Laboratorio 5:
Arduino Nano 33 BLE: GPIO y TinyML

Jafet Soto Arrieta, B77543
Universidad de Costa Rica, Escuela de Ingeniería Eléctrica
IE-0623 Laboratorio de Microcontroladores
29 de mayo del 2024
jafet.soto@ucr.ac.cr

***Resumen*—**

Se creará un clasificador/detector de máscara facial con un kit Arduino Nano 33 BLE y la cámara OV7675 para prevenir el contagio de enfermedades respiratorias. El proyecto incluye capturar 20 imágenes de personas con y sin mascarilla, y sin personas. El modelo se diseñará en Edge Impulse, empleando bloques de preprocesamiento de imagen y aprendizaje por transferencia, ajustando parámetros según el microcontrolador y validando el modelo. Finalmente, se ejecutará el modelo en el microcontrolador, utilizando un LED RGB para indicar detecciones: verde para cara con mascarilla, rojo para sin mascarilla y azul para sin cara.

I. Introducción

Se desarrollará un clasificador/detector de máscara facial utilizando un kit Arduino Nano 33 BLE con la cámara OV7675. Las personas infectadas con virus respiratorios suelen transmitirlos a través de las vías respiratorias, por lo que el uso de máscaras faciales es esencial para evitar la propagación de enfermedades. El detector debe cumplir con los siguientes requisitos:

- Uso del kit Arduino Nano 33 BLE y cámara OV7675.
- Construcción del conjunto de datos:
  1. Capturar 20 imágenes con mascarilla, variando ángulos, colores y personas.
  2. Capturar 20 imágenes sin mascarilla, variando los mismos parámetros.
  3. Capturar 20 imágenes sin personas frente a la cámara.
- Diseño del modelo en Edge Impulse:
  1. Utilizar y configurar un bloque de preprocesamiento de imagen.
  2. Generar el vector de características del preprocesamiento.
  3. Usar un bloque de modelo de aprendizaje por transferencia.
  4. Configurar los parámetros del modelo: ciclos de entrenamiento, tasa de aprendizaje, aumento de datos, selección del modelo considerando las características del microcontrolador.
  5. Realizar el entrenamiento.
- Validación del modelo con pruebas.
- Ejecución del modelo en el microcontrolador:
  1. Encender el LED azul si no se detecta una cara.
  2. Encender el LED rojo si se detecta una cara sin mascarilla.
  3. Encender el LED verde si se detecta una cara con mascarilla.

  La placa cuenta con un LED RGB para las indicaciones.

II. Nota teórica

II-A. Arduino Nano 33 BLE:

El Arduino Nano 33 BLE es un microcontrolador diseñado para aplicaciones de baja potencia y conectividad Bluetooth. Sus características generales incluyen un procesador ARM Cortex-M4 a 64 MHz, 256 KB de memoria flash y 32 KB de RAM. Cuenta con múltiples periféricos como ADC, PWM, UART, SPI e I2C, que permiten la interacción con diversos sensores y dispositivos. Su diagrama de bloques y diagrama de pines proporcionan una visión detallada de las conexiones y funcionalidades. En términos de características eléctricas, opera con un voltaje de 3.3V y tiene un consumo de corriente muy bajo, ideal para aplicaciones portátiles.

Este microcontrolador es fundamental para la recolección de datos, la ejecución del programa de detección de mascarillas y la conexión con el servidor de Edge Impulse.El Arduino Nano 33 BLE es un microcontrolador diseñado para aplicaciones de baja potencia y conectividad Bluetooth. Sus características generales incluyen un procesador ARM Cortex-M4 a 64 MHz, 256 KB de memoria flash y 32 KB de RAM. Cuenta con múltiples periféricos como ADC, PWM, UART, SPI e I2C, que permiten la interacción con diversos sensores y dispositivos. Su diagrama de bloques y diagrama de pines proporcionan una visión detallada de las conexiones y funcionalidades.

