

IE-0624 Laboratorio de Microcontroladores

Jorge Adán Mora Soto, B95222 Jafet David Gutiérrez Guevara, B73558
jorgeadan.mora@ucr.ac.cr jafet.gutierrez@ucr.ac.cr

1 de diciembre de 2022

Laboratorio 5

Arduino: GPIO, Giroscopio, comunicaciones, TinyML

Resumen

En el siguiente laboratorio se desarrolla un reconocedor de actividad humana el cual utiliza un kit Arduino Nano 33 BLE. Para esto se realiza un programa para el microcontrolador que capture la información del giroscopio y se envíe a la computadora por el puerto USB; esta información se procesa mediante un script de python que la guarde con etiqueta el tipo de movimiento que se efectúa y la ordene en un archivo CSV. Se registran 3 movimientos: «Up-Down», «Circle» y «Stationary». Con esta información muestreada se pretende realizar un modelo de red neuronal para cargar en el Arduino, de modo que este pueda identificar el movimiento realizado según la red cargada con datos históricos y frecuentes para determinado movimiento.

Sobre el modelo: se aspira a una red de dos capas con varias neuronas y 3 salidas (una para cada movimiento). Se entrena el modelo con el 60 % de los datos, se valida con el 20 % y se prueba con el último 20 % de los datos. Este modelo se procesa/exporta al microcontrolador mediante y gracias a la librería de TensorFlow. Los practicantes se apoyan en la plataforma de EdgeImpulse para perfeccionar el modelo. El objetivo se cumplió dentro de los parámetros evaluativos. Se exporta un modelo que detecta los 3 movimientos. Con respecto a la red neuronal, no se cumplió perfectamente el objetivo con el movimiento "Stationary" pues este se predice parcialmente; pero este conocimiento de redes neuronales no corresponde a criterios evaluativos del curso, por lo que se concluye una práctica exitosa.

Link del Proyecto:  <https://github.com/Jams1001/IE0624/tree/main/L5>

ID del último commit de interés: [75f7e2e](#)

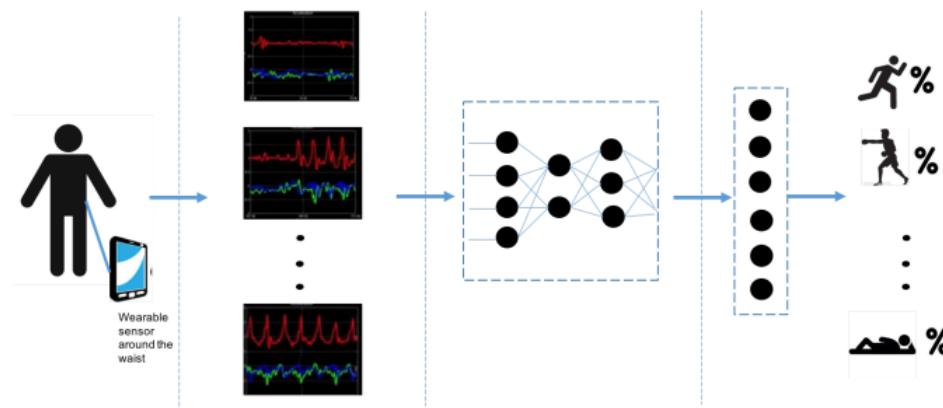


Figura 1: HAR.

1. Nota Teórica

1.1. Información general del MCU

El nRF52840 es el miembro más avanzado de la serie nRF52. Cumple con los desafíos de las aplicaciones sofisticadas que necesitan concurrencia de protocolos y un conjunto rico y variado de periféricos y funciones. Ofrece una generosa disponibilidad de memoria tanto para Flash como para RAM, que son requisitos previos para aplicaciones tan exigentes[1]. Se trata de un procesador ARM Cortex-M4 con unidad de coma flotante (FPU) que tiene un conjunto de instrucciones de 32 bits que implementa un superconjunto de instrucciones de 16 y 32 bits para maximizar la densidad del código y actuación [1].

Tiene numerosos periféricos e interfaces digitales, como SPI y QSPI de alta velocidad para conectarse a pantallas y flashes externos, PDM e I2S para micrófonos y audio digitales, y un dispositivo USB de alta velocidad para transferencia de datos y fuente de alimentación para recargar la batería[1]. A continuación en la tabla 1 se presenta las características eléctricas, en la figura 5 se presenta el diagrama de pines de la placa en donde se integra el MCU a utilizar, y en la figura 3 se presenta el diagrama de bloques.

MCU	nRF52840
Voltaje de operación	3.3V
Voltaje de entrada	21V
DC corriente por I/O Pin	15 mA
Frecuencia de reloj	64MHz
Memoria Flash	1MB (nRF52840)
SRAM	256KB (nRF52840)
EEPROM	none
I/O Digital Pins	14
PWM Pins	14 (Todos los pines digitales)
UART	1
SPI	1
I2C	1
Analog Input Pins	8 (ADC 12 bit 200 ksamples)
Analog Output Pins	Only through PWM (no DAC)
External Interrupts	all digital pins
LED_BUILTIN	13
USB	Native in the nRF52840 Processor

