UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DESARROLLO DE UN SISTEMA DE INTERACCIÓN PARA APLICACIONES AUDIOVISUALES CON MICROPYTHON

María Bayo Olmeda

Tutor: Germán Ramos Peinado

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2021-22

Valencia, 2 de diciembre de 2021

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV- 000030

Resumen

Este trabajo aborda el desarrollo de un sistema electrónico de bajo coste que permita la interacción hombre-máquina en tiempo real para aplicaciones audiovisuales. Este sistema captará y procesará la información de varios sensores y la enviará en distintos formatos y de forma inalámbrica a terceras aplicaciones softwares empleadas en el mundo audiovisual. De esta forma, el usuario podrá interactuar con estos softwares con sus propios movimientos y gestos.

Como sistema microcontrolador se ha empleado un dispositivo ESP32 por su bajo coste, gran potencia de cálculo, y capacidad de comunicaciones inalámbricas por Buetooth y WiFi. Se incorporarán diversos sensores enfocados a la interacción como acelerómetros, giróscopos, magnetómetros, emisores y receptores de ultrasonidos, resistencias flexibles.

Se ha empleado el lenguaje de programación interpretado Python en su versión reducida para sistemas embebidos MicroPython, evaluando a la vez su potencialidad en este tipo de aplicaciones frente a lenguajes compilados como Arduino en C/C++ que son los habitualmente usados.

Para la comunicación con terceras aplicaciones a través de comunicaciones inalámbricas IP se ha usado el formato JSON y el protocolo OSC, ambos ampliamente empleados en este tipo de aplicaciones.

Finalmente, el sistema se ha validado con los software SoundCool y TouchOSC para la interacción con aplicaciones de generación y procesado de audio y video.

Resum

Aquest treball aborda el desenvolupament d' un sistema electrònic de baix cost que permetra la interacció home-màquina en temps real per a aplicacions audiovisuals. Aquest sistema captarà i processarà la informació de diversos sensors i l' enviarà en diferents formats i de forma inalàmbrica a terceres aplicacions softwares empleades en el món audiovisual. D'aquesta forma, l'usuari podrà interactuar amb aquests softwares amb els seus propis moviments i gestos.

Com a sistema microcontrolador s'ha empleat un dispositiu ESP32 pel seu baix cost, gran potència de càlcul, i capacitat de comunicacions Buetooth i WiFi. S' incorporaran diversos sensors enfocats a la interacció com acceleròmetres, giròscops, magnetòmetres, emissors i receptors d' ultrasons, resistències flexibles.

S'ha usat el llenguatge de programació interpretat Python en la seva versió reduïda per a sistemes embeguts MicroPython, evaluant alhora la seua potencialitat en aquest tipus d'aplicacions enfront de llenguatges compilats com Arduino en C/C++ que són els habitualment usats.

Per a la comunicació amb terceres aplicacions a través de comunicacions inalàmbriques IP s'ha usat el format JSON i el protocol OSC, tots dos àmpliament emprats en aquest tipus d'aplicacions.

Finalment, el sistema s'ha validat amb els programari SoundCool i TouchOSC per a la interacció amb aplicacions de generació i processament d'àudio i video.

Abstract

This project addresses the development of a low-cost electronic system that enables real-time human-machine interaction for audiovisual applications. This system will capture and process information from various sensors and send the information in different formats wirelessly to third-party software applications used in the audiovisual world. Hence, the user will be able to interact with these software using his/her own movements and gestures.

An ESP32 has been used as a microcontroller system for its low cost, high computing power, and wireless communications capability by Buetooth and WiFi. Various interaction-focused sensors such as accelerometers, gyroscopes, magnetometers, ultrasonic emitters and receptors, flexible resistors, and others have been employed.

The Python programming language has been chosen in its reduced version for embedded systems MicroPython, evaluating its potential in such applications, instead of the usual Arduino in C/C++ compiled languajes.

JSON and OSC formats, widely used in this type of application, have been used for communication with third-party applications over wireless IP communications.

Finally, the system has been validated with the SoundCool and TouchOSC softwares for interacting with audio and video generation and processing applications.

Índice

Capítulo 1. Objetivos

El objetivo de este trabajo es el desarrollo de un sistema de interacción hombre-máquina en tiempo real para aplicaciones audiovisuales que sea de bajo coste y siga la filosofía "háztelo tú mismo" DIY en inglés de *Do It Yourself*.

Para poder interactuar con las aplicaciones audiovisuales se hará uso de distintos sensores: un emisor y receptor de ultrasonidos (HC-SR04), un acelerómetro de seis ejes (MPU6886) y una resistencia flexible (Flex Sensor 2.2). Los datos generados por cada sensor serán recogidos y procesados mediante el microprocesador ESP32 incorporado en el hardware M5Stack Fire. El ESP32 dispone de capacidad de conexión WiFi y Bluetooth de forma que se utilizará la primera para establecer las conexiones oportunas para el funcionamiento del proyecto. Para la comunicación con terceras aplicaciones se empleará el formato de datos JSON y el protocolo OSC, ampliamente usado en este tipo de aplicaciones.

Se creará un servidor web en el ESP32 para establecer los parámetros de funcionamiento y poder configurar el proyecto de forma cómoda a través de cualquier dispositivo con un navegador web.

Una vez configurado todo, para comprobar el funcionamiento se enviarán los datos mediante UDP a un ordenador que representará gráficamente los valores recibidos y pueda hacer de almacén para la grabación de los datos de todos los sensores.

Finalmente se validará la interacción del sistema haciendo uso del software SoundCool, de creación musical, sonora y audiovisual colaborativo, y del software TouchOSC mandando los mensajes OSC oportunos a partir de los datos procesados de los sensores.

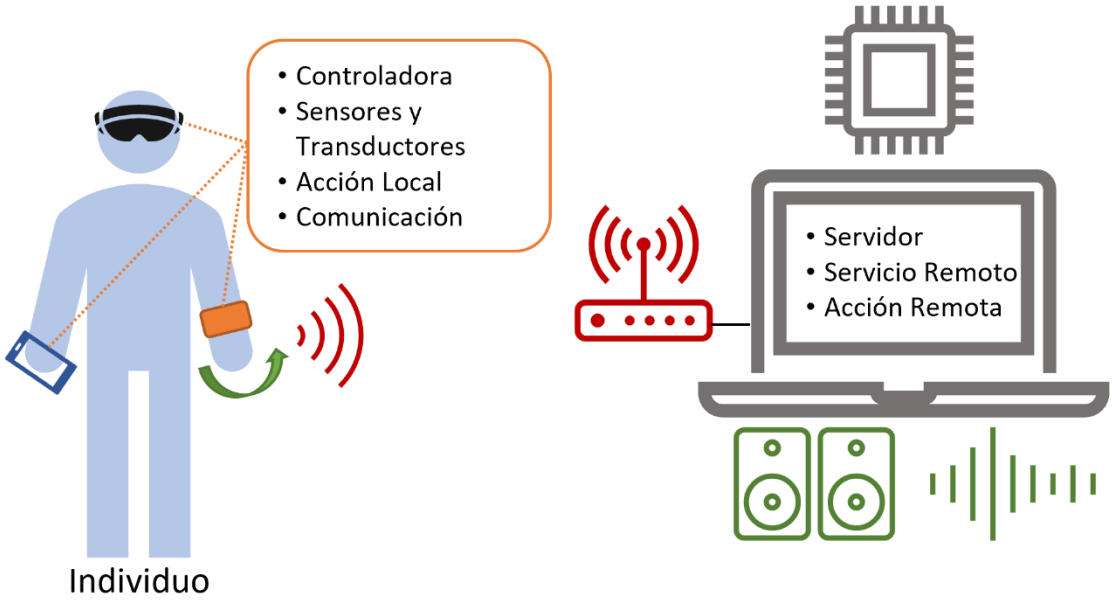

Figura 1. Esquema sistema de interacción.

Capítulo 2. Aspectos técnicos Hardware

En este capítulo se explicarán los elementos empleados para el desarrollo del trabajo.

2.1 M5stack

Para el desarrollo de una aplicación de interacción hombre-máquina existen múltiples sistemas o módulos de desarrollo que pueden utilizar distintos lenguajes de programación. En este caso se ha elegido emplear el módulo M5Stack Fire [1] de la empresa M5Stack [2] fundada en 2017 por Jimmy Lai. Esta empresa está asociada con Espressif [3], empresa desarrolladora del SoC ESP32 [4], de forma que sus productos utilizan el microcontrolador ESP32 como núcleo. Estos dispositivos gozan de gran popularidad dado su bajo coste y gran capacidad de cálculo y comunicaciones inalámbricas, empleados sobre todo en aplicaciones IoT (Internet of Things).

M5STACK

¡Error! No se encuentra el origen de la referencia.
Figura 2. Logo M5Stack.

Figura 3. M5Stack Fire.

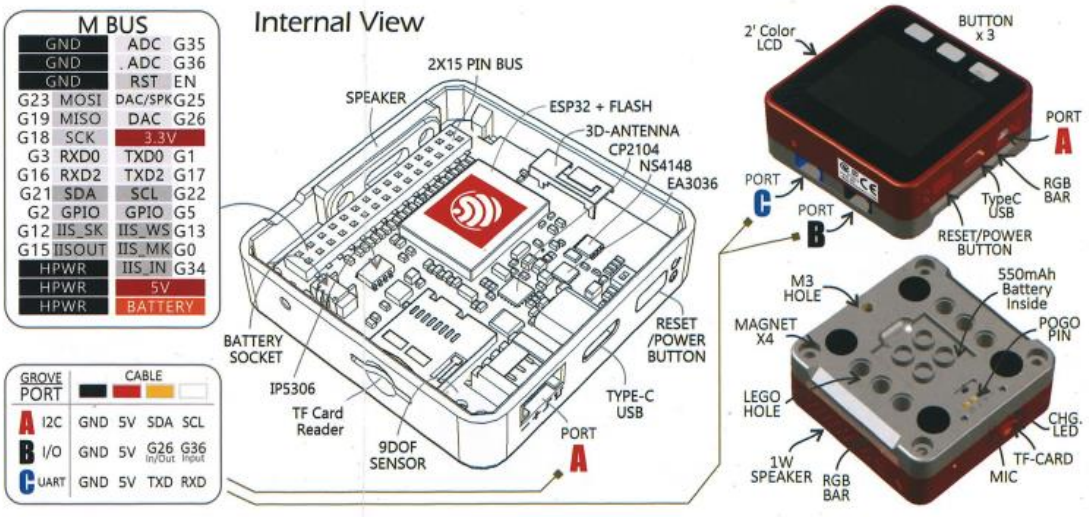

Figura 4. Esquema M5Stack Fire.

El M5Stack Fire en un sistema de desarrollo muy compacto y versátil que tiene las siguientes características:

- ESP32 de doble core de punto flotante a 240 MHz
- Memoria Flash 16MB
- PSRAM 8MB
- Puertos
  - Tipo C
  - GROVE (I2C+I/O+UART)
- Altavoz
- 3 botones
- LCD en color de 320x240 píxeles
- Sensores MEMS
  - BMM150 - Magnetómetro
  - MPU6886 – Unidad inercial con acelerómetro y giróscopo
- Batería 500 mA @ 3.7 V
- Antena 3D 2.4 GHz
- Tamaño: 54mm x 54mm x 30.5mm

2.1.1 ESP32

El ESP32 es un microcontrolador desarrollado por la empresa Espressif y lanzado en 2016, sucesor del ESP8266 [5] de 2014. El ESP32 en un SoC (System on Chip) diseñado para aplicaciones IoT, dispositivos móviles o dispositivos electrónicos portátiles. A diferencia de su antecesor, el ESP32 tiene versiones con doble núcleo y un procesador más potente, también dispone de más pines GPIO y más disponibilidad de buses de comunicación, como son I2C, UART, CAN-BUS, ISP o I2S. También dispone de mayor capacidad de memoria, mayor potencia, cifrado por hardware, diferentes sensores, mejor gestión de la energía y bluetooth.

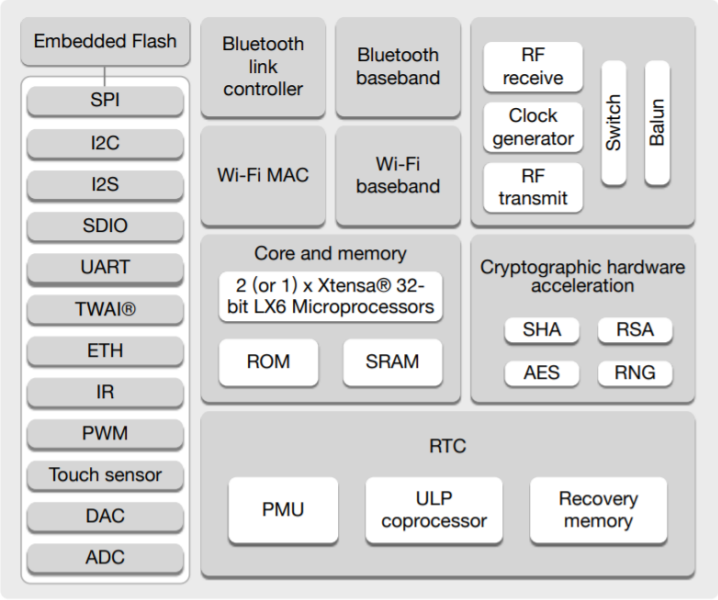

Figura 5. Diagrama de bloques funcional del módulo ESP32.

A pesar de las mejores características, el ESP8266 sigue siendo ampliamente utilizado debido a su menor coste y a que lleva más tiempo en el mercado.

El módulo M5Stack Fire incorpora un ESP32-D0WDQ6-V3, una versión del ESP32 con dos núcleos que trabaja a 240 MHz, 600 DMIPS, con 520 kb de SRAM, Wi-Fi dual mode y Bluetooth.

2.1.2 MPU6886 6 ejes

El primer sensor que se ha empleado es el MPU6886 [6] incorporado en el M5Stack. Se trata de un dispositivo de rastreo de movimiento de 6 ejes que combina un giroscopio de 3 ejes y un acelerómetro de 3 ejes. Incluye un ADC de 16 bits, filtros digitales programables, un sensor de temperatura embebido e interrupciones programables. Este sensor permite la conexión mediante I2C [7] o SPI.

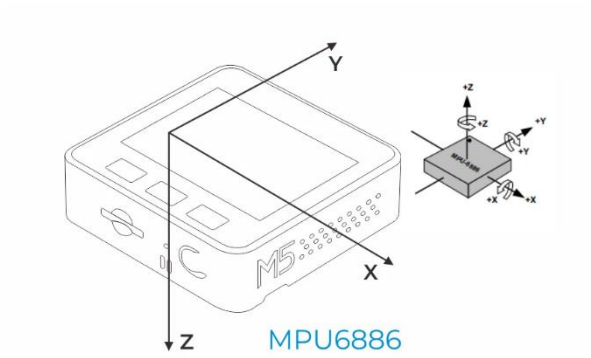

Figura 6. Esquema del MPU6886.