En términos de características eléctricas, opera con un voltaje de 3.3V y tiene un consumo de corriente muy bajo, ideal para aplicaciones portátiles. Este microcontrolador es fundamental para la recolección de datos, la ejecución del programa de detección de mascarillas y la conexión con el servidor de Edge Impulse.

II-A1. Diagrama de Pines: El Arduino Nano 33 BLE cuenta con una amplia gama de pines que facilitan su integración en diversos proyectos:

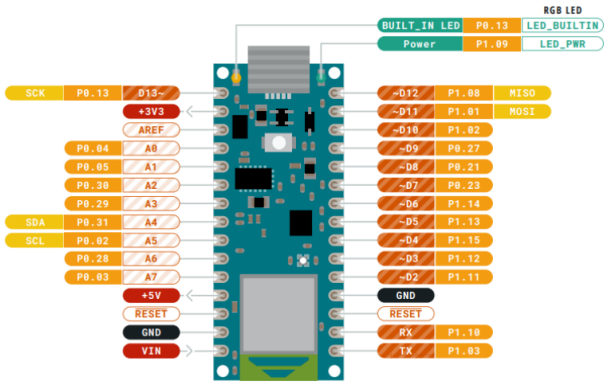

Figura 1: Diagrama de pines del Arduino Nano 33 BLE [1].

Alimentación:

- **VUSB:** Entrada de 5V desde USB.
- **3.3V:** Salida regulada de 3.3V.
- **GND:** Pines de tierra.

Pines Digitales (D0-D13):

- **PWM:** D3, D5, D6, D9, D10, D11.
- **Interrupciones:** D2, D3.

Pines Analógicos (A0-A7):

- Lectura de señales analógicas con resolución de 12 bits.

Comunicación:

- **UART (D0, D1):** Comunicación serial.
- **I2C (A4, A5):** Pines SDA y SCL.
- **SPI (D10, D11, D12, D13):** Comunicación rápida con periféricos.

Otros Pines:

- **RESET:** Reinicia el microcontrolador.
- **AREF:** Referencia de voltaje analógico.
- **VIN:** Entrada de voltaje no regulado.

El microcontrolador incluye un LED RGB controlado por D13 (rojo), D14 (verde) y D15 (azul) para retroalimentación visual.

II-A2. Microcontrolador nRF52840: El nRF52840 es un microcontrolador de Nordic Semiconductor basado en el núcleo ARM Cortex-M4 a 64 MHz. Es conocido por su bajo consumo de energía y capacidades avanzadas de conectividad inalámbrica. Sus características incluyen:

- **Memoria:** 1 MB de flash y 256 KB de RAM.
- **Conectividad:** Bluetooth 5, Bluetooth Mesh, 802.15.4, ANT, y 2.4 GHz propietario.
- **Periféricos:** ADC, SPI, UART, I2C, PWM, USB 2.0, y GPIO.
- **Seguridad:** Motor criptográfico con soporte para AES, ECC, y SHA-256.

El nRF52840 es ideal para aplicaciones de Internet de las Cosas (IoT) y dispositivos portátiles.