Tabla 1: Características generales y eléctricas

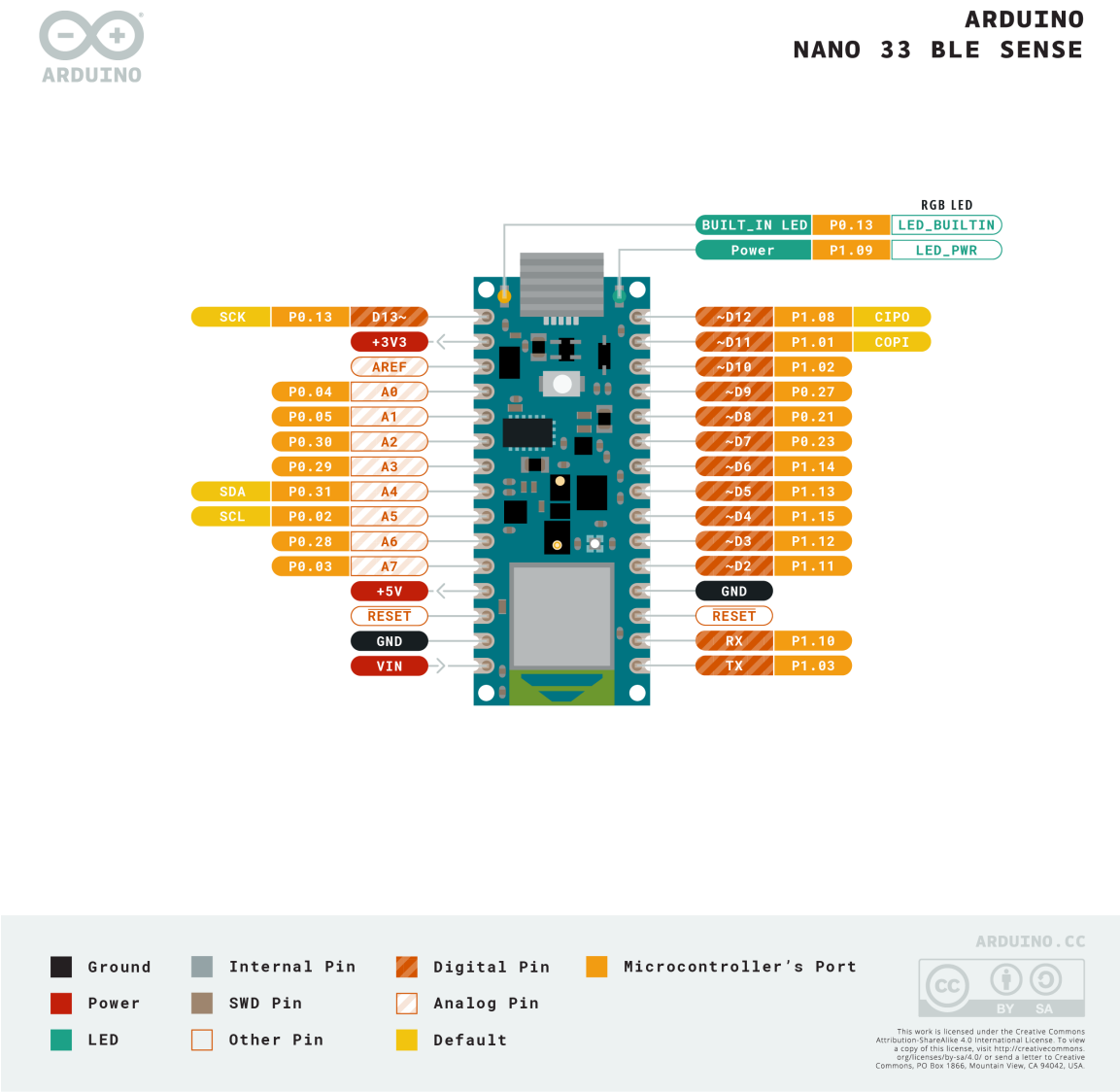


Figura 2: Diagrama de pines [2]

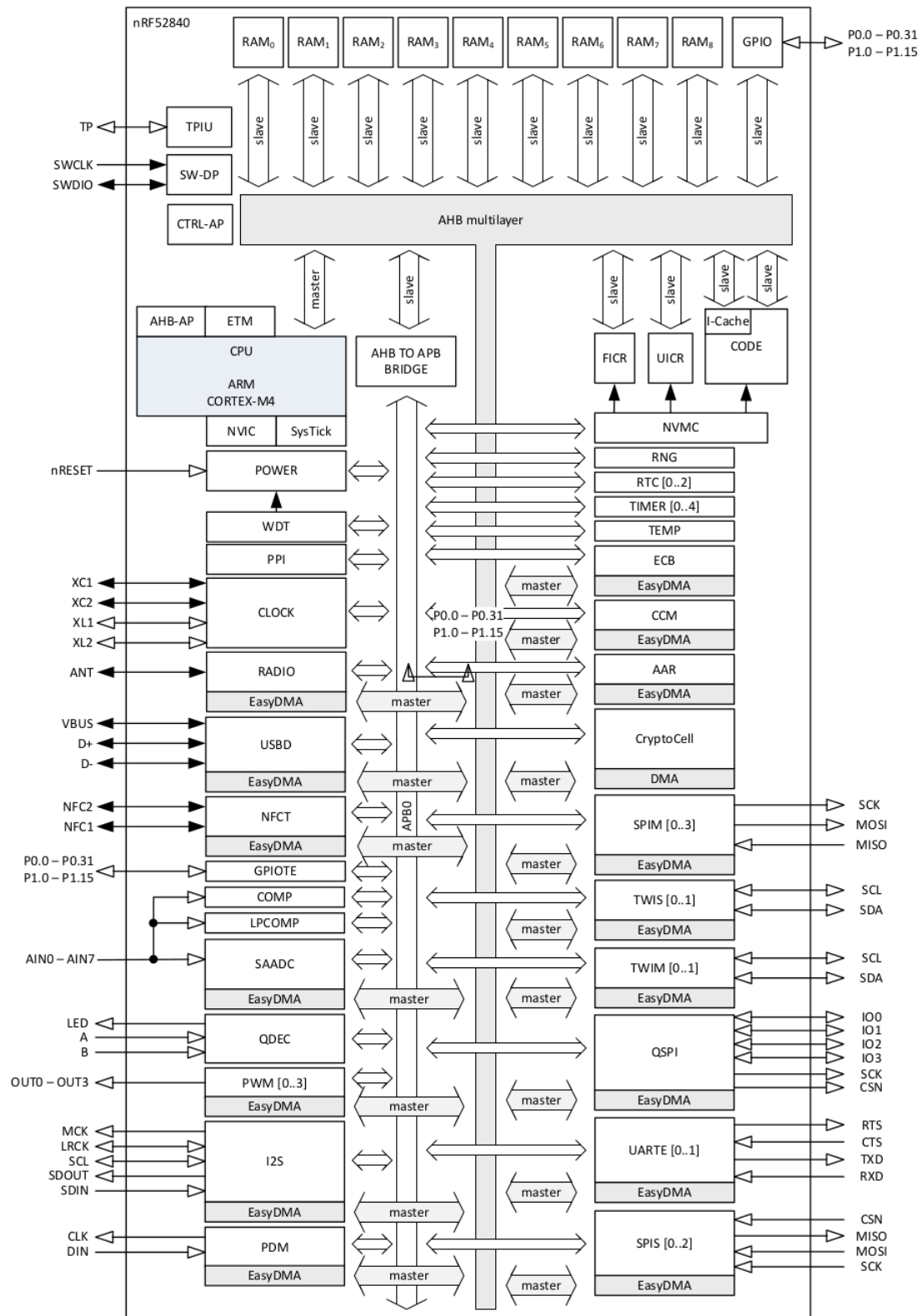


Figure 1: Block diagram

Figura 3: Diagrama de bloques [1]

1.2. Sobre el Nano 33 BLE Sense

El Arduino Nano 33 BLE Sense es una excelente opción para cualquier principiante, fabricante o profesional para comenzar con el aprendizaje automático integrado. Se basa en el microcontrolador nRF52840 y se ejecuta en el sistema operativo Arm Mbed. El Nano 33 BLE Sense no solo cuenta con la posibilidad de conectarse a través de Bluetooth Low Energy, sino que también viene equipado con sensores para detectar color, proximidad, movimiento, temperatura, humedad, audio y más [2]. Sobre sus periféricos y principales características:

- **u-blox NINA-B306:** Módulo Bluetooth 5 de bajo consumo de u-blox, con antena interna [2].
- **IMU for Motion Detection:** La unidad de medición inercial LSM9DS1 cuenta con un acelerómetro, un giroscopio y un magnetómetro 3D y le permite detectar la orientación, el movimiento o las vibraciones en su proyecto [2]. Se trata de un sistema en paquete que presenta un sensor de aceleración lineal digital 3D, un sensor digital 3D, un sensor de velocidad angular y un sensor magnético digital 3D [3].
 - 3 canales de aceleración, 3 de velocidad angular canales, 3 canales de campo magnético [3].
 - Interfaces seriales SPI / I^2C [3].
 - 16-bit de data-output [3].
- **Omnidirectional Digital Microphone:** El micrófono MP34DT05 permite capturar y analizar el sonido en tiempo real y puede usarse para crear una interfaz de voz para su proyecto [2].
- **Proximity and Gesture Detection:** El chip APDS-9960 permite medir la proximidad digital y la luz ambiental, así como detectar colores y gestos RGB [2].
- **Barometric Pressure Sensor:** El LPS22HB detecta la presión barométrica y permite una salida de datos de presión de 24 bits entre 260 y 1260 hPa. Estos datos también se pueden procesar para calcular la altura sobre el nivel del mar de la ubicación actual [2].
- **Temperature and Humidity Detection:** The HTS221 capacitive digital sensor measures relative humidity and temperature. It has a temperature accuracy of ± 0.5 °C (between 15-40 °C) and is thereby perfectly suited to detect ambient temperature [2].

