Reconocimiento de Audios Arduino Nano BLE Tiny ML

Diego Iván Perea Montealegre (2185751) diego.perea@uao.edu.co

Esteban Ramos Gómez (2181034) esteban.ramos@uao.edu.co

Brayan Fernando Castro Balanta brayan.castro@uao.edu.co

Facultad de Ingeniería, Universidad Autónoma de Occidente

Cali, Valle del Cauca

Resumen - Mediante este documento realizaremos la entrega de este último mini proyecto para la asignatura de inteligencia artificial en sistemas embebidos, para esta ocasión realizamos el reconocimiento de comandos de voz mediante la utilización de un Arduino 32 BLE, google Colab y Edge Impulse.

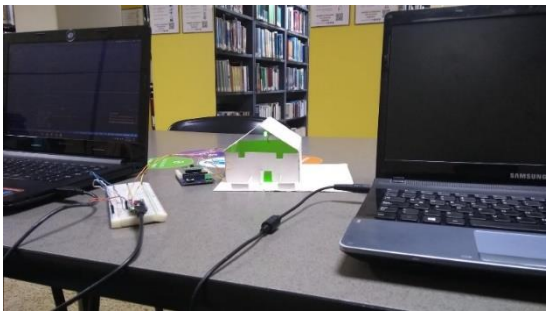


Ilustración 1: Representación del mini proyecto

Índice de Términos – Redes Neuronales, Inteligencia Artificial, Muestreo, Blueooth, Arduino, Windows, Android.

I. INTRODUCCIÓN

En los dispositivos móviles, gracias a Google es muy fácil hacer que estos puedan reconocer comandos de voz básicos como lo son, encender nuestro celular, llamar a "Valeria", pronóstico para el día, etc. Esto es gracias a que se ha implementado un modelo de redes neuronales que permite que el dispositivo pueda extraer las características de la voz humana mediante el micrófono que incorporan, pasan por una caja negra de comandos y procesos donde tenemos como salida la ejecución del comando dicho por la persona.

A esto también se le conoce como Speech to Text.

Ilustración 2: Imagen tomada de -> https://developer.nvidia.com/blog/how-to-build-domain-specific-automatic-speech-recognition-models-on-gpus/

En pocas palabras, esto es el proceso que realizar

un sistema embebido o procesador de algún dispositivo móvil o de escritorio de poder captar la voz humana y traducirlo a lenguaje de máquina para que esta pueda realizar alguna tarea en específico.

Para el procesamiento de estas señales es necesario realizar un espectrograma

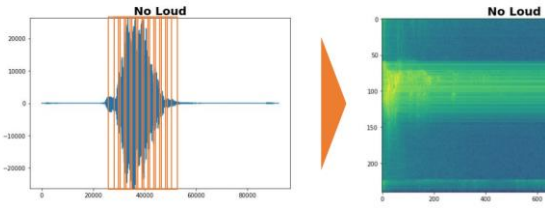


Ilustración 3: Imagen tomada de:https://campus.uaovirtual.edu.co/pluginfile.php/597273/mod_resource/content/2/Tiny%20ML%20UAO%20Course%20Presentation%2012%20Key%20Word%20Spotting.pdf

II. OBJETIVOS DEL PROYECTO

- Implementar un modelo de reconocimiento de audio en un sistema embebido – Arduino BLE 32.
- Reconocer almenos 5 comandos por medio del Arduino BLE 32.
- Realizar un despliegue de manera física del modelo entrenado en el Arduino BLE 32.

También para la realización de este proyecto se ha pensado en los objetivos de desarrollo sostenible en este caso hablamos de la industria, innovación e infraestructura.

Ilustración 4: Imagen tomada de -> https://www.un.org/sustainabledevelopment/es/infrastructure/

Mediante este proyecto, el objetivo es motivar a las futuras generaciones para que aprenden a desarrollar sus modelos de redes neuronales para poder realizar sistemas que puedan identificar los comandos de las personas para realizar diferentes acciones o procesos.

Un ejemplo claro es que las personas necesiten abrir una puerta a distancia mediante comandos de voz, o en una reunión deseen encender el televisor, etc.

III. ETAPA DE REVISIÓN

A. Etapa Final

Mediante este documento realizaremos la entrega del ultimo mini proyecto para Inteligencia Artificial en sistemas embebidos, este documento será enviado al portal UAO donde nuestro profesor va a calificar este trabajo y así mismo realizar sus apreciaciones al mismo. Este debe ser enviado el 22 de noviembre del 2022 antes de las 11:59 PM.

IV. DESARROLLO

Se configuró el arduino nano ble sense para interactuar con la toma de datos de audio en edge impulse, por lo que se siguieron las indicaciones de gua de conexión de conectar el arduino blse sense en Windows.

Arduino Nano 33 BLE Sense

The Arduino Nano 33 BLE Sense is a tiny development board with a Cortex-M4 microcontroller, motion sensors, a microphone and BLE - and it's fully supported by Edge Impulse. You'll be able to sample raw data, build models, and deploy trained machine learning models directly from the studio. It's available for around 30 USD from Arduino and a wide range of distributors.