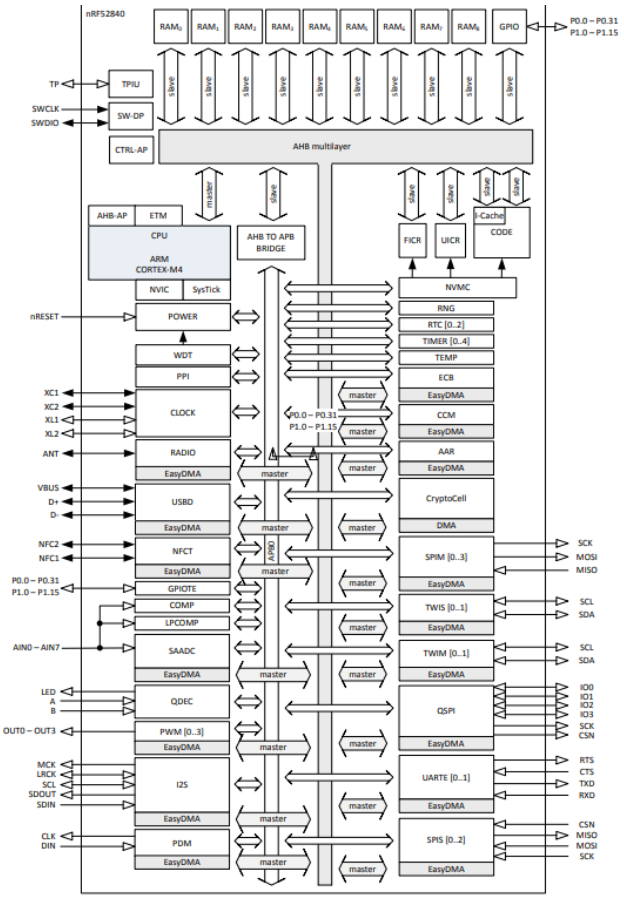

Figura 2: Diagrama de bloques del microcontrolador nRF52840 [1].

II-B. Cámara OV7675:

La cámara OV7675 es un sensor de imagen CMOS utilizado para capturar imágenes de alta calidad en aplicaciones de visión artificial.

Sus características incluyen una resolución de 640x480 píxeles, soporte para imágenes en color (RGB) y en escala de grises, y capacidad de captura a diferentes ángulos y condiciones de iluminación. Los registros de configuración permiten ajustar parámetros como exposición, balance de blancos y ganancia, adaptándose a las necesidades del proyecto.

Esta cámara se integra con el Arduino Nano 33 BLE para capturar imágenes con y sin mascarilla, así como imágenes sin personas, constituyendo el conjunto de datos necesario para el entrenamiento del modelo.

II-C. *Edge Impulse:*

Edge Impulse es una plataforma de desarrollo que facilita la creación y despliegue de modelos de aprendizaje automático en dispositivos de borde. Permite el preprocesamiento de imágenes, el diseño de modelos de aprendizaje por transferencia y la configuración de parámetros como ciclos de entrenamiento y tasa de aprendizaje.

El proceso de diseño del modelo en Edge Impulse incluye la generación de vectores de características a partir de imágenes y el uso de bloques de transferencia de aprendizaje para mejorar la precisión del modelo. La plataforma también permite la validación y prueba del modelo antes de su despliegue en el microcontrolador.

Edge Impulse se conecta con el Arduino Nano 33 BLE para recibir y procesar los datos capturados, facilitando así el entrenamiento y la implementación del modelo de detección de mascarillas.

II-D. *Diseño del Circuito:*

El kit Arduino Tiny Machine Learning es una plataforma diseñada para facilitar la implementación de modelos de aprendizaje automático en dispositivos de baja potencia, como el Arduino Nano 33 BLE. Integra herramientas como Edge Impulse para el entrenamiento y despliegue de modelos directamente en el microcontrolador, permitiendo aplicaciones en campos como la detección de patrones en sensores o el control inteligente de dispositivos, todo dentro de un entorno accesible para desarrolladores y entusiastas de la electrónica y la programación.

Para este laboratorio únicamente es necesario conseguir el kit Arduino Tiny Machine Learning. Por lo tanto, el hardware necesario para el proyecto es:

Nombre	Descripción	Precio
Arduino Tiny Machine Learning Kit	Plataforma que facilita la implementación de modelos de aprendizaje automático en dispositivos de baja potencia como el Arduino Nano 33 BLE. Incluye acceso a Edge Impulse para el desarrollo y despliegue de modelos.	$60
Total		$60

III. DESARROLLO Y ANÁLISIS.

III-A. *Captura de datos*

Para generar el banco de datos necesario para el proyecto, se planeó utilizar la cámara OV7675 del kit Tiny Machine Learning. Sin embargo, durante esta etapa, se presentaron problemas de comunicación entre la máquina virtual y el controlador Arduino Nano 33 BLE, lo que impidió la correcta captura de imágenes.