1.3. Funciones de interés soportadas

- **pinMode():** Configura el pin especificado para que se comporte como una entrada o una salida [4].
- **analogRead():** Lee el valor del pin analógico especificado. Las placas Arduino contienen un convertidor analógico a digital multicanal de 10 bits. Esto significa que mapeará los voltajes de entrada entre 0 y el voltaje de funcionamiento (5 V o 3,3 V) en valores enteros entre 0 y 1023. En un Arduino UNO, por ejemplo, esto produce una resolución entre lecturas de: 5 voltios / 1024 unidades o 0,0049 voltios (4,9 mV) por unidad [4].
- **map():** Vuelve a asignar un número de un rango a otro. Es decir, un valor de fromLow se asignaría a toLow, un valor de fromHigh a toHigh, valores intermedios a valores intermedios, etc. Reasigna la escala analógica a la escala digital en la resolución disponible [4].

- **delay():** Pausa el programa por la cantidad de tiempo (en milisegundos) especificado como parámetro. (Hay 1000 milisegundos en un segundo) [4].
- **analogWrite():** Escribe un valor analógico (onda PWM) en un pin. Se puede usar para encender un LED con diferentes brillos o impulsar un motor a varias velocidades. Después de una llamada a `analogWrite()`, el pin generará una onda rectangular constante del ciclo de trabajo especificado hasta la próxima llamada a `analogWrite()` (o una llamada a `digitalRead()` o `digitalWrite()`) en el mismo pin. Recibe como parámetros `analogWrite(pin, value)`, en donde `pin` corresponde el pin de Arduino para escribir y `value` al ciclo de trabajo que va entre 0 (siempre apagado) y 255 (siempre encendido) [4].

1.4. Diseño

Nuevamente para este proyecto no existe un diseño electrónico, fue suficiente con la utilización del Nano 33 BLE Sense. El fin del laboratorio es usar exitosamente los periféricos de la misma y es por esto que se detallaron en las secciones anteriores.

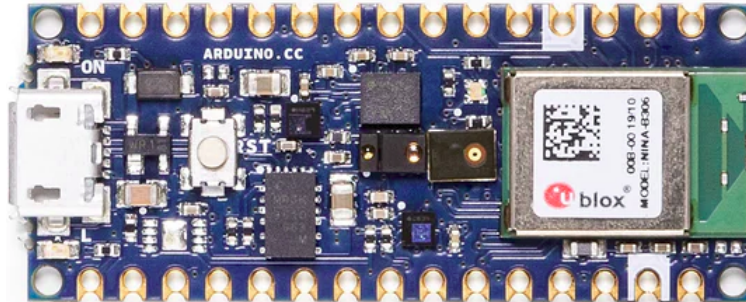


Figura 4: Arduino Nano 33 BLE Sense [2].

1.5. Componentes complementarios

Para esta práctica únicamente se trabaja con la placa Nano 33 BLE Sense. Sus abundantes periféricos cumplen con todo lo necesario para la realización, por lo que únicamente se debe conectar a la alimentación y comunicación con la computadora. El precio de esta placa ronda los 40.5USD.

1.6. Conceptos

1.6.1. AI

La inteligencia artificial (o por sus siglas en inglés «AI»), la capacidad de una computadora digital o un robot controlado por computadora para realizar tareas comúnmente asociadas con seres inteligentes. El término se aplica con frecuencia al proyecto de desarrollar sistemas dotados de los procesos intelectuales característicos de los humanos, como la capacidad de razonar, descubrir significado, generalizar o aprender de experiencias pasadas [5].

1.6.2. Machine learning en microcontroladores

El aprendizaje automático, mejor conocido como *machine learning*, es una rama de la inteligencia artificial dirigida al desarrollo de herramientas que permiten que las máquinas aprendan sin estar explícitamente programadas para ello. Con dichas herramientas, se puede hacer que distintos dispositivos sean capaces identificar patrones entre un conjunto de datos, y a partir de ellos, realizar predicciones. El *machine learning* puede utilizarse para crear tecnologías inteligentes que faciliten la vida de sus usuarios, como Google Assistant, para dar un ejemplo [6].

Sin embargo, la aplicación del *machine learning* a menudo requiere de una gran cantidad de recursos, que pueden incluir un potente servidor en la nube o un computador con una considerable capacidad de procesamiento. Afortunadamente, ahora es posible ejecutar la inferencia de aprendizaje automático en hardware diminuto y de baja potencia, como los microcontroladores. Al traer el *machine learning* a estos dispositivos se puede potenciar la inteligencia de miles de millones de equipos que se utilizan cotidianamente, y sin depender de hardware costoso o conexiones robustas a Internet [6].

Una de las herramientas más populares para la implementación de *machine learning* es TensorFlow. Este es un framework de aprendizaje automático, de código abierto y de Google para entrenar y ejecutar modelos de reconocimiento de patrones. Como parte de los esfuerzos de TensorFlow, Google también desarrolló una versión optimizada, destinada a ejecutar modelos de TensorFlow en microcontroladores. Dicha versión tiene el nombre de TensorFlow Lite For Microcontrollers. Este se adhiere a las restricciones requeridas en entornos embebidos, es decir, tiene un tamaño binario pequeño, no requiere el soporte de ningún sistema operativo, ninguna librería estándar de C o C++, o la asignación memoria dinámica [6].

1.6.3. Red Neuronal artificial

Consiste en una metodología de inteligencia artificial que enseña a las computadoras a procesar datos de una manera inspirada en el cerebro humano, con patrones o modelos previamente cargados de esos mismos eventos. Es un tipo de proceso de aprendizaje automático, llamado aprendizaje profundo, que utiliza nodos o neuronas interconectados en una estructura en capas que se asemeja al cerebro humano. Crea un sistema adaptativo que las computadoras usan para aprender de sus errores y mejorar continuamente. De esta forma las redes neuronales artificiales intentan resolver problemas complicados, como resumir documentos o reconocer rostros, reconocer movimientos, y todo esto en búsqueda de una mayor precisión para conquistar los límites de las capacidades humanas [7].

2. Desarrollo / Análisis de Resultados

2.1. Análisis de SW

2.1.1. recoder.py

Este archivo entregado en la ruta `/IE0624/L5/src`, únicamente se encarga de recibir la información serial enviada por el microcontrolador. Recibe 6 variables: `header = ['aX', 'aY', 'aZ', 'gX', 'gY', 'gZ']` correspondientes a las 3 componentes de la aceleración y el giroscopio del LSM9DS1. Con esta información escribe los archivos con los datos para el movimiento específico que se está haciendo en ese momento en formato de `csv`.