You can also use the Arduino Tiny Machine Learning Kit to run image classification models on the edge with the Arduino Nano and attached OV7675 camera module (or connect the hardware together via jumper wire and a breadboard if purchased separately).

The Edge Impulse firmware for this development board is open source and hosted on GitHub: edgeimpulse/firmware-arduino-nano-33-ble-sense.

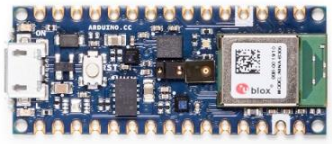


Ilustración 5: Descripción y componentes del arduino BLE 33

En donde se siguió el paso más sencillo para configurarlo debido que edge impulse nos da una guía de paso a paso pero esta dura mucho tiempo , por lo que seguirá descargando el cli.

Installing dependencies

To set this device up in Edge Impulse, you will need to install the following software

1. Edge Impulse CLI.
2. Arduino CLI.
   - Here's an instruction video for Windows.
   - The Arduino website has instructions for macOS and Linux.
3. On Linux:
   - GNU Screen: install for example via `sudo apt install screen`.

Problems installing the CLI?
See the Installation and troubleshooting guide.

Ilustración 6: Independencias Arduino CLI

Descargándolo en diferentes sistemas operativos en installacion cli en este caso en Windows 64 bits , dado esto se descarga el frimware necesario.

2. Update the firmware

The development board does not come with the right firmware yet. To update the firmware:

1. Download the latest Edge Impulse firmware, and unzip the file.

Todos los archivos se deben combinar en una carpeta (el nombre puede sr cualquiera) en este caso el nombre será ArduinoCli. Antes de ejecutar el archivo .bat se debe conectar el arduino ble sense y se debe oprimir dos veces el botón hasta que titile o parpadee uno de los leds integrados ,

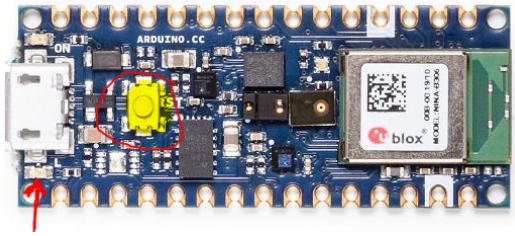


Ilustración 7: Vista superior Arduino BLE 33

Seguido de esto se debe ejecutar el archivo con extensión .bat que dará el inicio de la configuración del Arduino ble sense ,

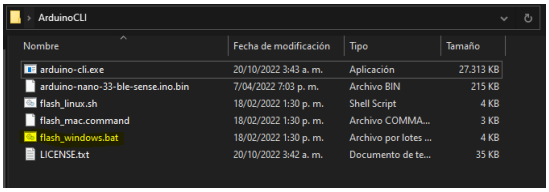


Ilustración 8: Ejecucion archivo .bat

Dentro de edge impulse se conecta y se comprueba la conexión , y se realiza la toma de datos en el cual serán 5 clases en el que son : casa on , casa off , ventilador on , ventilador off y fondo , esto con el fin de que cuando diga “casa on” pondrá en encendido un led representando una luz , y cuando se diga “casa off” la apagará , con el ventilador funciona de las misma manera en donde cuando se diga “ventilador on” se pondrá en funcionamiento el ventilador y cuando se diga “ventilador off” ;En la parte del fondo será representado con un led en donde detecte un fondo(ruido de fondo) se encenderá un led y cuando no lo detecte este se apagará. Se toma un total de 8 muestras de 10s para cada una de las clases y se realiza su

respectivo split para capturar los audios que se quieren extraer. Por lo que se da 84% en train y 16% en test.

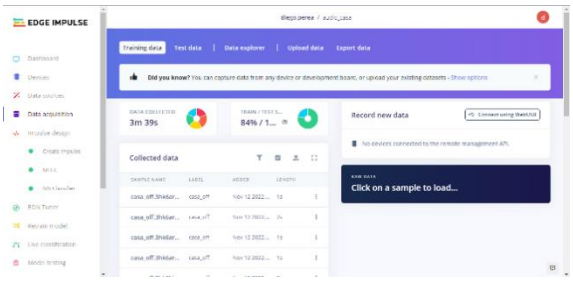


Ilustración 9: Interfaz Edge Impulse

En la creación del impulso se tomó por defecto o predeterminado los valores indicados de edge impulse con clasificación keras.

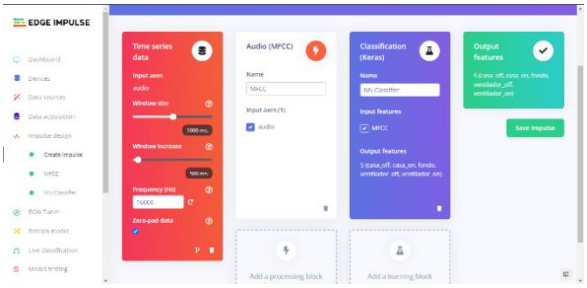


Ilustración 10: Seleccion de modelo - Edge Impulse

La sección de MFCC se dejó los valores predeterminados por edge impulse.

