

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624 Laboratorio de Microcontroladores

Proyecto final: Sistema de domótica con IA

Pablo Durán Segura B62416

Sergio Garino Vargas B73157

28 de noviembre de 2024

Índice

1. Resumen	3
2. Nota teórica	3
2.1. Características del Arduino Nano 33 BLE	3
2.2. Características del ESP8266	4
2.3. Periféricos	5
2.4. Registros	6
2.5. Arquitectura del proyecto	6
2.6. Componentes	7
2.7. IoT	7
3. Desarrollo y análisis de resultados	9
4. Conclusiones	19
5. Recomendaciones	19

1. Resumen

El presente proyecto integra inteligencia artificial e IoT para desarrollar un sistema de domótica capaz de controlar dispositivos mediante comandos de voz. Para ello, se utiliza un Arduino Nano BLE 33, donde se carga un modelo de inteligencia artificial encargado de reconocer los comandos de voz emitidos por el usuario. Una vez procesados, estos comandos se transmiten de manera serial a un microcontrolador ESP8266 que los enviará a otro ESP8266 a través de un servidor MQTT. El modelo de IA se construye en la plataforma Edge Impulse para facilitar tareas como la configuración de la red neuronal.

El primer ESP8266 actúa como intermediario entre el sistema de reconocimiento de voz y la red IoT. Este dispositivo publica los comandos recibidos en un servidor MQTT, permitiendo que otros dispositivos conectados puedan suscribirse y recibir las instrucciones de manera eficiente.

Un segundo ESP8266 se conecta al servidor MQTT para suscribirse a los tópicos de los comandos publicados. Este microcontrolador es el encargado de interpretar los comandos y ejecutar acciones sobre los dispositivos conectados, en este caso, un bombillo y un abanico. Todos los microcontroladores utilizados, tanto el Arduino Nano BLE 33 como los ESP8266, fueron programados utilizando Arduino IDE.

En el siguiente enlace se puede acceder al repositorio de github: https://github.com/PabloJoseD/Proyecto_micros.

Palabras claves: *Arduino, ESP8266, IoT, MQTT, EdgeImpulse, micrófono, redes neuronales, domótica.*

2. Nota teórica

2.1. Características del Arduino Nano 33 BLE

El Arduino Nano 33 BLE tiene las siguientes características.[\[1\]](#)

- Procesador 64 MHz Arm Cortex-M4F
- 1 MB Flash + 256 KB RAM
- Bluetooth 5
- 14 pines digitales y 8 analógicos.
- Periféricos UART, I2C, SPI, y PWM.
- Unidad de Medición Inercial LSM9DS1 (acelerómetro, giroscopio y magnetómetro)
- Barómetro y sensor de temperatura LPS22HB, sensor de humedad relativa HTS221, micrófono integrado.

A continuación se muestra el diagrama de pines del Arduino.

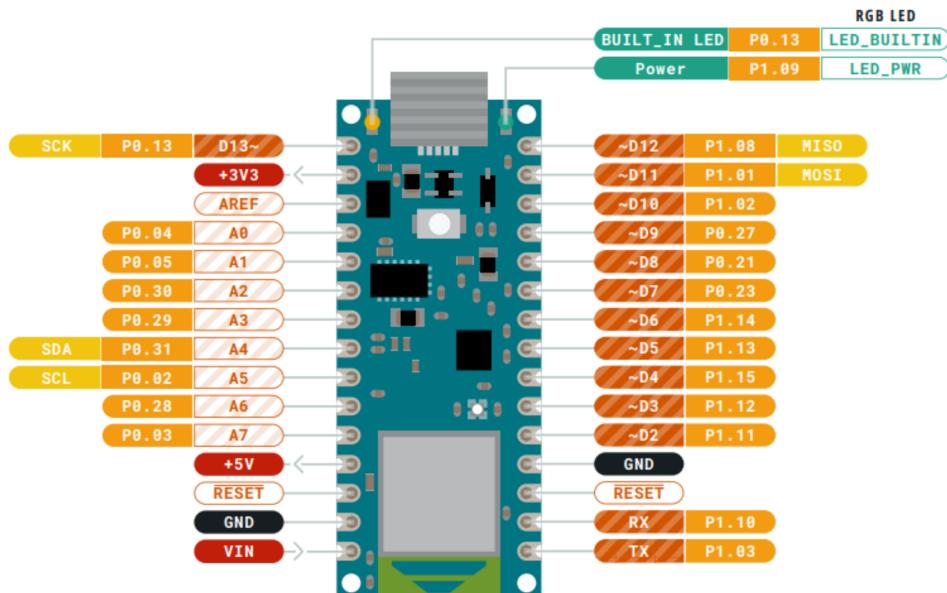


Figura 1: Diagrama de pines del Arduino BLE 33.[1]

2.2. Características del ESP8266

El ESP8266 tiene las siguientes características.[2]

- Procesador Tensilica L106 32-bit de hasta 160MHz.

- Memoria RAM <50kB + 1MB Flash.
 - Memoria flash externa de hasta 16MB.
 - Transmisor y receptor de 2.4 GHz.
 - Soporta TCP/IP y protocolo 802.11 b/g/n WLAN MAC.
 - 17 pines GPIO.
 - Interfaz SPI, I2C, UART, PWM, IR, ADC (10 bits).

A continuación se muestra el diagrama de pines del ESP8266.

