Proyecto Final: Asistente Virtual Haciendo Uso de Arduino Nano 33 BLE con Reconocimiento de Voz y Conexión por Bluetooth

1st Ronny Granados
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica.
San José, Costa Rica
C03505
ronny.granadosperez@ucr.ac.cr

I. Introducción/Resumen

La automatización de tareas –en el contexto de un mundo tan acelerado como en el que vivimos– se ve grandemente favorecida con la existencia de los asistentes virtuales. Casi toda compañía ha implementado su propio tipo de inteligencia artificial en el formato de estos asistentes virtuales y los integra a sus dispositivos de una forma personificada, lo que permite una fácil comprensión de su uso que, a su vez, favorece en una mayor eficiecia en las tareas del diario vivir.

II. Objetivo General y Específicos

1. Realizar exitosamente distintas acciones mediante un control por voz que será implementado en el Arduino.
2. Hacer uso de los sensores con los que cuenta la placa, así como cualquier otro que se desee implementar y utilizar la información extraída para ser entregada al usuario como forma de respuesta por parte del asistente virtual.
3. Establecer exitosamente una conexión inalámbrica vía Bluetooth con un dispositivo móvil para poder así generar acciones desde dicho dispositivo.

III. Alcances

Se desea conseguir un Asistente Virtual que sea capaz de realizar múltiples tareas como encender y apagar luces, reproducir música y ofrecer la temperatura de la habitación y que sea capaz de identificar estos comandos por medio de la voz de un usuario y así poder interpretar, mediante un modelo de aprendizaje automatizado, cuál fue la palabra que se le dijo. Además de esto, debe ser capaz de ejecutar estas tareas pero enviándolas al Arduino desde un dispositivo conectado por Bluetooth.

IV. Justificación

Asistentes virtuales tales como Siri o Alexa surgen ante una sociedad globalizada y cada se vuelven más inteligentes en sus tareas y más autónomos.

Estos asistentes deben estar muy bien implementados e incorporar algoritmos complejos que permitan realizar las distintas tareas que soportan de manera eficiente.

Este proyecto es una exploración y un acercamiento al desarrollo de dispositivos IoT.

V. Marco Teórico

V-A. Tiny Machine Learning - Arduino Nano 33 BLE

El Arduino Nano por sí solo es un microcontrolador muy completo el cual posee múltiples sensores integrados como lo es un giroscopio de 9 ejes (LSM9DS1), conexión a Bluetooth Low Energý (BLE), sensor de gestos y proximidad (APDS9960), presión barométrica y temperatura (LPS22HB), micrófono integrado y un LED RGB integrado. Cuenta con un procesador de arquitectura 64 MHz Arm Cortex-M4F, 1 MB de Flash + 256 KB de RAM.

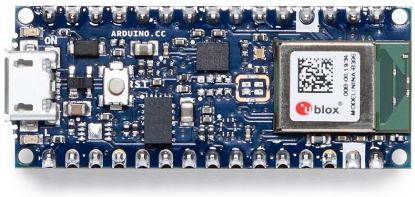

Figura 1. Arduino Nano 33 BLE Sense [1].

En lo que al Kit de machine learning se refiere, este posee una cámara (OV7670) y un tipo de rack en el cual se integran tanto el microprocesador como la cámara sin necesidad de una placa de pruebas externa. Este kit posee soporte para una plataforma de desarrollo en línea para entrenar modelos de machine learning con los datos obtenidos en tiempo real de

los distintos sensores ya mencionados. Para efectos de este proyecto solamente fue necesario el módulo de micrófono incorporado ya en la placa Nano.

Figura 2. Arduino Nano 33 BLE Sense Machine Learning Kit.

Parte de los objetivos de este proyecto es, también, poder leer información de los demás sensores, como lo es el LPS22HB, para medir presión atmosférica y temperatura. Dentro del código de Arduino se ofrece la opción de trabajar con una librería especializada en este sensor y permite hacer las lecturas de una forma muy sencilla: únicamente llamando la librería, inicializando el módulo y guardando la lectura en una variable para poder desplegarla:

```
#include <Arduino_LPS22HB.h>

void setup(){
    if (!BARO.begin()) {
        Serial.println("Failed to initialize!");
    while (1);
  }
}

void loop(){
    float temperature = BARO.readTemperature();
    Serial.print(temperature);
}
```

Este fue el único sensor, aparte del micrófono, que es requerido para este proyecto.