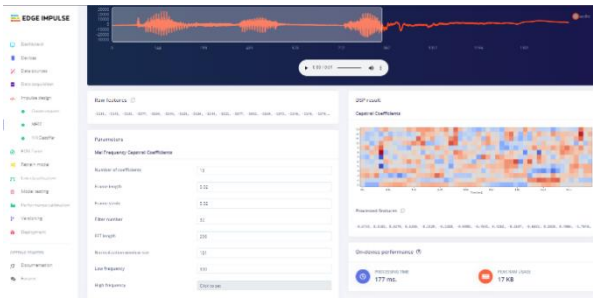


Ilustración 11: Selección NN Clasiffiearse

Por último, la sección de NN Clasiffierse modificó la parte de number of training cycles con 200 debido a que mejoraba la precisión del modelo.

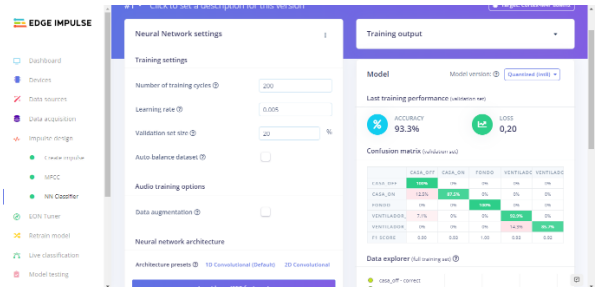


Ilustración 12: Matriz de precisión y confución

En su arquitectura se tomó con convoluciones 2D y números de filtros prudentes como lo fue el primer filtro con 16 con medida de kernel de 3 y su correspondiente dropout de 0.25 para evitar overfitting, en su seguida convolución 2d el filtro es de 32 con medida de kernel de 3 y su correspondiente dropout de 0.25 , estos números de filtros y convoluciones fueron robadas de intentar con prueba y error cual era la mejor precisión adquirida.

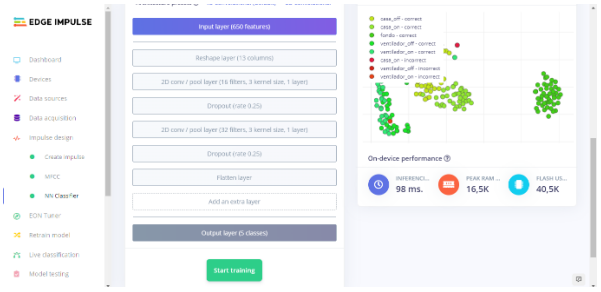


Ilustración 13: Numero de neuronas y % de error de clasificación

En el deployment se seleccionó como librería de arduino y se cuantificó para reducir su tamaño y latencia sin perder su eficacia al clasificar.

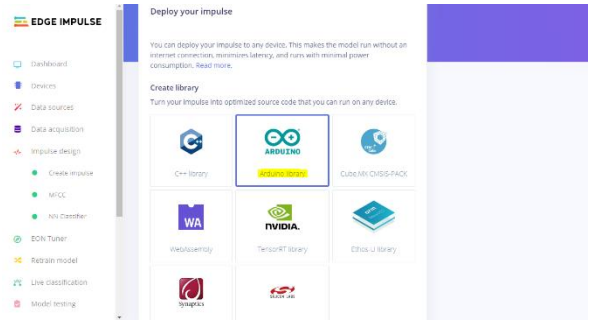


Ilustración 14: Deployment - Edge Impulse

Al momento de cuantificar se evidencia la utilidad de realizar por sus grandes comparaciones frente a la otra manera de realizarlo.

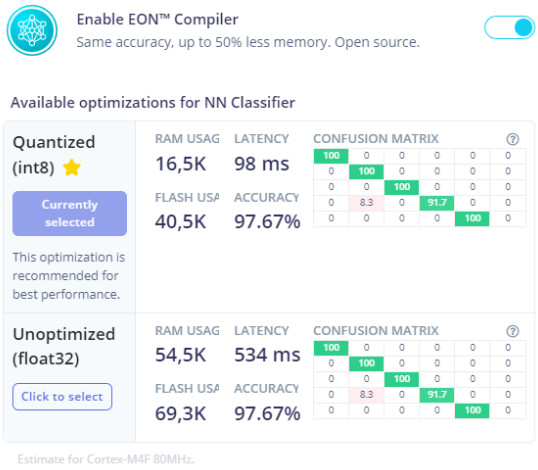


Ilustración 15: Resultados Modelo Final

En código se usó visual studio Code son la extensión de PlatformIO para desarrollar el código e implementación de una forma muy cómoda debido a que solo se arrastra la carpeta generada a lib y en código de main.cpp solo es nombrar dicha librería.