2.1.2. Modelo

A continuación se explica las secciones con las que cuenta el archivo que genera el modelo. Para esto referencie el archivo entregado como `arduino_tinymml_workshop.ipynb`, el cual corresponde a un notebook de google colab.

Esta sección no es de principal interés para los fines evaluativos del laboratorio. Consiste primeramente en graficar los datos del `csv` procesado por el `recoder.py`. Este primer punto únicamente es con fines ilustrativos.

Seguidamente se debe entrenar la red neuronal del modelo con las muestras por cada uno de los 3 gestos deseados a predecir. De modo que se normalizan los datos de de 0 a 1 y se definen las entradas del modelo.

Luego se debe dividir las muestras para generar, validar y entrenar el modelo. Esta es la sección de Aleatorizar y dividir los pares de entrada y salida para el entrenamiento. dividiendo aleatoriamente los pares de entrada y salida en conjuntos de datos: 60 % para entrenamiento, 20 % para validación (para medir qué tan bien se está desempeñando el modelo durante el entrenamiento) y 20 % para prueba (para probar el modelo después del entrenamiento).

El siguiente punto importante es construir y entrenar la red neuronal, lo cual se hace mediante un modelo de TensorFlow con la API de alto nivel de Keras.

La siguiente sección de interés es la validación del desempeño del modelo mediante la gráfica de la pérdida (definida mediante el `.error cuadrático medio` como la función de pérdida) para ver cuándo el modelo deja de mejorar. Esto se realiza de diferentes formas y entre esas el cálculo del error absoluto medio es otra métrica para juzgar el rendimiento del modelo.

Seguidamente viene la validación en donde se ponen los datos de prueba en el modelo y trazar las predicciones.

Una vez validado, se convierte el modelo al formato TensorFlow Lite sin cuantificación lo cual finaliza el protagonista del generador del modelo.

Para poder convertir el modelo anterior en una librería de arduino se debe ejecutar la siguiente sintaxis:

```
!echo "const unsigned char model[] = {" > /content/model.h
!cat gesture_model.tflite | xxd -i      >> /content/model.h
!echo "};"                             >> /content/model.h
```

A pesar de que esto no es de carácter principal en la evaluación, el generador del modelo cumple su función pues proporciona un modelo funcional para cargar en el microcontrolador.

2.1.3. Firmwares

`data_senser.ino`

El primer sketch, llamado `data_senser.ino`, fue un firmware que se desarrolló inicialmente para acceder a las lecturas del giroscopio y de aceleración que registra el Arduino Nano 33 BLE. Además de acceder a dichos registros, este firmware se encarga de enviar los datos leídos en cada ciclo al puerto serial, para que estos puedan ser recopilados desplegados y guardados por el script `recorder.py`.

`imu.ino`

El segundo sketch, llamado `imu.ino`, es el firmware en el cual se implementa el modelo generado para la detección de movimientos. En este sketch primero se incluye la librería `Arduino_LSM9DS1`, la cual permite leer los valores del acelerómetro, magnetómetro y giroscopio de la IMU LSM9DS1 en el Arduino Nano 33 BLE Sense. Después se incluyen los encabezados asociados a todas las funciones de TensorFlow Lite que se van a usar más adelante. El último archivo que se incluye es `model.h`, el cual contiene el modelo entrenado de la red neuronal que se generó anteriormente. Posteriormente, se crea un array para asignar el índice de gestos a un nombre, como se muestra a continuación:

```
const char* GESTURES[] = {  
    "circle",  
    "up-down",  
    "stationary",  
};
```

Si el Arduino detecta un movimiento válido accederá a uno de los elementos dentro del array `GESTURES` para escribirlo al puerto serial.

Dentro de la configuración inicial se inicializa el puerto serial, así como también el IMU. También se obtiene la representación TFL de la matriz de bytes del modelo, se crea un interprete para correr el modelo, se asigna memoria para sus tensors de entrada y salida y se obtienen punteros para estos últimos dos.

Por otro lado, en el lazo de ejecución se tienen variables de tipo flotante para almacenar los datos leídos de aceleración y el giroscopio. Después se espera a que la cantidad de muestras recolectadas sea la suficiente como para determinar que se produjo un movimiento significativo. Luego, se normalizan los datos recopilados de la IMU entre 0 y 1, y los resultados son almacenados en el tensor de entrada del modelo. Por último, se ejecuta la inferencia y se escribe en el puerto serial el movimiento detectado por el modelo, como se muestra a continuación:

```
for (int i = 0; i < NUM_GESTURES; i++) {  
    Serial.print(GESTURES[i]);  
    Serial.print(": ");  
}
```

2.2. Análisis de HW

Nuevamente, no existen parámetros electrónicos por analizar pues la electrónica únicamente se basa en el Nano 33 BLE Sense; el cual tiene la calidad de un producto de la compañía Arduino. De igual forma, se presenta una imagen con una demostración de su funcionamiento en la figura 5.

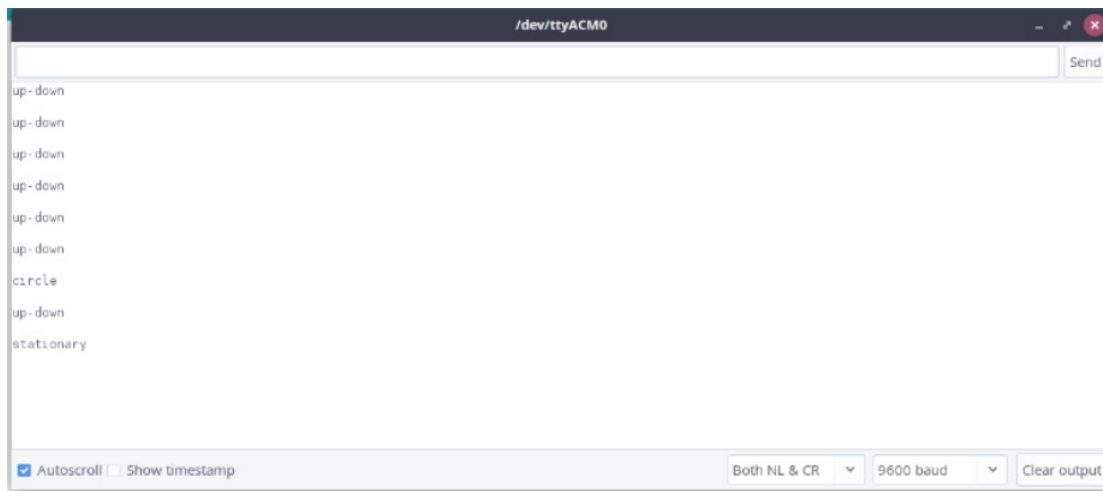


Figura 5: Salida serial del arduino posterior a la carga del firmware .