V-B. Edge Impulse

Como ya se introdujo, Edge Impulse es una plataforma de desarrollo para machine learning basada en la nube y que, además, posee soporte para el kit de Arduino utilizado. Es muy intuitivo de usar y permite llevar el aprendizaje automatizado a muchísimos más usuarios, pues los modelos de entrenamiento y de clasificación son seleccionados automáticamente por la plataforma, restándole al usuario únicamente recolectar los datos para entrenamiento y seguir, paso a paso, las instrucciones hasta llegar a generar la respectiva librería de Arduino.

Ya se se tenía cierto grado de familiarización con esta tecnología pues se tuvo que utilizar previamente en el curso y la firma de utilización es la misma; sin embargo, las aplicaciones son muy diferentes. En este caso, como se desea implementar un asistente virtual por medio de la voz, se utilizó la plataforma para entrenar un modelo de machine learning con varios comandos. Estos comandos sientan las bases de este proyecto, y estas son:

- Apagar
- Encender
- Hora
- Idle
- Play
- Temperatura

Cada uno de los elementos anteriores corresponde a una clase y requirieron de la recolección de datos de manera individual. La forma en que esto se realizó fue mediante la misma plataforma de Edge Impulse, en un intervalo de 20 segundos diciendo repetidas veces la frase que se deseara utilizar. Al final de la recolección se tuvieron un total 7 minutos aproximadamente de datos para entrenamiento; se intentó también que las muestras fueran lo más distintas entre sí que fuera posible para lograr un entrenamiento más robusto. Al final de esta etapa se obtuvo un modelo con una precisión del 98.7 % con la siguiente matriz de confusión:

	APAGAR	ENCENDE	HORA	IDLE	PLAY	STOP	TEMPERAT
APAGAR	92.3%	0%	0%	0%	0%	0%	7.7%
ENCENDER	0%	100%	0%	0%	0%	0%	0%
HORA	0%	0%	100%	0%	0%	0%	0%
IDLE	0%	0%	0%	100%	0%	0%	0%
PLAY	0%	0%	0%	0%	100%	0%	0%
STOP	0%	0%	0%	0%	0%	100%	0%
TEMPERATUF	0%	0%	0%	0%	0%	0%	100%
F1 SCORE	0.96	1.00	1.00	1.00	1.00	1.00	0.96

Figura 3. Matriz de confusión obtenida para el modelo de machine learning entrenado.

Esta matriz permite estudiar los aciertos y errores que tiene el modelo al momento de entrenarse [2]. Como se puede observar, entonces, en donde muestra más debilidad es en la clasificación de **Apagar**, tomándolo como **Temperatura**, pero el porcentaje de fallo es muy bajo por lo que no afecta en el comportamiento final del modelo, para la aplicación que se desea realizar en este escenario.

Una vez listo el modelo, se procede a exportar como una librería de Arduino, etapa en la que se desarrollará más adelante.

V-C. Bluetooth Low Energy

Como parte de los objetivos de este proyecto se encuentra poder establecer una conexión inalámbrica con algún otro dispositivo en una red cercana; para efectos de la demostración de este proyecto se utilizó el teléfono celular. La placa Arduino Nano cuenta con un módulo NINA-B306la el cual soporta Bluetooth 5 optimizado para bajo consumo de energía, lo que hace a esta placa muy eficiente energéticamente hablando. La forma en la que se inicializó este módulo dentro del código de Arduino fue basado en el ejemplo mostrado en [3]. La implementación requiere de múltiples líneas de código, por

lo cual no se mostrará aquí, sin embargo, se puede explicar brevemente su funcionamiento:

- Crear el servicio de Bluetooth e indicar que el dispositivo central va a leer y escribir datos.
- Dar un nombre a la placa para poder identificarla desde otros dispositivos mediante `BLE:setLocalName()`.
- Durante el tiempo que exista una conexión con otro dispositivo, la placa leerá todos los datos que sean enviados desde el celular desde la aplicación LightBlue.

V-C1. Aplicación LightBlue: Está aplicación se encuentra disponible para iOS y es muy sencilla de utilizar; se debe ir a la sección de periféricos disponibles y ubicar el Arduino con el nombre brindado o como **Arduino** pues algunas veces el nombre personalizado no se muestra.

A partir del momento en que la conexión exista se puede iniciar a enviar datos desde la aplicación; únicamente pueden enviarse números que serán convertidos a bits posteriormente, por lo tanto es necesaria una adecuada implementación de cómo van a manejarse estos datos. En el caso de este proyecto se trabajó por medio de máquinas de estado Estas máquinas de estado estaban numeradas para así poder enviar ese valor desde el celular. Los números escogidos para cada uno de los casos:

- **0x00**: APAGAR
- **0x01**: ENCENDER
- **0x04**: PLAY
- **0x05**: STOP

No existe la opción de ir al estado **IDLE** pues este estado existe por defecto, si no se encuentra en ninguno de los otros, se encuentra en **IDLE**.