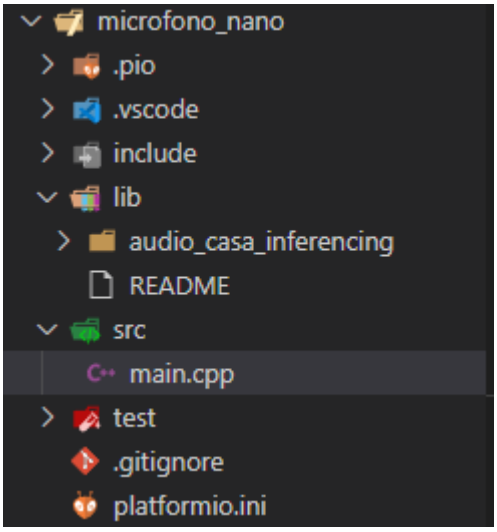


Ilustración 16: Selección PlatformIO

Se tomó base del código de ejemplo de arduino ble sense detección de micrófono , modificando y agregando solo para nuestro contexto .en donde la el porcentaje de clasificación debería ser mayor a 90% en la clasificación de “casa on” , “casa off” y “fondo” , y mayor a 85% en la clasificación de “ventilador on” y “ventilador off”deido a estas últimas al momento del modelo de clasificación observamos que le era de dificultad realizar la clasificación por su extensa palabra de audio comparada con la anteriores.

```
void puente(int pred_index){
switch (pred_index)
  {
    case 0:     // casa off:      [0] ==> All casa off  d6
        if(mayor>0.90)
       digitalWrite(luz, LOW);
       ei_printf("casa apagada");
      break;

    case 1:     // casa on:       [1] ==> Green casa on
        if(mayor>0.90){
        digitalWrite(luz,HIGH);
        digitalWrite(fondo,LOW);
        ei_printf("casa prendido"); }
      break;

    case 2:     // fondo:   [2] ==> Red fondo
        if(mayor>0.90){
      digitalWrite(fondo,HIGH);// pin d7
      ei_printf("fondo dado");}
      break;

    case 3:      //ventilador off:[3] ==> Blue
      if(mayor>0.85){
      digitalWrite(ventilador, LOW); //ventilador d7
      digitalWrite(fondo,LOW);
      ei_printf("ventilador apagado");}
      break;
    case 4:      //ventilador on:[3] ==> Blue
        if(mayor>0.85){
      digitalWrite(ventilador, HIGH);
      digitalWrite(fondo,LOW);
      ei_printf("ventilador prendido");}
      break;
  }

}

```

Ilustración 17: Escala de precisión para los comandos

Toda la parte de código del arduino ble sense para entender las anteriores variables estará anexado en este documento

Para que el motor tenga una buena potencia de movimiento se usa un Arduino UNO para que funcione como batería, por esto se debe compartir la tierra en el arduino uno y Arduino nano bles , el código es el siguiente en el Arduino UNO.

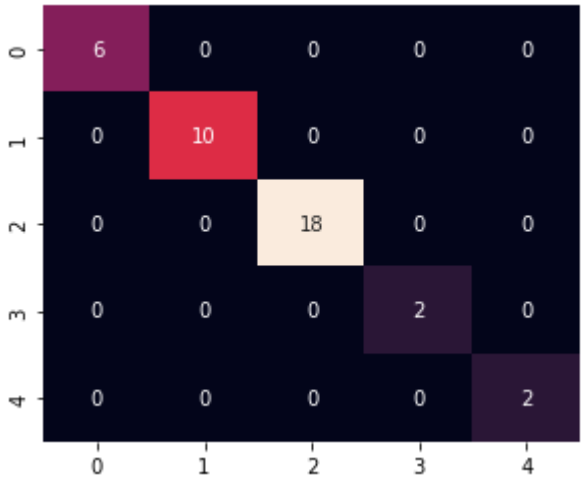
Hemos realizado un puente mediante un Arduino 1 para poder encender nuestro motor y las luces de la casa por medio de este código.

sketch_nov19a

```
int ventilador =9;
int a=HIGH;

void setup() {
  // put your setup code here, to run once:
    pinMode(7,INPUT);
    pinMode(ventilador,OUTPUT);
    Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  a=digitalRead(7);
  if(a==HIGH){
    digitalWrite(ventilador,HIGH);
    }else{
      digitalWrite(ventilador,LOW);
      }

  Serial.println(a);
}
```

Ilustración 18: Código Puente Arduino uno

También se realizó de una forma en Colab para así crear una biblioteca tipo extensión de arduino (.h) para que pueda implementarse en el Arduino nano ble sense



V. DIAGRAMA DE BLOQUES

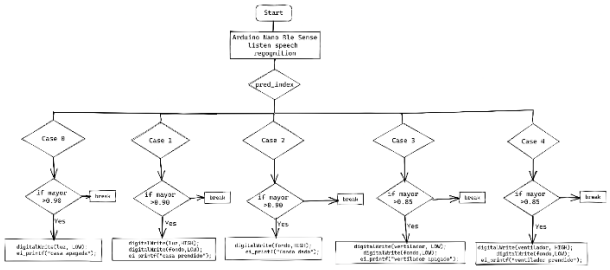


Ilustración 19: Diagrama de Bloques - Ver Anexos

VI. FUENTES DE INSPIRACION

La implementación física está en contexto de la domótica, que es en una casa que tiene luces inteligentes y ventilador inteligente en que atreves del audio del lenguaje humano lo detecta, por lo que se prende y apaga el objeto mencionado.

