



TELÉCOMUNICACIÓN

Campus Sur
POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Diseño y construcción de un controlador MIDI de estructura distribuida a partir del modelo Behringer FCB1010

AUTOR: M^a Elena Nieto Bernad

TITULACIÓN: Ingeniería de Sonido e Imagen

TUTOR: Lino Pedro García Morales

DEPARTAMENTO: Ingeniería Audiovisual y Comunicaciones

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Pedro Cobos Arribas

TUTOR: Lino Pedro García Morales

SECRETARIO: Jorge Grundman Isla

Fecha de lectura:

Calificación:

El Secretario,

Agradecimientos

A mi tutor, Lino, por su disponibilidad, ayuda y apoyo durante todo este largo proceso.

A Anna, por explicarme microprocesadores con ejemplos de “Chicas Malas”. A Patri, por todas las veces que me hizo ir a la biblioteca, aunque luego ella no apareciese.

A Antía, por ser meyalayer e porlle tanto interese en entender estas páxinas.

A mi abuelo Mateo, por todas las veces que dejó en blanco el sudoku del periódico para que lo hiciese yo. A mi abuela Maribel, por los huevos al vaso y las tardes jugando a las tiendas.

Pero, sobre todo, a mi madre, por todos los sacrificios que ha hecho por mí, por hacerme sentir valiosa y por enseñarme a querer bonito.

Resumen

Diseño y construcción de un controlador MIDI de estructura distribuida a partir del modelo Behringer FCB1010.

Este proyecto aporta una solución tecnológica a los problemas encontrados por los usuarios habituales de la pedalera FCB1010 de la marca Behringer. A diferencia del resto de modificaciones expuestas en los foros especializados, que adaptan el controlador a una unidad multiefectos, este proyecto enfoca su uso a la aplicación JamUp, el *software* multiefectos más utilizado para iPhone o iPad.

El diseño y el código resultante se comparten para que cualquier poseedor del dispositivo original pueda realizar la modificación que se propone. Por esta razón se procura que resulte lo más económico posible.

El nuevo diseño está basado en una estructura distribuida compuesta de distintos módulos interconectados mediante el protocolo I2C. Para crear el bus de comunicación se utiliza cable telefónico y conectores RJ11, lo que hace muy fácil añadir, intercambiar y eliminar módulos. Cada uno de ellos cuenta con un microcontrolador que alberga su lógica; el del modulo principal es ESP32 y el del resto de módulos es Attiny85. Todos se han programado usando el *software* Arduino de libre distribución. Cabe destacar que, para el módulo de la pantalla LCD (*Liquid-Crystal Display* o Pantalla de Cristal Líquido) se adquiere un módulo preparado para conectar mediante I2C que ya se encuentra programado.

El Módulo Principal gobierna la lógica global del dispositivo: recibe eventos de los módulos secundarios y controla los modos de funcionamiento y los menús de configuración. A partir de toda la información recibida envía los comandos MIDI (*Musical Instrument Digital Interface* o “Interfaz Digital de Instrumentos Musicales”). Este envío se hace tanto por cable como por Bluetooth BLE (*Bluetooth Low Energy* o “Bluetooth de Bajo Consumo”). Esto permite controlar la aplicación sin necesidad utilizar un cable entre pedalera y móvil o iPad.

La configuración, uno de los puntos a mejorar destacado por la mayoría de los usuarios, también se realiza por este tipo de Bluetooth. Los comandos se cargan en el dispositivo mediante un fichero JSON (*JavaScript Object Notation* o “Notación de Objeto de JavaScript”), que se envía desde el teléfono móvil.

El procedimiento elegido para abordar el proyecto se fundamenta en un sistema distribuido basado en el paradigma de los sistemas complejos. A partir de un planteamiento inicial en el que se definen los diferentes módulos, estos se abordan uno a uno. Se concreta su diseño, se construye un módulo de prueba y se realiza el programa. Posteriormente se construye el módulo final y se integra en el

dispositivo. Una vez instalados todos se efectúan pruebas en el bus de comunicación para conseguir intercambiar información entre todos los módulos. Finalmente, se elabora el programa principal que opera con los datos recibidos procedentes de los módulos secundarios.

La incorporación de la pantalla LCD, la función *stompbox* con encendido de LEDs y los *displays* de 7 segmentos que muestran el *preset* hacen de él un controlador MIDI muy intuitivo. Se consigue centralizar en la pedalera controladora toda la información, siendo innecesaria cualquier consulta o interacción directa con el teléfono móvil o *tablet*. Durante la interpretación, la sensación es muy parecida a la de controlar una pedalera multiefectos.

Asimismo, su uso resulta mucho más cómodo por el empleo de Bluetooth BLE para enviar los comandos MIDI y la simplificación de la configuración mediante ficheros JSON.

Se considera que los resultados del proyecto son satisfactorios, aunque tienen cierto margen de mejora. Como se ha comentado, uno de los objetivos principales del proyecto es que otros usuarios lo reproduzcan adaptándolo a sus necesidades. Por esta razón, aunque se proponen una serie de modificaciones que se considera que podrían resultar interesantes, se deja a elección de los usuarios el camino a seguir.

Abstract

MIDI controller design and construction following a distributed structure based on Behringer FCB1010 model.

This project tries to provide a technological solution to main problems encountered by Behringer's FCB1010 users. It adapts this MIDI (Musical Instrument Digital Interface) foot controller to JamUp application, the most used multi-effects software for iPhone or iPad. This project differs from the rest of the modifications found in specialized forums, which adapt the foot controller to multi-effects rack units.

The code and design will be shared. In such way, every device user will be able to reproduce this modification. For this reason, we try the materials to be as cheap as we can.

The new design follows a distributed structure. A range of modules interconnected by I2C protocol (Inter-Integrated Circuit). Telephone cable and RJ11 (Register Jack 11) connectors are used in order to construct the communication bus. It makes very easy to add, change and remove modules. Every module has a microcontroller which runs its logic. The main module microcontroller is ESP32. The secondary modules microcontroller are Attiny85, all programmed using Arduino free software. It is noted that, for the LCD (Liquid-Crystal Display), it is used an already prepared I2C module

The main module controls device logic. It also controls operating modes and configuration options. It receives events from secondary modules and send MIDI (Musical Instrument Digital Interface) commands via cable and BLE Bluetooth (Bluetooth Low Energy). It allows you to control de application with no need for cable between the device and the iPad or iPhone.

The configuration, which is one of the points of improvement highlighted by the original device users, is also made via BLE Bluetooth. In order to configure the machine, the user sends a JSON (JavaScript Object Notation) file containing commands from the mobile phone to the device.

The chosen methodology to approach the project is based on its distributed structure, based on complex systems paradigm. The different modules are determined and, afterwards, they are handled one by one. First of all, test model is designed up and constructed. After that, the logic is programmed. Then, the final module is constructed and integrates into the device. Once all modules are integrated, the bus communication is tested in order to exchange information

between all the modules. Finally, the main module logic, which handles all the received data by secondary modules, is programmed.

The new MIDI foot controller is a very intuitive instrument. This was possible thanks to the new LCD screen, the stompbox functionality with LED activation and the seven segment displays which shows the current preset. Practically all the information provided by the iPad or iPhone is centralized in the controller. In such way the user does not need to check or tap the tablet. It makes the device utilisation very similar to a multieffect foot controller utilisation.

Moreover, the Bluetooth BLE makes the device much more comfortable. You can use it to send MIDI commands but also to configurate the unit with a very simple JSON file.

In conclusion, although there is margin for improvement, we reckon we achieve satisfactory results. As we mentioned before, one of the project goals is to get other people to replicate it, tailoring it to their needs. Therefore, even though we propose some interesting modifications, we reckon that the point is to let the community choose which path to follow.

Índice de contenidos

Resumen	5
Abstract	7
Índice de contenidos	9
Índice de Figuras	11
Índice de Tablas	14
Lista de acrónimos	15
Lista de Términos Anglosajones	16
1 Introducción	17
2 Marco tecnológico	19
2.1 Tecnología y música	19
2.2 MIDI	21
2.2.1 El nacimiento del MIDI	21
2.2.2 El impacto del MIDI	22
2.2.3 Características del MIDI	22
2.3 Filosofía DIY y Arduino	24
2.4 Pedalera Behringer FCB1010	25
2.4.1 Cambio de memoria EPROM	26
2.4.2 Otros proyectos	26
2.5 Aplicación JamUp	27
3 Especificaciones y restricciones de diseño	31
3.1 Descripción funcional del dispositivo original	31
3.2 Descripción técnica del dispositivo original	33
3.3 Especificaciones del nuevo controlador	38
3.4 Metodología	39
4 Descripción de la solución propuesta	41
4.1 Descripción general de la propuesta	41
4.1.1 Funcionamiento del dispositivo	44
4.2 Módulos secundarios	58
4.2.1 Módulo Pulsador LED	59
4.2.2 Módulo Pulsador	64
4.2.3 Módulo Pedal de Expresión	65
4.2.4 Módulo Display	68

4.2.5 Módulo LCD	71
4.3 Módulo Principal	73
4.3.1 Diseño	73
4.3.2 Lógica	76
5 Resultados	91
6 Presupuesto	93
6.1 Módulo Pulsador LED	93
6.2 Módulo Pulsador	94
6.3 Módulo Pedal de Expresión	95
6.4 Módulo Display	96
6.5 Módulo LCD	96
6.6 Módulo principal	97
6.7 Presupuesto total	98
7 Conclusiones	101
8 Referencias	105
9 Anexos	107
9.1 Manual de usuario	107
9.1.1 Conexión Bluetooth	107
9.1.2 Funcionamiento del dispositivo	109
9.1.3 Configuración del dispositivo	110
9.2 Esquemas electrónicos	125
9.2.1 Módulos Secundarios	125
9.2.2 Módulo Principal	127
9.2.3 Conexionado	128
9.3 Índice de Figuras	129
9.4 Índice de Tablas	130

Índice de Figuras

Figura 1. Conector DIN de 5 pines.	23
Figura 2. Controlador MIDI Behringer FCB1010. Imagen superior: vista frontal; Imagen inferior: vista superior.	31
Figura 3. Vista posterior del controlador sin la cubierta. [6]	34
Figura 4. Placas de los pedales de expresión. Izq: lugar en el que se encuentran; Dcha: placa en detalle. [6]	34
Figura 5. Placa de conectores MIDI. Izq: lugar en el que se encuentran; Dcha: placa en detalle. [6]	35
Figura 6. Placa de conectores Switch. Izq: posición en la que se encuentran; Dcha: placa en detalle. [6]	35
Figura 7. Conector, interruptor y transformador. Imagen superior: posición en la que se encuentran. [6].....	35
Figura 8. Vista frontal de la placa principal. [6]	36
Figura 9. Arrays de transistores y resistencias de la placa principal. [6].....	36
Figura 10. Parte derecha de la placa principal del dispositivo. [6]	37
Figura 11. Algunos de los circuitos integrados de la placa principal en detalle. [6]	37
Figura 12. Vista posterior de la placa principal del dispositivo. [6]	37
Figura 13. Montaje de prueba realizado para el desarrollo de los distintos módulos.	40
Figura 14. Tramas de datos de un mensaje del protocolo I2C.	41
Figura 15. Izq: cable telefónico (cuatro vías); Dcha: conector RJ11 con el cable telefónico insertado.	43
Figura 16. Esquema del bus I2C.	43
Figura 17. Pulsadores del cambio de modo mediante pulsación larga.....	44
Figura 18. Pulsadores de la funcionalidad stombox.	46
Figura 19. Esquema con la numeración de pulsadores, los comandos de efectos y cambio de preset.	46
Figura 20. Pulsaciones asociadas a comandos del Modo Preset.	47
Figura 21. Pulsaciones asociadas a comandos del Modo Jam.	48
Figura 22. Pulsaciones asociadas a comandos del Modo Sampler.	48
Figura 23. Apariencia del nuevo controlador MIDI.....	49
Figura 24. Opciones de configuración en la pantalla LCD.	49

Figura 25. Diagrama de uso del Modo de Configuración SLD (salvado del estado de los LEDs).....	51
Figura 26. Diagrama de uso del Modo de Configuración LLD (cargado del estado de los LEDs).....	52
Figura 27. Ejemplo de un fichero JSON que sirve para configurar todos los comandos Control Change.....	53
Figura 28. Ejemplo de un fichero JSON que sirve para configurar algunos comandos de Modo Libre.....	56
Figura 29. Diagrama de uso del Modo de Configuración DSP (añadir dispositivo al bus).	57
Figura 30. Esquema de distribución de los módulos. Vista posterior del dispositivo.	58
Figura 31. Microcontrolador Attiny85. Izq: imagen del dispositivo; Dcha: esquema de pines.	59
Figura 32. Localización de los Módulos Pulsador LED en la parte posterior del dispositivo.	59
Figura 33. Esquema electrónico del Módulo Pulsador LED.	61
Figura 34. Uno de los diez Módulos Pulsador LED que componen el dispositivo.	61
Figura 35. Diagrama de pseudocódigo del Módulo Pulsador LED.....	62
Figura 36. Localización de los Módulos Pulsador en la parte posterior del dispositivo.	64
Figura 37. Izq: Esquema electrónico del Módulo Pulsador; Dcha: uno de los módulos que componen el dispositivo.....	64
Figura 38. Localización de los Módulos Pedal de Expresión en la parte posterior del dispositivo.....	65
Figura 39. Esquema electrónico del Módulo Pedal de Expresión.	66
Figura 40. Uno de los dos Módulos Pedal de Expresión que componen el dispositivo.	66
Figura 41. Diagrama de pseudocódigo del Módulo Pedal de Expresión.	67
Figura 42. Localización de los Módulos Display en la parte posterior del dispositivo.	68
Figura 43. Esquema electrónico del Módulo Display.....	69
Figura 44. Módulo Display instalado en el dispositivo.....	70

Figura 45. Diagrama de pseudocódigo del Módulo Display	71
Figura 46. Localización de los Módulos Display en la parte posterior del dispositivo.	72
Figura 47. Módulo LCD. Izq: vista general del módulo; Dcha: vista en detalle de la instalación del conector RJ11.	72
Figura 48. Localización del Módulo Principal en la parte posterior del dispositivo. A la izquierda, la placa principal. A la derecha, la placa MIDI.	73
Figura 49. Esquema de pines del microcontrolador ESP32.	74
Figura 50. Placa principal del Módulo Principal.....	75
Figura 51. Placa MIDI del Módulo principal. Izq: vista frontal; Dcha: vista posterior.	75
Figura 52. Esquema electrónico del Módulo Principal.....	76
Figura 53. Diagrama de pseudocódigo de Modo Preset.	78
Figura 54. Diagrama de pseudocódigo de los pedales de expresión.	80
Figura 55. Diagrama de pseudocódigo del Modo Jam.	81
Figura 56. Diagrama de pseudocódigo del Modo Sampler	82
Figura 57. Diagrama de pseudocódigo del Modo Libre.	83
Figura 58. Diagrama de pseudocódigo del modo de configuración SLD.	84
Figura 59. Esquema de contenido de los bytes ocupados en la memoria flash del microcontrolador ESP32.	85
Figura 60. Diagrama de pseudocódigo del modo de configuración LLD.	86
Figura 61. Diagrama de pseudocódigo del modo de configuración EXP.	88

Índice de Ecuaciones

Ecuación 1. Pendiente de la recta de calibración.	88
Ecuación 2. Desplazamiento de la recta de calibración.	88

Índice de Tablas

Tabla 1. Asignación de funciones MIDI a los pulsadores. [6]	32
Tabla 2. Tipos de mensajes MIDI	54
Tabla 3. Relación entre pines del registro de desplazamiento y del display de siete segmentos.	68
Tabla 4. Materiales necesarios para la construcción de los diez Módulos Pulsador LED	94
Tabla 5. Coste de la mano de obra para el desarrollo y construcción del Módulo Pulsador Led.	94
Tabla 6. Materiales necesarios para la construcción de los dos Módulos Pulsador.	94
Tabla 7. Coste de la mano de obra para el desarrollo y construcción del Módulo Pulsador.	95
Tabla 8. Materiales necesarios para la construcción de los dos Módulos Pedal de Expresión.	95
Tabla 9. Coste de la mano de obra para el desarrollo y construcción del Módulo Pedal de Expresión.....	95
Tabla 10. Materiales necesarios para la construcción del Módulos Display.	96
Tabla 11. Coste de la mano de obra para el desarrollo y construcción del Módulo Display	96
Tabla 12. Materiales necesarios para la construcción del Módulo LCD.....	96
Tabla 13. Coste de la mano de obra para el desarrollo y construcción del Módulo LCD.....	97
Tabla 14. Materiales necesarios para la construcción del Módulo Principal.	97
Tabla 15. Coste de la mano de obra para el desarrollo y construcción del Módulo Principal.	97
Tabla 16. Presupuesto de material adicional para el desarrollo y construcción del proyecto.	98
Tabla 17. Desglose del coste material del proyecto.	99
Tabla 18. Coste total de la mano de obra de todo el proyecto desglosado por módulos.....	99
Tabla 19. Coste total del proyecto	99

Lista de acrónimos

ACK	<i>Acknowledgement</i> (Acuse de Recibo)
BLE	<i>Bluetooth low energy</i> (Bluetooth de baja energía)
DIY	<i>Do It Yourself</i> (Hazlo Tú Mismo)
EEPROM	<i>Electroically Erasable Programmable Read-Only Memory</i> (Memoria Programable de Solo Lectura Borrable Eléctricamente)
EPROM	<i>Erasable Programmable Read-Only Memory</i> (Memoria Borrable Programable de Solo Lectura)
GND	<i>Ground</i> (Tierra)
GPIO	<i>General Purpose Input/Output</i> (Entrada/Salida de Propósito General)
I2C	<i>Inter-Integrated Circuit</i> (Circuito Inter-Integrado)
ISP	<i>In-system programming</i> (Programación en el sistema)
IEC	International Electrotechnical Commission (Comisión Electrotécnica Internacional)
JSON	<i>JavaScript Object Notation</i> (Notación de Objeto de JavaScript)
LCD	<i>Liquid Cristal Display</i> (Pantalla de Cristal Líquido)
LED	<i>Light Emitting Diode</i> (Diodo Emisor de Luz)
LSByte	<i>Least Significant Byte</i> (Byte Menos Significativo)
MIDI	<i>Musical Instrument Digital Interface</i> (Interfaz Digital de Instrumentos Musicales)
MIPS	Millón de Instrucciones Por Segundo
MSByte	<i>Most Significant Byte</i> (Byte Más Significativo)
NAMM	<i>National Association of Music Merchants</i> (Asociación Nacional de Comerciantes de Música)
NACK	<i>Negative Acknowledgement</i> (Acuse de Recibo Negativo)
PWM	<i>Pulse Width Modulation</i> (Modulación por Ancho de Pulso)
RAE	Real Academia Española
RAM	<i>Random Access Memory</i> (Memoria de Acceso Aleatorio)
RJ11	<i>Register Jack 11</i> (Conector Registrado 11)
SCL	<i>Serial Clock</i> (reloj)
SD Card	<i>Secure Digital Card</i> (Tarjeta con Seguridad Digital)
SDA	<i>Serial Data</i> (datos)
SPI	<i>Serial Peripheral Interface</i> (Interfaz Periférica Serial)
SRAM	<i>Static Random Access Memory</i> (Memoria Estática de Acceso Aleatorio)
UART	<i>Universal Asynchronous Receiver-Transmitter</i> (Transmisor-Receptor Asíncrono Universal)

USI	<i>Universal Synthesizer Interface</i> (Interfaz Universal de Sintetizadores)
USB	<i>Universal Serial Bus</i> (Bus Universal en Serie)
VCC	<i>Voltage Common Collector</i> (Colector Común de Voltaje)

Lista de Términos Anglosajones

Array	Tipo de dato estructurado que contiene elementos del mismo tipo que se guardan en posiciones de la memoria contiguas.
Display	Para referirse al <i>display</i> de 7 segmentos, la traducción más correcta es “visualizador”.
Hardware	Elementos físicos de una máquina.
Jam	Interpretar música conjuntamente sin previo ensayo.
Looper	Dispositivo que graba y reproduce sonidos que se repiten en bucle.
Rack	Estante metálico que se utiliza para albergar un dispositivo tecnológico.
Sampler	Instrumento musical electrónico que reproduce sonidos grabados o precargados.
Software	Programas, datos, procedimientos y pautas que permiten ejecutar tareas en un sistema informático.
Stompbox	Pedal que recibe una señal de audio y le aplica un efecto a su salida.
Pad	Panel percutible o pulsable que se encuentra en instrumentos musicales electrónicos como cajas de ritmos.
Plugins	Programa informático que sirve para ampliar aplicaciones existentes.
Preset	Configuración de los distintos parámetros de un dispositivo.
Protoboard	Placa de pruebas o placa de inserción.
Switch	Comutador.
Tablet	Tipo de ordenador portátil con pantalla táctil y de mayor tamaño que un teléfono móvil.
Tap	Pulsaciones que sirven para establecer el tempo en un aparato electrónico.

1 Introducción

La pedalera FCB1010 de la marca Behringer tiene una gran comunidad de adeptos sobre todo por su bajo coste y su alta calidad de construcción. Entre los usuarios se comparten quejas, dudas y preocupaciones, por lo que navegando por Internet se pueden conocer sus necesidades fácilmente. Atendiendo a estas se fijan los requisitos del nuevo dispositivo. Los más destacables: facilitar la configuración y añadir funcionalidad *stompbox* (la capacidad de asociar un pedal a un efecto).

En estos foros se proponen distintas modificaciones para mejorar la interactividad del dispositivo que ofrece el *software* del fabricante. La mayoría de ellas han quedado prácticamente obsoletas, ya que adaptan el uso del controlador a unidades multiefectos, cuyo precio es mayor de mil euros. Estos procesadores están cayendo en desuso frente al auge de los *plugins software*, que ofrecen una gran calidad a precio muy bajo. Por esta razón, se propone enfocar este proyecto al uso de la pedalera con el *software* multiefectos más utilizado por usuarios de Apple: la aplicación JamUp.

El objetivo es transformar el dispositivo original en una pedalera totalmente adaptada a JamUp, con conexión Bluetooth BLE, función *stompbox*, interfaz amigable con pantalla LCD (*Liquid Cristal Display* o “Pantalla de Cristal Líquido”), y configuración sencilla por Bluetooth. Sin embargo, no solo se limita su uso a la aplicación, sino que también provee un modo libre con gran flexibilidad. Además, todo ello se construye como un sistema distribuido compuesto de módulos interconectados mediante el protocolo I2C (*Inter-Integrated Circuit* o “Circuito Inter-Integrado”). En las siguientes páginas se realiza un recorrido por el planteamiento del proyecto y los resultados obtenidos. En primer lugar, en el apartado “marco tecnológico”, se aporta una visión general del campo de conocimiento en el que se encuentra el proyecto con el objetivo de justificar su necesidad. En el siguiente apartado, “especificaciones y restricciones de diseño”, se establecen los requisitos de diseño a partir de las necesidades de los usuarios y del dispositivo original. Posteriormente, en el apartado “descripción de la solución propuesta” se presenta el diseño y la lógica de cada uno de los módulos que componen el dispositivo.

En el apartado de “resultados” se analizan las modificaciones realizadas respecto a las características del modelo original. Seguidamente, en el apartado “presupuesto” se calcula el coste material y de mano de obra del proyecto. Finalmente, en las conclusiones se reflexiona acerca del cumplimiento de los objetivos iniciales y posibles mejoras que se podrían incorporar.