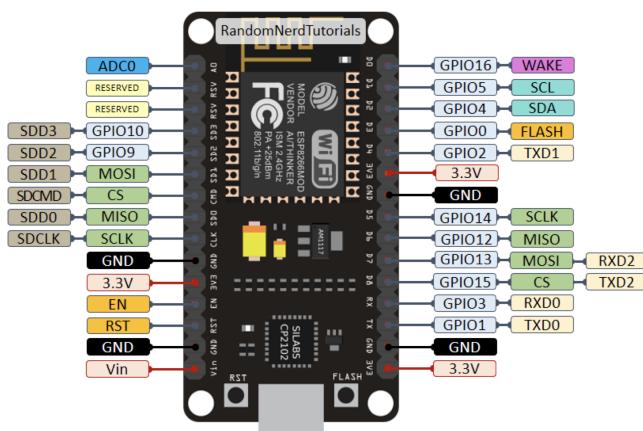


Figura 2: Diagrama de pines del ESP8266.[3]

2.3. Periféricos

En este caso no se utilizaron periféricos, solo el micrófono que ya está incorporado en el Arduino.

A continuación, se detallan algunas características de dicho componente.[3]

- Módulo MP34DT05.
 - AOP = 122.5 dbSPL.
 - 64 dB SNR.
 - Sensitividad omnidireccional

- Sensitividad $-26 \text{ dBFS} \pm 3 \text{ dB}$.

2.4. Registros

Para este proyecto no se utilizó ningún registro del Arduino. Todos los programas se desarrollaron por medio de bibliotecas que contienen funciones útiles para manejar los sensores y la comunicación serial.

2.5. Arquitectura del proyecto

El proyecto consta de dos etapas. Una primera etapa donde están conectados el Arduino Nano 33 BLE y un ESP8266 por medio de cables. La segunda etapa contiene otro ESP8266 que maneja el bombillo y el abanico. Se utilizaron relés para separar la etapa de potencia con la etapa de electrónica. A continuación, se muestra un diagrama con la arquitectura del proyecto.

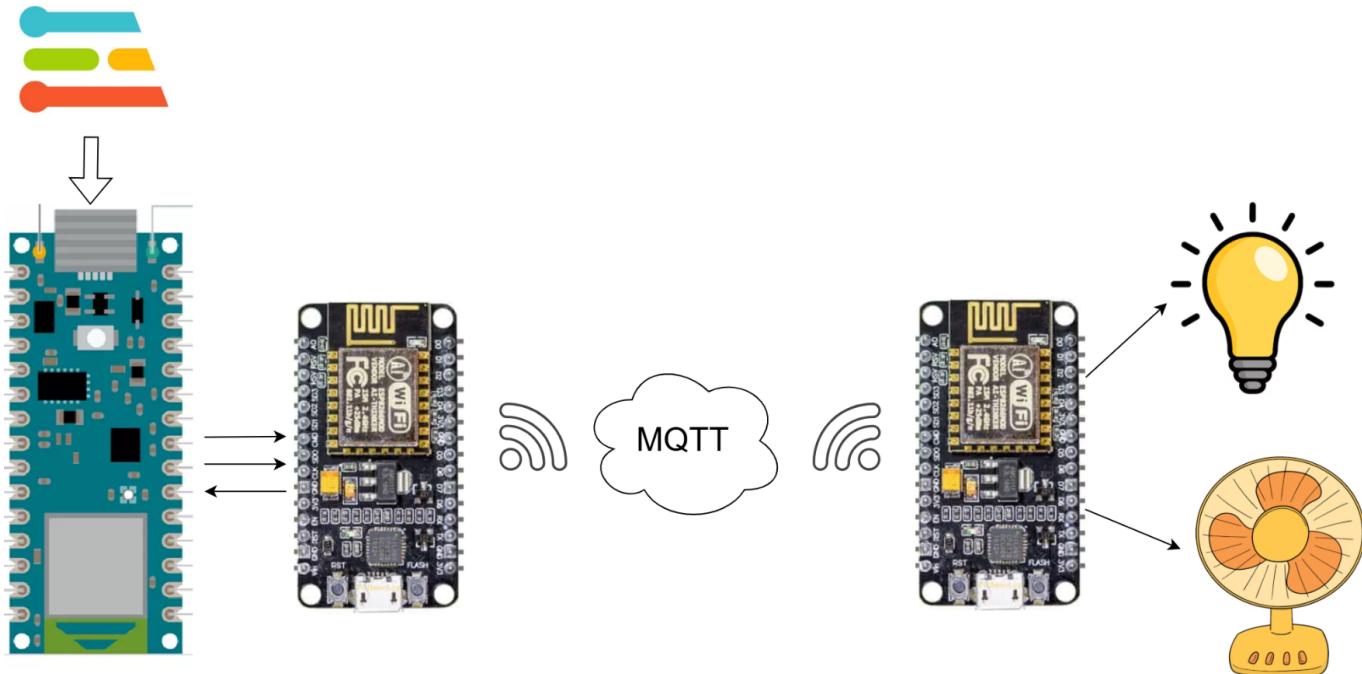


Figura 3: Arquitectura del proyecto. Creación propia

2.6. Componentes

Componente	Precio unitario	Cantidad
Arduino nano 33 BLE	22913.34	1
ESP8266	5581.78	2
Módulo relé 5V	1132.11	2
Abanico 12V	8407.08	1
Bombillo LED 9W	663.72	1
Plafon	752.21	1
Enchufe 15A 125V	380.53	1
Cable eléctrico 12 (1m)	300	2
Total	47190.68	

Tabla 1: Tabla de componentes

2.7. IoT

Es relevante conocer el concepto de Internet de las cosas (IoT) ya que es un pilar fundamental del proyecto pues la comunicación de las dos etapas se da a través de internet de forma inalámbrica, para ello utiliza el protocolo MQTT.

