Interfaz gestual para interacción con música generativa
Gestural interface for interaction with generative music

Trabajo de Fin de Máster
Curso 2022–2023

Autor
Manuel Muriel Molina

Director
Jose Ignacio Gómez Pérez
Jaime Sánchez Hernández

Máster en Internet de las Cosas
Facultad de Informática
Universidad Complutense de Madrid

Interfaz gestual para interacción con música generativa
Gestural interface for interaction with generative music

Trabajo de Fin de Máster en Internet de las Cosas
Departamento de Arquitectura de computadores

Autor
Manuel Muriel Molina

Director
Jose Ignacio Gómez Pérez
Jaime Sánchez Hernández

Convocatoria: *Septiembre 2023*
Calificación: *9*

Máster en Internet de las Cosas
Facultad de Informática
Universidad Complutense de Madrid

5 de septiembre de 2023

Agradecimientos

Aprovecho esta sección para agradecer principalmente a mi familia por animarme siempre a seguir adelante y por hacer posible mis estudios de máster tan lejos de casa.

Agradecer también a mis compañeros y compañeras de clase por hacer este año más ameno y por compartir las alegrías y las frustraciones tanto dentro como fuera del aula.

Gracias a la empresa Bosch por acogerme como alumno en prácticas durante 5 meses, en concreto a mis tutores de empresa por estar pendientes de mí y facilitarme lo máximo posible mi estancia en la empresa.

Por último agradecer a mis tutores, Nacho y Jaime por facilitar el desarrollo de este proyecto y por la ayuda brindada frente a los problemas que iban surgiendo.

Resumen

Interfaz gestual para interacción con música generativa

En este proyecto hemos desarrollado un sistema de música interactiva controlado mediante la posición de la mano vista por una cámara y los gestos que se hagan en el aire. La posición de la mano, detectada por la cámara y procesada con MediaPipe, nos permitirá interactuar con diferentes parámetros como el tono, el volumen o la distribución aplicada. Los gestos que hagamos con la mano al completo, detectados con la placa Nicla Sense ME, nos permiten pausar la música o reanudarla.

La música sobre la cual haremos esta interacción es una pieza de música generativa, es decir, una composición producida en tiempo real de modo algorítmico, de acuerdo con un conjunto específico de reglas lógicas de control. Partiendo de un código de música generativa seremos capaces de cambiar parámetros de dicho código para modificar el sonido que se genera.

Este proyecto pretende acercar las aplicaciones del Internet de las Cosas a un ámbito del entretenimiento, viendo así las posibilidades que puede abrir esta tecnología a la hora de aplicarlo a la música así como desarrollar nuestra creatividad al interactuar con ella de diversas formas. Se trata principalmente de un trabajo exploratorio para investigar sobre dicha interacción musical utilizando elementos y herramientas vistas durante este máster.

Aunque el objetivo principal sea el entretenimiento también sirve como demostración del potencial de las diferentes herramientas usadas como la placa Nicla Sense ME, la cual es capaz de realizar la inferencia de un gesto previamente entrenado en cuestión de milisegundos, poniendo en valor la importancia de la computación en el borde. Además el eje central de este proyecto es una Raspberry Pi 4, un mini ordenador económico y portable capaz de soportar un sistema IoT complejo.

Palabras clave

IoT, Música Generativa, Nicla Sense ME, ESP32, MediaPipe, Python, Edge Impulse, Interfaz Gestual

Abstract

Gestural interface for interaction with generative music

In this project we have developed an interactive music system controlled by the position of the hand seen by a camera and the gestures made in the air. The position of the hand, detected by the camera and processed with MediaPipe, will allow us to interact with different parameters such as tone, volume or distribution. The gestures we make with the whole hand, detected with the Nicla Sense ME board, allow us to pause the music or resume it.

The music on which we will do this interaction is a piece of generative music, that is, a composition produced in real time in an algorithmic way, according to a specific set of logical control rules. Starting from a generative music code we will be able to change parameters of this code to modify the sound that is generated.

This project aims to bring the applications of the Internet of Things to a field of entertainment, thus seeing the possibilities that this technology can bring when applied to music and develop our creativity as we interact with it in various ways. It is mainly an exploratory work to investigate about this musical interaction using elements and tools seen during this master.

Although the main objective is entertainment, it also serves as a demonstration of the potential of the different tools used such as the Nicla Sense ME board, which is able to perform the inference of a previously trained gesture in a matter of milliseconds, highlighting the importance of edge computing. In addition, the backbone of this project is a Raspberry Pi 4, an inexpensive and portable mini-computer capable of supporting a complex IoT system.

Keywords

IoT, Generative Music, Nicla Sense ME, ESP32, MediaPipe, Python, Edge Impulse, Gestural Interface

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Veo la música como arquitectura fluida"
— Joni Mitchell

El entretenimiento en cualquiera de sus formas es una actividad que nos puede proporcionar diversión y distracción, ayudándonos a relajarnos y evadirnos de los problemas cotidianos que debemos afrontar [30]. Una de las formas de entretenimiento más populares es la música, que es el arte de combinar sonidos y silencios de manera armoniosa y expresiva. La música puede incluso llegar a definirnos como personas ya que a través de ella podemos reflejar nuestros gustos y emociones. Además nos influye en nuestro estado de ánimo, pues puede hacernos sentir felices, tranquilos o motivados.

Una parte muy importante de la música es que puede mejorar nuestro rendimiento cognitivo, ya que estimula el cerebro y favorece la memoria, la atención y la creatividad. Este punto es en el que queremos hacer más hincapié ya que este proyecto combina la música con la interacción por parte del usuario, favoreciendo la creatividad.

En los últimos años, la tecnología y la computación han provocado una revolución en cuanto a las opciones de las que disponemos para entretenernos [13], ofreciéndonos nuevas posibilidades y experiencias. Por ejemplo los efectos visuales han permitido crear imágenes cada vez más realistas así como simular cosas imposibles como monstruos, explosiones o escenarios fantásticos. El streaming nos permite acceder a una gran variedad de contenidos audiovisuales, como películas, series o música a través de internet y sin necesidad de descargarlos. La computación ha facilitado el desarrollo de aplicaciones, servicios y dispositivos que permiten acceder a estos contenidos de forma personalizada y flexible, mientras que antiguamente la única forma de disfrutar de algunas de estas formas de entretenimiento era una tarea más compleja.

La relación entre la música y la tecnología es muy estrecha, tanto que esta última ha influido en la evolución de la música a lo largo de la historia [2]. La tecnología ha afectado a la música desde la creación y la producción hasta su distribución y consumo. Respecto a la creación musical, la tecnología ha permitido el desarrollo de nuevos instrumentos musicales, como los sintetizadores, las cajas de ritmos, los samplers o los controladores MIDI, que amplían las posibilidades sonoras y creativas de los músicos. La tecnología también ha facilitado el uso de programas informáticos y aplicaciones móviles que permiten componer, editar, mezclar y masterizar música de forma digital. Cada vez es más sencillo para una persona común adentrarse en el mundo de la música y usar herramientas que anteriormente estaban destinadas estrictamente a profesionales del sector, acercando así la creación musical a mucha más gente.

En conclusión, la tecnología y la música están íntimamente relacionadas y se retroali-

mentan mutuamente. Uniendo la música tal y como la conocemos con herramientas IoT como las placas de desarrollo que veremos obtenemos un proyecto novedoso que nos hace partícipes en la reproducción de la música.

1.1. Motivación

Como sabemos el entretenimiento es una parte crucial en la sociedad, por lo tanto así como la sociedad va cambiando con el tiempo el entretenimiento tampoco es algo estático, sino que se adapta a los cambios sociales, culturales, tecnológicos y económicos que se producen en el mundo.

La sociedad actual se caracteriza por ser dinámica, compleja y diversa. Los avances tecnológicos o la globalización son algunos de los factores que influyen en la forma en que las personas se entretienen [14]. Debido a esto vemos necesario crear nuevas formas de entretenimiento que respondan a las necesidades, intereses y expectativas de los diferentes públicos.

Estas nuevas formas de entretenimiento deben ser innovadoras y creativas, es decir, se buscan en ellas experiencias originales, sorprendentes y estimulantes para el público. Otro punto clave es que sean interactivas y participativas, que permitan al público ser protagonista de modo que le permitan elegir, opinar y colaborar en solitario o junto a más gente en el proceso de entretenimiento. Esto se puede ver en plataformas como Netflix, que incluyen películas interactivas en las que el espectador puede tomar decisiones que cambian el destino del protagonista [11]. Además gran parte del entretenimiento pretende ser educativo y cultural, contribuyendo así al desarrollo personal y social del público, fomentando el aprendizaje, la reflexión y el respeto por la diversidad. Ciertos contenidos de entretenimiento nos pueden acercar a ámbitos que de otro modo no hubiéramos conocido o simplemente no nos hubieran interesado.

Teniendo todo lo anterior en mente pensamos que crear nuevas formas de entretenimiento es una necesidad en una sociedad cambiante continuamente, siendo capaz de adaptarse a tales cambios y ofrecer experiencias satisfactorias para el público. Es por ello que en este trabajo queremos aportar una forma de entretenimiento basada en la interacción directa del usuario con la música que se genera en tiempo real, uniendo esto además a los conocimientos sobre tecnología y dispositivos IoT adquiridos durante este máster y durante las prácticas en empresa, donde he estado mucho más cerca que la tecnología que se usa y he tenido la oportunidad de usar hardware como la Nicla Sense ME. De esta forma también se pretende acercar a los usuarios a la música generativa así como a dispositivos IoT de una manera interactiva y entretenida.

1.2. Objetivos

El objetivo general de este proyecto es disponer de un sistema interactivo junto a la generación de música en tiempo real, controlada por la posición y los gestos que se realicen con la mano. A partir de este objetivo nos hemos propuesto, para llevar a cabo este proyecto, los siguientes objetivos específicos:

- Detección de un gesto dibujado en el aire usando la placa Nicla Sense ME, explotando su capacidad lo máximo posible al hacer la inferencia en la propia placa.
- Detección y seguimiento de la mano usando MediaPipe, realizando esto con la potencia limitada de la Raspberry.

- Reproducción de música generativa a partir de unos parámetros de entrada haciendo uso del lenguaje Python y herramientas como Pyo.
- Interacción del usuario con la música generada permitiendo modificar su ejecución en tiempo real.
- Uso de Raspberry Pi como ordenador principal del proyecto, beneficiándonos de su tamaño reducido y su portabilidad.

1.3. Plan de trabajo

Partiendo de los objetivos que se han descrito en la sección anterior los cuales se revisaron con los tutores de este trabajo se planteó un plan de trabajo necesario para conseguir dichos objetivos.

1.3.1. Exploración de las herramientas hardware disponibles

Para desarrollar este proyecto se debe investigar el hardware del que disponemos, entre el que se encuentran las placas de desarrollo de Bosch, como la Nicla Sense ME, otras placas utilizadas en el máster como la ESP32 o elementos mas potentes sobre los que ejecutar la mayor parte del software como la Raspberry Pi 4.

1.3.2. Investigación de trabajos previos

Una vez que sabemos qué podemos usar para el desarrollo se investigan otros proyectos que usen dicho hardware y se relacionen con nuestra visión. De este modo partimos de una base la cual nos ayuda con la elección del software, ya que hay una gran cantidad de herramientas disponibles.

1.3.3. Exploración del software disponible

Sabiendo las necesidades de este proyecto y teniendo en cuenta los proyectos ya desarrollados debemos hacer una investigación y elección del software que usaremos para programar las herramientas hardware. Necesitamos un software que sea capaz de reproducir una melodía generativa con la cual se pueda interactuar y que además permita una fácil comunicación con el resto de dispositivos involucrados.

1.3.4. Desarrollo de música generativa

Una vez que tenemos las herramientas hardware y software elegidas funcionando de manera conjunta podemos pasar a la creación de una pieza musical generativa. Esta melodía será infinita e irá cambiando con el tiempo respecto a unas reglas establecidas en el código. Además debe permitir modificar ciertos parámetros mientras se está ejecutando.

1.3.5. Desarrollo de interfaz gestual

Por último debemos crear una interfaz gestual la cual envía información al código que reproduce la melodía generativa y cambie su comportamiento. Esta interfaz se basará en la posición de la mano respecto a una cámara y en los gestos que se dibujen en el aire.

1.4. Estructura del documento

En esta sección veremos brevemente como se estructura este documento, haciendo una visión general de cada uno de los capítulos que lo conforman.

- **Introducción.** En este capítulo se incluye la motivación del proyecto, los objetivos, el plan de trabajo y la estructura del documento.
- **Estado de la cuestión.** Se revisan proyectos anteriores que se relacionan directamente con los conceptos de música generativa e interacción con ella, así como la detección de gestos.
- **Arquitectura hardware del sistema.** Definición de los elementos hardware que conforman este proyecto, describiéndolos y viendo las posibilidades que ofrecen al usarlos en el desarrollo.
- **Arquitectura software del proyecto.** Definición de las herramientas software que se han usado en el desarrollo del proyecto. Se detalla en qué dispositivo se ejecutan y para qué se usan.
- **Desarrollo del proyecto.** En este capítulo se detalla paso a paso como ha sido el desarrollo del proyecto, integrando las herramientas hardware y software vistas anteriormente, desde su conexión hasta el uso en detalle de cada una.
- **Conclusiones y Trabajo Futuro:** En el último capítulo se hace un repaso de los logros obtenidos en este proyecto, a la vez que se obtienen las conclusiones, además se revisa el trabajo futuro al que puede dar lugar.

1.5. Disponibilidad del proyecto

La totalidad del código desarrollado para este proyecto se encuentra disponible en un repositorio de Github [9] público creado para este fin. Además se añaden vídeos mostrando el proyecto en funcionamiento.

Capítulo 2

Estado de la Cuestión

En este capítulo veremos casos de uso relacionados con el desarrollo de este proyecto, los cuales nos han servido como base o como medio para conocer herramientas útiles en nuestro proyecto. Para ello se hizo una investigación de casos de uso anteriores relacionados con la detección de gestos, en concreto con la placa de desarrollo Nicla Sense ME. En cuanto a la música generativa la investigación nos sirvió para conocer el software disponible que se podía usar para generar dicha música. Por último nos centramos en proyectos de interacción musical sobre todo basados en la unión de tecnología y música.

Un proyecto que sirvió como base para la detección de gestos usando la placa Nicla Sense ME [23] es el realizado por Justin Lutz [29]. En este proyecto se describe un caso de uso en el cual la placa se integra en la manga de una chaqueta (lo más cerca posible de la muñeca). La finalidad de este proyecto es que se use en sesiones de senderismo y la placa nos sirva para diversas acciones relacionadas con esta actividad. Al dibujar una 'C' en el aire se añade un marcador en el mapa de la aplicación móvil que incorpora el proyecto, también posee otras funciones como monitorizar la presión atmosférica para avisarnos de una posible tormenta. Este proyecto nos sirve de base para aprender a usar la plataforma Edge Impulse [18], una parte de clave de nuestro proyecto al permitirnos entrenar los diferentes gestos y proporcionar una librería de Arduino para que se realice la inferencia en la propia Nicla Sense ME.

Como nuestro propósito es interactuar en tiempo real con la interpretación de una pieza musical, hemos encontrado muy adecuado el uso de música algorítmica, en particular la música generativa. Para ello se han investigado proyectos de música generativa o procedural.

