

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

Laboratorio #5: STM32/Arduino: GPIO, Giroscopio, comunicaciones, TinyML

Denisse Ugalde Rivera C07893
Alonso José Jiménez Anchía B63561

Profesor: MSc. Marco Villalta Fallas

III-2023

Índice

1. Introducción	1
2. Nota teórica	2
2.1. Información general del Arduino Nano 33 BLE	2
2.1.1. Características	2
2.1.2. Diagrama de bloques	2
2.1.3. Diagrama de pines	2
2.2. Características eléctricas	4
2.3. Periféricos	4
2.4. Diseño de circuito	4
2.5. Lista de componentes y precios	5
3. Desarrollo	6
3.1. Funcionalidad del programa	6
3.1.1. Script de Arduino IMU_Capture.ino	6
3.1.2. Script de Python SensorDataCollector.py	6
3.2. Script de Python de Lab5_Google_Colab.ipynb	6
3.3. Script de Arduino imu_classifier.ino	7
3.4. Funcionalidad electrónica	7
3.4.1. Movimiento flex (inglés) o flexión	7
3.4.2. Movimiento punch (inglés) o golpe	10
3.4.3. Movimiento círculo	12
4. Conclusiones y Recomendaciones	14
5. Bibliografía	15
6. Apéndice	15

Índice de figuras

1.	Diagrama de bloques para el Arduino Nano 33 BLE	2
2.	Diagrama de pines para el Arduino Nano 33 BLE	3
3.	Especificaciones de los pines para el Arduino Nano 33 BLE	3
4.	Especificaciones eléctricas el Arduino Nano 33 BLE	4
5.	Diseño del circuito electrónico.	5
6.	Posición inicial de la mano antes de hacer la flexión.	8
7.	Posición final de la mano después de hacer la flexión.	9
8.	Resultados en terminal del movimiento flexión.	9
9.	Posición inicial de la mano antes de hacer el golpe.	10
10.	Posición final de la mano después de hacer el golpe.	11
11.	Resultados en terminal para el movimiento golpe.	12
12.	Movimiento del círculo en sentido antihorario.	13
13.	Resultados en terminal para el movimiento círculo.	13

1. Introducción

La práctica para este laboratorio se centra en el uso del Arduino Nano 33 BLE y TinyML para el reconocimiento de actividad humana (HAR). Con el estudio de conceptos relacionados con la inteligencia artificial y Machine Learning en sistemas embebidos, se busca utilizar sensores de movimiento para capturar datos que posteriormente son utilizados para entrenar modelos capaces de identificar diferentes tipos de actividades. Se utiliza Arduino IDE para escribir el código y cargar el programa a la placa, con Edge Impulse se crea y entrena el modelo de machine learning (TinyML) con los datos recogidos por el Arduino para lograr el reconocimiento de actividad humana, y para leer/almacenar los datos enviados por el Arduino a la computadora se usa el serial monitor.

Se logra crear un programa el cual detecta distintos movimientos como flexión, golpe o un círculo al mover el arduino, esto mediante los sensores de giroscopio y el acelerómetro. Este tipo de programas es muy utilizado en tecnología de industria y comercial, tal es el ejemplo de los controles de consolas para videojuegos los cuales tienen un sistema similar que detecta e identifica movimientos realizados con dichos controles.

En el siguiente link se encuentra el repositorio de trabajo: https://github.com/NisseUR/IE-0624_Lab5

2. Nota teórica

2.1. Información general del Arduino Nano 33 BLE

2.1.1. Características

Los microcontroladores utilizados en las placas Arduino están integrados en miles de millones de dispositivos de uso cotidiano, un ejemplo son los dispositivos *wearables* (en inglés), o bien, usables, ponibles. También, otros ejemplos son los drones, impresoras 3D, juguetes, arroceras, enchufes inteligentes y lavadoras, entre otros. Estos MCU de bajo costo buscan hacer más accesible el IoT. [1]. Este MCU se configura a partir de la plataforma de código abierto llamada Arduino. Asimismo, el MCU de la placa Arduino Nano BLE 33 se trata de un Arm Cortex-M4 que funciona a 64 MHz, con 1 MB de memoria Flash y 256 KB de RAM. [1]

2.1.2. Diagrama de bloques

Se muestra el diagrama de bloques MCU:

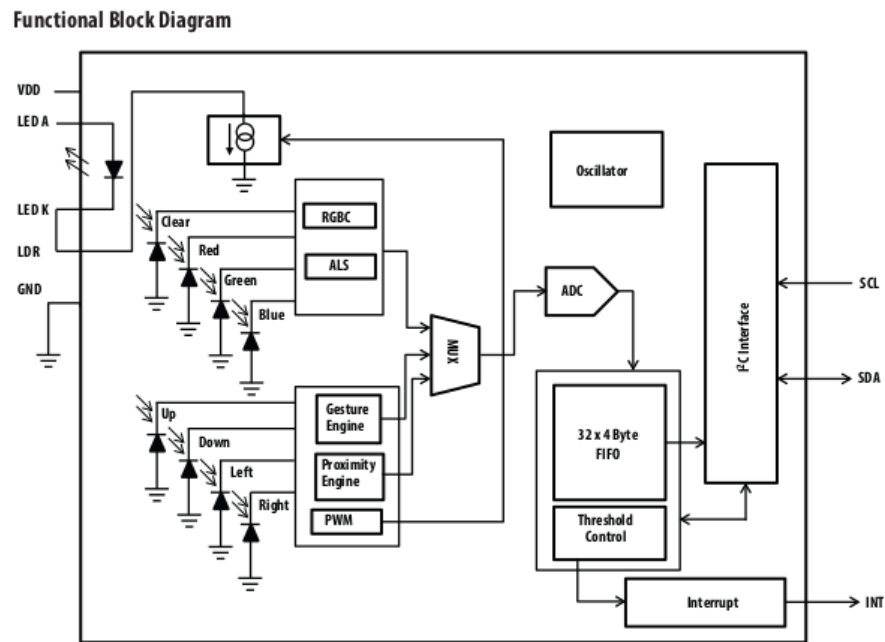


Figura 1: Diagrama de bloques para el Arduino Nano 33 BLE

2.1.3. Diagrama de pines

A continuación se muestra el diagrama de pines del Arduino Nano 33 BLE obtenido de la hoja de datos:

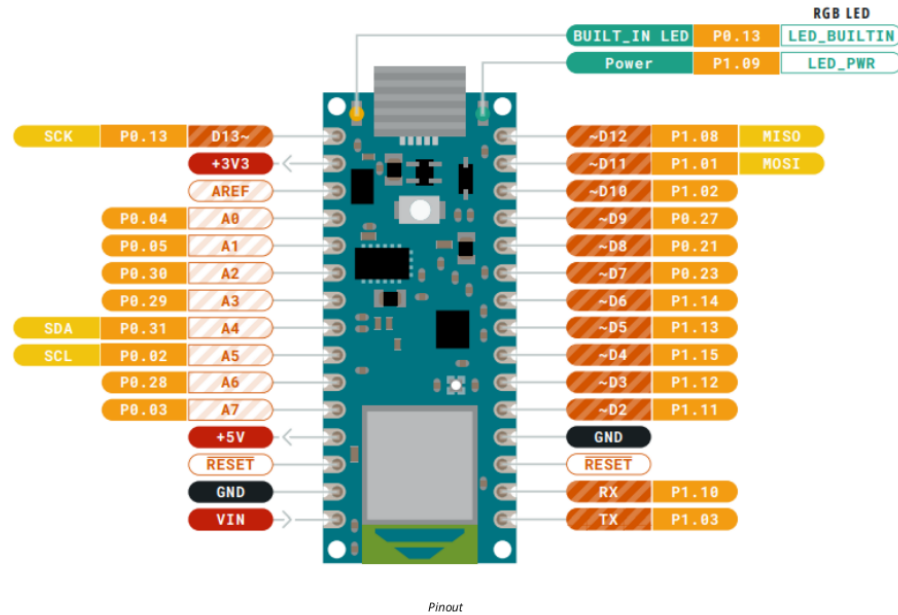


Figura 2: Diagrama de pines para el Arduino Nano 33 BLE

I/O Pins Configuration

Pin	Name	Type	Description
1	SDA	I/O	I ² C serial data I/O terminal - serial data I/O for I ² C-bus
2	INT	O	Interrupt - open drain (active low)
3	LDR		LED driver input for proximity IR LED, constant current source LED driver
4	LEDK		LED Cathode, connect to LDR pin when using internal LED driver circuit
5	LEDA		LED Anode, connect to V _{LEDA} on PCB
6	GND		Power supply ground. All voltages are referenced to GND
7	SCL	I	I ² C serial clock input terminal - clock signal for I ² C serial data
8	V _{DD}		Power supply voltage

Figura 3: Especificaciones de los pines para el Arduino Nano 33 BLE

2.2. Características eléctricas

Absolute Maximum Ratings over operating free-air temperature range (unless otherwise noted)*

Parameter	Symbol	Min	Max	Units	Conditions
Power supply voltage ^[1]	V _{DD}		3.8	V	
Input voltage range	V _{IN}	-0.5	3.8	V	
Output voltage range	V _{OUT}	-0.3	3.8	V	
Storage temperature range	T _{stg}	-40	85	°C	

* Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

Note 1. All voltages are with respect to GND.

Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Units
Operating ambient temperature	T _A	-30		85	°C
Power supply voltage	V _{DD}	2.4	3.0	3.6	V
Supply voltage accuracy, V _{DD} total error including transients		-3		+3	%
LED supply voltage	V _{LEDA}	3.0		4.5	V

Operating Characteristics, V_{DD} = 3 V, T_A = 25 °C (unless otherwise noted)

Parameter	Symbol	Min	Typ	Max	Units	Test Conditions
IDD supply current ^[1]	I _{DD}		200	250	μA	Active ALS state PON = AEN = 1, PEN = 0
			790			Proximity, LDR pulse ON, PPulse = 8 (I _{LDR} not included)
			790			Gesture, LDR pulse ON, GPulse = 8 (I _{LDR} not included)
			38			Wait state PON = 1, AEN = PEN = 0
			1.0	10.0		Sleep state ^[2]
V _{OL} INT, SDA output low voltage	V _{OL}	0		0.4	V	3 mA sink current
I _{LEAK} leakage current, SDA, SCL, INT pins	I _{LEAK}	-5		5	μA	
I _{LEAK} leakage current, LDR P ₁ pin	I _{LEAK}	-10		10	μA	
SCL, SDA input high voltage, V _{IH}	V _{IH}	1.26		V _{DD}	V	
SCL, SDA input low voltage, V _{IL}	V _{IL}			0.54	V	

Notes

1. Values are shown at the VDD pin and do not include current through the IR LED.

2. Sleep state occurs when PON = 0 and I2C bus is idle. If Sleep state has been entered as the result of operational flow, SAI = 1, PON will be high.

Figura 4: Especificaciones eléctricas el Arduino Nano 33 BLE

2.3. Periféricos

Se utilizan los periféricos utilizados son el acelerómetro y el giroscopio integrados en el Arduino Nano 33 BLE, donde estos sensores son parte del módulo LSM9DS1 y con los cuales se permite medir la aceleración y la orientación angular.

2.4. Diseño de circuito

En la figura 5, se aprecia la placa Arduino Board Nano 33 BLE Sense Lite montada en una placa extra para facilidad. También, un cable USB está transmitiendo la información hacia la PC y viceversa. Entonces, el diseño es simple en términos de la electrónica empleada; la mayor parte de la elaboración del laboratorio se llevo a cabó en programación.