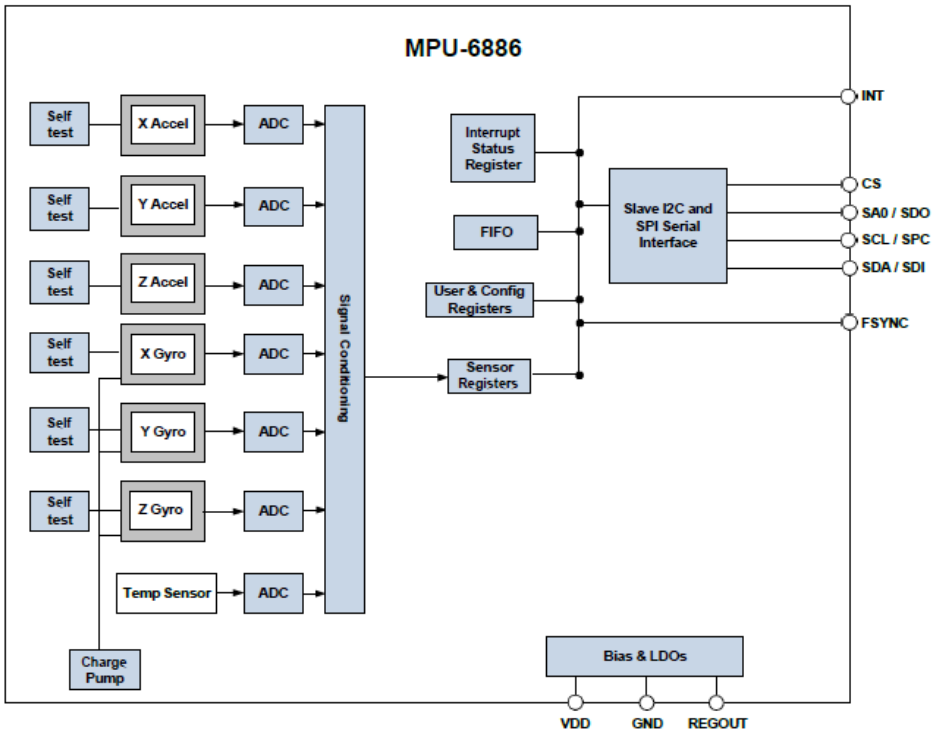

Figura 7. Diagrama de bloques del MPU6886

El MPU6886 está conectado mediante el bus de dos hilos I2C (Inter Integrated Circuit) a los pines GPIO22 y GPIO21 del ESP32, que corresponden a los pines con las líneas SCL y SDA respectivamente, y a los pines de alimentación de 5V y GND. La dirección para la conexión I2C del módulo indicada es la 0x68.

2.2 Bus de datos I2C

I2C (Inter-Integrated Circuit) es un bus de datos serie desarrollado por Phillips Semiconductors [8] en 1982. Está diseñado como un bus maestro-esclavo, el maestro inicia la comunicación y el esclavo responde. Es posible disponer de varios maestros, pero solo uno de ellos actúa como maestro y el otro como esclavo alternándose el control según un sistema de arbitraje.

El bus I2C utiliza únicamente dos líneas de señal: SCL (Serial Clock) y SDA (Serial Data) que necesitan de resistencias de pull-up externas para fijar el valor 1 lógico. La señal SCL es generada únicamente por el maestro, y permite sincronizar todos los dispositivos del bus, y la señal SDA puede ser generada por el maestro o por el esclavo para mandar datos. El bus I2C también emplea dos líneas de conexión: VDD y GND.

Existen dos modos de comunicación: Maestro-Transmisor y Esclavo-Receptor o Maestro-Receptor y Esclavo-Transmisor. En este caso se emplea el modo Maestro-Receptor y Esclavo-Transmisor ya que queremos recibir los datos capturados por el sensor.

2.3 Sensor de ultrasonidos HC-SR04

Como el M5Stack no incorpora ningún sensor de distancia, se ha utilizado el sensor de ultrasonidos externo HC-SR04 [9]. Este sensor manda pulsos a 40 kHz y recibe su eco, pudiendo obtenerse la distancia recorrida mediante la fórmula 2.1 ya que el propio módulo indica con una señal a nivel alto el tiempo que transcurre hasta la recepción del eco.

$$\boldsymbol{distancia} = \frac{\boldsymbol{tiempo\ a\ nivel\ alto\ x\ velocidad\ del\ sonido}(\mathbf{340}\ \boldsymbol{m/s})}{\mathbf{2}} \qquad \textbf{(2.1)}$$

Este módulo es capaz de medir distancias desde 2cm hasta 400cm con una precisión de hasta 3mm, con un ángulo de medida de 15 grados. El módulo incluye el transmisor de ultrasonidos, el receptor y el circuito de control, siendo muy sencillo su uso, ya que sólo hay que medir el tiempo a nivel alto de la señal.

Figura 8. Sensor HC-SR04.

El HC-SR04 dispone de 4 pines:

- Vcc: alimentación a 5 voltios.
- Trig: entrada de pulsos de disparo.
- Echo: salida de pulsos de eco.
- GND: tierra 0V.

Para utilizar el sensor se manda una señal a nivel alto durante, al menos, 10µs en la entrada de disparo. El módulo manda ocho pulsos de 40 kHz y detecta si la señal vuelve. A partir del tiempo que dura la señal de salida a nivel alto y mediante la ecuación 2.1 se puede obtener la distancia recorrida por los pulsos.

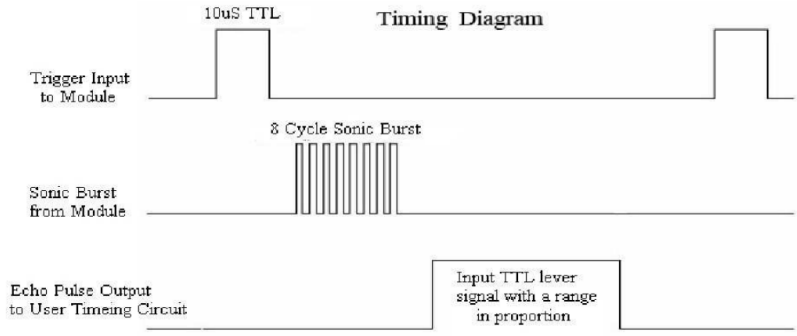

Figura 9. Diagrama temporal comportamiento HC-SR04.

Cabe destacar, que, al tener una tensión de alimentación de 5 voltios, la señal de salida tendrá un valor máximo cercano a 5 voltios. Por este motivo, para evitar dañar el M5Stack, es necesario implementar un divisor resistivo para poder obtener una señal máxima de salida de 3.3 voltios.

2.4 Flex sensor 2.2

El flex sensor 2.2 [10], es una resistencia flexible cuyo valor de impedancia cambia cuando se flexiona. Los sensores de flexión están fabricados con tinta de carbón, un material conductor cuyas partículas de la tinta se separan cuando la resistencia se dobla incrementando el valor de la resistencia. Al doblarla en el sentido contrario las partículas de tinta se juntan disminuyendo la resistencia.

Figura 10. Sensor Flex 2.2.

Características del sensor flex sensor 2.2:

- Resistencia plana: 25 kΩ ± 30%
- Resistencia doblada 180º: valor mínimo dos veces el valor de resistencia sin doblar.
- Potencia máxima: 0.5 W DC, 1 W pico

En la hoja de datos se indica el circuito propuesto para su uso. También se indica que se puede prescindir del amplificador operacional implementando un simple divisor de tensión entre el sensor y una resistencia.

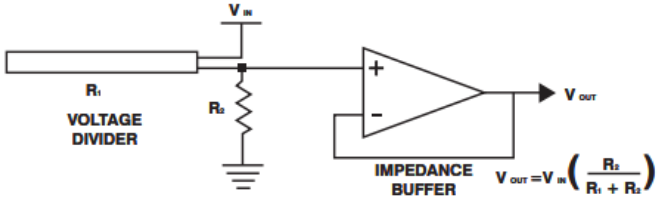

Figura 11. Circuito propuesto para utilizar el sensor flexible.

Capítulo 3. Aspectos técnicos Software

Los sistemas M5Stack permiten utilizar distintas plataformas de desarrollo que son: UIFlow [11], Arduino [12] o Micropython [13].

3.1 Plataformas de diseño de M5Stack

3.1.1 UIFlow

UIFlow es una plataforma de programación desarrollada por M5Stack y lanzada en septiembre de 2018. Permite programar de forma visual mediante bloques o a través de Python [14] como se puede ver en el botón central superior de la Figura 12.

Dispone de una versión web y también de una versión de escritorio.

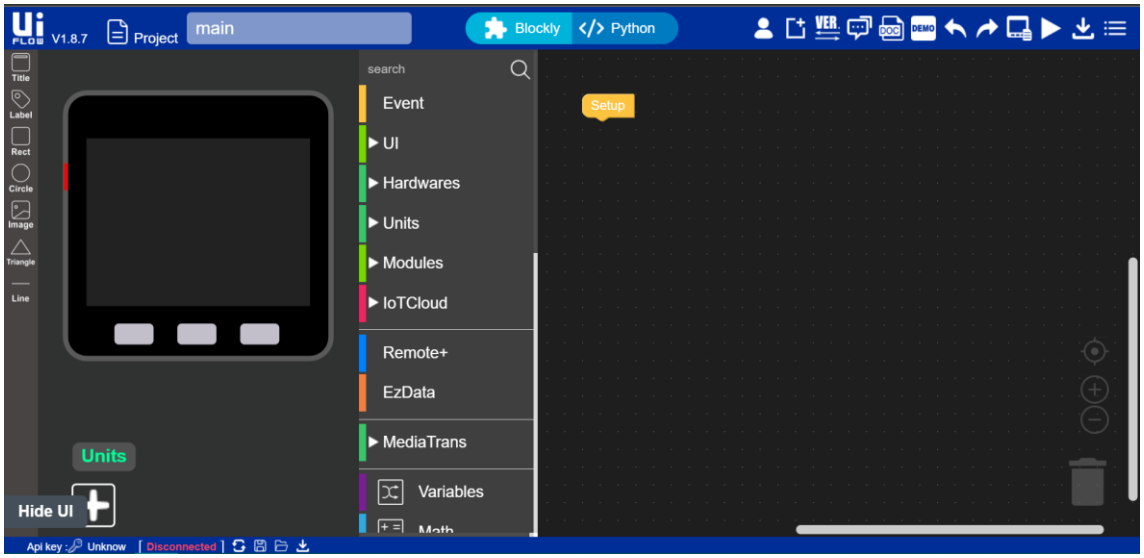

Figura 12. IDLE UiFlow versión web.

Figura 13. Conexión del sistema M5Stack con UiFlow.

Para conectar el dispositivo y poder programarlo se debe instalar el firmware recomendado en la página web de M5Stack, para cada dispositivo. Esto permite conectar el dispositivo a la red y generar una Api Key para permitir la conexión.

3.1.2 Arduino

Arduino es una plataforma de código abierto que dispone de hardware y software fácil de utilizar para desarrollo de proyectos electrónicos. Incluye soporte para distintas placas de desarrollo de bajo coste y multitud de módulos con diferentes sensores. El lenguaje utilizado para programar en Arduino es C++, con una capa de abstracción mayor que incluye librerías propias de Arduino de más alto nivel para simplificar la programación con respecto a C++ directo, acercando su uso a usuarios con poca o nula experiencia previa en programación.

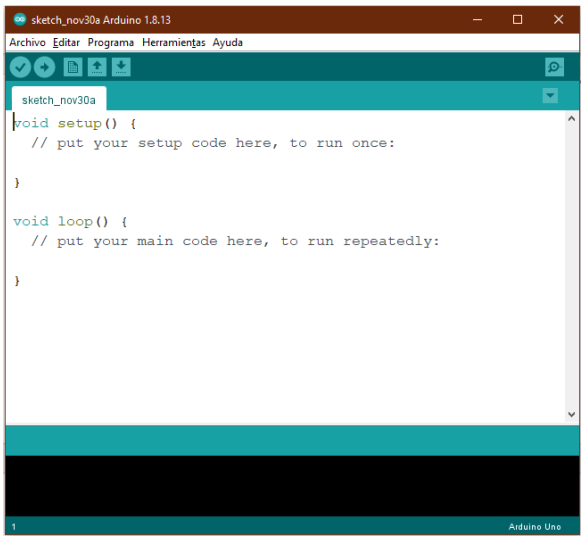

Figura 14. Entorno de desarrollo Arduino.

Existe la posibilidad de utilizar el entorno de desarrollo de Arduino para trabajar con los módulos M5Stack añadiendo en el gestor de placas de desarrollo la url *https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json.*

3.1.3 Micropython

Micropython es una implementación de Python 3 reducida especialmente desarrollada para microcontroladores o entornos con restricciones de cómputo y de cantidad de memoria. Incluye muchas de las librerías estándar de Python con capacidades limitadas, reduciendo así el tamaño del firmware necesario.

Micropython está enfocado para ser ejecutado en entornos *Bare-Metal* sin el soporte de un sistema operativo, y con centenares de KBytes de RAM a lo sumo.

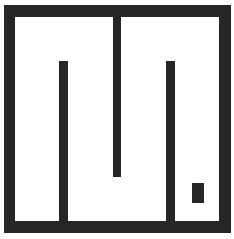

Figura 15. Logo Micropython.

Surge como una iniciativa personal del físico australiano Damien George con la idea ejecutar en microcontroladores *scripts* en entornos *Bare-metal* para conseguir desarrollos más rápidos y eficientes, y tener una curva de aprendizaje más corta que C/C++ que permitiera acercar a más gente el desarrollo de proyectos electrónicos y perder el miedo a la programación. El proyecto fue publicado en KickStarter en 2013 con gran éxito.

Micropython es un proyecto de código abierto bajo licencia MIT, y del que se dispone en GitHub todo el código necesario para compilarlo y adaptarlo a múltiples plataformas y microcontroladores, denominados *ports*.

En la página web de M5Stack se recomienda utilizar el *port* de Micropython desarrollado por ellos que incluye el UIFlow, y utilizar Visual Studio Code junto a una extensión también desarrollada por M5Stack que permite acceder al dispositivo de forma sencilla.

El firmware de UIFlow dispone de multitud de funciones de las cuales no se va a hacer uso, de forma que finalmente se empleó el firmware disponible en el repositorio de GitHub m5stack/UIFlow-Code [15]. Este repositorio dispone de la mayor parte de las bibliotecas utilizadas en UIFlow, pero el firmware ocupa menor espacio ya que no incluye el menú de arranque y otras funciones que no eran necesarias en este caso.

3.2 Firmware utilizado

Los proyectos electrónicos llevan años desarrollándose en C/C++ en plataformas como Arduino o Eclipse. En 2014 apareció Micropython como lenguaje alternativo de desarrollo basado en Python 3, que ha ido evolucionando hasta la versión actual V1.17. Siendo éste un lenguaje bastante nuevo, todavía no es muy utilizado a nivel profesional. Su poco uso está actualmente centrado en aplicaciones IoT, con empresas como Pycom [16]. No obstante, cada vez hay más sistemas de desarrollo de bajo coste que lo van soportando y el número de foros de internet, publicaciones, cursos online, etc., sobre Micropython va creciendo de forma más que notable.

Se ha planteado realizar este proyecto en Micropython para evaluar su uso en aplicaciones interactivas audiovisuales, ya que este es un campo en el que no se está utilizando actualmente y sigue prevaleciendo el uso de Arduino. Además, actualmente Python goza de gran popularidad.