3. Conclusiones y Recomendaciones

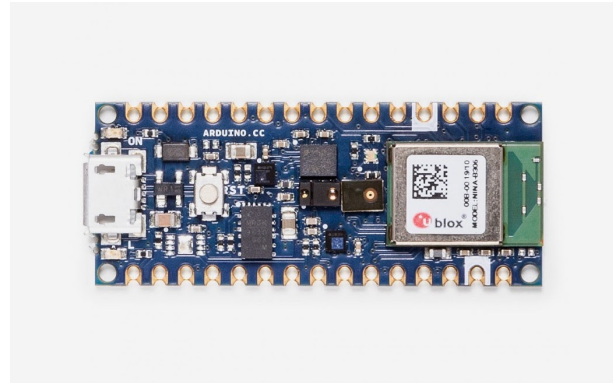
- Se concluye que el mcu integrado en el arduino nano 33 BLE Sense es una poderosa herramienta para implementar modelos de inteligencia artificial y redes neuronales. Sus periféricos y funciones adicionales como que se puede programar con el lenguaje de programación Python a través de OpenMV IDE, son pluses que hacen una gran herramienta para proyectos serios y de altura para proyectos con AI. Sumado a que el IDE que utiliza es de mayor comodidad con respecto a su antiguo predecesor.
- Se recomienda la utilización de Google Colab Notebooks para agilizar la generación del modelo con el fin de aprovechar los recursos existentes y no consumir la capacidad de los recursos de los practicantes. Esto debido a que en primera instancia se tuvo problemas por la tardanza en la generación del modelo, la capacidad de memoria de las computadoras locales, entre otras cosas.
- Se recomienda la futura implementación del mismo proyecto pero implementado con un STM32 para obtener el panorama completo y comparar las dificultades en la generación de modelos para diferentes microcontroladores.

Bibliografía

- [1] N. Semiconductor, “nRF52840”, Nordic Semiconductor ASA, Datasheet 4413417v1.1, feb. de 2019.
- [2] Arduino, “Arduino® Nano 33 BLE Sense”, Arduino, Product Reference Manual SKU: ABX00031, nov. de 2022.
- [3] life.augmented - STMicroelectronics, “LSM9DS1”, life.augmented - STMicroelectronics, Datasheet DocID025715 Rev 3, nov. de 2015.
- [4] Arduino. “Language Reference”. Visitado en noviembre 29 de 2022. (2022), dirección: <https://www.arduino.cc/reference/en/>.

- [5] B. Copeland. “artificial intelligence”. Visitado en noviembre 29 de 2022. (nov. de 2011), dirección: <https://www.britannica.com/technology/artificial-intelligence>.
- [6] M. Natraj. “AI Magic Wand with TensorFlow Lite for Microcontrollers and Arduino”. Visitado en noviembre 23 de 2022. (mar. de 2022), dirección: <https://codelabs.developers.google.com/magicwand#0>.
- [7] AWS. “What Is A Neural Network?” Visitado en noviembre 29 de 2022. (nov. de 2022), dirección: <https://aws.amazon.com/what-is/neural-network/>.

4. Apéndices



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



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)

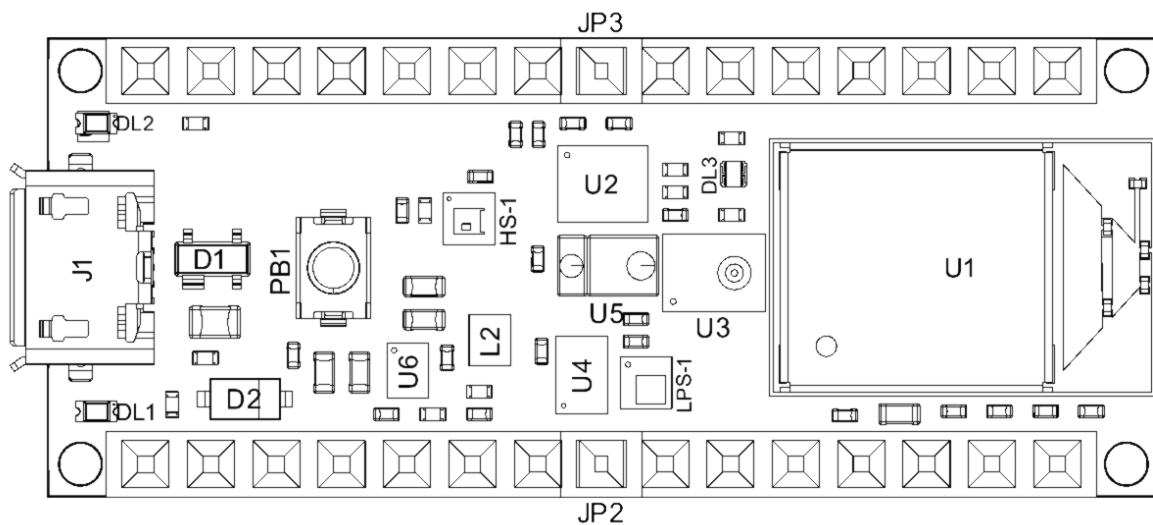
1.2 Power Consumption

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:



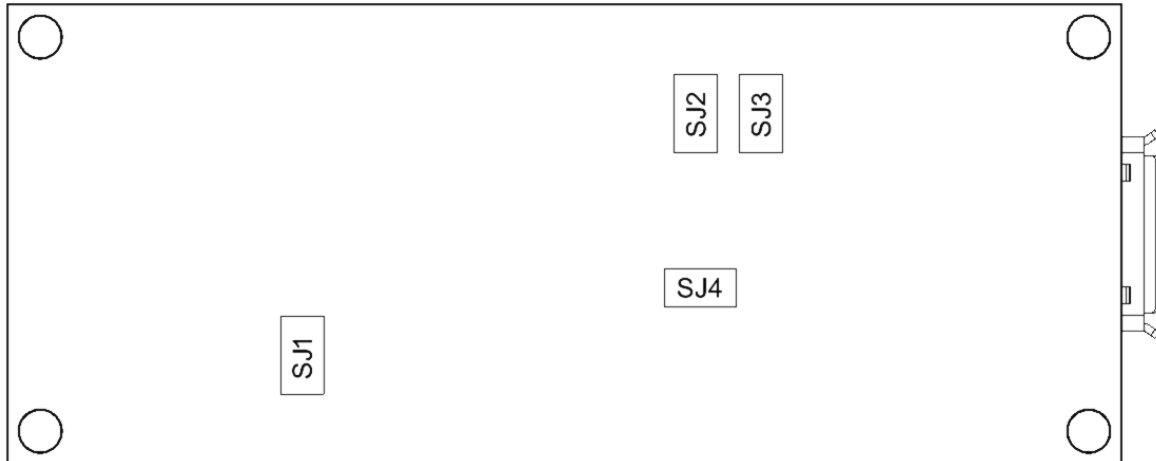
Board topology top

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 I²C 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 I²C 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 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.