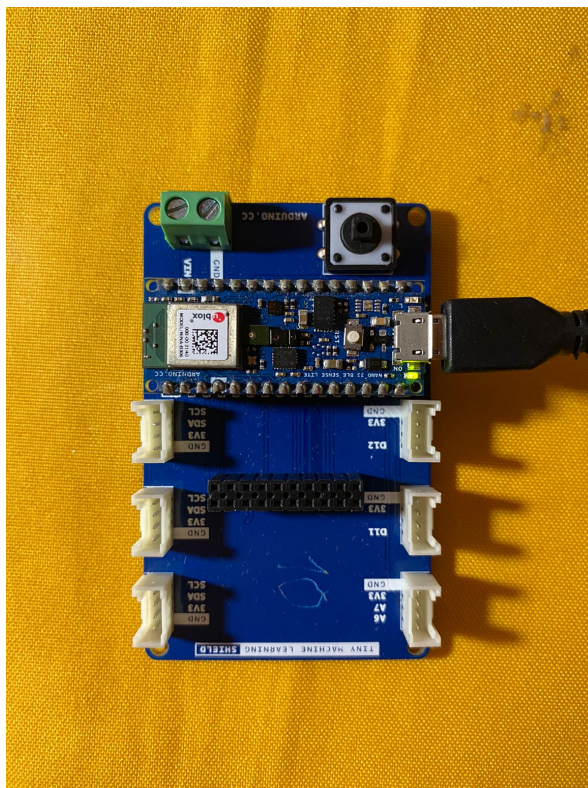


Figura 5: Diseño del circuito electrónico.

2.5. Lista de componentes y precios

Componente	Precio en colones
Arduino Nano 33 BLE Sense Lite	23 000

3. Desarrollo

3.1. Funcionalidad del programa

La mayor parte del código empleado se basa en el tutorial *Get Started With Machine Learning on Arduino* [1], con simples modificaciones para la integración de un tercer movimiento. El tutorial enseña como entrenar y usar modelos de machine learning para el Arduino Nano 33 BLE Sense.

3.1.1. Script de Arduino IMU_Capture.ino

Este ejemplo utiliza la IMU (Unidad de Medición Inercial) integrada para comenzar a leer datos de aceleración y giroscopio y los imprime en el Monitor Serie durante un segundo cuando se detecta un movimiento significativo. También, el código permite utilizar el Visualizador Serie, herramienta del Arduino IDE, para graficar los datos. La biblioteca Arduino_LSM9DS1 se usa para inicializar y leer los datos del IMU. Con la función de setup, se inicializa la comunicación serial con la consola así como con el IMU. También se verifica si la inicialización fue exitosa y se imprime el encabezado aX,aY,aZ,gX,gY,gZ para poder clasificar los datos.

Con el bucle loop, específicamente en el primer while, se está siempre esperando por la confirmación de aceleración en el movimiento de la placa. Al haber aceleración, se leen los 3 ejes respectivos y se estpa constantemente chequeando si la suma de las aceleraciones cruza el umbral de celeración. Cuando se cruza el umbral de aceleración, el conteo de muestras leídas se vuelve cero.

El segundo while se cumple siempre y cuando la muestras de datos leídos sea menor al número de muestras. Es en este while donde se leen los datos del ejes respectivos del IMU para lo que es aceleración y los ejes respectivos del giroscopio. Todos estos datos son enviados por medio del puerto serial a la computadora, y son formateados para una fácil lectura. Por último, finalizada la lectura y toma de muestras, se agrega al puerto serial una línea vacía que significa que la toma de muestras ha finalizado y se puede realizar la siguiente. Este espacio vacío también es importante para el entrenamiento y uso del machine learning a aplicar más adelante.

3.1.2. Script de Python SensorDataCollector.py

Se crea un script de Python llamado SensorDataCollector.py para leer datos del puerto serial enviados por el código Arduino IMU_Capture.ino mencionado previamente. Luego, con este script de Python, se puede leer los datos desde el puerto serie, se procesa esta información extrayendo los valores numéricos de las lecturas realizadas del acelerómetro y el giroscopio, donde finalmente se guardan estos datos en un archivo CSV para realizar el análisis que sea necesario. En resumen, el sketch .ino funciona como un emisor de datos, y el script de Python actúa como el receptor de los valores generados.

3.2. Script de Python de Lab5_Google_Colab.ipynb

Al script original [1] se le agrega un *gesture* más, que para este laboratorio es el movimiento de hacer un círculo con la mano. Este script de Python es específico para correr en Google Colab donde se entrena el modelo de aprendizaje automático utilizando los datos recopilados de

la placa Arduino en el sketch `IMU_Capture.ino` y del script `SensorDataCollector.py`. Los datos se encapsulan en documentos `.csv` entre 10-11 muestras por movimiento. Colab proporciona un cuaderno Jupyter que permite ejecutar el entrenamiento de TensorFlow en un navegador web, y así evitar utilizar los recursos de nuestras portátiles. Luego, a partir del tutorial [1], se entrena el modelo a partir de machine learning siguiendo los siguientes pasos:

1. Configurar el entorno de Python: Se configuran las dependencias necesarias para el cuaderno. [1]
2. Cargar los datos de `punch.csv`, `flex.csv` y `circulo.csv`.
3. Analizar y preparar los datos: Se analizan los archivos CSV y se transforman a un formato que se utilizará para entrenar la red neuronal. [1]
4. Construir y entrenar el modelo: Se construye y entrena un modelo de TensorFlow utilizando la API de alto nivel Keras. [1] `tf.keras` es la API de alto nivel de TensorFlow para construir y entrenar modelos de aprendizaje profundo. Se utiliza para la creación rápida de prototipos, la investigación de vanguardia (estado-del-arte) y en producción. [2]
5. Convertir el modelo entrenado a TensorFlow Lite: Se convierte el modelo al formato TFLite. [1]
6. Codificar el modelo en un archivo de encabezado de Arduino: El paso final del Colab genera el archivo `model.h` para descargar e incluir en nuestro proyecto de clasificación de gestos en el Arduino IDE. [1]

3.3. Script de Arduino `imu_classifier.ino`

Este script hace uso del IMU de 9 ejes presente en la placa para empezar a medir la orientación de la placa (datos del giroscopio) así como la aceleración. Cuando tiene los suficientes muestras de datos, hace uso del modelo TensorFlow Lite (Micro) o bien, del archivo `modelo.h` para clasificar el movimiento ejecutado por el usuario y clasifica dicho movimiento entre los posibles: flexión, golpe o círculo. La correspondencia numérica según el movimiento ejecutado se refleja en terminal. Se tienen 3 simples movimientos registrados a identificar por el algoritmo: flexión, golpe y círculo; todos realizados con la mano del usuario. Una parte importante de este sketch es la biblioteca TensorFlow Lite Micro para Arduino, la cual está diseñada para ejecutar modelos de aprendizaje automático y la cual es compatible con la mayoría de las placas basadas en Arm Cortex M, como el Arduino Nano 33 BLE Sense. El repositorio se puede encontrar en <https://github.com/tensorflow/tflite-micro-arduino-examples#how-to-install>.