En un artículo de 1996 sobre una charla del artista Brian Eno [25] podemos encontrar una reflexión sobre la música generativa así como la historia del artista creando este tipo de música. En esta charla Brian define la música generativa como *una música que se crea a partir de unos pocos elementos básicos que se combinan y se transforman según unas reglas simples, dando lugar a una variedad prácticamente infinita de resultados.* Según Brian la música generativa es una música que no tiene un principio ni un final fijos, sino que se adapta al contexto y al oyente, creando una experiencia única cada vez. Brian también habla de sus propios experimentos con la música generativa, como sus obras ambientales (por ejemplo, Music for Airports [10]), donde utiliza grabaciones con sonidos sutiles y repetitivos que se mezclan al azar, creando un fondo sonoro prácticamente infinito. Brian dice que su intención con estas obras es crear una música que no sea un objeto fijo y acabado, sino un proceso continuo y vivo, que pueda interactuar con el entorno.

En cuanto a investigación dentro del campo de la música generativa podemos destacar la trayectoria del grupo de investigación en Tecnología Musical de la Universidad Pompeu Fabra. Junto a su director Xavier Serra el grupo lleva desde 1994 [1] especializándose en las tecnologías digitales relacionadas con el sonido y la música. Con más de cuarenta investigadores procedentes de ámbitos muy diferentes y complementarios, es un grupo con una gran actividad investigadora tanto a nivel nacional como internacional. En sus proyectos, al igual que en el nuestro, se destaca la uso de la tecnología más actual junto a la producción musical. Entre ellos han destacado la colaboración con la empresa japonesa Yamaha en el desarrollo del producto Vocaloid [16], la creación del instrumento digital interactivo Reactable [15] y la creación de la plataforma de referencia en el intercambio libre de sonidos Freesound.org. El grupo quiere contribuir a la mejora de las tecnologías de la información y las comunicaciones relacionadas con el sonido y la música, llevando a cabo una investigación competitiva a nivel internacional y al mismo tiempo transfiriendo sus resultados a la sociedad.

En algunos casos dichos proyectos no solo incluyen la generación de música sino también algún tipo de interacción con la misma. En el ámbito de los videojuegos este tipo de música generativa es muy habitual, como se explica en [24]. En dicho artículo se define como música interactiva a los sonidos que están directamente relacionados con una acción del jugador. En nuestro caso esta acción se puede corresponder con un gesto concreto que hagamos.

Respecto a la parte de la música interactiva propiamente dicha nos encontramos con proyectos como el desarrollado por Mostafa Lofti [28], el cual usa MediaPipe [20] para detectar la posición de los dedos y simular un piano. Otro proyecto clave hecho con MediaPipe es el proyecto AirSynth [33] , el cual nos propone un sonido de sintetizador modificable respecto a la posición de la mano y su apertura. Este proyecto usa SuperCollider [31] para generar la música, siendo esta opción menos práctica que la usada en nuestro proyecto, el cual se ha desarrollado con Pyo [8].

Como hemos visto existen numerosos proyectos que podemos usar como base para el nuestro, añadiendo más funcionalidades y combinándolos para un fin de entretenimiento que da lugar a desarrollar la creatividad a través de la interacción con música en tiempo real en base a una interfaz gestual. En conclusión la unión de música generativa con interacción del usuario usando tecnología actual nos abre la posibilidad de parametrizar de manera natural y sencilla diversos elementos que pueden afectar a la propia composición, mientras que en una pieza convencional lo natural sería afectar únicamente a la interpretación (tempo, dinámicas...) como haría un director de orquesta.

Capítulo 3

Arquitectura hardware del sistema

En este capítulo veremos una descripción en detalle del hardware que se ha elegido para el desarrollo de este proyecto, indicando para qué se usa exactamente así como las características que posee.

Para elegir el hardware en este proyecto partimos de las necesidades que teníamos en mente. Como uno de los objetivos de este proyecto es explorar tecnologías para el reconocimiento de gestos hemos buscado dos opciones para reconocer dichos gestos. La primera de ellas era una placa con acelerómetro y giroscopio capaz de ser entrenada con unos gestos previamente elegidos y que pudiera hacer una inferencia en pocos milisegundos. La misma placa o una adicional debía poder también mandar dichos datos de manera inalámbrica al ordenador que ejecutara el código principal. Este ordenador debe incorporar o tener la posibilidad de añadirle una cámara la cual necesitamos para reconocer la posición de la mano, siendo esta la segunda opción para el reconocimiento de gestos. Por lo tanto otra necesidad era un ordenador lo suficientemente potente para ejecutar el código que reproduce la melodía generativa y que pudiera recibir de manera inalámbrica la información que enviaban las placas descritas anteriormente. En la Figura 3.1 podemos ver un esquema de los dispositivos que acabamos de describir.

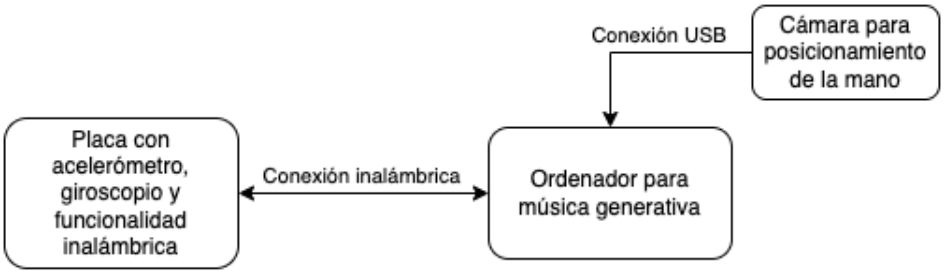

Figura 3.1: Esquema de los dispositivos hardware necesarios para el proyecto.

Tras investigar las posibilidades que teníamos junto a las necesidades descritas anteriormente acordamos usar los dispositivos que se verán a continuación. Por una parte tendremos la placa Nicla Sense ME junto a la ESP32 conectadas por puerto serie UART y montadas sobre un guante. Este conjunto de placas se comunican mediante Bluetooth Low Energy a la Raspberry Pi 4, la cual tiene conectada una cámara USB.

3.1. Nicla Sense ME

La placa de desarrollo elegida para la detección de gestos realizados con la mano en este proyecto es la Nicla Sense ME [23], denominada así ya que es capaz de monitorizar movimiento y entorno. Se ha elegido esta placa ya que la había podido usar durante el desarrollo de mis prácticas curriculares en la empresa Bosch, siendo por lo tanto una placa con la que ya tenía experiencia. De esta placa destaca su pequeño tamaño en unión a 4 sensores principales desarrollados por Bosch Sensortech que veremos más adelante. En la Figura 3.2 podemos ver la placa en detalle.

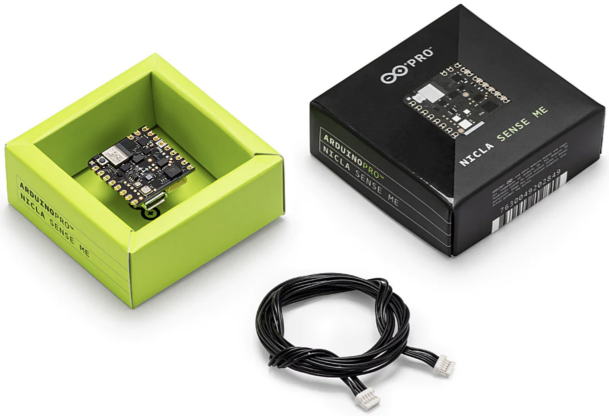

Figura 3.2: Placa Nicla Sense ME usada en el desarrollo de este proyecto.

Esta placa permite a los usuarios desarrollar fácilmente sistemas IoT ya que admite su programación a través del IDE de Arduino y posee varias librerías con ejemplos. Además consume muy poca energía al hacer uso de la tecnología Bluethooth BLE para comunicarse con otros dispositivos. Este ahorro de energía junto con la posibilidad de alimentarse con una batería Li-Po hace que obtengamos una autonomía bastante alta. Gracias al LED que incorpora el usuario puede obtener feedback de la ejecución que se está llevando a cabo en la placa.

Debido a estas características la placa puede colocarse en un guante sobre la mano para reconocer los gestos que se le han entrenado previamente.

En cuanto a las características técnicas de esta placa destacamos el microcontrolador nRF52832 de Nordic dentro del módulo ANNA-B112. Dicho microcontrolador está construido sobre un microcontrolador ARM Cortex-M4 con una unidad de punto flotante que funciona a 64 MHz. Los códigos a ejecutar se almacenan dentro de la FLASH interna de 512 KB del nRF52832 que se comparte con el gestor de arranque. Junto a esto el usuario tiene 64 KB de SRAM a su disposición. Aunque el módulo funciona a 1,8 V, un conversor de nivel puede ajustar el nivel lógico entre 1,8 V y 3,3 V.

Los sensores que posee la placa son los siguientes:

- **BHI260AP:** Hub de sensores inteligentes integrados con Inteligencia Artifial que poseen un IMU de 6 ejes (3 ejes de acelerómetro + 3 ejes de giroscopio) siendo capaz de detectar actividad. Esto se consigue gracias a una CPU Synopsys DesignWare ARC™ EM4™ de 32 bits.

- **BMP390:** Sensor de presión de alto rendimiento capaz de operar entre 300 y 1250 hPa.

- **BMM150:** Magnetómetro con un rango de ±1300 μT en los ejes X e Y, y un rango de ±2500 μT en el eje Z.
- **BME688:** Sensor de medioambiente capaz de medir presión, humedad y temperatura. Además el sensor de gas inteligente que incorpora la placa permite determinar el índice de calidad del aire detectando un amplio rango de gases, incluidos Compuestos Orgánicos Volátiles (COV).

Para este proyecto en concreto usaremos el sensor BHI260AP ya que se trata del sensor de movimiento enfocado a smartwatches, medidores de actividad, smartphones, etc. Se trata de un sensor de muy bajo consumo programable que combina un núcleo Fuser2 y un IMU de 6 ejes (giroscopio y acelerómetro). Se comunica con el core de la Nicla Sense ME vía I2C y SPI. Posee también una memoria flash dedicada de 2MB para almacenar código así como los datos, por ejemplo los datos que usa el algoritmo de calibración de Bosch. Gracias a la CPU que incorpora podremos añadir un modelo de inteligencia artificial que se ejecutará en la propia placa detectando en tiempo real el gesto que se está haciendo permitiendo así la computación en el borde.

Aún con su tamaño muy reducido esta placa de desarrollo nos ofrece una gran cantidad de pines que podemos usar el desarrollo de proyectos IoT. Entre ellos destacamos los pines RX y TX usados en este proyecto para comunicarse por puerto serie y el pin VIN usado para su alimentación. Tanto estos pines como el resto de los disponibles se pueden observar en la Figura 3.3, de los cuales destacamos los pines RX, TX y VIN.

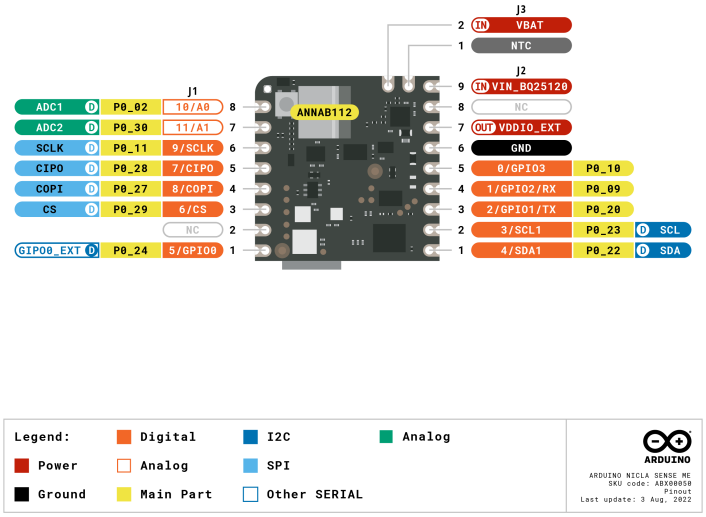

Figura 3.3: Esquema de los pines que posee la placa de desarrollo Nicla Sense ME.

En el IDE de Arduino que usaremos para programar esta placa tenemos disponible la librería Arduino_BHY2 la cual será necesario instalar para que los sketch que desarrollemos funcionen correctamente. Esta librería incluye algunos ejemplos que nos pueden servir como base para desarrollar sobre ellos, ya que incluyen el funcionamiento de los sensores, actualizaciones de firmware, uso del LED incorporado en la placa, entre otras muchas funciones.

Podemos ejecutar un sketch en la Nicla para ver un ejemplo de uso y observar en tiempo real los datos recogidos por los diferentes sensores que posee [7]. Usamos el IDE de Arduino para cargar el sketch en la Nicla y accedemos a la web que se indica en el mismo. En esta web hacemos click en el botón *CONNECT* y seleccionamos la Nicla de entre los dispositivos disponibles. Una vez hecho esto estaremos viendo en tiempo real los datos obtenidos por la placa como podemos ver en la Figura 3.4.

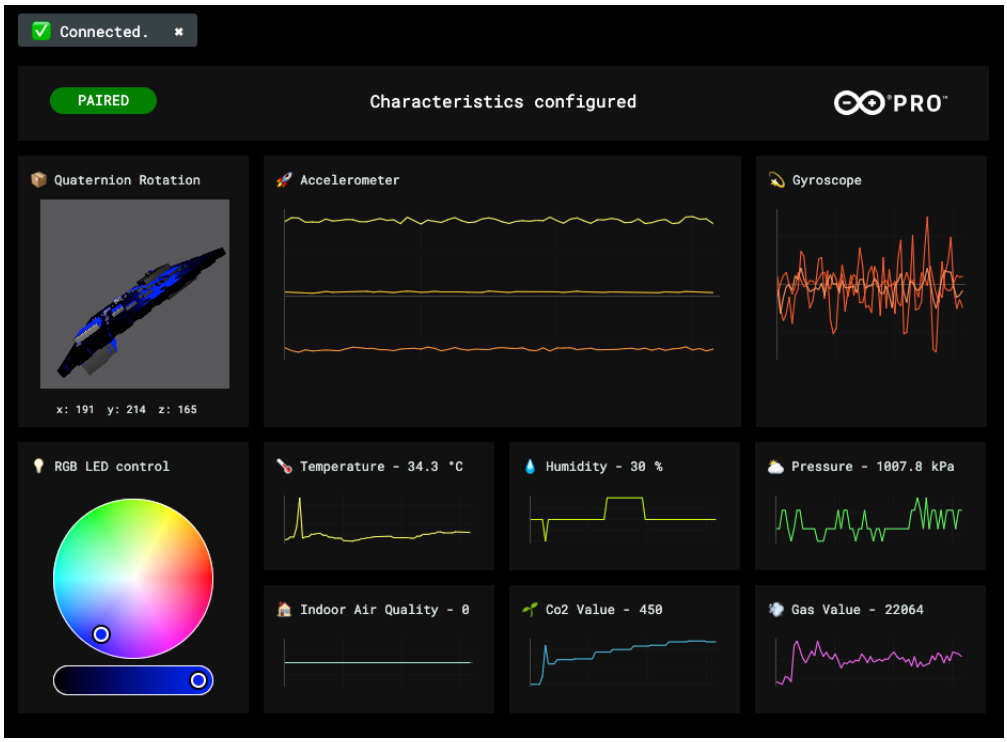

Figura 3.4: Aplicación web que nos permite conectar la Nicla a través de BLE y obtener los datos de los sensores en tiempo real.