El término IoT, o Internet de las cosas, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos. Gracias a la llegada de los chips de ordenador de bajo coste y a las telecomunicaciones de gran ancho de banda, ahora tenemos miles de millones de dispositivos conectados a Internet. Esto significa que los dispositivos de uso diario, como los cepillos de dientes, las aspiradoras, los coches y las máquinas, pueden utilizar sensores para recopilar datos y responder de forma inteligente a los usuarios.[4]

Como se mencionó previamente, la comunicación es posible gracias a MQTT, el cual es un protocolo de transporte de mensajería de publicación/suscripción cliente-servidor. Es ligero, abierto, sencillo y está diseñado para ser fácil de implementar. Estas características lo hacen ideal para su uso en muchas situaciones, incluidos los entornos limitados como la comunicación Máquina a Máquina (M2M) y el Internet de las Cosas (IoT) en los que se requiere un pequeño y/o el ancho de banda de la red es escaso. El protocolo utiliza el puerto 1883. [5]

El protocolo MQTT fue inventado en 1999 por Andy StanfordClark (IBM) y Arlen Nipper (Arcom, ahora Cirrus Link). Ellos necesitaban un protocolo de mínima pérdida de batería y mínimo ancho de banda para conectar con oleoductos vía satélite. En dos inventores especificaron varios requisitos para el futuro protocolo: [5]

- Aplicación sencilla
- Entrega de datos con calidad de servicio
- Ligereza y eficiencia del ancho de banda
- Agnóstico a los datos
- Conciencia de sesión continua

El modelo Pub/Sub se puede describir con la siguiente imagen:

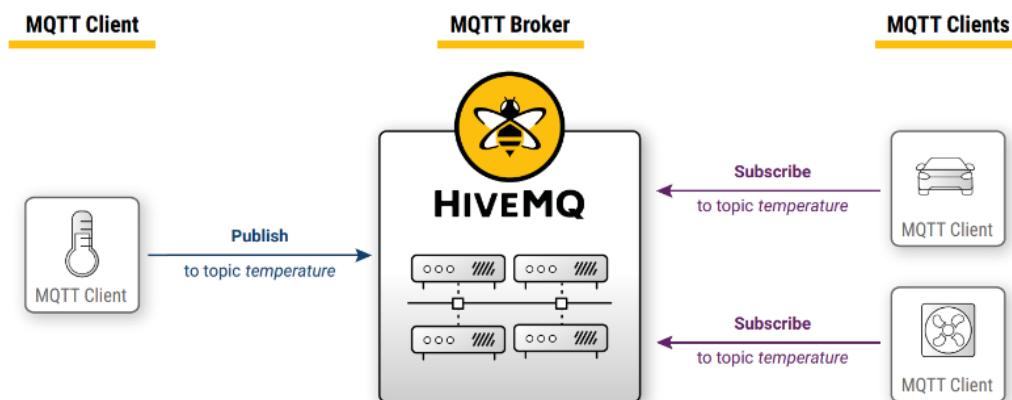


Figura 4: Modelo Pub/Sub MQTT

A diferencia del modelo Cliente/Servidor en MQTT se tiene un broker que sirve de mediador entre clientes, los cuales pueden publicar en algún tópico para enviar información, o bien el cliente puede suscribirse al tópico para recibir la información se esté publicando ahí.

3. Desarrollo y análisis de resultados

La primera etapa del proyecto es el transmisor, esta consta del Arduino y una ESP8266, tal como se ve en el siguiente diagrama.

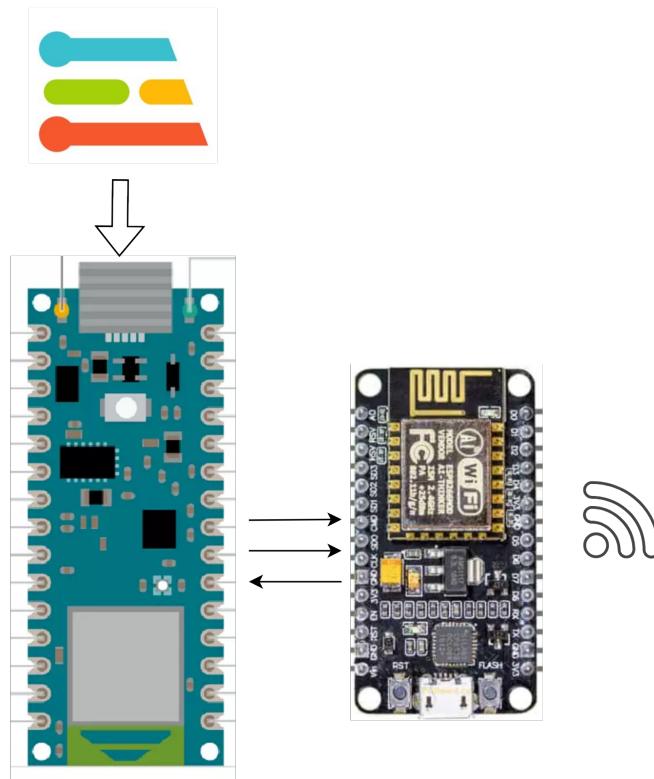


Figura 5: Transmisor

En la propuesta inicial solo se tenía el ESP, sin embargo, se tuvo que incorporar el Arduino ya que la plataforma Edge Impulse no tiene soporte para la placa, mientras que el Arduino además de tener este tipo soporte ya incorpora el micrófono que se necesita para controlar la red por voz.

La comunicación entre el Arduino Nano y el ESP es mediante los pines seriales, se tomó como base el ejemplo en [6], el cual utiliza la biblioteca `<SoftwareSerial.h>` soportada en ambas placas y con ayuda de ChatGPT [7] se pudieron hacer las correcciones necesarias, de manera que el pin TX del arduino se conecta con el pin D2 del ESP, el pin Rx del Arduino se conecta con D3 del ESP y se deben conectar ambas tierras.

Con base en la interpretación que hace el Arduino del comando de voz este envía un número al

ESP, las posibilidades se describen a continuación.