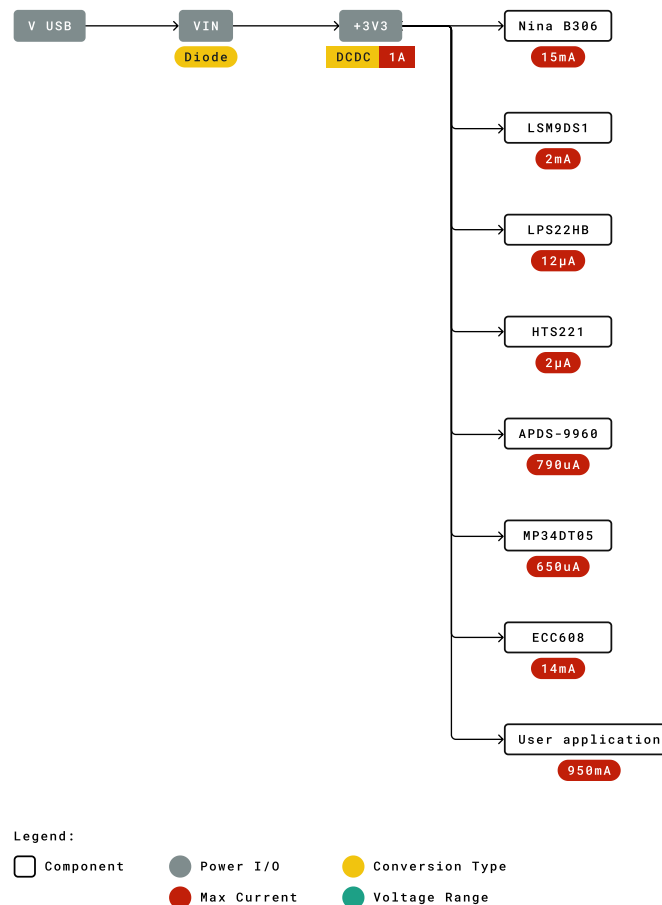
2.8 Digital Microphone

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.

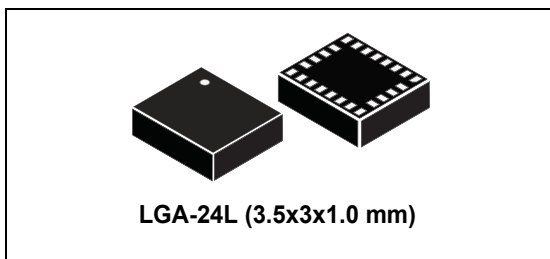
3.6 Board Recovery

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

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

8.14	INT_CFG_M (30h)	67
8.15	INT_SRC_M (31h)	67
8.16	INT_THS_L(32h), INT_THS_H(33h)	68
9	Package information	69
9.1	Soldering information	69
9.2	LGA package information	69
10	Revision history	71

List of tables

Table 1.	Device summary	1
Table 2.	Pin description	11
Table 3.	Sensor characteristics	12
Table 4.	Electrical characteristics	13
Table 5.	Temperature sensor characteristics	14
Table 6.	SPI slave timing values	15
Table 7.	I ² C slave timing values	16
Table 8.	Absolute maximum ratings	17
Table 9.	Gyroscope operating modes	20
Table 10.	Operating mode current consumption	20
Table 11.	Accelerometer turn-on time	20
Table 12.	Gyroscope turn-on time	21
Table 13.	Serial interface pin description	28
Table 14.	I ² C terminology	28
Table 15.	Transfer when master is writing one byte to slave	29
Table 16.	Transfer when master is writing multiple bytes to slave	29
Table 17.	Transfer when master is receiving (reading) one byte of data from slave	29
Table 18.	Transfer when master is receiving (reading) multiple bytes of data from slave	29
Table 19.	Accelerometer and gyroscope SAD+Read/Write patterns	30
Table 20.	Magnetic sensor SAD+Read/Write patterns	30
Table 21.	Accelerometer and gyroscope register address map	38
Table 22.	Magnetic sensor register address map	40
Table 23.	ACT_THS register	41
Table 24.	ACT_THS register description	41
Table 25.	ACT_DUR register	41
Table 26.	ACT_DUR register description	41
Table 27.	INT_GEN_CFG_XL register	41
Table 28.	INT_GEN_CFG_XL register description	42
Table 29.	INT_GEN_THS_X_XL register	42
Table 30.	INT_GEN_THS_X_XL register description	42
Table 31.	INT_GEN_THS_Y_XL register	42
Table 32.	INT_GEN_THS_Y_XL register description	42
Table 33.	INT_GEN_THS_Z_XL register	43
Table 34.	INT_GEN_THS_Z_XL register description	43
Table 35.	INT_GEN_DUR_XL register	43
Table 36.	INT_GEN_DUR_XL register description	43
Table 37.	REFERENCE_G register	43
Table 38.	REFERENCE_G register description	43
Table 39.	INT1_CTRL register	43
Table 40.	INT1_CTRL register description	44
Table 41.	INT2_CTRL register	44
Table 42.	INT2_CTRL register description	44
Table 43.	WHO_AM_I register	45
Table 44.	CTRL_REG1_G register	45
Table 45.	CTRL_REG1_G register description	45
Table 46.	ODR and BW configuration setting (after LPF1)	45
Table 47.	ODR and BW configuration setting (after LPF2)	46
Table 48.	CTRL_REG2_G register	47

Table 49.	CTRL_REG2_G register description	47
Table 50.	CTRL_REG3_G register	47
Table 51.	CTRL_REG3_G register description	47
Table 52.	Gyroscope high-pass filter cutoff frequency configuration [Hz].	48
Table 53.	ORIENT_CFG_G register	48
Table 54.	ORIENT_CFG_G register description	48
Table 55.	INT_GEN_SRC_G register	48
Table 56.	INT_GEN_SRC_G register description	49
Table 57.	OUT_TEMP_L register	49
Table 58.	OUT_TEMP_H register	49
Table 59.	OUT_TEMP register description	49
Table 60.	STATUS_REG register	49
Table 61.	STATUS_REG register description	50
Table 62.	CTRL_REG4 register	50
Table 63.	CTRL_REG4 register description	51
Table 64.	CTRL_REG5_XL register	51
Table 65.	CTRL_REG5_XL register description	51
Table 66.	CTRL_REG6_XL register	51
Table 67.	CTRL_REG6_XL register description	52
Table 68.	ODR register setting (accelerometer only mode)	52
Table 69.	CTRL_REG7_XL register	52
Table 70.	CTRL_REG7_XL register description	53
Table 71.	Low pass cutoff frequency in high resolution mode (HR = 1)	53
Table 72.	CTRL_REG8 register	53
Table 73.	CTRL_REG8 register description	53
Table 74.	CTRL_REG9 register	54
Table 75.	CTRL_REG9 register description	54
Table 76.	CTRL_REG10 register	54
Table 77.	CTRL_REG10 register description	54
Table 78.	INT_GEN_SRC_XL register	54
Table 79.	INT_GEN_SRC_XL register description	55
Table 80.	STATUS_REG register	55
Table 81.	STATUS_REG register description	55
Table 82.	FIFO_CTRL register	56
Table 83.	FIFO_CTRL register description	56
Table 84.	FIFO mode selection	56
Table 85.	FIFO_SRC register	57
Table 86.	FIFO_SRC register description	57
Table 87.	FIFO_SRC example: OVR/FSS details	57
Table 88.	INT_GEN_CFG_G register	57
Table 89.	INT_GEN_CFG_G register description	58
Table 90.	INT_GEN_THS_XH_G register	58
Table 91.	INT_GEN_THS_XL_G register	58
Table 92.	INT_GEN_THS_X_G register description	58
Table 93.	INT_GEN_THS_YH_G register	59
Table 94.	INT_GEN_THS_YL_G register	59
Table 95.	INT_GEN_THS_Y_G register description	59
Table 96.	INT_GEN_THS_ZH_G register	59
Table 97.	INT_GEN_THS_ZL_G register	59
Table 98.	INT_GEN_THS_Z_G register description	59
Table 99.	INT_GEN_DUR_G register	59
Table 100.	INT_GEN_DUR_G register description	60