Los lenguajes compilados (como Arduino con C/C++) primero convierten el código a lenguaje máquina, que es el código nativo del procesador en cuestión que se ejecuta. Esto permite que se realicen optimizaciones destinadas a la ejecución del código en tiempo de compilación, siendo su ejecución más eficiente además de requerir, en general, menor memoria. Eso sí, ante cualquier cambio de código, hay que recompilar todo el programa y volcarlo de nuevo en la memoria flas des dispositivo, proceso costoso en tiempo.

Python es un lenguaje de programación interpretado, es decir, no realiza el paso previo de compilar antes de ejecutar, generando un *byte-code* intermedio en tiempo de ejecución, el cuál es ejecutado por el intérprete, que es el firmware que se graba en la memoria flash del dispositivo una única vez. Esto provoca que el código no pueda ser optimizado tanto para su ejecución, por lo que será más lento que el código compilado, y requerirá de más memoria RAM, pero facilita y acelera el desarrollo del código al programador. Además, al interpretar el código en tiempo real si se trabaja en distintas plataformas no es necesario generar un archivo de compilación distinto para cada plataforma, únicamente se necesita un archivo. Ante un cambio de código, únicamente hay que actualizar los ficheros de texto .py en el flash, por lo que los cambios son muy rápidos.

Otra de las grandes ventajas es que Micropython incorpora un terminal tipo REPL (Read-Execute-Print-Loop) que permite interactuar con el dispositivo con comandos en Python desde cualquier ordenador que tenga un terminal. Esto permite hacer multitud de pruebas, evaluar variables, etc.

Python tiene un muy alto grado de abstracción, no se necesita definir el tipo de las variables y este se ajusta al inicializar la variable y si se asigna a otro tipo de dato no hay ningún problema de compatibilidad. No hace uso de llaves, para agrupar los distintos bloques de código se hace uso de tabulaciones. Tampoco necesita indicar que se ha terminado una línea de código añadiendo ‘;’. Finalmente, al ser un software libre y abierto, dispone de una gran comunidad y un gran número de bibliotecas para todo tipo de necesidades. Todas estas características permiten que aprender y utilizar Python sea sencillo y rápido.

Como se ha indicado, para el desarrollo de este trabajo se ha decidido utilizado el lenguaje Micropython que, como ya se ha comentado, es un subconjunto de bibliotecas de Python con

funciones reducidas para utilizar en sistemas con memoria limitada y por tanto está indicado para microcontroladores.

La principal ventaja a la hora de utilizar Micropython es la facilidad de uso que tiene. No es necesario compilar el código lo que agiliza el proceso de desarrollo ya que las compilaciones pueden tardar en completarse incluso varios minutos. Esto permite poder programar el microcontrolador directamente desde el REPL para probar y así poder detectar errores de manera muy sencilla. También permite establecer un *pseudo* punto de ruptura mediante:

```
while True:
    pass
```

Esta orden permite parar la ejecución en un punto concreto y de esta manera ejecutando una interrupción desde el teclado, mediante Ctrl+C, se puede acceder a la memoria y comprobar los valores de las variables en ese instante de tiempo.

A la hora de programar en sistemas con memoria limitada es importante tener en cuenta el uso de la memoria importando únicamente las bibliotecas y módulos necesarios, evitando el uso de espaciados, comentarios y líneas en blanco innecesarias, e incluso, en el caso de Micropython se puede utilizar 2 espacios en vez de 4 para las indentaciones de los bloques. También existe una biblioteca llamada gc (*garbage collector*), recolector de basura, que se encarga de agrupar y recuperar la memoria ya no necesaria. Es una buena práctica utilizar el método gc.collect() de vez en cuando para recuperar toda la memoria posible.

A diferencia de Python, Micropython incluye una biblioteca exclusiva llamada **machine** que permite acceder al hardware de bajo nivel, ante la falta de sistema operativo que realice la función.

Micropython dispone de diferentes *ports* para distintos microcontroladores y aquitecturas. Estos ports incluyen definiciones de múltiples placas de desarrollo sobre las cuales se puede ejecutar. Estos códigos están disponibles en GitHub [17] de forma que se puede modificar y personalizar si no existe ninguna configuración compilada que se ajuste a las necesidades requeridas.

Como se ha comentado, se ha utilizado el firmware disponible en el repositorio de GitHub m5stack/UIFlow-Code. Aquí encontramos bibliotecas disponibles para todos los módulos del M5Stack entre las que encontramos wifiCfg, para conectar de forma sencilla, MPU6050, para adquirir los datos del MPU8668 (es compatible) y los métodos lcd, btnA, btnB y btnC de la biblioteca m5stack, que se utilizan para controlar la pantalla y los botones del módulo, entre otros.

3.3 Herramientas de desarrollo

Para el desarrollo de este trabajo se han empleado los IDEs (Integrated Develoment Environment) Visual Studio Code [18] y Thonny [19] que permiten programar utilizando el lenguaje Micropython.

3.3.1 Visual Studio Code

Visual Studio Code es un editor de código desarrollado por Microsoft ampliamente utilizado para el desarrollo de todo tipo de aplicaciones. Es compatible con multitud de lenguajes de programación y tiene la capacidad de autocompletar las funciones disponibles o las variables ya utilizadas agilizando la programación. Este editor cuenta con extensiones desarrolladas por terceros para facilitar el trabajo.

Figura 16. Logo Visual Studio Code.

En este caso se ha utilizado la extensión **vscode-m5stack-mpy** [20] que se recomienda en el repositorio de GitHub para trabajar con el M5Stack en Micropython. Esta extensión permite conectar con el puerto serie de forma sencilla y poder acceder a la memoria flash del microcontrolador. Para poder acceder al puerto serie dl M5Stack es necesario instalar el controlador **CP2104 Driver** [21].

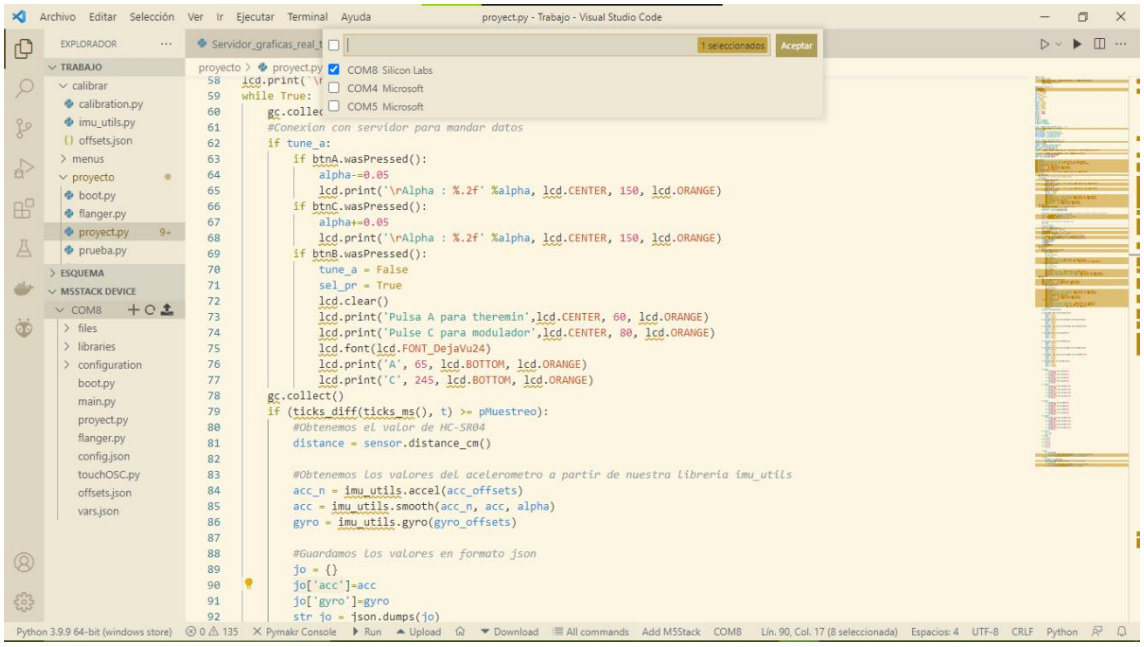

Figura 17. Entorno de desarrollo Visual Studio Code con extensión M5Stack.

La desventaja de utilizar este entorno de desarrollo es que no se dispone de REPL (Read-Eval-Print-Loop) de forma que solo se puede probar código guardándolo y ejecutándolo desde la memoria. Para poder sacar el máximo partido se intentó utilizar la extensión de **pymakr** [22] desarrollada por Pycom que añade una consola REPL al terminal para poder acceder a la placa de desarrollo, pero debido a la desincronización de la extensión, en la fecha de desarrollo de este proyecto, por una actualización de VSCode, fue imposible trabajar con ella y se decidió emplear un editor que facilitara el uso de REPL.

3.3.2 Thonny

Thonny es un IDE de Python especialmente diseñado para principiantes. Este entorno de desarrollo tiene Python 3.7 integrado de forma que no sería necesario instalar Python aparte, pero permite el uso de otras instalaciones de Python si fuera necesario.

Figura 18. Logo Thonny

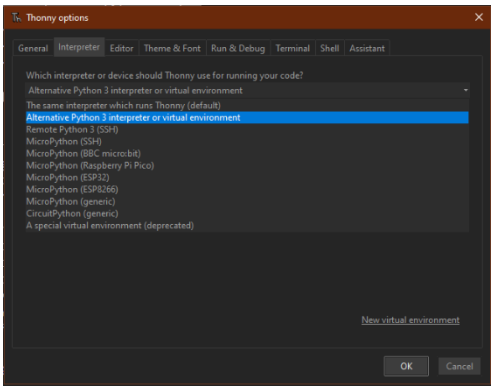

Figura 19. Selección de interprete IDE Thonny.

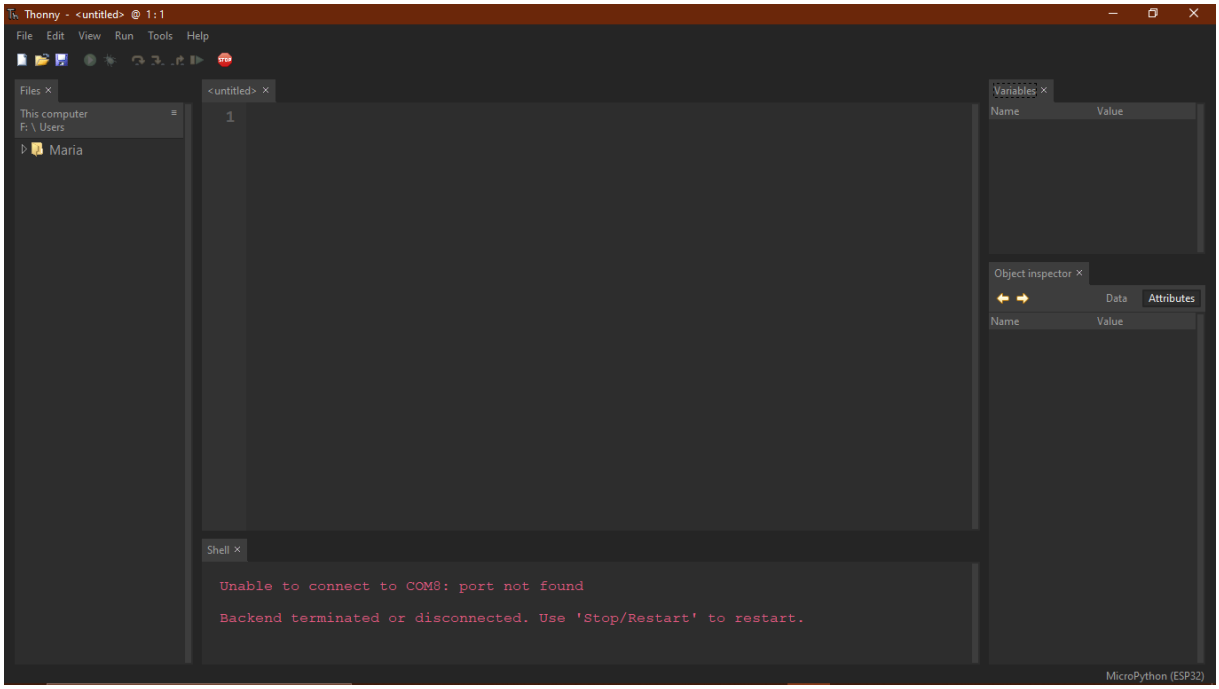

Figura 20. Entorno de desarrollo de Thonny.

Este IDE permite también el desarrollo en Micropython. Thonny dispone de un intérprete de Micropython genérico, pero también de varios intérpretes específicos como los son para ESP32, ESP8266 o Raspberry Pi Pico, como se puede ver en la Figura 19.

Además de disponer de un intérprete específico para ESP32, este IDE permite instalar o actualizar el firmware del microcontrolador con los ajustes más comunes. En otro caso, indica que haga uso de **esptool** [23], una utilidad de la línea de comandos que permite la comunicación con el bootloader de la ROM de los microcontroladores ESP8266 y ESP32, proporcionada por Espressif. Esta herramienta es necesaria para volcar el firmware de Micropython al dispositivo.

Como se puede ver en la Figura 20 se pueden añadir más opciones para facilitar la prueba del código como son las ventanas de *Variables* y *Object inspector*. En variables se puede acceder a las variables en memoria que se van generando si trabajamos desde la REPL o a las variables guardadas justo antes del punto de ruptura como se ha comentado en el apartado 3.2. Object inspector permite acceder a los atributos o datos de las variables ya que estas pueden ser módulos, funciones o variables comunes tipo int, char, float…

Por último, gracias a disponer del terminal REPL (también denominado *Shell*), otra característica que facilita la depuración es la función print() que permite mostrar en la consola aquello que

queramos en tiempo real, así como que nos dibuje gráficas en tiempo real (Plot) de los datos enviados a la consola.

Estas funcionalidades facilitan la depuración a la hora de arreglar errores permitiendo entender que sucede dentro del microcontrolador.

Para la mayor parte del desarrollo del proyecto se ha utilizado Thonny debido a las múltiples ventajas descritas.

3.4 GitHub

Otra herramienta utilizada ha sido el repositorio de software GitHub [24] que permite disponer de control de versiones del código generado. Esto facilita el desarrollo ya que, si se borra parte del código por error o se elimina algún archivo, es posible recuperarlo de versiones anteriores si se ha llevado un buen control.

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos que utiliza el sistema de versiones Git. GitHub es utilizado principalmente para el desarrollo de código, permitiendo almacenar las distintas versiones de código realizadas y colaborar entre varias personas a la vez. Los proyectos almacenados en GitHub son mayoritariamente públicos de forma que el código desarrollado se comparte y puede utilizarse por cualquier persona. Gracias a este aspecto se han podido encontrar bibliotecas publicadas en GitHub y no ha sido necesario programarlas desde cero facilitando el desarrollo del proyecto.

A la hora de utilizar GitHub existen dos repositorios, un repositorio local (o varios) que se encuentra en el ordenador o dispositivo que se esté utilizando para desarrollar el código y otro repositorio en la nube (este repositorio si es único). En el repositorio local se guardan los archivos desarrollados y posteriormente se sincronizan ambos repositorios guardando el contenido en local a la nube.