En el anexo se incluyen los esquemas electrónicos y un manual enfocado al usuario de la pedalera para mostrar casos de uso y operación de configuración.

2 Marco tecnológico

2.1 Tecnología y música

La tecnología ha acompañado a la música durante prácticamente toda su historia, desde que el ser humano comenzó a utilizar instrumentos para producir sonidos; de hecho existe una ciencia, la organología, que estudia los instrumentos musicales y su clasificación.

La RAE (Real Academia Española) define **tecnología** como el “conjunto de teorías y técnicas que permiten el aprovechamiento práctico del conocimiento científico”. Acogiéndonos a esta definición, cualquier instrumento musical es un aparato tecnológico. Por ejemplo, un violín se basa en que cuerdas de diferentes grosores, longitudes y tensiones vibran a distinta frecuencia, es decir, producen sonidos de diferente altura, y producen resonancias complejas que conforman un sonido único.

La división geométrica de la cuerda fue una cuestión que estudiaron los pitagóricos en el siglo V aC; construyeron el monocordio, un instrumento musical de una sola cuerda que les sirvió de apoyo en sus investigaciones. Hubo, por tanto, un aprovechamiento práctico de un conocimiento científico. Hace más de 2500 años la tecnología y la música ya iban de la mano.

Actualmente sigue existiendo cierta creencia popular de que la música cuanto menos tecnológica es más “pura”, tiene más verdad. No obstante, la síntesis del sonido se entiende, cada vez más, como una evolución natural de la creación musical. Y es que, a lo largo de la historia de la música siempre se ha buscado generar nuevos timbres y sonoridades.

Un ejemplo de ello podría ser la invención del saxofón. En la década de 1840, Adolphe Sax tuvo la idea de crear “un instrumento con una boquilla de caña como la del clarinete, el cuerpo de un fíge y las propiedades acústicas de la flauta” [1]. La creación de Sax fue resultado de su entorno (pasaba el día trabajando en la tienda de instrumentos de su padre) y sus conocimientos (era clarinetista). De igual manera, como nuestra realidad actual está basada fundamentalmente en la tecnología digital, usamos *software* para encontrar de nuevos sonidos.

Sin entrar aún en la era digital, muchos avances científicos se han sido aplicados a la música. En los años 30 apareció la guitarra eléctrica, basada en la Ley de Inducción Electromagnética enunciada por Faraday un siglo antes: “El voltaje inducido a través de un conductor que se desplaza transversal a un campo magnético es proporcional a la velocidad del conductor” [2]. Se utilizaron unos imanes permanentes con bobina de cobre para crear el campo magnético: las denominadas pastillas. Sabiendo que, el movimiento de la cuerda (el conductor),

dentro de ese campo magnético provocaría una corriente en la bobina proporcional a la amplitud del movimiento y de la frecuencia a la que vibra la cuerda.

Este instrumento provocó la aparición y posterior auge del *rock and roll*. Este género ocasionó un giro de 180 grados en la historia de la música y trajo consigo un cambio en la sociedad, sobre todo en la forma de comportarse de la juventud. Rápidamente, el *rock* pasó de ser un género marginal a estar de moda. Los jóvenes adoptaron la estética y la conducta asociada a este tipo de música y la guitarra eléctrica se convirtió en un símbolo de rebeldía.

Como es obvio, este cambio tan profundo tuvo un fuerte impacto en la industria musical. El perfeccionamiento de la grabación musical, unido a la aparición del fenómeno pop con *The Beatles*, generó muchos beneficios, sobre todo, a las grandes compañías discográficas del momento. Aunque inicialmente hubo reticencias a la música grabada (había músicos que incluso tenían miedo de que fuese el fin de la música en vivo), la industria supo sacar provecho económico de los cambios que se produjeron en la primera mitad del siglo XX. Sin embargo, no se adaptaron tan bien al auge de la música digital.

Lo digital hizo su primera aparición en el mundo de la música con el MIDI (*Musical Instrument Digital Interface* o “Interfaz Digital de Instrumentos Musicales”). Este estándar tecnológico permite la comunicación entre dispositivos mediante un protocolo, una interfaz digital y unos conectores específicos.

A finales de los años 70, los sintetizadores analógicos eran cada vez más comunes. Solían ser monofónicos y se controlaban por el voltaje producido por sus teclados. Estos voltajes no eran iguales en los diferentes dispositivos de las distintas compañías. Con el MIDI, apareció una interfaz universal y, con ella, la posibilidad de generar una enorme cantidad de timbres con teclados de diferentes fabricantes.

El auge de este estándar coincidió con la llegada de los ordenadores personales, los *samplers* y los sintetizadores digitales, que almacenaban sonidos programados previamente y se reproducían accionando un pulsador. Se empezó a extender el uso del sintetizador en los distintos géneros musicales y tomó fuerza el *Synth Pop*, precursor del *Dance*.

Pero no solo provocó un cambio en el sonido, sino también en la manera de hacer música y, sobre todo, en su democratización.

2.2 MIDI

2.2.1 El nacimiento del MIDI

A finales del siglo XIX era posible hacer realidad a la idea de generar sonido de manera electrónico. En 1897, Thaddeus Cahill construyó el Dynamophone (o Telharmonium), considerado el primer sintetizador de sonido. Esta máquina realizaba síntesis aditiva mediante la rotación de generadores electromagnéticos que producían impulsos eléctricos que se convertían en sonido a través de receptores telefónicos.

Durante la primera mitad del siglo XX, la generación de sonido “electrónico” siguió su avance. En un principio, controlar el sonido de estos aparatos era algo costoso. Por ejemplo, si se quería realizar un cambio a una frecuencia concreta, era preciso rotar el potenciómetro del generador de funciones hasta encontrarla. En 1961, apareció finalmente el control por voltaje, que permitía hacer cambios en la señal principal mediante señales de control. De esta forma se realizaban cambios de frecuencia, timbre, forma de onda e intensidad. Los dispositivos de control y producción se conectaban unos a otros como si de unidades independientes se tratasesen, lo que desembocó en la popularización de los sintetizadores modulares. En 1964, finalmente, Robert Moog y Don Buchla comercializaron, de forma paralela, lo que hoy conocemos como sintetizador, un sistema modular basado en el control por voltaje.

Aunque el control por voltaje supuso un gran avance, “era muy limitado tanto en prestaciones como en flexibilidad” [3]. Además, no había un acuerdo entre fabricantes, por lo que era complejo conectar dispositivos de marcas distintas.

En este contexto, la empresa *Sequencial Circuits*, encabezada por Dave Smith y Chet Wood, desarrollaron un protocolo de comunicación al que llamaron USI (*Universal Synthesizer Interface* o “Interfaz Universal de Sintetizadores”). El USI fue presentado en 1981 con el objetivo de que se convirtiera en un estándar adoptado por todos los fabricantes. Ofrecía una transmisión serie a 19,2 kilobaudios, niveles lógicos TTL (0 (de 0 a 5V) y jacks estéreos convencionales como conectores.

En el NAMM (*National Association of Music Merchants* o “Asociación Nacional de Comerciantes de Música”) de 1982, se acordó entre los fabricantes aumentar la velocidad de transmisión a 31,25 kilobaudios y añadir optoacopladores a los terminales de entrada para evitar interferencias, ruidos y desastres. Lo acordado, unido a las mejoras aportadas por las compañías japonesas, es lo que actualmente se conoce como MIDI.

Han pasado casi 40 años desde su creación y la tecnología MIDI, con las adaptaciones que ha ido aceptando a lo largo de los años, sigue siendo de vital

importancia en la grabación y producción de música profesional. A día de hoy, las modificaciones que ha sufrido el protocolo han sido mínimas, lo que confirma la gran efectividad que tuvo desde su lanzamiento.

La universalidad del formato se considera “un ejemplo precursor de lo que ahora se denomina tecnología de código abierto (*open source*)” [4]. Si bien los creadores son conscientes de los beneficios que les podría haber reportado, el objetivo de su creación fue conseguir que lo implantaran todos los fabricantes y que cualquier persona tuviese acceso.

2.2.2 El impacto del MIDI

La repercusión de este estándar tecnológico fue brutal. El músico argentino Cineplexx expone muy acertadamente que “se separó la tecla del sonido”. Para exemplificar la importancia de esto cita el ejemplo de la banda Kraftwerk, una de las bandas pioneras de música electrónica, de la que dice que “utilizaba doscientos teclados analógicos distintos”. La posibilidad de crear todos esos sonidos con un solo teclado, no solo supuso mayor comodidad para los grupos que ya tenían todo ese equipamiento analógico, sino que le dio la oportunidad de hacer ese tipo de música a grupos con menos recursos; es decir, democratizó la producción musical.

Por otro lado, la música pop también utilizó mucho este estándar en sus primeros años de vida. De hecho, fue un factor importante en su estandarización, ya que este avance satisfacía la necesidad que este estilo tenía de secuenciar frases tocadas por sintetizadores. El MIDI era tan fácil y accesible que, en cierta manera, redefinió la música pop de estos años, tiñéndola con un toque electrónico. Esto supuso además el nacimiento de una gran cantidad de nuevos estilos.

Cabe destacar la importancia del MIDI en la producción y grabación musical. Este estándar, unido a la revolución de la grabación digital y a la democratización de los ordenadores personales, permite grabar con una calidad más que aceptable, piezas de cualquier estilo de música con muy poco presupuesto, lo que ha supuesto una transformación mayúscula en la industria musical.

2.2.3 Características del MIDI

En resumen, el MIDI es un estándar tecnológico que define un protocolo de transmisión de datos, una interfaz digital y unos conectores. Es decir, un acuerdo entre fabricantes y países sobre cómo codificar, transmitir y recibir información con el objetivo de facilitar la interconexión de equipos.

En el protocolo que define el estándar MIDI cada comando, en general, consta de 3 bytes que se transmiten en serie, es decir, uno detrás de otro, a una velocidad de 31250 bits por segundo. El primer byte es el Status Byte o byte de estado, que indica

qué función activar y en qué canal. Algunas de las funciones más utilizadas son: “*Note On*” (activar nota), “*Note Off*” (desactivar nota), “*System Exclusive*” (sistema exclusivo), *Control Change* (cambio de controlador) y *Program Change* (cambio de programa). Puede operar en 16 canales diferentes numerados del 0 al 15. Cada dispositivo acepta el comando del canal en el que esté configurado para recibir datos.

Hay dos tipos de mensajes de canal y mensajes de sistema.

- Mensajes de canal: actúan solo en el canal que tienen especificado en el byte de estado. Hay de dos tipos:
 - o Mensajes de voz: dedicados a la interpretación. Son “*Note On*” (activar nota), “*Note Off*” (desactivar nota), *Control Change* (cambio de controlador) y *Program Change* (cambio de programa).
 - o Mensajes de modo: indican al sintetizador la distribución de las voces. Hay dos: *Omni on/off* (para recibir mensajes por todos los canales o solo por uno) y *Mono on/off* (para saber si cada canal toca una nota o es polifónico)
- Mensajes de sistema: afectan a todos los canales. Hay de 3 tipos:
 - o Mensajes comunes, como pueden ser los de afinación.
 - o Mensajes de tiempo real, como los mensajes de reloj.
 - o Mensajes *SysEx* (de sistema exclusivo), que son mensajes exclusivos que los dispositivos de la misma marca o modelo comparten para intercambiar información.

Además, en la norma MIDI se definen unos conectores: los DIN de 5 pines, con el de la Figura 1, de los que solo se utilizan 3:

- PIN 2: tierra.
- PIN 3: salida de la señal.
- PIN 4: entrada de la señal.

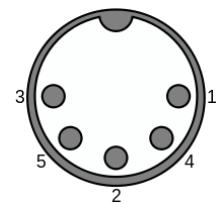


Figura 1. Conector DIN de 5 pines.

Los dispositivos MIDI suelen tener 3 conectores DIN de 5 pines hembra incorporados:

- MIDI IN: entrada de señales MIDI al dispositivo.
- MIDI OUT: salida de las señales MIDI generadas en el dispositivo.
- MIDI THRU: salida de una copia de las señales recibidas por MIDI IN; lo que permite conectar hasta 32 dispositivos MIDI.

2.3 Filosofía DIY y Arduino

Los orígenes de la filosofía DIY (*Do It Yourself* o “Hazlo Tú Mismo”) se remonta al siglo XVIII, durante la revolución industrial, cuando comienza la fabricación en serie y con ello, el capitalismo. Como respuesta surgen los primeros movimientos anticapitalistas, uno de ellos el DIY.

Casi dos siglos más tarde, la cultura punk rescata esta filosofía y comienzan a fabricar sus propias prendas e instrumentos. Sin embargo, no es hasta principios del siglo XXI cuando este movimiento contracultural empieza a pertenecer a la cultura de masas. Esto se debe en parte a la mayor conciencia ecológica, pero, sobre todo, a la expansión de Internet, donde hay tutoriales para hacer tú mismo prácticamente todo lo que puedas imaginar.

Para sus adeptos, las ventajas del DIY están claras: el ahorro, la personalización y el aprendizaje. Además, la gran mayoría disfrutan el proceso de fabricación. Según el producto final y el enfoque, sus seguidores se diferencian entre “*crafters*” (“artesanos”) y “*makers*” (“manitas”). Al primer grupo pertenecen los que realizan manualidades, la mayoría con intención decorativa. El segundo grupo, trata de resolver problemas, muchas veces de forma innovadora. La mayoría de las soluciones son tecnológicas y el componente que más se repite, en los proyectos de electrónica, es, sin duda, el microcontrolador Arduino.

En la página oficial, Arduino se define como una plataforma electrónica de código abierto basada en la facilidad de uso de *software* y *hardware*. En definitiva, son placas capaces de leer entradas digitales y analógicas (normalmente proveniente de sensores) y de transformarlas en salidas (normalmente de actuación) siguiendo una lógica.

Estos microcontroladores nacen en el año 2005 en Ivrea (Italia), como respuesta a la necesidad de los estudiantes de “Interaction Ivrea”, un instituto tecnológico, de tener dispositivos programables de bajo coste. Se buscaba que estos nuevos aparatos, además de económicos, fuesen lo suficientemente simples para que alguien sin altos conocimientos técnicos los pudiese utilizar.

Estos dispositivos se popularizaron rápidamente y, siguiendo esta filosofía de bajo coste y sencillez, aparecieron otros microcontroladores similares de otras marcas, aunque ninguna tan popular como Arduino. Este tipo de aparatos se han usado para todo tipo de proyectos, entre otros, para construir o personalizar controladores MIDI.

Debido a su bajo precio y robustez, el controlador Behringer FCB1010 es ideal para introducir este tipo de modificaciones. Por esta razón, navegando por la red se

pueden encontrar distintos proyectos que hacen uso de esta pedalera. Las mejoras presentes en la mayoría de ellos son la inclusión de una pantalla LCD y la facilitación de la configuración.

2.4 Pedalera Behringer FCB1010

Desde su creación, el MIDI tiene diferentes usos, por ejemplo: grabación de instrumentos VST (*Virtual Studio Technology* o “Tecnología de Estudio Virtual”), lanzamiento de secuencias, control de dispositivos, etc. En dependencia de para lo que se quiera utilizar, se necesita un tipo de controlador u otro. De esta manera, si lo que queremos es controlar un órgano emulado mediante VST, interesa un controlador MIDI en forma de teclado. Si, por otro lado, se quiere que nos sirva para introducir sonidos rítmicos como si de una caja de ritmos se tratase, nos interesaría más un controlador compuesto de diferentes *pads* (paneles percutibles).

El dispositivo Behringer FCB1010 es un controlador MIDI en forma de pedalera. Cada *preset* cuenta con dos pedales de expresión y diez pulsadores con LED a los que se pueden asignar cinco comandos *Program Change*, dos *Control Change* y un *Note on*.

Su diseño permite al usuario lanzar eventos MIDI mientras se está interpretando un instrumento (como puede ser una guitarra, bajo, teclado, etc.). Este controlador MIDI sería indicado, por ejemplo, para un grupo en el que el guitarrista es el encargado de activar y desactivar secuencias pregrabadas que se utilizan durante una interpretación en directo o incluso para controlar sus efectos.

Asimismo, por su fisonomía sería ideal para el manejo de *softwares* multiefectos. Sus 10 pulsadores y 2 pedales de expresión permitirían al usuario interactuar con los diferentes pedales de efecto como si fuesen pedales de efecto independientes. Sin embargo, este dispositivo no cuenta con esta función. Solamente se pueden enviar dos comandos CC por *preset* y los LEDs no quedan encendidos mostrando que el efecto está activado, sino que se encienden un momento y se apagan.

Este atractivo diseño, su bajo precio (apenas 120 euros) y su robustez, lo convierten en uno de los dispositivos MIDI favoritos por los usuarios a la hora de introducir modificaciones. Por esta razón, navegando por Internet es posible encontrar una gran cantidad de proyectos relacionados con la incorporación de mejoras a este aparato.

2.4.1 Cambio de memoria EPROM

La memoria EPROM (*Erasable Programmable Read-Only Memory* o Memoria Programable Borrible de Solo Lectura) es un chip de memoria de solo lectura no volátil, es decir, que no necesita energía para mantener la información guardada en ella. En este chip se encuentra la lógica del controlador. Toda la lógica del FCB1010 está programada en un EPROM, por ello el cambio de la memoria EPROM es una modificación que realizan la mayoría de usuarios de la pedalera FCB1010. De hecho, una gran parte de ellos no concibe utilizar el dispositivo con la memoria EPROM original, ya que la configuración es “tediosa”, para algunos incluso “imposible”; lo que se puede traducir en un *look and feel* “defectuoso” poco amistoso.

Hay dos chips diferentes muy populares: el EurekaPROM y el UnO. Este último es el más recomendado en los foros especializados, ya que tiene una programación más flexible. El precio de ambos chips ronda los 30 euros, gastos de envío incluidos. La instalación, aunque es abordable para alguien con poco contacto con la electrónica, no deja de ser un poco delicada por la fragilidad de los pines. Muchos usuarios cuentan que han tenido que hacer varios intentos y, por tanto, comprar varias memorias, antes de conseguir realizar la instalación de manera exitosa.

Se desarrollaron para sacar el máximo partido al *hardware* de la pedalera FCB1010 cuando se conecta a dispositivos multiefectos de guitarra como el Axe-Fx II de Fractal Audio Systems. Aunque este tipo de procesadores sigan ofreciendo un gran rendimiento, hoy en día, estos dispositivos se están viendo relevados por *softwares* emuladores, que son mucho más económicos (incluso los hay gratuitos) y cada vez de mejor calidad.

Estos chips, además de corregir ciertos errores, simplifican notablemente la configuración, que se puede realizar mediante un *software*. Además, se añade la funcionalidad *stompbox*, con la que se puede asignar un efecto a cada pulsador de la primera fila. Asimismo, entre otras cosas, se mejora el comportamiento de los pedales de expresión y el de la función MIDI *merge* (mezclado MIDI).

2.4.2 Otros proyectos

Por otro lado, hay una gran cantidad de proyectos, la mayoría desarrollados con Arduino, que van más allá y buscan una mayor personalización de la máquina. Aunque no es posible explicar detalladamente cada proyecto, a continuación, se recoge una colección de los mejor acogidos por la comunidad de adeptos del FCB1010, junto a las mejoras que introducen.

- PU-2 (*Practical Usage 2*): prácticamente el proyecto más conocido, por la claridad con la que está explicado y por todas las mejoras que introduce.

Utiliza el Arduino Mega ya que dispone de una gran cantidad de entradas. Mantiene la funcionalidad del FCB1010 y además añade características como conexión USB (bus universal en serie), pantalla de diez dígitos, alimentación *phantom* (fantasma), envío de comandos retardados, cancelar comandos anteriores o lectura de *SD Card* (*Secure Digital Card* o “Tarjeta Digital Segura”).

- FCBInfinity: este proyecto modifica la pedalera para utilizarla con la unidad multiefectos Axe-Fx II. Añade una pantalla LCD que muestra el efecto utilizado, el afinador, el estado del *looper* y las pulsaciones por minuto. Además, permite asignar cada efecto a un pulsador.
- FCB1010 mod: elimina los pedales de expresión y añade cinco potenciómetros para controlar dos procesadores de audio distintos. En su caso, una unidad multiefectos y un armonizador.

Los proyectos son muchos y muy variados, pero tienen algo en común: las modificaciones se realizan atendiendo a las necesidades del usuario que lo realiza. Además, prácticamente todos los proyectos encontrados son llevados a cabo por guitarristas que utilizan la pedalera para controlar unidades multiefecto. Se observa que las necesidades que se repiten son:

- Pantalla LCD.
- Asignación de un efecto a cada pulsador (funcionalidad *stompbox*).
- Posibilidad de controlar otras funciones del procesador. Por ejemplo, un *looper*.
- Conexión USB (*Universal Serial Bus* o Bus Serie Universal).
- Simplificación de la configuración.

2.5 Aplicación JamUp

Sobre todo, en la música popular, es muy común que los intérpretes, principalmente guitarristas, utilicen distintos efectos durante un mismo concierto o incluso durante una misma canción. Los cambios de timbre se deben poder realizar sin entorpecer la interpretación. Surge entonces la necesidad de controlar el sonido con los pies. Para ello hay distintas opciones.

La primera de ellas es la interconexión de pedales de efectos independientes, que modifican el sonido de una forma concreta, y los combinan entre sí. Aquí hay dos cuestiones: el precio y la flexibilidad. A mayor número de pedales (es decir, cuanto más dinero invertido), más posibilidades de la hora de modificar el sonido.

Por otro lado, hay pedaleras compactas, como la Line6 Helix, que ofrecen *software* y un *hardware* en forma de pedalera que hace muy cómodo su uso. Normalmente su precio elevado, pero no se necesita comprar ningún controlador adicional.

Asimismo, las unidades multiefectos como el como el Axe-Fx II ofrecen *software* y *hardware*. Sin embargo, el *hardware* es un aparato que no está preparado para que el músico interactúe con él durante una interpretación. Para controlar este tipo de dispositivos en directo es necesario disponer de un controlador de pie, con el que se puedan modificar sus parámetros sin tener que dejar de tocar. El precio de este tipo de artilugios es elevado y, además, hay que contar con que el gasto será mayor con la adquisición del controlador. La mayor parte de las modificaciones de la pedalera Behringer FCB1010 que se encuentran por Internet se enfocan en su utilización con estos tipos de dispositivos.

Por último, está la opción de adquirir un *plugin software*. Esta solución solo ofrece un *software* que debe ejecutarse en *hardware* genérico (una *tablet* o un ordenador). Estos *softwares* son muy flexibles, ya que cuentan con una gran variedad de efectos, y su precio es bastante menor. Mientras que las soluciones anteriores pueden superar los 1000 euros, los *softwares* más utilizados (Amplitube 4, Guitar Rig 5 Pro o Positive Grid BIAS FX) no superan los 200 euros. De igual manera que en la opción anterior, hay que contar con la adquisición de un controlador MIDI para calcular el coste total.

Cada vez es mayor el número de usuarios que optan por esta opción, debido a su versatilidad, calidad y bajo precio. Por esta razón, se decide enfocar la pedalera a su uso con un *software* de este estilo, ya que se considera que, de esta manera, el proyecto aporta valor a la comunidad. Así, se pone a disposición de los usuarios pedalera económica que permite controlar el *software* que está ejecutándose en su *tablet* en una actuación en directo.