3.2. ESP32

En el proyecto también se incluye una placa de desarrollo basada en el SOC ESP32, en concreto el modelo ESP32-DevKitC V4 [26]. Esta placa tiene la función principal de recibir los datos de la Nicla Sense vista en la Sección 3.1 a través del puerto serie y enviarlos a la Raspberry usando Bluetooth. Es necesaria ya que durante el desarrollo del proyecto descubrimos que la placa Nicla Sense ME no tenía la memoria RAM necesaria para ejecutar la inferencia del gesto y la librería Bluetooth a la vez, por lo que se decidió añadir esta placa para ejecutar dicha funcionalidad Bluetooth. Esta placa se puede observar en la Figura 3.5.

Se trata una placa compacta basada en el módulo ESP32-WROOM-32, que tiene un procesador de doble núcleo, WiFi y Bluetooth integrados, y una gran variedad de funciones periféricas. Es comúnmente utilizada para crear prototipos de aplicaciones de IoT, controlar dispositivos, o simplemente experimentar con su potencial.

El módulo ESP32-WROOM-32 tiene un chip ESP32-D0WD con 4 MB de memoria flash y 520 KB de SRAM. Respecto a sus características físicas posee un conector micro USB que sirve para alimentar la placa, programar el módulo, y comunicarse con el ordenador a través de un puente USB-UART.

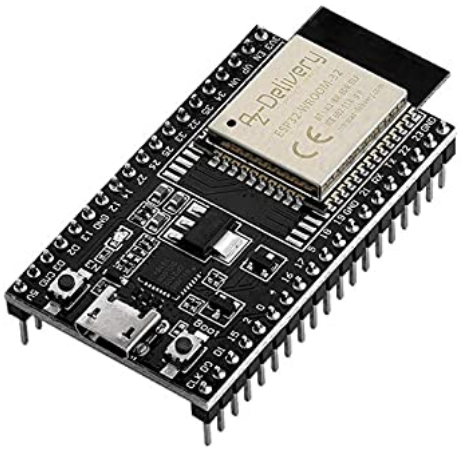

Figura 3.5: Placa de desarrollo ESP32 desarrollada por Espressif.

También tiene dos botones: uno para resetear el módulo (EN) y otro para entrar en el modo de descarga de firmware (BOOT), un LED que se enciende cuando se conecta la alimentación de 5V y expone la mayoría de los pines del módulo a través de dos bloques de cabeceras en ambos lados de la placa. Estos pines pueden usarse para conectar sensores, actuadores, pantallas, o cualquier otro dispositivo compatible con el ESP32. Algunos elementos de su Hardware como el LED, el puerto Micro USB o los botones EN y Boot los podemos ver en la Figura 3.6.

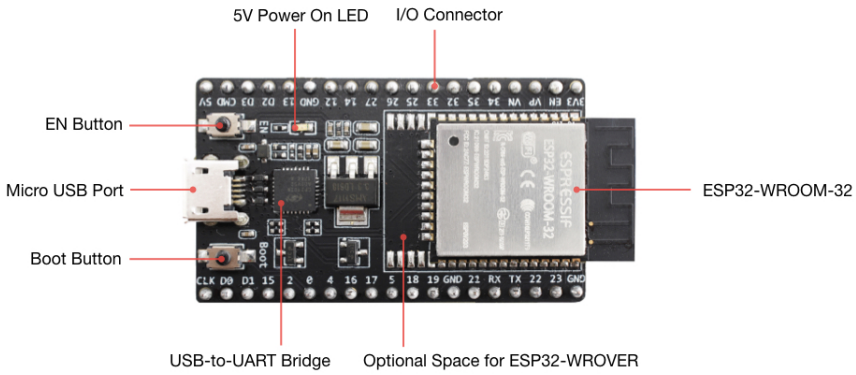

Figura 3.6: Esquema de la placa de desarrollo ESP32.

Para la programación de la placa se puede usar el IDE de Arduino, MicroPython, o el framework oficial de Espressif (ESP-IDF). En nuestro caso usaremos el IDE de Arduino descrito en la Sección 4.2.

3.3. Raspberry Pi 4

El componente hardware principal de este proyecto es una Raspberry Pi 4 [22], la cual es la encargada de generar la música interactiva en base a la posición de la mano y de los gestos. En la Figura 3.7 podemos ver una imagen de la Raspberry Pi 4, que posee una

dimensiones aproximadas de 85mm x 53mm, aunque en nuestro caso se encuentra en una caja haciendo que sea más voluminosa. En ella se ejecutará el modelo de MediaPipe que veremos en la Sección 4.3 además de recibir mediante Bluetooth el gesto detectado por la Nicla Sense ME.

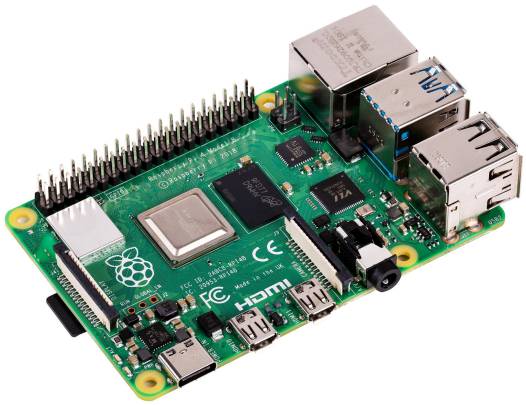

Figura 3.7: Raspberry Pi 4 usada en el proyecto.

Se ha elegido este mini ordenador, ampliamente usado en diversos proyectos como centro multimedia, servidor, estación de trabajo o domótica ya que ofrece una buena potencia en un tamaño muy reducido. Posee un procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72, un SoC Broadcom BCM2711 y 2GB de memoria RAM. Además posee un gran número de conexiones físicas, como 2 puertos micro-HDMI, cuatro puertos USB, dos de ellos USB 3.0, que permiten transferir datos hasta diez veces más rápido. Incluye también una ranura para tarjeta micro SD donde se almacena el sistema operativo y los datos. Respecto a la conectividad destacamos Wi-Fi 802.11ac y Bluetooth 5.0.

En este proyecto la Raspberry se encuentra en una caja a medida con el fin de proteger los componentes, además cabe destacar que se le ha añadido una cámara USB que es la encargada de detectar la posición de la mano haciendo uso de MediaPipe.

3.4. Conectividad

La conexión entre la placa Nicla Sense ME y la ESP32 se realiza mediante la comunicación UART [32]. La conexión UART (universal asynchronous receiver / transmitter, por sus siglas en inglés) es uno de los protocolos de comunicación entre dispositivos más extendidos. Al usar esta comunicación podemos comunicar dispositivos con diferentes tipos de protocolos serie que involucren la transmisión y envío de información.

La comunicación en UART puede ser simple (los datos se envían en una sola dirección), semidúplex(cada extremo se comunica, pero solo uno al mismo tiempo), o dúplex completo(ambos extremos pueden transmitir simultáneamente). En UART, los datos se transmiten en forma de tramas a través de los cables de conexión.

En la Figura 3.8 podemos ver un ejemplo de conexión entre dos dispositivos usando UART. Cada dispositivo posee dos señales, una de transmisión (Tx) y una de recepción (Rx). Cada una de ellas hace la función referida a su nombre. La interfaz UART no usa una señal de reloj para sincronizar estas conexiones, como podemos encontrar en otras formas de comunicación. En su caso utiliza una transmisión asíncrona basada en la frecuencia de ambos dispositivos, por lo que si no coinciden nos podemos encontrar con discrepancias en el envío de la información.

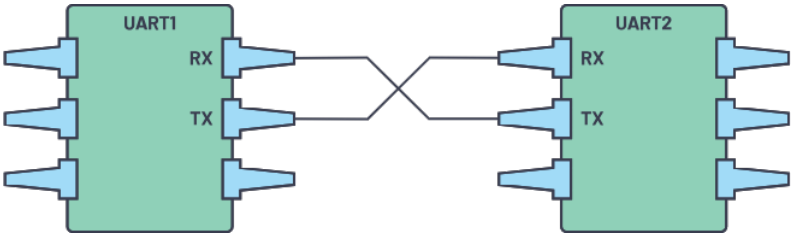

Figura 3.8: Esquema de la conexión UART entre dos dispositivos. Obtenida de https:// www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

Como ya sabemos uno de los motivos principales para usar la placa ESP32 es enviar información a través del protocolo Bluetooth BLE (Bluetooth Low Energy) [27], el cual es un tipo de Bluetooth que consume mucha menos energía que el Bluetooth clásico. Esto se debe a que está diseñado para facilitar las aplicaciones de IoT que requieren conexiones inalámbricas de corto y mediano alcance, con un bajo consumo de batería y un bajo coste de implementación. Este protocolo se usa para comunicar la placa ESP32 con la Raspberry Pi 4, como veremos en detalle en las Secciones 5.2 y 5.4.

El protocolo Bluetooth BLE opera en la banda ISM de 2.4 GHz, pero a diferencia del Bluetooth clásico, permanece en modo de suspensión constantemente, excepto cuando se inicia una conexión. Los tiempos de conexión reales son solo de unos pocos milisegundos. Esto hace que los dispositivos BLE puedan funcionar durante largos períodos sin agotar la batería, transmitiendo datos periódicamente.

En este protocolo tenemos dos tipos de dispositivos, los clientes y los servidores. Un cliente es el dispositivo que inicia la conexión y solicita datos al servidor. Un servidor es el dispositivo que espera la conexión y proporciona datos al cliente. En nuestro caso la Raspberry Pi actuará como cliente mientras que la ESP32 actúa como servidor.

Para definir como se estructuran y se acceden a los datos en un dispositivo BLE se usa el protocolo GATT (Generic Attribute Profile) [12]. Este protocolo utiliza una tabla de atributos que contiene los datos organizados en servicios y características. Un servicio es una colección de datos relacionados que proporciona una funcionalidad específica a un dispositivo BLE. Por ejemplo, un servicio puede ser el de frecuencia cardíaca, el de batería o el de localización. Cada servicio tiene un identificador único llamado UUID (Universally Unique Identifier), que lo distingue de otros servicios. Estos identificadores son un número de 128 bits (16 bytes) que en la mayoría de los casos garantiza ser único globalmente. Los UUID se usan en diversos protocolos y aplicaciones. Por otro lado una característica es un dato individual que pertenece a un servicio. Por ejemplo, una característica puede ser el valor de la frecuencia cardíaca, el nivel de la batería o la coordenada geográfica. Cada característica también tiene un UUID único y unas propiedades que definen cómo se puede leer, escribir o notificar su valor.

En la Figura 3.9 podemos ver un ejemplo de tabla GATT relacionada con un servicio de ritmo cardíaco. Este servicio posee varias características las cuales poseen cada una un ID y un valor así como ciertos permisos de escritura o lectura.

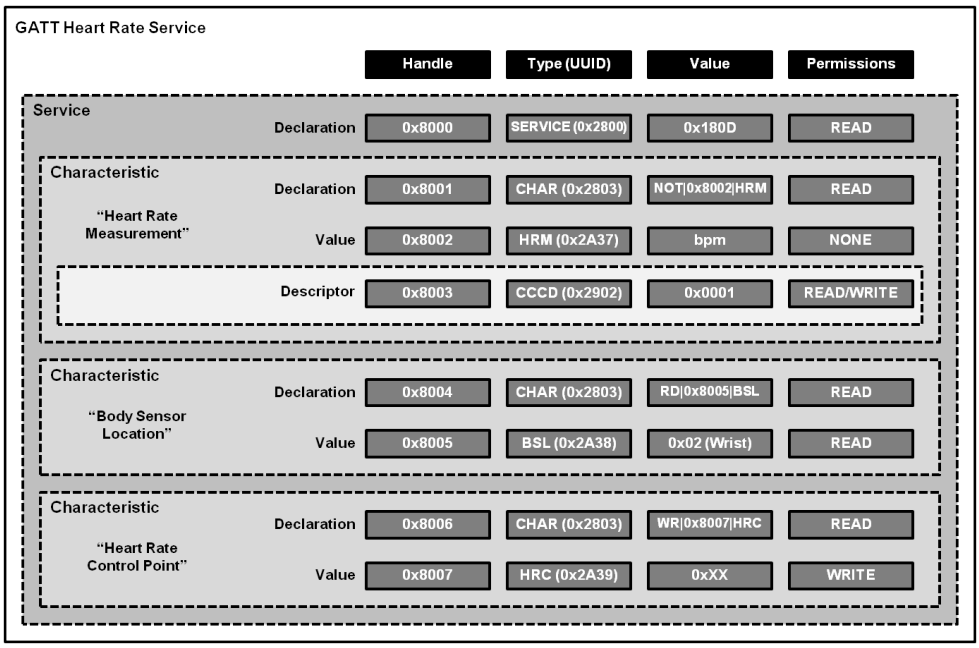

Figura 3.9: Esquema de una tabla GATT. Obtenida de https://microchipdeveloper.com/wireless:ble-gatt-data-organization

Capítulo 4

Arquitectura software del sistema

En este capítulo veremos el software elegido para el desarrollo del proyecto, desde las herramientas de Machine Learning hasta el entorno de desarrollo utilizado, pasando por los lenguajes de programación usados y las librerías elegidas.

El software se ha elegido de acuerdo a las necesidades del proyecto, las cuales requieren un lenguaje de programación sencillo de usar que se pueda ejecutar en la Raspberry Pi 4 y que tenga soporte para las librerías que necesitamos, una herramienta que obtenga los valores de acelerómetro y giroscopio de la placa Nicla Sense ME y nos permita hacer una inferencia en base a dichos datos, otra herramienta que nos facilite la detección de la posición de la mano respecto a la cámara y por último una herramienta que nos permita programar y reproducir una melodía generativa, la cual es una parte clave de este proyecto. Junto a esto necesitamos también un entorno de desarrollo que nos permita programar las placas de desarrollo elegidas.

El lenguaje de programación elegido para el desarrollo necesario en la Raspberry Pi 4 ha sido Python (versión 3.9) [21] . Hemos elegido este lenguaje ya que es ampliamente usado y fácil de aprender, posee una sintaxis simple y similar al idioma inglés, permite escribir un código más eficiente que otros lenguajes así como ejecutarlo tan pronto como se escribe y posee una gran cantidad de bibliotecas y soporte de la comunidad.

Algunas de las desventajas que podemos encontrar sobre este lenguaje es que puede ser más lento que otros lenguajes compilados, no es el más adecuado para desarrollo móvil y puede presentar incompatibilidades entre diferentes versiones. Sin embargo ninguna de estas desventajas es crucial para el proyecto desarrollado.

Sabiendo las necesidades que tenemos y habiendo investigado las opciones disponibles se ha hecho la elección del software que se describe en esta sección, relacionándolo con los elementos hardware que hemos descrito anteriormente. En la placa Nicla Sense ME se ha usado el IDE de Arduino para ejecutar el modelo de inteligencia artifical creado en Edge Impulse. En la placa ESP32 se usa de nuevo el IDE de Arduino para programar un sketch que usa Bluetooth BLE para para mandar la información necesaria a la Raspberry Pi. Por último en la Raspberry Pi 4 se hace uso de Python para ejecutar varios scripts que incorporan las librerías Pyo, MediaPipe y el protocolo OSC.

