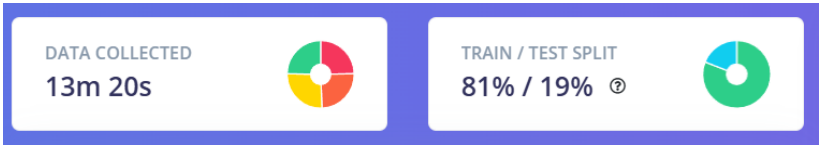


Figura 17: 4 Clases definidas
Elaboración propia

SAMPLE NAME	LABEL	ADDED	LENGTH	
vidrios_02	vidrios	Yesterday, 15:...	4s	⋮
vidrios_08	vidrios	Yesterday, 15:...	4s	⋮
vidrios_05	vidrios	Yesterday, 15:...	4s	⋮
vidrios_01	vidrios	Yesterday, 15:...	4s	⋮
sirena_polic_45	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_48	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_50	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_49	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_42	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_33	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_36	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_35	sirena	Yesterday, 15:...	4s	⋮

Figura 18: Audios `.WAV` de 4 segundos a 16 kHz pt1
Elaboración propia

SAMPLE NAME	LABEL	ADDED	LENGTH	
sirena_polic_12	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_11	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_08	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_09	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_01	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_05	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_07	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_06	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_03	sirena	Yesterday, 15:...	4s	⋮
sirena_polic_04	sirena	Yesterday, 15:...	4s	⋮
ladrido_45	ladrido	Yesterday, 15:...	4s	⋮
ladrido_48	ladrido	Yesterday, 15:...	4s	⋮

Figura 19: Audios .WAV de 4 segundos a 16 kHz pt2
Elaboración propia

6.3.4. Create Impulse

Aquí se define el flujo de procesamiento que transforma la señal cruda del micrófono en un conjunto de características (features) que pueda entender un modelo de machine learning. Se utilizó el bloque **MFCC (Mel Frequency Cepstral Coefficients)**, una técnica ampliamente usada en el procesamiento de voz y audio, que captura las propiedades acústicas esenciales del sonido en un formato compacto y robusto frente al ruido.

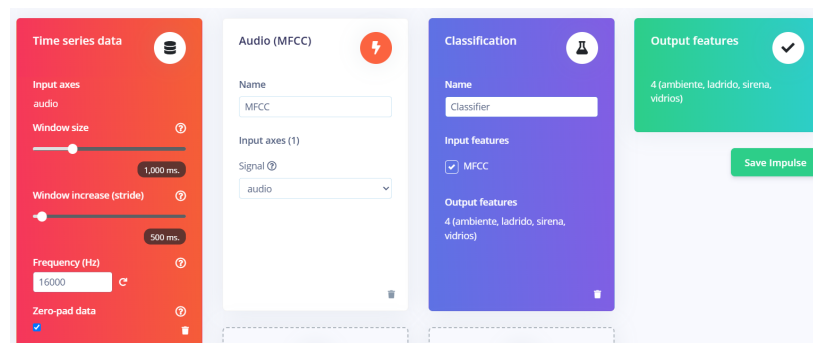


Figura 20: Impulso
Elaboración propia

6.3.5. MFCC

Durante la fase de procesamiento en Edge Impulse, uno de los pasos fundamentales fue la generación de características del audio mediante el uso de **MFCC (Mel Frequency Cepstral Coefficients)**. Este tipo de extracción de características es ampliamente utilizado en tareas de reconocimiento de audio, ya que permite representar de forma compacta y efectiva la información relevante del espectro de frecuencias del sonido.

Los **MFCC** transforman la señal de audio en un conjunto de coeficientes que reflejan cómo percibe el oído humano las diferentes frecuencias. En lugar de analizar directamente las ondas de sonido crudas, se extraen patrones temporales y espectrales más robustos, lo que facilita al modelo de clasificación detectar características distintivas entre diferentes sonidos como ladridos, vidrios rotos, sirenas, etc.