3.4. Funcionalidad electrónica

3.4.1. Movimiento flex (inglés) o flexión

En la figura 6, se tiene la posición inicial para el movimiento de flexión tipo levantando una mancuerna. Note la posición de la placa en la mano. Consecutivamente, el movimiento se hace rápido para “despertar” el LSM9DS1 para que inicie la toma de datos. La toma de datos se da en el puerto serial COM4 puesto que se utilizó Windows y estos datos pasan a escribirse en un archivo `.csv` a través del script de Python previamente mencionado. Un movimiento lento no hará que se inicie la toma de datos, sin embargo, esto se nota al monitorear la terminal tanto del Arduino IDE con la opción Serial Monitor, o bien, a través de `arduino-cli` (antes

cerrar Arduino IDE) y monitorear el puerto desde esta terminal arduino-cli con el comando `arduino-cli monitor -p COM4`.

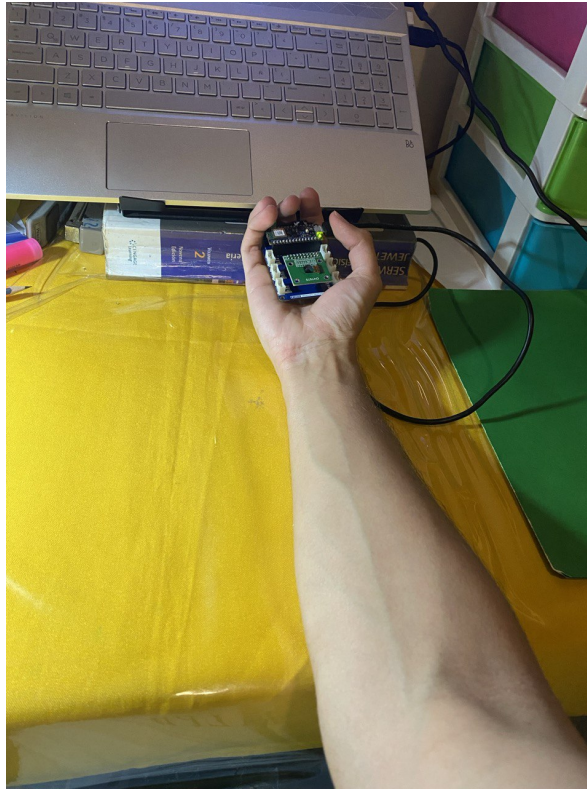


Figura 6: Posición inicial de la mano antes de hacer la flexión.

La figura 8 presenta la posición final del movimiento. Este movimiento se realizó un total de 10 veces, lo cual fue suficiente debido a los altos porcentajes de eficiencia a la hora de identificar el movimiento presentes en los resultados en figura 8.

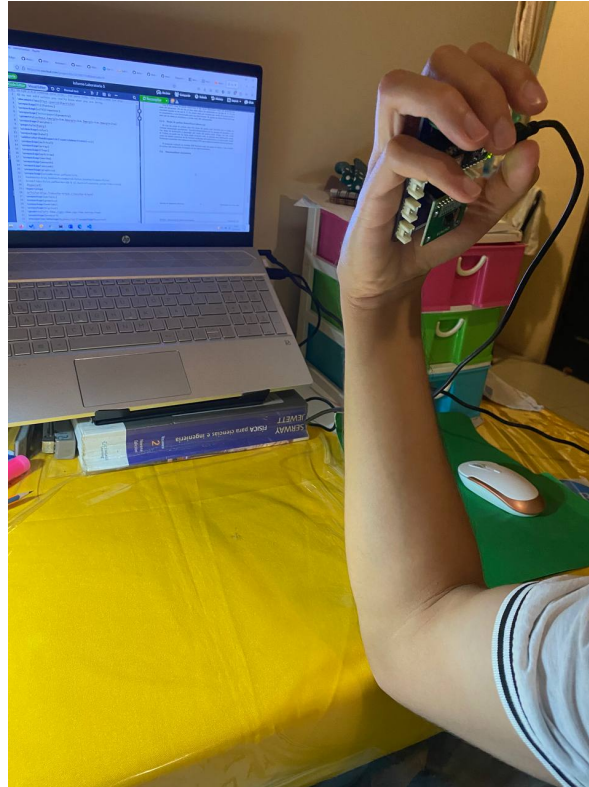


Figura 7: Posición final de la mano después de hacer la flexión.

En la figura 8, se encuentran los resultados en terminal luego de obtener el modelo.h a través del Google Colab discutido en sección 3.2. La figura 8 muestra los resultados en “porcentaje” (un 0.999942 es igual a decir 99.9942 %) sumamente cercanos al 100 %, por lo cual, el modelo es exacto y preciso. En la figura se aprecian 3 resultados, en los cuales todos muestran un 99.99 % de identificación para el movimiento flexión.

```
punch: 0.000010  
flex: 0.999942  
circulo: 0.000049  
  
punch: 0.000005  
flex: 0.999966  
circulo: 0.000029  
  
punch: 0.000006  
flex: 0.999975  
circulo: 0.000019
```

Figura 8: Resultados en terminal del movimiento flexión.

3.4.2. Movimiento punch (inglés) o golpe

A continuación, se analizará la toma de pruebas y análisis de resultados para el movimiento golpe. En la figura 9, se observa la posición inicial del golpe, la mano derecha se ubica cerca del pecho del usuario. Note la posición del arduino. Luego, un súbito golpe hacia el frente sucede para activar la toma de datos y poderla observar terminal.

