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

Proyecto final

Ronny Granados Pérez - C03505

# Repaso de Objetivos

- Realizar exitosamente distintas acciones mediante un control por voz que será implementado en el Arduino.
- 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.
- Establecer exitosamente una conexión inalámbrica vía Bluetooth con un dispositivo móvil para poder así generar acciones desde dicho dispositivo.

# Justificación

Ante una sociedad globalizada surge la necesidad de completar tareas de forma más eficiente, de ahí los asistentes virtuales.

Este proyecto busca implementar un prototipo de asistente virtual con el fin de comprender todo el trabajo que conlleva y comprender mejor el desarrollo IoT.

# Tecnologías Utilizadas

- **Arduino Tiny Machine Learning Kit**
- **Edge Impulse**
- **Conexión vía Bluetooth**
- **App LightBlue**
- **Python (TKinter y Spotipy)**

# Edge Impulse

Se consiguió entrenar el modelo de **machine learning** para 7 clases:

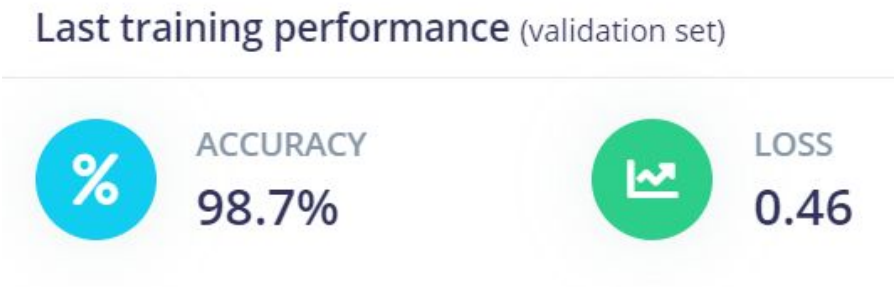
- Apagar, Encender, Hora, Idle, Play, Stop y Temperatura

Se utilizaron **únicamente 5** de esas clases.

# Edge Impulse: Construcción del Modelo

Se utilizó un **algoritmo de clasificación** para poder separar las clases de las muestras.

Para entrenar el modelo: **red neuronal convolucional**



# Arduino IDE

Se importa el modelo entrenado como una **librería**. Esta librería contiene ejemplos de aplicación para el uso del **micrófono**.

Basado en este código se inició la construcción de la **máquina de estados** que regirá el programa.

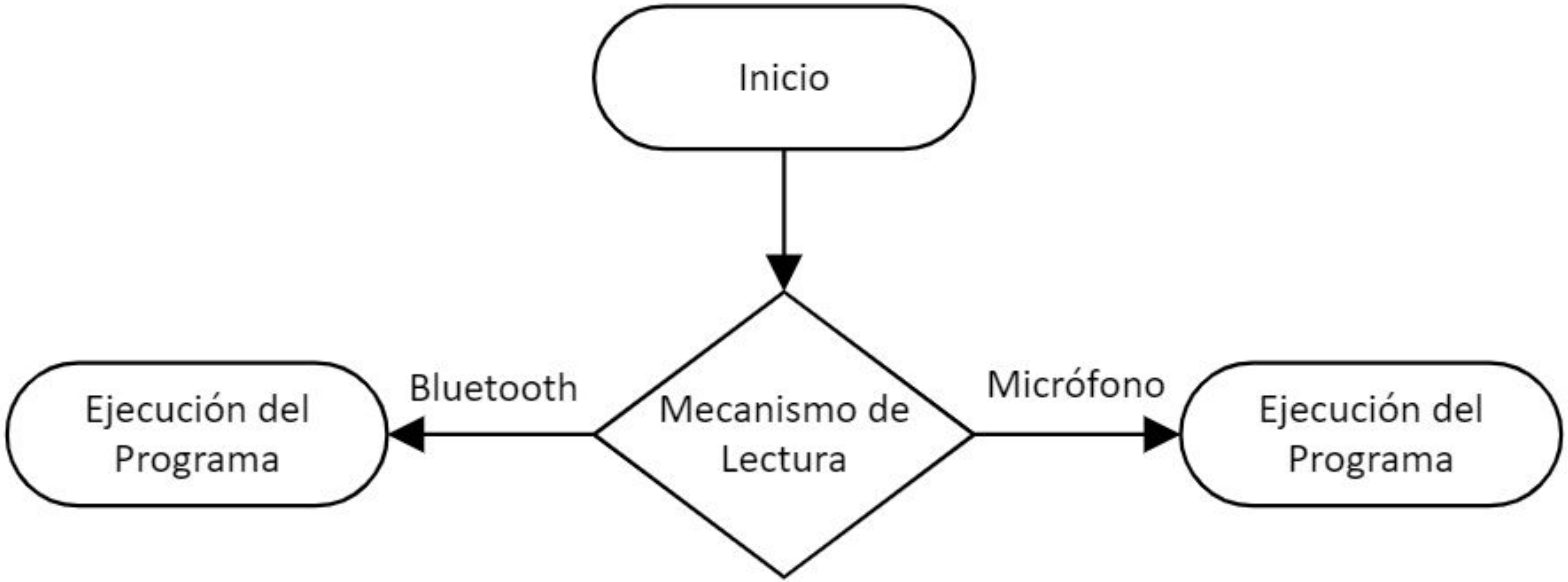
Cinco estados posibles: **Idle, Encender, Apagar, Play, Stop**