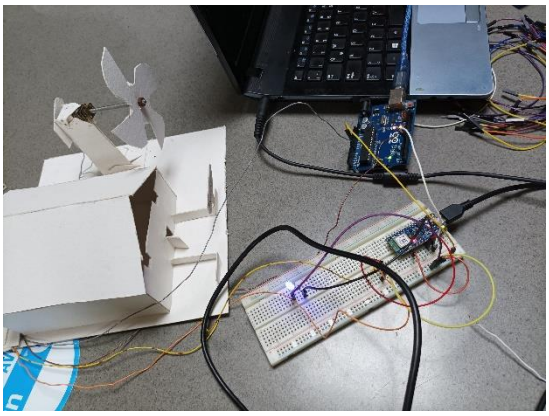


Ilustración 20: Indicador de ruido de fondo (LED AZUL)

Tomando como referencia la opción que tiene google con (Google Home) que nos permite realizar pequeñas acciones en nuestra casa como desde poner música en nuestro equipo de sonido, encender la TV, encender las luces de nuestro cuarto o apagar el aire acondicionado, hemos decidido realizar una pequeña maqueta para poder controlar las luces de esta, encender el ventilador y finalizando distinguir el sonido del ambiente mediante un led azul.

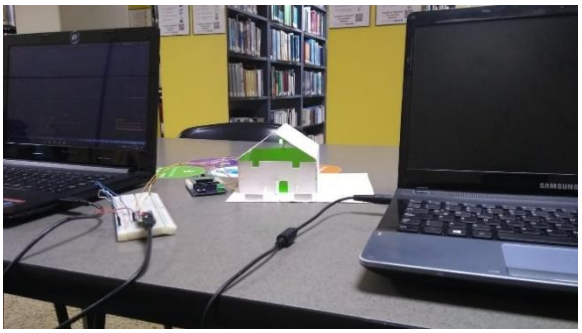


Ilustración 21: Primera Prueba luces de la casa

Como se puede apreciar, en la maqueta que hemos realizado, primero se implementó el encendido y el apagado de la casa por medio de los comandos de voz.

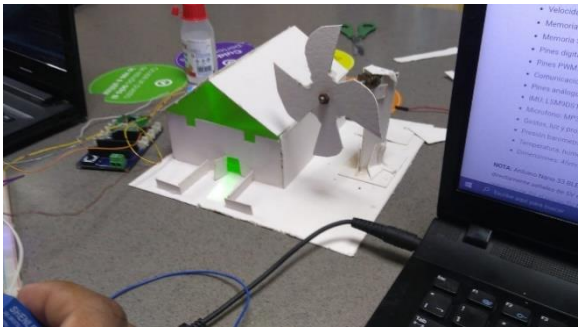


Ilustración 22: Implementación del ventilador

Y para finalizar hemos realizado la implementación del ventilador para nuestra casa por medio de los comandos de voz que hemos mencionado anteriormente.

Aquí adjuntamos el video donde esta nuestra sustentación mediante el video que hemos hecho para youtube.

VIDEO: https://youtu.be/fkeDxcOSXCU

VII. AGRADECIMIENTOS Y PALABRAS PERSONALES

Como grupo de trabajo, queremos darle las gracias a nuestro profesor por toda la enseñanza prestada y brindada durante estos 4 meses de semestre en la Universidad Autónoma de Occidente en la asignatura de Inteligencia Artificial para sistemas Embebidos, la verdad hemos aprendido bastante durante este tiempo que hemos estado viendo clase con usted y queremos reconocer la gran labor que ha realizado durante este semestre 2022 – 03. Nosotros ya que estamos finalizando carrera y materias en la universidad, nuevamente queremos darle las gracias por todo y esperamos que nos volvamos a ver pronto a futuro. Saludos y hasta pronto

Att: Diego Ivan Perea, Esteban Ramos Gomez, Brayan Castro.

Ilustración 23: Foto ultima tarde durante la realización de este mini proyecto - Universidad Autónoma de Occidente

VIII. CONCLUCIONES

- La toma de datos de los audios en su split es de vital importancia debido a que estas entradas serán de manera concurrente en la clasificación de los audios

-La plataforma de edge impulse en la toma de datos, entrenamiento y despliegue de los modelos entrenado, le indica a los desarrolladores la facilidad de usarlo e implementarlo en diferentes sistemas embebidos.