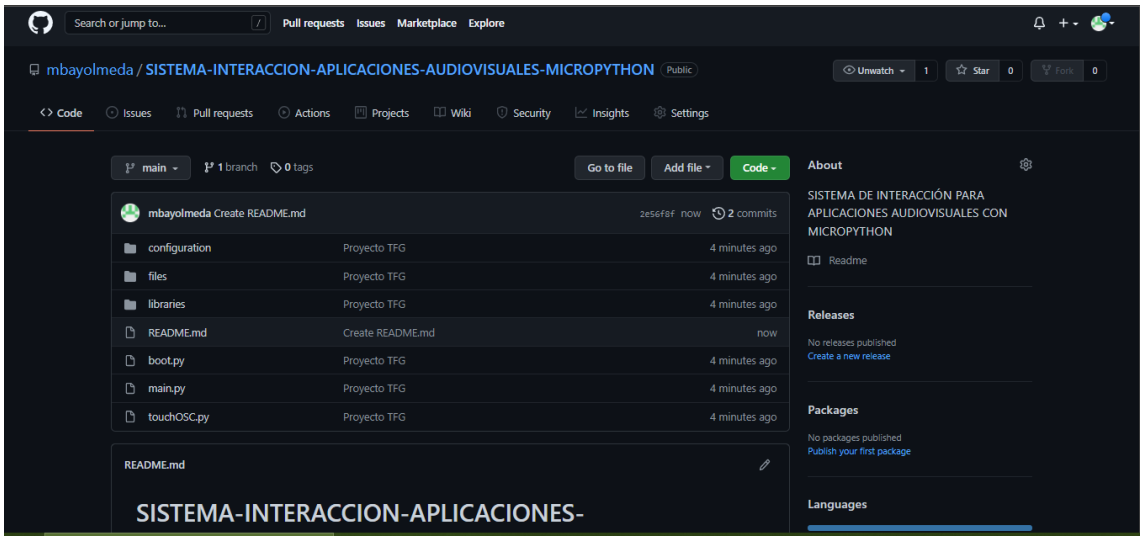

Figura 21. Repositorio GitHub del proyecto.

En primer lugar, se clona el repositorio online generado y así se crea el repositorio local mediante el comando *git clone*. Cuando se quiere actualizar el repositorio online se utilizan los comandos *git add*, *git commit* y finalmente, *git push.*

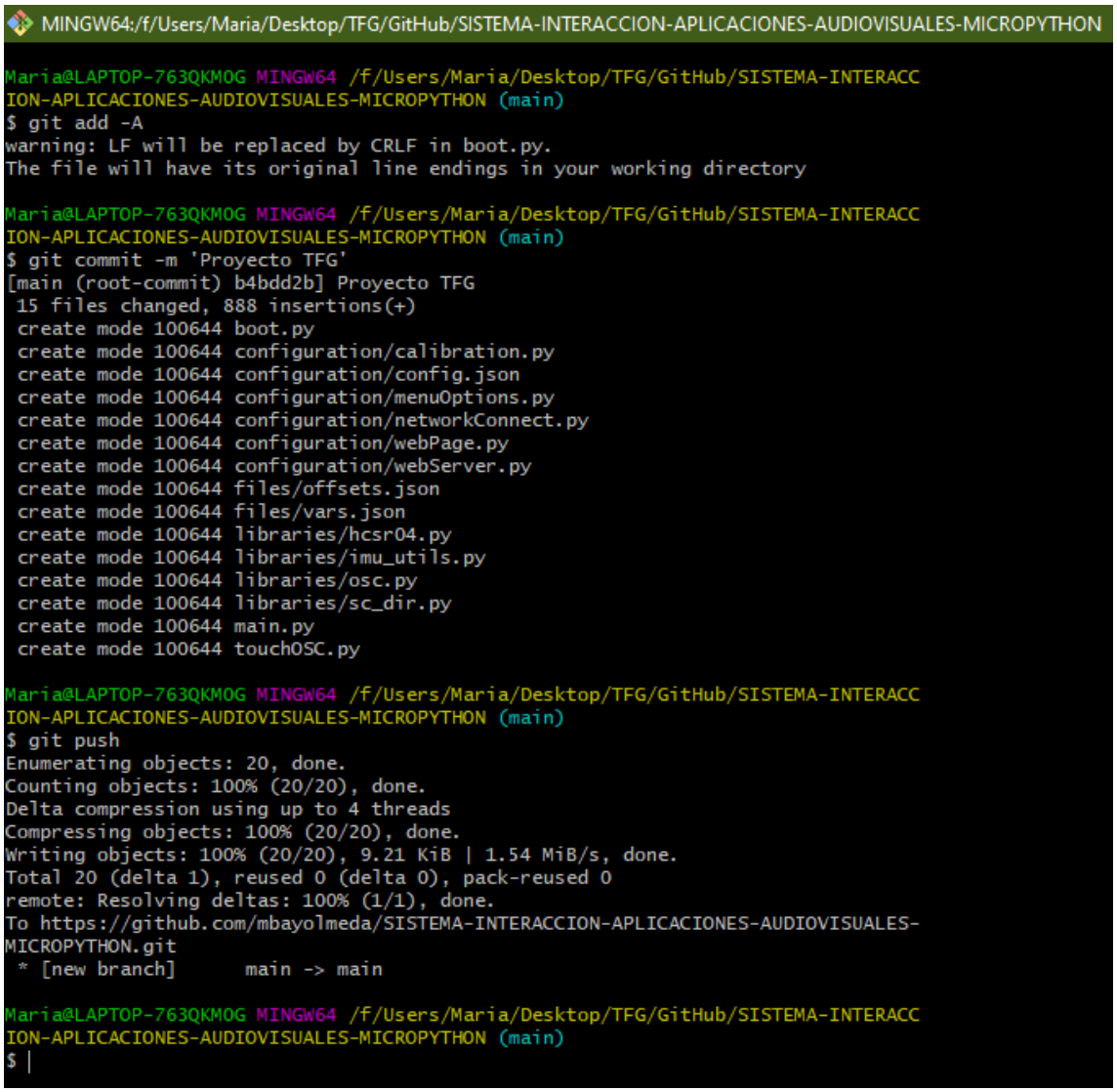

Figura 22. Sincronización repositorio GitHub con el repositorio local.

3.5 Protocolo OSC

Open Sound Control [25] es un protocolo de codificación de datos para la comunicación entre hardware y aplicaciones. Este protocolo está especialmente pensado para aplicaciones de audio, siendo una alternativa al protocolo MIDI [26] (Musical Instrument Digital Interface) pero más flexible.

MIDI es un estándar que describe un protocolo cerrado (sin capacidad de ampliarse), una interfaz digital y los conectores que permiten que varios instrumentos musicales electrónicos, ordenadores y otros dispositivos relacionados se conecten y comuniquen entre sí, publicado en 1983. Los mensajes utilizados en MIDI están descritos de forma específica, debido a esto es un protocolo cerrado. Los mensajes MIDI suelen ser de 7 bits llegando a alcanzar un máximo de 14 bits y la comunicación es unidireccional.

En 2002 se publicó la especificación 1.0 de OSC. Este estándar permite una mayor flexibilidad a la hora de estructurar los mensajes y éstos pueden ser fácilmente entendidos por las personas y no únicamente por las máquinas. OSC permite mandar datos de forma continua sin importar el medio de transmisión, pero suelen emplearse sockets UDP para el transporte de los datos.

Este protocolo está pensado para ser abierto, preciso, flexible y sencillo. Para permitir coordinar de manera precisa y simultánea la entrega y ejecución de los mensajes dispone de la estructura 'bundle', mediante la cual se pueden mandar varios mensajes a la vez. Debido a su flexibilidad, solo ha sido necesaria una revisión en 2009 para ampliar los tipos de datos soportados.

La estructura de los mensajes OSC está formada por una etiqueta de dirección de estilo URI (Uniform Resource Identifier) personalizable que permite la organización de las direcciones, el tipo de datos que se van a mandar y los datos.

Los mensajes tienen la siguiente forma: /tag_1/tag_2…/tag_n,dataType data

Las direcciones las define el diseñador del sistema facilitando el posterior direccionamiento de los mensajes de forma jerárquica, las etiquetas de las direcciones se separan mediante '/'.

Los tipos de datos se separan de las etiquetas mediante el separador ','. Los tipos de datos disponibles son los siguientes:

Etiqueta OSC	Tipo de argumento
i	Int32
f	Float32
s	Cadena ASCII terminada en NULL
b	Blob

Tabla 1. Tipos de datos estándar versión 1.0.

También es posible utilizar argumentos no estándar de los cuales algunos se añadieron al estándar en la revisión 1.1.

Etiqueta OSC	Tipo de argumento
T	True: sin bits en los datos
F	False: sin bits en los datos
N	Null: sin bits en los datos
I	Impulse: activador de eventos
t	Timetag: etiqueta de tiempo en formato NTP

Tabla 2. Tipos de datos estándar añadidos en la versión 1.1.

Etiqueta OSC	Tipo de argumento
h	int64
d	float64
c	carácter ASCII mandado como 32 bits
r	color RGBA de 32 bits
m	mensaje MIDI de 4 bytes
[]	array

Tabla 3. Tipos de datos no estándar.

El mensaje final, llamado paquete OSC, debe tener un número de bytes múltiplo de 4, teniendo en cuenta que los datos deben ser múltiplo de 4 por sí mismos, la dirección y las etiquetas de tipos

de datos también deben ser múltiplos de 4. Para cumplir esto, los tipos de datos cuya longitud no sea múltiplo de 4 se rellenará con 0s y la etiqueta del mensaje también se rellenará con 0s si es oportuno.

Por ejemplo, el mensaje `/oscillator/4/frequency 440.0` para cambiar un valor de un oscilador en un dispositivo, se codificará como:

```
2f (/)  6f (o)  73 (s)  63 (c)
69 (i)  6c (l)  6c (l)  61 (a)
74 (t)  6f (o)  72 (r)  2f (/)
34 (4)  2f (/)  66 (f)  72 (r)
65 (e)  71 (q)  75 (u)  65 (e)
6e (n)  63 (c)  79 (y)  0 ()
2c (,)  66 (f)  0 ()    0 ()
43 (C)  dc (Ü)  0 ()    0 ()
```

Figura 23. Codificación mensajes OSC.

Se puede observar en verde la dirección `/oscillator/4/frequency`, en azul el *type tag* ,f, en rojo el valor del float32 `0x43dc0000` que corresponde a 440.0 en *big-endian*, y en naranja los ceros 0 añadidos a los campos para hacerlos de tamaño múltiplo de 4 bytes.

3.6 Aplicación SoundCool

SoundCool [27] es un sistema colaborativo para la creación sonora, musical y visual por medio de móviles, tabletas, Kinect y Hololens. Mediante los distintos dispositivos mencionados es posible controlar distintos módulos musicales o visuales. Este sistema fue desarrollado en la Universidad Politécnica de Valencia dirigido por Jorge Sastre.

Figura 24. Logo de SoundCool.

Esta aplicación dispone de una versión para ordenador desde donde se configura el proyecto con los módulos a utilizar y distintas versiones para las diferentes plataformas disponibles desde las que, conectándose a la dirección IP del ordenador del proyecto, se pueden controlar los distintos módulos.

Figura 25. Pantalla principal aplicación SoundCool PC.

En la figura anterior se puede ver el aspecto de la aplicación para ordenador y todos los módulos disponibles para trabajar. En el lado derecho se encuentran los módulos de vídeo y en el izquierdo los de audio. Una vez se eligen y se configuran los módulos, introduciendo la dirección IP del ordenador y el puerto asignado a algún módulo se puede conectar desde, en este caso como ejemplo, un dispositivo móvil para controlar dicho módulo.

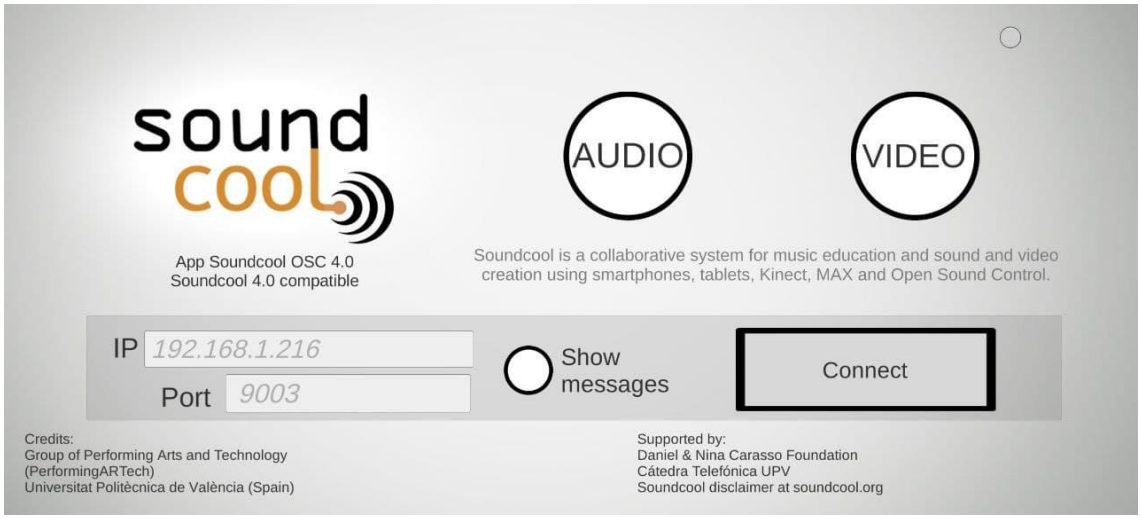

Figura 26. Pantalla inicial SoundCool en dispositivos móviles.

Figura 27. Módulo player aplicación PC.

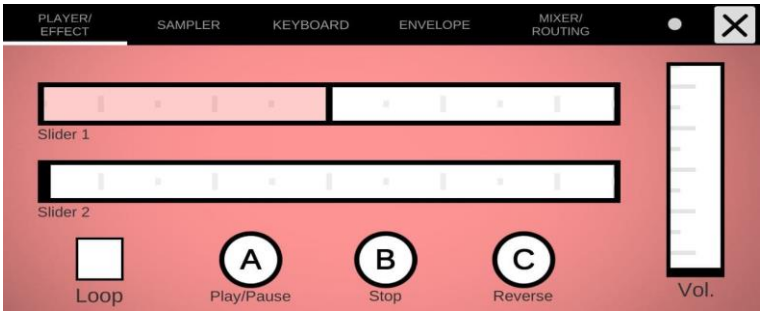

Figura 28. Player aplicación móvil.

En la Figura 27 se puede ver el aspecto del módulo PLAYER en la versión de ordenador y en la Figura 28 la versión móvil. No es posible desde el dispositivo móvil realizar configuraciones, como en este caso seleccionar la pista de audio a utilizar, pero, si se pueden controlar los botones y los faders que permiten ajustar los parámetros, de reproducción en este caso.

Como se puede ver en la Figura 28, para la parte de audio, en la aplicación hay cinco menús disponibles. De esta forma, según el tipo de módulo se utilizará uno u otro para controlarlo.

Para la comunicación entre el servidor alojado en el ordenador y los clientes, como el móvil del ejemplo, se utiliza el protocolo OSC. En la Figura 26 se puede ver que existe la opción *Show messages*. Esta opción muestra el mensaje OSC que se manda al interactuar con cada uno de los controles disponibles en la aplicación. Conociendo las direcciones de los controles y el tipo de dato a mandar, se puede replicar y mandar mensajes desde cualquier dispositivo mediante conexión a internet y codificando los mensajes de manera adecuada para que cumplan con el protocolo OSC.