- 0: Se envía un 0 si el modelo determina que lo más probable es que se haya dicho “VIENTO”
- 1: 0: Se envía un 1 si el modelo determina que lo más probable es que se haya dicho “LUZ”
- 2: 0: Se envía un 2 si el modelo determina que lo más probable es que se haya dicho “APAGAR”

Para que el Arduino Nano BLE 33 Sense pueda detectar los comandos de voz se entrenó un modelo de TinyML utilizando la plataforma Edge Impulse, el proyecto está abierto y se puede acceder mediante el siguiente enlace <https://studio.edgeimpulse.com/public/566701/live>.

Se utilizó como referencia la documentación de Edge Impulse, en especial un tutorial de reconocimiento de comandos [8] y una guía de como utilizar Arduino Nano BLE 33 Sense con la plataforma [9].

Para construir el modelo de reconocimiento de voz lo primero fue recolectar las muestras un 81% se usó para entrenamiento y el otro 19% para validación del modelo, en total se recolectaron cincuenta y cinco minutos de muestras, cada una dura un segundo y se distribuyen en cinco etiquetas.

1. LUZ: Permite controlar el bombillo
2. VIENTO: Permite controlar el abanico
3. APAGAR: Permite apagar todo
4. Ruido: Esto es ruido blanco
5. Desconocido: Esto son palabras aleatorias diferentes a los comandos de control.

Para recolectar las muestras de los tres comandos se utilizaron cinco personas, tres voces masculinas y dos femeninas, para ampliar el rango de frecuencias y tonos de voz y que así el modelo pueda recibir comandos de diferentes personas.

El set de muestras de ruido y palabras desconocidas fue obtenido de Edge Impulse. [10]

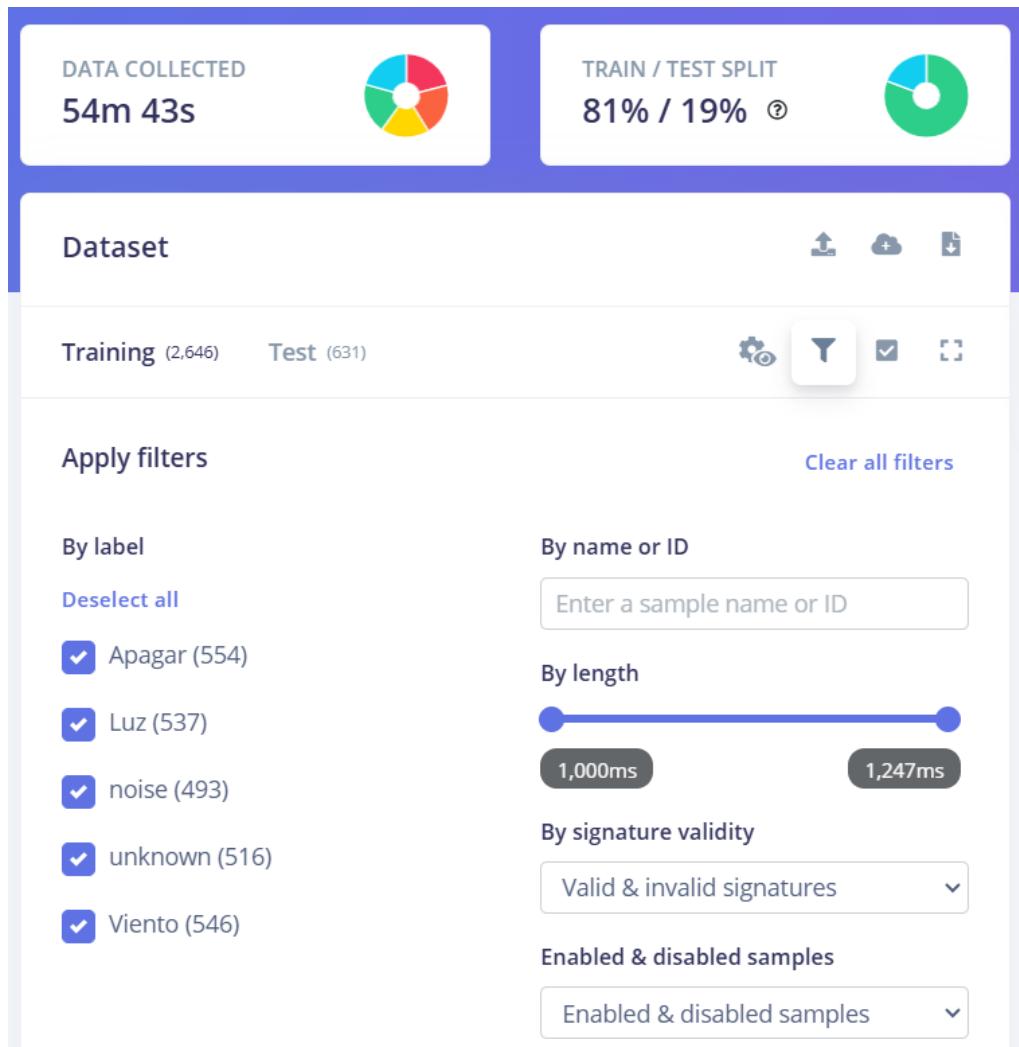


Figura 6: Adquisición de datos

Con los datos recolectados se debe crear un impulso, el cual consta de cuatro partes, la series de tiempo con los datos, un bloque de procesamiento Audio (MFCC) el cual extrae características de las señales de audio mediante coeficientes cepstrales de frecuencias Mel, ideales para la voz humana [11], para el aprendizaje se usa un bloque de clasificación el cual aprende patrones a partir de datos y puede aplicarlos a datos nuevos. Ideal para categorizar movimientos o reconocer audio [12]. En el último bloque se especifica que el modelo tiene cinco salidas, una por cada etiqueta.