Se elige utilizar la aplicación JamUp, de Positive Grids, por ser el procesador multiefectos de guitarra y bajo más descargado en iPhone y iPad. Y, además, porque es uno de los más aplaudidos por la prensa especializada. Hay dos versiones: una gratuita y otra de pago (solo 22 euros), que ofrece más posibilidades. La versión gratuita cuenta con un amplificador y 6 efectos, mientras que la versión de pago tiene 6 amplificadores completos y 16 pedales y efectos de *rack* Premium que incluyen distorsión, compresión, *tape delay* (simulación de un retardo analógico de cinta magnética), trémolo y *spring reverb* (reverberación de muelle). Además, puedes contar con 46 canales de amplificación y cuarenta efectos clásicos mediante paquetes de expansión.

Ambas versiones cuentan con:

- *Jam player*: función para cargar canciones y reproducirlas en la aplicación mientras que tocas. Se puede cambiar la tonalidad y tempo de la base para poder improvisar con distintas escalas y ritmos.
- *Phrase Sampler*: esta función te permite grabar sonidos y crear tus propios *loops* (bucles) sobre los que improvisar.

- *ToneSharing*: plataforma en la que puedes compartir tus *presets* y descargar los de otros usuarios y artistas.
- *Preset manager*: gestor de *presets* con 128 vacantes.
- Grabadora completa de 8 canales.
- Afinador
- Metrónomo.

Además, se puede combinar con BIAS, una aplicación del mismo fabricante que te permite diseñar y modelar amplificadores. Incluye réplicas de los 36 amplificadores más “deseados” de la historia. En ellos se pueden realizar ajustes en profundidad. Por ejemplo, es posible intercambiar válvulas, preamplificadores, transformadores, *tone stacks* (filtros), micrófonos, etc.; todo ello gracias a técnicas de modelado físico, convolutivas, etc. Todo esto se traduce en una aplicación de gran calidad y flexibilidad, que puede ser utilizada para cualquier estilo de música, a precio muy bajo.

3 Especificaciones y restricciones de diseño

3.1 Descripción funcional del dispositivo original

El Behringer FCB1010 es un controlador MIDI en forma de pedalera. Permite al usuario tener 10 bancos, cada uno con 10 *presets* editables. Puede transmitir en el mismo *preset* 5 comandos *Program Change* (Cambio de Programa) y 2 comandos *Control Change* (Cambio de Control). También envía comandos *Note-On* para usar funcionalidades como el *Tap-Tempo* (ajuste de tempo).

En la Figura 2 se muestran dos vistas del controlador donde vienen indicadas sus distintas partes, que se explican en los siguientes párrafos.

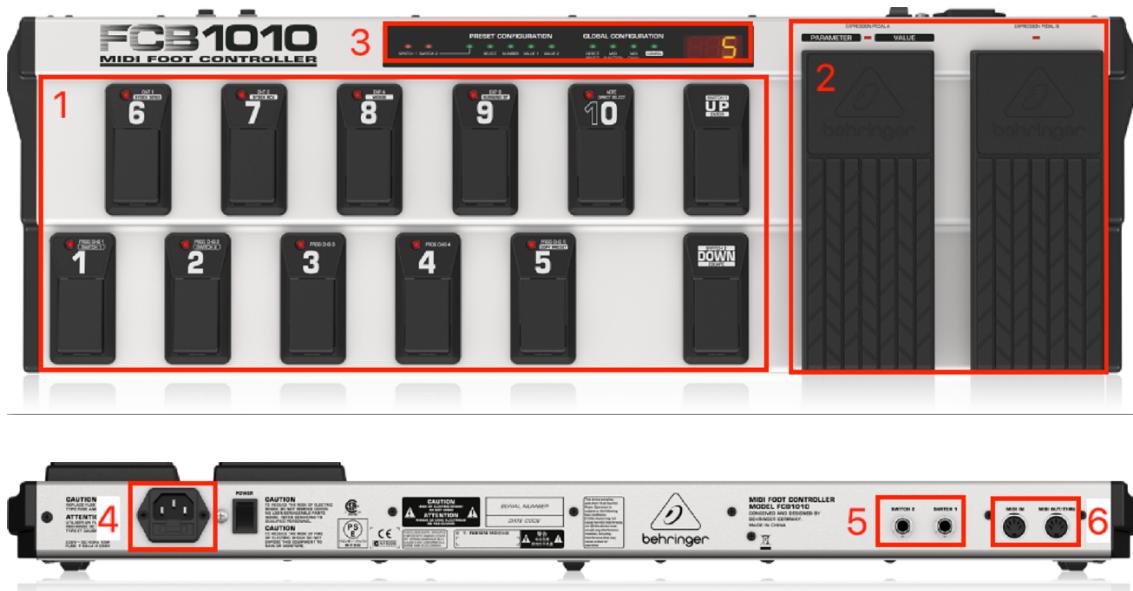


Figura 2. Controlador MIDI Behringer FCB1010. Imagen superior: vista frontal; Imagen inferior: vista posterior.

1. 12 pulsadores. Los pulsadores del uno al diez se emplean para cambiar los *presets*, programar la pedalera, introducir valores y activar la función de selección directa de *preset*. Cada uno lleva asociado un LED. Los pulsadores UP (arriba) y DOWN (abajo) se utilizan para navegar entre los diferentes bancos y niveles de programación.
2. 2 pedales de expresión, con los que se puede realizar una modificación continua de los valores.
3. Indicador con LEDs y *display*. Los LEDs sirven para dar indicaciones cuando se programa la pedalera y el *display* de 7 segmentos informa sobre el *preset* y banco cargado.

4. Fuente de alimentación integrada. Tiene un conector hembra de tres espigas IEC (*International Electrotechnical Commission* o “Comisión Electrotécnica Internacional”) que se conecta a la red. Es decir, en Europa funcionará con una tensión de 220 voltios y a una frecuencia de 50 hercios.
5. 2 líneas de conexión con desacoplo galvánico programables para, por ejemplo, realizar la conmutación entre canales de un amplificador de guitarra. Los conectores son *jacks* de 6,35mm
6. Conectores MIDI *in* (entrada MIDI) y MIDI *out/thru* (salida MIDI). Al MIDI *in* se puede conectar un secuenciador o dispositivo similar para recargar contenido de la memoria previamente. En la toma MIDI *out/thru* se conectan los equipos que se quieren controlar con la pedalera. Además, mediante la función MIDI *merge* (mezcla MIDI) con *Soft Through* (suavizado) se pueden aportar señales del MIDI *in* y mezclarlas con el flujo de datos.

Tal y como destacan la mayoría de sus usuarios, la configuración por defecto es algo tediosa. A continuación, se detalla el procedimiento que se debería realizar para configurar un comando sencillo: cambio al *preset* 23 en el canal 7 cuando se pulsa el pulsador 1.

Para configurar el canal MIDI por el que se envía el comando se debe ir a la configuración global del dispositivo. Esto se hace presionando *DOWN* mientras que se enciende la controladora. Se encienden entonces los LEDs de *Global Configuration* (Configuración Global) y entonces:

1. Se presiona el pulsador *UP* y la luz indicadora verde cambia a *Function Selection* (Selección de Función).
2. Se presiona el pulsador 1 para que se muestre en el *display* de 7 segmentos el canal MIDI actual para el comando *Program Change* (cambio de programa). En la Tabla 1 se muestran los comandos que envían los distintos pulsadores.

Tabla 1. Asignación de funciones MIDI a los pulsadores. [5]

Tecla	Función Midi
1	PRG CHG 1
2	PRG CHG 2
3	PRG CHG 3
4	PRG CHG 4
5	PRG CHG 5
6	CNT 1
7	CNT 2
8	EXP A
9	EXP B
10/0	NOTE

3. Se presiona *UP* y la luz indicadora verde cambia a MIDI Chann (canal MIDI) y el *display* de 7 segmentos empieza a parpadear.
4. Selecciona el canal MIDI en dos dígitos, es decir, si quieres que el canal sea el 7 pulsa el 0 y después el 7. El número se mostrará en el *display* de 7 segmentos. Después presiona *UP* y la luz verde indicadora cambiará a MIDI *Config Selection* (selección de configuración MIDI).
5. Por último, se mantiene el pulsador *DOWN* cinco segundos y el *display* de 7 segmentos mostrará “00”.

Para programar que un pulsador envíe un comando concreto es necesario realizar lo siguiente:

1. Se presiona el pulsador al que se quiere asociar el comando. En el ejemplo planteado, el pulsador uno.
2. Se mantiene pulsado el pulsador *DOWN* durante cinco segundos. Esto hace que la pedalera entre en modo programación del pulsador.
3. Se presiona el pulsador *UP*.
4. Si el LED del pulsador que se quiere seleccionar no está encendido, se mantiene el pulsador hasta que se encienda el led.
5. Se presiona el pulsador de nuevo. Esta vez empieza a parpadear. Entones se presiona *UP*. El *display* de 7 segmentos comienza entonces a parpadear mostrando el comando actual (si son los pulsadores del 1 al 5 será *Program Change*).
6. Se introduce el valor que se quiere enviar en el comando mediante los pulsadores. En el ejemplo planteado, para el *Program Change* 23, se pulsará el pulsador 2 y después el 3.
7. Finalmente se presiona *UP* y se mantiene el pulsador *DOWN* durante 5 segundos.

Este procedimiento se debe realizar para todos los comandos que se quieran enviar con el controlador. De forma similar, también se tendrán que configurar los pedales de expresión. No resulta sorprendente que esto sea complejo para los usuarios. Además, son muchos los que se quejan de no encontrar información clara sobre la configuración, ya que el manual no es de fácil comprensión.

3.2 Descripción técnica del dispositivo original

Siguiendo el blog *Practical Usage*, donde es posible encontrar uno de los proyectos de modificación de la pedalera Behringer FCB1010 más conocidos, se va a realizar una descripción técnica del dispositivo original. Se adjuntan las fotos que aparecen en este mismo blog. [6]

Nota: en esta explicación se mencionan los circuitos integrados (IC – *Integrated Circuit*) conforme están inscritos en las placas en las que se encuentran, numerados del número 1 al 14. En la mayoría se añade el modelo de IC separado por 2 puntos.

Por ejemplo, el circuito integrado número 5 cuyo modelo es 74HC130 figura como IC5: 74HC130.

Al quitar la tapa trasera del aparato se encuentra lo que se observa en la Figura 3.



Figura 3. Vista posterior del controlador sin la cubierta. [6]

Como se puede ver, hay diferentes placas o circuitos impresos:

- Pulsadores y LEDs: son las dos placas paralelas alargadas que aparecen en horizontal ocupando casi todo el espacio. Hay una placa por cada fila de pulsadores. En cada una hay un conector de 14 cables. De estos, 5 son para LEDs, 6 para pulsadores, uno para tierra (GND) y otro para el colector común de voltaje (VCC). En la placa superior solo están en funcionamiento 13, pero en la inferior hay 2 cables de VCC (5V), por lo que se completan de esta manera los catorce pines.
- Pedales de expresión: hay dos pequeñas placas, cada una asociada a un pedal de expresión. Cuentan con un fototransistor y un plástico en escala de grises que se mueve cuando lo hace el pedal. Esto hace que se genere un voltaje variable que se envía a un convertidor analógico-digital en la placa principal. En la Figura 4 se puede observar dónde se encuentran estas dos placas y cómo son en detalle.



Figura 4. Placas de los pedales de expresión. Izq: lugar en el que se encuentran; Dcha: placa en detalle. [6]

- Conectores MIDI: contiene un optoacoplador (IC1: H11L1) requerido por el estándar MIDI para proteger la electrónica interna de la externa que se conecta mediante el conector DIN de 5 pines. En la Figura 5 podemos observar dónde se encuentra esta placa y cómo es en detalle.

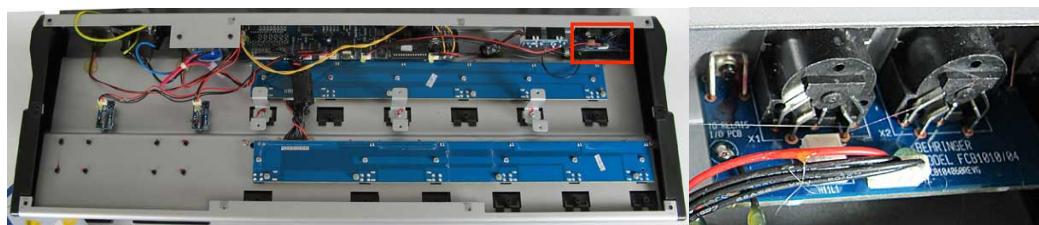


Figura 5. Placa de conectores MIDI. Izq: lugar en el que se encuentran; Dcha: placa en detalle. [6]

- Conectores de conmutación (*Switch*): en la Figura 6 se muestra dónde se encuentran y cómo están insertados.



Figura 6. Placa de conectores Switch. Izq: posición en la que se encuentran; Dcha: placa en detalle. [6]

- Alimentación: cuenta con un interruptor, un conector y un transformador que convierte la corriente alterna de la red en 9 voltios de corriente continua. Posteriormente, en la placa principal, hay un regulador de 5 voltios, ya que esta funciona con tecnología TTL (*transistor-transistor logic*). En la Figura 7 se puede observar dónde se encuentran las distintas partes.

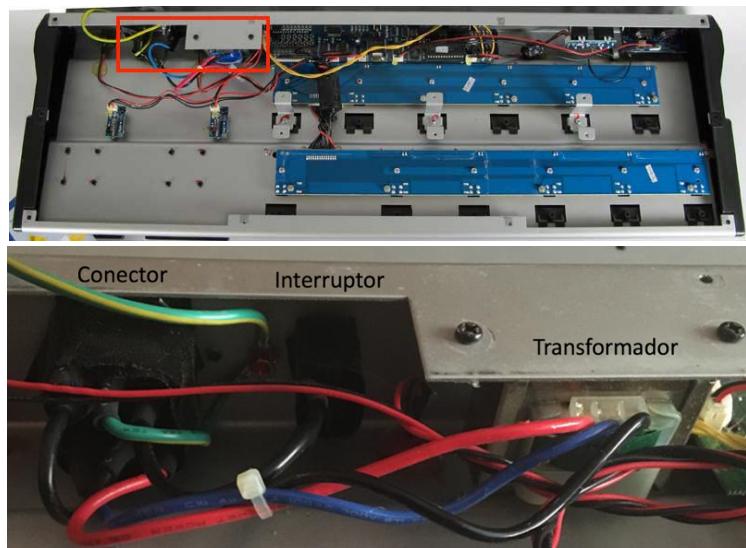


Figura 7. Conector, interruptor y transformador. Imagen superior: posición en la que se encuentran. [6]

Imagen inferior: detalle.

- Placa principal: de forma resumida consta de un microcontrolador con RAM (*Random Access Memory* o “Memoria de Acceso Aleatorio”), una memoria que contiene el *firmware* (programa informático que establece la

lógica a bajo nivel del dispositivo), algunos IC para controlar el *display* de 7 segmentos y otro IC para recibir los datos de los pedales. En la Figura 8 se puede observar la placa principal con sus distintos circuitos integrados.



Figura 8. Vista frontal de la placa principal. [6]

Como se puede observar en la Figura 9, en la parte izquierda de la placa principal están los *arrays* de transistores y resistencias para el funcionamiento del *display* de 7 segmentos.

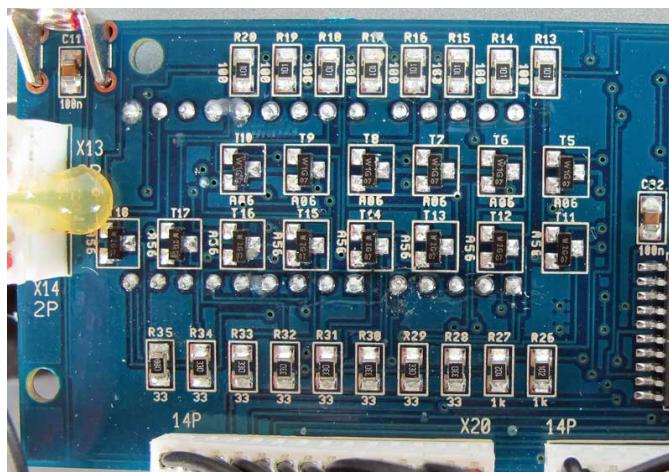


Figura 9. Arrays de transistores y resistencias de la placa principal. [6]

En la parte central se encuentran los siguientes circuitos integrados (IC):

- IC10, IC3, IC11: 74HC273D: flip-flop octal de tipo D. Dos de ellos se usan para controlar los dígitos del *display* de 7 segmentos y el otro se usa para recibir las entradas de los pedales.
- IC12: C0832C: convertidor analógico-digital de 8 bits para convertir la señal del pedal a digital.
- IC2: T805: regulador de voltaje a cinco voltios.
- IC7: TL7705ACD: supervisor de alimentación para el chip principal
- IC6: Atmel62- 24C16A: módulo de memoria EEPROM (*Electrically Erasable Programmable Read-Only Memory* o Memoria Programable de Solo Lectura Borrable Eléctricamente) de 16K.
- IC8: Philips P80C32SBAA: microcontrolador.
- IC4: 74HC04: inversor hexadecimal.
- IC5: 74HC130: multiplexor para los circuitos integrados 3, 10 y 11 (los flip-flops octales de tipo D).
- IC9: 74HC373: *latches* (*late memory* o puerta lógica *RESET*) de tipo D.
- IC14: M5M5256FP-85: RAM de 32K x 8bit

- IC13: 27C256: EEPROM de 256K. Es el circuito integrado de mayor tamaño y se encuentra a la derecha de la placa. Se puede observar fácilmente en la Figura 10. Se ha ido actualizando a lo largo de la vida comercial del controlador, por lo que dos dispositivos pueden tener versiones de la memoria diferentes. Esta memoria es la que cambian algunos usuarios para conseguir mejoras.



Figura 10. Parte derecha de la placa principal del dispositivo. [6]

Parte de estos circuitos integrados se pueden ver en la Figura 11, que se muestra a continuación.

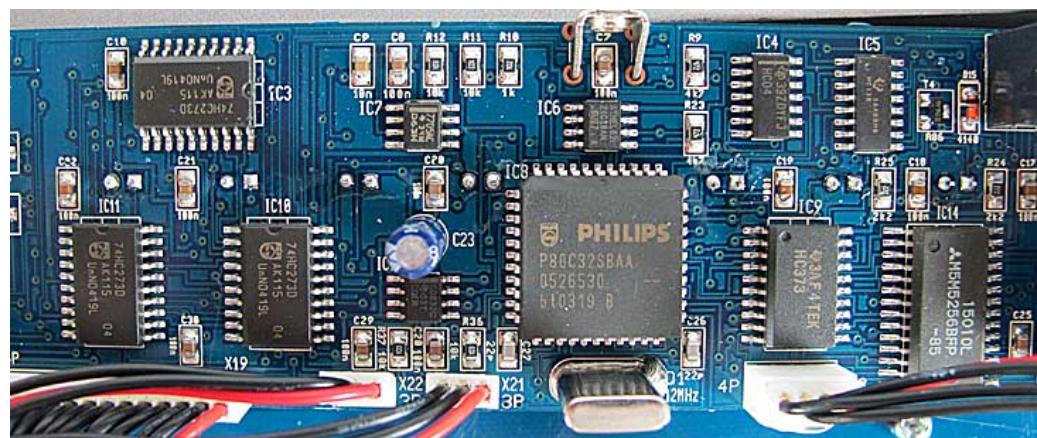


Figura 11. Algunos de los circuitos integrados de la placa principal en detalle. [6]

Por último, como se observa en la Figura 12, en la parte posterior de la placa principal encontramos los *displays* de 7 segmentos y los distintos LEDs de configuración que se muestran en la parte frontal del dispositivo.

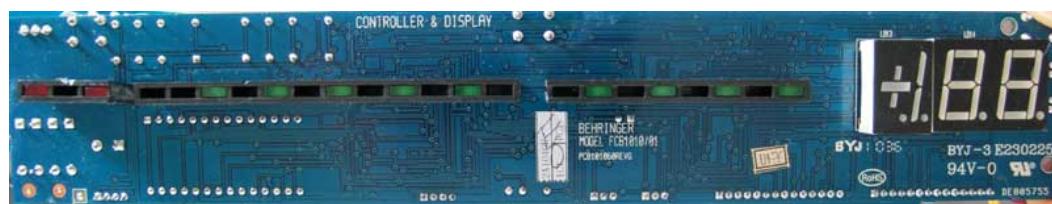


Figura 12. Vista posterior de la placa principal del dispositivo. [6]

3.3 Especificaciones del nuevo controlador

El Behringer FCB1010 es un controlador MIDI en forma de pedalera que destaca por su robustez. Mide 687x22x60 milímetros y su peso sobrepasa los cuatro kilogramos (4,34 kg). Esto, unido a su bajo precio (apenas 120 euros), hace que este dispositivo sea uno de los más vendidos y también, más modificados en proyectos DIY.

Como se comenta en el apartado anterior, se hace una revisión de los proyectos desarrollados y de las necesidades que los usuarios comparten una y otra vez en foros, se establece que el controlador a desarrollar debe cumplir al menos los siguientes requisitos:

- Pantalla LCD para una mejor comunicación entre máquina y usuario.
- Asignación de un efecto a cada pulsador para poder interactuar con los distintos efectos como si de pedales analógicos se tratases.
- Posibilidad de controlar otras funciones del procesador. Por ejemplo, un *looper*.
- Conexión USB.
- Simplificación de la configuración

Por otro lado, se decide enfocar este proyecto al uso del controlador con la aplicación *JamUp*, tanto con la versión gratuita como con la de pago. No se centra, como los proyectos previos, en una unidad multifunciones. Como se explicaba en el apartado anterior, estos dispositivos están cayendo en desuso frente al auge de los *softwares* de propósito similar.

El nuevo dispositivo está adaptado para que su uso con la aplicación resulte intuitivo y sencillo. Se ha buscado que la experiencia de uso sea lo más parecida a la de una pedalera multifunciones. Además, se puede conectar a *JamUp* mediante *Bluetooth* y enviar los comandos MIDI tanto por *Bluetooth* como por cable.

Sin embargo, el uso de la pedalera no se limita a la aplicación. Tiene también un modo libre en el que se pueden enviar hasta 25 comandos diferentes por medio de diferentes tipos de pulsaciones y 2 comandos con los pedales de expresión. La configuración de ambos modos es sencilla y se realiza mediante el envío de un JSON (*JavaScript Object Notation*, “Notación de Objeto de JavaScript”) por *Bluetooth*.

Además, se ha decidido descentralizar la electrónica de manera tal que se trata de un sistema distribuido compuesto de módulos conectados por I2C. Esto tiene muchas ventajas, entre ellas que se pueden quitar o añadir nuevos módulos al gusto, que una avería en un módulo no impide que la máquina siga funcionando y que es fácilmente reparable.

Como lo atractivo del proyecto es que sea de utilidad para la comunidad de usuarios de la pedalera, el código comparte en Github (<https://github.com/elenieto/fcb1010>). Este es, por tanto, un proyecto *Open Source* o “de código abierto”. Siguiendo esta idea, una de las consideraciones que se tiene en cuenta es que se pueda realizar desde casa con materiales fácilmente obtenibles y económicos. Por esta razón, cada uno de los módulos del sistema distribuido tiene un microcontrolador Attiny85 (se pueden conseguir por 1 euro aproximadamente) excepto el módulo principal, que contará con un microcontrolador ESP32, cuyo precio es aproximadamente 3 euros. Observe que se trata de una red de microcontroladores operando de manera independiente y asíncrona.