3.7 Aplicación TouchOSC

TouchOSC [28] es una aplicación disponible para sistemas Windows, macOS, Linux, iOS y Android. Esta aplicación permite mandar y recibir mensajes de distintas conexiones al mismo tiempo bien sean mensajes MIDI u OSC. Tiene soporte para conexiones OSC sobre UDP y sobre TCP y también conexiones MIDI en cable o Wireless.

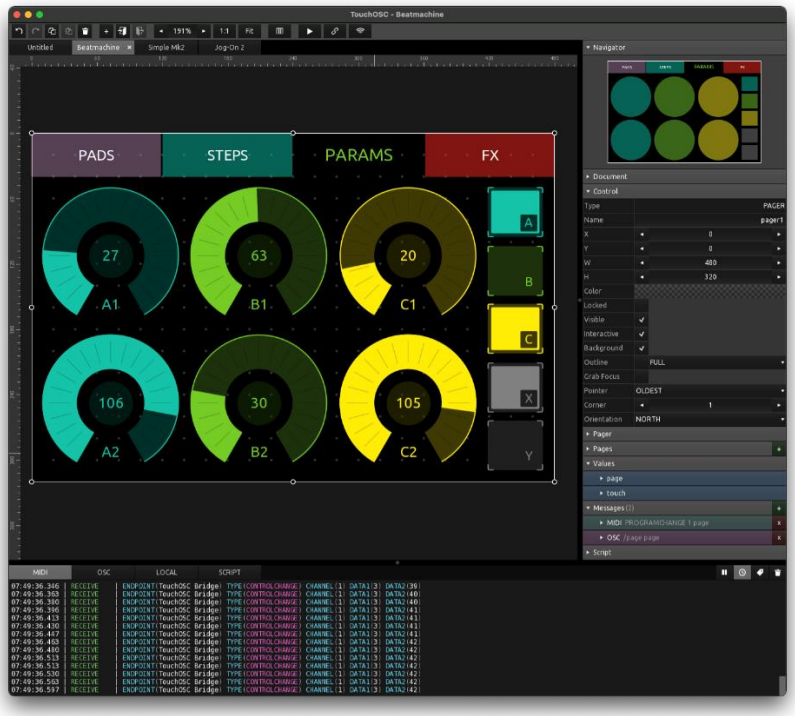

Figura 29. Ejemplo sistema TouchOSC.

TouchOSC dispone de distintos tipos de controles de colores y contenedores para controlar los valores de los mensajes a enviar o representar los valores de los mensajes a recibir. Permite montar sistemas de interacción compatibles con multitud de aplicaciones como son: Ableton Live/ Max for Live, Native Instruments Tracktor, REAPER, Reason, Cycling ’74 Max… entre otros.

Capítulo 4. Desarrollo y resultados

Para facilitar el uso de los archivos desarrollados, estos se han dividido en subcarpetas dentro del sistema de archivos de la memoria flash del M5Stack Fire que incorpora Micropython. De esta manera en vez de encontrar más de diez archivos juntos, se pueden encontrar divididos en código de configuración, archivos de datos y bibliotecas, quedando únicamente boot.py, main.py y touchOSC.py sin clasificar.

Micropython ejecuta de forma automática el fichero boot.py, seguido del main.py desde el que debemos dar acceso a nuestro código.

- /flash
  - /configuration
    - networkConnection.py
    - menuOptions.py
    - calibration.py
  - /files
    - offsets.json
    - vars.json
  - /libraries
    - hcsr04.py
    - imu_utils.py
    - osc.py
  - boot.py
  - main.py
  - touchOSC.py

4.1 Lectura de los sensores

4.1.1 Lectura MPU6886

Para obtener los datos capturados por el MPU6886, se ha utilizado la biblioteca mpu6050 [29], que es compatible con nuestro sensor, disponible en el firmware instalado. Esta biblioteca simplifica en gran medida el trabajo, de forma que únicamente hace falta ejecutar tres funciones para obtener los datos de aceleración, giro e ypr (*Yaw*, *Roll*, *Pitch*).

Para conseguir una mejor aproximación de los resultados obtenidos del MPU6886, se ha diseñado imu_utils.py, que incluye funciones que permiten calcular valores de offset para las señales a utilizar y posteriormente combinarlos con los valores obtenidos de la señal para obtener un mejor resultado final, así como un filtro paso bajo para poder eliminar parte del ruido de la señal de aceleración.

En primer lugar, se calibra el sensor. Esta calibración consiste en calcular los valores obtenidos en reposo. En este caso para calcular los offsets se sitúa el M5Stack en una superficie plana con la pantalla hacia arriba de forma que la aceleración en el eje x debería ser 0, en el eje y debería ser 0 y en el eje z, debido a la actuación de la aceleración gravitatoria y dado que los valores de aceleración se miden en g, el valor debería ser 1. En el caso del giróscopo, al estar quieto y no haber rotación todos los valores que devuelva deberían ser 0. Para calcular dichos offsets se ha programado la función accel_gyro_offsets(), descrita a continuación, en el archivo imu_utils.py.

```
def accel_gyro_offsets(n):
    zero_off = [0.0,0.0,0.0]
    off_acc = [0.0,0.0,0.0]
    off_gyro = [0.0,0.0,0.0]
    for i in range(n):
        a_xyz = accel(zero_off)
        g_xyz = gyro(zero_off)
        for j in range(3):
            off_acc[j] += 1/n*a_xyz[j]
            off_gyro[j] += 1/n*g_xyz[j]
        off_acc[2] -=1.0 #Componente gravedad
        return off_acc, off_gyro
```

Para poder disponer de los offsets calculados y poder utilizarlos al realizar las medidas, los valores obtenidos se guardan en un archivo en formato JSON llamado offsets.json.

En imu_utils.py también se encuentra la función smooth() que implementa el filtro paso bajo de primer orden que elimina el ruido y también los pequeños movimientos erróneos que pudieran interferir en el tratamiento de los datos.

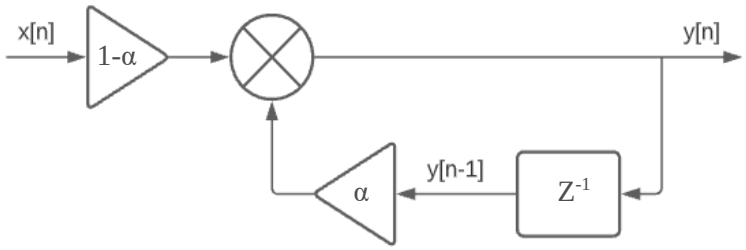

Figura 30. Diagrama filtro paso bajo de primer orden.

```
def smooth(new, prev, alpha):
    sm = [0.0]*len(new)
    for i in range(len(new)):
        sm[i] = new[i]*(1-alpha) + prev[i]*alpha
        prev[i] = sm[i]
    return sm
```

La lectura de este sensor se realiza en el archivo main.py cada pMuestreo milisengundos, valor que se ha fijado a 50 ms. Se ha utilizado la función ticks_ms() y ticks_diff() de la biblioteca time para poder esperar intervalos tiempo sin parar la ejecución del código como sucedería al utilizar sleep_ms().

4.1.2 Lectura HC-SR04

Para utilizar el sensor HC-SR04, se ha hecho uso de la biblioteca hcsr04 disponible en el repositorio de github micropython-hcsr04 [30]. En esta biblioteca se aplican los conceptos explicados en el apartado 2.3 de forma que únicamente es necesario definir los pines a utilizar y la distancia máxima que alcanzará el sensor, que se asigna a la variable dist mediante el servidor web implementado que se explicará en el apartado 4.4.

```
sensor = hcsr04.HCSR04(trigger_pin=26, echo_pin=36, echo_timeout_us=30*2*dist)
distance = sensor.distance_cm()
```

Como se ha comentado en el apartado 2.3, para evitar dañar el M5Stack, se debe conectar el sensor utilizando un divisor de tensión que reduzca el valor máximo de salida de 5V a 3.3V. Para ello se han empleado 3 resistencias de 39 kΩ.

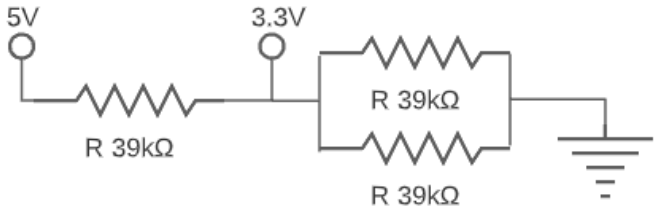

Figura 31. Esquema divisor resistivo.

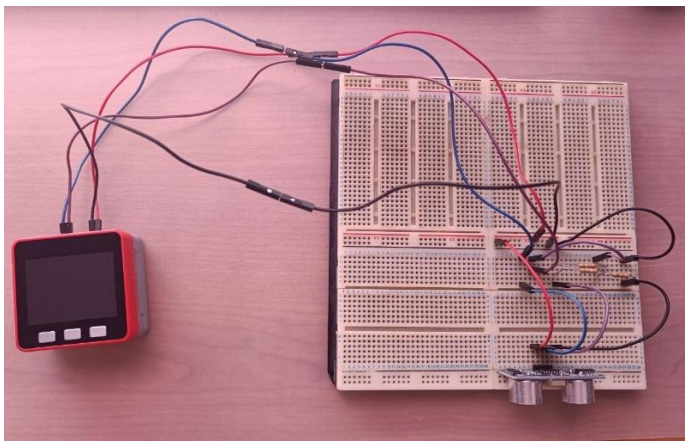

Figura 32. Conexión HC-SR04.

En la Figura 32 se puede ver el divisor resistivo mencionado anteriormente y la conexión del módulo. Los cables negro y rojo corresponden a GND y 5V respectivamente, y los cables azul y morado corresponden al trigger y al echo respectivamente conectados a los pines 26 y 36 de módulo disponibles desde el puerto B.

4.1.3 Lectura de la resistencia Flex Sensor 2.2

Para obtener el valor de tensión referente a las variaciones del Flex Sensor 2.2 se ha utilizado la clase ADC de la librería machine. Esta clase permite, a partir de un pin con características ADC, obtener el valor de lectura del pin de manera muy sencilla.

```
adc = machine.ADC(36)
adc.atten(3)
adc.width(3)
V1 = adc.read()
```

En este caso se ha utilizado el pin 36, que es el único pin con las características necesarias accesible desde fuera del M5Stack. Las funciones adc.atten() y adc.width() permiten ajustar la atenuación de entrada para fijar la tensión máxima de entrada y el número de bits utilizados para el valor medido. En este caso se han elegido 11 dB de atenuación, permitiendo llegar a 3.6 V de entrada como máximo y 12 bits para poder representar el valor medido.

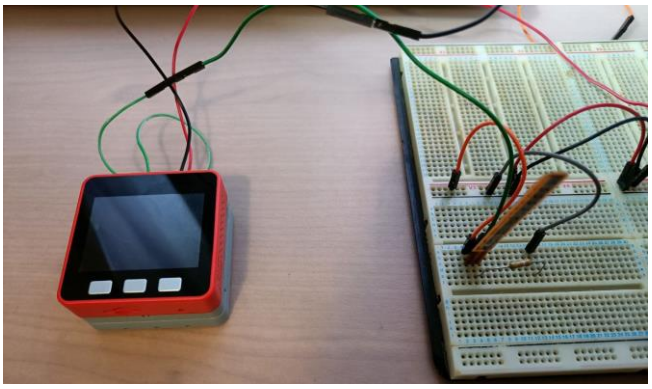

Figura 33. Montaje resistencia flexible.

Para obtener el valor de la resistencia flexible se realiza un divisor de tensión como se ha explicado en el apartado 2.4, en este caso se ha utilizado una impedancia de 39 kΩ para el divisor. En la imagen podemos ver como el cable naranja corresponde a 5V, el cable gris a GND y el cable verde se conecta desde el divisor de tensión al pin 36 del ESP32 incorporado en la M5Stack.

4.2 Servidor web M5Stack

Como se ha comentado anteriormente, es necesario fijar ciertos parámetros para el correcto funcionamiento del sistema en función de las condiciones. Para poder modificar estos parámetros sin tener que entrar al código y cambiarlo manualmente se ha diseñado un servidor web utilizando sockets TCP que permite ajustar los valores y guardarlos en la memoria para su posterior uso.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
```

Para indicar que un socket es TCP se utiliza socket.SOCK_STREAM, si fuera UDP se utilizaria socket.SOCK_DGRAM. Como se va a generar una página web, se utiliza el puerto 80 ya que este es el que utilizan los navegadores para el protocolo HTTP. A continuación, se bloquea la ejecución del código a la espera de la conexión de un cliente. Para poder conectarse se muestra en la pantalla la dirección IP en la que está alojado el servidor creado. Mediante cualquier navegador se puede acceder a dicha dirección.

Figura 34.

Para diseñar el servidor web se ha utilizado parte del código descrito en el tutorial ESP32/ESP8266 Micropython Web Server – Control Outputs [31] y se ha modificado para ajustarlo a las necesidades de este proyecto.

En el archivo webPage.py se encuentra la descripción HTML de la página web y en el archivo webServer.py el código Python que se ejecuta.

En este servidor se fija la distancia máxima que podrá calcular el sensor HC-RS04 y se determina un valor orientativo de alpha, la variable del filtro paso bajo del acelerómetro, que podrá modificarse posteriormente si se encuentra necesario.

Al conectarse al servidor aparece en la pantalla del cliente la imagen de la Figura 35. Mediante los botones se modifican los valores y cuando son los deseados, pulsando el botón submit se manda la información de vuelta al M5Stack para su procesamiento.

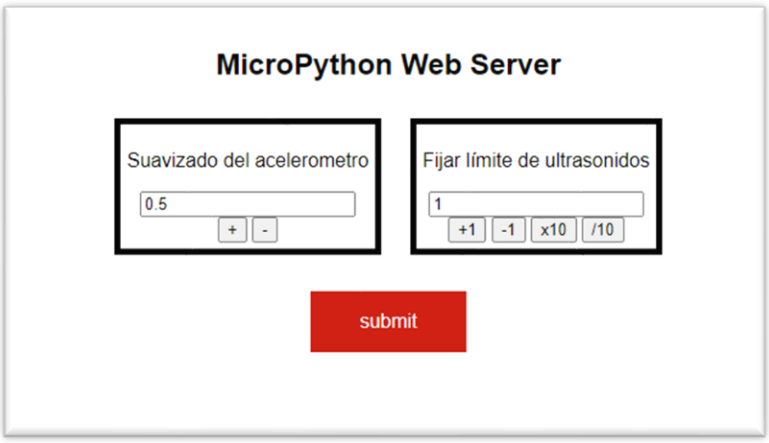

Figura 35. Servidor web.

Como se ha visto, la conexión del servidor TCP se realiza mediante un socket, de modo que es necesario conectar el sistema M5Stack a la red. Para ello se hace uso del módulo de Wifi del

Figura 36. Meú (elegir red wifi).

Figura 37. Submenú (red de casa).

M5Stack Fire. En el firmware instalado se encuentra la biblioteca wifiCfg, esta biblioteca simplifica el uso de la biblioteca network de Python de forma que con una única instrucción podemos realizar la conexión.