En la Figura 4.1 podemos ver un esquema de las herramientas que acabamos de nombrar así como de dónde se ejecutarán.

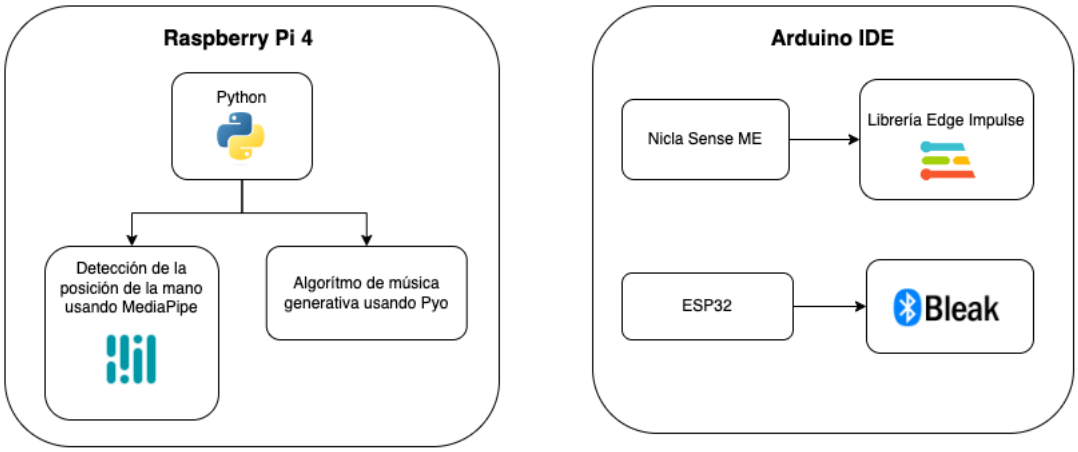

Figura 4.1: Esquema de las necesidades software para el proyecto.

4.1. Edge Impulse

La plataforma elegida para crear el modelo de inteligencia artificial que incorporará la Nicla Sense es Edge Impulse [18]. Es una plataforma de aprendizaje automático que se centra en el desarrollo y la implementación de modelos de inteligencia artificial en dispositivos como microcontroladores, sistemas embebidos y otros dispositivos de baja potencia. Se ha elegido esta plataforma ya que como hemos visto anteriormente hay un gran número de proyectos que usan Edge Impulse junto a la Nicla Sense ME, teniendo así bastante documentación sobre el uso de la misma.

El uso de Edge Impulse se basa en cuatro elementos principales: captura de datos, entrenamiento de modelos, pruebas y despliegue del modelo en los dispositivos. En la figura 4.2 podemos ver un esquema de estos elementos. La plataforma permite a los desarrolladores capturar datos en tiempo real de los sensores de los dispositivos, creando así un dataset. A continuación se utiliza dicho dataset para entrenar modelos de aprendizaje automático. Finalmente, se pueden implementar estos modelos en los dispositivos para realizar inferencias en tiempo real.

La plataforma Edge Impulse sirve como un puente entre la investigación de aprendizaje automático y la implementación en dispositivos, permitiendo a los desarrolladores de todo el mundo crear modelos de aprendizaje automático que funcionen en dispositivos de baja potencia. Algunos de los casos de uso desarrollados con Edge Impulse incluyen compañías internacionales de gran reputación como Sony, NASA o HP.

Edge Impulse incluye una amplia variedad de herramientas de captura de datos, así como bibliotecas para sensores, bibliotecas de adquisición de datos y herramientas de preprocesamiento de datos. Ofrece soporte oficial para una gran cantidad de dispositivos, entre los que destacan dispositivos de Arduino, los Nordic Semi o los de Silicon Labs. También cuenta con herramientas de entrenamiento de modelos, junto con una amplia variedad de modelos preentrenados y personalizables, así como herramientas para el ajuste de diferentes parámetros. El uso de Edge Impulse abarca diversos tipos de datos, desde análisis de datos de acelerómetros, temperatura, imágenes o incluso audio. En la Figura 4.3 podemos ver algunos de los tipos de bloques de procesamiento que incluye la plataforma, en nuestro caso usaremos el análisis espectral ya que como indica es el adecuado para datos de acelerómetro.

Una vez hemos entrenado el modelo a nuestro gusto tenemos a nuestra disposición una

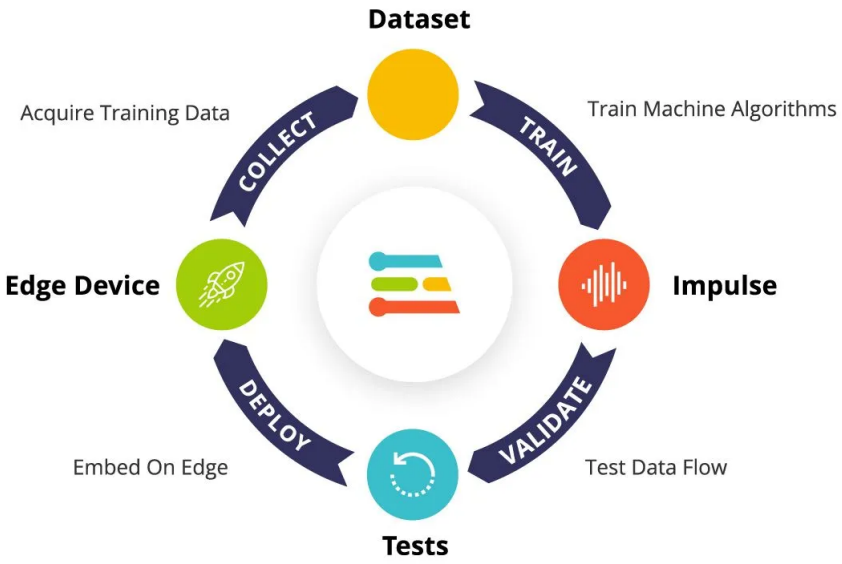

Figura 4.2: Flujo de trabajo de la plataforma Edge Impulse. Obtenido de https://edgeimpulse.com/blog/getting-started-with-edge-impulse

DESCRIPTION	AUTHOR	RECOMMENDED	
Spectral Analysis Great for analyzing repetitive motion, such as data from accelerometers. Extracts the frequency and power characteristics of a signal over time.	Edge Impulse	★	Add
IMU (Syntiant) Syntiant only. Great for analyzing repetitive motion, such as data from accelerometers. Extracts the frequency and power characteristics of a signal over time.	Syntiant	★	Add
Flatten Flatten an axis into a single value, useful for slow-moving averages like temperature data, in combination with other blocks.	Edge Impulse		Add
Image Preprocess and normalize image data, and optionally reduce the color depth.	Edge Impulse		Add
Audio (MFCC) Extracts features from audio signals using Mel Frequency Cepstral Coefficients, great for human voice.	Edge Impulse		Add

Figura 4.3: Bloques de procesamiento disponibles en Edge Impulse.

amplia variedad de formas de despliegue. En nuestro caso nos interesa exportar el modelo como librería de Arduino, aunque también tenemos la posibilidad de exportarlo como una librería de C++, un paquete WebAssembly o un binario para Linux entre otros.

El uso de esta herramienta es gratuito, aunque incluye ciertas limitaciones como 20 minutos de datos entrenados, 4GB de datos, un solo administrador y limitaciones en el uso de compiladores. Existe una versión de pago denominada Enterprise teams que elimina

estas limitaciones además de añadir otras ventajas como colaboración entre proyectos de la organización, acceso ilimitado a la API o optimización del compilador. En el desarrollo de nuestro proyecto hemos usado la versión gratuita siendo esta más que suficiente.

En la Sección 5.1 se desarrollará en detalle el uso de esta plataforma y se explicará como se puede usar para entrenar los gestos personalizados.

4.2. Arduino IDE

Para la programación tanto de la placa Nicla Sense como de la placa ESP32 se ha elegido el entorno de desarrollo de Arduino, denominado Arduino IDE 2 [17], principalmente por su facilidad de uso y su compatibilidad con la mayoría de placas de desarrollo, incluyendo las librerías necesarias para el desarrollo del proyecto.

El entorno de desarrollo Arduino IDE es una aplicación multiplataforma de código abierto que permite a los desarrolladores escribir y cargar programas en placas compatibles con Arduino, usando los lenguajes C y C++. Incluye un editor de código, un compilador y un cargador de programas.

A continuación veremos algunas de las características más importantes de este entorno de desarrollo:

- **Boards Manager:** Herramienta dedicada gestionar las placas compatibles con el IDE, permitiendo buscar las diferentes placas y añadiéndolas al sistema.
- **Library Manager:** Herramienta para la gestión de las librerías necesarias para el proyecto a desarrollar, permitiendo buscar e instalar las librerías necesarias desde el propio IDE.
- **Debug en tiempo real:** Te permite depurar el código paso a paso, viendo el estado de las variables y los registros, y estableciendo puntos de interrupción.
- **Serial Monitor:** Te permite la comunicación con la placa de desarrollo mediante el puerto serie, enviando y recibiendo datos en tiempo real.

Posee una interfaz simple e intuitiva para el usuario, haciendo así que el aprendizaje de este entorno de desarrollo sea sencillo. Nos ayuda a distinguir las diferentes partes del código con el esquema de colores que utiliza y nos pone a nuestra disposición acceso rápido a algunas funciones como la depuración del código o la compilación y carga en el dispositivo. En la Figura 4.4 podemos ver la interfaz en detalle, destacando cómo se muestra el código así como información adicional sobre los dispositivos que tenemos conectados. En esta caso el código que se muestra es un ejemplo de la librería de la placa Nicla Sense ME.

4.3. MediaPipe

MediaPipe es un framework de código abierto desarrollado por Google [20] usado para construir pipelines de Machine Learning multimodales (videos, audio, datos de series temporales) y multiplataforma (iOS, Android, web, dispositivos en el borde). Está optimizado con el rendimiento y con la inferencia en el dispositivo en mente.

MediaPipe permite personalizar y desplegar soluciones con Machine Learning en el dispositivo usando APIs ligeras. También se puede hacer uso de modelos pre-entrenados o entrenar modelos propios con la herramienta MediaPipe Model Maker. Además ofrece un visualizador para ayudarte a diseñar y depurar tus pipelines.

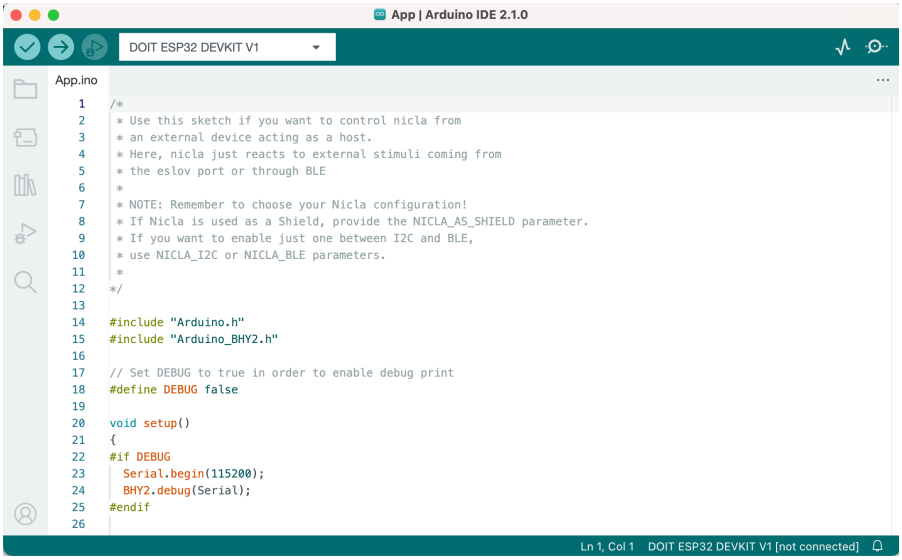

Figura 4.4: Captura de la interfaz del entorno de desarrollo Arduino IDE 2.

Esta tecnología es usada por Google en muchos de sus productos como Google Lens, Google Meet, Youtube y Google Photos. Al ser de código abierto puede ser usada para que el resto de personas puedan ofrecer soluciones innovadoras, eficientes y escalables.

Algunos ejemplos de soluciones que ofrece MediaPipe son:

- Detección y seguimiento de cara
- Estimación de la pose de la mano y reconocimiento de gestos
- Detección y segmentación de objetos
- Síntesis y reconocimiento del habla
- Realidad virtual y realidad aumentada

En concreto para este proyecto vamos usar la detección de puntos de referencia (landmarks) en la mano [19]. Con esto sabremos la posición de varios puntos clave de la mano, podremos detectar la posición de los dedos y la palma respecto a la pantalla y entre ellos. En la Figura 4.5 podemos ver un esquema de la puntos que se detectan en la mano, siendo un total de 21 repartidos de manera homogénea entre los dedos y la palma de la mano. Esto nos proporciona una gran libertad a la hora de ejecutar la detección.

Este modelo se ha entrenado con aproximadamente treinta mil imágenes del mundo real así como una gran cantidad de modelos de manos sintéticas impuestas en diversos fondos.

Para su correcto funcionamiento se usan dos modelos. El primero es el de detección de la palma el cual localiza la mano en la imagen de entrada, y el modelo de puntos de referencia que identifica los 21 puntos que hemos visto anteriormente sobre la imagen de la mano recortada definida por el modelo anterior.

Como el modelo de detección de la palma puede tomar más tiempo al ejecutarlo en vídeo o en tiempo real el modelo de detección de puntos usa el área delimitada en un principio para localizar la región en los siguientes fotogramas. El modelo de detección de palma solo se vuelve a ejecutar en caso de que el modelo de detección de puntos no sea capaz de identificar una mano. Esto hace que el modelo general sea más eficiente.

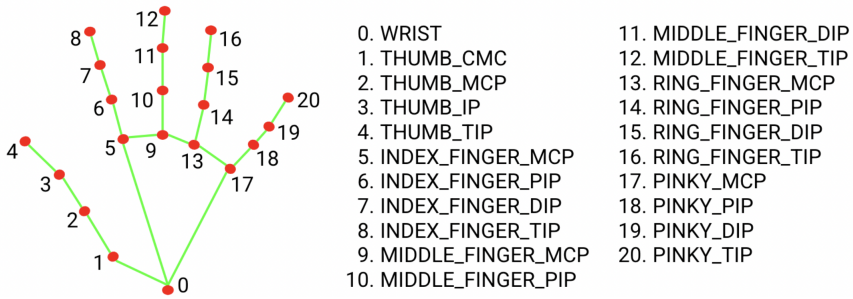

Figura 4.5: Esquema de los puntos de referencia que proporciona el modelo de MediaPipe. Obtenido de https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

4.4. Pyo

Pyo es un módulo de Python dedicado a la generación y procesamiento de señales de audio, convirtiéndolo en una herramienta similar a Supercollider pero desarrollado como módulo de Python. Usando este módulo los usuarios son capaces de introducir directamente cadenas de procesamiento de señales de audio en sus scripts de Python o en sus proyectos, además de poder manipularlas en tiempo real a través del interpreter.