Para programarlos se utiliza el *software* Arduino, que es gratuito y ya de sobra conocido para el colectivo que realiza proyectos DYI. Se espera que esto motive a los usuarios a realizar el proyecto, ya que siendo conocedores de la herramienta podrán cambiar el código original a su gusto para adaptarlo a sus necesidades.

3.4 Metodología

El proyecto sigue estructura distribuida, basada en el paradigma de los sistemas complejos. Cada módulo realiza tareas simples que, en conjunto, permite modelar comportamientos complejos. Como expone Octavio Miramontes, “el proceso de interacciones puede generar comportamientos colectivos y globales. Es decir, conductas que no están definidas en los elementos individuales pero que emergen como un proceso colectivo”. [7]

La metodología sigue esta misma filosofía. El abordaje de los módulos por separado de forma independiente facilita la localización y el aislamiento de problemas para su posterior resolución. Además, que surja un bloqueo en un módulo (por ejemplo, porque se necesite cierto material) no supone la paralización total del proyecto. Es decir, el diseño y construcción de los distintos módulos puede avanzar de manera paralela.

Para el planteamiento inicial del proyecto, se establece la funcionalidad del dispositivo completo. Posteriormente, se fracciona en módulos y se especifica cuál va a ser el funcionamiento de cada uno de ellos y qué datos necesitan recibir y transmitir.

Se concretan las características físicas de cada módulo. Por ejemplo, para un pulsador con LED se necesita conectar un pulsador y un LED al microcontrolador Attiny85. Se determinan las resistencias y otros componentes necesarios para realizar la conexión. Se realiza el montaje en una placa de pruebas de manera totalmente aislada al dispositivo. Es decir, aunque finalmente se vaya a utilizar el

LED del controlador original, el montaje se realiza con un componente LED independiente.

Se realiza el código de cada módulo sobre la instalación en la placa de pruebas. Además de resultar más cómodo a la hora de quitar y poner el controlador para programarlo con el ordenador, esto asegura que el montaje que se instale en el dispositivo funcione. A continuación, en la Figura 13, se muestra una fotografía del montaje de prueba realizado para el desarrollo de las placas.

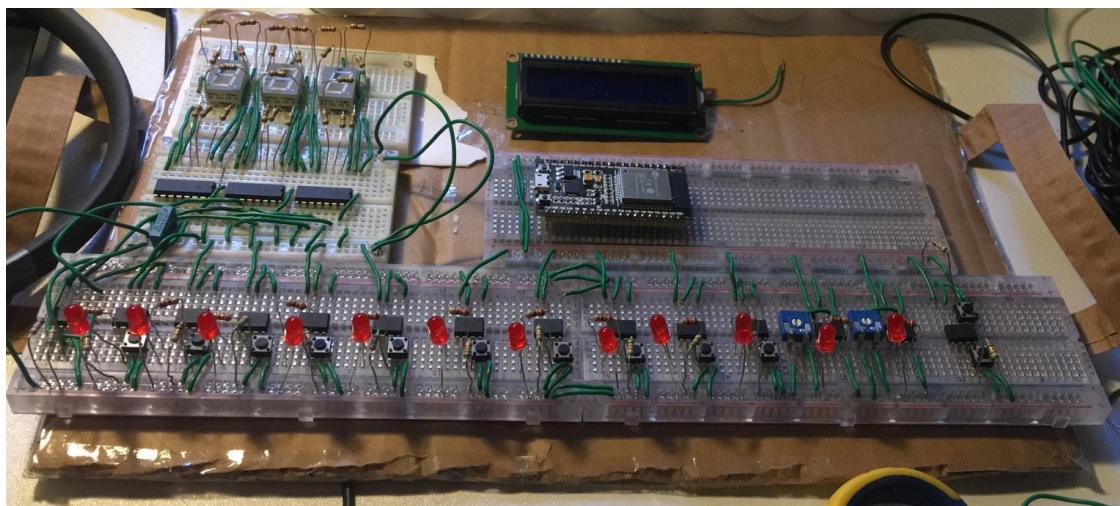


Figura 13. Montaje de prueba realizado para el desarrollo de los distintos módulos.

Se realizan pruebas de comunicación entre los distintos módulos. Por ejemplo, es de gran importancia realizar ensayos de comunicación entre los módulos de los pulsadores y el Módulo Principal.

Para realizar pruebas de conexión entre más de 3 módulos fue necesario realizar la instalación en el dispositivo para un buen funcionamiento del protocolo I2C. Conforme se van instalando los módulos se prueban, para certificar, entre otras cosas, que los componentes que se van a aprovechar del dispositivo original (por ejemplo, LEDs y pulsadores), funcionan correctamente con los elementos introducidos (por ejemplo, resistencias).

Finalmente, se pone en funcionamiento el bus de comunicación I2C todos los módulos. Se comprueba el correcto funcionamiento de todos y se completa el programa principal en el microcontrolador ESP32.

4 Descripción de la solución propuesta

4.1 Descripción general de la propuesta

El dispositivo se plantea como un sistema distribuido compuesto de distintos módulos conectados mediante I2C (*Inter Integrated Circuit* o “Circuito Inter-Integrado”); un estándar de comunicación rápido y simple, ampliamente utilizado en el diseño de sistemas electrónicos.

Este protocolo permite intercambiar información entre microcontroladores mediante dos cables: SDA (*Serial Data* o “datos”) y SCL (*Serial Clock* o “reloj”). Se realiza entonces una comunicación serial y síncrona: la información se envía bit por bit sincronizada por una señal de reloj a través de una sola vía. La velocidad máxima de transmisión en modo estándar es de 100 kilobits por segundo.

En la comunicación, los dispositivos pueden ser “maestros” o “esclavos”. Mientras que el número de maestros es ilimitado, como máximo puede haber 1008 esclavos. Cada mensaje se divide en tramas de datos como las que se observan en la Figura 14.

Condición de inicio	Dirección (7 a 10 bits)	Lectura/Escritura	ACK/NACK	Información (sección 1)	ACK/NACK	Información (sección 2)	ACK/NACK	Condición de paro
---------------------	-------------------------	-------------------	----------	-------------------------	----------	-------------------------	----------	-------------------

Figura 14. Tramas de datos de un mensaje del protocolo I2C.

Se diferencia en esta trama:

- Condición de inicio: SDA cambia de nivel de voltaje alto a bajo y deja SCL a nivel alto.
- Dirección: 7 a 10 bits con la dirección del esclavo con el que se quiere comunicar el maestro.
- Bit de lectura o escritura: si es nivel bajo el maestro indica que va a enviar información al esclavo. En cambio, si es alto, estará solicitándosela.
- Información (sección 1): 8 bits de los que se envía primero el más significativo.
- ACK/NACK: se devuelve al remitente un bit ACK (*acknowledgement* o “acuse de recibo”) si la información fue recibida con éxito y un bit NACK (*negative acknowledgement* o “acuse de recibo negativo”) en caso contrario.
- Información (sección 2): mismo funcionamiento que para la sección 1.
- ACK/NACK: mismo funcionamiento que el ACK/NACK anterior.
- Condición de Paro: cuando se envía toda la información, el maestro puede detener la transmisión mediante un cambio de nivel bajo a nivel alto en SDA cuando SCL pasa de nivel bajo a alto.

El funcionamiento del protocolo es el siguiente:

1. El maestro envía la condición de inicio a cada esclavo. Esto significa que cambia el nivel de SDA a bajo y deja SCL en estado alto.
2. El maestro envía la dirección del esclavo al que está dirigida la información.
3. Todos los esclavos comparan su dirección con la recibida. Si coincide, el esclavo en cuestión envía un ACK de nivel bajo al maestro y deja SDA a nivel bajo. Los demás, cuya dirección es no coincidente, dejan SDA a nivel alto.
4. El maestro envía un bit para indicar si quiere enviar información (nivel bajo) o solicitar información (nivel alto).
5. El maestro o el esclavo envía la primera sección de información (8 bits). Inmediatamente después se envía un ACK/NACK para comprobar si los datos se han transmitido correctamente.
6. Se envía la siguiente sección de información y posteriormente su ACK/NACK correspondiente.
7. Una vez enviadas las dos secciones el maestro puede activar la condición de paro, es decir, cambia SDA a nivel alto cuando SCL cambia a nivel alto.

Se elige este protocolo de comunicación por ser sencillo y efectivo. Con solo dos cables soporta múltiples maestros y esclavos. Además, tiene confirmación de información recibida mediante los bits de ACK/NACK. Se descartan otras opciones. Entre ellas el protocolo SPI (*Serial Peripheral Interface* o “interfaz periférica serial”) que es más rápido, pero utiliza cuatro cables y no tiene confirmación. También el UART (*Universal Asynchronous Receiver-Transmitter* o “transmisor-receptor asíncrono universal”), cuya velocidad es baja y no puede soportar múltiples esclavos y maestros.

La placa principal recibe suministro mediante un alimentador que convierte la corriente alterna de la red a corriente continua y suministra al ESP32 un voltaje de cinco voltios y una corriente de 2 amperios. Por tanto, para alimentar los distintos módulos, además de los dos cables del protocolo I2C (SDA y SCL), hay otros dos cables que tienen que conectar los módulos: VCC y GND. Se utilizan conectores RJ11 (que tienen cuatro pines) y cable telefónico (que lleva cuatro vías en su interior) como se muestra en la Figura 15, por su amplia disponibilidad en el mercado y facilidad de uso. Se podría considerar como aportar un nuevo uso a una tecnología popular. De esta manera, los módulos se pueden manipular de manera independiente y conectarse y desconectarse entre sí muy fácilmente. Además, queda el interior del dispositivo más ordenado y el riesgo de desconexión de algún cable es mucho menor.



Figura 15. Izq: cable telefónico (cuatro vías); Dcha: conector RJ11 con el cable telefónico insertado.

Mediante la interconexión de los distintos módulos se establece el bus de comunicación I2C, del que se muestra un ejemplo en la Figura 16. Cabe destacar, que se denomina así porque, aunque físicamente es una cadena, lógicamente es un bus.

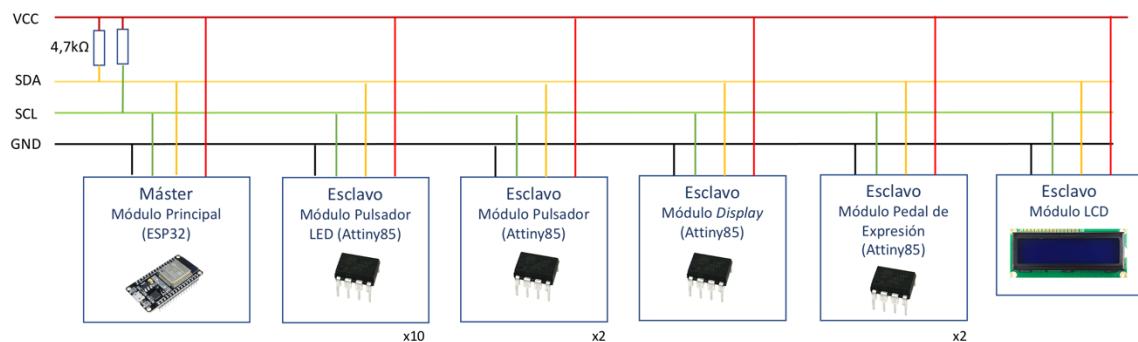


Figura 16. Esquema del bus I2C.

4.1.1 Funcionamiento del dispositivo

El dispositivo cuenta con 3 modos de funcionamiento orientados a la aplicación JamUp (*Preset*, *Jam* y *Sampler*) y un modo libre. Por defecto, el controlador MIDI se inicia en Modo *Preset*, el modo básico de funcionamiento de la aplicación. De hecho, al acceder al Modo *Jam* o *Sampler* se añaden funciones, pero las de Modo *Preset* se mantienen. Para acceder a los distintos modos de funcionamiento se utiliza una pulsación larga en los pulsadores con LED de la fila superior. Los modos están asociados a los pulsadores como se observa en la Figura 17.

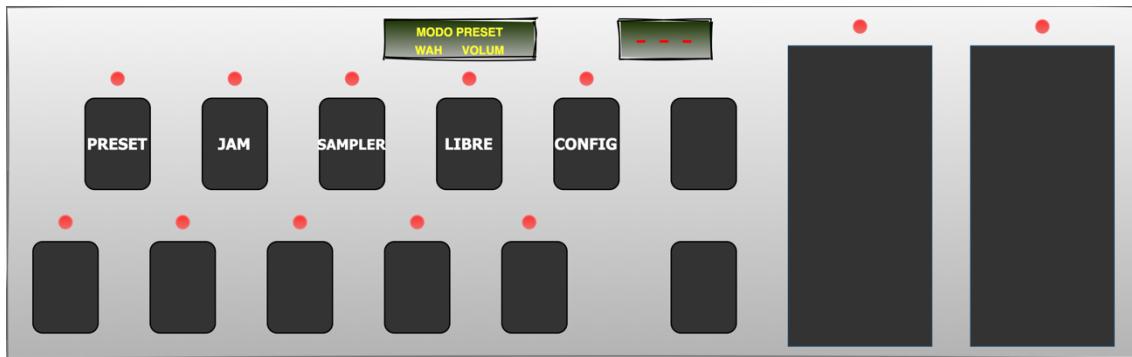


Figura 17. Pulsadores del cambio de modo mediante pulsación larga.

El menú de configuración se dispone de seis opciones: LLD (carga del estado de los LEDs), SLD (salvado del estado de los LEDs), CC (carga de comandos CC), ML (configuración del modo libre), EXP (calibración de pedales de expresión) y DSP (añadir dispositivo a bus I2C).

Puesto que el uso del dispositivo está orientado a la aplicación JamUp, a la hora de realizar el diseño, se tiene en cuenta “las órdenes” que esta aplicación puede recibir. A continuación, se muestra un listado de las distintas acciones a las que se puede asignar un comando MIDI en la aplicación:

- Efectos:
 - AMP: apagar/encender amplificador.
 - NG: apagar/encender puerta de ruido.
 - STOMP: apagar/encender *stomp*.
 - DLY: apagar/encender *delay* (retardo).
 - MOD: apagar/encender modulación.
 - RVB: apagar/encender *reverb* (reverberación).
 - FILT: apagar/encender efecto de filtro.
 - FX: apagar/encender efecto seleccionado.
- Preset:
 - PRESET +: cambiar al siguiente *preset*.
 - PRESET -: cambiar al anterior *preset*.
 - PAGE -: cambiar a la página anterior.
 - PAGE +: cambiar a la página siguiente.

- PRESET A: cambiar a la página de *preset A*.
- PRESET B: cambiar a la página de *preset B*.
- PRESET C: cambiar a la página de *preset C*.
- PRESET D: cambiar a la página de *preset D*.
- Pedales de expresión
 - Volumen: establecer el volumen de salida.
 - *Cry Wah*: cambiar la profundidad del *cry-wah* (filtro paso banda de frecuencia central variable).
 - *Pitch shifter*: modificar la altura de la nota.
- Utilidades
 - METRONOME: apagar/encender metrónomo.
 - TAP TEMPO: hacer *tap* (pulsaciones) para establecer el tempo del metrónomo.
 - TUNER: apagar/encender afinador.
- *Jam*:
 - PLAY/STOP: parar y continuar.
 - SPEED +: aumentar la velocidad de la pista de fondo.
 - SPEED -: disminuir la velocidad de la pista de fondo.
 - PITCH +: aumentar el tono de la pista de fondo.
 - PITCH -: disminuir el tono de la pista de fondo.
 - VOLUME +: incrementar el volumen de la pista de fondo.
 - VOLUME -: disminuir el volumen de la pista de fondo.
- *Sampler*
 - REC: empezar/parar grabación.
 - UNDO: rehacer el último bucle.
 - PLAY: reproducir/parar.
 - DUB: empezar/parar bucle.

Todos estos comandos se envían por el canal MIDI número 1, por lo que hay que configurar ese canal en los ajustes de la aplicación. En el manual de usuario que figura en el anexo se detalla paso a paso cómo se realiza esta configuración.

Durante Modo *Preset* se envían los comandos de pedales de expresión, efectos, *preset* y utilidades. En Modo *Jam*, además de los disponibles en el Modo *Preset*, se añaden los comandos *jam*. De igual manera, en Modo *Sampler*, deben funcionar los de Modo *Preset* y los comandos *sampler*.

Se establecen tres tipos de pulsaciones: corta, larga y doble. Se prioriza asignar la pulsación corta a las acciones que requieren inmediatez o que se utilizan más. Se considera indispensable la función *stompbox*, es decir, que a cada pedal se pueda asignar un efecto para activar y desactivar este con una pulsación corta. A

continuación, en la Figura 18, se presenta un esquema de cómo se envían los comandos de la sección de efectos.

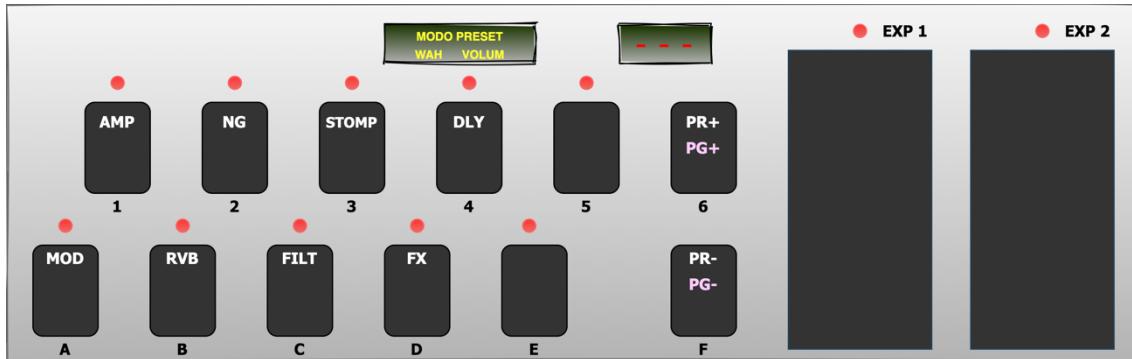


Figura 18. Pulsadores de la funcionalidad stompbox.

Además, se renombran los pedales para la gestión de bancos y *presets*. En la aplicación se numeran mediante un número y una letra. El número va del uno al cuatro y la letra de la A a la D. Por esta razón, los seis pedales de la fila superior serán números del uno al seis y los seis pedales de la fila inferior serán letras de la A a la F.

Para cambiar de *presets* se realiza una pulsación corta en los pulsadores sin led, el 6 (PRESET+) y el F (PRESET-). Para cambiar de página se realiza una pulsación larga en estos mismos pulsadores. Para cambiar entre los *preset* A, B, C o D del mismo número se utilizan los pulsadores A, B, C y D.

En la Figura 19 se muestra un esquema con la numeración (en negro) y comandos explicados hasta el momento. En blanco, las pulsaciones cortas. En rosa, las pulsaciones largas. Las pulsaciones dobles, cuando las haya, se indican en azul.

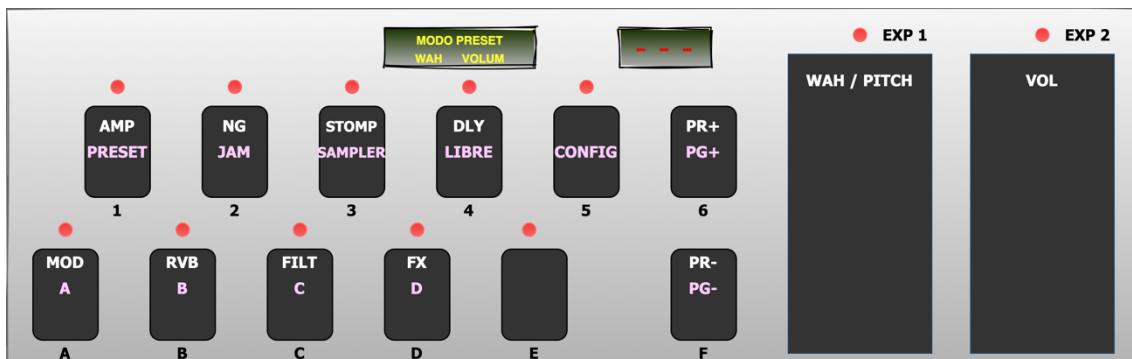


Figura 19. Esquema con la numeración de pulsadores, los comandos de efectos y cambio de preset.

Pedales de expresión

En la aplicación JamUp hay disponibles 3 funcionalidades para los pedales de expresión: Volumen, *Wah* y *Pitch shifter*. La pedalera solo dispone de 2 pedales de expresión, por lo que uno tiene que albergar las 2 funciones y conmutar cuando se pulse un pulsador de cierta manera. Se decide que, por defecto, los pedales de expresión estén asignados a volumen (el de la derecha) y al efecto *wah* (el de la izquierda). Cuando se efectúa una pulsación doble en el pulsador 6 el pedal de volumen pasa a albergar el efecto *Pitch shifter*.

Utilidades

Las utilidades se controlan en el pulsador E. Con una pulsación corta se activa el *tap*, con una pulsación larga se activa/desactiva el afinador y con una doble activa/desactiva el metrónomo.

A continuación, en la Figura 20, se recopilan todos los comandos del Modo *Preset*.

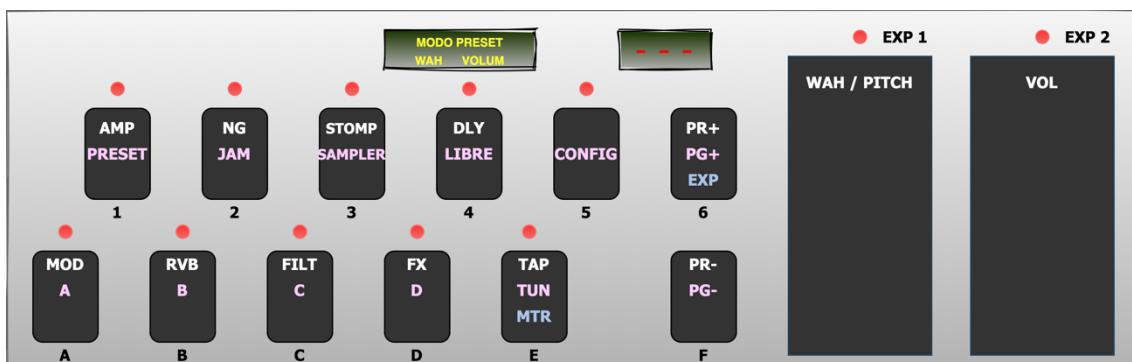


Figura 20. Pulsaciones asociadas a comandos del Modo Preset.