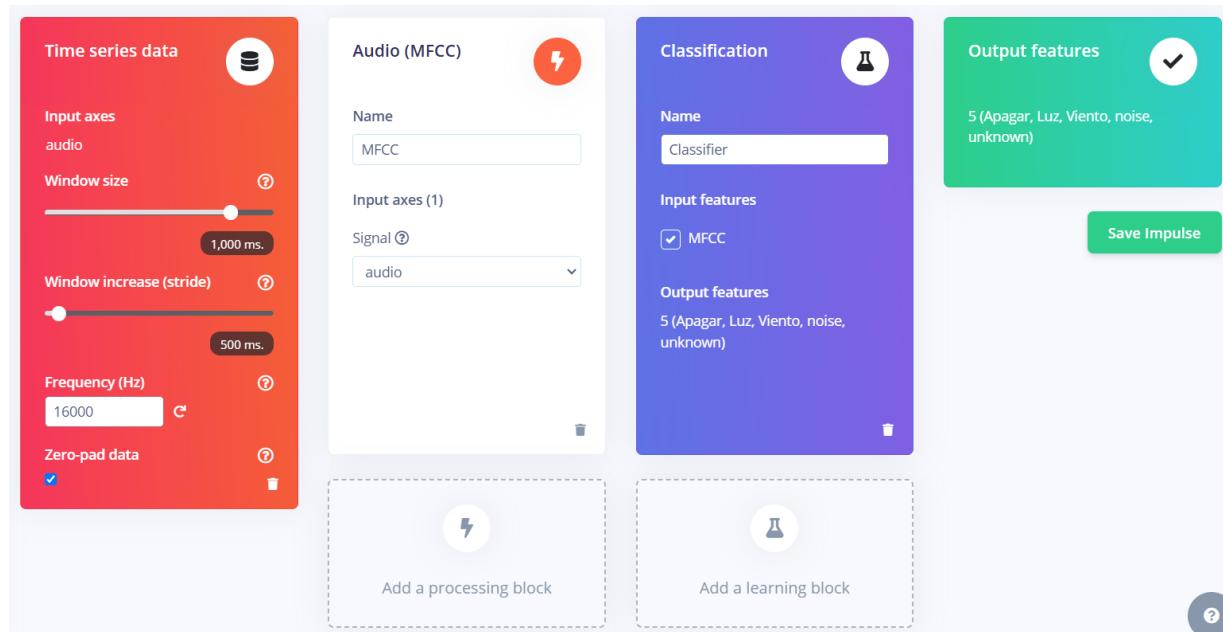


Figura 7: Crear impulso

Antes de entrenar el modelo está el procesamiento de los datos, aquí interesa que la plataforma sea capaz de agrupar y separar las muestras, la siguiente figura es un plano 2D, en el que se puede ver cómo las muestras son lo suficientemente buenas para que en la etapa de procesamiento estas sean distinguibles entre sí.

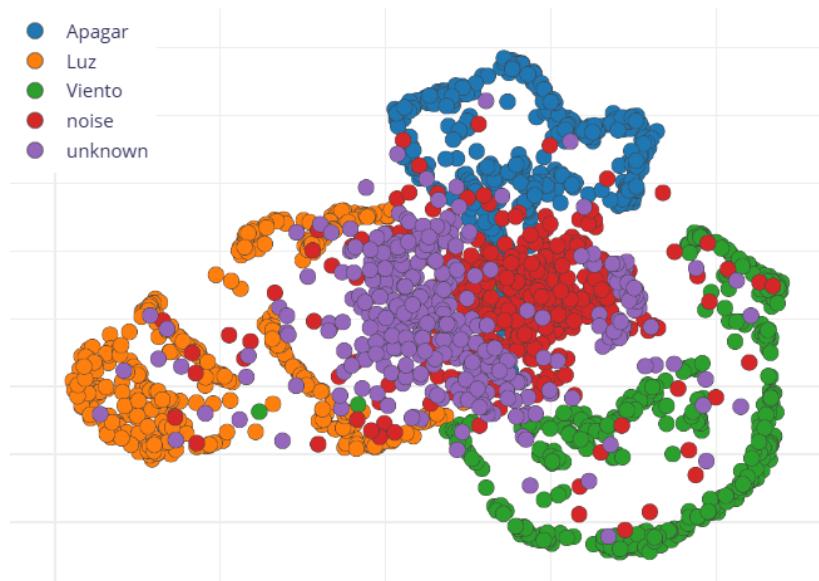


Figura 8: Procesamiento

Se usó la arquitectura que Edge Impulse recomienda para la red neural de estas características (clasificación de audio), esta tiene una capa de entrada, seis capas ocultas donde se identifican veinticuatro neuronas, y una capa de salida con cinco clases .

Neural network architecture

Architecture presets ⓘ [1D Convolutional \(Default\)](#) [2D Convolutional](#)