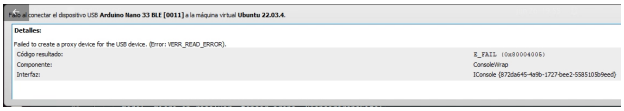

Figura 3: Problema de comunicación entre la máquina virtual y el controlador Nano 33 BLE.

Debido a estos inconvenientes, se decidió generar las imágenes necesarias utilizando una cámara externa. Este enfoque nos permite continuar con el laboratorio y aprender a generar impulsos en Edge Impulse, a pesar de las dificultades técnicas encontradas.

III-B. *Generación del modelo*

Para la creación del modelo de detección de uso de mascarilla, utilizamos la plataforma Edge Impulse. Esta plataforma nos permite generar un modelo que se pueda conectar con el microcontrolador Arduino Nano 33 BLE y la camara del kit.

Se capturaron un total de 60 imágenes para el proyecto, distribuidas en tres carpetas que contienen las configuraciones siguientes:

- **20img_con_mascarilla:** 20 imágenes de personas con mascarilla, variando ángulos, colores y personas.
- **20img_sin_mascarilla:** 20 imágenes de personas sin mascarilla, variando ángulos, colores y personas.
- **20img_sin_persona:** 20 imágenes sin personas frente a la cámara.

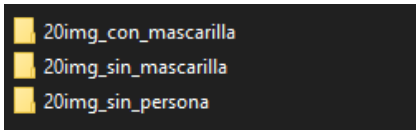

Figura 4: Carpetas con las configuraciones de las imágenes capturadas.

Estas imágenes se utilizarán en Edge Impulse para entrenar el modelo, empleando bloques de preprocesamiento de imagen y aprendizaje por transferencia. Configuramos los parámetros del modelo, incluyendo los ciclos de entrenamiento, la tasa de aprendizaje, y el aumento de datos, considerando las características del microcontrolador.

Procedemos a definir la sección de Create impulse en Edge Impulse, donde utilizamos tres bloques: uno de procesamiento de imágenes, otro de preprocesamiento de imágenes y otro de transfer learning. Estos bloques se combinan para generar un modelo que utilice las imágenes mencionadas anteriormente y sea capaz de determinar si la persona frente a la cámara lleva mascarilla o no.

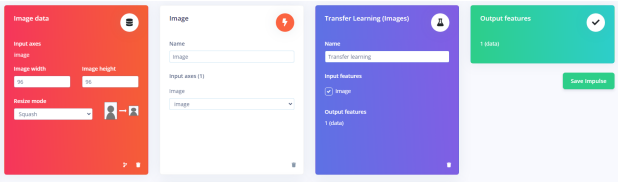

Figura 5: Sección de Impulse Design en Edge Impulse.

El **Color Depth** se mantiene como RGB para conservar la estructura original de las fotos y asegurar la semejanza con los datos capturados.

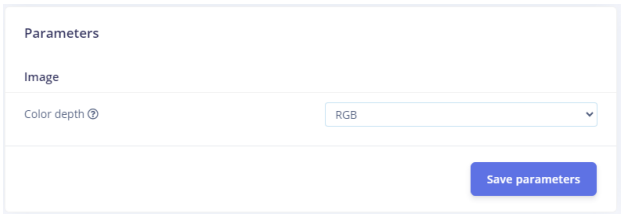

Figura 6: Configuración de **Color Depth** como RGB en Edge Impulse.

Generamos las características (features) a partir de las imágenes capturadas para entrenar el modelo. Este paso es crucial para extraer información relevante de las imágenes que permitirá al modelo distinguir entre personas con y sin mascarilla.

