Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0624 – Laboratorio de Microcontroladores
III Ciclo 2022

Informe final de Proyecto: Clasificación de audio con Arduino Nano 33 BLE Sense

Profesor:
MSc. Marco Villalta Fallas

Estudiantes:
Raquel Sofía Corrales Marín B92378
Alexa Carmona Buzo B91643

3 de marzo de 2023

1. Introducción.

Como punto de partida para el proyecto correspondiente al curso *Laboratorio de Microcontroladores*, se parte de la propuesta de implementar una aplicación capaz de clasificar distintos sonidos a partir de audios. Para implementar la aplicación se utilizará como punto central la tarjeta *Arduino Nano 33 BLE* con los respectivos dispositivos electrónicos y digitales requeridos para concluir de manera satisfactoria los objetivos planteados, ejemplo de esto, la plataforma de IoT Thingsboard.

La clasificación de sonidos se llevará a cabo gracias a la capacidad del microcontrolador anteriormente mencionado, de soportar la biblioteca de software y además, open source *Tensor Flow*, esta última tiene por objetivo el aprendizaje automático y la inteligencia artificial. De modo tal que al hacer uso de esta biblioteca se pueda desarrollar un modelo de aprendizaje automático, algunas de las características más importantes es que utiliza modelos previamente entrenados o también permite entrenar el modelo, así como una robusta documentación en el campo del Machine Learning.

Este proyecto se enfoca en utilizar un modelo de machine learning capaz de diferenciar y clasificar entre tres sonidos diferentes que indiquen algún tipo de angustia o dolor como gritos, chillidos o llanto, por medio de un micrófono. Una vez el sonido ha sido reconocido y clasificado, este será enviado por medio de un puerto serial, donde un script recibirá dicho dato y será direccionado a un dashboard de Thingsboard.

Como referencia para este proyecto, se toma un programa en Arduino que utiliza redes neuronales para el reconocimiento de comandos de voz 'sí' y 'no' con la misma librería de machine learning [1]. Continuando con proyectos de reconocimiento de voz por medio de TensorFlow, se toma como referencia un tutorial que muestra cómo preparar el modelo para el reconocimiento de palabras claves como 'arriba', 'abajo', 'derecha' e 'izquierda' por medio de Python [2].

2. Objetivos y alcances.

2.1. Objetivo General

- Clasificar audios a partir de un modelo de TensorFlow haciendo uso del microcontrolador Arduino Nano 33 BLE Sense.

2.2. Objetivos Específicos

- Obtener y procesar los datos necesarios para las etapas de entrenamiento y prueba del modelo de clasificación.
- Implementar el modelo de clasificación de TensorFlow para las clases seleccionadas (gritos, chillidos o llanto).
- Mostrar resultados de la clasificación de los audios en un dashboard de Thingsboard.

3. Alcances

El presente proyecto busca mostrar una implementación básica de funcionalidades del machine learning en microcontroladores; de acuerdo con los objetivos y finalidad del proyecto, solo se explorará el módulo de reconocimiento de sonidos y palabras por lo que otros módulos como el reconocimiento de movimientos e imágenes quedan fuera de los alcances del proyecto. Asimismo, se describe el procedimiento a seguir desde una perspectiva general; es decir, no

se profundizará en los conceptos más específicos de la implementación del machine learning y será referido desde los conceptos más básicos e introductorios. Con respecto a las etapas, se divide el proyecto en 4 secciones principales: recolección de datos, entrenamiento e importación del modelo, implementación de la aplicación en el microcontrolador y envío de datos a Thingsboard. Para el análisis se realizará una comparación entre los resultados obtenidos a partir del cambio de varios parámetros del micrófono, el algoritmo de entrenamiento para el modelo y los sonidos utilizados; sin embargo, el proyecto no pretende utilizar las condiciones que permitan alcanzar una precisión de más de cierto porcentaje sino que se limita a manipular las condiciones mencionadas para definir la precisión más alta posible. De igual manera, queda fuera de los alcances del proyecto la justificación sobre los cambios de precisión causados por la manipulación de los parámetros ya que esto se relaciona directamente con la teoría de los algoritmos de entrenamiento.

4. Justificación

El machine learning es una rama de la inteligencia artificial y las ciencias de la computación, que se enfoca en el uso de algoritmos y datos para imitar el aprendizaje humano y que mejora continuamente [3]. El machine learning es utilizado actualmente en muchas aplicaciones diarias como la detección de caras en filtros de teléfonos móviles, en el análisis de imágenes y los autos autónomos. A pesar de la gran variedad de usos y el alcance del machine learning, los microcontroladores como el Arduino Nano 33 BLE son capaces de ejecutar aplicaciones que contienen machine learning. Lo anterior significa que es posible contar con dispositivos económicos y competentes que faciliten tareas como la vigilancia de una casa o de una persona con capacidades diferentes. En este caso, el reconocimiento de sonidos podría alertar a una persona con dificultades para oír o que se encuentre lejos de la fuente de ruido que algo se encuentra fuera de lugar.

5. Metodología (Lenguaje de programación y bibliotecas utilizadas).

El Arduino Nano 33 BLE se programa en el lenguaje de programación C++ con la biblioteca open source llamada TensorFlow y su biblioteca simplificada-orientada a microcontroladores TensorFlowLite, además, la biblioteca de Arduino, EloquentTinyML que permite llevar los modelos de machine learning a una simplificación mayor en placas de naturaleza Arduino.

El uso del Arduino Nano 33 BLE y la librería TensorFlow requiere una investigación de la hoja de datos del dispositivo [4] y un estudio del uso de las funciones de TensorFlow del módulo de reconocimiento de sonidos. Posteriormente, es necesaria la toma de los audios para los sets de entrenamiento y validación. Una vez completada esta preparación previa, se procede a programar el microcontrolador con el modelo para realizar la clasificación, finalmente, se procede al envío de los resultados a la plataforma de IoT, Thingsboard. Finalmente, se ejecutan una serie de pruebas con audios en tiempo real, capturados por el micrófono incorporado en la placa para verificar el funcionamiento del programa.