V-D. Arduino IDE

En esta sección se explicará cómo está implementado el código de Arduino y por qué funciona.

Lo primero es la importación del modelo entrenado de machine learning, el cual es capaz de identificar por medio de lecturas del micrófono si la palabra o frase corresponde a una de las 7 categorías especificadas anteriormente. Este modelo debe ser cargado como una nueva librería de Arduino, por lo que incluye algunos ejemplos que varían dependiendo el dispositivo y el sensor que se quiera utilizar. Para est ecaso se utilizó **Nano BLE Sense** y el ejemplo llamado **Microphone**. Inicialmente este solo hace lecturas del micrófono e imprime el porcentaje de acierto, realizando de esta forma la clasificación.

Este archivo se modificó implementando una máquina de estados con las siguientes propiedades:

- Se inicia en el estado **IDLE**.
- Cuando se tenga una predicción mayor a 0.7 en cualquiera de los demás estados se saltará inmediatamente a este.
- El estado **ENCENDER** enciende el LED verde y vuelve a pasar a **IDLE** pero el LED queda encendido pues se maneja mediante una variable booleana.
- El estado **APAGAR** realiza lo mismo que **ENCENDER** pero la acción que realiza es apagar el LED y permanecer así hasta que la variable booleana sea modificada.
- Los estados **PLAY** y **STOP** no hacen nada, pero si se cae a uno de estos dos estados no se pasa a **IDLE** de una vez, se queda acá hasta que se indique al programa que debe ir al otro. Es decir: si estoy en **PLAY** debo ir a **STOP** para poder generar transición.

Se realizó esta máquina de estados dos veces; una de ellas fue con el modelo entrenado por medio de la voz y la otra para ser utilizada mediante el módulo Bluetooth, sin embargo, pese a que se implementó dos veces, funcionan por separado y la construcción de los saltos y condiciones es exactamente la misma. Esto quiere decir que el comportamiento será el mismo sin importar qué mecanismo se esté utilizando.

A este código le corresponde la tarea de enviar vía puerto serial el estado actual y el estado del LED, pues esto es suficiente para darle vida al asistente virtual mediante herramientas externas.

V-E. Python: interfaz gráfica y reproducción de música

Mediante un programa de Python fue posible leer los datos enviados por el Arduino por el puerto serial. Tal como se mencionó antes, mediante este canal solo se transmitía el estado actual y el estado del LED. El estado del LED es modificado directamente desde el código de Arduino; sin embargo, para los estados de **PLAY/STOP** es este código el que se encarga de operarlos. El programa consiste en dos archivos distintos: **spotify.py** y **main.py**. El primero se encarga de acceder a mi cuenta personal de Spotify mediante el acceso como administrador que se consigue desde este **LINK** y así permite tener un historial completo de las canciones reproducidas, playlists, búsqueda de artistas y demás. Anidado a esto, permite acceso a la última canción reproducida y poder reproducirla o pausarla, esto mediante el método `play_music()` y `pause_mausic()` respectivamente.

Por otra parte, el segundo archivo se encarga de inicializar y operar la interfaz gráfica mediante la librería TKinter y llamando a el archivo anterior para que, si se recibe uno de los dos comandos, aplique los cambios a la cuenta de Spotify. La interfaz gráfica es muy sencilla: cuenta con una línea que indica el estado actual, otra indicando cómo se encuentra el LED y otra indicando la canción que se está reproduciendo.

VI. Desarrollo/Análisis de Resultados

Las tecnologías utilizadas fueron discutidas en la sección anterior, de modo que la integración de estas es lo que se tratará en la sección de resultados y, en general, cómo fue el desempeño de la aplicación final.

Pese a que el resultado de precisión del modelo fue muy alto, las muestras se tomaron en periodos de 1 segundo lo cual afectó a la posterior implementación cuando se probó en el Arduino en la vida real. Como se entrenó el modelo con muestras de este tamaño, el Arduino hacía lecturas cada segundo y este, al ser un periodo muy corto, afectaba la forma en que el modelo interpretaba las frases, muchas veces interpretándolas de manera incorrecta. Esto no tiene nada que ver con el modelo o el Arduino como tal, sino en la forma en que se realizó el entrenamiento. La solución a esto era volver

a entrenar el modelo, recolectando los datos de nuevo, pero este proceso es muy largo y tedioso y al punto en que se encontraba el proyecto y la cercanía de la fecha de entrega se decidió trabajar con lo que ya se tenía.