Las herramientas que ofrece Pyo van desde las más básicas como operaciones matemáticas sobre las señales de audio o procesamiento básico (filtros, retrasos, etc.) hasta algoritmos complejos para la creación de granulación del sonido y otras manipulaciones creativas del audio. Pyo introduce soporte para el protocolo OSC (Open Sound Control) el cual ha sido clave en este trabajo, para facilitar las comunicaciones entre software, el protocolo MIDI, generación de eventos de sonido y controlar parámetros.

Con esto destacamos que Pyo permite la creación de cadenas de procesamiento de señales muy sofisticadas con todo los beneficios de un lenguaje de programación maduro y ampliamente usado como es Python.

En el Github oficial del proyecto [8] podemos encontrar documentación ampliada así como algunos ejemplos de código para ver cómo funciona. En la Figura 4.6 podemos ver uno de estos ejemplos usando Pyo, en este caso se crea una onda sinusoidal, lo que hará que escuchemos un tono continuamente. Para ello se debe crear y ejecutar el servidor, crear la onda y por último abrir la interfaz gráfica que nos dará los controles para reproducirla.

4.4.1. Protocolo OSC

El protocolo OSC [34] que hemos nombrado anteriormente es un protocolo de comunicación diseñado para la transmisión de datos de control de música y sonido en tiempo real. Es flexible y extensible ya que permite a los usuarios enviar y recibir datos de control de parámetros de sonido y música entre diferentes dispositivos y aplicaciones. Fue desarrollado en el CNMAT por Adrian Freed y Matt Wright en 2002.

El protocolo OSC consiste en una nomenclatura simbólica al estilo URI, que permite identificar los destinatarios y los parámetros de los mensajes. Los mensajes pueden contener datos simbólicos y numéricos de alta resolución, así como etiquetas de tiempo para sincronizar los efectos. Los mensajes se pueden agrupar en paquetes para enviarlos simultáneamente.

Las ventajas de OSC son la interoperabilidad, la precisión, la flexibilidad y una mejor organización y documentación. OSC permite conectar distintas aplicaciones y dispositivos

```
"""
02-sine-tone.py - The "hello world" of audio programming!

This script simply plays a 1000 Hz sine tone.

"""
from pyo import *

# Creates and boots the server.
# The user should send the "start" command from the GUI.
s = Server().boot()
# Drops the gain by 20 dB.
s.amp = 0.1

# Creates a sine wave player.
# The out() method starts the processing
# and sends the signal to the output.
a = Sine().out()

# Opens the server graphical interface.
s.gui(locals())
```

Figura 4.6: Código de un ejemplo sencillo del uso de Pyo.

con fines como la interpretación musical o el control de espectáculos, sin depender del medio de transmisión ni del tipo de datos. OSC es una alternativa al estándar MIDI, que tiene limitaciones en la resolución y el espacio de parámetros.

Capítulo 5

Desarrollo del proyecto

En este capítulo se describirá en detalle el desarrollo que se ha seguido en este proyecto, indicando cómo funcionan las conexiones entre los dispositivos y los protocolos que se usan en ellas. Primero tenemos la explicación relativa a la parte del desarrollo de la Nicla Sense ME y la ESP32 encargadas de detectar el gesto dibujado en el aire y por otra parte tenemos el desarrollo realizado en la Raspberry Pi 4 encargada de reproducir la melodía generativa y aplicar la interacción por parte del usuario junto a la detección de la posición de la mano.

5.1. Detección de gestos usando Nicla Sense ME

En esta primera sección vamos a ver paso por paso cómo se ha desarrollado la detección de gestos en la plataforma Edge Impulse, siguiendo los pasos de la propia documentación [3]. Antes de comenzar este proceso hay que crear una cuenta en la plataforma y crear un nuevo proyecto.

El primer paso es cargar la Nicla Sense ME con el sketch para la ingesta de datos, que nos permitirá conectar la placa con la herramienta de consola de comandos de Edge Impulse y que mandará los datos a la propia plataforma al obtenerlos desde la Nicla Sense ME. Este sketch se ha modificado para que solamente se lean los datos del acelerómetro y giroscopio, descomentando dichas líneas de código y comentando los demás sensores como se puede ver en la Figura 5.1.

```
#define SAMPLE_ACCELEROMETER
#define SAMPLE_GYROSCOPE
// #define SAMPLE_ORIENTATION
// #define SAMPLE_ENVIRONMENTAL
// #define SAMPLE_ROTATION_VECTOR
```

Figura 5.1: Código necesario para recibir solo los datos de los sensores de acelerómetro y giroscopio.

Una vez cargado el sketch en la Nicla instalamos la herramienta CLI de Edge Impulse [4]. Tras esto ejecutamos *edge-impulse-data-forwarder* en el terminal e introducimos los datos de nuestra cuenta de Edge Impulse, además seleccionamos el proyecto al que queremos conectar la Nicla. Esta herramienta se encarga de recoger los datos de los sensores de

la placa que tenemos conectada por USB y los envía al proyecto correspondiente en Edge Impulse.

En la aplicación web de Edge Impulse disponemos de varios apartados a través de los cuales configuraremos y crearemos el modelo de inferencia que necesitamos. Vamos a ver un resumen de los apartados que usaremos durante el desarrollo:

- **Devices:** En este apartado podemos ver los dispositivos que tenemos vinculados al proyecto así como los sensores que poseen cada uno y el estado actual del dispositivo. También nos permite añadir un nuevo dispositivo externo o usar los sensores de nuestro ordenador como dispositivo.

- **Data acquisition:** Desde aquí podemos recoger muestras de datos de los dispositivos configurados anteriormente, seleccionando el tiempo total de cada muestra. Además podemos ver todas las muestras que hemos tomado, eliminarlas y elegir el porcentaje de muestras de entrenamiento y de test.

- **Impulse design:** Este es el apartado más complejo de la plataforma, en él podemos configurar al máximo el modelo que vamos a crear pasando por tres fases. En la primera denominada *Create Impulse* podemos configurar parámetros relacionados con los sensores y con los datos de salida. En la segunda denominada *Spectral features* podemos configurar parámetros a aplicar sobre los datos en crudo y generaremos datos sobre la relevancia de cada sensor a la hora de hacer la inferencia. Por último en *Classifier* configuraremos parámetros relativos con el entrenamiento, como el número de ciclos de entrenamiento y las capas de neuronas. Una vez entrenado podemos ver el rendimiento del modelo y la matriz de confusión.

- **Retrain model:** Si añadimos datos nuevos y queremos volver a entrenar el modelo aquí encontramos una manera rápida de entrenarlo sin tener que pasar por el proceso completo.

- **Live classification:** En este apartado podemos tomar una muestra al momento y Edge Impulse ejecutará la inferencia, indicándonos la probabilidad que tiene dicho gesto de ser cada uno de los entrenados.

- **Deployment:** Por último tenemos el apartado relacionado con el despliegue, en el cual podemos exportar el modelo que acabamos de crear como librería para diferentes lenguajes.

Comenzaremos por la pestaña *Devices* y ahí nos debe aparecer la Nicla lista para usar. Cabe destacar que en el proceso de adquisición de datos la Nicla debe de estar constantemente conectada al ordenador a través del cable.

Con la Nicla conectada a la aplicación web entramos a la pestaña *Data acquisition*, en la cual podemos ver un resumen de todo el dataset que contiene los gestos entrenados además de poder guardar un nuevo registro indicando su etiqueta. De esta manera iremos repitiendo los gestos que deseemos entrenar posteriormente hasta tener una buena cantidad. Tras haber ido probando hemos determinado que con alrededor de 30 entradas para un gesto empezamos a obtener buenos resultados, añadiendo también unas 10 entradas adicionales para el conjunto de test. Hemos entrenado dos gestos para la interacción de la música en este proyecto, siendo el primero de ellos el gesto denominado *Corte*, el cual podemos ver en la Figura 5.2 y que se usará para pausar la reproducción de la música. Se trata de un gesto inspirado en el que ejecuta un director de orquesta para finalizar una pieza.

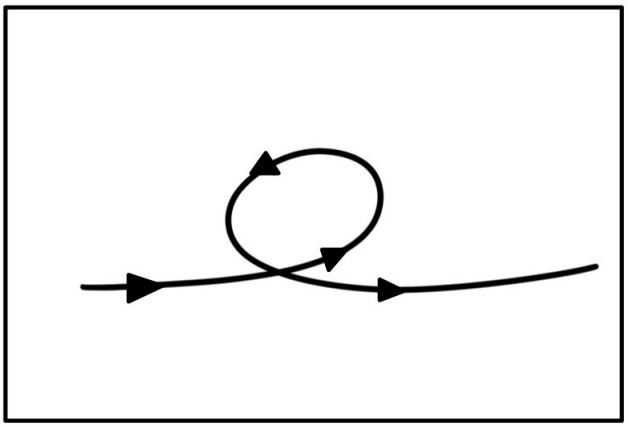

Figura 5.2: Gesto para pausar la reproducción de la música.

El otro gesto que hemos entrenado es el gesto *Inicio*, el cual se puede ver en la Figura 5.3 y se usa para reanudar la reproducción de la música. Este gesto está inspirado en el que realiza el director de orquesta para indicar un compás de dos tiempos.

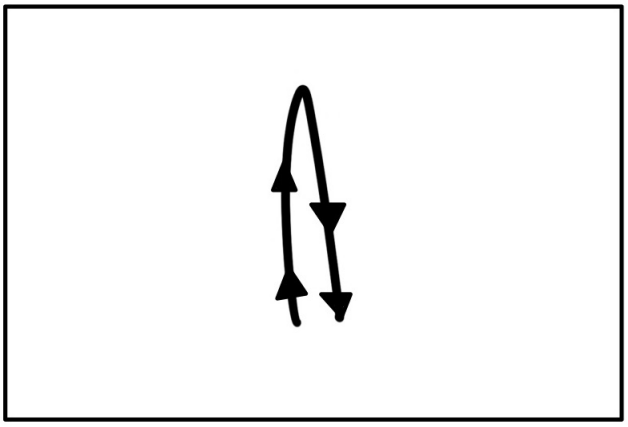

Figura 5.3: Gesto para reanudar la reproducción de música.

Con todos los datos recopilados para entrenar el modelo de Inteligencia Artifical es hora de crear dicho modelo y configurar los parámetros que usará así como la clasificación que se realizará. Para ello entramos en la pestaña *Impulse Design*, donde deberemos ir pasando por cada una de las tres fases (Create impulse, Spectral features y Classifier).

Empezamos por *Create Impulse*, en la cual debemos elegir diversos parámetros. El tamaño de ventana indica en milisegundos la cantidad de datos que se procesarán en cada clasificación, en nuestro caso configuramos 3000 ya que habíamos tomado muestras de 3 segundos. El incremento de ventana nos indica cuánto se puede alargar la ventana en caso de que la muestra sea más grande, lo dejamos por defecto a 200 ms ya que como hemos dicho todas nuestras muestras son de 3 segundos. Como bloque de procesamiento hemos elegido un análisis espectral, indicado para datos de acelerómetro y giroscopio, en este apartado es donde podemos elegir si queremos descartar alguno de estos datos aunque tras probar varias opciones hemos concluido que lo óptimo es usar tanto acelerómetro como giroscopio. Los dos últimos apartados (clasificación y salidas) los dejamos por defecto ya que no tenemos nada que configurar, solo nos informan de las salidas posibles. En la Figura

5.4 podemos ver los parámetros elegidos que acabamos de describir.

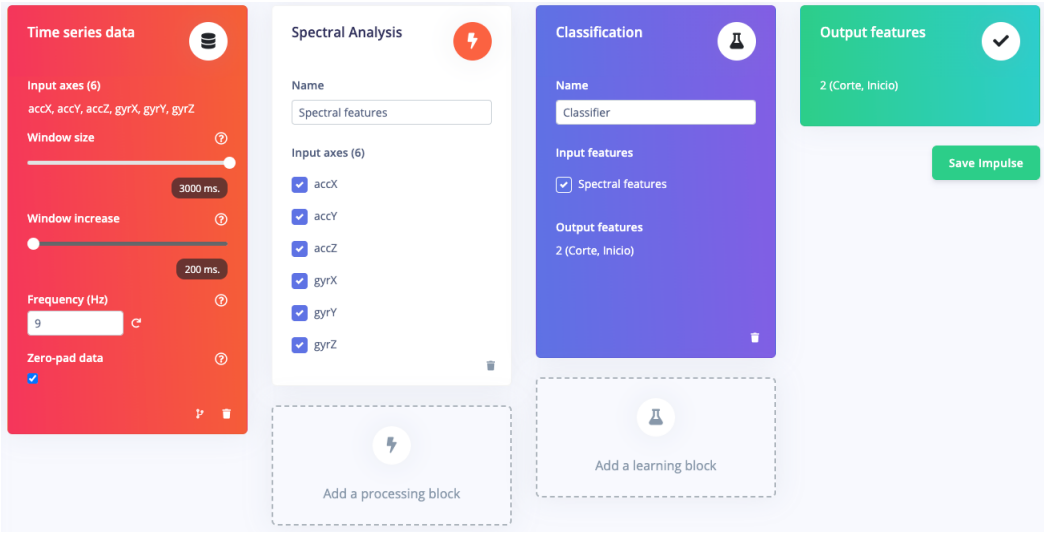

Figura 5.4: Captura de pantalla de la interfaz del apartado *Create Impulse*.

En *Spectral features* podemos ver un resumen de los parámetros y dejaremos la configuración por defecto ya no hemos visto que influyan de manera muy relevante sobre los resultados obtenidos en este proyecto. Guardamos dichos parámetros y a continuación podemos ver un resumen del tiempo total de datos que poseemos así como de los resultados posibles, aquí solo debemos hacer click en el botón de *Generate features* para finalmente generarlas.

El último paso es el *Classifier*, en el cual podemos modificar parámetros como el número de ciclos de entrenamiento y cambiar la arquitectura de la red neuronal, añadiendo capas con las neuronas que deseemos. Estos parámetros se pueden ver en la Figura 5.5, en nuestro caso hemos configurado 300 ciclos de entrenamiento con una arquitectura de 3 capas de neuronas, la primera capa 20 neuronas, la segunda de 10 y una última de 5. Estos parámetros se han definido tras hacer pruebas con diferentes valores en las que hemos determinado que estos valores nos arrojaban buenos resultados. Una vez obtenidos los parámetros que queremos hacemos click en el botón *Start training* y esperamos a que se complete el entrenamiento. Una vez terminado podemos ver a la derecha los resultados obtenidos. En la Figura 5.5 observamos dichos resultados con los cuales hemos conseguido una clasificación excelente ya que en todos los casos ha clasificado el gesto correctamente, visible en la matriz de confusión que genera.

Para probar el modelo que acabamos de crear podemos ir a la pestaña *Live classification* y con la Nicla Sense ME conectada hacemos click en *Start Sampling*, lo que recogerá en ese mismo instante los valores de los sensores durante el tiempo indicado, los pasará por el modelo que hemos entrenado y nos devolverá el resultado. En la Figura 5.6 vemos en la parte de abajo como resultado que hemos realizado un gesto de corte con una probabilidad de 0.99 sobre 1, siendo efectivamente este el gesto que se realizó.