Jam

En Modo *Jam* sirve para controlar la función *Jam* de la aplicación JamUp, que permite tocar encima de una pista de fondo a la que, mediante comandos MIDI CC, se le puede cambiar la velocidad, la altura y el volumen. Para cada una de estas cualidades del sonido modificables hay asociados 2 comandos: uno para aumentar y otro para disminuir. Por esta razón, para este modo se organizan los pulsadores en columnas. De cada columna, la fila de arriba sirve para aumentar y la de abajo para disminuir. De esta manera, el pulsador 1 sirve para aumentar velocidad y el pulsador A para disminuir la velocidad. En la columna de al lado, el pulsador 2 sirve para aumentar la altura y el pulsador B para disminuirla. Por último, el pulsador 3 sirve para aumentar volumen y el pulsador C para disminuirlo. Para todos estos nuevos comandos asociados a este modo se utilizan pulsaciones dobles, ya que a su vez está en funcionamiento todo lo detallado anteriormente. A continuación, en la Figura 21, se representan todas las funciones disponibles en Modo *Jam* asociadas a su tipo de pulsación.

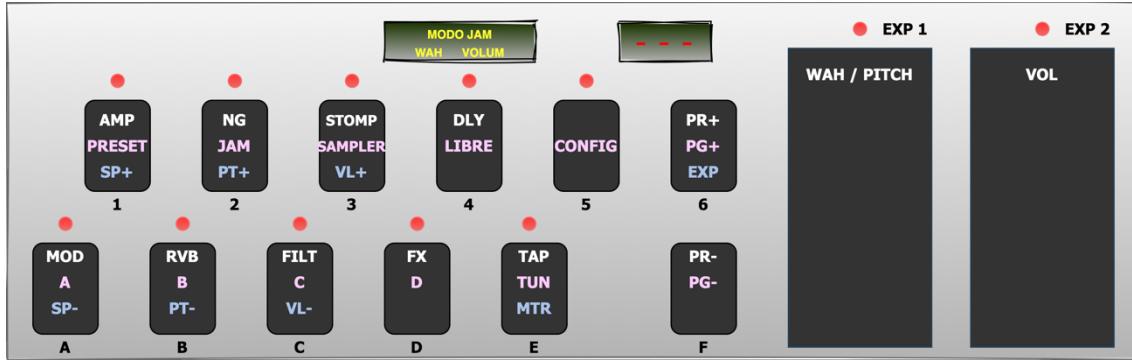


Figura 21. Pulsaciones asociadas a comandos del Modo Jam.

Sampler

La función *Sampler* de la aplicación JamUp es un *looper* (“bucleador”). Es decir, nos permite realizar grabaciones y reproducirlas en bucle, añadiendo capas. Esto es muy útil para fabricar una base sobre la que improvisar. Además, se mantienen la funcionalidad completa de Modo *Preset*. Por esta razón, a la hora de asignar pulsadores a este modo, surge una dificultad: se necesita bastante rapidez al enviar el comando para realizar los bucles correctamente y ya se tienen las pulsaciones cortas de casi todos los pulsadores asignadas.

Se realiza un análisis y la única función que necesita realmente esta rapidez es el *dub* (empezar o parar bucle). Así que es a este comando al que se le asigna la única pulsación corta que quedaba por asignar, la del pulsador 5. Para los demás comandos se utilizan las pulsaciones dobles de este mismo pulsador, el 5, (*undo* o deshacer), del pulsador 4 (*play* o reproducir) y del D (*rec* o grabación). En la Figura 22 se representan todas las funciones disponibles en Modo *Jam* asociadas a su tipo de pulsación.

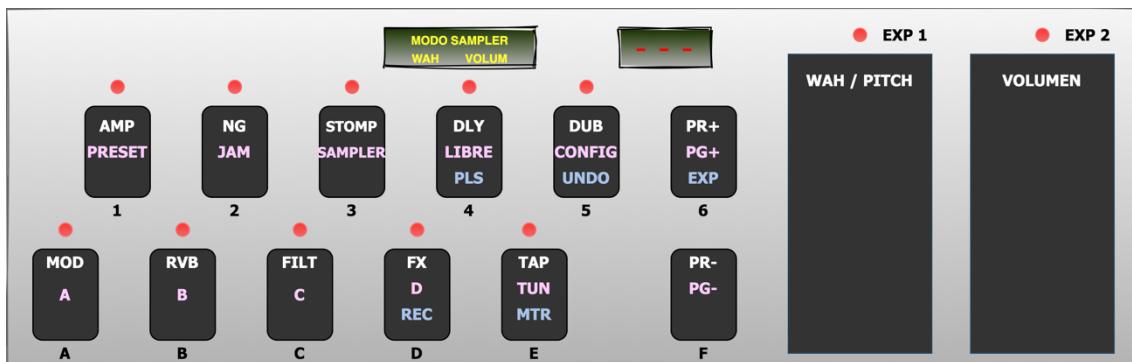


Figura 22. Pulsaciones asociadas a comandos del Modo Sampler.

Conociendo la dificultad que puede suponer memorizar todos estos controles para los usuarios primerizos, se añaden etiquetas a los pulsadores y pedales de expresión del dispositivo.

Cada pulsador contiene las 3 acciones asociadas a los distintos tipos de pulsaciones. El texto verde para pulsación corta, el rojo para pulsación larga y el azul para pulsación doble. Además, encima de los pulsadores se añaden sus nuevos nombres. En la Figura 23 se puede observar el resultado final.

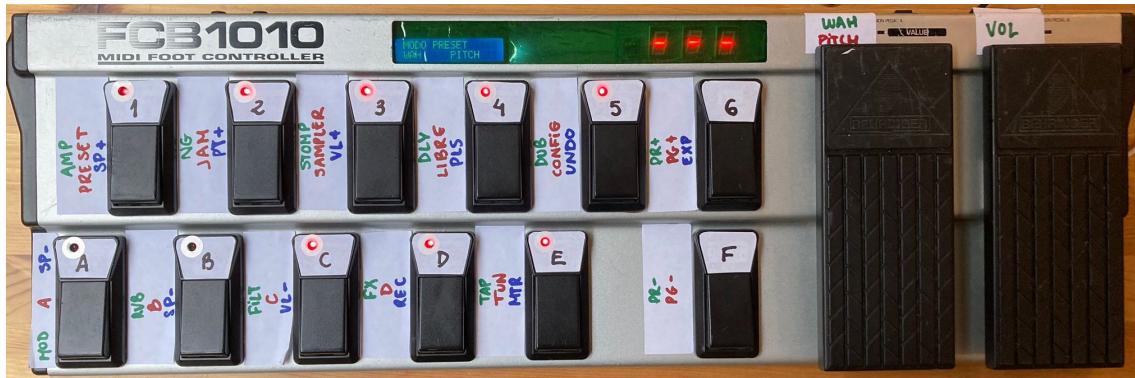


Figura 23. Apariencia del nuevo controlador MIDI.

Por otro lado, para el correcto funcionamiento de la pedalera, se tienen que realizar una serie de ajustes. Cuando accedes al menú de configuración puedes elegir una de las seis opciones que se muestran por pantalla pulsando los pulsadores 1, 2, 3, 4, 5 o A, como se puede ver en la Figura 24.



Figura 24. Opciones de configuración en la pantalla LCD.

SLD: salvado del estado de los LEDs

Uno de los aspectos negativos del dispositivo original es que no se sabe bien cuándo se había accionado un efecto y cuándo no. Al pulsar un pedal se enciende su LED correspondiente un momento, pero esta luz no se mantiene encendida mostrando que el efecto está funcionando. En el nuevo dispositivo esto sí ocurre. Una vez accionas un efecto, la luz queda encendida hasta que se desactiva.

Además, se añade una funcionalidad adicional que hace de la pedalera un dispositivo mucho más amigable. Se ofrece la posibilidad de guardar el estado de los LEDs de 8 de sus *presets*, con el objetivo de que cuando se cambie a uno de ellos las luces asociadas a los efectos activos se enciendan. De esta manera, el usuario de un solo vistazo puede saber qué efectos están activos y cuáles no. Esta funcionalidad es muy útil para las actuaciones en directo, en las que se suelen manejar varios *presets* y el cambio de uno a otro puede ser muy rápido.

Para guardar el estado inicial de los LEDs de cada *preset*, lo primero es acceder al menú de configuración (mediante una pulsación larga en el pulsador 5) y efectuar una pulsación corta en el pulsador 1 como se indica en la pantalla LCD (opción SLD). A continuación, se detallan los pasos a seguir.

1. En la pantalla LCD, se muestra la opción de configuración (“guardado LEDs”) y “Preset: ” en la fila inferior, ya que espera que el usuario introduzca el primer carácter del *preset* que quiere configurar.
2. Se pueden guardar el estado de los LEDs de 8 *presets* (1A, 1B, 1C, 1D, 2A, 2B, 2C y 2D) todos asignados a banco 1. Para seleccionar el *preset* se accionan los pulsadores 1 o 2 y posteriormente los pulsadores del A al D. Así, si se quisieran guardar el estado de los LEDs del *preset* 2C, en este paso se debe pulsar el botón 2.
3. En la pantalla aparece el carácter introducido. En el caso de ejemplo, aparece “2”. La pedalera queda en este estado hasta que el usuario introduzca el segundo carácter.
4. Se introduce el segundo carácter con los pulsadores de la fila inferior (de la A a la D). En el caso de ejemplo, se acciona el pulsador C.
5. Aparece en la pantalla LCD el *preset* introducido.
6. Aparece en la pantalla LCD “Seleccione LEDs” y se apagan todos los LEDs de la pedalera.
7. En este momento el usuario debe encender los LEDs que deben estar activos cuando se cambie al *preset* seleccionado mediante pulsaciones en los pulsadores correspondientes.
8. Una vez seleccionados, se realiza una pulsación larga en el pulsador 6 para guardar.
9. Aparece “GUARDADO” en la pantalla LCD. La pedalera pasa a Modo *Preset* y queda cargado el estado de los LEDs que se acaba de guardar. En los *displays* aparece el *preset* también.

Estos pasos se encuentran ilustrados en el diagrama de uso de la Figura 25. El código de colores para los distintos tipos de pulsaciones sigue siendo el mismo: blanco para pulsaciones cortas, rosa para pulsaciones largas y azul para pulsaciones dobles.

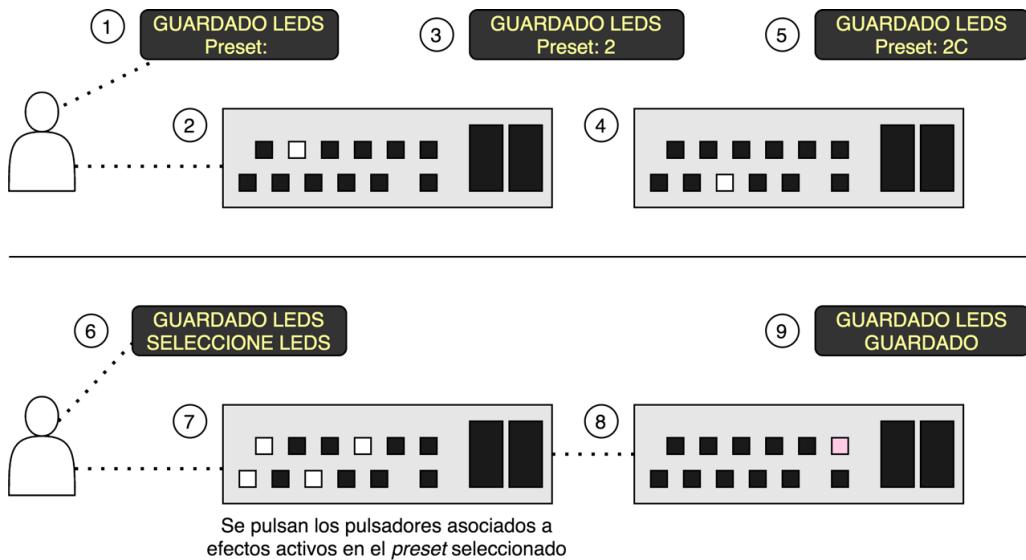


Figura 25. Diagrama de uso del Modo de Configuración SLD (salvado del estado de los LEDs).

LLD: carga del estado de los LEDs

Cuando se cambie a los *presets* con luces LEDs guardadas mediante los comandos MIDI como, por ejemplo, el de PRESET+, estas luces aparecen. Pero, además, es posible cargar las luces de un *preset* en concreto desde la configuración. Mediante la opción 2 del menú de configuración, es posible introducir el que se desea cargar y las luces LEDs asociadas se iluminan. Para ello, estos son los pasos a seguir:

1. En la pantalla LCD, se muestra la opción de configuración (“cargado LEDs”) y “Preset: ” en la fila inferior, ya que espera que el usuario introduzca el primer carácter del *preset* que quiere configurar.
2. Se selecciona el primer carácter del *preset* a cargar de igual manera que en la opción de configuración SLD. Por ejemplo, si se desea cargar el *preset* 2C, en este paso se debe accionar el pulsador 2.
3. En la pantalla aparece el carácter introducido. En el caso de ejemplo, aparece “2”. La pedalera queda en este estado hasta que el usuario introduzca el segundo carácter.
4. Se introduce el segundo carácter con los pulsadores de la fila inferior (de la A a la D). En el caso de ejemplo, se acciona el pulsador C.
5. Aparece en la pantalla LCD el *preset* introducido.
6. Aparece “PRESET CARGADO” en la pantalla LCD.
7. La pedalera pasa a Modo *Preset*, en los displays aparece el *preset* seleccionado y se encienden los LEDs correspondientes.

A continuación, en la Figura 26 se muestra el diagrama de uso de esta opción de configuración.

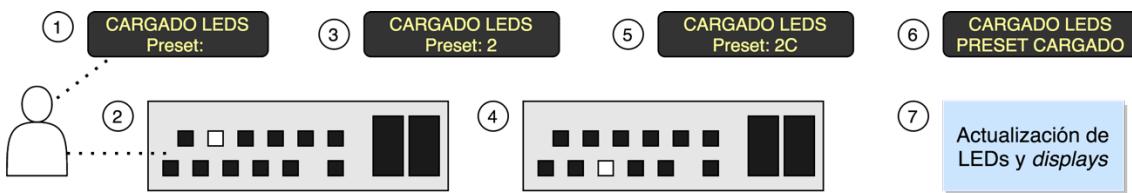


Figura 26. Diagrama de uso del Modo de Configuración LLD (cargado del estado de los LEDs).

Cada vez que se cargue un nuevo *preset* de cualquiera de las maneras, este se guarda en la memoria *flash* del ESP32. Esta memoria tiene 512 bytes en los que se pueden guardar datos de manera permanente. Así, cuando se encienda el dispositivo, se carga el último *preset* que ha estado funcionando en la pedalera.

CC: carga de comandos CC

Se debe configurar en número de comando CC para cada una de las acciones que se pueden realizar mediante comandos MIDI en la aplicación. Para ello se utiliza un fichero JSON que se envía al microcontrolador ESP32 desde el teléfono móvil mediante *Bluetooth BLE* (*Bluetooth Low Energy* o *Bluetooth* de baja energía).

El Bluetooth BLE es un protocolo de bajo consumo que, como su propio nombre indica, necesita una potencia muy baja, a diferencia del protocolo convencional. Esto se debe a que se encuentra en modo reposo la mayor parte del tiempo, excepto cuando se realiza una transmisión. Estas transmisiones suelen ser de poca cantidad de datos. Consigue un rango de alcance aceptable y funciona con dispositivos de distintos fabricantes. Además, se considera bastante seguro, ya que utiliza cifrado de 128 bits y autenticación. También corrige errores de transmisión sin que haga falta que se reenvíe información.

Para realizar este tipo de conexión entre el teléfono móvil y el microcontrolador ESP32 se debe utilizar una aplicación específica. Hay varias disponibles en la *App Store* (aplicación para descargar aplicaciones de Apple). En el desarrollo de este proyecto se utiliza una llamada “nRF Connect”. Esta aplicación permite escanear los dispositivos y empezar el intercambio de datos con uno de ellos. Es posible tanto transmitir como recibir datos. En nuestro caso, se utiliza tanto para transmitir un texto en forma de JSON para realizar la configuración, como para recibir los comandos MIDI.

Este texto se puede generar escribiendo en cualquier editor de texto en formato JSON. Sin embargo, se recomienda descargar alguna aplicación en la que se pueda realizar de manera más intuitiva. Una de las mejores es “Jayson”, que también se puede encontrar en la *App Store*. En este tipo de aplicaciones se introduce clave y valor y se genera el fichero JSON automáticamente.

A continuación, en la Figura 27, se muestra un ejemplo del JSON que se debe enviar al dispositivo. Cada una de las acciones a las que se le puede asociar un comando tendrá una clave. A esta clave se le asociará el valor que queramos. Por ejemplo, si se quiere asociar el CC12 a activar/desactivar el amplificador, hay que asociar a la clave “AMP” el número “12”. En JamUp hay que realizar esta misma asociación en el menú de configuración MIDI.

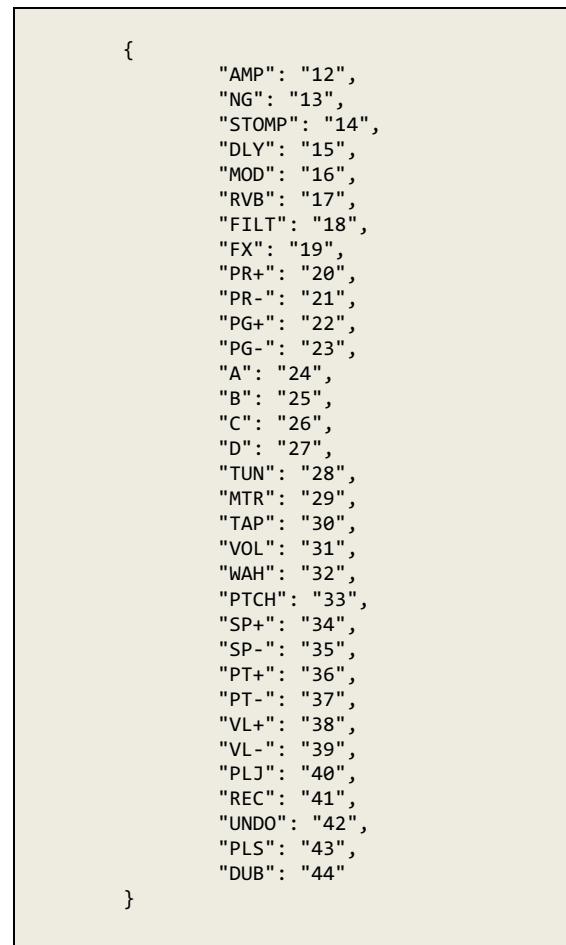


Figura 27. Ejemplo de un fichero JSON que sirve para configurar todos los comandos Control Change.

El texto de ese fichero se copia de manera muy sencilla en la aplicación “nRF Connect” y se envía al ESP32. El microcontrolador carga estos comandos y los asocia a los pulsadores como se ha expuesto antes. Quedan cargados, pero si se quieren guardar para que se mantengan aunque se apague el dispositivo, hay que acceder al modo CC del menú de configuración, que guardará en la memoria *flash* cualesquiera que sean los comandos asociados a los pulsadores en el momento en el que se acceda a él.

El procedimiento que hay que llevar a cabo para establecer la conexión entre microcontrolador ESP32 y el teléfono móvil y para crear y enviar el JSON se

encuentra detallado en el “Manual de Usuario” que se incluye en los anexos de este documento. En esas páginas se adjuntan capturas de pantalla de los pasos a seguir en las dos aplicaciones nombradas: “Jayson” y “nRF Connect”.

ML: configuración del modo libre

De igual manera que en el apartado anterior, para asociar los comandos MIDI a los distintos pulsadores se utiliza un fichero JSON. Cada pulsador tiene tres claves, una para cada tipo de pulsación (S* para cortas, L* para largas y D* para dobles). En esta ocasión, los pulsadores se numerarán del 0 al 9 empezando por el que antes fue llamado pulsador 1 hasta el F que sería el 9. Así, la clave de la pulsación larga del pulsador 4 será L3. Cada clave lleva asociados tres valores: uno para cada byte del comando MIDI a completar, ya que ahora el tipo de comando no está limitado a *Control Change* (CC o Cambio de Control).

El primer byte es el de estado, que indica qué función activar y en qué canal. Los 4 bits que se marcan como nnnn o N en hexadecimal son los de canal. N va de 0 al 15 para canales del 1 al 16. En la Tabla 2 se muestran los valores del byte de estado de los distintos tipos de mensajes MIDI junto al significado que tienen los valores del segundo y tercer byte para cada uno. Cabe destacar que el byte más significativo del segundo y tercer byte es 0, lo que quiere decir que sus valores estarán comprendidos entre 0 y 127.

Tabla 2. Tipos de mensajes MIDI

Función	Binario	Hex	Decimal	Byte 2	Byte 3
<i>Note Off</i>	1000 nnnn	8 N	128 + N	Altura	Velocidad
<i>Note On</i>	1001 nnnn	9 N	144 + N	Altura	Velocidad
<i>Aftertouch Polifónico</i>	1010 nnnn	A N	160 + N	Altura	Presión
<i>Cambio de Control</i>	1011 nnnn	B N	176 + N	Nº de control	Nuevo valor
<i>Cambio de Programa</i>	1100 nnnn	C N	192 + N	Nº programa	
<i>Aftertouch de Canal</i>	1101 nnnn	D N	208 + N	Presión	
<i>Pitch Bend</i>	1110 nnnn	E N	224 + N	<i>MSByte</i>	<i>LSByte</i>
Sistema Exclusivo	1111 nnnn	F N	240 + N		

La descripción de los distintos tipos de mensajes se añade a continuación, con la intención de no sobrecargar la tabla:

- *Note Off*: soltar una tecla.
- *Note On*: pulsar una tecla.
- *Aftertouch Polifónico*: presionar una tecla después de haberla tocado. Cada nota tiene su propio valor de presión.
- *Cambio de Control*: accionar algún tipo de controlador como, por ejemplo, un pedal.
- *Cambio de Programa*: cambiar un programa o sonido.

- *Aftertouch* de Canal: presionar una tecla después de haberla tocado. La variación de presión afecta a todo el canal MIDI.
- *Pitch Bend*: modificar la altura del sonido. Se suele accionar con una rueda o palanca.
- Sistema Exclusivo: enviar mensajes que solo afectan a una unidad de un fabricante o modelo determinado.

Para algunos de los comandos no tiene sentido su envío mediante los pulsadores, aún así se le da esta libertad al usuario. En cambio, a los pedales de expresión se le pueden asignar 4 comandos: *Aftertouch* de Canal, *Aftertouch* Polifónico, *Control Change*, y *Pitch Bend*. El movimiento del pedal en los dos primeros modifica el valor del byte de presión. En el tercero, el byte de valor. En cambio, en el comando *Pitch Bend* se utilizan los dos bytes para mandar el valor de la entrada con el objetivo de alterar el sonido de una manera más fluida. De estos dos bytes se utilizan 14 bits, lo que significa que se pueden representar 16383 valores, que van desde -8192 a 8192.

Aquí hay una limitación: la entrada del Attiny85 en la que se está realizando la lectura analógica tiene una profundidad de 10 bits. Es decir, se leen valores enteros entre 0 y 1023. Se utilizan el valor máximo y mínimo obtenidos de la calibración para enviar valores comprendidos entre los esperados, pero evidentemente no se conseguirá que el cambio sea tan fluido.

En la Figura 28 se muestra el fichero de configuración de un Modo Libre que haría lo siguiente:

- Canal 3 – Pulsador A – Pulsación corta: Cambio de Programa al 27.
- Canal 10 – Pulsador 3 – Pulsación larga: *Note On* de C5 con una velocidad de 64.
- Canal 16 – Pulsador C – Pulsación doble: Cambio de Control 8 a un valor de 16.
- Canal 5 – Pedal de expresión 1 – *Aftertouch* polifónico. Nota D4.