5.1. Materiales/Recursos empleados

En el desarrollo e implementación del proyecto solo fue necesario utilizar el Arduino Nano BLE 33 Sense y el micrófono MP34DT05 incorporado en la placa, de modo que no se requirió ningún componente electrónico/digital adicional.

6. Marco teórico.

Identificar lo que representa un audio se denomina *clasificación de audio*. En el mundo de la inteligencia artificial y el machine learning un modelo de clasificación de audio está entrenado bajo el objetivo de reconocer distintos eventos de audio, por ejemplo, es posible poder entrenar un modelo capaz de reconocer audios que representan tres eventos diferentes, entiéndase distintas palabras, sonidos como aplausos, entre otros [5].

Los procesos que llevan a desarrollar modelos computacionales capaces de clasificar eventos nuevos a partir de un previo entrenamiento del mismo con datos similares constituye lo que es hoy el aprendizaje computacional o machine learning. Pero es importante entender lo que existe detrás de todo esto, esa caja negra capaz de desarrollar comportamientos computacionales autónomos que hasta hace un par de décadas parecían situaciones del futuro, hoy no son más que el presente.

Esa caja negra, además, considerada la parte más difícil y misteriosa es la de entrenar una red neuronal con la capacidad de predecir y clasificar eventos (imágenes, audios, movimientos, objetos, rostros). Las redes neuronales en su interior están formadas por capas y nodos desde los cuales la información se transmite a las demás capas, de ahí el nombre de redes neuronales por su analogía con el proceso biológico que ocurre en el cerebro con las neuronas y los axones. A continuación, de manera más específica los elementos más destacables de las redes neuronales.

En primer lugar, en la estructura de las redes neuronales las unidades básicas son las neuronas, estas últimas se organizan en capas, en la figura 6 se muestra un ejemplo de una red neuronal, mostrando los nodos, capas de entrada y salida, así como las capas internas o escondidas como son comúnmente conocidas [6].

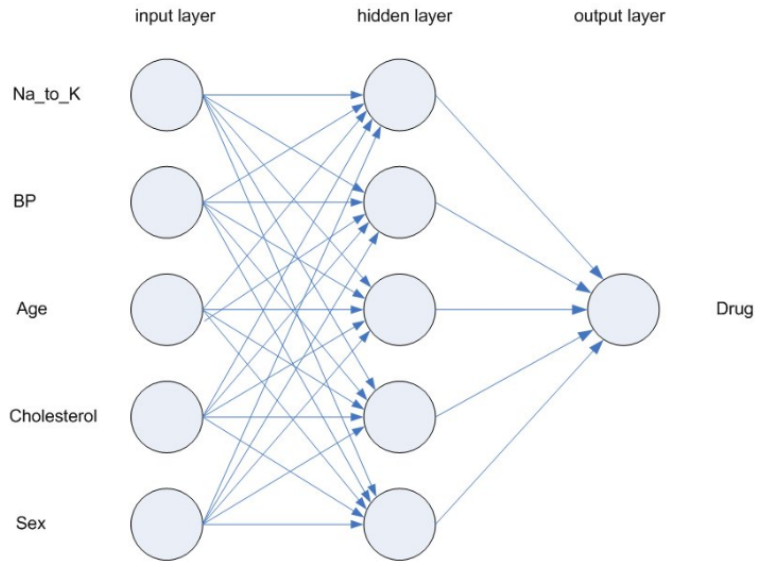


Figura 1: Estructura de una red neuronal [6].

Las capas o layers están definidas por tres etapas del proceso distintas, *las capas de entrada o input layers* es la que contiene las neuronas que representan los datos que serán utilizados por la red neuronal para entrenarse, la cantidad de estas está definida por la cantidad de atributos con los que cuentan los datos bajo estudio. *Las capas de ocultas o hidden layers*, usualmente las redes cuentan con muchas capas de estas, donde estas son conectadas con más capas de este tipo a través de las neuronas. Finalmente, están las *las capas de salida o output layer*, las cuales son encargadas de entregar los resultados a la salida, usualmente en implementaciones de clasificación, la capa de salida cuenta la misma cantidad de neuronas que de clases en las cuales se desea clasificar los datos [7].

Las neuronas o nodos son las que comunican la información entre capas como ya se mencionó, en cada una de estas se encuentra una función que comienza a subdividir los datos [7].

En cuanto a los datos, estos son divididos en tres grupos distintos, el primero es el *set de datos para training*, este es el utilizado para entrenar a la red neuronal con la finalidad de que esta aprenda patrones sobre los datos, *set de datos para validation*, este debe contener datos únicos, este set es utilizado para encontrar los mejores hiperparámetros de la red, por ejemplo, cantidad óptima de capas, nodos, entre otros. El *set de datos para test*, utilizado como su nombre lo indica para testear o comprobar el nivel de precisión de la red neuronal, en muchas aplicaciones se busca que estos datos sean generados en tiempo real, donde de manera inmediata o casi inmediata la aplicación debería ser capaz de realizar la clasificación [7].

Las *épocas*, corresponden al número de veces que se pasan los datos de entrenamiento a través de la red neuronal para que esta aprenda de los datos [7].

Ahora, es importante conocer que existen distintos algoritmos de aprendizaje supervisado, estos algoritmos son los utilizados para entrenar las redes neuronales y crear los modelos. Uno de estos algoritmos y muy conocido es el *Support vector machine (SVM)*, este lo que hace es encontrar un hiperplano que separa de la mejor forma posible dos clases diferentes de puntos de datos. Un aspecto importante a tomar en cuenta es que este algoritmo solo puede encontrar este hiperplano en problemas que permiten separación lineal; de modo que si este es el caso de los que se busca en la clasificación, el algoritmo maximizará este margen flexible permitiendo un pequeño número de clasificaciones erróneas [8].