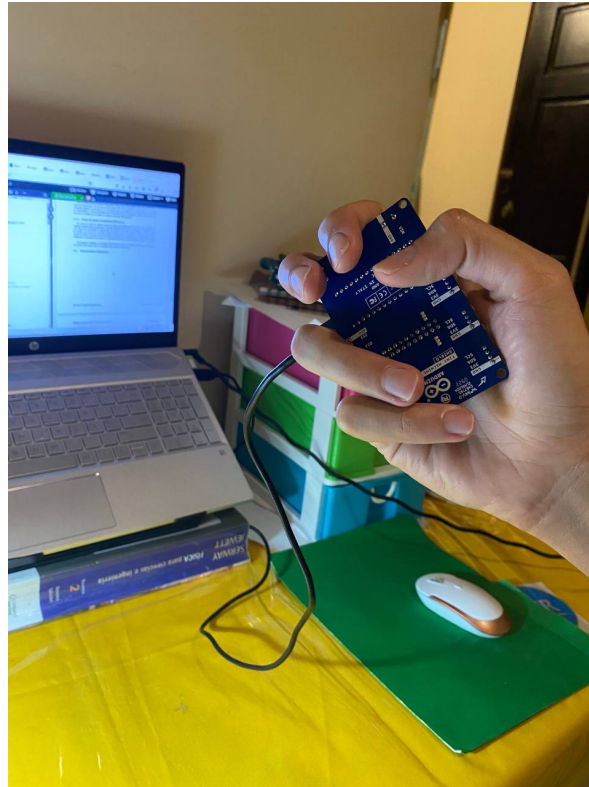


Figura 9: Posición inicial de la mano antes de hacer el golpe.

En la figura 10, se observa la posición final del golpe hacia al frente, muy cerca de la laptop. La toma de muestras para este movimiento fue de 10-11 aproximadamente. Justo después de la posición final, hay que mover la mano hacia la posición final de forma lenta para no activar accidentalmente una nueva toma de datos.

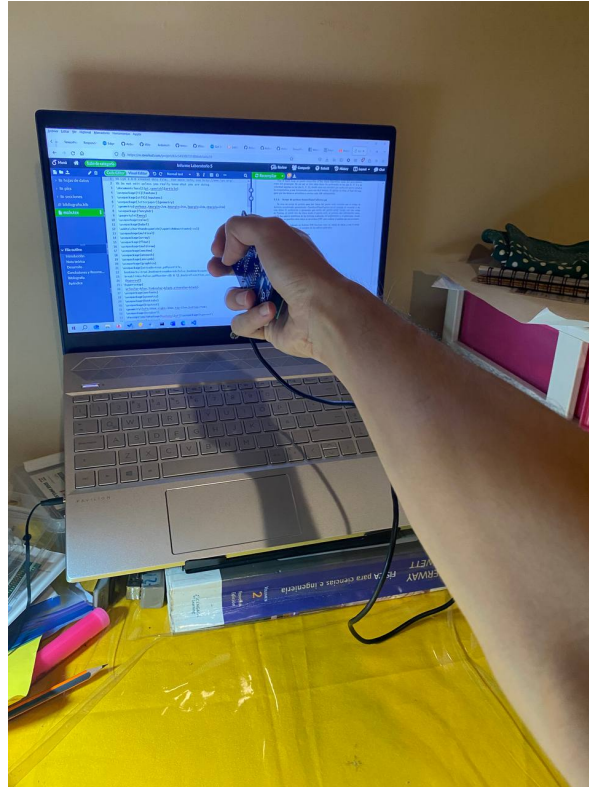
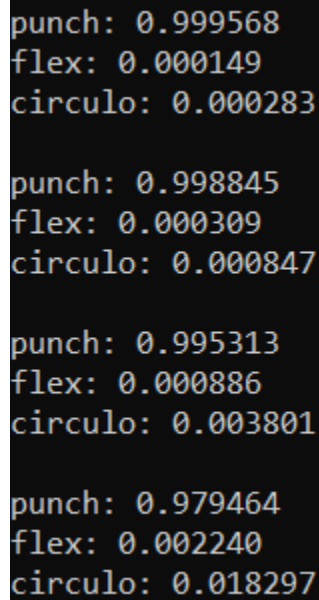


Figura 10: Posición final de la mano después de hacer el golpe.

En la figura 11, se observan 4 resultados del movimiento golpe luego de subir el sketch con el modelo.h obtenido del Google Colab. Al igual que el análisis al movimiento de flexión, aquí se obtienen eficiencias al reconocimiento del movimiento mayores al 97 %. El modelo.h ha sido hasta el momento un modelo excelente capaz de reconocer el tipo de movimiento en prueba.



```
punch: 0.999568  
flex: 0.000149  
circulo: 0.000283  
  
punch: 0.998845  
flex: 0.000309  
circulo: 0.000847  
  
punch: 0.995313  
flex: 0.000886  
circulo: 0.003801  
  
punch: 0.979464  
flex: 0.002240  
circulo: 0.018297
```

Figura 11: Resultados en terminal para el movimiento golpe.

3.4.3. Movimiento círculo

El tercer y último movimiento puesto a prueba fue el movimiento de hacer un círculo con la mano derecha en el sentido antihorario, tal y como se aprecia en la figura 12. En esta figura se muestra la posición inicial del arduino (ver forma en que se sujeta) y se muestra el recorrido a seguir a través de la flecha verde. El movimiento es súbito y rápido. Al igual que para los movimientos anteriores, se hizo un total de 10 pruebas para la creación del modelo con machine learning.