# Arduino IDE

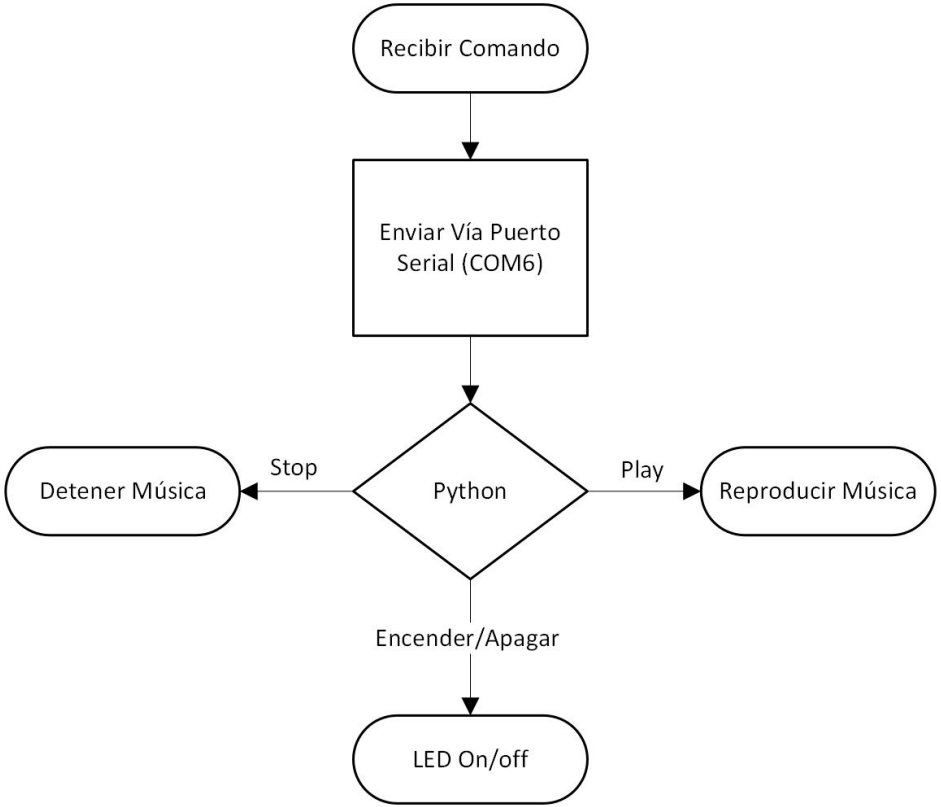
Este código realiza **4 tareas esenciales** para el adecuado funcionamiento del programa:

- Contención del **modelo entrenado**.
- Llevar adecuado flujo de trabajo mediante **FSM**
- **Apagar/Encender** el LED
- **Enviar datos** por el puerto serial.

# Flujo del Programa



# FSM


Recibir Comando
Enviar Vía Puerto Serial (COM6)
Python
Stop
Detener Música
Play
Reproducir Música
Encender/Apagar
LED On/off

# Bluetooth Low Energy (BLE)

Protocolo de **baja energía** que permite la conexión **inalámbrica** entre el Arduino y algún dispositivo externo vía Bluetooth.

Permite **escribir** o **leer** datos a través del puerto serial.

En este caso solo **escribe** al Arduino, alternando entre las **5 clases** mencionadas anteriormente.

# LightBlue

Aplicación disponible para iOS que facilita la comunicación con el **Arduino**.

Mediante esta app se pueden **escribir** los estados a la placa.

- 0 - **APAGAR**
- 1 - **ENCENDER**
- 4 - **PLAY**
- 5 - **STOP**

# Puerto Serial

El programa cargado a la placa envía únicamente 2 datos al puerto serial:

- **Estado actual**
- **Estado del LED**

```
Estado: IDLE
   LED: 0
```

# Python

Mediante 2 programas de Python se pudo **dar vida** al Asistente Virtual.

**Archivo principal**: lee el puerto serial, construye el GUI y sigue un flujo de estados similar.

**Archivo de Spotify:** accede a una cuenta personal de spotify, reproduce o detiene la música.

# Interfaz Gráfica en Python

# Interfaz Gráfica en Python

# Resultados: Control por Voz

El control por **voz** hace **lecturas** del micrófono en intervalos de **1 segundo**.

Presenta problemas al hacer las lecturas pero **no al interpretarlas**.

**Origen del problema:** las muestras de Edge Impulse (378 muestras) fueron particionadas de ese tamaño. Dentro del programa no se podría modificar.

# Conclusiones y Recomendaciones

Programa **muy bien implementado** con fallos en la captura de la voz que vienen directamente de **Edge Impulse**.

La integración con Python y Spotify dentro de la interfaz gráfica muestra los alcances que pueden tener este tipo de proyectos.

**Recomendación:** volver a entrenar el modelo con intervalos de tiempo más largos (~ 2 segundos).

# Objetivos No Alcanzados

Parte de este proyecto era leer información del sensor **`LPS22HB`** cuando fuera llamado el comando **Temperatura**.

No se consiguió la integración de estos módulos. Una posible causa es que ambos utilizaban el mismo canal para leer información de la placa (micrófono y sensor).

En tiempo real, se deseaba obtener la hora del sistema, sin embargo esto no se realizó.

# Gracias!!