Esta conversión es esencial para mejorar la capacidad de generalización del modelo y para reducir la complejidad computacional durante la inferencia en el microcontrolador. Gracias a esta técnica, el sistema puede operar en tiempo real de forma más eficiente y con mayor precisión.



Figura 21: MFCC
Elaboración propia

6.3.6. Classifier

Se diseñó una red neuronal (modelo **Dense NN**) que toma como entrada los coeficientes MFCC y devuelve una predicción de la clase correspondiente al sonido capturado. El modelo fue entrenado utilizando el algoritmo de optimización **Adam**, y se utilizó la función de pérdida **categorical crossentropy**, común en problemas de clasificación multiclase.

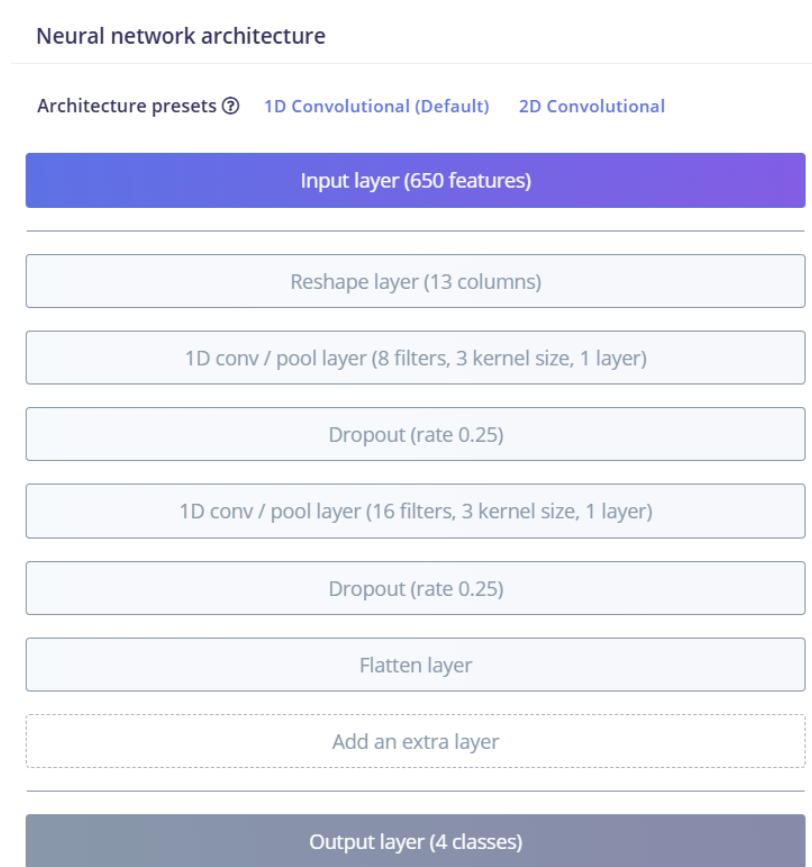


Figura 22: Neural network architecture
Elaboración propia

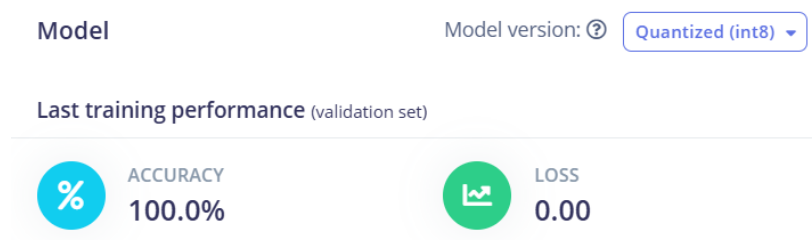


Figura 23: Last training performance
Elaboración propia

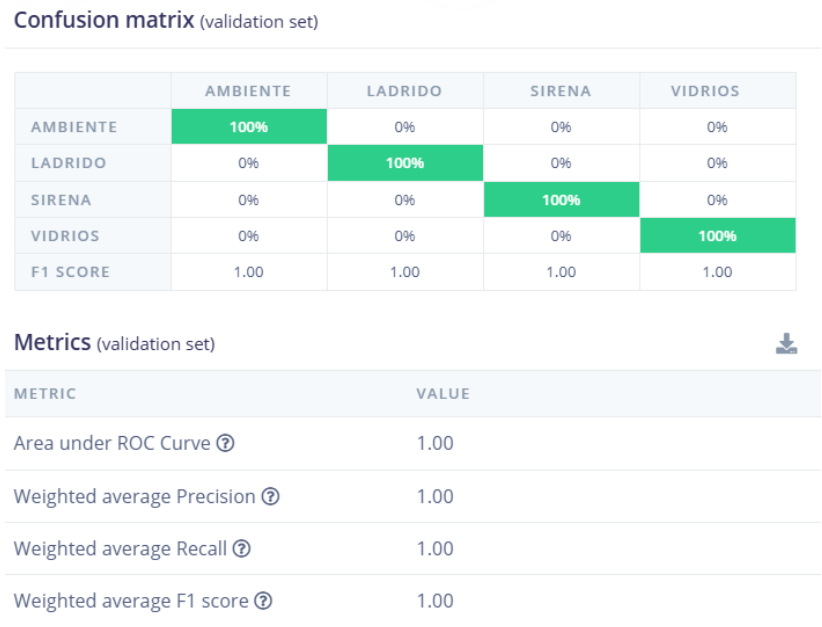


Figura 24: Confusion matrix
Elaboración propia

El modelo entrenado ocupa aproximadamente **45.9 KB** de memoria flash y **12.5 KB** de RAM, lo cual es adecuado para el microcontrolador Arduino Nano 33 BLE Sense Lite, que cuenta con 1 MB de flash y 256 KB de RAM. Estos valores aseguran que el modelo pueda ser implementado y ejecutado localmente sin problemas de memoria.



Figura 25: Data explorer
Elaboración propia

6.3.7. Model Testing

Una vez entrenado el modelo, se realizó una evaluación utilizando los datos de prueba reservados. Se obtuvo una matriz de confusión que permitió observar visualmente el desempeño de la clasificación, así como métricas clave como:

- Precisión (Accuracy): proporción de predicciones correctas.

Results



Figura 26: Precisión (Accuracy)
Elaboración propia

- Recall y F1-Score: especialmente útiles si hay clases desbalanceadas.

Estas métricas ayudaron a validar que el modelo era suficientemente preciso para ser desplegado.