Debido a que muchas veces el micrófono capturaba una palabra y el modelo lo interpretaba como otra distinta y esto muchas veces hacía que el programa de Python se cayera. Esto ocurría únicamente con los estados de reproducción y pausa de la música, pues si ya se estaba reproduciendo y se volvía a llamar el mismo comando se generaba un error y lo mismo con el modo de pausa. El siguiente pseudo-código pertenece al archivo **main.py** y muestra la forma en la que este problema fue tratado:

```
if state == "PLAY" and past_state != "PLAY":
        play_music()
elif state == "PLAY" and past_state == "PLAY":
        pass
elif state == "STOP" and past_state != "STOP":
        stop_music()
elif state == "STOP" and past_state == "STOP":
        pass
else:
        pass
```

La implementación de este código evitó que el programa se cayera de forma no intencional y permitir un flujo de trabajo continuo. Lo que se hizo fue que si el estado anterior era distinto a **PLAY** o **STOP**, entonces se reproducía o detenía la música, según correspondiese, de lo contrario, omitía la acción.

Por otra parte, con el módulo de bluetooth la comunicación era mucho mejor, se conectaba sin problemas al celular pero, al momento de escribir, había que enviar el dato dos veces para que pudiera interpretar el estado que realmente se quería formar en el dispositivo, fuera de esto, no hubo mayor inconveniente.

Por último, se deseaba poder leer información de los sensores y poder mostrarla en la interfaz gráfica, tal como con la música, pero al momento de llamar la lectura del sensor LPS22HB de temperatura al puerto serial el resultado era de 0 en todo momento. El posible motivo de esto puede ser que el MCU estaba leyendo datos tanto del micrófono como del sensor y que al momento de querer llevar todas al puerto serial existía un conflicto debido al canal que usaban para transmitirse. Esto es solamente una de las posibles causas del fallo de la lectura de la temperatura. Existían porciones del código del modelo entrenado que no se comprendía a profundidad por lo que se desconoce si alguna de las líneas bloqueaba o ignoraba la lectura de otros sensores.

En lo que a resultados del Arduino real se refiere, no existe ninguna demostración visual que aporte mayor relevancia a esta sección pues lo único que se realizaba con la placa era el encendido y el apagado del LED color verde. Este solamente se encendía –a como ya se ha explicado– al enviarlo al estado **ENCENDER**. Por lo tanto, el resultado más evidente es la interfaz gráfica en funcionamiento. Las siguientes figuras fueron tomadas en tiempo real mientras el Asistente Virtual estaba en funcionamiento:

Figura 4. Interfaz gráfica: estado **PLAY**.

Figura 5. Interfaz gráfica: estado **STOP**.

Al momento de cerrar la interfaz gráfica el Asistente Virtual seguía enviando datos por le puerto serial, pero los únicos estados que funcionarían serían el de **ENCENDER** y **APAGAR** pues para los estados correspondientes a la música se debe tener el acceso al programa de Python pues es desde aquí donde se accede a la cuenta personal.

VII. Conclusiones y Recomendaciones

A manera de conclusión se debe mencionar que el resultado final de este proyecto genera un gran estado de satisfacción pues, pese a haber objetivos que no pudieron ser alcanzados, la integración de todos los módulos de este Asistente Virtual fue un éxito. Un entrenamiento del modelo de machine learning con una ventana de muestras más grande (1.5-2 segundos) sería suficiente para permitirle al Arduino leer exactamente la palabra que se desea. Una recomendación dada pro el profesor fue que se recolectaran muestras de todos los estados cuando se tuviese ruido de fondo alto para hacerlo aún más robusto.

Poniendo este modelo estrenado correctamente a funcionar con el Arduino podría abrir una puerta a un sinfín de tareas que se podrían cumplir con este asistente. Sin duda fue un muy buen primer acercamiento a lo que es el desarollo e implementación de dispositivos IoT.

REFERENCIAS

[1] Arduino Nano 33 BLE with headers. https://store.arduino.cc/products/arduino-nano-33-ble-with-headers, 2024.

[2] Barrios, J. La matriz de confusión y sus métricas. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/, 2019.

[3] Troya, F. Controlling RGB LED Through Bluetooth. https://docs.arduino.cc/tutorials/nano-33-ble-sense/bluetooth/, 2022.