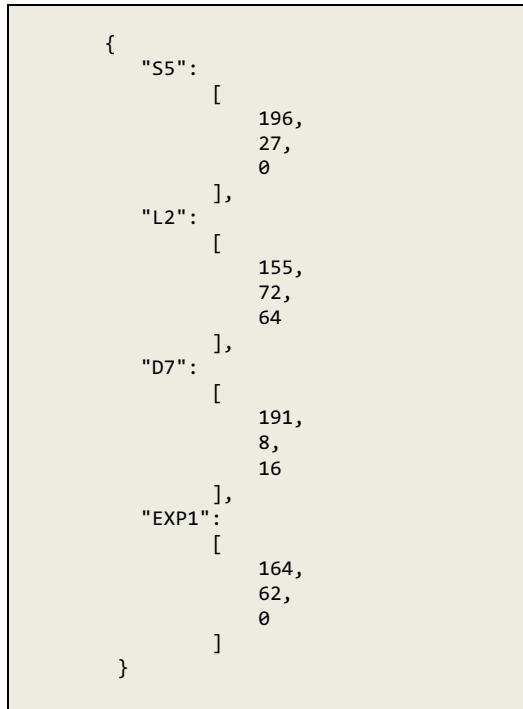


Figura 28. Ejemplo de un fichero JSON que sirve para configurar algunos comandos de Modo Libre.

EXP: calibración de pedales de expresión

Para que los pedales de expresión del dispositivo tengan un funcionamiento correcto es necesario un proceso de calibración. Aunque no es estrictamente necesario realizar esto cada vez que se active la pedalera, sí es recomendable hacerlo con cierta asiduidad.

Los valores leídos por el Attiny están comprendidos entre 0 y 1023. Sin embargo, nunca se suelen recibir valores menores de 350, ya que no se llega a recibir cero voltios. Como los valores a enviar en los comandos MIDI van de 0 a 127, excepto en el Pitch Bend, habrá que asociar los valores leídos a valores que enviar de manera que tenga sentido. Es decir, si el valor más bajo leído es 350, cuando se lea este valor se enviará 0, y si el más alto es 1023, se enviará 127. Para ello hay que hacer una lectura del valor mínimo y máximo que envían los pedales, que puede variar. Esto es lo que se hace en la calibración.

Para acceder a esta opción, se realiza una pulsación larga en el pulsador 5 para entrar al menú de configuración y se selecciona de nuevo el pulsador 5 para acceder a EXP (calibración de pedales de expresión), como aparece indicado en la pantalla LCD.

Primero, la pantalla indica “MÍNIMO”, por lo que se bajan los pedales de expresión hasta el mínimo. Se registra el valor mínimo de los leídos. A los pocos segundos, la pantalla indicará “MÁXIMO”, por lo que se hace lo contrario, es decir, poner los pedales de expresión en su máximo. Se registra el valor máximo de los leídos. A los segundos aparecerá “GUARDADO”. Los pedales han sido calibrados.

DSP: añadir dispositivo a bus I2C

Pulsando el pulsador A se accede a la última opción del menú, donde se puede añadir un nuevo módulo al bus I2C. A continuación, se detallan los pasos a seguir para añadir un dispositivo cuya dirección es la 24:

1. La pantalla LCD muestra la opción de configuración en la fila superior (NUEVO MÓDULO) y, en la fila inferior el texto “Direcc (0-99): “, que hace alusión a que la máquina espera que el usuario introduzca una dirección del 0 al 99. Las direcciones del bus I2C ocupadas actualmente por los distintos módulos son: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 y 16. No se puede asignar ninguna de estas direcciones al nuevo módulo.
2. Se introduce el primer carácter de la dirección. Para disponer de todos los caracteres numéricos necesarios, los pulsadores se numeran del 0 al 4 en la fila superior y del 5 al 9 en la fila inferior. Los dos pulsadores sin LED no introducen ningún número. En el caso de ejemplo, se pulsa el tercer botón de la fila superior para introducir el primer dígito, que es un 2.
3. La pantalla LCD muestra el dígito introducido en la pantalla.
4. Se introduce ahora el segundo carácter. En el caso de ejemplo, un 4. Para ello se pulsa el quinto botón de la fila superior.
5. La pantalla LCD muestra la dirección introducida.
6. Aparece en la pantalla “MÓDULO AÑADIDO”, la dirección se guarda y se redirige a Modo *Preset*.

En la Figura 29 se muestra el diagrama de uso de esta opción de configuración, que ilustra los pasos que se acaban de detallar.

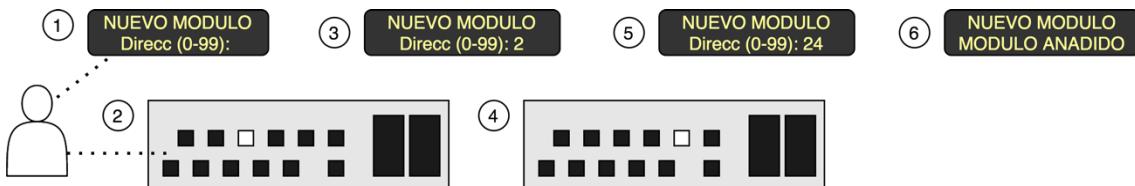


Figura 29. Diagrama de uso del Modo de Configuración DSP (añadir dispositivo al bus).

A partir del momento en el que se realiza la salva de la dirección del nuevo módulo, en todos los modos se tendrá en cuenta este. El controlador está preparado para recibir comandos MIDI del nuevo módulo y enviarlos por Bluetooth y por el conector DIN de 5 pines. Asimismo, está limitado para solo poder añadir un módulo adicional, ya que tal y como está diseñado, añadir varios módulos con sus respectivos cables comprometería la estabilidad del bus. Cabe destacar que, si se reinicia el dispositivo hay que volver a añadir la dirección.

4.2 Módulos secundarios

Una vez se ha expuesto el funcionamiento completo de la máquina, se explica el diseño y la lógica de cada uno de los distintos módulos que lo componen. En la Figura 30 se muestra cómo se disponen y un ejemplo de cómo se podrían interconectar para utilizar poca longitud de cable.

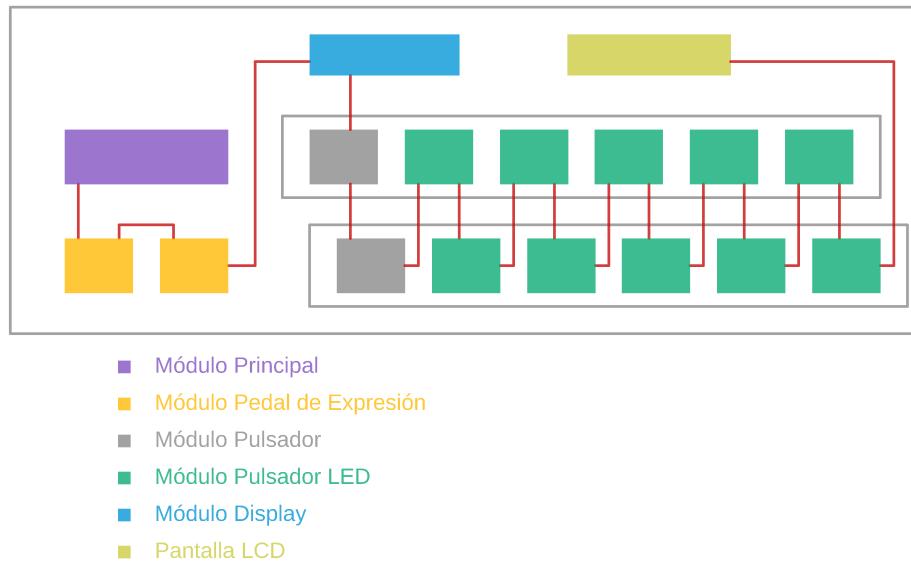


Figura 30. Esquema de distribución de los módulos. Vista posterior del dispositivo.

El “cerebro” de cada uno de los módulos secundarios (excepto de la pantalla LCD) es el microcontrolador Attiny85. Se elige este dispositivo porque tiene un tamaño muy pequeño, es barato, de bajo consumo y es capaz de cumplir los requisitos de los módulos secundarios de nuestro proyecto, tanto por número de pines como por espacio en la memoria. Además, es compatible con el *software* Arduino. En este proyecto se ha programado utilizando un Arduino UNO como ISP (*In-system programming* o “Programación en el sistema”).

Este microcontrolador cuenta con:

- 6 pines GPIO (*General Purpose Input/Output* o “Entrada/Salida de Propósito General”).
- 4 canales de convertidor analógico-digital de 10 bits.
- Memoria *Flash* de 8KB.
- Memoria EEPROM de 512 Bytes.
- Memoria SRAM (*Static Random Access Memory* o Memoria Estática de Acceso Aleatorio) de 512 Bytes.

Además, para su pequeño tamaño tiene un gran rendimiento, llegando a ejecutar 1 MIPS (Millón de Instrucciones Por Segundo) por megahercio (MHz). En este proyecto, los microcontroladores trabajan a 8MHz y están alimentados con cinco voltios.

A continuación, en la Figura 31, se muestra un esquema de los pines y una fotografía de cómo es el dispositivo.

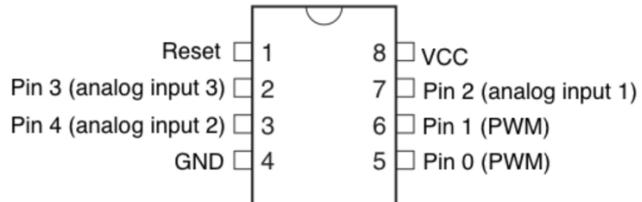


Figura 31. Microcontrolador Attiny85. Izq: imagen del dispositivo; Dcha: esquema de pines.

4.2.1 Módulo Pulsador LED

El sistema distribuido cuenta con 10 Módulos Pulsador LED. Cada uno de ellos controla un pulsador con LED de los 10 que hay en la pedalera. El Attiny85 tiene, por tanto, una entrada (el pulsador) y una salida (el LED). En la Figura 32 se muestra cómo están distribuidos estos módulos en la máquina.



Figura 32. Localización de los Módulos Pulsador LED en la parte posterior del dispositivo.

El microcontrolador debe diferenciar entre las 3 pulsaciones posibles del pulsador (corta, larga y doble). Además, según el tipo de pulsación debe encender el LED de una manera u otra.

- Si la pulsación es corta, el LED cambia de estado. Es decir, si estaba apagado se enciende y, si estaba encendido se apaga.

- Si la pulsación es larga, el LED realiza un parpadeo largo y vuelve a su estado original. Es decir, si el LED estaba apagado, después del parpadeo largo queda apagado.
- Si la pulsación es doble, el LED realiza un parpadeo corto y vuelve a su estado original. Es decir, si el LED estaba apagado, después del parpadeo corto queda apagado.

Además, el Attiny85 tiene que conectarse al bus I2C. De esta manera puede intercambiar datos con el microcontrolador ESP32 del módulo principal:

- Envía: notificación de pulsación efectuada. Se informa del tipo de pulsación (corta, larga o doble) y de la procedencia (qué pulsador se ha pulsado).
- Recibe: cambios de estado del LED.

Hay que destacar que el *hardware* de este microcontrolador no viene preparado para que se pueda utilizar I2C, sino que se realiza una simulación mediante *software* para utilizar los pines PB0 y PB2 como SDA y SCL respectivamente.

Diseño

Al Attiny se conecta el pulsador y el LED de la placa original. Se realizan pruebas para determinar cómo está colocado el led. Se observa que se activa cuando el pin esté a nivel bajo. Se coloca una resistencia de 120 Ohmios para limitar la corriente.

Por otro lado, se conecta el pulsador a otro pin y se añade una resistencia *pull-up* de 10 kΩ para evitar que el puerto esté en estado flotante (indefinido). De esta manera se limita la corriente y se asegura que si se recibe un nivel bajo es porque el interruptor ha sido accionado y no por señales parásitas. Además, hay otras dos resistencias *pull up*, esta vez de 4,7 kΩ, que sirven para estabilizar el nivel lógico de los bits en el bus de comunicación I2C.

En los módulos correspondientes a los pulsadores 5 y E se puede observar que hay más cables entrantes, no sólo el pulsador y el LED. Estos son VCC y GND, que sirven para alimentar a las placas alargadas horizontales que albergan los pulsadores. Como se ha comentado, la placa de arriba tenía un cable VCC y otro de tierra. Estos se conectan en el Módulo Pulsador LED del pulsador 5. Por otra parte, la placa de abajo contaba con dos cables VCC y uno de tierra. Estos se conectan al módulo del pulsador E.

Por último, hay que destacar los dos conectores RJ11 que estarán presentes en cada módulo para la interconexión de estos. Se recuerda que a los conectores hembra llegan las señales VCC, GND, SDA y SCL, que posteriormente son transmitidas al resto de módulos mediante cable telefónico, que cuenta con cuatro vías. La Figura 33 es el esquema electrónico del módulo, donde se puede observar lo detallado anteriormente.

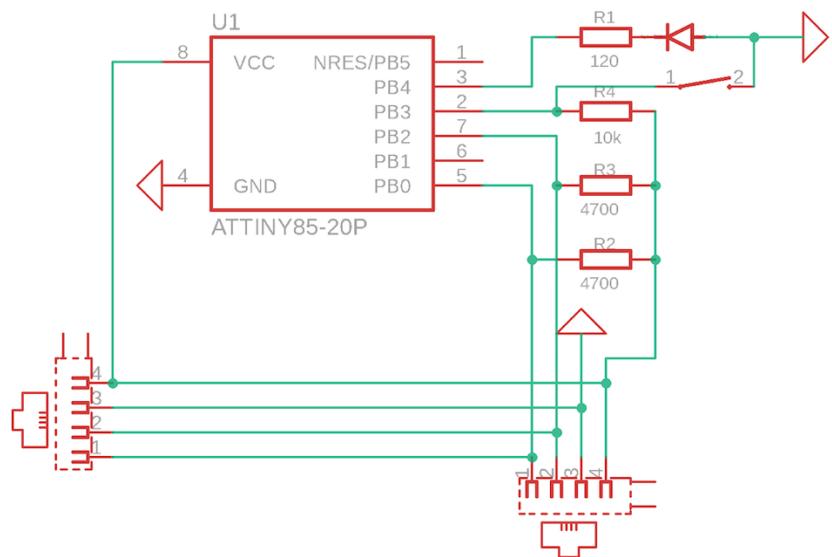


Figura 33. Esquema electrónico del Módulo Pulsador LED.

En la Figura 34, se muestra una imagen del resultado final de la construcción del módulo.

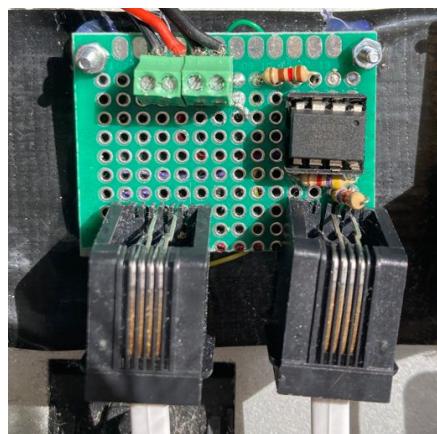


Figura 34. Uno de los diez Módulos Pulsador LED que componen el dispositivo.

Lógica

El código del microcontrolador debe diferenciar entre los 3 tipos de pulsaciones:

- Corta: el pulsador se mantiene pulsado menos de 400 milisegundos.
- Larga: el pulsador se mantiene pulsado más de 400 milisegundos.
- Doble: dos pulsaciones cortas con menos de medio segundo de diferencia.

Como se ha visto en el diseño, el pulsador está a nivel alto (cinco voltios) y mientras está pulsado pasa a nivel bajo. Para captar el tipo de pulsación que se efectúa se controla el flanco de bajada y el de subida. Se guarda en qué momento se

produce el flanco de bajada y cuándo se produce el de subida y se observa cuánto tiempo ha pasado. Si es más de 400 milisegundos será una pulsación larga. Si es menos, puede ser una pulsación corta o una doble. Si ha habido un flanco de bajada de una pulsación anterior hace menos de medio segundo será una pulsación doble.

Si no es así hay que esperar un tiempo hasta poder asegurar que no es la primera pulsación de una pulsación doble. Cuando hayan pasado 300 milisegundos sin que haya habido otra pulsación se notifica la pulsación corta.

Esta lógica está recogida en el diagrama de la Figura 35.

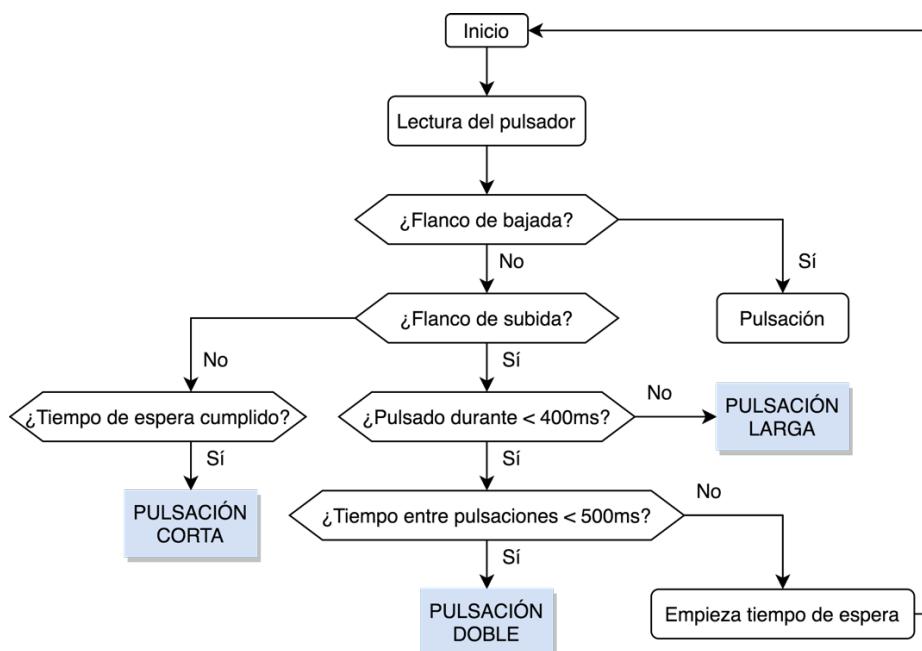


Figura 35. Diagrama de pseudocódigo del Módulo Pulsador LED.

Hay que destacar que, para considerar que estamos ante un flanco de bajada o de subida, debe haber pasado el tiempo de rebote, que está fijado en cien milisegundos. Para determinar este valor se realizaron numerosas pruebas hasta dar con el menor tiempo que aportara cierto margen de seguridad.

Pero, una vez el Attiny85 determina qué pulsación se ha efectuado, ¿qué hace? Principalmente realiza dos acciones:

1. Mostrar la salida correspondiendo en el LED asociado. Como se ha comentado antes:
 - a. Pulsación corta: cambia de estado. Si estaba apagado pasa a encendido.
 - b. Pulsación larga: parpadeo largo y se mantiene en el estado que estaba.

- c. Pulsación doble: parpadeo corto y se mantiene en el estado que estaba.
2. Enviar el byte de datos al máster. Este byte tiene que indicar:
- a. Qué pulsador ha sido accionado. Se reservan 4 bits para representar esta información. Estos serán los bits menos significativos. Hay que aclarar que el máster detecta la dirección de la que vienen los datos pero en el programa principal resulta útil recibir el número de pulsador y no la dirección.
 - b. Si el mensaje notifica una pulsación de alguna manera. Se ha fijado que se utilice el quinto y sexto bit (empezando la cuenta por el menos significativo). Solo se notifica una pulsación si el valor de estos bits es cero.
- ¿Por qué no se recibe información del módulo solamente cuando haya una pulsación? El I2C es un protocolo síncrono. Los dispositivos que conforman el bus pueden ser másters (en este proyecto, el microcontrolador ESP32) o esclavos (el resto de los microcontroladores de módulos secundarios). Los esclavos no pueden enviar información al máster sin que este la haya solicitado. Por tanto, para que los esclavos puedan realizar notificaciones a tiempo real, se debe usar *polling* o sondeo. Esta técnica consiste en la consulta constante a los esclavos por parte del máster con el objetivo de saber si hay algún evento. La parte positiva de esta técnica frente a las interrupciones es que este procedimiento evita colisiones, ya que el máster consulta a los esclavos de una forma ordenada, por lo que no es posible recibir información de dos microcontroladores a la vez. Por otro lado, hay una desventaja, existe un tiempo de *polling* (tiempo entre dos consultas). En la pedalera este tiempo es de aproximadamente 60 milisegundos. No supone un retardo perceptible, ya que se considera lo suficientemente pequeño para no entorpecer el buen funcionamiento de la interacción con la máquina.
- c. Qué tipo de pulsación se ha efectuado. Hay tres tipos de pulsaciones, por lo que solo necesitaremos dos bits para enviar esta información. Se utilizan los dos bits más significativos.
 - Corta: 00
 - Larga: 01
 - Doble: 10

Por otro lado, cabe resaltar que, en el bus I2C, el Attiny85 no solo envía datos, sino que también recibe información del máster para determinar el estado del LED. El módulo principal puede enviar a cualquier Módulo Pulsador LED un booleano verdadero o un falso para encender o apagar su LED.

4.2.2 Módulo Pulsador

En el controlador MIDI hay dos pulsadores sin LED. A continuación, en la Figura 36, se muestra una imagen de la vista posterior del dispositivo para indicar dónde se encuentran.

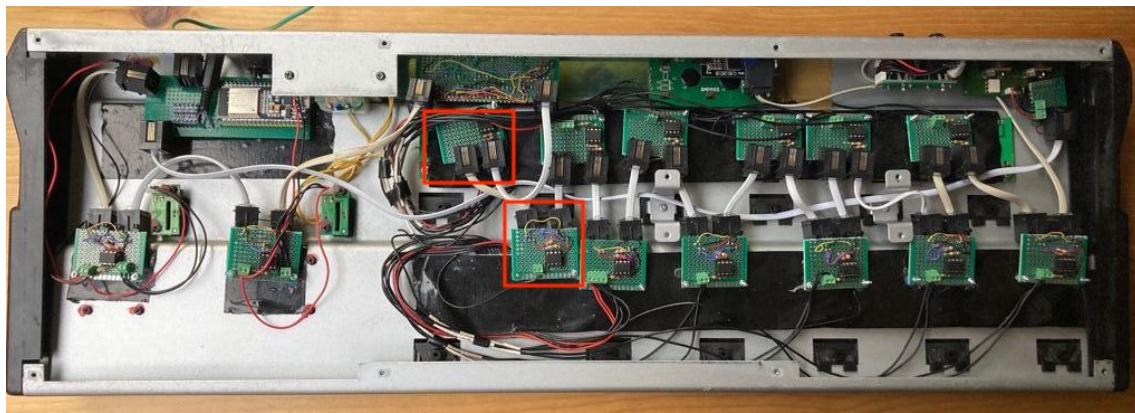


Figura 36. Localización de los Módulos Pulsador en la parte posterior del dispositivo.

El Módulo Pulsador funciona prácticamente igual que el Módulo Pulsador LED. La única diferencia es que, como su propio nombre indica, el Módulo Pulsador no tiene LED. Por esa razón, el LED no figura en el esquema electrónico que se muestra en la Figura 37. Además, en esta misma figura se muestra también la apariencia de la construcción final de uno de estos módulos.