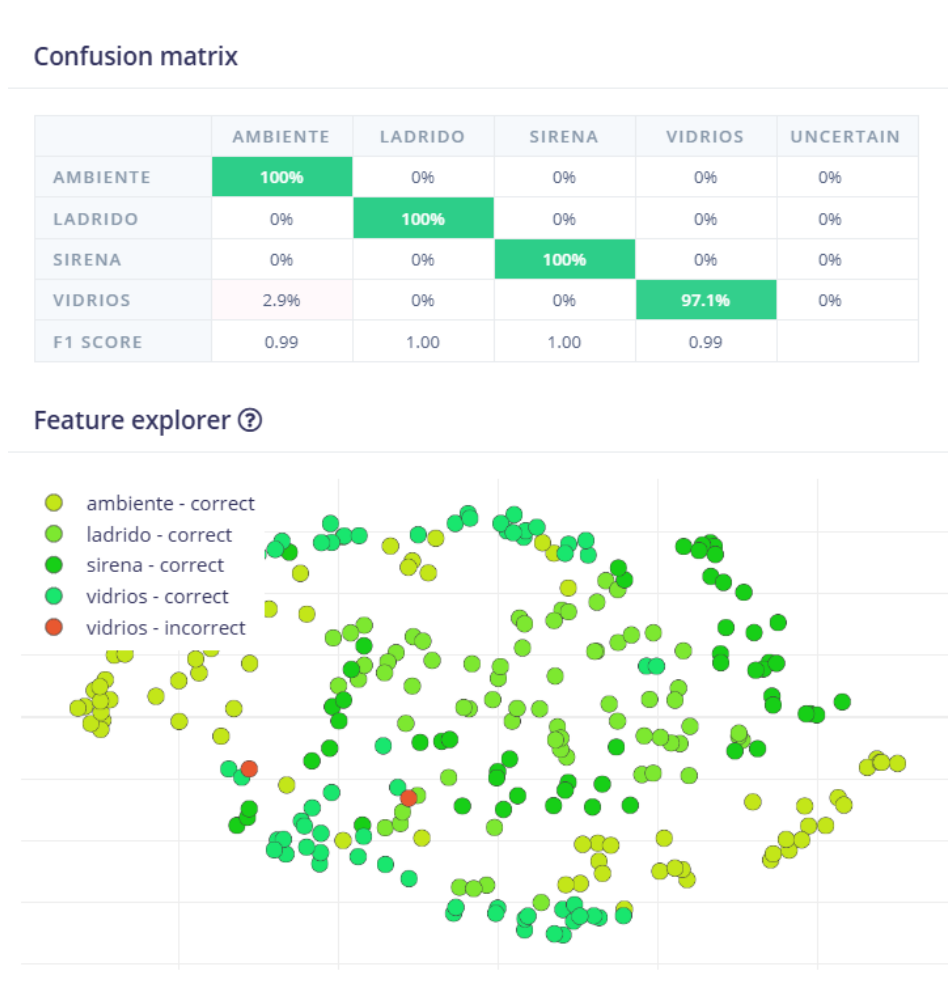


Figura 27: Resultados del model testing
Elaboración propia

6.3.8. Implementación del modelo en el microcontrolador

Con el modelo entrenado y validado, se procedió a generar un firmware en formato de **librería C++** para **Arduino**, utilizando la función de **Deployment** de Edge Impulse. Este paquete contiene el modelo optimizado, los bloques de preprocesamiento (MFCC) y ejemplos de código para ejecutar inferencias en la placa.


Se seleccionó un ejemplo de inferencia continua que permite al Arduino:


- Capturar audio en ventanas de tiempo predefinidas (por ejemplo, 1 segundo).
- Procesar las señales capturadas aplicando MFCC.
- Ejecutar la inferencia del modelo entrenado utilizando TensorFlow Lite Micro.
- Mostrar las probabilidades de cada clase por el puerto serial.

Este enfoque permite una **inferencia en tiempo real** completamente **offline**, es decir, sin necesidad de conexión a internet ni procesamiento en la nube.

Configure your deployment


You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)





SELECTED DEPLOYMENT
Arduino library
 An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS
 Model optimizations can increase on-device performance but may reduce accuracy.



TensorFlow Lite

Quantized (int8)

Selected ✓

	MFCC	CLASSIFIER	TOTAL
LATENCY	154 ms.	3 ms.	157 ms.
RAM	15.4K	6.4K	15.4K
FLASH	-	51.5K	-
ACCURACY			-

Figura 28: build
Elaboración propia

6.3.9. Validación funcional

Durante las pruebas finales se pudo comprobar que el sistema era capaz de identificar correctamente los sonidos para los que fue entrenado, distinguiendo entre un ladrido de perro, una sirena, un vidrio roto o simplemente ruido ambiental. Las predicciones se imprimían por el puerto serial, junto con sus probabilidades, lo cual permitió verificar su funcionamiento en situaciones reales.