-Tener más datos de entrenamiento y testeo hace que mejore la precisión y loss a medida que se realiza el modelo.

- Tener diferentes filtros 2d de la convolución ayuda en el desempeño de la precisión, así como también el número de ciclos.

-La inteligencia artificial llegó para quedarse y será la herramienta ideal en la transformación de la mejora para la humanidad, nos ayudará y mejorará en múltiples actividades en que serán incluso inimaginables, por eso usando deep learning en tareas complejas como detección de audios de forma de espectrogramas facilita la clasificación de esta y mejora con las convoluciones en 2d, para así mejora la precisión (acurracy) y perdida(loss) indicando la buena clasificación de dichos audios. Todo esto nos trae a preguntarnos ¿Cuál será la próxima aplicación de la inteligencia artificial en sistemas embebidos que transforme la humanidad?

IX. REFERENCIAS

Anexos

[1] "Arduino Nano 33 BLE". Arduino Official Store. https://store.arduino.cc/products/arduino-nano-33-ble (accedido el 22 de noviembre de 2022).

[2] "Arduino Nano 33 BLE - ABX00030 - Geek Factory". Geek Factory. https://www.geekfactory.mx/tienda/tarjetas/arduino/arduino-nano-33-ble/ (accedido el 22 de noviembre de 2022).

[3] "¿ Qué es el sonido? - Soporte Multimedia Perú". Soporte Multimedia Perú. https://soportemultimedia.com/que-es-el-sonido/ (accedido el 22 de noviembre de 2022).

[4] "Clasificación de sonido con Redes Neuronales - Diego Calvo". Diego Calvo. https://www.diegocalvo.es/clasificacion-de-sonido-mediante-redes-neuronales-convolucionales/ (accedido el 22 de noviembre de 2022).