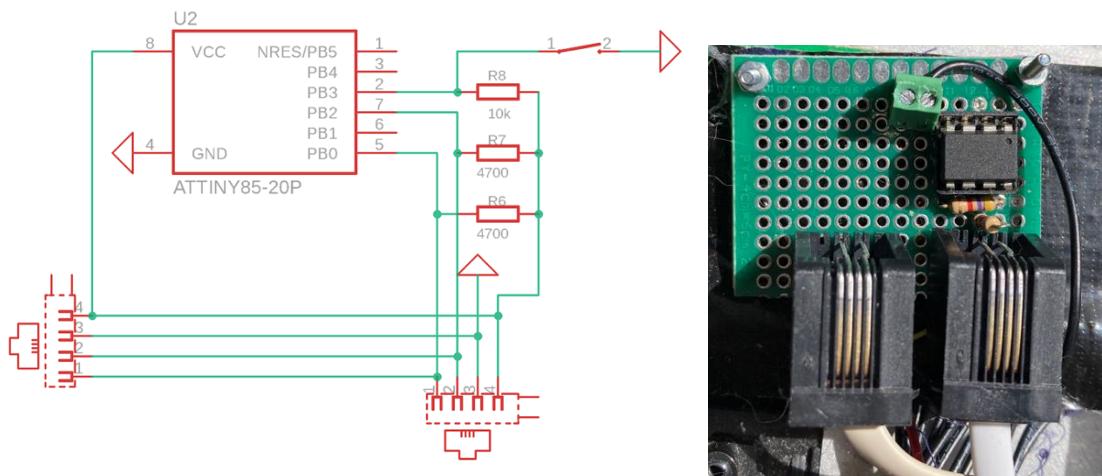


Figura 37. Izq: Esquema electrónico del Módulo Pulsador; Dcha: uno de los módulos que componen el dispositivo.

Por otro lado, la lógica para determinar el tipo de pulsación es la misma a la explicada en el subapartado 4.2.1 Módulo Pulsador LED. Por tanto, ya ha sido descrito su funcionamiento.

La única modificación respecto a la lógica del módulo ya expuesto es que se elimina el código relativo al LED. Es decir, una vez identificada la pulsación, simplemente se forma el byte a enviar para notificar el evento, pero no hay ningún cambio visible para el usuario como ocurre en el Módulo Pulsador LED con el parpadeo o cambio de estado del led.

4.2.3 Módulo Pedal de Expresión

Hay dos pedales de expresión en el dispositivo y, por lo tanto, habrá dos Módulos Pedal de Expresión asignados a ellos. Cada uno de ellos debe realizar la lectura analógica del pedal de expresión y controlar el brillo del LED, que se enciende mientras que se mueve el pedal. Se puede observar dónde se encuentran situados en la Figura 38, que muestra la vista posterior de la máquina.



Figura 38. Localización de los Módulos Pedal de Expresión en la parte posterior del dispositivo.

Diseño

Este módulo cuenta con un microcontrolador Attiny85 al que se conectan el pedal de expresión y el LED del dispositivo original. Por lo tanto, al Módulo Pedal de Expresión se insertan:

- 3 cables de la placa original del pedal. Uno correspondiente a VCC, otro a tierra y el de la señal de salida. Este último se corresponde al voltaje de salida cuando se cambia el valor de la resistencia variable por medio del movimiento del pedal de expresión. Cabe destacar que se aprovechan las dos resistencias en serie de la placa inicial.
- 2 cables del led: uno para VCC y otro para controlar el led, que luce cuando en ese pin haya nivel bajo.

A continuación, en la Figura 39, se muestra un esquema electrónico de este módulo.

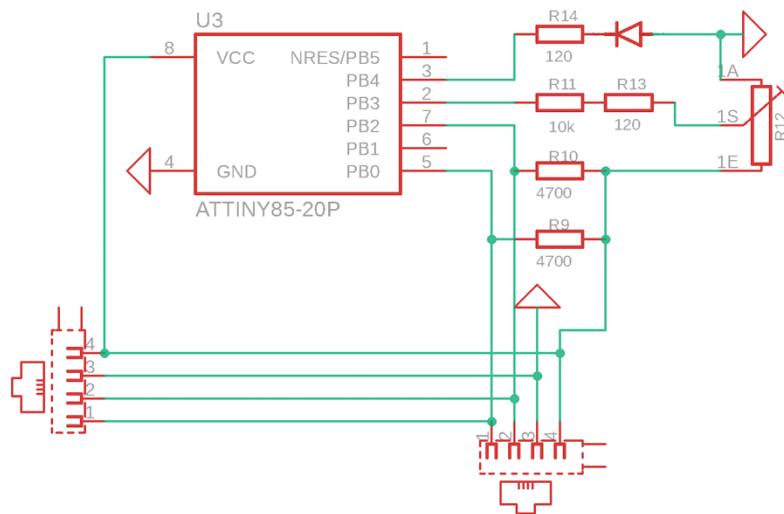


Figura 39. Esquema electrónico del Módulo Pedal de Expresión.

En la Figura 40 se puede observar una fotografía del montaje final de uno de los Módulos Pedal de Expresión.

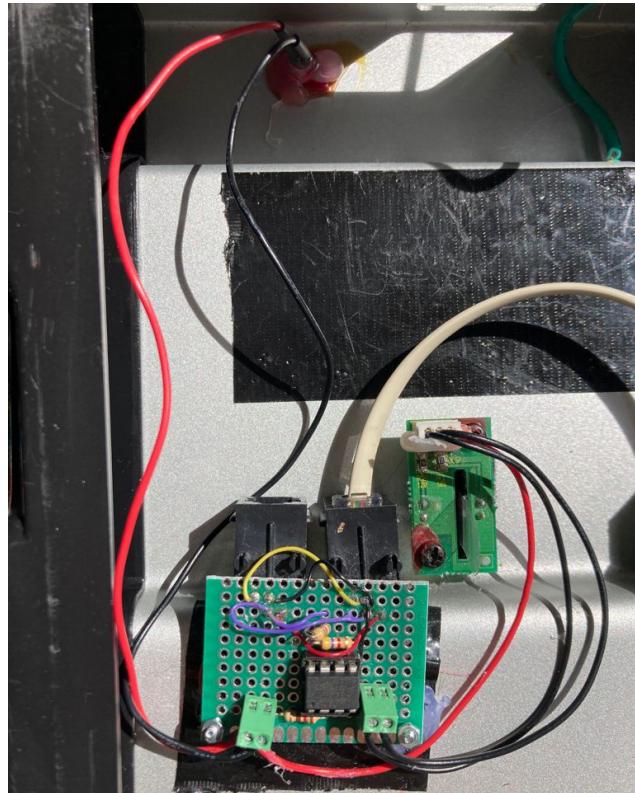


Figura 40. Uno de los dos Módulos Pedal de Expresión que componen el dispositivo.

Lógica

La lógica de este módulo es muy sencilla. La mayor complejidad del código de los pedales de expresión reside en el programa del módulo principal, que es el encargado de calibrarlos y de enviar los comandos MIDI generados.

El Módulo Pedal de Expresión realiza la lectura del pin analógico. Se obtienen valores comprendidos entre 0 y 1023. Se envía este valor por I2C al máster. El envío de datos será continuo. Sin embargo, como se ve más adelante cuando se explique la lógica del módulo principal, solo se envía el comando MIDI si el valor recibido, calibrado para estar comprendido entre 0 y 127, ha variado en más de cinco unidades. Esto se hace así porque los valores obtenidos de la lectura analógica oscilan un poco, aunque el pedal de expresión no esté siendo accionado.

El Attiny controla también el encendido del LED. Se enciende cuando hay un cambio significativo de los valores de entrada de la lectura analógica, es decir, cuando verdaderamente se está utilizando el pedal de expresión. Después de realizar varias pruebas se establece que para que el LED se encienda debe haber 20 unidades de diferencia con la lectura anterior.

A continuación, en la Figura 41, se representa la lógica en un diagrama de pseudocódigo.

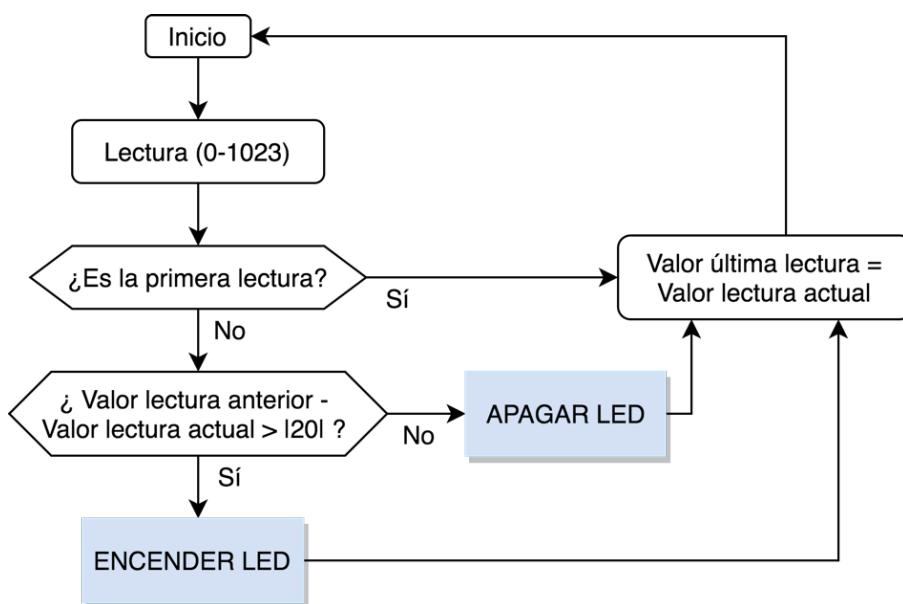


Figura 41. Diagrama de pseudocódigo del Módulo Pedal de Expresión.

4.2.4 Módulo Display

Como el dispositivo principal, nuestra pedalera también cuenta con 3 *displays* de 7 segmentos. Estos sirven para mostrar el banco y el *preset* en curso. En la Figura 42 se muestra el lugar en el que se encuentra colocado el módulo en el dispositivo.



Figura 42. Localización de los Módulos Display en la parte posterior del dispositivo.

Los bancos o páginas irán del 1 al 4 y los *presets*, como ya se ha comentado, tendrán dos dígitos: un número del 1 al 8 y una letra de la A a la D. Se puede acceder por medio de la carga del estado de los LEDs (LLD) a los ocho primeros *presets* del banco 1. A los demás se llega navegando mediante los pulsadores 6 y F. Con pulsaciones largas en estos pulsadores se cambia de página y con pulsaciones cortas de *preset*.

Tanto para el diseño como para la lógica de este módulo, se ha elegido de referencia el post de Tauno Erik en la página web *Hackster* [7], donde explica cómo ha conseguido controlar cuatro *displays* de 7 segmentos de ánodo común mediante un Attiny85 y cuatro registros de desplazamiento de 8 bits. Para este diseño, se adapta el hardware y el código a 3 *displays*.

Diseño

El diseño, aunque está basado en el del post, ha sufrido algunas modificaciones. Entre ellas cabe destacar que se han cambiado la correspondencia entre pines de los registros de desplazamiento y los del *display* de 7 segmentos con la intención de que las conexiones en la placa resultasen un poco más sencillas, ya que los *displays* de ambos proyectos tienen una disposición de pines diferente. La relación entre pines del registro y del *display* se muestra en la Tabla 3.

Tabla 3. Relación entre pines del registro de desplazamiento y del display de siete segmentos.

Registro	OA	OB	OC	OD	OE	OF	OG	OH
Display	g	f	dp	c	d	e	a	b

Hay 3 pines del registro conectados al microcontrolador Attiny85: el de reloj, el de datos y el de *latch* (*late memory*). Los pines de reloj y de *latch* de los distintos registros están interconectados. El pin de datos está conectado a la entrada de datos del primer registro. La salida de datos de este primer registro está conectada a la entrada del siguiente y así sucesivamente. En este caso solo hay 3 *displays* y por tanto, 3 registros. Se puede observar el conexionado en el esquema electrónico de la Figura 43.

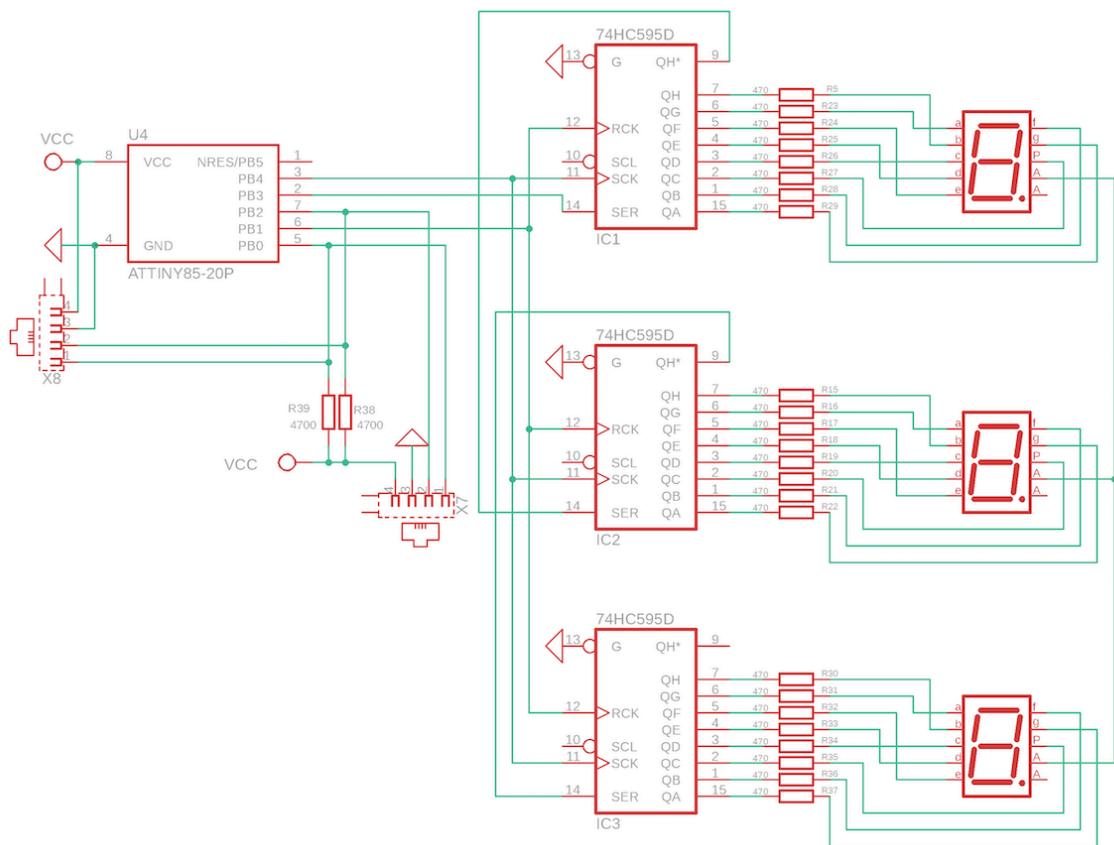


Figura 43. Esquema electrónico del Módulo Display

Por último, se muestra una fotografía del resultado final de la construcción del módulo en la Figura 44.

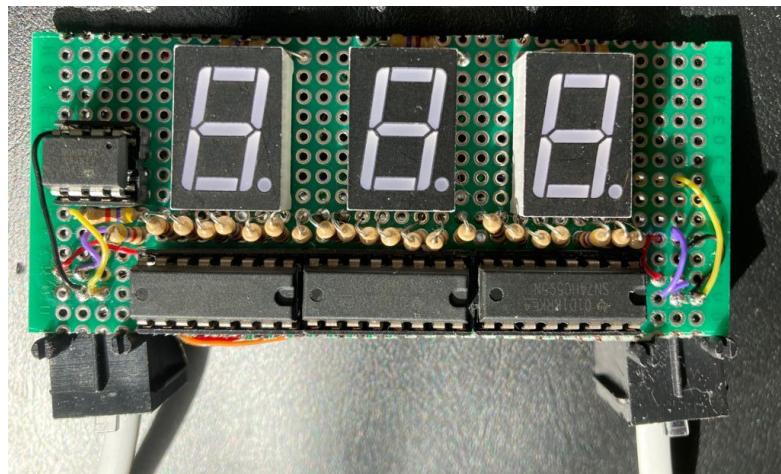


Figura 44. Módulo Display instalado en el dispositivo.

Lógica

Los valores a representar se envían desde el Módulo Principal y llegan al Attiny mediante el bus I2C. Llegan de tres en tres bytes, uno para cada *display*. Cada bit representa si un segmento debe estar apagado o encendido. Pero ¿cómo llega el nivel deseado a cada pin del *display* de 7 segmentos?

El código de este módulo se basa en la función *shiftOut* (*dataPin*, *clockPin*, *value*). Esta función va enviando por el pin de datos (*dataPin*), el valor (*value*) en serie empezando por el bit más significativo. El pin representa el dato cuando se produce un flanco de subida en la señal de reloj.

Así, el valor (nivel alto o bajo) entrará por el pin SER del registro (datos en serie) y se almacenará ahí hasta que llegue el siguiente valor. Entonces, se desplazará el valor anteriormente albergado al pin QA y así sucesivamente. Conforme se cargan nuevos valores en el pin SER, los antiguos se desplazan.

Cuando todos los pines del primer registro tienen un valor, al producirse el desplazamiento, el valor de salida corresponde al valor de entrada del registro siguiente. De esta manera todos los pines de los registros tomarán un valor, bajo o alto, lo que provocará que haya segmentos iluminados (los que estén a nivel bajo) y otros que no (los de nivel alto).

Cabe destacar que el desplazamiento solo se produce mientras que el pin *latch* está a nivel bajo. Una vez estén cargados todos los niveles de los distintos segmentos, el pin *latch* vuelve a tener nivel alto y los valores quedan fijados hasta que el máster envíe un nuevo valor a representar y se repita la operación.

El procedimiento que se acaba de detallar se recoge en el diagrama de la Figura 45.

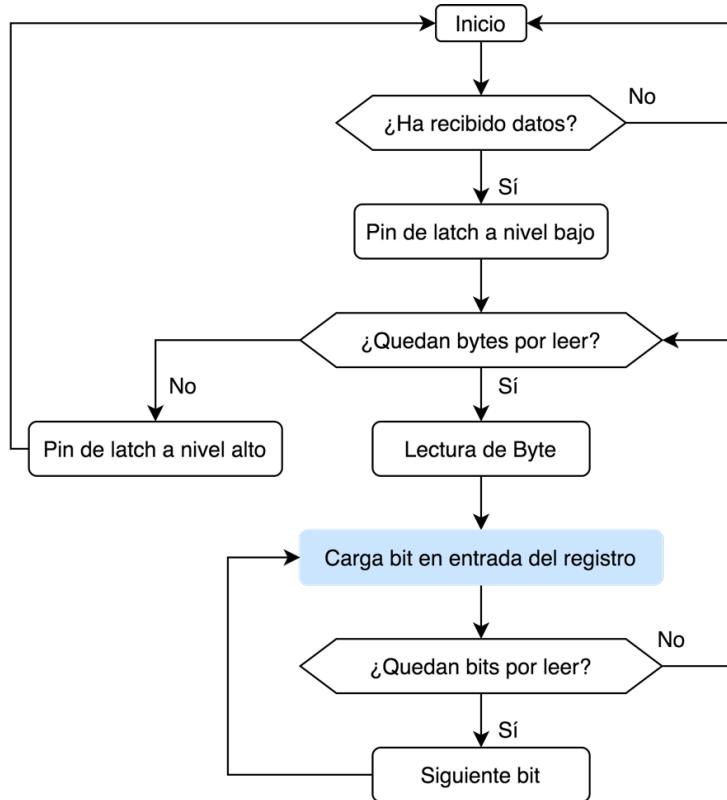


Figura 45. Diagrama de pseudocódigo del Módulo Display.

Se realizan un par de recordatorios para la mejor comprensión del diagrama:

- La lectura de bits se realiza comenzando por el bit más significativo hasta llegar al menos significativo.
- Cuando se carga bit en la entrada del registro se produce un desplazamiento de una posición de los bits anteriormente cargados. El bit que estaba en el pin QA pasa al QB y así sucesivamente.

4.2.5 Módulo LCD

Es el módulo secundario más sencillo, el único que no cuenta con el Attiny85. Se compone únicamente de una pantalla alfanumérica LCD de 16x2 caracteres. Tiene integrado el módulo de conexión a la red I2C, por lo que no hace falta utilizar un microcontrolador adicional. Sus 4 pines corresponden a las señales SDA, SCL, VCC (5V) y GND. En la Figura 46 se puede ver dónde se encuentra situado en el dispositivo.



Figura 46. Localización de los Módulos Display en la parte posterior del dispositivo.

La única adaptación que se hace en cuanto al diseño es colocarle un conector RJ11 para poder conectarla a nuestro bus I2C. Para ello se corta una porción pequeña de placa de fibra de vidrio y se insertan los pines de la pantalla y el conector hembra. Se colocan cables que unen los pines de la pantalla a los pines del conector hembra correspondientes como se puede ver en la Figura 47.

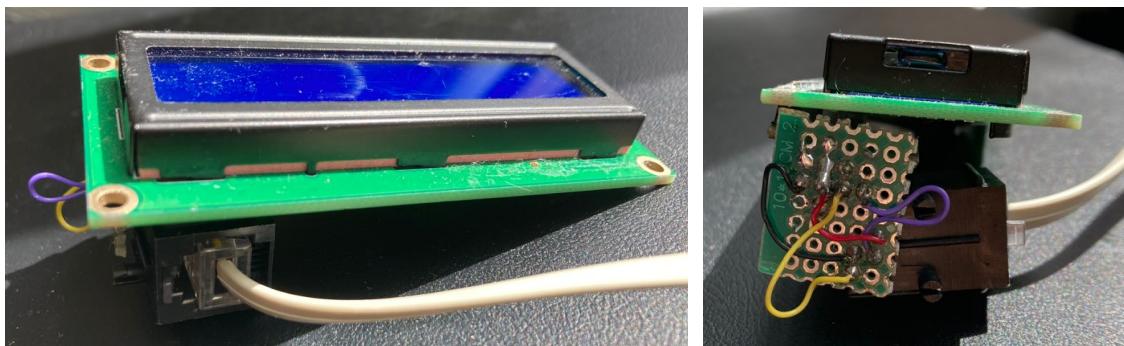


Figura 47. Módulo LCD. Izq: vista general del módulo; Dcha: vista en detalle de la instalación del conector RJ11.

El máster envía los distintos caracteres a representar. Esto se realiza de manera muy sencilla en el programa principal mediante el uso de una librería de Arduino llamada *Liquid Crystal I2C*.

4.3 Módulo Principal

4.3.1 Diseño

El Módulo Principal está compuesto de una placa principal y una placa MIDI. La placa principal cuenta con un microcontrolador ESP32, un conector DC hembra para la alimentación, un conversor de niveles lógicos de 3,3 voltios a 5 voltios y tres conectores RJ11 hembra (dos para el bus I2C y uno para conectar la placa MIDI). Los conectores MIDI. La placa MIDI es un lazo de corriente simple con dos conectores DIN de 5 pines, uno para las señales de entrada y otro para las señales de salida. Se puede observar dónde se encuentra en la Figura 48.