```
from lib import wifiCfg
wifiCfg.doConnect(ssid, pswd, lcdShow=True)
```

Para no tener que añadir la red y la contraseña de la red a la que conectarse, en el archivo networkConnect.py se han guardado las tres redes que se han utilizado durante el desarrollo del proyecto y mediante los botones del M5Stack se puede elegir a cuál de las redes conectarse.

4.3 Servidor Python

Una vez se ha realizado la conexión WiFi, se han calibrado los sensores y se han definido los parámetros de diseño, se obtienen los datos del MPU6886 y se mandan mediante sockets UDP, en formato JSON, a un servidor ejecutado en un ordenador que permite representar gráficamente los datos recibidos, así como almacenarlos para un uso futuro.

4.3.1 Servidor con sockets

Para recibir los datos y posteriormente poder representarlos de forma gráfica, se ha decidido utilizar sockets UDP porque permiten el intercambio de datos en tiempo real. La primera versión se realizó en TCP, pero no se conseguía ver un cambio inmediato en las gráficas al realizar movimientos con el acelerómetro.

Python dispone de la biblioteca select [32] que incluye funciones para monitorizar entradas/salidas de datos. Esta biblioteca incluye la función select.poll() que permite registrar o dejar de registrar archivos para posteriormente monitorizarlos para eventos de lectura/escritura. En el API de Python se indica que en máquinas con sistema operativo Windows solo se pueden utilizar las funciones disponibles para monitorizar sockets, pero en este caso la función poll() no se encontraba disponible, así que se ha utilizado la biblioteca selectors [33] como alternativa.

La biblioteca selectors está construida sobre select. Se recomienda utilizar esta biblioteca ya que es de más alto nivel y por tanto más sencilla de utilizar para el usuario final. En primer lugar, se crea un objeto selector. Mediante la opción default, se genera el selector más eficiente posible según la plataforma que se esté utilizando.

```
mysel = selectors.DefaultSelector()
```

El siguiente paso es definir la función que se ejecutará cuando se detecte el evento, y el objeto que monitorizar. Una vez definidos, se registra el objeto para su monitorización mediante la siguiente mysel.register(). En este caso se está monitorizando el socket s, para eventos de lectura y se ejecuta la función read(). Para monitorizar eventos de lectura debería cambiarse selectors.EVENT_READ por selectors.EVENT_WRITE.

```
def read(sock, mask):
    global data
    data = sock.recv(1024)
    return data

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(server_address)
mysel.register(s, selectors.EVENT_READ, read)
```

Este registro permite evitar que la escucha del socket sea bloqueante para aplicaciones que lo necesiten. También cierra el socket correctamente si se produce algún problema, de forma que evita errores. En el este caso, se requería un socket bloqueante para evitar que se ejecutara el código antes de que llegaran los datos que procesar. Para ello se ha añadido el siguiente código:

```
for key, mask in mysel.select():
       callback = key.data
       callback(key.fileobj, mask)
```

Con este bloque se permite que el código esté a la espera del evento, es decir que sea bloqueante, y si se prescinde de este bloque el servidor será no bloqueante permitiendo la ejecución simultánea mientras espera un evento.

4.3.2 Gráficas en tiempo real no bloqueantes

Para dibujar gráficas, Python dispone de una gran biblioteca llamada matplotlib [34] con multitud de funcionalidades. En este caso se ha utilizado matplotlib.pyplot, esta es una interfaz de matplotlib que permite realizar gráficos de forma similar a Matlab [35]. También se ha utilizado la biblioteca numpy [36] que permite vectorizar datos y trabajar con ellos de manera sencilla, y también dispone de multitud de operaciones y funciones matemáticas, que no se han utilizado en este caso.

Para gráficas estáticas existen multitud de posibilidades y es muy sencillo de implementar. Para gráficas con desplazamiento en tiempo real, existe más dificultad a la hora de generarlas porque muchos de los métodos disponibles son bloqueantes por lo que únicamente se puede dedicar el código a la gráfica. En este caso se necesita generar la gráfica a partir de los datos recibidos por un socket por lo que la gráfica no puede bloquear la ejecución del código porque si no podrían perderse datos a recibir.

Para actualizar la gráfica de forma no bloqueante se han utilizado el siguiente código inspirado en el código de la página "Cómo trazar datos en tiempo real usando Matplotlib" de DelftStack.com [37].

```
import matplotlib.pyplot as plt
plt.ion()
figure, ax = plt.subplots(1,2,figsize=(12,6))
line1, = ax[0].plot(x, y_accX, color='r', label = 'x', linewidth=0.8)
```

En primer lugar, se activa el modo interactivo mediante plt.ion(), esto permite utilizar canvas.draw() y canvas.flush_elements(). A continuación, se genera la figura del tamaño que se quiera y con los subplots deseados, en este caso 2 subplots en la misma línea. Se genera una gráfica añadiendo los vectores x e y deseados y se puede personalizar cambiando los colores o el ancho. Para la explicación solo se menciona una línea para facilitar la comprensión, pero se han utilizado seis líneas, tres para cada subplot.

```
line1.set_xdata(x)
line1.set_ydata(y_accX)
```

Después de actualizar los valores de x y de y_accX, en este caso, se actualiza los valores a representar mediante set_xdata() y set_ydata().