Table 101.	OFFSET_X_REG_L_M register	62
Table 102.	OFFSET_X_REG_H_M register	62
Table 103.	OFFSET_Y_REG_L_M register	62
Table 104.	OFFSET_Y_REG_H_M register	62
Table 105.	OFFSET_Z_REG_L_M register	62
Table 106.	OFFSET_Z_REG_H_M register	62
Table 107.	WHO_AM_I_M register	63
Table 108.	CTRL_REG1_M register	63
Table 109.	CTRL_REG1_M register description	63
Table 110.	X and Y axes operative mode selection	63
Table 111.	Output data rate configuration	63
Table 112.	CTRL_REG2_M register	64
Table 113.	CTRL_REG2_M register description	64
Table 114.	Full-scale selection	64
Table 115.	CTRL_REG3_M register	64
Table 116.	CTRL_REG3_M register description	64
Table 117.	System operating mode selection	65
Table 118.	CTRL_REG4_M register	65
Table 119.	CTRL_REG4_M register description	65
Table 120.	Z-axis operative mode selection	65
Table 121.	CTRL_REG5_M register	65
Table 122.	CTRL_REG5_M register description	65
Table 123.	STATUS_REG_M register	66
Table 124.	STATUS_REG_M register description	66
Table 125.	INT_CFG_M register	67
Table 126.	INT_CFG_M register description	67
Table 127.	INT_SRC_M register	67
Table 128.	INT_SRC_M register description	67
Table 129.	INT_THS_L_M register	68
Table 130.	INT_THS_H_M register	68
Table 131.	LGA (3.5x3x1 mm) 24-lead package mechanical data	70
Table 132.	Document revision history	71

List of figures

Figure 1.	Pin connections	10
Figure 2.	Recommended power-up sequence	14
Figure 3.	SPI slave timing diagram	15
Figure 4.	I ² C slave timing diagram	16
Figure 5.	Switching operating modes	19
Figure 6.	Multiple reads: accelerometer only	21
Figure 7.	Multiple reads: accelerometer and gyroscope	21
Figure 8.	Accelerometer and gyroscope digital block diagram	22
Figure 9.	Magnetometer block diagram	22
Figure 10.	Bypass mode	23
Figure 11.	FIFO mode	24
Figure 12.	Continuous mode	25
Figure 13.	Continuous-to-FIFO mode	25
Figure 14.	Bypass-to-Continuous mode	26
Figure 15.	LSM9DS1 electrical connections	27
Figure 16.	Accelerometer and gyroscope read and write protocol	31
Figure 17.	Accelerometer and gyroscope SPI read protocol	32
Figure 18.	Multiple byte SPI read protocol (2-byte example)	32
Figure 19.	Accelerometer and gyroscope SPI write protocol	33
Figure 20.	Multiple byte SPI write protocol (2-byte example)	33
Figure 21.	Accelerometer and gyroscope SPI read protocol in 3-wire mode	33
Figure 22.	Magnetic sensor read and write protocol	34
Figure 23.	Magnetic sensor SPI read protocol	35
Figure 24.	Multiple byte SPI read protocol (2-byte example)	35
Figure 25.	Magnetic sensor SPI write protocol	36
Figure 26.	Multiple byte SPI write protocol (2-byte example)	36
Figure 27.	SPI read protocol in 3-wire mode	37
Figure 28.	INT_SEL and OUT_SEL configuration gyroscope block diagram	47
Figure 29.	Wait bit disabled	60
Figure 30.	Wait bit enabled	61
Figure 31.	LGA (3.5x3x1 mm) 24-lead package outline	69

nRF52840

Product Specification

v1.1

5	Power and clock management	55
5.1	Power management unit (PMU)	55
5.2	Current consumption	55
5.2.1	Electrical specification	56
5.3	POWER — Power supply	61
5.3.1	Main supply	61
5.3.2	USB supply	66
5.3.3	System OFF mode	67
5.3.4	System ON mode	68
5.3.5	RAM power control	68
5.3.6	Reset	69
5.3.7	Registers	70
5.3.8	Electrical specification	80
5.4	CLOCK — Clock control	82
5.4.1	HFCLK controller	83
5.4.2	LFCLK controller	84
5.4.3	Registers	87
5.4.4	Electrical specification	96
6	Peripherals	99
6.1	Peripheral interface	99
6.1.1	Peripheral ID	99
6.1.2	Peripherals with shared ID	100
6.1.3	Peripheral registers	100
6.1.4	Bit set and clear	100
6.1.5	Tasks	100
6.1.6	Events	100
6.1.7	Shortcuts	101
6.1.8	Interrupts	101
6.2	AAR — Accelerated address resolver	102
6.2.1	EasyDMA	102
6.2.2	Resolving a resolvable address	102
6.2.3	Use case example for chaining RADIO packet reception with address resolution using AAR	103
6.2.4	IRK data structure	103
6.2.5	Registers	103
6.2.6	Electrical specification	107
6.3	ACL — Access control lists	107
6.3.1	Registers	109
6.4	CCM — AES CCM mode encryption	111
6.4.1	Key-stream generation	111
6.4.2	Encryption	112
6.4.3	Decryption	112
6.4.4	AES CCM and RADIO concurrent operation	113
6.4.5	Encrypting packets on-the-fly in radio transmit mode	113
6.4.6	Decrypting packets on-the-fly in radio receive mode	114
6.4.7	CCM data structure	115
6.4.8	EasyDMA and ERROR event	116
6.4.9	Registers	116
6.4.10	Electrical specification	123
6.5	COMP — Comparator	123
6.5.1	Differential mode	124
6.5.2	Single-ended mode	125
6.5.3	Registers	127
6.5.4	Electrical specification	134

6.6 CRYPTOCELL — ARM TrustZone CryptoCell 310	135
6.6.1 Usage	136
6.6.2 Always-on (AO) power domain	136
6.6.3 Lifecycle state (LCS)	136
6.6.4 Cryptographic key selection	137
6.6.5 Direct memory access (DMA)	137
6.6.6 Standards	138
6.6.7 Registers	138
6.6.8 Host interface	139
6.7 ECB — AES electronic codebook mode encryption	142
6.7.1 Shared resources	142
6.7.2 EasyDMA	142
6.7.3 ECB data structure	142
6.7.4 Registers	143
6.7.5 Electrical specification	145
6.8 EGU — Event generator unit	146
6.8.1 Registers	146
6.8.2 Electrical specification	148
6.9 GPIO — General purpose input/output	148
6.9.1 Pin configuration	149
6.9.2 Registers	151
6.9.3 Electrical specification	156
6.10 GPIOTE — GPIO tasks and events	157
6.10.1 Pin events and tasks	157
6.10.2 Port event	158
6.10.3 Tasks and events pin configuration	158
6.10.4 Registers	159
6.10.5 Electrical specification	163
6.11 I ² S — Inter-IC sound interface	163
6.11.1 Mode	164
6.11.2 Transmitting and receiving	164
6.11.3 Left right clock (LRCK)	165
6.11.4 Serial clock (SCK)	165
6.11.5 Master clock (MCK)	166
6.11.6 Width, alignment and format	166
6.11.7 EasyDMA	168
6.11.8 Module operation	170
6.11.9 Pin configuration	172
6.11.10 Registers	173
6.11.11 Electrical specification	182
6.12 LPCOMP — Low power comparator	183
6.12.1 Shared resources	184
6.12.2 Pin configuration	184
6.12.3 Registers	185
6.12.4 Electrical specification	191
6.13 MWU — Memory watch unit	191
6.13.1 Registers	192
6.14 NFCT — Near field communication tag	205
6.14.1 Overview	206
6.14.2 Operating states	208
6.14.3 Pin configuration	209
6.14.4 EasyDMA	209
6.14.5 Frame assembler	210
6.14.6 Frame disassembler	211