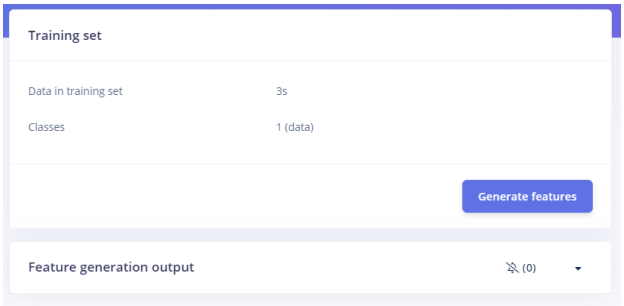

Figura 7: Generación de características en Edge Impulse.

Al analizar el gráfico de características generadas, observamos la presencia de tres grupos distintos. Este resultado es coherente con la implementación que estamos desarrollando, ya que refleja las significativas diferencias entre las imágenes de personas con mascarilla, sin mascarilla y sin personas.

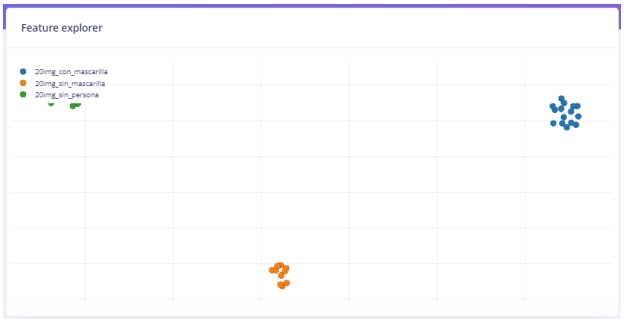

Figura 8: Gráfico de características generadas en Edge Impulse.

Para la implementación del modelo, hemos seleccionado el módulo MobileNetV1 96x96 0.2 OFFICIALLY SUPPORTED, diseñado para clasificar imágenes de manera eficiente con un tamaño de entrada de 96x96 píxeles. Este módulo es ideal para nuestra aplicación debido a su soporte explícito para imágenes RGB, lo cual es coherente con las imágenes capturadas por la cámara OV7675. Consume aproximadamente 83.1K de RAM y 218.3K de ROM con la configuración y optimizaciones predeterminadas, lo que lo hace adecuado para aplicaciones en dispositivos de borde como el Arduino Nano 33 BLE.

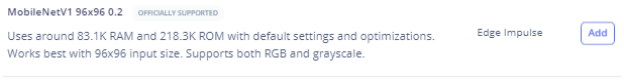

Figura 9: Configuración del módulo MobileNetV1 96x96 en Edge Impulse.

En la configuración de la red neuronal, seleccionamos 55 ciclos de entrenamiento, una tasa de aprendizaje de 0.005 y un conjunto de validación con un tamaño del 35 %. Estos parámetros, combinados con el módulo MobileNetV1 96x96 0.2, están diseñados para diferenciar de manera general entre las imágenes capturadas, enfocándose en cambios amplios en la cobertura facial. Esta estrategia buscó lograr una detección robusta de la presencia o ausencia de mascarillas, alcanzando una precisión del 100 % en las pruebas realizadas.