El *Random Forest* consiste en otro algoritmo de aprendiza y además, en un metaestimador que ajusta una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobreajuste [9].

Existe otro algoritmo de aprendizaje supervisado *Naive Bayes* este aplica el teorema de Bayes con la suposición ingenua de independencia condicional entre cada par de características dado el valor de la variable de clase [10]. Este método se llama varias veces consecutivas en diferentes partes del conjunto de datos para implementar el aprendizaje. Este algoritmo de aprendizaje es útil y mayormente utilizado cuando todo el conjunto de datos es demasiado grande para caber en la memoria.

7. Desarrollo.

Para el desarrollo del proyecto se siguió el siguiente flujo de trabajo, donde primero era necesario recolectar muestras de datos de diferentes audios a través del micrófono incorporado en el Arduino. Seguidamente se procede a tomar los datos generados por el Arduino y se alimentan a un script de Python que entrena el modelo y lo importa de manera que sea posible utilizarlo en el microcontrolador. Luego se crea un firmware que utiliza el micrófono en el Arduino para captar los sonidos y los clasifica de acuerdo al modelo creado y procede a enviarlos por el puerto serial. Finalmente, un segundo script de Python envía la información a Thingsboard para su fácil visualización.

El flujo de trabajo descrito se muestra en el siguiente diagrama.

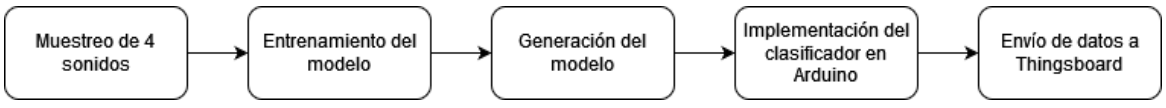


Figura 2: Flujo de implementación de un clasificador de sonidos por medio de TensorFlow

Como primer paso, es necesario obtener una serie de muestras de las cuatro clases para entrenar el modelo; cada muestra se compone de 64 entradas que caracterizan el sonido por medio de un número flotante como resultado de modulación por densidad de pulsos. En la figura 3 se muestra la salida del serial monitor con los vectores de 64 entradas.

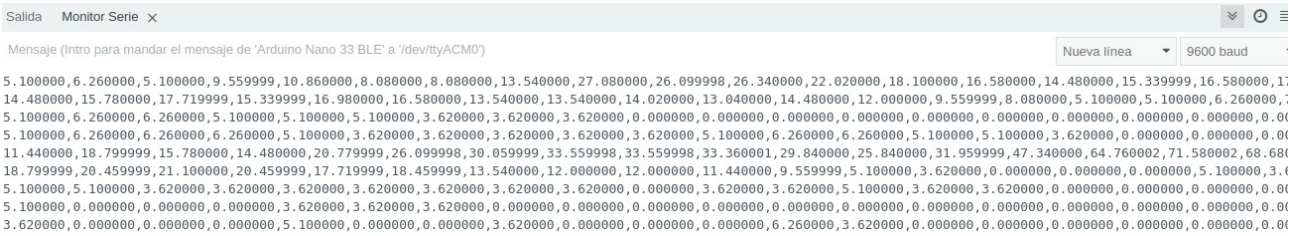


Figura 3: Serial monitor, toma de datos de voz.

Luego de que se obtuvo el set de datos, se realizó el entrenamiento del modelo, se implementó la creación del modelo con 3 algoritmos de entrenamientos supervisados distintos *Support Vector Machine, RandomForest, Naive Bayes*, de los cuales en la tabla 1 se muestra el accuracy alcanzado para cada uno de ellos.

Algoritmo de entrenamiento.	Accuracy
SVM, Support Vector Machine	36 %
RandomForest	52 %
Naive Bayes	32.79 %

Tabla 1: Accuracy para los distintos algoritmos de entrenamiento.

En el firmware del clasificador, el cual se encarga de captar el sonido y clasificarlo entre las cuatro clases, se envía el resultado a través del puerto serial; tal como se muestra en la figura 4.

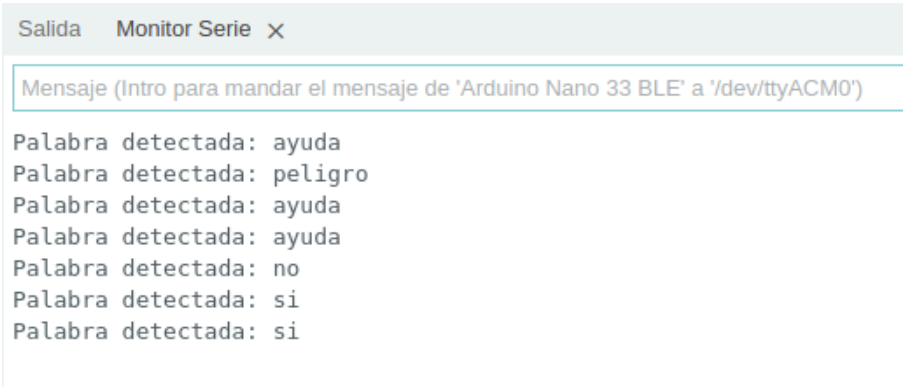


Figura 4: Salida del monitor serial del Arduino donde se realiza la clasificación.

Como se mencionó anteriormente, el firmware se encarga de enviar la palabra clasificada a través del puerto serial. Un script de Python se ejecuta de manera que lee la información enviada por el puerto en el que el Arduino está conectado. Este script también se encarga de organizar los datos en un diccionario de Python que luego se transforma en una estructura JSON; esta estructura se envía al servidor de Thingsboard luego de establecer una conexión segura con este.

En la imagen 5 se observa la clasificación de palabras en un dashboard de Thingsboard.