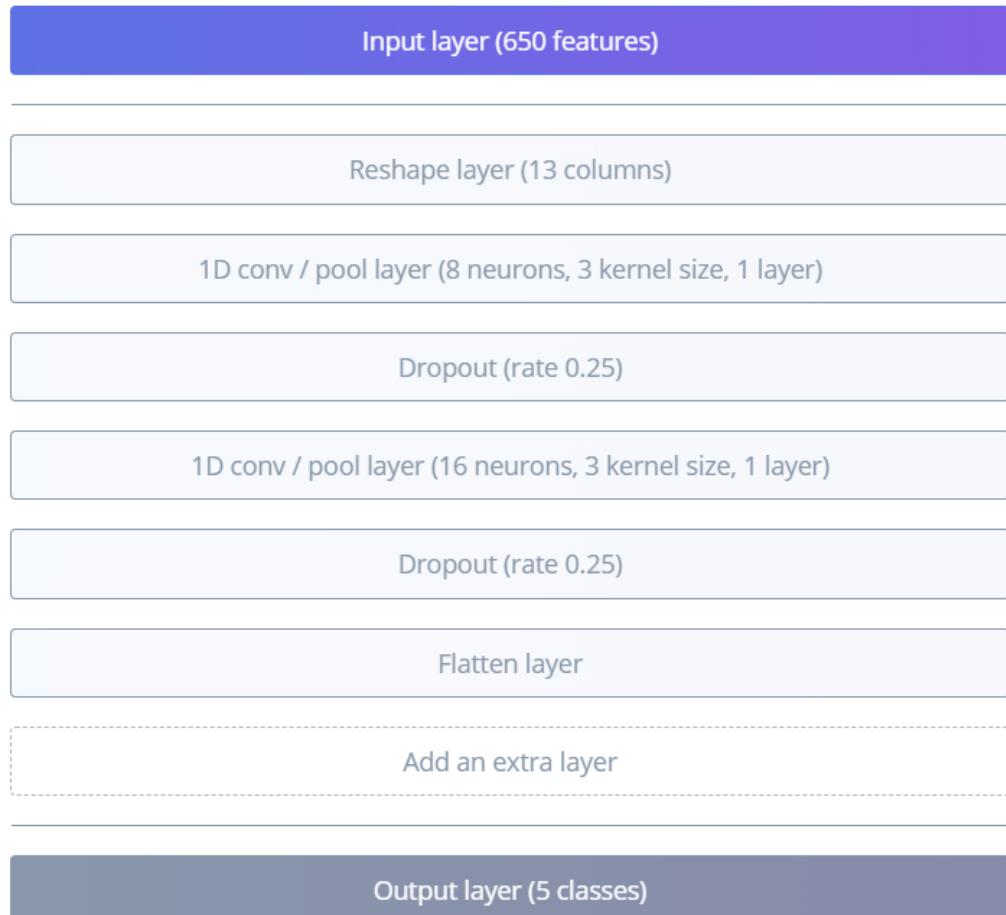


Figura 9: Arquitectura red neuronal

El resultado del entrenamiento es que la red tiene un 97% de precisión, y tal como se evidencia en la siguiente imagen la principal fuente de error está en que identifica ruido como desconocido y viceversa, lo cual no nos afecta.

Last training performance (validation set)



Figura 10: Resultado entrenamiento

La siguiente imagen muestra un plano con la clasificación que la red puede hacer con el set de datos de entrenamiento, los puntos verdes son clasificaciones correctas, los puntos rojos son incorrectos.

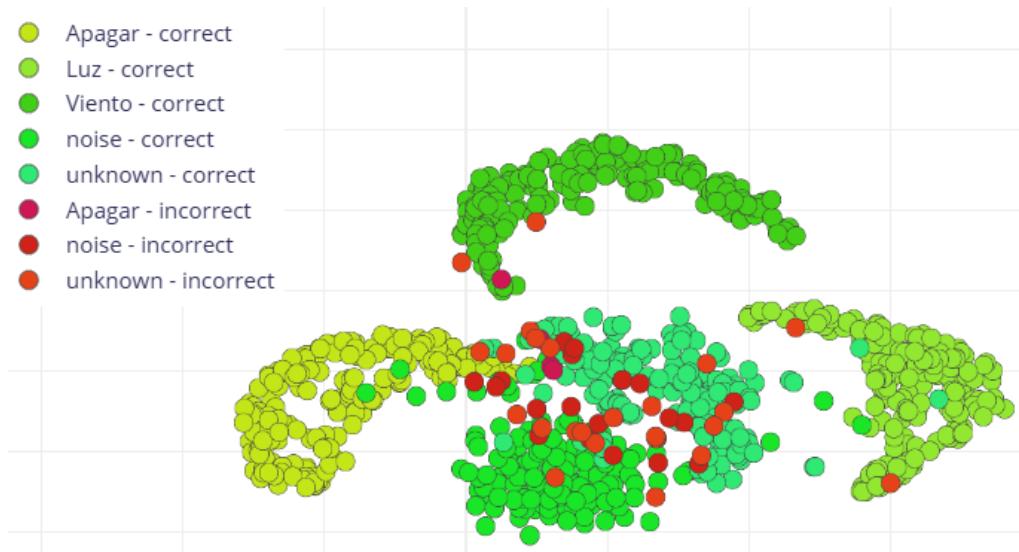


Figura 11: Clasificación con el set de entrenamiento

Otro resultado de la red es que hace inferencias cada 4ms, usa como máximo 3.8 kB de RAM y el tamaño del modelo 32.3 kB.



Figura 12: Desempeño en hardware

Para validar que la red funciona bien se reservó una parte (19 %) de las muestras que la red neuronal nunca ha visto y se le pide que las clasifique, esto muestra que en realidad la precisión de la red es del 93.5 %.