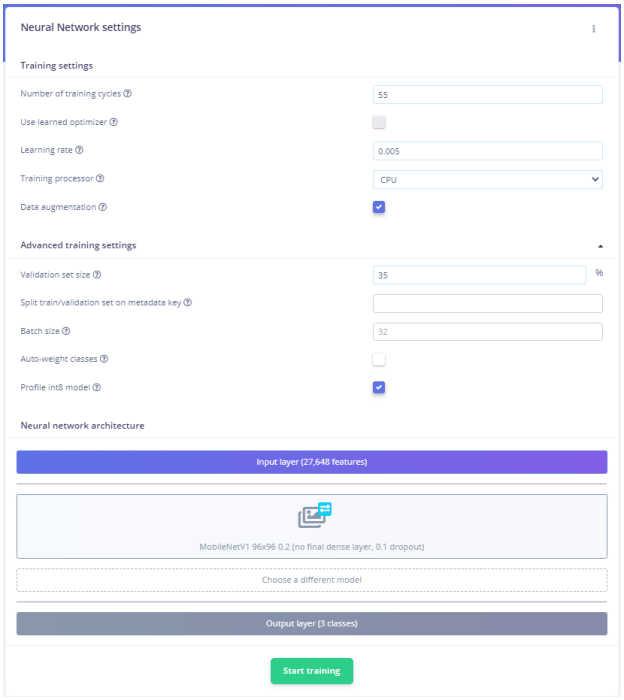

Figura 10: Configuración de Neural Network Settings en Edge Impulse.

Figura 11: Resultados de precisión mostrando un 100 % de accuracy en la detección de mascarillas.

Para validar la efectividad del modelo de detección de mascarillas, procedimos a la fase de pruebas utilizando el conjunto de datos reservado para ello. En esta etapa, realizamos pruebas exhaustivas para verificar la capacidad del modelo entrenado para distinguir entre las tres clases definidas: personas con mascarilla, personas sin mascarilla y sin personas.

Utilizamos imágenes diferentes a las del entrenamiento para evaluar la generalización del modelo ante nuevas capturas y condiciones. Este proceso nos permitió confirmar que el modelo logra una detección precisa y consistente, validando así su robustez y eficacia en la aplicación real.

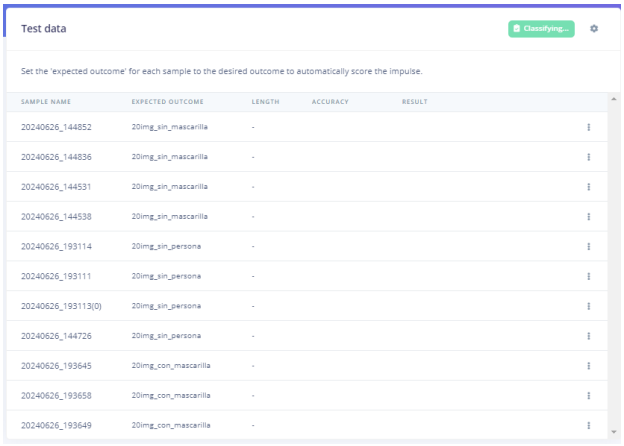

Figura 12: Resultados de las pruebas de validación del modelo de detección de mascarillas.

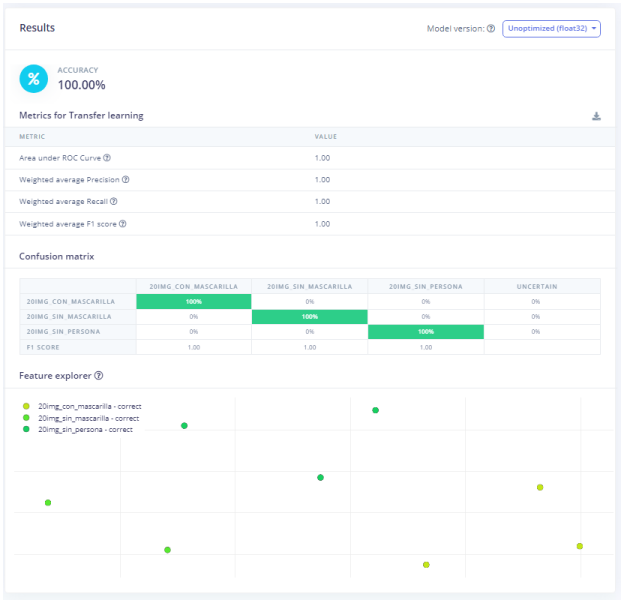

Figura 13: Resultados detallados de las pruebas de validación del modelo de detección de mascarillas.