```
figure.canvas.draw()
figure.canvas.flush_events()
```

Finalmente, para actualizar la gráfica mostrada se utiliza canvas.draw() que redibuja la figura actual y canvas.flush_elements() aguanta el evento de la GUI hasta que se procesan los eventos de la UI actualizando los valores continuamente.

Para representar los datos del MPU6886 se ha dividido la figura en dos sublplots. En el primero se representa la aceleración en cada uno de los ejes y en el segundo se representa el giro.

En la Figura 38 podemos observar el efecto del filtro paso bajo descrito en el apartado 4.1.1 aplicando un alpha de 0.95 al inicio y un alpha de 0.05 a continuación. Se puede comprobar de esta manera como se reduce el ruido. A la hora de utilizar los datos del sensor para la interacción, es preferible utilizar valores de alfa entre 0.5 y 0.8, de esta manera reducimos el ruido sin llegar a distorsionar o ralentizar los cambios de valores.

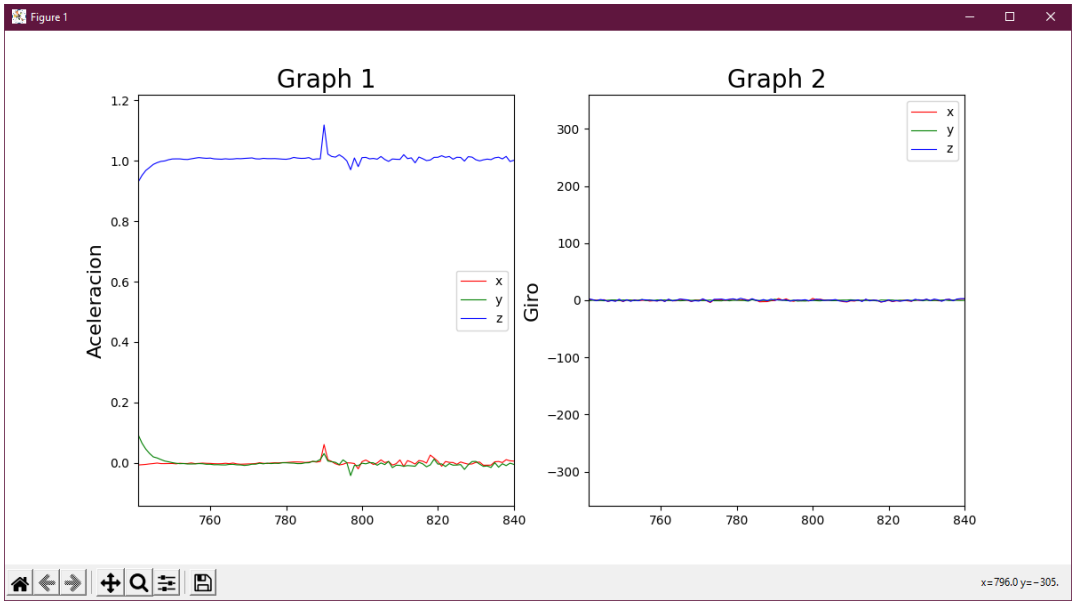

Figura 38. Diferencia de suavizado con alpha = 0.95 y alpha = 0.05**.**

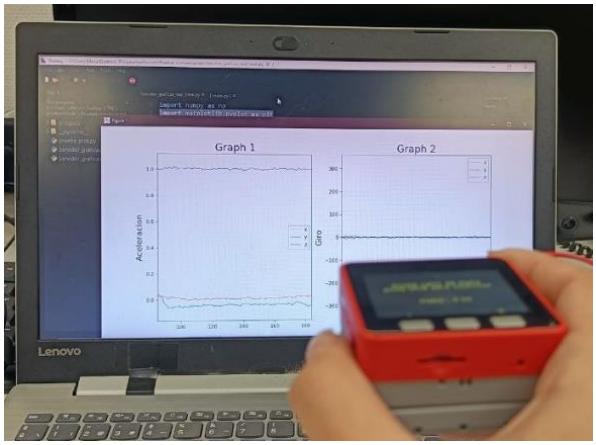

Figura 40. Gesto: quieto.

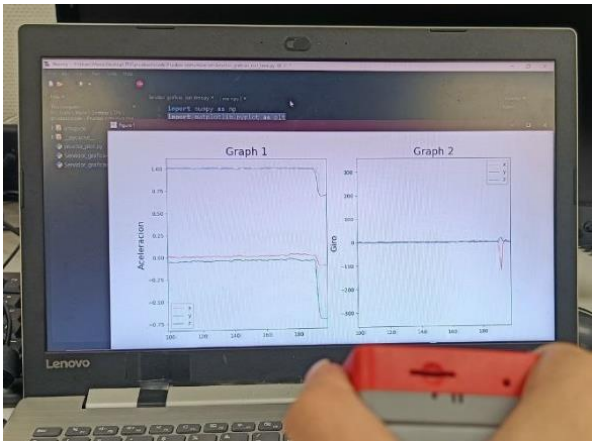

Figura 40. Gesto: inclinar hacia adelante.

En la Figura 41 se puede ver de manera más clara la respuesta del MPU6886 cuando se pasa de tener el M5Stack quieto con la pantalla hacia arriba a inclinado y finalmente a la posición inicial. Cuando el eje x tiene un valor de aproximadamente 1560, se pasa de tener el M5Stack sujeto con la pantalla hacia arriba a inclinarlo hacia adelante y cuando el eje x tiene un valor de casi 1580 se vuelve a la posición inicial.

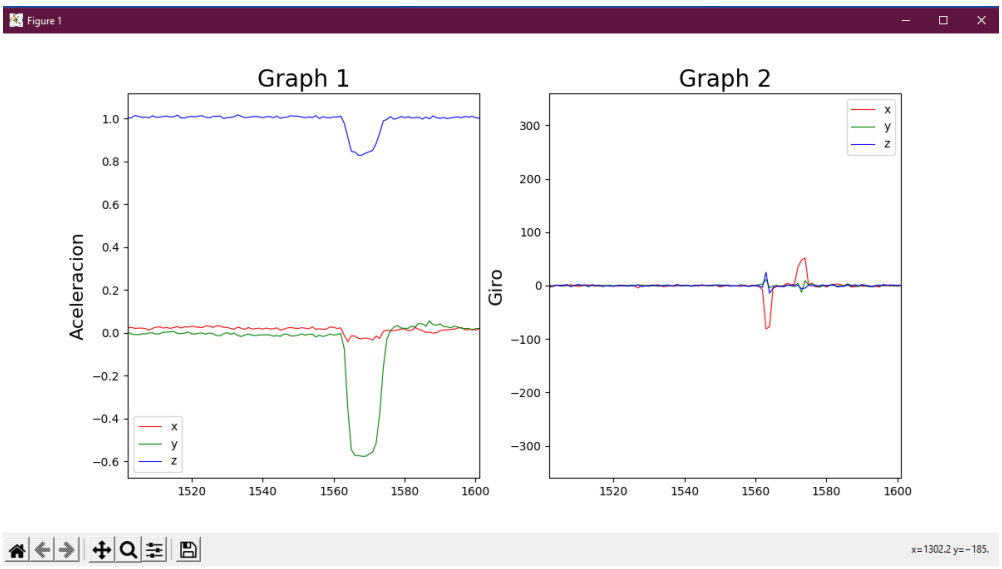

Figura 41. Cambio de gesto con el M5Stack.

4.4 Comunicación OSC

Existen multitud de aplicaciones que permiten la interacción mediante mensajes OSC. En este caso, para probar la interacción mediante los sensores se han elegido utilizar de manera sencilla las aplicaciones SoundCool y TouchOSC.

Para mandar los mensajes de acuerdo con las especificaciones del protocolo se ha diseñado la biblioteca osc.py basándose en el módulo client.py del repositorio de GitHub SpotlightKid/micropython-osc [38].

Osc.py está diseñada específicamente para mandar mensajes a la aplicación SoundCool, de modo que no incluye todas las especificaciones de OSC, solo las necesarias.

Utilizar bibliotecas para mandar mensajes ralentiza el proceso de mandar los mensajes, pero permite agilizar la programación de dichos mensajes. Debe valorarse si prima la velocidad de ejecución o la velocidad de desarrollo del código. La alternativa a utilizar bibliotecas es mandar los mensajes en bruto, lo cual permite una ejecución más ágil, pero se debe prestar especial atención al rellenado con ceros de las direcciones para cumplir con las especificaciones del protocolo.

Para los mensajes mandados a la aplicación SoundCool se ha utilizado la biblioteca osc.py y para mandar los mensajes a la aplicación TouchOSC se han escrito directamente como cadena de bytes.

4.4.1 Métodos de interacción con los datos de los sensores

Para mandar los datos a la aplicación es necesario traducir los valores obtenidos de los sensores a datos que la aplicación que se vaya a emplear pueda interpretar. En este punto se utilizarán los datos mandados a la aplicación SoundCool como ejemplo.

Los valores que se mandan a la aplicación SoundCool, en caso de ser datos de tipo float, deben estar comprendidos entre 0 y 1. Para traducir los valores de los sensores a datos interpretables por el receptor existen varios métodos.

4.4.1.1 Incremental

El método incremental consiste en modificar el dato a enviar aumentando o disminuyendo su valor en función de la respuesta del sensor ante los estímulos. Esta modificación puede seguir una regresión lineal, exponencial, logarítmica… en función de las necesidades.

Este método se ha utilizado para aumentar el valor de un fader de la aplicación SoundCool mediante el uso del acelerómetro. El método incremental se ha utilizado en conjunto con el método de gestos que se explicará posteriormente en el apartado Gestos4.4.1.3.

En este caso, no se ha aplicado ninguna ecuación, simplemente se han modificado los incrementos en función del valor de aceleración del eje de interés. Como el rango del acelerómetro es de ±1 g y los valores que admite SoundCool están comprendidos entre 0 y 1, si el valor de aceleración está entre 0.1 y 0.2 aumentamos el valor enviado a SoundCool en 0.005, si el valor de aceleración se encuentra entre 0.2 y 0.4 el valor enviado aumenta en 0.01, si el valor de aceleración se encuentra entre 0.4 y 0.6 el valor enviado aumenta en 0.015 y si el valor de aceleración se encuentra entre 0.6 y 0.8 el valor enviado aumenta en 0.02. Si el valor de aceleración es negativo, se aplican los mismos tramos de forma decremental.

4.4.1.2 Absoluto

El método absoluto, consiste en mandar un valor proporcional al medido que entre dentro del rango de valores admitidos de la aplicación o el valor medido directamente si este es admisible.

Este método se aplica a los valores obtenidos del sensor HC-SR04. El valor obtenido se pondera con el valor máximo definido para el sensor de forma que el resultado se encuentra entre 0 y 1.

4.4.1.3 Gestos

El método de gestos consiste en asociar ciertos estímulos concretos en forma de gestos y asociarlos a un valor concreto.

En este caso, se ha aplicado este método a los gestos inclinar M5Stack hacia la derecha, hacia la izquierda, hacia delante, hacia atrás y quieto. Para saber si se ha ejecutado alguno de los gestos se comprueba a la vez que se cumplen dos condiciones para cada uno de los gestos. La primera condición es un cambio brusco de valor del giroscopio en el eje correspondiente y la segunda un valor distinto de 0 en la aceleración del eje correspondiente.

Gesto	Valor giro	Eje giro	Valor aceleración	Eje aceleración
Inclinar hacia la derecha	> +20º	y	< -0.2	x
Inclinar hacia la izquierda	< -20º	y	> 0.2	x
Inclinar hacia adelante	< -20º	x	< -0.2	y
Inclinar hacia atrás	> 20º	x	> 0.2	y
Quieto	< -20º	x, y	-0.2 <acc < 0.2	x, y

Tabla 4. Definición de gestos MPU6886.

4.4.2 Aplicación SoundCool

Para probar el resultado del proyecto con la aplicación SoundCool se han preparado varias configuraciones. Los módulos utilizados para estas demostraciones son: delay, mixer, signal generator, player y speaker.

4.4.2.1 Simulación de un theremín

En la primera configuración se pretende simular un theremín, un instrumento musical electrónico compuesto por dos antenas y una caja. La antena recta controla la frecuencia y la otra el volumen. Cada antena se controla con una mano y dependiendo de la posición relativa de las manos a las antenas se cambiará la frecuencia y el volumen.

Figura 42. Theremín

Para simular este efecto, en el generador de señal se configura una onda sinusoidal cuya frecuencia se controla mediante el sensor HC-SR04 y cuyo volumen se controla mediante la inclinación hacia adelante o hacia atrás del M5Stack.

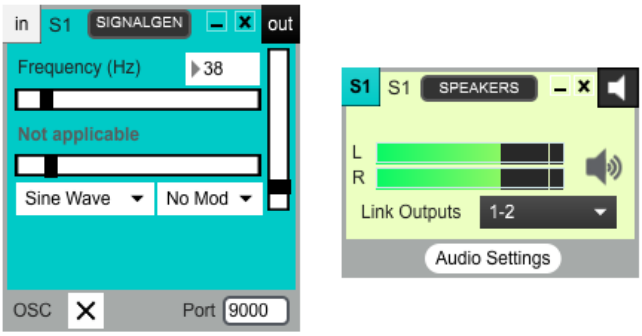

Figura 43. Configuración theremín en SoundCool.

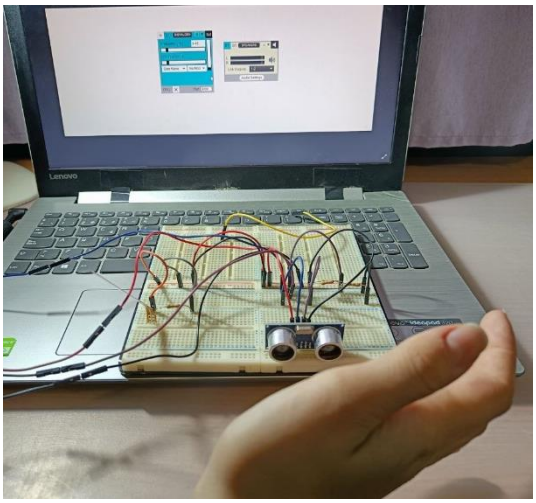

Figura 45. Variación de la frecuencia.

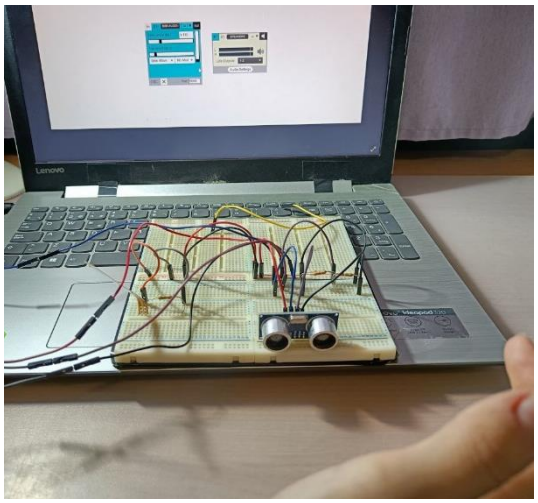

Figura 45. Variación de la frecuencia.

Como podemos ver en la Figura 45 y en la Figura 45, la barra de la frecuencia cambia al modificarse la distancia a la que se sitúa la mano del HC-SR04. A menor distancia menor frecuencia y a mayor distancia mayor frecuencia.

El problema que puede plantearse es que la superficie de la mano es bastante reducida por lo que es fácil que al moverla se salga del trayecto de los pulsos del sensor, de forma que no lograrían rebotar de forma correcta. Utilizando la mano se logran resultados decentes, pero utilizando un objeto plano con mayor superficie, en este caso se ha utilizado la tapa de una caja de cartón para la prueba, los resultados son mucho mejores pudiendo recorrerse todo el rango de distancia sin provocar saltos bruscos en la frecuencia.

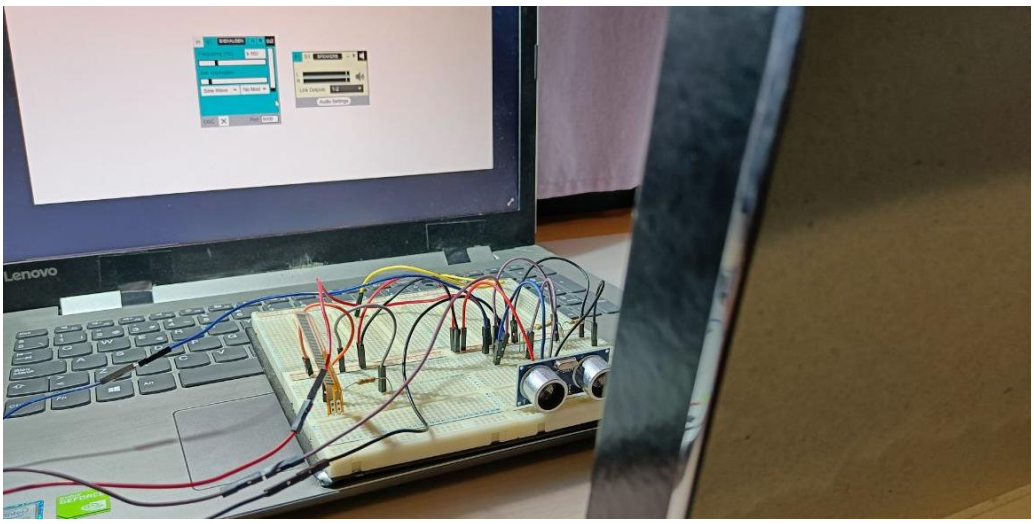

Figura 46. Utilización de objeto con mayor superficie para la interacción con el sensor.

Para que, si apartamos la mano, o el objeto que estemos utilizando para interactuar con este sensor, no se pierda el último valor adquirido, como el sensor devuelve un número negativo cuando no encuentra ningún obstáculo en el rango definido, el mensaje OSC con el valor actualizado solo se

manda si la salida del sensor es mayor que cero. Es importante porque de esta manera es posible no tener que estar manteniendo un valor deseado de distancia si se quiere utilizar un mismo valor durante un tiempo.

4.4.2.2 Modulación FM

La segunda configuración sirve para comprobar el efecto sonoro que ejerce la modulación FM sobre una señal utilizando los sensores descritos para modificar los parámetros de frecuencia e índice de modulación.

La modulación FM es un tipo de modulación angular que permite transmitir información en forma de onda mediante otra onda portadora modificando su frecuencia. Este tipo de modulación se utiliza en bandas de alta frecuencia para aplicaciones relacionadas con el audio como la transmisión de radio o la transmisión del audio de televisión, entre otras.

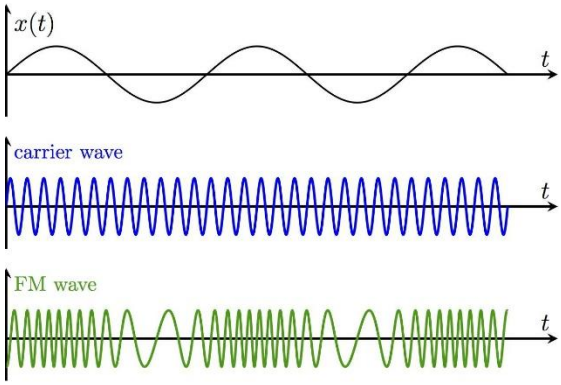

Figura 47. Modulación FM.

Para simular este efecto mediante SoundCool, el primer generador de señal se configura con una onda sinusoidal cuya frecuencia se controla mediante la inclinación hacia los lados del M5Stack, esta señal será la onda de información. En el segundo generador se configura también una onda sinusoidal, pero a ésta se le añade modulación FM de forma que esta segunda onda será la portadora. En el segundo generador la frecuencia se controla mediante la inclinación hacia adelante y hacia atrás del M5Stack y el factor de modulación se controla mediante el sensor HC-SR04. Finalmente se conecta la salida del segundo generador al altavoz de forma que al modificar los parámetros descritos se producen cambios en la onda modulada. Como la frecuencia instantánea fe la onda modulada es proporcional a la frecuencia de la onda portadora, y en este caso esta puede tomar valores que el oído humano es capaz de procesar, se apreciar el efecto de la modulación al escucharlo.

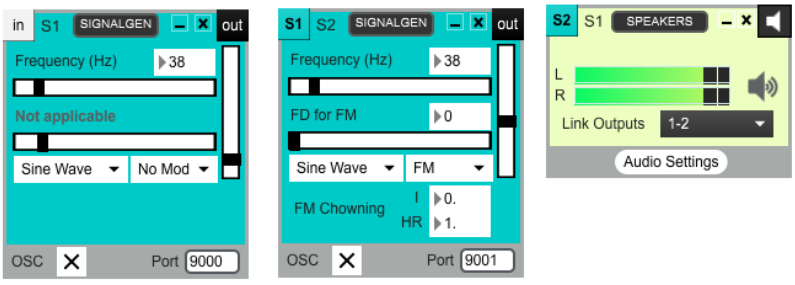

Figura 48. Configuración onda modulada en SoundCool.

4.4.2.3 Efecto Flanger

Otra configuración simula el efecto flanger, este efecto se utiliza comúnmente con guitarras y sintetizadores produciendo un característico sonido metalizado. Esto se consigue duplicando la

señal original y aplicándole un retardo menor de 10 ms, si el retardo es mayor el efecto producido ya no sería flanger y sería más parecido a un chorus. La realimentación de la señal retardada permite acentuar dicho efecto.

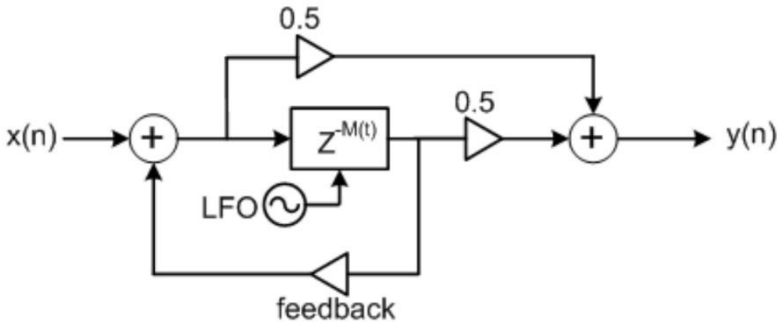

Figura 49. Esquema funcionamiento efecto flanger.

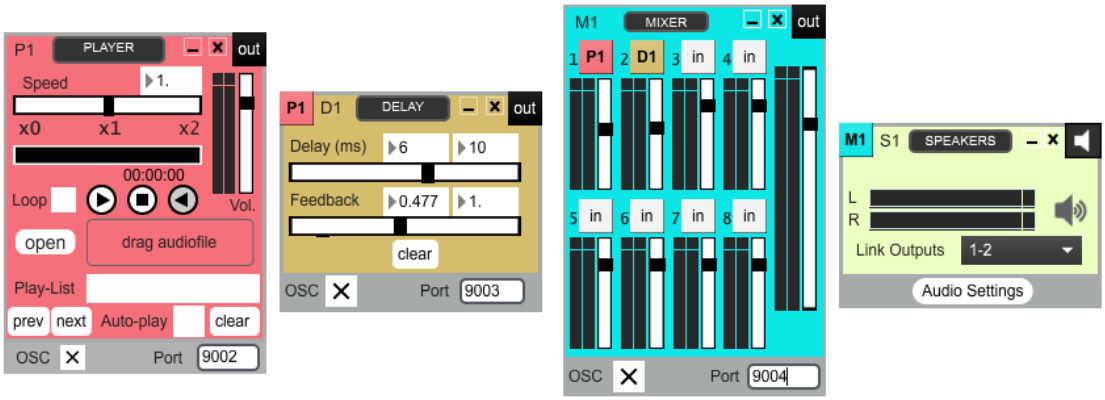

Figura 50. Configuración efecto flanger en SoundCool.

La aplicación SoundCool permite añadir distintos efectos que se pueden descargar de internet o generar mediante otras aplicaciones. En este caso, en lugar de utilizar uno de esos efectos, se ha decidido utilizar distintos módulos para simular este efecto sonoro. La desventaja que plantea implementar el efecto flanger mediante los módulos disponibles en SoundCool, es la falta de precisión del parámetro delay del módulo con el mismo nombre. En esta aplicación el incremento único del retardo es de 1 ms y no es posible cambiarlo. Esto provoca que, a la hora de utilizar el efecto, al modificar el retardo haya cambios bruscos.

Para poder controlar la respuesta de este efecto se han utilizado el MPU6886 y el Flex Sensor 2.2. No es posible utilizar el Flex sensor y el HC-SR04 simultáneamente sin abrir el hardware y acceder al bus de pines que tiene dentro. Esto se debe a que en el M5Stack Fire, solo quedan accesibles tres puertos: un puerto I2C, un puerto UART y un puerto de Entrada/Salida. Los dos sensores mencionados utilizan el pin 36 y no es posible utilizarlo a la vez. Por este motivo en ninguna de las demostraciones se han utilizado los tres sensores a la vez.

En este caso también se han utilizado los botones disponibles en el M5Stack Fire de forma que el botón central, btnB, controla el botón Play/Pause del módulo player. Los botones laterales, btnA y btnC, se han utilizado para controlar el valor de delay incrementando la variable que lo controla o decrementándola. Para controlar los niveles de las señales en el mezclador se ha utilizado la misma configuración de gestos e incrementos descrita anteriormente para el procesado de datos del MPU6886. Finalmente, el feedback del módulo delay se ha controlado mediante la resistencia flexible.

Se ha comprobado que el rango de valores entre el valor en reposo y el valor cuando se flexiona 180º, es mayor al doblar hacia atrás, tomando como cara delantera la que muestra las barras

blancas y negras. Al doblar la resistencia en este sentido su valor disminuye y al doblarla en el sentido contrario aumenta.

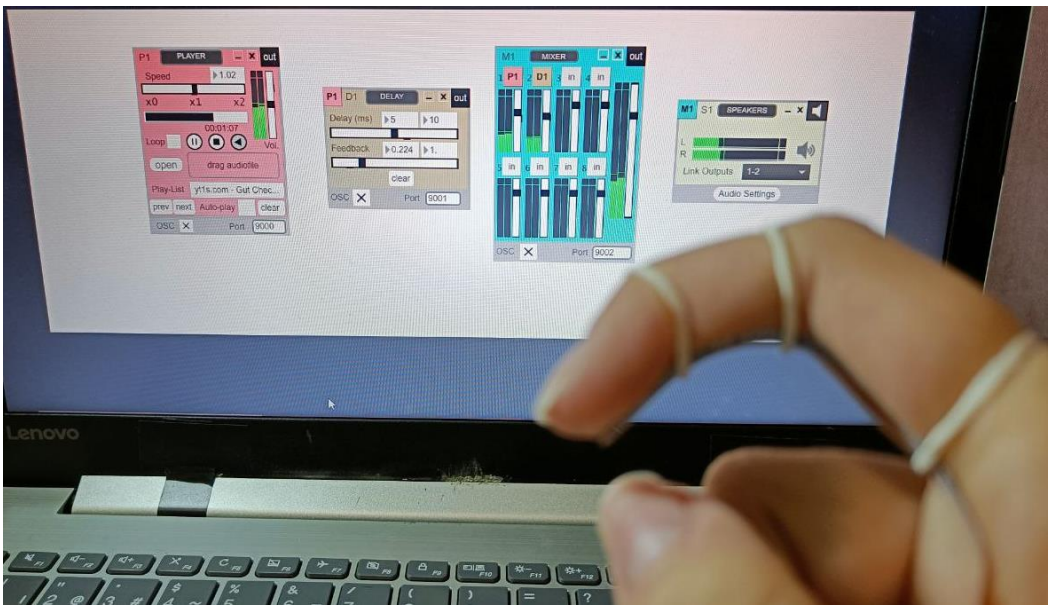

Figura 51. Control del parámetro feedback mediante el Flex Sensor 2.2.

Para facilitar el manejo de la resistencia, ésta ha sido enganchada a la parte interna dedo índice mediante gomas, de esta manera la resistencia era mucho más fácil de flexionar que al situarla en la parte externa del dedo.

A la hora de procesar los datos de la resistencia, se ha utilizado el método absoluto aplicando una proporcionalidad lineal. Como los datos obtenidos tienen cifras entre una y tres decenas de millares, estos han de proporcionarse para equivaler al rango de valores de 0 a 1. Para evitar los casos límite se ha reducido el rango de 0.05 a 0.95.

A la hora de decidir como aumentar el valor de feedback y como disminuirlo, se ha llegado a la conclusión de que la posición de reposo es mejor que equivalga a un nivel de feedback de 0 y que al doblar la resistencia este nivel aumente. De esta forma, si se estira el dedo por error, simplemente se anulará el feedback, si se tomara el caso contrario, al estirar el dedo se producirían sonidos muy molestos al realimentar la totalidad de la señal.

Las resistencias flexibles no son muy estables ya que se ven afectadas por pequeñas perturbaciones. Por este motivo se ha decidido utilizar la función smooth() de imu_utils.py para eliminar el ruido y se ha conseguido un mejor resultado.

Por último, se han obtenido los valores máximos y mínimos medidos a partir de la resistencia, que se encuentran rondando los 2300 y 1500 respectivamente, y se ha calculado la ecuación de la recta que relaciona estos valores con los valores deseados 0.05 y 0.95. El valor que se manda a la aplicación SoundCool es aquel obtenido a partir de aplicar la proporcionalidad lineal calculada al valor obtenido del sensor.

4.4.3 Aplicación TouchOSC

Para probar la interacción con los sensores con otra aplicación, se ha diseñado este pequeño sistema en TouchOSC. El elemento radial se controla mediante el valor de aceleración del eje z del MPU6886, el elemento xy se controla mediante los valores de aceleración de los ejes x e y del MPU6886 y el elemento fader se controla mediante el valor de tensión calculado a partir de la resistencia flexible.

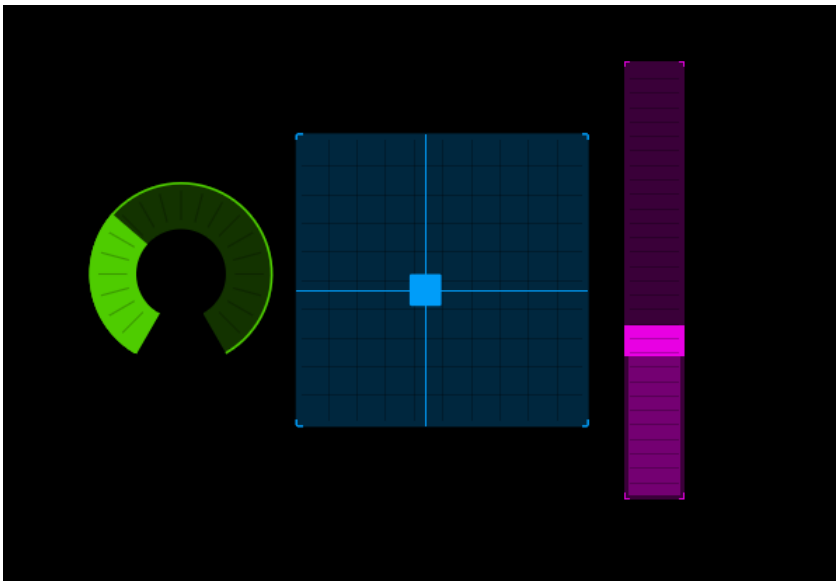

Figura 52. Sistema TouchOSC diseñado.

Para poder representar los valores obtenidos de los sensores, se mandan mensajes OSC, en este caso se han mandado los mensajes en bruto. También se ha empleado el método absoluto de interpretación de datos puesto que en esta aplicación es posible ajustar el rango de valores admitidos.

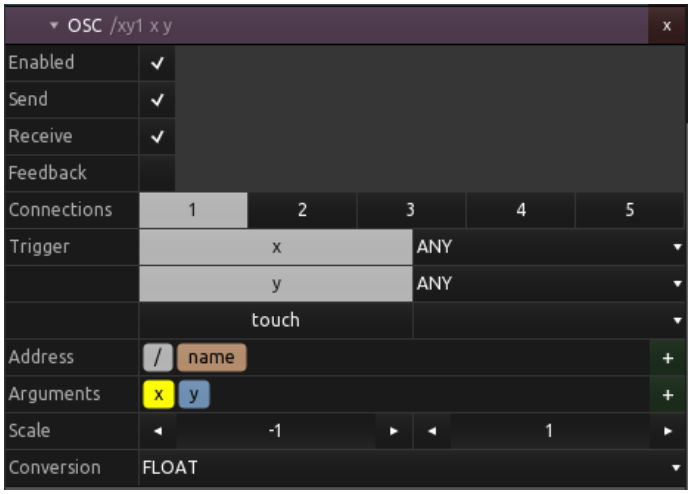

Figura 53. Menú configuración mensajes OSC del elemento xy.

En la Figura 53 se muestra la configuración del apartado OSC del elemento xy incluido en el sistema TouchOSC diseñado. Se puede comprobar que se realiza un escalado para poder representar valores más allá de 0 y 1, que son los valores por defecto. Para poder monitorizar los valores negativos se ha modificado el escalado al rango -1, 1. Para cubrir todos los valores leídos por el microprocesador a partir de la resistencia flexible también se ha escalado el rango de 0 a 2800.

Como se ha comentado en la introducción de la comunicación OSC, se han escrito los mensajes para mandar a TouchOSC de manera directa. Los mensajes utilizados son los siguientes:

```
s.sendall(b'/xy1\00\00\00,ff\00'+struct.pack('>f',acc[0])+struct.pack('>f',acc[1])
)
s.sendall(b'/radial2\00\00\00,f\00\00' + struct.pack('>f',acc[2]))
s.sendall(b'/fader9\00,f\00\00' + struct.pack('>f',V1))
```

Como se puede comprobar, se debe tener en cuenta el número de bytes de la dirección para así rellenar con ceros y cumplir con las especificaciones del protocolo.

Se ha empleado la librería estándar struct con el método pack para poder enviar los datos de los valores en float directamente en bruto (raw), byte a byte. La cadena '>f' indica que se incluya un float en big-endian. Los ceros añadidos como \00 son los necesarios en cada caso para cumplimentar hasta múltiplos de 4 la longitud de cada campo.

4.5 Otras aplicaciones

Como se ha comentado en el apartado 3.7, TouchOSC es utilizada para montar sistemas de control para interactuar con otras aplicaciones que utilicen OSC o MIDI como protocolo de comunicación. En el caso de este proyecto interesan aquellas que permiten la interacción mediante OSC ya que utilizando un mayor número de sensores sería posible montar sistemas de control de igual manera que se ha hecho, pero para aplicaciones más complejas.

Capítulo 5. Conclusiones y líneas futuras

Tras la realización de este trabajo, podemos asegurar que Micropython es un lenguaje de programación que permite simplificar y acelerar el desarrollo de sistemas interactivos, algo que se considera una novedad.

El bajo coste de los sistemas basados en ESP32 (los hay por menos de 5€) y de los sensores necesarios, permite desarrollar aplicaciones interactivas en muy poco tiempo, añadiendo capacidades de comunicación inalámbricas por WiFi o Bluetooth.

Comparado con sistemas similares basados en Arduino, además de disponer de mucha más capacidad de cálculo, se aumenta la productividad al trabajar con un lenguaje de más alto nivel e interpretado como MicroPython, y se dispone de comunicaciones inalámbricas, algo muy recomendable para hacer sensores y actuadores que van a ser manejados por personas en movimiento.

A modo de resumen, a lo largo de este trabajo fin de grado se ha:

- Empleado Micropython en aplicaciones interactivas audiovisuales.
- Manejado y procesado las señales de múltiples sensores empleados en aplicaciones de interacción.
- Realizado comunicaciones TCP y UDP entre el dispositivo y un ordenador a través de WiFi.
- Implementado un sencillo servidor web en el dispositivo para su configuración remota.
- Realizado una arquitectura cliente-servidor en el que toda la información captada de los sensores del dispositivo M5Stack se enviada a un ordenador en formato JSON que las grafica en tiempo real y almacena para su posterior procesado o análisis.
- Empleado el protocolo OSC para la interacción con SoundCool y TouchOSC a través de conexiones UDP, con ejemplos reales de aplicaciones de interacción audiovisual.

Este trabajo me ha permitido desarrollar de una manera aplicada muchos de los conocimientos adquiridos durante mis estudios, habiendo alcanzado con éxito los objetivos inicialmente propuestos. Además, me ha permitido aprender nuevos lenguajes de programación como Python y su versión reducida MicroPython, emplear comunicaciones IP, etc.

Como líneas futuras, se podría plantear aumentar la tipología de sensores como cámaras, posición y distancia con dispositivos de tiempo de vuelo *time-of-flight*, realizar un seguimiento de los mismos, etc. También podría estudiarse el nuevo protocolo OCA (Open Control Alliance) planteado por el AES (Audio Engineering Society) como AES70.

Capítulo 6. Bibliografía

[1] "M5Stack Fire," [Online]. Available: https://docs.m5stack.com/en/core/fire.

[2] "M5Stack," [Online]. Available: https://docs.m5stack.com/en/core/fire.

[3] "Espressif," [Online]. Available: https://www.espressif.com/.

[4] "ESP32 Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[5] "ESP8266 Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[6] "MPU6886 Datasheet," [Online]. Available: https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/docs/datasheet/core/MPU-6886-000193%2Bv1.1_GHIC_en.pdf.

[7] "I2C," [Online]. Available: https://www.i2c-bus.org/.

[8] "Phillips Semiconductors," [Online]. Available: https://www.nxp.com/.

[9] "HC-SR04 Datasheet," [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[10] "Flex sensor 2.2," [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEX%20SENSOR%20DATA%20SHEET%202014.pdf.

[11] "UiFlow," [Online]. Available: https://flow.m5stack.com/.

[12] "Arduino," [Online]. Available: https://www.arduino.cc/.

[13] "Micropython," [Online]. Available: https://micropython.org/.

[14] "Python," [Online]. Available: https://www.python.org/.

[15] "m5stack/UIFlow-Code," [Online]. Available: https://github.com/m5stack/UIFlow-Code.

[16] "Pycom," [Online]. Available: https://pycom.io/.

[17] "Ports Micropython," [Online]. Available: https://github.com/micropython/micropython/tree/master/ports.

[18] "Visual Studio Code," [Online]. Available: https://code.visualstudio.com/.

[19] "Thonny," [Online]. Available: https://thonny.org/.

[20] "vscode-m5stack-mpy," [Online]. Available: https://marketplace.visualstudio.com/items?itemName=curdeveryday.vscode-m5stack-mpy.

[21] "Driver comunicación serie M5Stack," [Online]. Available: https://shop.m5stack.com/pages/download.

[22] "Extensión Pymakr," [Online]. Available: https://marketplace.visualstudio.com/items?itemName=pycom.Pymakr.

[23] "esptool," [Online]. Available: https://github.com/espressif/esptool.

[24] "GitHub," [Online]. Available: https://github.com/.

[25] "OpenSoundControl.org," [Online]. Available: https://opensoundcontrol.stanford.edu/.

[26] "MIDI Association," [Online]. Available: https://www.midi.org/specifications.

[27] "SoundCool," [Online]. Available: https://soundcool.org/.

[28] "TouchOSC," [Online]. Available: https://hexler.net/touchosc.

[29] "mpu6050," [Online]. Available: https://github.com/m5stack/UIFlow-Code/blob/master/lib/mpu6050.py.

[30] "micropython-hcsr04," [Online]. Available: https://github.com/rsc1975/micropython-hcsr04.

[31] "ESP32/ESP8266 MicroPython Web Server – Control Outputs," [Online]. Available: https://microcontrollerslab.com/esp32-esp8266-micropython-web-server/.

[32] "Select," [Online]. Available: https://docs.python.org/3/library/select.html.

[33] "Selectors," [Online]. Available: https://docs.python.org/3/library/selectors.html.

[34] "Matplotlib," [Online]. Available: https://matplotlib.org/.

[35] "Matlab," [Online]. Available: https://es.mathworks.com/products/matlab.html.

[36] "Numpy," [Online]. Available: https://numpy.org/.

[37] "Cómo trazar datos en tiempo real usando Matplotlib," [Online]. Available: https://www.delftstack.com/es/howto/matplotlib/how-to-plot-in-real-time-using-matplotlib/#canvas.draw-junto-con-canvas_flush_events.

[38] "SpotlightKid/micropython-osc," [Online]. Available: https://github.com/SpotlightKid/micropython-osc.

Capítulo 7. Anexos

Códigos desarrollados disponibles en el repositorio de GitHub creado para este trabajo https://github.com/mbayolmeda/SISTEMA-INTERACCION-APLICACIONES-AUDIOVISUALES-MICROPYTHON