Una vez esté creado el modelo final y estemos conformes con el resultado vamos a la pestaña *Deployment* y seleccionamos la librería de Arduino. Hacemos click en *Build* y se nos descargará la librería para poder implementarla en la Nicla Sense ME y que podamos hacer la inferencia sin necesidad de Edge Impulse ni de las herramientas CLI, siendo la Nicla Sense ME completamente autónoma.

Sin embargo, el código que nos devuelve Edge Impulse no está listo para usar directa-

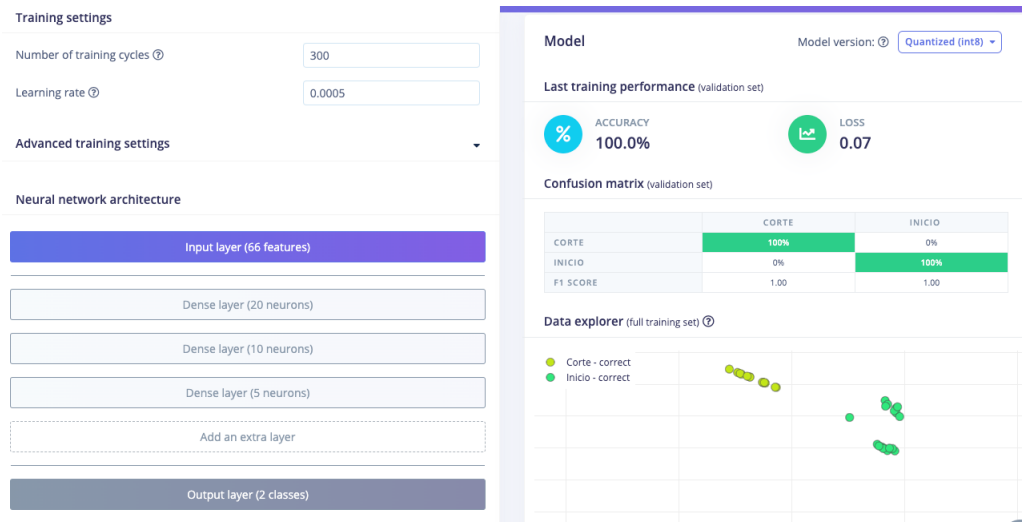

Figura 5.5: Captura de pantalla de la interfaz del apartado *Classifier*.

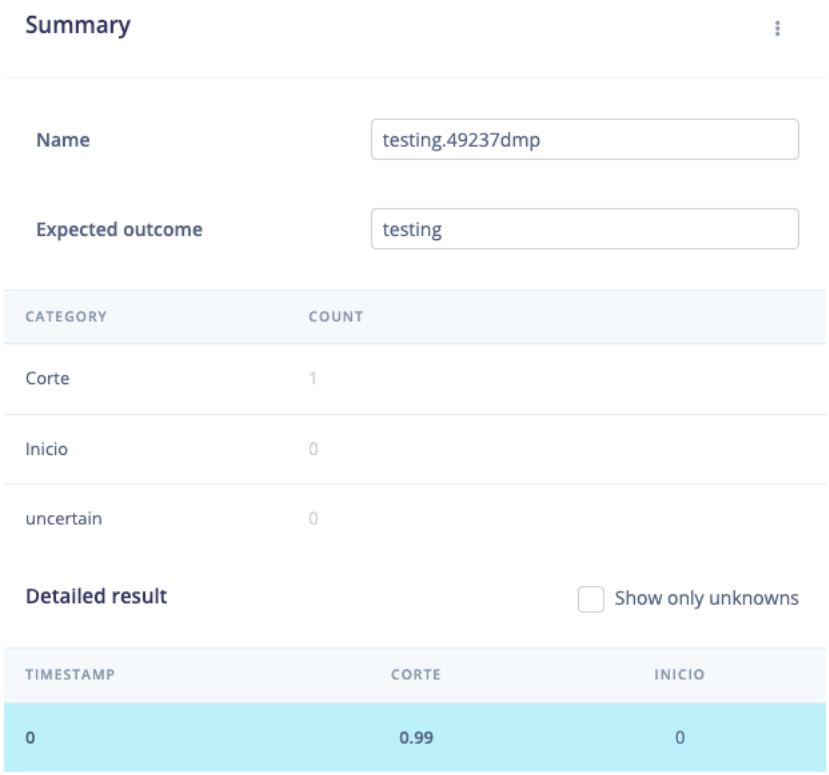

Figura 5.6: Resultado de una clasificación en vivo en la cual detecta correctamente el gesto ejecutado.

mente en el proyecto ya que envía los resultados por el puerto serie sin tener el formato adecuado para que la ESP32 los envíe por Bluetooth. Es por esto que se ha tenido que modificar el sketch añadiendo la funcionalidad necesaria. En la Figura 5.7 podemos ver una parte clave de este código. Si se detecta un gesto con más de un 90 % de probabilidad se comprobará cual de los dos gestos posibles se ha detectado y se envía el número correspondiente a dicho gesto. Si el gesto detectado es inconcluso se envía también el número

correspondiente.

```
gesture = result.classification[maxLabel].label;
if (maxValue > 0.9) {
  if(gesture == "Corte"){
    Serial.write('1');
  }else if (gesture == "Inicio"){
    Serial.write('2');
  }
}else{
  Serial.write('3');
}
```

Figura 5.7: Código desarrollado para que se envía por serial el número correspondiente al gesto detectado.

El código completo que ejecuta la Nicla Sense ME se puede encontrar en el Github del proyecto [9] .

5.2. Envío del gesto detectado a través de Bluetooth

Siguiendo el desarrollo explicado en la sección anterior el siguiente paso es que el resultado del gesto detectado en la placa Nicla llegue por Bluetooth hasta la Raspberry, donde se procesará y se ejecutará la acción correspondiente. La idea original era que la Nicla realizara la inferencia y el envío, sin embargo nos hemos topado con numerosos problemas para conseguir esto dado que la memoria RAM que posee esta placa no es suficiente para realizar ambas tareas. Por ello nos vamos a ayudar de una placa adicional, en este caso una ESP32, la cual recibirá a través del puerto serie el gesto detectado y será dsta la encargada de enviarlo vía Bluetooth.

Para el envío del gesto detectado a través del puerto debemos conectar el pin RX de la Nicla al pin 17 de la ESP32, que hará las veces de pin TX. Por otro lado conectaremos el pin TX de la Nicla al pin 16 de la ESP32, que hará de pin RX. En la Figura 5.8 se puede ver un esquema de esta conexión, también se incluye la conexión del pin Vin de la Nicla al pin de 3.3V de la ESP32, necesario para alimentar la Nicla. De esta forma para alimentar ambas placas sólo debemos conectar la ESP32 por USB y esta se encargará de alimentar la Nicla Sense ME.

Una vez vista la programación de la placa Nicla vamos a ver en detalle la programación de la placa ESP32 para que mande el dato recibido por el puerto serie vía Bluetooth. El código completo que se ejecuta en la ESP32 se puede revisar también en el Github de este proyecto [9].

Por último, habiendo programado todas las funciones de Bluetooth en el código como la creación del servicio y la característica, nos vamos a centrar en la función loop que está en continua ejecución. Como se observa en la Figura 5.9 lo primero que hacemos es añadir una condición para indicar que la placa Nicla está disponible y en caso de no estar conectada no entrará en el resto de la ejecución. Si el puerto serie está operativo se crea una variable de tipo char que será el dato recibido. Cada vez que se reciba un dato la variable cambiará su valor y se escribirá en la característica Bluetooth previamente definida, notificando también al dispositivo conectado de que se ha enviado un dato nuevo. Para ello volvemos al concepto de característica explicado en la Sección **??**. Una vez sobreescrito el valor lanzamos la función *notify()* que avisará a la Raspberry Pi de que hay un nuevo valor en

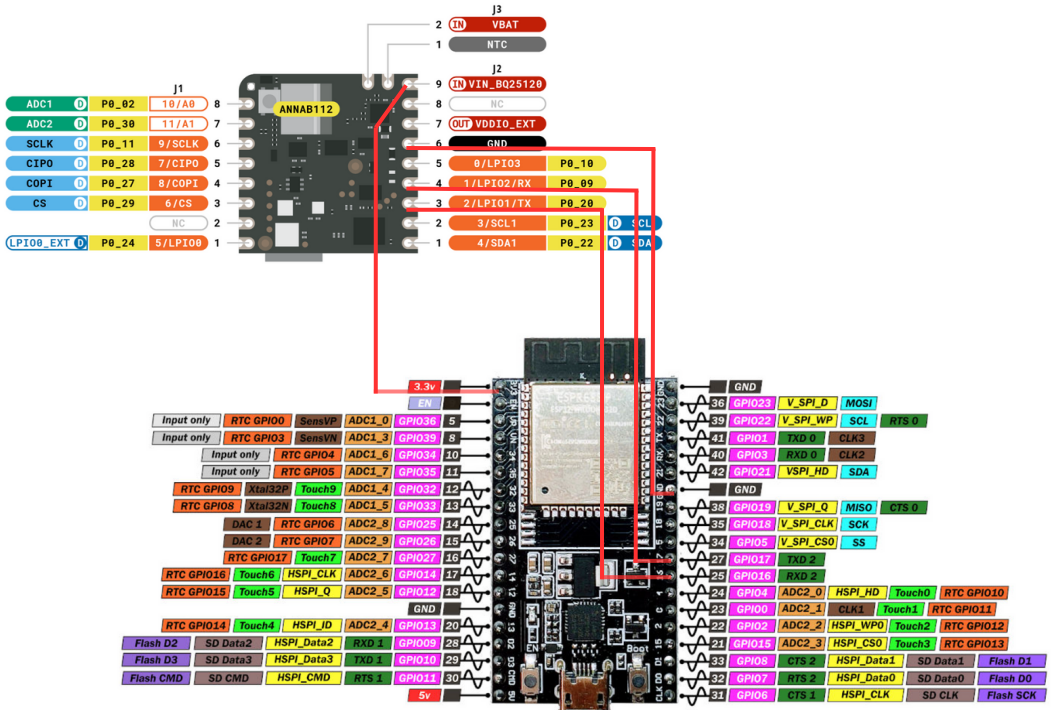

Figura 5.8: Esquema de las conexiones de los pines realizadas entre la placa Nicla Sense ME y la placa ESP32.

dicha característica.

```
void loop() {
    if (SerialNICLA.available()) {
      char data = SerialNICLA.read();
      // Procesamos el dato recibido y establecemos el valor de la caracteristica
      value = data;
      pCharacteristic->setValue((uint8_t*)&value, sizeof(value));
      pCharacteristic->notify();
  }
}
```

Figura 5.9: Código la función loop ejecutada en la ESP32.

En la Sección 5.4 continuaremos viendo cómo se conecta la Raspberry a la placa ESP32 y cómo recibe el dato de la característica Bluetooth definida.

5.3. Detección de la posición de la mano usando MediaPipe

Como se ha definido en la Sección 4.3, el software elegido para poder detectar la posición de la mano con 21 puntos de referencia ha sido MediaPipe, que se ejecutará en la Raspberry Pi 4.

Una vez instalada la librería en Python ya podemos incluirla en nuestros proyectos

importándola como cualquier otra librería. En el código debemos de crear el modelo de Hand Landmarks desde MediaPipe e inicializarlo con los parámetros deseados.

A continuación vamos a ver algunos de estos parámetros que podemos configurar:

- **running_mode :** Permite configurar el modo de ejecución para los Hand Landmarks, pudiendo elegir entre IMAGE, VIDEO o LIVE_STREAM.
- **num_hands :** Permite cambiar el número de manos que detecta el modelo, siendo el mínimo 1.
- **min_hand_detection_confidence :** Indica el valor mínimo de fiabilidad que se requiere para reconocer la mano completa. Este valor debe estar entre 0.0 y 1.0.
- **min_hand_presence_confidence :** Indica el valor mínimo de fiabilidad para el reconocimiento de los puntos de referencia sobre la mano. Este valor debe estar entre 0.0 y 1.0.
- **min_tracking_confidence :** Indica el valor mínimo de fiabilidad para considerar que la detección de la mano ha sido válida. Este valor debe estar entre 0.0 y 1.0.

En la Figura 5.10 se puede ver el fragmento de código utilizado para la creación del modelo de reconocimiento de manos, junto con los parámetros establecidos para hacerlo más eficiente.

```
# Grabbing the Hand Model from Mediapipe and
# Initializing the Model
mp_hand = mp.solutions.hands
hands = mp_hand.Hands(
        min_detection_confidence=0.2,
        model_complexity=0,
        max_num_hands=1
)
```

Figura 5.10: Código para la creación del modelo Hand Landmarks junto a los parámetros definidos.

Una vez que hemos creado el modelo MediaPipe nos da la libertad para poder leer la posición concreta de cualquier punto de referencia en base a tres coordenadas. Esto nos permite dibujar los puntos sobre la mano al usar la cámara así como saber la posición para ejecutar las acciones que queramos. En nuestro caso usamos el punto de referencia número 9 que se corresponde con el comienzo del dedo corazón, el cual nos sirve como punto central de la mano y gracias a esto sabemos la posición de la mano respecto al eje X y al eje Y. Hecho esto al ejecutar el programa veremos la cámara y los puntos de referencia dibujados sobre la mano que está detectando como se ve en la Figura 5.11

Al detectar los puntos de referencia y saber sus posiciones respecto a los ejes el código se encarga de procesar esta información y envía, usando el protocolo OSC, la información necesaria para interacción con la música. En la Figura 5.12 vemos un fragmento de este código que en este caso usa el punto de referencia número 9, el cual es el punto central de la mano. Dependiendo del valor de este dato se enviará mediante OSC el número correspondiente.

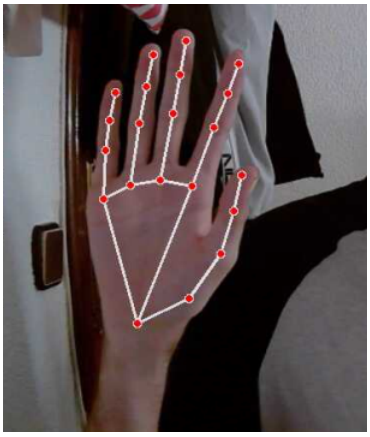

Figura 5.11: Imagen de la cámara en la Raspberry Pi detectando los puntos de referencia en la mano.

```
if id==9 :
        num9y = landmark.y
        num9x = landmark.x

        if num9y > 0.8:
                sender.send_message("/estado", int(2))
        elif num9y > 0.6:
                sender.send_message("/estado", int(3))
        elif num9y > 0.4:
                sender.send_message("/estado", int(4))
        elif num9y > 0.2:
                sender.send_message("/estado", int(5))
        else:
                sender.send_message("/estado", int(6))
```

Figura 5.12: Fragmento del código que usa MediaPipe para obtener el valor del punto de referencia número 9 y enviar el mensaje OSC.

5.4. Conexión entre dispositivos y software

En las secciones anteriores hemos visto en detalle cómo funcionan las placas Nicla Sense ME y ESP32, las cuales se encargan de hacer la inferencia del gesto que realizamos y de mandar la información necesaria a través de Bluetooth. En esta sección nos vamos a centrar en los scripts de Python que se ejecutan en la Raspberry Pi y que se encargan de recibir dicho dato, procesarlo y reenviarlo usando el protocolo OSC.