ANEXOS

DIAGRAMA DE BLOQUES

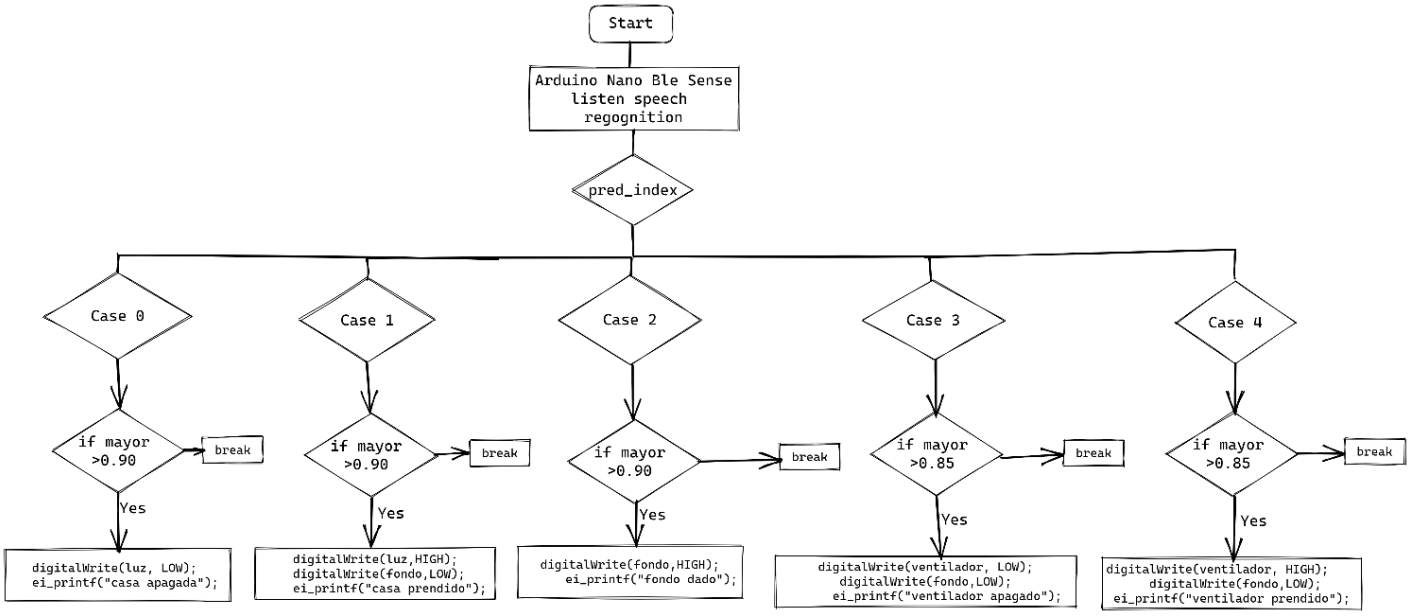


Ilustración 24: ANEXO 1 - Diagrama de bloques

CODIGO ARDUINO NANO BLE

```
#include <Arduino.h>


/* Edge Impulse ingestion SDK
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */

// If your target is limited in memory remove this macro to save 10K RAM
#define EIDSP_QUANTIZE_FILTERBANK   0

/**
 * Define the number of slices per model window. E.g. a model window of 1000
ms
 * with slices per model window set to 4. Results in a slice size of 250 ms.
 * For more info: https://docs.edgeimpulse.com/docs/continuous-audio-
sampling
 */
#define EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW 3

/*
 ** NOTE: If you run into TFLite arena allocation issue.
 **
 ** This may be due to may dynamic memory fragmentation.
 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt
(create
 ** if it doesn't exist) and copy this file to
 **
`<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed_core>/<core_version>/`.
 **
 ** See
 ** (https://support.arduino.cc/hc/en-us/articles/360012076960-Where-are-
the-installed-cores-located-)
 ** to find where Arduino installs cores on your machine.
```

```
 **
 ** If the problem persists then there's not enough memory for this model
and application.
 */

/* Includes ---------------------------------------------------------------
*/
#include <PDM.h>
#include <audio_casa_inferencing.h>
//COMIENZA PUENTE
int ventilador=D7;
int luz=D6;
int fondo=D8;

float mayor=0.0;




/** Audio buffers, pointers and selectors */
typedef struct {
    signed short *buffers[2];
    unsigned char buf_select;
    unsigned char buf_ready;
    unsigned int buf_count;
    unsigned int n_samples;
} inference_t;

static inference_t inference;
static bool record_ready = false;
static signed short *sampleBuffer;
static bool debug_nn = false; // Set this to true to see e.g. features
generated from the raw signal
static int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);

/**parte de declaracion
 * @brief      PDM buffer full callback
 *             Get data and call audio thread callback
 */
static void pdm_data_ready_inference_callback(void)
{
    int bytesAvailable = PDM.available();

    // read into the sample buffer
    int bytesRead = PDM.read((char *)&sampleBuffer[0], bytesAvailable);
```

```
    if (record_ready == true) {
        for (int i = 0; i<bytesRead>> 1; i++) {
            inference.buffers[inference.buf_select][inference.buf_count++] =
sampleBuffer[i];

            if (inference.buf_count >= inference.n_samples) {
                inference.buf_select ^= 1;
                inference.buf_count = 0;
                inference.buf_ready = 1;
            }
        }
    }
}

/**
 * @brief      Init inferencing struct and setup/start PDM
 *
 * @param[in]  n_samples  The n samples
 *
 * @return     { description_of_the_return_value }
 */
static bool microphone_inference_start(uint32_t n_samples)
{
    inference.buffers[0] = (signed short *)malloc(n_samples * sizeof(signed
short));

    if (inference.buffers[0] == NULL) {
        return false;
    }

    inference.buffers[1] = (signed short *)malloc(n_samples * sizeof(signed
short));

    if (inference.buffers[1] == NULL) {
        free(inference.buffers[0]);
        return false;
    }

    sampleBuffer = (signed short *)malloc((n_samples >> 1) * sizeof(signed
short));

    if (sampleBuffer == NULL) {
        free(inference.buffers[0]);
        free(inference.buffers[1]);
```

```
        return false;
    }

    inference.buf_select = 0;
    inference.buf_count = 0;
    inference.n_samples = n_samples;
    inference.buf_ready = 0;

    // configure the data receive callback
    PDM.onReceive(&pdm_data_ready_inference_callback);

    PDM.setBufferSize((n_samples >> 1) * sizeof(int16_t));

    // initialize PDM with:
    // - one channel (mono mode)
    // - a 16 kHz sample rate
    if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) {
        ei_printf("Failed to start PDM!");
    }

    // set the gain, defaults to 20
    PDM.setGain(127);

    record_ready = true;

    return true;
}

/**
 * @brief      Wait on new data
 *
 * @return     True when finished
 */
static bool microphone_inference_record(void)
{
    bool ret = true;

    if (inference.buf_ready == 1) {
        ei_printf(
            "Error sample buffer overrun. Decrease the number of slices per
model window "
            "(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW)\n");
        ret = false;
    }
```

```
    while (inference.buf_ready == 0) {
        delay(1);
    }

    inference.buf_ready = 0;

    return ret;
}

/**
 * Get raw audio signal data
 */
static int microphone_audio_signal_get_data(size_t offset, size_t length,
float *out_ptr)
{
    numpy::int16_to_float(&inference.buffers[inference.buf_select ^
1][offset], out_ptr, length);

    return 0;
}

/**
 * @brief      Stop PDM and release buffers
 */
static void microphone_inference_end(void)
{
    PDM.end();
    free(inference.buffers[0]);
    free(inference.buffers[1]);
    free(sampleBuffer);
}



/**
 * @brief      Arduino setup function
 */
void setup()
{