III-C. Carga en el MCU

Ingresamos ahora a la sección de despliegue (”deployment”) en Edge Impulse para llevar nuestro modelo de detección de mascarillas al microcontrolador. Esta fase es crucial para implementar el modelo entrenado en un entorno operativo real, asegurando que el Arduino Nano 33 BLE pueda ejecutar

la detección de mascarillas de manera efectiva y eficiente. A continuación, exploraremos cómo configuramos y cargamos el modelo en el MCU utilizando las herramientas proporcionadas por Edge Impulse.

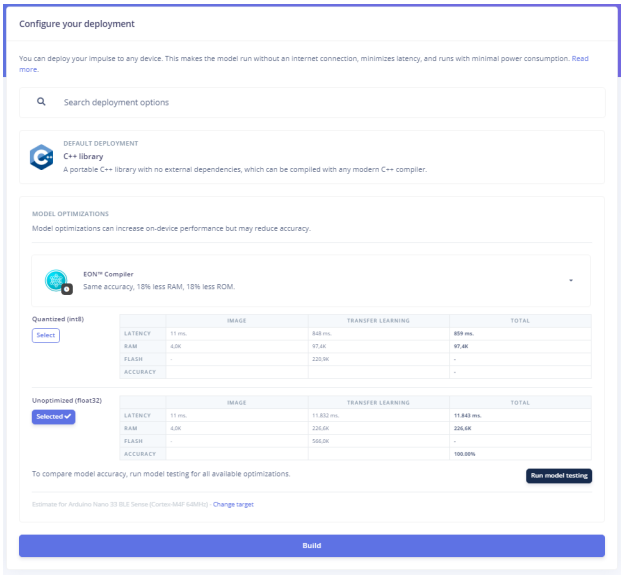

Figura 14: Sección de despliegue (deployment) en Edge Impulse para configurar y cargar modelos en el microcontrolador.

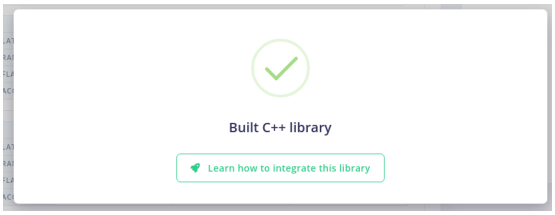

Figura 15: Confirmación de la creación exitosa de la librería C en Edge Impulse para el modelo de detección de mascarillas.

Para facilitar el acceso y la integración de nuestro modelo de detección de mascarillas en el Arduino Nano 33 BLE, hemos agregado la librería como un archivo comprimido ‘.zip‘ al entorno de desarrollo integrado (IDE) de Arduino. Esto nos permite incorporar fácilmente las funciones y el modelo TensorFlowLite entrenado directamente en el proyecto.

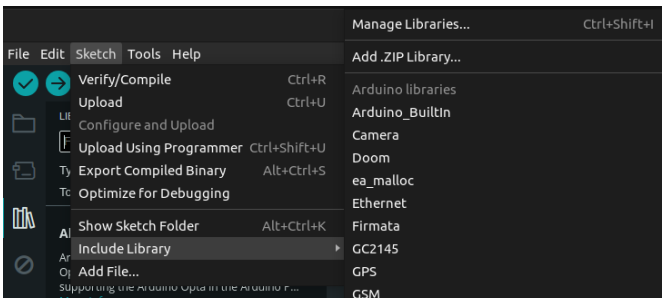

Figura 16: Archivo ‘.zip‘ de la librería TensorFlowLite añadido al entorno de desarrollo integrado (IDE) de Arduino.