En el script necesario para conectarse por Bluetooth a la ESP32 se usa principalmente la librería Bleak [5]. Esta librería se comporta como un cliente GATT que puede conectarse a dispositivos BLE que actúan como servidores GATT. Está diseñada para proporcionar una API de Python asíncrona y multiplataforma para conectar y comunicarse con sensores u otros dispositivos BLE. Soporta leer, escribir y recibir notificaciones de los servidores GATT, así como una función para descubrir dispositivos BLE.

También usamos en este script la librería python-osc [6]. Gracias a esta librería podemos implementar servidores y clientes OSC (Open Sound Control) en Python. Como ya hemos visto anteriormente, OSC es un protocolo de comunicación entre computadoras, sintetizadores de sonido y otros dispositivos multimedia que está optimizado para la tecnología de

redes moderna. Permite enviar y recibir mensajes OSC con diferentes tipos de datos, como enteros, flotantes, cadenas, MIDI, timestamps y blobs. Además ofrece varias opciones para crear servidores OSC, como bloqueantes, con hilos, con procesos o con asyncio.

Durante el desarrollo de este proyecto se han ido creando diversos códigos para entender y aprender todas las funciones de estas librerías, hasta llegar al código final que permite conectarnos a la placa ESP32 y recibir la información en el momento en el que haya un valor nuevo, es decir, que la placa ESP32 mande una notificación. Para ello en el código se debe indicar la dirección de la placa así como el UUID tanto del servicio como de la característica, ambos UUID se encuentran definidos en el código que ejecuta la ESP32. En la Figura 5.13 podemos ver en detalle el fragmento de código que se ejecuta haciendo uso de la librería Bleak para conectarse a la placa ESP32 y activar la función que se ejecutará cada vez que se reciba la notificación de que hay un dato nuevo.

```
async def run(address, service_uuid, char_uuid):
        async with BleakClient(address) as client:
                await client.start_notify(char_uuid, notif_handler)

                while True:
                        await asyncio.sleep(1)
```

Figura 5.13: Código que se encarga de conectarse al dispositivo BLE y comenzar la función de notificación.

La función que se ejecuta cada vez que se recibe una notificación por parte de la ESP32 se denomina 'notif_handler' y se puede ver en la Figura 5.14. Lo que hace esta función es procesar el dato recibido de la característica BLE y enviarlo a través del protocolo OSC usando la función 'send_message'.

```
async def notif_handler(uuid, value: bytearray):
        int_value = struct.unpack("i", value)[0]
        print("Dato: ", int(int_value))
        print("Byte_array: ", value)
        sender.send_message("/estado", int(int_value))
```

Figura 5.14: Función de notificación encargada de procesar el dato recibido y enviarlo por OSC.

5.5. Música interactiva con interfaz gestual

Finalmente todo el desarrollo que hemos descrito en las secciones anteriores se pone en uso en el código que ejecuta la librería Pyo, que como ya sabemos es la encargada de reproducir la música generativa sobre la que interactuaremos. En la Figura 5.15 podemos ver el código que se usa para generar la melodía, la cual está compuesta por diferentes generadores de ondas y de notas que van cambiando conforme se reproduce la melodía.

Ahora vamos a ir viendo la función de cada uno de los elementos vistos en el código. El primero es el denominado *wav*, que llama a la función *SquareTable()* para crear un generador de onda cuadrada. A continuación el elemento llamado *env* crea una tabla de

```
s.start()
wav = SquareTable()
env = CosTable([(0,0), (100,1), (500,.3), (8191,0)])
met = Metro(.125, 12).play()
amp = TrigEnv(met, table=env, mul=.1)
pit = TrigXnoiseMidi(met, dist='loopseg', x1=20, scale=1, mrange=(48,84))
out = Osc(table=wav, freq=pit, mul=amp).out()
```

Figura 5.15: Código que usa la librería Pyo para reproducir una pieza de música generativa.

fragmentos interpolados de una onda coseno. Se añade también un objeto metrónomo (*met*) para generar disparadores periódicos que hagan sonar las notas. El elemento *amp* crea una envolvente usando los objetos que acabamos de describir. Después tenemos *pit* que vuelve a usar el metrónomo junto a otros parámetros para crear una distribución de ruido que genera un nuevo valor cada vez que se llama a la función.

Por último el objeto *out* incorpora los elementos anteriormente descritos en un oscilador que utiliza *wav* como señal, *pit* como frecuencia de ese oscilador y *amp* como envolvente. Este oscilador conforma la melodía al completo. De este objeto destacamos la función *.out()*, que es la encargada de enviar el sonido al dispositivo de salida conectado a la Raspberry.

En este código también se incluye el procesamiento de los mensajes OSC recibidos tanto de MediaPipe como del gesto detectado por la Nicla Sense ME. En la Figura 5.16 podemos ver cómo se cambian los parámetros como la distribución o la duración de las notas en base al número recibido por OSC, más adelante veremos en detalle esta interacción.

```
# Mano abierta/cerrada
if number==1:
        pit.setDist('weibull')
if number==0:
        pit.setDist('poisson')

#Eje Y
if number==2:
        wav.setOrder(1)
if number==3:
        wav.setOrder(2)

...

# Eje X
if number==7:
        amp.setDur(2)
if number==8:
        amp.setDur(1.5)
```

Figura 5.16: Código que usa la librería Pyo para reproducir una melodía generativa.

Al ejecutar el script de MediaPipe el cual inicia la cámara y nos muestra los puntos de referencia de la mano junto al script de Pyo que se acaba de describir podemos empezar a interactuar con la melodía que se genera.

Tenemos la posibilidad de mover la mano por toda la imagen que captura la cámara para cambiar diversos parámetros. Si la movemos en el eje X cambiaremos la duración de las notas que están sonando, es decir, al situar la mano en el centro del eje X las notas tendrán una duración estándar, mientras que si la movemos hacia la derecha en este eje conseguiremos cada vez una duración mayor de las mismas, llegando incluso al doble de la duración estándar. Al mover la mano hacia la izquierda en el eje X reduciremos la duración consiguiendo que duren hasta 4 veces menos que la duración estándar.

Respecto al movimiento en el eje Y nos encontramos con un comportamiento similar, pero en este caso se cambiará el volumen y el orden de algunas de las notas. En el centro del eje Y obtendremos el volumen estándar, mientras que si subimos en el eje subiremos el volumen y si bajamos la mano, el volumen hará lo mismo.

Por último se ha implementado la posibilidad de abrir y cerrar la mano para cambiar el tono de las notas. Si mantenemos la mano abierta se oirá un tono más grave mientras que si la cerramos las notas sonarán mucho mas agudas. Esto se consigue al cambiar la distribución del objeto *pit*, con una distribución Weibull obtenemos las notas más graves mientras que con una distribución Poisson obtenemos las más agudas.

En cuanto a los gestos detectados por la placa Nicla Sense ME destacamos que no tienen que ser capturados por la cámara sino que deben de ser dibujados en el aire. Estos gestos están inspirados en los que realiza un director de orquesta al dirigir la misma, aunque la ejecución sea similar no pretenden tener la misma utilidad que estos. Los gestos son los vistos en la Sección 5.1, recordamos que entrenamos el gesto *Corte* para pausar la reproducción de la música y el gesto *Inicio* para reanudarla.

De esta forma podemos crear melodías interminables que van cambiando con el tiempo, participando activamente en su generación y teniendo la posibilidad de generar diferentes emociones según queramos, como tensión o relajación.

Capítulo 6

Conclusiones y Trabajo Futuro

En el último capítulo de este trabajo vamos a hacer un repaso por los objetivos que teníamos, concluyendo si se han cumplido y qué dificultados nos hemos encontrado para conseguir llevarlos a cabo. Como trabajo futuro abriremos las puertas a las vías de desarrollo que se pueden tener para ampliar este trabajo y mejorarlo en diversos aspectos.

Como ya sabemos el objetivo principal de este proyecto es permitir la interacción del usuario con la melodía generativa que se reproduce. Dada la naturaleza exploratoria de este proyecto decidimos desarrollar dos formas de interacción, una basada en la posición de la mano vista por una cámara y otra basada en los gestos que se dibujen en el aire a partir de acelerometría gracias a una pequeña placa colocada en la mano. Estos objetivos se determinaron teniendo en cuenta que existen herramientas tecnológicas que los hacen posibles. En cuanto a hardware tenemos disponibles pequeñas placas autónomas que permiten hacer la inferencia en la propia placa mientras que en software tenemos librerías específicas para la detección de la mano las cuales facilitan el desarrollo devolviendo unos datos normalizados. Consideramos un acierto haber elegido Python como lenguaje de programación para el desarrollo de la mayor parte del proyecto ya que posee una gran documentación y apoyo por parte de la comunidad. Gracias a esto hemos podido integrar los elementos de hardware elegidos de una manera sencilla y que facilite la comunicación entre ellos.

Se han conseguido integrar tecnologías diversas de hardware y software, incluyendo entre ellas un modelo de Inteligencia Artificial. De este modelo también queremos destacar su facilidad de entrenamiento ya que Edge Impulse nos facilita mucho este proceso. La plataforma puede parecer abrumadora al principio debido a la gran cantidad de opciones y parámetros a configurar que nos permite, pero una vez entendido esto el proceso de entrenamiento resulta sencillo. Esto nos anima a expandir el proyecto en este marco explotando aún más el potencial de la plataforma.

Este proyecto ha sido posible gracias a los avances tecnológicos de los últimos años y la expansión de estas herramientas al público general de una manera relativamente asequible. De esta forma también se pretende acercar conceptos como música generativa y dispositivos IoT a la gente desde el entretenimiento, haciendo este acercamiento más fácil y mostrándoles un punto de partida a futuros desarrollos de una manera diferente a la convencional.

La conclusión principal que obtenemos es que este proyecto abre un campo inmenso respecto al uso de música generativa y las formas en las que podemos interactuar con ella, entrando así en la unión de tecnología y música con una visión innovadora en la cual encontramos un campo muy interesante sobre el que seguir trabajando para ofrecer nuevas

experiencias al público de estos proyectos.

Personalmente me ha parecido un trabajo muy interesante y todo un reto. Ha sido la primera vez que desarrollaba un sistema IoT desde cero y por suerte la investigación de trabajos anteriores ha sido de gran ayuda. En cuanto a la parte musical tenía muy poco conocimiento sobre ella y he aprendido bastante a base de ejemplos y de desarrollar melodías en Pyo para entender cada elemento. Me resulta muy gratificante comprobar que el objetivo principal se ha cumplido, sobre todo teniendo en cuenta las trabas que han surgido durante el desarrollo que por momentos me hacían pensar que el proyecto no iba a salir adelante.

En cuanto al coste del proyecto en total vamos a ver el precio de los elementos hardware principalmente. En la Tabla 6.1 se encuentra el desglose de los diferentes elementos junto al precio aproximado de cada uno.

Dispositivo	**Uso**	**Precio**
Nicla Sense ME	Detección de gestos	70€
ESP32-DevKitC V4	Envío de información por BLE	10€
Raspberry Pi 4 Model B	Ejecución de música generativa e interacción con ella	50€
HOMSCAM Webcam	Detección de la posición de la mano	15€

Tabla 6.1: Tabla de precios de los dispositivos hardware.

En el apartado software hemos reducido el coste a 0€ usando librerías de código abierto y la versión gratuita de herramientas como Edge Impulse, las cuales hemos comprobado que son más que suficientes para este desarrollo. Por lo tanto el coste total del proyecto parte de 145€.

6.1. Dificultades del proyecto

Como se ha comentado anteriormente durante el desarrollo nos hemos encontrado con diferentes problemas para los cuales hemos necesitado modificar el proyecto de alguna manera para poder llevarlo a cabo. El principal problema fue encontrarnos con que la placa Nicla Sense ME no poseía la potencia necesaria para ejecutar el modelo de inteligencia artificial que reconoce el gesto a la vez que ejecutaba la librería de Bluetooth para enviarlo. Esto supuso un gran problema en el desarrollo y un retraso considerable en el plan de trabajo, ya que pensamos que la Nicla Sense ME podía ejecutar ambas librerías sin problema. Tras investigar en esta materia vimos que era un problema común en diversos proyectos y que puede tener solución si se optimizara la librería de la propia Nicla Sense ME por parte del equipo de Arduino. Para solventar este problema decidimos añadir una segunda placa de desarrollo, en este caso la placa ESP32, la cual ya conocía y había usado anteriormente. Esta placa se encargaría de ejecutar la librería Bluetooth y enviar el dato que recibe por serial desde la Nicla Sense ME. Otro de los problemas que nos encontramos también fue relativo a la potencia, en este caso de la Raspberry Pi 4, la cual sufría ralentizaciones al momento de ejecutar la librería de MediaPipe junto a la previsualización de la cámara. Este problema tuvo una solución más sencilla ya que pudimos optimizar los parámetros

por defecto que incluía MediaPipe para así conseguir un mejor rendimiento y que no se ralentizara.

Pese a los problemas encontrados y el largo desarrollo finalmente conseguimos llegar a los objetivos establecidos, consiguiendo la melodía generativa capaz de ser modificada en tiempo real por el usuario.

6.2. Trabajo Futuro

En cuanto al trabajo futuro partiendo de este proyecto destacaremos varias vías de investigación. La primera de ellas sería desarrollar una melodía generativa mucho más compleja que incorpore más elementos. Con el tiempo limitado y la importancia del desarrollo para los componentes hardware solamente conseguimos crear una melodía generativa sencilla basada en un ejemplo de Pyo. Sin embargo es más que suficiente para entender el concepto de música generativa y ver el potencial que puede llegar a tener.

Otra línea de investigación sería añadir más variedad a la interacción que conseguimos usando MediapPipe, ya que actualmente detecta la posición en los ejes X e Y además de detectar si la mano está abierta o cerrada. Las funcionalidades que se pueden añadir podrían ir relacionadas con la posición individual de cada dedo o añadir soporte para dos manos a la vez permitiendo de esta manera una interacción mucho mayor con la melodía generativa.

Respecto a la funcionalidad de detección de gestos dibujados en el aire hemos explicado que los gestos entrenados están inspirados en los que hace un director de orquesta. Pensamos que un avance en este campo puede ir relacionado con añadir una funcionalidad completamente igual a un director de orquesta, para lo que posiblemente sea necesario tener una placa en cada mano y recoger los datos de ambas placas a la vez. De esta manera podríamos conseguir una funcionalidad mas fiel a la dirección de orquesta y podríamos crear melodías complejas cambiando más parámetros o superponiendo instrumentos y sonidos.

Chapter 7

Introduction

Entertainment in any of its forms is an activity that can provide us with fun and distraction, helping us to relax and escape from the daily problems we have to face [30]. One of the most popular forms of entertainment is music, which is the art of combining sounds and silences in a harmonious and expressive way. Music can even come to define us as people since through it we can reflect our emotions. It also influences our mood, as it can make us feel happy, calm or motivated.

A very important part of music is that it can improve our cognitive performance, since it stimulates the brain and favors memory, attention and creativity. This point is the one we want to emphasize the most, since this project combines music with user interaction, favoring creativity.