    // put your setup code here, to run once:
    Serial.begin(115200);
     pinMode(ventilador,OUTPUT);

     pinMode(luz,OUTPUT);
```

```
     pinMode(fondo,OUTPUT);

// ****************
// Lo Nuevo
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(LEDR, OUTPUT);
    pinMode(LEDG, OUTPUT);
    pinMode(LEDB, OUTPUT);

    // Ensure the LED is off by default.
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(LEDR, HIGH);
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);
// ****************
// Lo Nuevo

    // comment out the below line to cancel the wait for USB connection
(needed for native USB)
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");

    // summary of inferencing settings (from model_metadata.h)
    ei_printf("Inferencing settings:\n");
    ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
    ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
    ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT /
16);
    ei_printf("\tNo. of classes: %d\n",
sizeof(ei_classifier_inferencing_categories) /
                                            sizeof(ei_classifier_inferencing
_categories[0]));

    run_classifier_init();
    if (microphone_inference_start(EI_CLASSIFIER_SLICE_SIZE) == false) {
        ei_printf("ERR: Failed to setup audio sampling\r\n");
        return;
    }
}


void puente(int pred_index){
switch (pred_index)
  {
    case 0:     // casa off:      [0] ==> All casa off  d6
```

```
        if(mayor>0.90)
       digitalWrite(luz, LOW);
       ei_printf("casa apagada");
      break;

    case 1:     // casa on:       [1] ==> Green casa on
        if(mayor>0.90){
        digitalWrite(luz,HIGH);
        digitalWrite(fondo,LOW);
        ei_printf("casa prendido"); }
      break;

    case 2:     // fondo:   [2] ==> Red fondo
        if(mayor>0.90){
      digitalWrite(fondo,HIGH);// pin d7
      ei_printf("fondo dado");}
      break;

    case 3:      //ventilador off:[3] ==> Blue
      if(mayor>0.85){
      digitalWrite(ventilador, LOW); //ventilador d7
      digitalWrite(fondo,LOW);
      ei_printf("ventilador apagado");}
      break;
    case 4:      //ventilador on:[3] ==> Blue
        if(mayor>0.85){
      digitalWrite(ventilador, HIGH);
      digitalWrite(fondo,LOW);
      ei_printf("ventilador prendido");}
      break;
  }

}



/**
 * @brief      Arduino main function. Runs the inferencing loop.
 */


// ****************
// Lo Nuevo
void turn_off_leds(){
    digitalWrite(LEDR, HIGH);
```

```
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);
}
// ****************
// Lo Nuevo
/*
 * Idle:      [0] ==> All OFF
 * lift:       [1] ==> Green ON
 * maritime:   [2] ==> Red ON
 * terrestrial:[3] ==> Blue ON
 * Anomaly         ==> LED_BUILTIN ON
 */

// ****************
// Lo Nuevo
void turn_on_leds(int pred_index) {
  switch (pred_index)
  {
    case 0:     // Idle:      [0] ==> All OFF
      turn_off_leds();
      digitalWrite(LEDB, LOW);
      break;

    case 1:     // lift:       [1] ==> Green ON
      turn_off_leds();
      digitalWrite(LEDG, LOW);
      break;

    case 2:     // maritime:   [2] ==> Red ON
      turn_off_leds();
      digitalWrite(LEDR, LOW);
      break;

    case 3:      //terrestrial:[3] ==> Blue ON
      turn_off_leds();

      break;
  }
}
// ****************
// Lo Nuevo



void loop()
```

```
{
    bool m = microphone_inference_record();
    if (!m) {
        ei_printf("ERR: Failed to record audio...\n");
        return;
    }

    signal_t signal;
    signal.total_length = EI_CLASSIFIER_SLICE_SIZE;
    signal.get_data = &microphone_audio_signal_get_data;
    ei_impulse_result_t result = {0};

    EI_IMPULSE_ERROR r = run_classifier_continuous(&signal, &result,
debug_nn);
    if (r != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", r);
        return;
    }

 // ****************
// Lo Nuevo
    int pred_index = 0;
    float pred_value = result.classification[0].value;
 // ****************
// Lo Nuevo

    if (++print_results >= (EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW)) {
        // print the predictions
        ei_printf("Predictions ");
        ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
            result.timing.dsp, result.timing.classification,
result.timing.anomaly);
        ei_printf(": \n");
        for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
            ei_printf("    %s: %.5f\n", result.classification[ix].label,
                      result.classification[ix].value);

 // ****************
// Lo Nuevo
        if (result.classification[ix].value > pred_value){
          pred_index = ix;
          pred_value = result.classification[ix].value;
        }
 // ****************
// Lo Nuevo
```

```
        }

    mayor=pred_value;
// Lo Nuevo
        turn_on_leds (pred_index);
 // ****************
// Lo Nuevo

//llamada a puente
puente(pred_index);


#if EI_CLASSIFIER_HAS_ANOMALY == 1
        ei_printf("    anomaly score: %.3f\n", result.anomaly);

#endif

        print_results = 0;
    }

}



#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR !=
EI_CLASSIFIER_SENSOR_MICROPHONE
#error "Invalid model for current sensor."
#endif
```