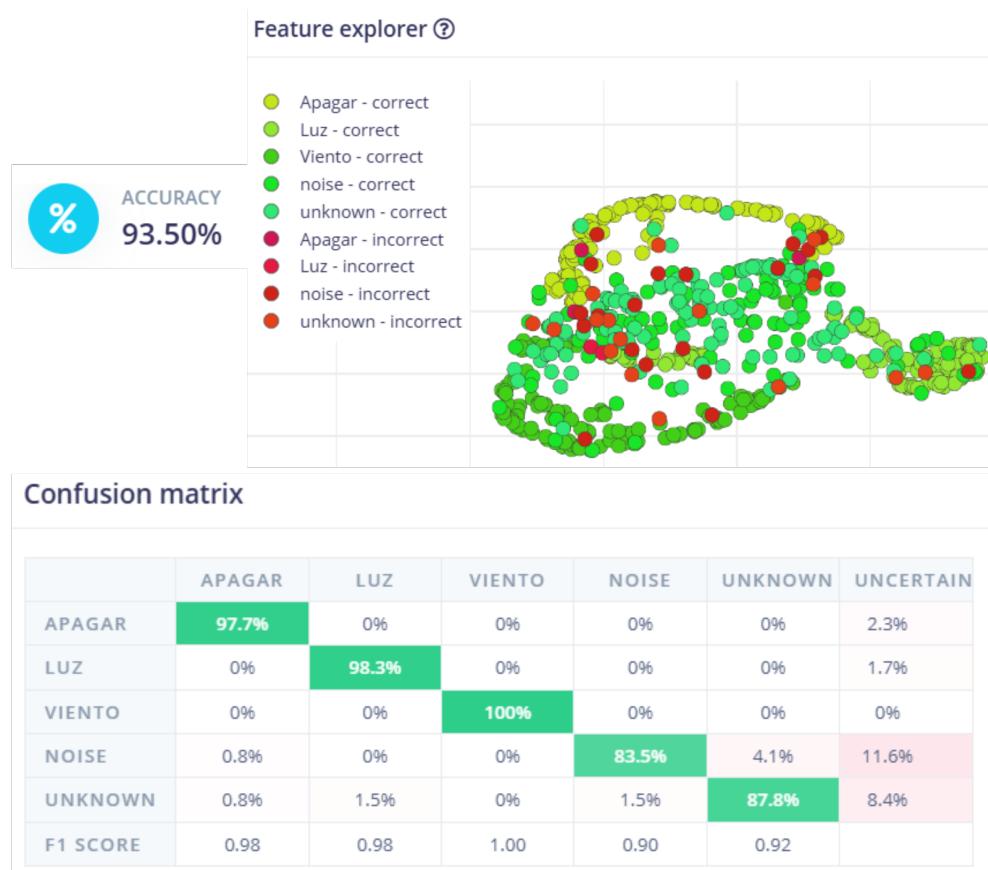


Figura 13: Validación del modelo

Para desplegar el modelo se debe exportar desde Edge Impulse como una librería, se instala en el Arduino IDE como una librería .zip, para utilizarla se tomó como base un ejemplo que ya viene incluido en la misma librería, llamado nano_ble33_sense_microphone_continuous, sin embargo, este solo permite ver la clasificación que el modelo da cuando se le habla.

The screenshot shows the TensorFlow Model Optimizer interface. At the top, it says "SELECTED DEPLOYMENT" and "Arduino library". Below that is the Arduino logo. A description states: "An Arduino library with examples that runs on most Arm-based Arduino development boards." In the middle section, titled "MODEL OPTIMIZATIONS", it says "Model optimizations can increase on-device performance but may reduce accuracy." On the left, there's a circular icon with a neural network symbol. Next to it, it says "EON™ Compiler" and "Same accuracy, 40% less RAM, 48% less ROM." To the right of this is a dropdown arrow. Below these sections is a table comparing different quantization levels:

Quantized (int8)	MFCC	CLASSIFIER	TOTAL
Selected ✓	266 ms.	4 ms.	270 ms.
RAM	15.4K	3.8K	15.4K
FLASH	-	32.3K	-
ACCURACY			93.82%

Figura 14: Despliegue del modelo

Para modificarlo se siguió lo explicado en [13], lo que se hizo fue tomar las tres salidas del modelo que corresponden a los comandos LUZ, VIENTO y APAGAR, si la predicción es alguno de esos comandos se envía un número al ESP, tal como se explicó antes.

```

1 // Send data to serial port based on prediction
2 if (result.classification[0].value > 0.85) { // APAGAR 2
3   digitalWrite(led_pin, HIGH);
4   Serial1.println(2);
5   Serial.println("Sent: " + String(2));
6 } else if (result.classification[1].value > 0.85) { // LUZ 1
7   digitalWrite(led_pin, HIGH);
8   Serial1.println(1);
9   Serial.println("Sent: " + String(1));
10 } else if (result.classification[2].value > 0.85) { // VIENTO 0
11   digitalWrite(led_pin, HIGH);
12   Serial1.println(0);

```

```
13     Serial.println("Sent: " + String(0));  
14 } else {  
15     digitalWrite(led_pin, LOW);  
16 }
```

El ESP8266 recibe estos datos por los pines seriales y se encarga de enviarlos a un servidor MQTT para que el receptor pueda tomar esos datos, para programar esta lógica se tomó como referencia lo descrito en [14], el ESP publica en el tópico eie/ucon/proy/luz, el broker MQTT utilizado es de acceso libre y se encuentra disponible en broker.emqx.io.

Para la segunda etapa la denominamos receptor y utiliza un ESP8266 para controlar el bombillo y el abanico. Para esto solo se utilizaron dos pines GPIO configurados como salida. Dichos pines se ponen en alto cuando se quieren cerrar los contactos de la bobina del relé.

Por medio de estos pines se hizo el manejo de los relés que controlan el abanico y el bombillo. Los relés utilizados traen resistencias y diodos para evitar daños por sobre corrientes. Además tienen un LED incorporado para indicar cuando los contactos del relé están cerrados.

Para alimentar el bombillo se hizo una conexión a 110V utilizando cable eléctrico y un enchufe. Inicialmente se utilizó cable Spt, sin embargo fue complicado realizar la conexión del relé ya que los contactos quedaban flojos. Para solucionar esto se utilizaron dos cables individuales. De esta forma, no había que doblar los cables y los contactos se mantenían fijos.

El ESP8266 de esta etapa capta los comandos de voz enviados por el ESP8266 de la primera etapa. Para esto se conecta a un servidor donde los comandos se envían por MQTT, dicho proceso se realizó con la ayuda de la guía en [14].