In recent years, technology and computing have caused a revolution in the options available to us for entertainment [13], offering new possibilities and experiences. For example, visual effects have made it possible to create increasingly realistic images as well as to simulate impossible things such as monsters, explosions or fantastic scenarios. Streaming allows us to access a wide variety of audiovisual content, such as movies, series or music through the Internet and without the need to download them. Computing has facilitated the development of applications, services and devices that allow access to these contents in a personalized and flexible way, whereas in the past the only way to enjoy some of these forms of entertainment was a more complex task.

The relationship between music and technology is very close, so much so that the latter has influenced the evolution of music throughout history [2]. Technology has affected music from creation and production to distribution and consumption. Regarding music creation, technology has allowed the development of new musical instruments, such as synthesizers, drum machines, samplers and MIDI controllers, which expand the sonic and creative possibilities for musicians. Technology has also facilitated the use of software and mobile applications that allow composing, editing, mixing and mastering music digitally. It is becoming easier and easier for the average person to enter the world of music and use tools that were previously intended strictly for music professionals, bringing music creation to many more people.

In conclusion, technology and music are intimately related. Combining music as we know it with IoT tools such as the development boards that we will see, we obtain an innovative project that makes us participants in the reproduction of music.

7.1. Motivation

As we know, entertainment is a crucial part of society, therefore, just as society changes over time, entertainment is not static, but adapts to social, cultural, technological and economic changes that occur in the world.

Today's society is characterized by being dynamic, complex and diverse. Technological advances or globalization are some of the factors that influence the way people entertain themselves [14]. Because of this, we find the need to create new forms of entertainment that respond to the needs, interests and expectations of different audiences.

These new forms of entertainment must be innovative and creative, in other words, original, surprising and stimulating experiences for the audience. Another key point is that they must be interactive and participatory, allowing the audience to be the protagonist in a way that allows them to choose, give their opinion and collaborate alone or with other people in the entertainment process. This can be seen in platforms such as Netflix, which includes interactive movies in which the viewer can make decisions that change the fate of the protagonist [11]. In addition, much of the entertainment is intended to be educational and cultural, thus contributing to the personal and social development of the audience, encouraging learning, reflection and respect for diversity. Certain entertainment content can bring us closer to areas that otherwise we would not have known or simply would not have been interested in.

With all of the above in mind we think that creating new forms of entertainment is a necessity in a continuously changing society, being able to adapt to such changes and offer satisfactory experiences for the public. That is why in this work we want to bring a form of entertainment based on the direct interaction of the user with the music that is generated in real time, joining this also to the knowledge about technology and IoT devices acquired during this master and during my internship, where I have been much closer than the technology used and I have had the opportunity to use hardware such as the Nicla Sense ME. In this way it is also intended to bring users closer to generative music as well as IoT devices in an interactive and entertaining way.

7.2. Project Objectives

The general objective of this project is to have an interactive system with the generation of music in real time, controlled by the position and gestures made with the hand. From this objective we have proposed the following specific objectives to carry out this project:

- Detection of a gesture drawn in the air using the Nicla Sense ME board, exploiting its capability as much as possible by making the inference on the board itself.
- Hand detection and tracking using MediaPipe, performing this with the limited power of the Raspberry.
- Generative music playback from input parameters using Python language and tools such as Pyo.
- User interaction with the generated music allowing to modify its execution in real time.
- Use of Raspberry Pi as the main computer of the project, benefiting from its small size and portability.

7.3. Workplan

Based on the objectives described in the previous section, which were reviewed with the tutors of this work, a workplan was proposed to achieve these objectives.

7.3.1. Exploration of the available hardware tools

To develop this project we must investigate the hardware available to us, among which are the Bosch development boards, such as the Nicla Sense ME, other boards used in the master as the ESP32 or more powerful elements on which to run most of the software as the Raspberry Pi 4.

7.3.2. Researching previous work

Once we know what we can use for development we investigate other projects that use that hardware and relate to our vision. This way we start from a base which helps us with the choice of software, as there are a lot of tools available.

7.3.3. Exploration of the available software

Knowing the needs of this project and taking into account the projects already developed we must make an investigation of the software that we will use to program the hardware tools. We need a software that is able to play a generative melody that can be interacted with and that also allows an easy communication with the rest of the devices involved.

7.3.4. Generative music development

Once we have the chosen hardware and software tools working together we can move on to the creation of a generative piece of music. This melody will be infinite and will change over time according to some rules established in the code. It must also allow to modify certain parameters while it is running.

7.3.5. Gestural interface development

Finally we must create a gestural interface which sends information to the code that plays the generative melody and changes its behavior. This interface will be based on the position of the hand regarding the camera and on gestures drawn in the air.

7.4. Document structure

In this section we will briefly see how this document is structured, making an overview of each of the chapters that are part of it

- Introduction. This chapter includes the motivation for the project, the objectives, the workplan and the structure of the document.
- State of the art. Previous projects that directly relate to the concepts of generative music and interaction with it, as well as gesture detection, are reviewed.

- Hardware architecture of the system. Definition of the hardware elements that conform this project, describing them and seeing the possibilities they offer when using them in the development.
- Software architecture of the project. Definition of the software tools that have been used in the development of the project. It is detailed in which device they are executed and what they are used for.
- Project development. This chapter details step by step how the project has been developed, integrating the hardware and software tools previously seen, from their connection to the detailed use of each one.
- Conclusions and Future Work. The last chapter reviews the achievements of this project, while drawing conclusions, and reviews the future work it may lead to.

7.5. Project availability

The entire code developed for this project is available in a public Github repository [9] created for this purpose.

Chapter 8

Conclusions and Future Work

In the last chapter of this work we will review the objectives we had, concluding whether they have been met and what difficulties we have encountered in achieving them. As a future work, we will open the doors to the ways of development that can be taken to expand this work and improve it in various aspects.

As we already know the main objective of this project is to allow user interaction with the generative melody being played. Given the exploratory nature of this project we decided to develop two forms of interaction, one based on the position of the hand seen by a camera and the other based on gestures drawn in the air from accelerometry thanks to a small board placed on the hand. These objectives were determined taking into account that there are technological tools that make them possible. In terms of hardware we have available small autonomous boards that allow to make the inference on the board itself while in software we have specific libraries for the detection of the hand which facilitate the development by providing normalized data. We consider it a good decision to have chosen Python as the programming language for the development of most of the project as it has a great documentation and support from the community. Thanks to this we have been able to integrate the chosen hardware elements in a simple way that facilitates communication between them.

Several hardware and software technologies have been integrated, including an Artificial Intelligence model. From this model we also want to highlight its ease of training, as Edge Impulse makes this process much easier. The platform may seem overwhelming at first due to the large number of options and parameters to configure, but once this is understood, the training process is simple. This encourages us to expand the project in this framework to further exploit the potential of the platform.

This project has been possible thanks to the technological advances of recent years and the expansion of these tools to the general public in a relatively affordable way. In this way it is also intended to bring concepts such as generative music and IoT devices to people through entertainment, making this approach easier and showing them a starting point for future developments in a different way than the traditional approach.

The main conclusion we get is that this project opens a huge field regarding the use of generative music and the ways in which we can interact with it, thus entering the union of technology and music with an innovative vision in which we find a very interesting field on which to continue working to offer new experiences to the public of these projects.

Personally I found the work very interesting and challenging. It was the first time I developed an IoT system from scratch and luckily the research of previous works has been

a great help. As for the musical part I had very little knowledge about it and I have learned a lot based on examples and developing melodies in Pyo to understand each element. It is very rewarding to see that the main objective has been met, especially considering the obstacles that have arisen during the development that at times made me think that the project would not go ahead.

As for the total cost of the project we are going to see the price of the hardware elements mainly. In Table 8.1 you can find the breakdown of the different elements together with the approximate price of each one.

Device	**Application**	**Price**
Nicla Sense ME	Gesture detection	70€
ESP32-DevKitC V4	Send information through BLE	10€
Raspberry Pi 4 Model B	Execution and interaction with generative music	50€
HOMSCAM Webcam	Hand position detection	15€

Table 8.1: Table of hardware devices prices.

In the software area we have reduced the cost to 0€ using open source libraries and the free version of tools such as Edge Impulse, which we have found to be more than enough for this development. Therefore the total cost of the project starts at 145€.

8.1. Project difficulties

As previously mentioned during the development we encountered different problems for which we needed to modify the project in some way to be able to complete it. The main problem was that the Nicla Sense ME board did not have the necessary performance to run the artificial intelligence model that recognizes the gesture while running the Bluetooth library to send it. This was a major problem in the development and a considerable delay in the work plan, since we thought that the Nicla Sense ME could run both libraries without any problem. After researching this matter we realized that it was a common problem in several projects and that it could be solved if the Nicla Sense ME library itself was optimized by the Arduino team. To solve this problem we decided to add a second development board, in this case the ESP32 board, which I already knew and had used before. This board would be responsible for running the Bluetooth library and send the data received by serial from the Nicla Sense ME. Another problem we encountered was also related to the performance, in this case of the Raspberry Pi 4, which suffered slowdowns when running the MediaPipe library along with the camera preview. This problem had a simpler solution as we were able to optimize the default parameters included in MediaPipe to achieve better performance and not slow down.

Despite the problems encountered and the long development, we finally managed to reach the established objectives, achieving the generative melody capable of being modified in real time by the user.

8.2. Future work

In terms of future work based on this project we will highlight various lines of research. The first would be to develop a much more complex generative melody that incorporates more elements. With limited time and the importance of development for hardware components we only managed to create a simple generative melody based on a Pyo example. However it is more than enough to understand the concept of generative music and see the potential it can have.

Another line of research would be to add more variety to the interaction we achieve using MediapPipe, since it currently detects the position in the X and Y axes in addition to detecting whether the hand is open or closed. The functionalities that could be added could be related to the individual position of each finger or add support for two hands at the same time allowing a much greater interaction with the generative melody.

Regarding the functionality of detecting gestures drawn in the air we have explained that the trained gestures are inspired by those made by a conductor. We think that a breakthrough in this field could be related to adding a functionality completely equal to a conductor, for which it may be necessary to have a board in each hand and collect data from both boards at the same time. In this way we could achieve a functionality more faithful to the orchestra conductor and we could create complex melodies by changing more parameters or superimposing instruments and sounds.

Bibliografía

[1] Artículo sobre el grupo de investgación en tecnología musical. https://www.upf.edu/es/web/e-noticies/entrevistas/-/asset_publisher/wEpPxsVRD6Vt/content/el-grup-de-recerca-en-tecnologia-musical-compleix-25-anys/10193/. Consultado en julio, 2023.

[2] Cómo ha afectado la tecnología en la música. https://originalmusic.es/blog/como-ha-afectado-la-tecnologia-en-la-musica/. Consultado en agosto, 2023.

[3] Documentación sobre el uso de Edge Impulse en Nicla Sense ME. https://docs.edgeimpulse.com/docs/development-platforms/officially-supported-mcu-targets/arduino-nicla-sense-me. Consultado en mayo, 2023.

[4] Documentación sobre la instalación de la herramienta CLI. https://docs.edgeimpulse.com/docs/tools/edge-impulse-cli/cli-installation. Consultado en mayo, 2023.

[5] Documentación sobre la librería Bleak: . https://pypi.org/project/bleak/. Consultado en junio, 2023.

[6] Documentación sobre la librería Python-osc. https://pypi.org/project/python-osc/. Consultado en julio, 2023.

[7] Github de la demo de dashboard para la Nicla Sense ME. https://github.com/arduino/ArduinoAI/blob/main/NiclaSenseME-dashboard/NiclaSenseME/NiclaSenseME.ino. Consultado en mayo, 2023.

[8] Github de la librería Pyo. https://github.com/belangeo/pyo/tree/master. Consultado en junio, 2023.

[9] Github del código desarrollado para el TFM. https://github.com/manuelmur/TFM. Consultado en septiembre, 2023.

[10] How Brian Eno Created Ambient 1: Music for Airports. https://reverbmachine.com/blog/deconstructing-brian-eno-music-for-airports/. Consultado en julio, 2023.

[11] How Netflix makes its choose-your-own adventure movies. https://www.polygon.com/22712738/netflix-interactive-interview-ben-simms. Consultado en agosto, 2023.

[12] Introduction to Bluetooth Low Energy - GATT. https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt. Consultado en agosto, 2023.

[13] Los avances tecnológicos en el mundo del entretenimiento. https://revistabyte.es/actualidad-it/los-avances-tecnologicos-en-el-mundo-del-entretenimiento/. Consultado en agosto, 2023.

[14] Los cinco cambios que definen el entretenimiento del futuro. https://www.audiovisual451.com/los-cinco-cambios-que-definen-el-entretenimiento-del-futuro/. Consultado en agosto, 2023.

[15] Reactable - Music Technology Group. https://www.upf.edu/web/mtg/reactable. Consultado en julio, 2023.

[16] VOCALOID - the modern singing synthesizer. https://www.vocaloid.com/en/. Consultado en julio, 2023.

[17] Web de Arduino IDE. https://www.arduino.cc/en/software. Consultado en abril, 2023.

[18] Web de la plataforma Edge Impulse. https://www.edgeimpulse.com/. Consultado en mayo, 2023.

[19] Web de Mediapipe para la detección de Hand Landmarks. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker. Consultado en junio, 2023.

[20] Web del framework MediaPipe. https://developers.google.com/mediapipe. Consultado en junio, 2023.

[21] Web oficial del lenguaje python. https://www.python.org/. Consultado en mayo, 2023.

[22] Web oficial del producto Raspberry Pi 4. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. Consultado en mayo, 2023.

[23] Arduino. Documentación oficial de Nicla Sense ME. https://docs.arduino.cc/hardware/nicla-sense-me. Consultado en abril, 2023.

[24] Karen Collins. An introduction to procedural music in video games. *Contemporary Music Review*, 28(1):5–15, 2009.

[25] Brian Eno. A talk delivered in San Francisco, June 8, 1996 by Brian Eno. https://inmotionmagazine.com/eno1.html, 1996. Consultado en julio, 2023.

[26] Espressif. Documentación oficial de la placa ESP32. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html. Consultado en mayo, 2023.

[27] Robin Heydon and Nick Hunn. Bluetooth low energy. *CSR Presentation, Bluetooth SIG https://www. bluetooth. org/DocMan/handlers/DownloadDoc. ashx*, 2012.

[28] Mostafa Lotfi. Github del proyecto Paper-Piano. https://github.com/MustafaLotfi/Paper-Piano, 2023. Consultado en mayo, 2023.

[29] Justin Lutz. Arduino x K-Way - Gesture Recognition for Hiking. `https://docs.edgeimpulse.com/experts/featured-machine-learning-projects/arduino-kway-gesture-recognition-weather`, 2022. Consultado en abril, 2023.

[30] José Samuel Martínez López. Sociedad del entretenimiento (2): construcción sociohistórica, definición y caracterización de las industrias que pertencen a este sector. *Luciérnaga Comunicación*, 2015.

[31] James McCartney. Github del proyecto Supercollider. `https://github.com/supercollider/supercollider`, 1996. Consultado en mayo, 2023.

[32] Eric Peña and Mary Grace Legaspi. Uart: A hardware communication protocol understanding universal asynchronous receiver/transmitter. *Analog Dialogue*, 2020.

[33] Zach Scholl. Github del proyecto Airsynth. `https://github.com/schollz/airsynth`, 2021. Consultado en mayo, 2023.

[34] MATTHEW WRIGHT. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3), 2005.