Cabe destacar que todo el procesamiento se realiza directamente en el microcontrolador, demostrando así la **viabilidad técnica del aprendizaje automático embebido** en aplicaciones de monitoreo de seguridad y reconocimiento acústico.

```
Predictions (DSP: 308 ms., Classification: 8 ms., Anomaly: 0 ms.):
ambiente: 0.96094
ladrido: 0.00000
sirena: 0.00000
vidrios: 0.03906

Predictions (DSP: 309 ms., Classification: 8 ms., Anomaly: 0 ms.):
ambiente: 0.00000
ladrido: 0.00000
sirena: 0.00000
vidrios: 0.99609
```

Figura 29: Resultados
Elaboracion propia

7. Conclusiones y Recomendaciones

7.1. Conclusiones

- El proyecto demostró que es viable implementar modelos de aprendizaje automático en tiempo real sobre microcontroladores de bajo consumo como el Arduino Nano 33 BLE Sense Lite.
- Se logró una clasificación efectiva de sonidos mediante el uso de modelos ligeros y técnicas de procesamiento de señales, como MFCC, adecuadas para recursos embebidos.
- La integración con Edge Impulse facilitó significativamente todo el flujo de trabajo: desde la recolección de datos, diseño del modelo, entrenamiento, validación y despliegue.
- El sistema permite una respuesta inmediata ante eventos específicos del entorno, mostrando el potencial de estos dispositivos en aplicaciones reales de seguridad.

7.2. Recomendaciones

- Para mejorar la precisión del modelo, se recomienda ampliar el conjunto de datos, incluyendo mayor diversidad de fuentes sonoras y condiciones de grabación (ruido de fondo, distancia, etc.).
- Es posible integrar sensores adicionales como una cámara o un sensor de presencia, para complementar la detección acústica y obtener una solución multisensorial más robusta.
- Finalmente, se podría incorporar conectividad IoT (por ejemplo, MQTT o HTTP) para permitir la notificación remota de eventos y registro en plataformas en la nube.

Bibliografía

1. Arduino AG. *What is Arduino?*. Última revisión: 5 febrero 2018. Recuperado el 1 julio 2025, de <https://www.arduino.cc/en/Guide/Introduction>
2. Arduino. (s.f.). Arduino Tiny Machine Learning Kit. Arduino Official Store. <https://store.arduino.cc/products/arduino-tiny-machine-learning-kit>
3. Arduino. (s.f.). Nano 33 BLE Sense. Arduino Docs. <https://docs.arduino.cc/hardware/nano-33-ble-sense/>
4. Nordic Semiconductor. (s.f.). nRF52840 — key features. Nordic Semiconductor. https://docs.nordicsemi.com/bundle/ps_nrf52840/page/keyfeatures_html5.html
5. Arduino. (2025). *Arduino Nano 33 BLE Sense – Datasheet* (SKU ABX00031). Recuperado de <https://docs.arduino.cc/resources/datasheets/ABX00031-datasheet.pdf>
6. STMicroelectronics. (2021). *MP34DT05-A: MEMS Audio Sensor Omnidirectional Digital Microphone — Datasheet Rev. 5* (DS12239). Recuperado de <https://www.st.com/resource/en/datasheet/mp34dt05-a.pdf>
7. Google. (2021). *TensorFlow Lite – Guía*. Recuperado de <https://www.tensorflow.org/lite/guide?hl=es-419>
8. Edge Impulse & STMicroelectronics. (2025). *Edge Impulse – Partner Page*. Recuperado de https://www.st.com/content/st_com/en/partner/partner-program/partnerpage/Edge_Impulse.html
9. MathWorks. (2025). *Machine Learning — Qué es, cómo funciona y primeros pasos*. Recuperado de <https://la.mathworks.com/discovery/machine-learning.html>
10. IBM. (2025). *IBM. (n.d.). What is a neural network? IBM*. Recuperado de <https://www.ibm.com/think/topics/neural-networks>

8. Apéndices

8.1. Repositorio Git

Repositorio GitHub: https://github.com/Rossetas/MCU_Project