Figura 5: Panel de ThingsBoard para las distintas palabras detectadas

8. Análisis de resultados.

Tal como se mostró en los resultados, efectivamente se logró clasificar palabras en tiempo para las cuatro clases seleccionadas, sin embargo, la aplicación presenta algunas deficiencias, por ejemplo, que en ocasiones le toma decir más de 2 veces la palabra para que sea identificada y clasificada por el modelo de manera correcta, de modo que a continuación, se analizará en detalle las posibles causas del comportamiento que se obtuvo de la implementación del clasificador de audio haciendo uso del microcontrolador Arduino Nano 33 BLE Sense.

En primer lugar, es importante rescatar que la ganancia del micrófono y el threshold o umbral de sonido utilizado para la toma de datos y la respectiva clasificación son factores indispensables que afectan el proceso, es por ello que a pesar de haber utilizado distintos valores, el accuracy logrado no es tan alto como se esperaba.

Además, la toma de datos fue de 30 eventos para cada palabra con 64 entradas para cada uno de estos vectores de características con los que se entrenó la red neuronal, de modo que por los alcances del curso y la no búsqueda de los mejores hiperparámetros de la red, en términos de cantidad de neuronas, capas ocultas, pudo haber influido considerablemente en la precisión del modelo generado.

Es importante mencionar que se implementaron distintos algoritmos de aprendizaje supervisado; *Support vector machine (SVM)*, *Random Forest*, donde por la naturaleza de estos y su forma de entrenar, así como los parámetros que los rigen, se logró aumentar el accuracy en más del 10 %.

9. Conclusiones, observaciones y recomendaciones.

A partir de los resultados se evidencia la creación de un modelo para la clasificación de audios y su implementación en un microcontrolador como el Arduino nano 33 BLE por medio de redes neuronales y machine learning. Además fue posible ejecutar y observar la clasificación de 4 audios diferentes en tiempo real y enviar los resultados a una plataforma de IoT como Thingsboard. El proyecto permitió estudiar el comportamiento del clasificador con diferentes parámetros, a partir de los cuales se encontró que el modelo NOMBRE DEL MODELO tuvo la mejor precisión, así como el set de palabras tenía una mejor clasificación que el set de gritos

y sonidos de animales. Como era esperado, la modificación de los parámetros del micrófono tienen un impacto en el rendimiento ya que pueden hasta clasificar sonidos insignificantes.

Se recomienda basar las implementaciones en los ejemplos de TensorFlow ya que estos contienen código base que utiliza sensores y llamados a constructores y los modelos, los cuales son indispensables para completar el firmware. Además, se recomienda revisar el repositorio de TensorFlow Lite ya que algunos headers contienen errores. Por otro lado, se recomienda probar otros algoritmos de entrenamiento así como diferentes sets de datos para los eventos y entradas por palabra ya que un número mayor podría mejorar la precisión significativamente. Se podría además, considerar la clasificación de audios a partir de espectrogramas de generados a partir de estos mismos audios, ya que la clasificación de imágenes considera una amplia variedad de atributos.

10. Repositorio

Se incluye el enlace al repositorio en la plataforma GitHub, donde se incluye los archivos del proyecto.

`https://github.com/alebuzo/microcontroladores-proyecto`

Referencias

[1] TensorFlow Blog. *How-to Get Started with Machine Learning on Arduino*. 2019. URL: https://blog.tensorflow.org/2019/11/how-to-get-started-with-machine.html.

[2] TensorFlow. *Reconocimiento de audio simple: reconocimiento de palabras clave*. 2022. URL: https://www.tensorflow.org/tutorials/audio/simple_audio.

[3] IBM. *What is machine learning?* 2023. URL: https://www.ibm.com/topics/machine-learning.

[4] Arduino Documentation. *Nano 33 BLE Sense*. 2023. URL: https://docs.arduino.cc/hardware/nano-33-ble-sense.

[5] TensorFlow. *Clasificación de audio*. 2022. URL: https://www.tensorflow.org/lite/examples/audio_classification/overview?hl=es-419.

[6] IBM Documentation. *Nodo Red neuronal*. 2021. URL: https://www.ibm.com/docs/es/spss-modeler/saas?topic=SS3RA7_sub/modeler_mainhelp_client_ddita/clementine/trainnetnode_general.htm.

[7] V. Rodríguez. *Conceptos básicos sobre redes neuronales*. 2018. URL: https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales.

[8] MathWorks. *Support Vector Machine (SVM)*. 2023. URL: https://la.mathworks.com/discovery/support-vector-machine.html#:~:text=Support%5C%20vector%5C%20machine%5C%20(SVM)%5C%20es,reconocimiento%5C%20de%5C%20im%5C%C3%5C%A1genes%5C%20y%5C%20voz..

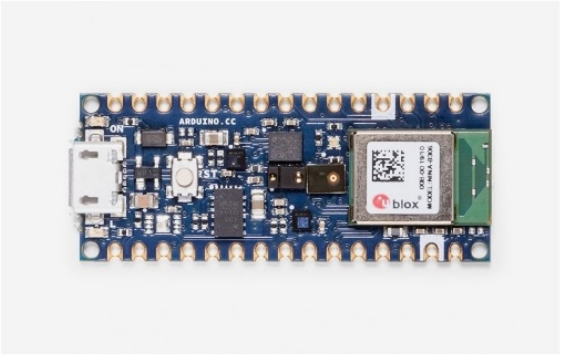
[9] ScikitLearn. *sklearn.ensemble.RandomForestClassifier*. 2023. URL: https://scikit%20learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[10] ScikitLearn. *sklearn.naive$_b$ayes.GaussianNB*. 2023. URL: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#.