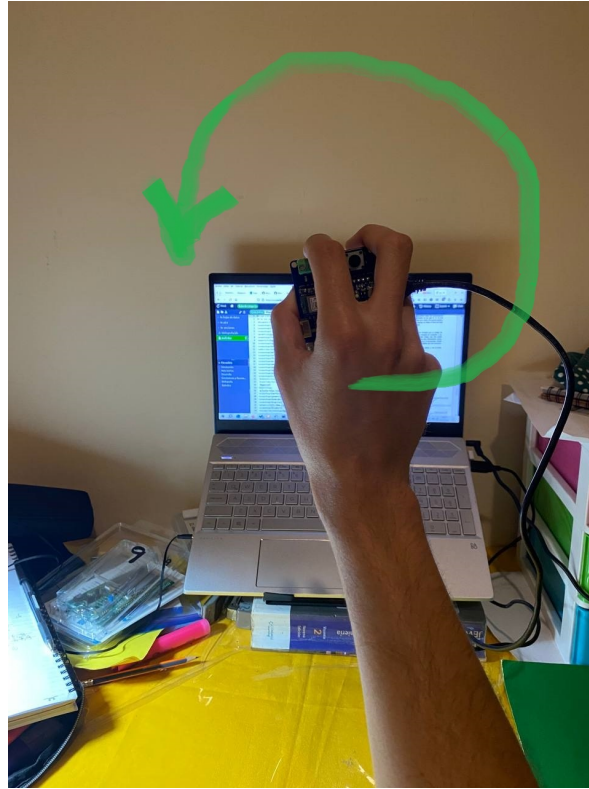


Figura 12: Movimiento del círculo en sentido antihorario.

En la figura 13, se presentan los resultados en terminal del movimiento círculo, o bien, el hacer un círculo con la mano. Se aprecian resultados de coincidencia del 99.82%, al menos en los 3 resultados en figura 13. Por lo tanto, el modelo.h entrenado a partir de las muestras y desarrollado a través del proceso de machine learning ha sido exitoso al reconocer los 3 movimientos discutidos en el presente laboratorio.

```
punch: 0.000293  
flex: 0.000424  
circulo: 0.999283  
  
punch: 0.001086  
flex: 0.000633  
circulo: 0.998281  
  
punch: 0.000647  
flex: 0.000856  
circulo: 0.998497
```

Figura 13: Resultados en terminal para el movimiento círculo.

4. Conclusiones y Recomendaciones

- Se logró capturar la información proveniente del giroscopio en la placa del Arduino, la cual pasó por el puerto serial y se guardó a través de un script de Python en un archivo .csv.
- Se utilizó de forma exitosa un script de Python en Google Colab para cargar las muestras, configurar la red neuronal, entrenar la red y la exportación del modelo.h; modelo obtenido con TensorFlow Lite.
- Se logró entrenar una red neuronal computacional con el 60 % de los datos de las muestras, se utilizó el 20 % de los mismos para lo que fue la validación y 20 % restante para la realización de pruebas.
- Se lograron coincidencias del casi 100 % en los movimientos detectados por el algoritmo que hizo uso del modelo de TensorFlow Lite exportado, es decir, todos los movimientos puestos a prueba con el modelo fueron acertados.
- Tomar el Arduino Nano BLE Sense Lite siempre en la misma posición y siguiendo de ejemplo las imágenes en sección desarrollo, para obtener eficiencias de detección de movimiento cercanas al 100 %.
- No utilizar Arduino IDE para subir el programa y el modelo que clasifica los movimientos, puesto que este desconecta el puerto serial y no permite que se suba el programa a la placa del Arduino.
- La librería Arduino_TensorFlowLite no funcionará en Windows si no mueve la carpeta tensor tensorflow dentro de la carpeta src dentro de la misma librería.
- Utilizar arduino-cli para compilar y cargar el programa en caso de que el Arduino IDE esté dando problema de desconexión del puerto serial a la hora de subir el programa a la placa de Arduino.
- Hacer uso de arduino-cli para evitar inconveniencias a la hora de monitorear los datos del puerto serial en terminal, esto porque si Arduino IDE está haciendo uso del puerto, en terminal no se podrán observar los valores.

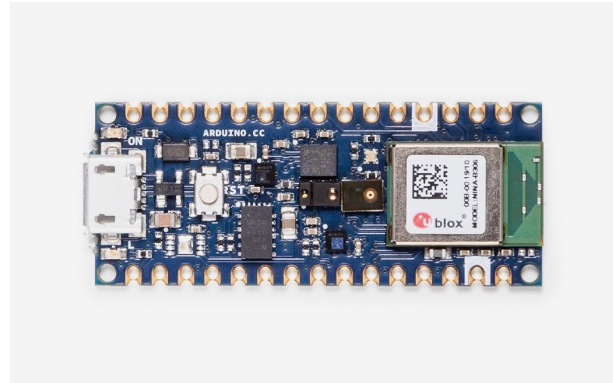
5. Bibliografía

Referencias

- [1] S. Mistry and D. Pajak, “Get started with machine learning on arduino,” 2024. [Online]. Available: <https://docs.arduino.cc/tutorials/nano-33-ble-sense/get-started-with-machine-learning/> 2.1.1, 3.1, 3.2, 1, 3, 4, 5, 6
- [2] TensorFlow, “Keras,” 2022. [Online]. Available: <https://www.tensorflow.org/guide/keras?hl=es-419> 4

6. Apéndice

Se incluyen las hojas de datos de todos los componentes pasivos y activos utilizados:



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

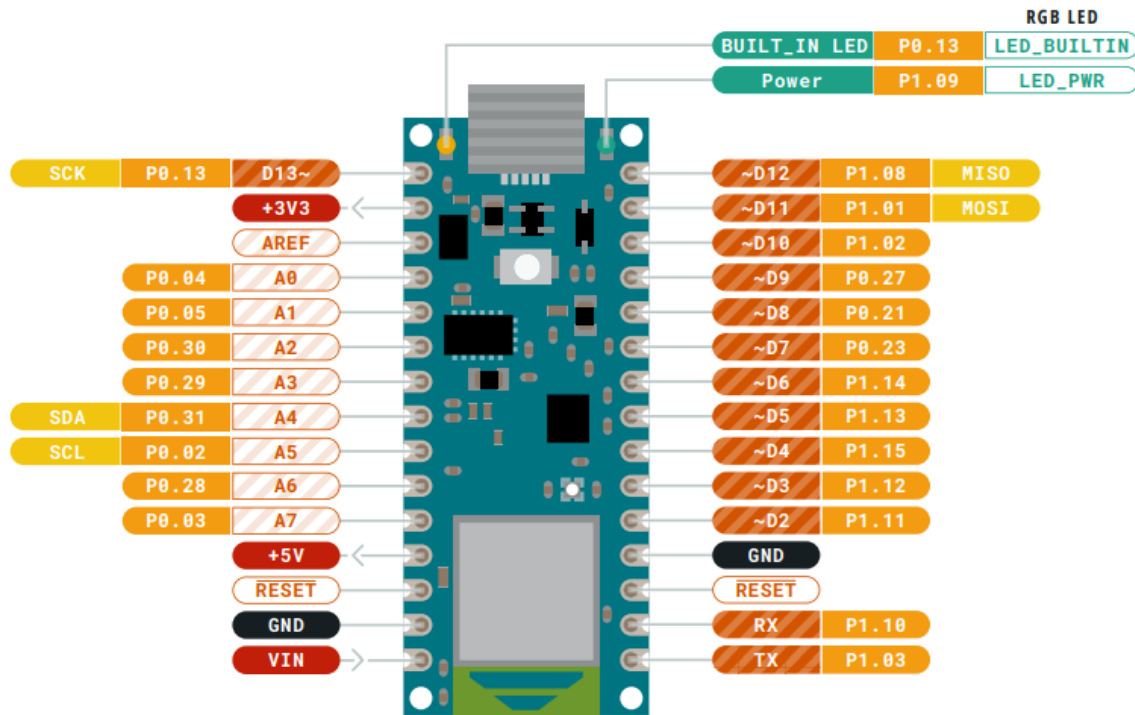


Features

- **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 ksp/s ADC
 - 128 bit AES/ECB/CCM/AAR co-processor
- **LSM9DS1** (9 axis IMU)
 - 3 acceleration channels, 3 angular rate channels, 3 magnetic field channels
 - $\pm 2/\pm 4/\pm 8/\pm 16$ g linear acceleration full scale
 - $\pm 4/\pm 8/\pm 12/\pm 16$ gauss magnetic full scale
 - $\pm 245/\pm 500/\pm 2000$ 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 rate/Interrupt 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: $\pm 3.5\%$ 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 dB SPL
 - 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



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

APDS-9960

Digital Proximity, Ambient Light, RGB and Gesture Sensor



Data Sheet



Description

The APDS-9960 device features advanced Gesture detection, Proximity detection, Digital Ambient Light Sense (ALS) and Color Sense (RGBC). The slim modular package, L 3.94 × W 2.36 × H 1.35 mm, incorporates an IR LED and factory calibrated LED driver for drop-in compatibility with existing footprints.

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 cancelation, dual 8-bit data converters, power saving inter-conversion delay, 32-dataset FIFO, and interrupt-driven I²C-bus 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.

Description continued on next page...

Applications

- Gesture Detection
- Color Sense
- Ambient Light Sensing
- Cell Phone Touch Screen Disable
- Mechanical Switch Replacement

Ordering Information

Part Number	Packaging	Quantity
APDS-9960	Tape & Reel	5000 per reel

Features

- Ambient Light and RGB Color Sensing, Proximity Sensing, and Gesture Detection in an Optical Module
- Ambient Light and RGB Color Sensing
 - UV and IR blocking filters
 - Programmable gain and integration time
 - Very high sensitivity – Ideally suited for operation behind dark glass
- Proximity Sensing
 - Trimmed to provide consistent reading
 - Ambient light rejection
 - Offset compensation
 - Programmable driver for IR LED current
 - Saturation indicator bit
- Complex Gesture Sensing
 - Four separate diodes sensitive to different directions
 - Ambient light rejection
 - Offset compensation
 - Programmable driver for IR LED current
 - 32 dataset storage FIFO
 - Interrupt driven I²C-bus communication
- I²C-bus Fast Mode Compatible Interface
 - Data Rates up to 400 kHz
 - Dedicated Interrupt Pin
- Small Package L 3.94 × W 2.36 × H 1.35 mm

Description (Cont.)

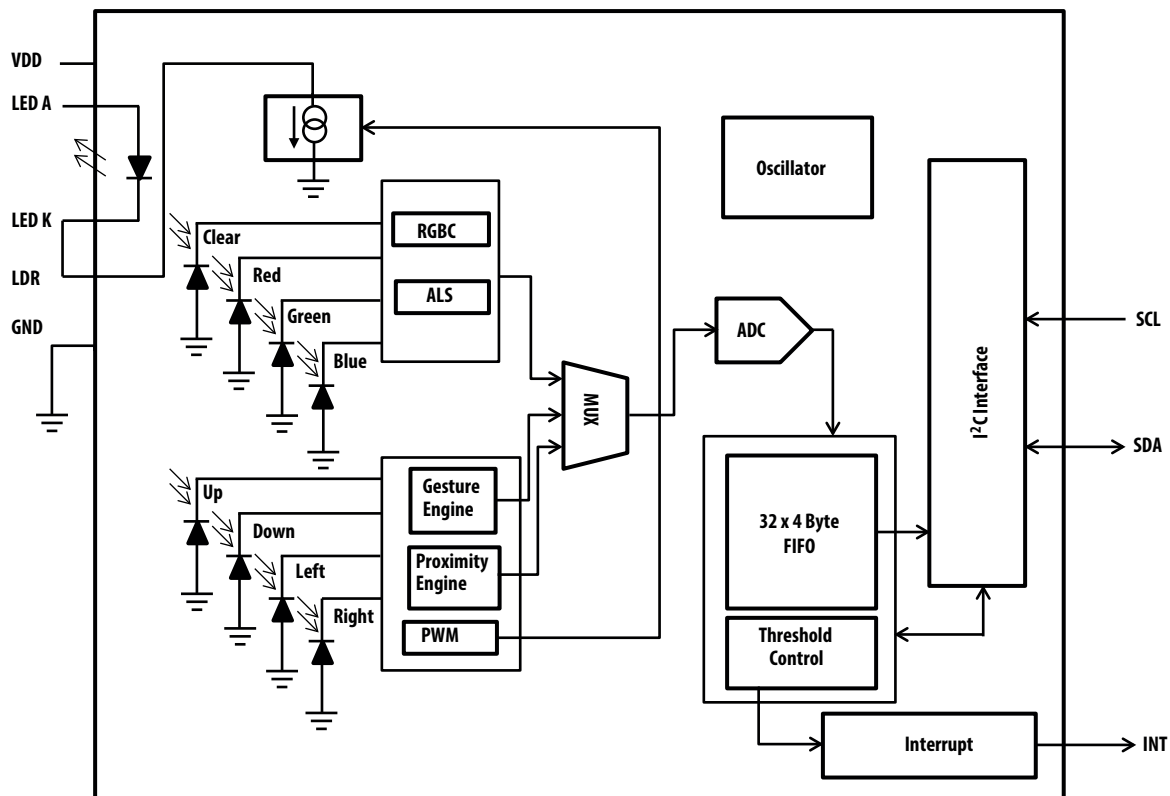
Proximity detection

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.