Para utilizar el modelo en el Arduino Nano 33 BLE, es necesario descomprimir el archivo ‘.zip‘ proporcionado por Edge Impulse. Esto agregará las siguientes carpetas al directorio del proyecto:

```
src
|-- CMakeLists.txt
|-- edge-impulse-sdk
|-- model-parameters
|-- README.txt
|-- tflite-model
```

Estas carpetas contienen los archivos necesarios para la integración y ejecución del modelo de TensorFlow Lite en el microcontrolador, asegurando que todas las dependencias y configuraciones estén correctamente organizadas.

Para completar la funcionalidad de nuestro sistema de detección de mascarillas, utilizamos el LED RGB integrado en la placa Arduino Nano 33 BLE para visualizar el estado de la detección.

Implementamos un Máquina de Estados Finitos (FSM) que define el comportamiento del LED en función de la señal de detección obtenida del modelo TensorFlow ejecutado en el microcontrolador. Utilizando la función ‘digitalWrite(LED, state)‘, donde ‘LED‘ puede ser ‘LEDR‘, ‘LEDG‘ o ‘LEDB‘, y ‘state‘ puede ser ‘LOW‘ o ‘HIGH‘, configuramos el estado del LED RGB para indicar claramente si se detecta una cara con mascarilla (LED verde), sin mascarilla (LED rojo) o ninguna cara detectada (LED azul).

Es muy importante mencionar que nos basamos en el código de ejemplo de la librería **Arduino_Tensorflowlite** [2], específicamente en el código de **person_detection**.

El código implementa un modelo de detección de personas utilizando TensorFlow Lite y Arduino. En la función setup(), se inicializa el modelo, se verifica su versión y se configura un MicroMutableOpResolver para las operaciones específicas del grafo del modelo. Se asigna memoria para los tensores del modelo y se verifica la configuración del tensor de entrada. En el bucle principal loop(), se captura una imagen, se ejecuta la inferencia del modelo y se procesan los resultados para determinar la presencia de personas, utilizando la función RespondToDetection() para manejar los puntajes de salida del modelo.

Para integrar y ejecutar el modelo en el Arduino Nano 33 BLE, comenzamos incluyendo la biblioteca del clasificador del modelo con la siguiente línea:

```
18  // ====================
19  #include "edge-impulse-sdk/classifier/ei_run_classifier.h"
20  // ====================
```

Figura 17: Inclusión de la biblioteca del clasificador en el código de ejemplo Arduino Nano 33 BLE.

IV. CONCLUSIONES Y RECOMENDACIONES.

En este proyecto, se propuso desarrollar un clasificador/-detector de uso de mascarilla facial utilizando el kit Arduino Nano 33 BLE y la cámara OV7675, con el objetivo de prevenir la propagación de enfermedades respiratorias. Se realizó un diseño exhaustivo en Edge Impulse para crear un modelo de detección eficiente, capaz de distinguir entre personas con mascarilla, sin mascarilla y sin personas.

A lo largo del desarrollo del proyecto, se lograron avances significativos en la captura de imágenes y en la creación del modelo en Edge Impulse. Sin embargo, debido a limitaciones de tiempo y problemas técnicos con la carga del archivo en el microcontrolador, no fue posible completar la implementación final y probar el sistema de detección en el Arduino Nano 33 BLE.

Es importante destacar que todos los diseños y preparativos para el proyecto fueron realizados de manera adecuada. La creación del conjunto de datos, el entrenamiento del modelo y la configuración de los parámetros se llevaron a cabo siguiendo las mejores prácticas y utilizando herramientas avanzadas. El modelo de detección de mascarillas ha mostrado una precisión del 100 % en las pruebas realizadas en la plataforma de Edge Impulse.

REFERENCIAS

[1] *Arduino Nano 33 BLE Datasheet.* Retrieved from https://docs.arduino.cc/resources/datasheets/ABX00030-datasheet.pdf (Arduino).

[2] *TensorFlow Lite Micro Examples for Arduino.* Retrieved from https://github.com/tensorflow/tflite-micro-arduino-examples (TensorFlow).