Los comandos se traducen en números, por ejemplo el comando “Viento” es igual a 0 y el comando “Luz” es igual a 1, por último el comando “Apagar” es igual a 2. A continuación se muestra el código que maneja estas condiciones.

```
1 // Toggle logic for each command  
2     if (message == "0") {
```

```
3     digitalWrite(Abanico, !digitalRead(Abanico)); // Toggle Abanico
4     Serial.println(digitalRead(Abanico) ? "Abanico ON" : "Abanico OFF");
5 } else if (message == "1") {
6     digitalWrite(Bombillo, !digitalRead(Bombillo)); // Toggle Bombillo
7     Serial.println(digitalRead(Bombillo) ? "Bombillo ON" : "Bombillo OFF");
8 } else if (message == "2") {
9     digitalWrite(Abanico, LOW);
10    digitalWrite(Bombillo, LOW);
11 } else {
12     Serial.println("Unknown command");
13 }
```

Una vez que el ESP recibe los comandos se activan los pines de los dispositivos que se deseen encender o apagar. El pin se pone en alto cuando el dispositivo se va a encender y cuando se ocupa apagar se pone en bajo. A continuación se muestra una imagen de la etapa de recepción.



Figura 15: Etapa de recepción.

Como se puede observar esta etapa está compuesta por un ESP8266 que maneja un bombillo que está conectado a 110V y un abanico de 12V. Ambos dispositivos se conectan a un relé para separar la etapa de potencia con la etapa electrónica. También se utilizó un cajón de metal para colocar la protoboard y realizar todas las conexiones sin que el sistema se viera desordenado.

4. Conclusiones

- El sistema combina inteligencia artificial y comunicación MQTT, logrando un control remoto eficaz para dispositivos del hogar.
- Se logró entrenar el modelo de IA con una precisión de 93.5 %, lo cuál es bastante positivo para el sistema desarrollado.
- El uso de hardware económico, junto con entornos de programación accesibles como Arduino IDE, permitió desarrollar un sistema de domótica efectivo.
- La implementación de comandos por voz simplifica la interacción con los dispositivos.
- El uso de Arduino Nano BLE 33 y ESP8266 potencia las capacidades del Arduino ya que por si solo no tiene conexión a internet.
- Gracias al protocolo MQTT, el sistema permite una comunicación en tiempo real, garantizando que los comandos se ejecuten de forma inmediata.
- Durante el desarrollo se encontraron diferentes retos relacionados con la precisión del modelo de IA y la sincronización entre dispositivos. Sin embargo, estos fueron solucionados con ajustes en la toma de muestras y modificación del código.

5. Recomendaciones

- Estudiar la documentación de EdgeImpulse sobre los dispositivos admitidos para escoger el más adecuado según la aplicación que se esté desarrollando.

- Completar los tutoriales de EdgeImpulse para manejar correctamente la herramienta y todas las opciones que esta permite.
- Capturar varios segundos con diferentes muestras y luego separarlas. Esto facilita y acelera en gran medida la toma de muestras.
- Tomar la mayor cantidad de muestras posibles para mejorar la precisión del modelo de IA. Si son muestras de voz es importante grabar diferentes voces para mejorar la respuesta del modelo.
- Se puede añadir seguridad a la comunicación MQTT para que la suscripción a los tópicos no sea pública y no se vaya a ver comprometido el sistema.

Referencias

- [1] Arduino. Nano 33 ble. [Online]. Available: <https://docs.arduino.cc/hardware/nano-33-ble/>
- [2] Espressif. Esp8266ex. [Online]. Available: <https://www.espressif.com/en/support/documents/technical-documents>
- [3] Randomnerdtutorials. Esp8266 pinout reference: Which gpio pins should you use? [Online]. Available: <https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>
- [4] AWS. ¿qué es iot (internet de las cosas)? Amazon Web Services. [Online]. Available: <https://aws.amazon.com/es/what-is/iot/#:~:text=El%20t%C3%A9rmino%20IoT%2C%20o%20Internet,como%20entre%20los%20propios%20dispositivos.>
- [5] HiveMQ. Mqtt mqtt 5 essentials. HiveMQ. [Online]. Available: <https://www.hivemq.com>
- [6] Arduino. Serial communication from arduino nano to esp8266. Arduino. [Online]. Available: <https://forum.arduino.cc/t/serial-communication-from-arduino-nano-to-esp8266/615666/5>
- [7] ChatGPT(OpenAI). (2024, November) Serial communication between arduino nano 33 ble sense and esp8266. Troubleshooting and solutions for serial communication issues, including

- SoftwareSerial pin mapping and debugging. [Online]. Available: <https://chatgpt.com/share/6747c242-b0fc-8011-861d-ccc4ea224a5d>
- [8] J. Jongboom. (2020, November) Responding to your voice. [Online]. Available: <https://docs.edgeimpulse.com/docs/tutorials/end-to-end-tutorials/responding-to-your-voice>
- [9] E. Impulse. (2020) Arduino nano 33 ble sense. [Online]. Available: <https://docs.edgeimpulse.com/docs/edge-ai-hardware/mcu/arduino-nano-33-ble-sense>
- [10] EdgeImpulse. (2024) Audio classification - keyword spotting. Apache 2.0. [Online]. Available: <https://studio.edgeimpulse.com/public/499022/latest>
- [11] E. Impulse. (2024) Audio mfcc. [Online]. Available: <https://docs.edgeimpulse.com/docs/edge-impulse-studio/processing-blocks/audio-mfcc>
- [12] EdgeImpulse. (2024) Classification. [Online]. Available: <https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/classification>
- [13] Digi-Key. (2004) How to do speech recognition with arduino | digi-key electronics. YouTube video. [Online]. Available: <https://studio.edgeimpulse.com/public/499022/latest>
- [14] D. Tao. (2024) Using mqtt on esp8266: A quick start guide. [Online]. Available: <https://www.emqx.com/en/blog/esp8266-connects-to-the-public-mqtt-broker>