6.14.7	Frame timing controller	212
6.14.8	Collision resolution	213
6.14.9	Antenna interface	214
6.14.10	NFCT antenna recommendations	214
6.14.11	Battery protection	215
6.14.12	References	215
6.14.13	Registers	215
6.14.14	Electrical specification	233
6.15	PDM — Pulse density modulation interface	234
6.15.1	Master clock generator	235
6.15.2	Module operation	235
6.15.3	Decimation filter	235
6.15.4	EasyDMA	236
6.15.5	Hardware example	237
6.15.6	Pin configuration	237
6.15.7	Registers	238
6.15.8	Electrical specification	244
6.16	PPI — Programmable peripheral interconnect	245
6.16.1	Pre-programmed channels	246
6.16.2	Registers	247
6.17	PWM — Pulse width modulation	251
6.17.1	Wave counter	252
6.17.2	Decoder with EasyDMA	255
6.17.3	Limitations	262
6.17.4	Pin configuration	262
6.17.5	Registers	263
6.18	QDEC — Quadrature decoder	271
6.18.1	Sampling and decoding	272
6.18.2	LED output	273
6.18.3	Debounce filters	273
6.18.4	Accumulators	274
6.18.5	Output/input pins	274
6.18.6	Pin configuration	274
6.18.7	Registers	275
6.18.8	Electrical specification	286
6.19	QSPI — Quad serial peripheral interface	286
6.19.1	Configuring peripheral	287
6.19.2	Write operation	287
6.19.3	Read operation	288
6.19.4	Erase operation	288
6.19.5	Execute in place	288
6.19.6	Sending custom instructions	288
6.19.7	Deep power-down mode	289
6.19.8	Instruction set	290
6.19.9	Interface description	290
6.19.10	Registers	295
6.19.11	Electrical specification	307
6.20	RADIO — 2.4 GHz radio	307
6.20.1	Packet configuration	308
6.20.2	Address configuration	309
6.20.3	Data whitening	310
6.20.4	CRC	310
6.20.5	Radio states	311
6.20.6	Transmit sequence	311

6.20.7 Receive sequence	313
6.20.8 Received signal strength indicator (RSSI)	314
6.20.9 Interframe spacing	314
6.20.10 Device address match	315
6.20.11 Bit counter	316
6.20.12 IEEE 802.15.4 operation	316
6.20.13 EasyDMA	324
6.20.14 Registers	325
6.20.15 Electrical specification	354
6.21 RNG — Random number generator	359
6.21.1 Bias correction	360
6.21.2 Speed	360
6.21.3 Registers	360
6.21.4 Electrical specification	363
6.22 RTC — Real-time counter	363
6.22.1 Clock source	363
6.22.2 Resolution versus overflow and the PRESCALER	363
6.22.3 COUNTER register	364
6.22.4 Overflow features	365
6.22.5 TICK event	365
6.22.6 Event control feature	365
6.22.7 Compare feature	366
6.22.8 TASK and EVENT jitter/delay	368
6.22.9 Reading the COUNTER register	370
6.22.10 Registers	371
6.22.11 Electrical specification	376
6.23 SAADC — Successive approximation analog-to-digital converter	376
6.23.1 Input configuration	377
6.23.2 Reference voltage and gain settings	379
6.23.3 Digital output	379
6.23.4 EasyDMA	379
6.23.5 Continuous sampling	381
6.23.6 Oversampling	381
6.23.7 Event monitoring using limits	381
6.23.8 Calibration	382
6.23.9 Registers	382
6.23.10 Electrical specification	397
6.24 SPI — Serial peripheral interface master	398
6.24.1 Functional description	398
6.24.2 Registers	401
6.24.3 Electrical specification	405
6.25 SPIM — Serial peripheral interface master with EasyDMA	406
6.25.1 SPI master transaction sequence	407
6.25.2 D/CX functionality	408
6.25.3 Pin configuration	409
6.25.4 EasyDMA	409
6.25.5 Low power	410
6.25.6 Registers	410
6.25.7 Electrical specification	421
6.26 SPIS — Serial peripheral interface slave with EasyDMA	422
6.26.1 Shared resources	423
6.26.2 EasyDMA	423
6.26.3 SPI slave operation	424
6.26.4 Pin configuration	426

6.26.5 Registers	426
6.26.6 Electrical specification	437
6.27 SWI — Software interrupts	439
6.27.1 Registers	439
6.28 TEMP — Temperature sensor	439
6.28.1 Registers	440
6.28.2 Electrical specification	446
6.29 TWI — I ² C compatible two-wire interface	446
6.29.1 Functional description	446
6.29.2 Master mode pin configuration	447
6.29.3 Shared resources	447
6.29.4 Master write sequence	448
6.29.5 Master read sequence	448
6.29.6 Master repeated start sequence	449
6.29.7 Low power	450
6.29.8 Registers	450
6.29.9 Electrical specification	458
6.30 TIMER — Timer/counter	459
6.30.1 Capture	460
6.30.2 Compare	460
6.30.3 Task delays	460
6.30.4 Task priority	460
6.30.5 Registers	461
6.31 TWIM — I ² C compatible two-wire interface master with EasyDMA	465
6.31.1 EasyDMA	466
6.31.2 Master write sequence	467
6.31.3 Master read sequence	468
6.31.4 Master repeated start sequence	469
6.31.5 Low power	470
6.31.6 Master mode pin configuration	470
6.31.7 Registers	470
6.31.8 Electrical specification	481
6.31.9 Pullup resistor	482
6.32 TWIS — I ² C compatible two-wire interface slave with EasyDMA	482
6.32.1 EasyDMA	485
6.32.2 TWI slave responding to a read command	485
6.32.3 TWI slave responding to a write command	486
6.32.4 Master repeated start sequence	487
6.32.5 Terminating an ongoing TWI transaction	488
6.32.6 Low power	488
6.32.7 Slave mode pin configuration	488
6.32.8 Registers	489
6.32.9 Electrical specification	499
6.33 UART — Universal asynchronous receiver/transmitter	499
6.33.1 Functional description	500
6.33.2 Pin configuration	500
6.33.3 Shared resources	501
6.33.4 Transmission	501
6.33.5 Reception	501
6.33.6 Suspending the UART	502
6.33.7 Error conditions	502
6.33.8 Using the UART without flow control	502
6.33.9 Parity and stop bit configuration	503
6.33.10 Registers	503