11. Apéndice

Arduino® Nano 33 BLE Sense

Product Reference Manual
SKU: ABX00031



Description

Nano 33 BLE Sense is a miniature sized module containing a NINA B306 module, based on Nordic nRF52480 and containing a Cortex M4F, a crypto chip which can securely store certificates and pre shared keys and a 9 axis IMU. The module can either be mounted as a DIP component (when mounting pin headers), or as a SMT component, directly soldering it via the castellated pads

Target areas:

Maker, enhancements, IoT application

Features

- **NINA B306 Module**
  - **Processor**
    - 64 MHz Arm® Cortex-M4F (with FPU)
    - 1 MB Flash + 256 KB RAM
  - **Bluetooth® 5 multiprotocol radio**
    - 2 Mbps
    - CSA #2
    - Advertising Extensions
    - Long Range
    - +8 dBm TX power
    - -95 dBm sensitivity
    - 4.8 mA in TX (0 dBm)
    - 4.6 mA in RX (1 Mbps)
    - Integrated balun with 50 Ω single-ended output
    - IEEE 802.15.4 radio support
    - Thread
    - Zigbee
  - **Peripherals**
    - Full-speed 12 Mbps USB
    - NFC-A tag
    - Arm CryptoCell CC310 security subsystem
    - QSPI/SPI/TWI/I²S/PDM/QDEC
    - High speed 32 MHz SPI
    - Quad SPI interface 32 MHz
    - EasyDMA for all digital interfaces
    - 12-bit 200 ksps ADC
    - 128 bit AES/ECB/CCM/AAR co-processor
- **LSM9DS1** (9 axis IMU)
  - 3 acceleration channels, 3 angular rate channels, 3 magnetic field channels
  - ±2/±4/±8/±16 g linear acceleration full scale
  - ±4/±8/±12/±16 gauss magnetic full scale
  - ±245/±500/±2000 dps angular rate full scale
  - 16-bit data output
- **LPS22HB** (Barometer and temperature sensor)
  - 260 to 1260 hPa absolute pressure range with 24 bit precision
  - High overpressure capability: 20x full-scale
  - Embedded temperature compensation
  - 16-bit temperature data output
  - 1 Hz to 75 Hz output data rateInterrupt functions: Data Ready, FIFO flags, pressure thresholds
- **HTS221** (relative humidity sensor)
  - 0-100% relative humidity range
  - High rH sensitivity: 0.004% rH/LSB
  - Humidity accuracy: ± 3.5% rH, 20 to +80% rH
  - Temperature accuracy: ± 0.5 °C,15 to +40 °C
  - 16-bit humidity and temperature output data

- **APDS-9960** (Digital proximity, Ambient light, RGB and Gesture Sensor)
  - Ambient Light and RGB Color Sensing with UV and IR blocking filters
  - Very high sensitivity – Ideally suited for operation behind dark glass
  - Proximity Sensing with Ambient light rejection
  - Complex Gesture Sensing
- **MP34DT05** (Digital Microphone)
  - AOP = 122.5 dbSPL
  - 64 dB signal-to-noise ratio
  - Omnidirectional sensitivity
  - –26 dBFS ± 3 dB sensitivity
- **ATECC608A** (Crypto Chip)
  - Cryptographic co-processor with secure hardware based key storage
  - Protected storage for up to 16 keys, certificates or data
  - ECDH: FIPS SP800-56A Elliptic Curve Diffie-Hellman
  - NIST standard P256 elliptic curve support
  - SHA-256 & HMAC hash including off-chip context save/restore
  - AES-128 encrypt/decrypt, galois field multiply for GCM
- **MPM3610** DC-DC
  - Regulates input voltage from up to 21V with a minimum of 65% efficiency @minimum load
  - More than 85% efficiency @12V

Contents

1 The Board

As all Nano form factor boards, Nano 33 BLE Sense does not have a battery charger but can be powered through USB or headers.

NOTE: Arduino Nano 33 BLE Sense only supports 3.3V I/Os and is **NOT** 5V tolerant so please make sure you are not directly connecting 5V signals to this board or it will be damaged. Also, as opposed to Arduino Nano boards that support 5V operation, the 5V pin does NOT supply voltage but is rather connected, through a jumper, to the USB power input.

1.1 Ratings

1.1.1 Recommended Operating Conditions

Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C (40 °F)	85°C (185 °F)

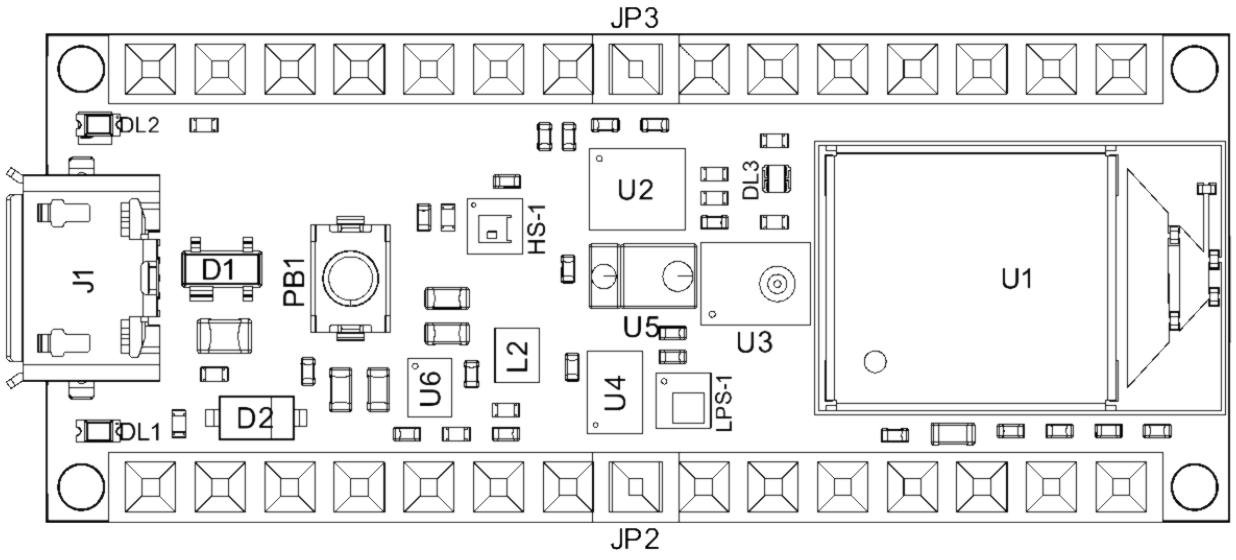
1.2 Power Consumption

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

2 Functional Overview

2.1 Board Topology

Top:

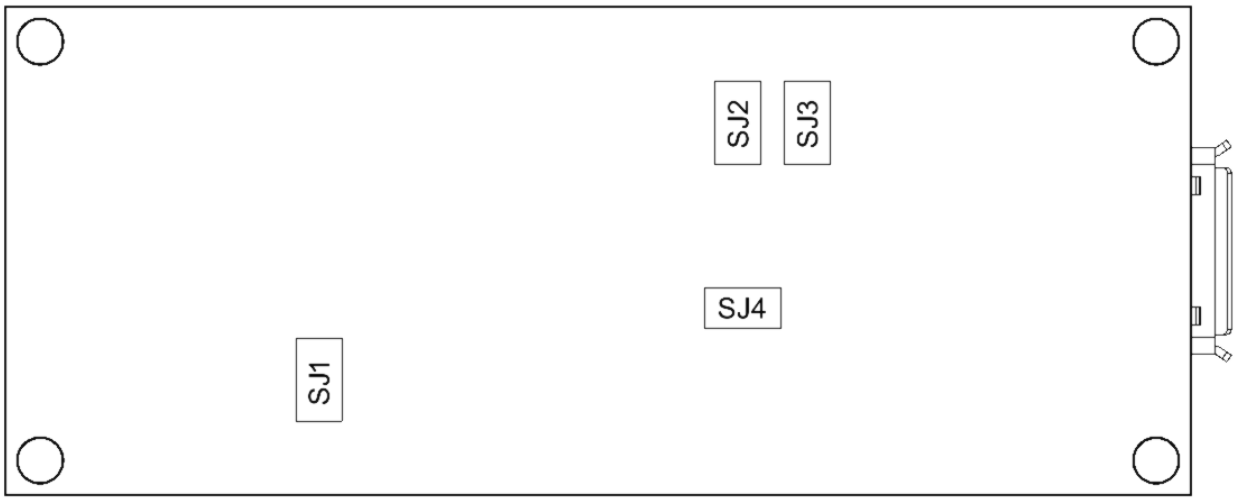


Board topology top

Ref.	Description	Ref.	Description
U1	NINA-B306 Module Bluetooth® Low Energy 5.0 Module	U6	MP2322GQH Step Down Converter
U2	LSM9DS1TR Sensor IMU	PB1	IT-1185AP1C-160G-GTR Push button
U3	MP34DT06JTR Mems Microphone	HS-1	HTS221 Humidity Sensor
U4	ATECC608A Crypto chip	DL1	Led L

Ref.	Description	Ref.	Description
U5	APDS-9660 Ambient Module	DL2	Led Power

Bottom:



Board topology bot

Ref.	Description	Ref.	Description
SJ1	VUSB Jumper	SJ2	D7 Jumper
SJ3	3v3 Jumper	SJ4	D8 Jumper

2.2 Processor

The Main Processor is a Cortex M4F running at up to 64MHz. Most of its pins are connected to the external headers, however some are reserved for internal communication with the wireless module and the on-board internal I^2C peripherals (IMU and Crypto).

NOTE: As opposed to other Arduino Nano boards, pins A4 and A5 have an internal pull up and default to be used as an I^2C Bus so usage as analog inputs is not recommended.

2.3 Crypto

The crypto chip in Arduino IoT boards is what makes the difference with other less secure boards as it provides a secure way to store secrets (such as certificates) and accelerates secure protocols while never exposing secrets in plain text.

Source code for the Arduino Library that supports the Crypto is available **[8]**

2.4 IMU

Arduino Nano 33 BLE has an embedded 9 axis IMU which can be used to measure board orientation (by checking the gravity acceleration vector orientation or by using the 3D compass) or to measure shocks, vibration, acceleration and rotation speed.

Source code for the Arduino Library that supports the IMU is available **[9]**

2.5 Barometer and Temperature Sensor

The embedded Barometer and temperature sensor allow measuring ambient pressure. The temperature sensor integrated with the barometer can be used to compensate the pressure measurement.

Source code for the Arduino Library that supports the Barometer is available **[10]**

2.6 Relative Humidity and Temperature Sensor

Relative humidity sensor measures ambient relative humidity. As the Barometer this sensor has an integrated temperature sensor that can be used to compensate for the measurement.

Source code for the Arduino Library that supports the Humidity sensor is available **[11]**

2.7 Digital Proximity, Ambient Light, RGB and Gesture Sensor

Source code for the Arduino Library that supports the Proximity/gesture/ALS sensor is available **[12]**

2.7.1 Gesture Detection

Gesture detection utilizes four directional photodiodes to sense reflected IR energy (sourced by the integrated LED) to convert physical motion information (i.e. velocity, direction and distance) to a digital information. The architecture of the gesture engine features automatic activation (based on Proximity engine results), ambient light subtraction, cross-talk cancellation, dual 8-bit data converters, power saving inter-conversion delay, 32-dataset FIFO, and interrupt driven I2C communication. The gesture engine accommodates a wide range of mobile device gesturing requirements: simple UP-DOWN-RIGHT-LEFT gestures or more complex gestures can be accurately sensed. Power consumption and noise are minimized with adjustable IR LED timing.

2.7.2 Proximity Detection

The Proximity detection feature provides distance measurement (E.g. mobile device screen to user's ear) by photodiode detection of reflected IR energy (sourced by the integrated LED). Detect/release events are interrupt driven, and occur whenever proximity result crosses upper and/ or lower threshold settings. The proximity engine features offset adjustment registers to compensate for system offset caused by unwanted IR energy reflections appearing at the sensor. The IR LED intensity is factory trimmed to eliminate the need for end-equipment calibration due to component variations. Proximity results are further improved by automatic ambient light subtraction.