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 producing 16-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.

Functional Block Diagram



I/O Pins Configuration

Pin	Name	Type	Description
1	SDA	I/O	I ² C serial data I/O terminal - serial data I/O for I ² C-bus
2	INT	O	Interrupt - open drain (active low)
3	LDR		LED driver input for proximity IR LED, constant current source LED driver
4	LEDK		LED Cathode, connect to LDR pin when using internal LED driver circuit
5	LEDA		LED Anode, connect to V _{LEDA} on PCB
6	GND		Power supply ground. All voltages are referenced to GND
7	SCL	I	I ² C serial clock input terminal - clock signal for I ² C serial data
8	V _{DD}		Power supply voltage

Absolute Maximum Ratings over operating free-air temperature range (unless otherwise noted)*

Parameter	Symbol	Min	Max	Units	Conditions
Power supply voltage ^[1]	V _{DD}		3.8	V	
Input voltage range	V _{IN}	-0.5	3.8	V	
Output voltage range	V _{OUT}	-0.3	3.8	V	
Storage temperature range	T _{stg}	-40	85	°C	

* Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

Note 1. All voltages are with respect to GND.

Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Units
Operating ambient temperature	T _A	-30		85	°C
Power supply voltage	V _{DD}	2.4	3.0	3.6	V
Supply voltage accuracy, V _{DD} total error including transients		-3		+3	%
LED supply voltage	V _{LEDA}	3.0		4.5	V

Operating Characteristics, V_{DD} = 3 V, T_A = 25 °C (unless otherwise noted)

Parameter	Symbol	Min	Typ	Max	Units	Test Conditions
IDD supply current ^[1]	I _{DD}		200	250	μA	Active ALS state PON = AEN = 1, PEN = 0
			790			Proximity, LDR pulse ON, PPulse = 8 (I _{LDR} not included)
			790			Gesture, LDR pulse ON, GPulse = 8 (I _{LDR} not included)
			38			Wait state PON = 1, AEN = PEN = 0
			1.0	10.0		Sleep state ^[2]
V _{OL} INT, SDA output low voltage	V _{OL}	0		0.4	V	3 mA sink current
I _{LEAK} leakage current, SDA, SCL, INT pins	I _{LEAK}	-5		5	μA	
I _{LEAK} leakage current, LDR P\pin	I _{LEAK}	-10		10	μA	
SCL, SDA input high voltage, V _{IH}	V _{IH}	1.26		V _{DD}	V	
SCL, SDA input low voltage, V _{IL}	V _{IL}			0.54	V	

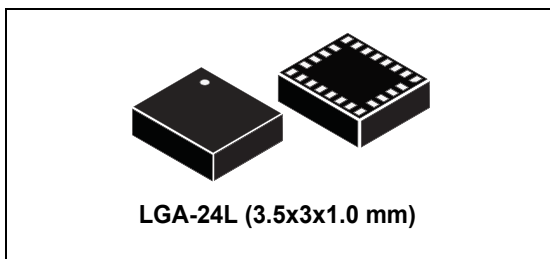
Notes

1. Values are shown at the VDD pin and do not include current through the IR LED.

2. Sleep state occurs when PON = 0 and I2C bus is idle. If Sleep state has been entered as the result of operational flow, SAI = 1, PON will be high.

iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer

Datasheet - production data



Features

- 3 acceleration channels, 3 angular rate channels, 3 magnetic field channels
- $\pm 2/\pm 4/\pm 8/\pm 16$ g linear acceleration full scale
- $\pm 4/\pm 8/\pm 12/\pm 16$ gauss magnetic full scale
- $\pm 245/\pm 500/\pm 2000$ dps angular rate full scale
- 16-bit data output
- SPI / I²C serial interfaces
- Analog supply voltage 1.9 V to 3.6 V
- “Always-on” eco power mode down to 1.9 mA
- Programmable interrupt generators
- Embedded temperature sensor
- Embedded FIFO
- Position and motion detection functions
- Click/double-click recognition
- Intelligent power saving for handheld devices
- ECOPACK[®], RoHS and “Green” compliant

Applications

- Indoor navigation
- Smart user interfaces
- Advanced gesture recognition
- Gaming and virtual reality input devices
- Display/map orientation and browsing

Description

The LSM9DS1 is a system-in-package featuring a 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor.

The LSM9DS1 has a linear acceleration full scale of $\pm 2g/\pm 4g/\pm 8/\pm 16$ g, a magnetic field full scale of $\pm 4/\pm 8/\pm 12/\pm 16$ gauss and an angular rate of $\pm 245/\pm 500/\pm 2000$ dps.

The LSM9DS1 includes an I²C serial bus interface supporting standard and fast mode (100 kHz and 400 kHz) and an SPI serial standard interface.

Magnetic, accelerometer and gyroscope sensing can be enabled or set in power-down mode separately for smart power management.

The LSM9DS1 is available in a plastic land grid array package (LGA) and it is guaranteed to operate over an extended temperature range from -40 °C to +85 °C.

Table 1. Device summary

Part number	Temperature range [°C]	Package	Packing
LSM9DS1	-40 to +85	LGA-24L	Tray
LSM9DS1TR	-40 to +85	LGA-24L	Tape and reel

1 Pin description

Figure 1. Pin connections