2.7.3 Color and ALS Detection

The Color and ALS detection feature provides red, green, blue and clear light intensity data. Each of the R, G, B, C channels have a UV and IR blocking filter and a dedicated data converter producing16-bit data simultaneously. This architecture allows applications to accurately measure ambient light and sense color which enables devices to calculate color temperature and control display backlight.

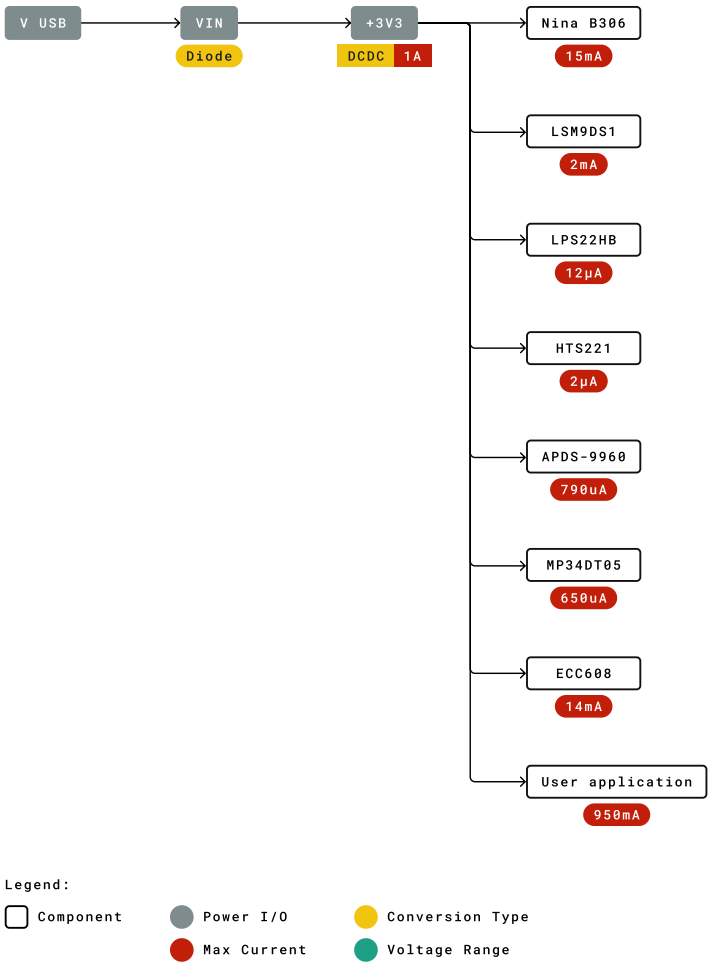
2.8 Digital Microphone

The MP34DT05 is an ultra-compact, low-power, omnidirectional, digital MEMS microphone built with a capacitive sensing element and an IC interface.

The sensing element, capable of detecting acoustic waves, is manufactured using a specialized silicon micromachining process dedicated to produce audio sensors

2.9 Power Tree

The board can be powered via USB connector, V_{IN} or V_{USB} pins on headers.



Power tree

NOTE: Since V_{USB} feeds V_{IN} via a Schottky diode and a DC-DC regulator specified minimum input voltage is 4.5V the minimum supply voltage from USB has to be increased to a voltage in the range between 4.8V to 4.96V depending on the current being drawn.

3 Board Operation

3.1 Getting Started - IDE

If you want to program your Arduino Nano 33 BLE while offline you need to install the Arduino Desktop IDE [1] To connect the Arduino Nano 33 BLE to your computer, you'll need a Micro-B USB cable. This also provides power to the board, as indicated by the LED.

3.2 Getting Started - Arduino Web Editor

All Arduino boards, including this one, work out-of-the-box on the Arduino Web Editor [2], by just installing a simple plugin.

The Arduino Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

3.3 Getting Started - Arduino IoT Cloud

All Arduino IoT enabled products are supported on Arduino IoT Cloud which allows you to Log, graph and analyze sensor data, trigger events, and automate your home or business.

3.4 Sample Sketches

Sample sketches for the Arduino Nano 33 BLE can be found either in the "Examples" menu in the Arduino IDE or in the "Documentation" section of the Arduino Pro website [4]

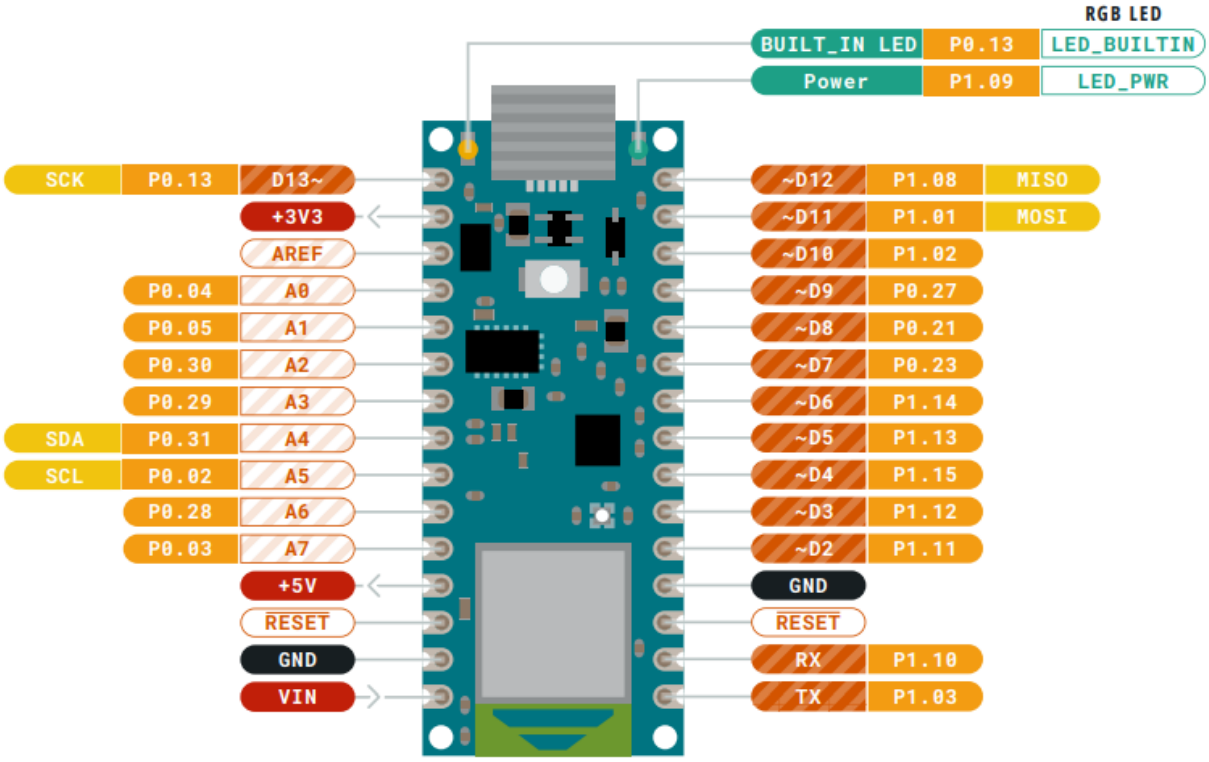
3.5 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[13]**, the Arduino Library Reference **[14]** and the on line store **[15]** where you will be able to complement your board with sensors, actuators and more.

3.6 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.

4 Connector Pinouts



Pinout

4.1 USB

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

4.2 Headers

The board exposes two 15 pin connectors which can either be assembled with pin headers or soldered through castellated vias.

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

Pin	Function	Type	Description
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

4.3 Debug

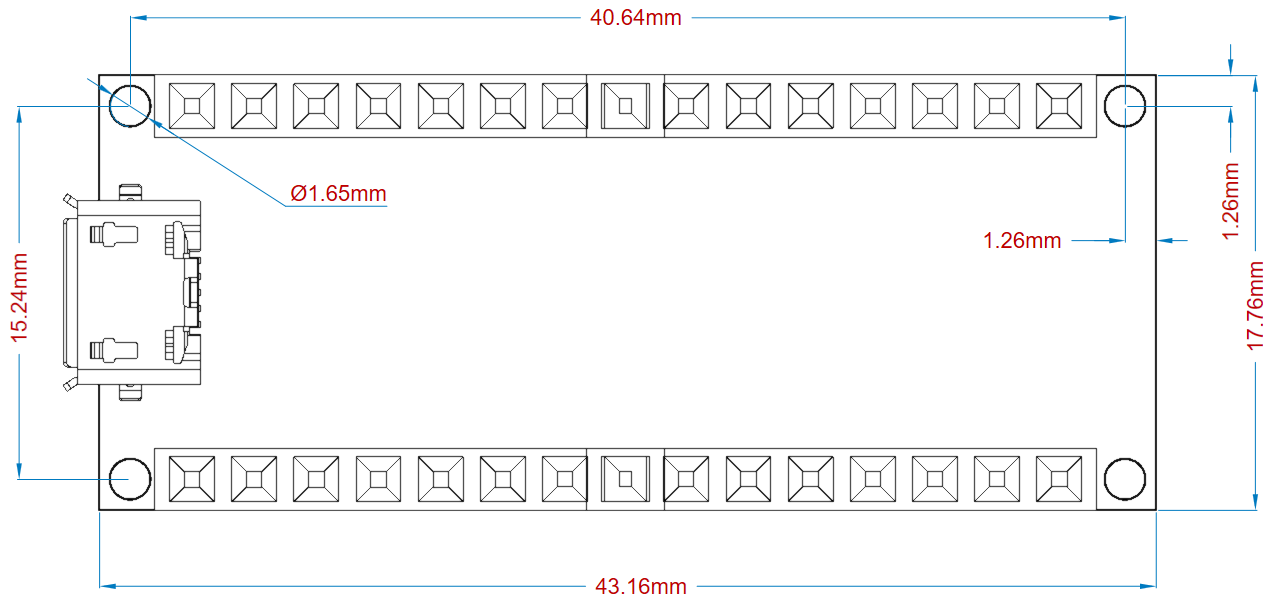
On the bottom side of the board, under the communication module, debug signals are arranged as 3x2 test pads with 100 mil pitch with pin 4 removed. Pin 1 is depicted in Figure 3 – Connector Positions

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

5 Mechanical Information

5.1 Board Outline and Mounting Holes

The board measures are mixed between metric and imperial. Imperial measures are used to maintain 100 mil pitch grid between pin rows to allow them to fit a breadboard whereas board length is Metric



Board layout

6 Certifications

6.1 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

6.2 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl} phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List"

(Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

6.3 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

7 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for license-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 2014/53/EU. This product is allowed to be used in all EU member states.

Frequency bands	Maximum output power (ERP)
863-870Mhz	5.47 dBm

8 Company Information

Company name	Arduino S.r.l
Company Address	Via Andrea Appiani 25 20900 MONZA Italy

9 Reference Documentation

Reference	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Forum	http://forum.arduino.cc/
Nina B306	https://content.u-blox.com/sites/default/files/NINA-B3_DataSheet_UBX-17052099.pdf
ECC608	https://ww1.microchip.com/downloads/aemDocuments/documents/SCBU/ProductDocuments/DataSheets/ATECC608A-CryptoAuthentication-Device-Summary-Data-Sheet-DS40001977B.pdf
MPM3610	https://www.monolithicpower.com/pub/media/document/MPM3610_r1.01.pdf
ECC608 Library	https://github.com/arduino-libraries/ArduinoECCX08
LSM6DSL Library	https://github.com/adafruit/Adafruit_LSM9DS1
LPS22HB	https://github.com/stm32duino/LPS22HB
HTS221 Library	https://github.com/stm32duino/HTS221
APDS9960 Library	https://github.com/adafruit/Adafruit_APDS9960
ProjectHub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/

10 Revision History

Date	Revision	Changes
08/03/2022	2	Reference documentation links updates
04/27/2021	1	General datasheet updates