Figura 48. Localización del Módulo Principal en la parte posterior del dispositivo. A la izquierda, la placa principal. A la derecha, la placa MIDI.

El dispositivo se alimenta mediante una fuente de alimentación de cinco voltios y dos amperios que se conecta a la red eléctrica. Esta fuente de alimentación tiene conector DC macho que se introduce en el conector DC hembra del módulo principal. Así, además de alimentar al ESP32 por su pin de 5 V, se suministra energía al resto de dispositivos mediante el pin VCC del conector RJ11.

El conversor se añade porque, aunque el ESP32 pueda ser alimentado con cinco voltios, la salida a nivel alto de sus pines es 3,3 voltios. Lo conveniente es que las señales SDA y SCL tengan el nivel de las del resto del bus, es decir, 5 voltios. Por esta razón se añade este componente, que convierte el nivel de las señales a 5 voltios.

Para este módulo se necesita un microcontrolador que cumpla ciertas exigencias. Hay que tener en cuenta que es el que alberga el programa principal, que contiene la lógica más compleja. Además, es un programa bastante largo, por lo que era necesario disponer de suficiente memoria para código. Asimismo, un requisito indispensable era que tuviera Bluetooth, preferiblemente BLE.

Otros aspectos que suponen un plus es que tuviese un consumo bajo y que fuese bastante utilizado en la comunidad Arduino, algo que es bastante útil a la hora de encontrar librerías y encontrar soluciones a problemas.

En principio, el número de pines no era un factor determinante, ya que, desde el principio el proyecto se planteó como un sistema distribuido con comunicación I2C, por lo que lo único que era necesario es que tuviese posibilidad de realizar este tipo de comunicación.

Si bien es verdad que el ESP32 tiene algunas características que no se utilizan, como la conectividad Wi-Fi, escoger un microcontrolador inferior, como el ESP8266, denominado “su hermano pequeño”, no solo suponía una pérdida de potencia de procesamiento sino una renuncia a la conectividad Bluetooth BLE. Como la diferencia de precio es insignificante (alrededor de un euro) se elige el ESP32 como “cerebro” del proyecto.

Como recoge un artículo de Luis Llamas en su página web [8], algunas de las especificaciones de este microcontrolador son:

- 32 pines GPIO.
- Conversor analógico digital de 12 bits y 18 canales.
- 2 conversores analógico digital de 8 bits.
- 16 salidas PWM (*Pulse Width Modulation* o modulación por ancho de pulso).
- Procesador Tensilica Xtensa de doble núcleo de 32 bits a 160 MHz, que puede ser hasta 240 MHz. Coprocesador de ultra baja energía.
- Wi-Fi 802.11 b/g/n 2.4GHz.
- Bluetooth version 4.2 BR/EDR y BLE.
- Memoria SRAM de 520 kB.y memoria flash externa de hasta 16MiB con encriptación.

A continuación, en la Figura 49, se muestra una fotografía del dispositivo en la que viene detallado el uso de sus pines.

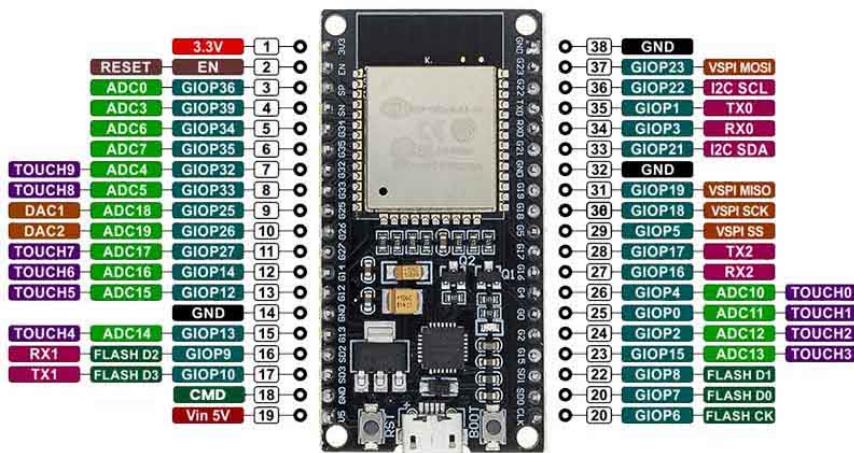


Figura 49. Esquema de pines del microcontrolador ESP32.

A continuación, en la Figura 50, se puede observar el resultado final de la construcción de la placa principal.



Figura 50. Placa principal del Módulo Principal.

Por otro lado, la placa MIDI alberga los conectores DIN de cinco pines, que sirven para transmitir los comandos MIDI por cable, además de por Bluetooth como se ha explicado antes. Se aprovecha la placa que se encontraba instalada en el dispositivo original, que tiene dos conectores DIN, uno para señales de entrada y otro para señales de salida.

De la placa salen cuatro cables: VCC, GND, TX (transmisor) y RX (receptor). VCC (5V) y tierra se conectan a los puntos del módulo principal correspondientes. El transmisor y receptor se conectan a los pines de transmisión y recepción (TX2 y RX2) del ESP32. Como son 4 vías, se decide realizar la conexión entre la placa MIDI y la placa principal con cable telefónico y conectores RJ11. La instalación del conector RJ11 hembra es la única modificación que sufre esta placa respecto a la del dispositivo original. Se puede ver el resultado en la Figura 51.

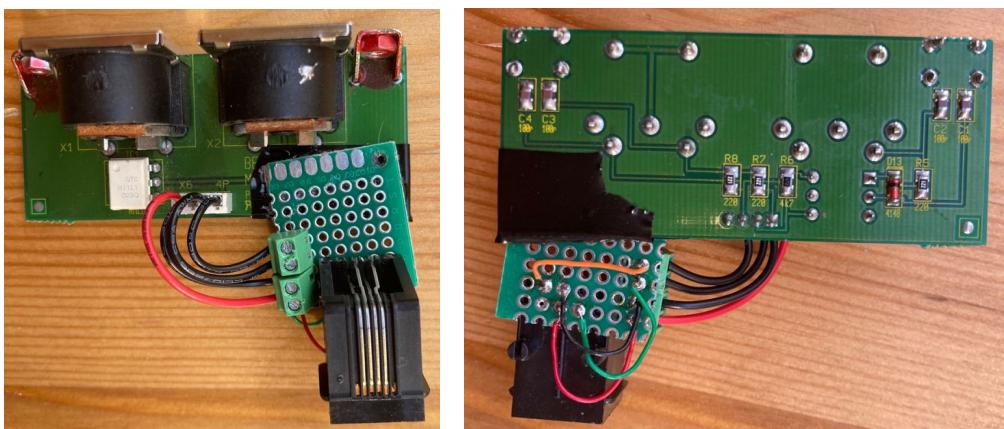


Figura 51. Placa MIDI del Módulo principal. Izq: vista frontal; Dcha: vista posterior.

Por último, en la Figura 52 se muestra el esquema electrónico del Módulo Principal en su totalidad.

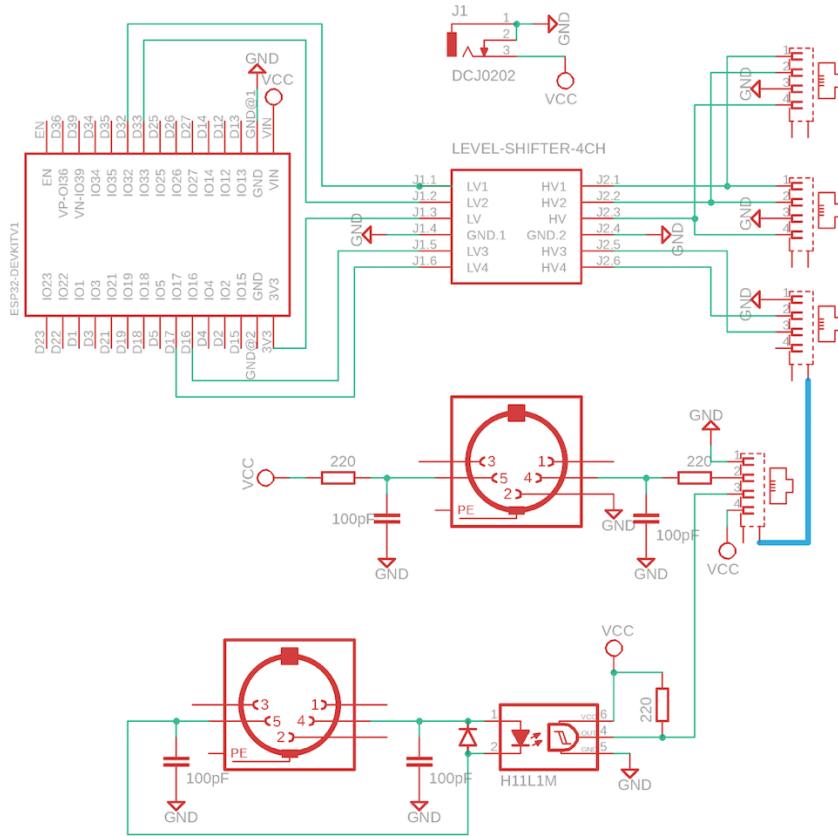


Figura 52. Esquema electrónico del Módulo Principal.

4.3.2 Lógica

Como ya se ha explicado el funcionamiento de los distintos modos a principio del apartado, en este subapartado simplemente se exponen los diagramas de pseudocódigo de cada uno de los distintos modos de funcionamiento, con algunas explicaciones que los clarifiquen.

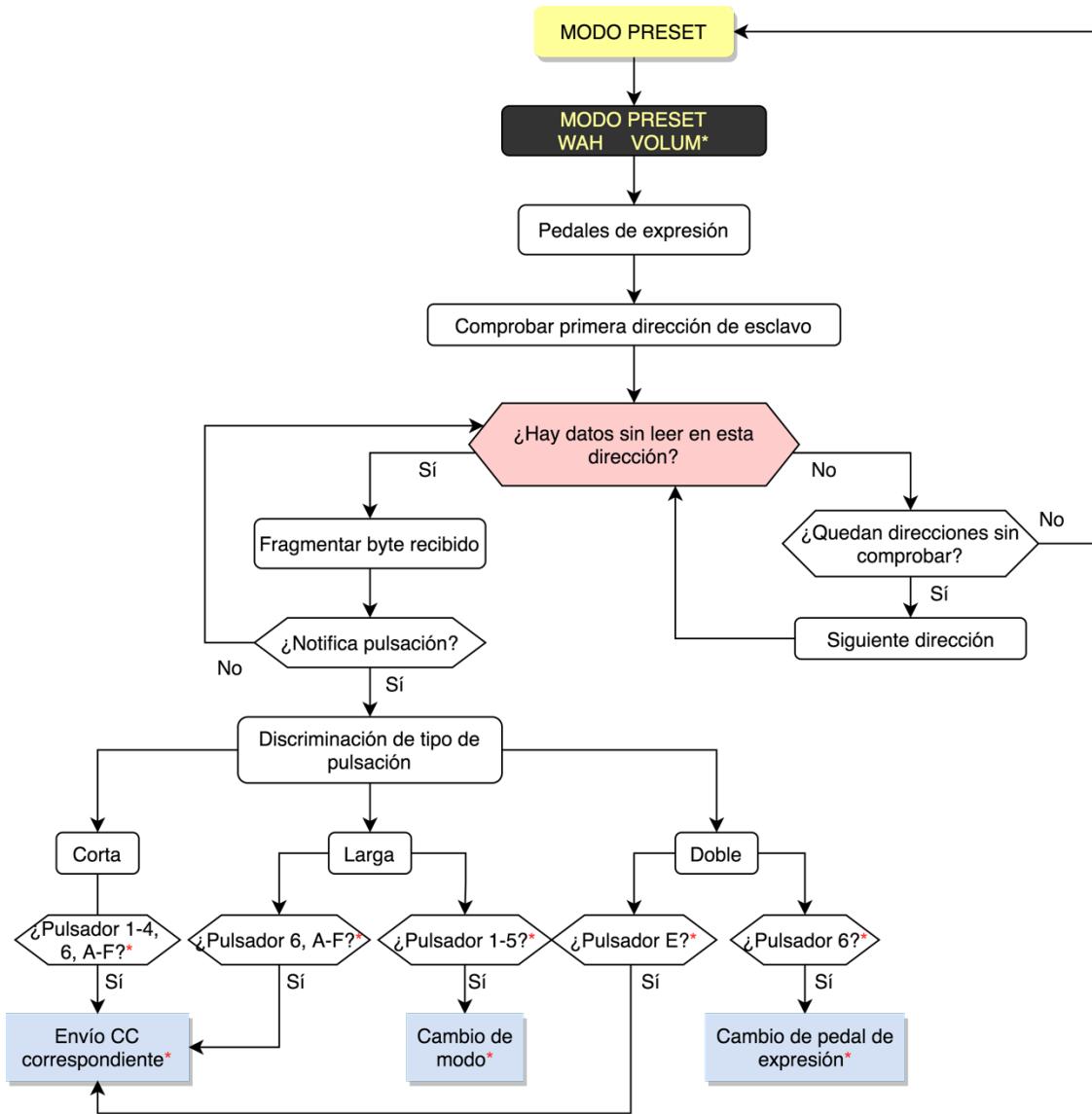
Cuando el microcontrolador empieza a funcionar, es decir, cuando se conecta a la red eléctrica mediante su fuente de alimentación, se crea el servidor, el servicio y la característica del *Bluetooth BLE*. De esta manera, el dispositivo queda preparado para iniciar una comunicación mediante *Bluetooth* en cualquier momento. Este código que pone en marcha el *Bluetooth* está basado en el creado por Neil Bags [9].

Seguidamente, se realiza una lectura de los valores guardados en la memoria *flash* y los asocia a las variables correspondientes. Se cargan los comandos CC asociados a las acciones disponibles en JamUP, los valores de la última calibración de los pedales de expresión, los del modo libre y el último *preset* cargado (y las luces encendidas asociadas).

A continuación, se consulta el modo en el que está el dispositivo. Por defecto, comienza en Modo *Preset*. Se recuerda que, para cambiar entre modos, se realiza una pulsación larga en uno de los pulsadores del 1 al 5, que corresponden respectivamente con Modo *Preset*, Modo *Jam*, Modo *Sampler*, Modo Libre y Configuración. En el caso de acceder al Menú de configuración se debe seleccionar opción: SLD (salvado del estado de los LEDs), LLD (carga del estado de los LEDs), CC (carga de comandos CC), ML (configuración del modo libre), EXP (calibración de pedales de expresión) y DSP (añadir dispositivo).

Modo *Preset*

En la Figura 53 se muestra la lógica del código que se ejecuta en bucle mientras que el ESP32 se encuentra en Modo *Preset*. Se puede observar, que cuando no quedan más direcciones de esclavos por comprobar, se vuelve a “inicio” y se ejecuta todo el código de nuevo. La lógica de los pedales de expresión se tratará a continuación por separado, ya que es un poco compleja para incluirla en el mismo diagrama y se entiende mejor si se explica en un diagrama aparte



Las cajas con el símbolo * redirigen a "¿Hay datos sin leer en esta dirección?" :

- Si son preguntas, cuando la respuesta es "No".
- Si son cajas azules, una vez que hayan realizado la acción.

Figura 53. Diagrama de pseudocódigo de Modo Preset.

En la pantalla LCD se muestra “MODO PRESET” en la fila de arriba y los efectos asociados al pedal de expresión en la fila de abajo. De esta manera, con un vistazo a la pantalla es posible saber los efectos de los que se dispone y si se necesita cambiarlos con una pulsación doble en el pulsador 6. Por defecto, se muestra “WAH” a la izquierda de la pantalla y “VOLUM” a la derecha. Si se cambia el de volumen por el de modificación de altura aparecerá “PITCH”. En el resto de los diagramas también se representan lo que se va a mostrar por la pantalla LCD en estos cuadros negros de letras amarillas.

Por otro lado, ¿a qué se refiere la caja del diagrama “fragmentar byte recibido”? Esta caja hace alusión al proceso de extracción de la información del byte de datos. Como se comentó en el apartado 4.2.1 Módulo Pulsador LED, el byte enviado transporta:

- El pulsador accionado en los 4 bits menos significativos.
- Si notifica una pulsación en el quinto y sexto bit. Nivel bajo en ambos indica que ha habido pulsación.
- La pulsación efectuada en los dos bits más significativos. 00 para corta, 01 para larga y 10 para doble.

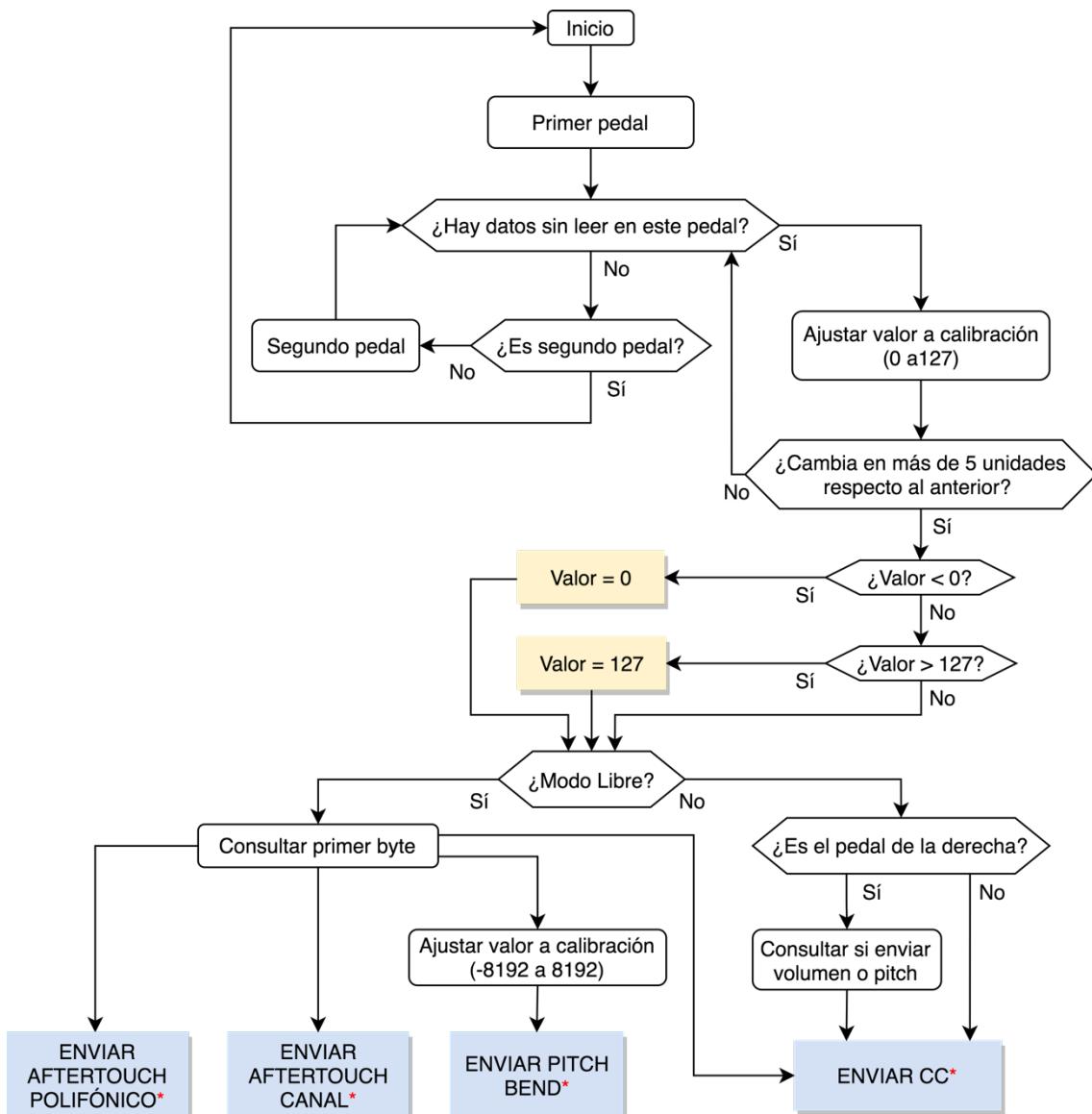
Hay una función que separa el byte en dos variables, una para el “pulsador de origen” del dato y otra para la “pulsación”, que también contiene los bits que indican si ha habido una pulsación real. Según estos valores se realizan las acciones que correspondan.

Para saber qué comando CC enviar en cada momento hay un *array* de comandos que se forma cuando se cargan los valores de la memoria *flash* y se puede sobrescribir cargando nuevos valores por medio del JSON de configuración. Después, suele haber 3 *arrays* por cada modo: uno con las pulsaciones cortas, otro con las largas y otro con las dobles. En cada uno de estos tres *arrays* el índice es el número de pulsador, numerados del 0 al 11. Los índices del 0 al 9 están asignados a los pulsadores con LED y el 10 y 11 al pulsador sin LED superior e inferior respectivamente. El valor que hay en esa posición es el índice del *array* de CC.

Véase el siguiente ejemplo. En el JSON de configuración hay una clave con un valor. Por ejemplo, la clave “AMP” con el valor “6”. Esto significa que es necesario enviar un CC6 para activar/desactivar el amplificador. Imagine que el valor asociado a la clave AMP se guarda en la posición 4 del *array* de CC (cc[4]). El amplificador está asociado a una pulsación corta en el primer pulsador de la fila superior. Por esta razón, cuando se realiza una pulsación corta, se consulta el valor del índice 0 en el array de pulsaciones cortas. En esta posición el valor que contiene es el 4. Así, al consultar cc[4], el resultado obtenido es 6. De esta manera, si se cargara un nuevo JSON de configuración y esta vez el amplificador se activara/desactivara con un comando CC10, al consultar cc[4] contiene el valor 10.

Sería más cómodo disponer de mapas en la que asociar clave y valor. Así podríamos mantener el nombre de los pulsadores como en el resto del proyecto, con números para la fila superior y letras para la fila inferior. Sin embargo, como no es posible utilizar mapas en el lenguaje Arduino, se tiene que utilizar el índice del *array* como clave para asociar los CC. Este procedimiento se utiliza también en Modo *Jam* y Modo *Sampler*.

A continuación, en el diagrama de la Figura 54 se explica el funcionamiento de los pedales de expresión, cuyo código se utiliza en Modo *Preset*, Modo *Sampler*, Modo *Jam* y Modo Libre. En los tres primeros modos solo se pueden enviar comandos MIDI *Control Change* (CC). En el último también es posible enviar *Pitch Bend*, *Aftertouch Canal* y *Aftertouch Polifónico*, dependiendo del primer byte asociado a EXP1 o EXP2 del JSON de configuración.



Las cajas con símbolo * redirigen a ¿hay datos sin leer en este pedal?

Figura 54. Diagrama de pseudocódigo de los pedales de expresión.

Modo Jam

Se representa la lógica de este modo por medio del diagrama de la Figura 55. Se observa que el código es muy similar al del Modo *Preset*, ya que la funcionalidad de este modo se debe mantener mientras el Modo *Jam* está en funcionamiento. El único cambio es que se tienen en cuenta pulsaciones dobles de más pulsadores, que son las que activan las funciones especiales que ofrece el modo. Los pedales de expresión, la carga de comandos CC y la fragmentación del byte recibido funciona de la misma manera ya expuesta cuando se explica el código del Modo *Preset*.

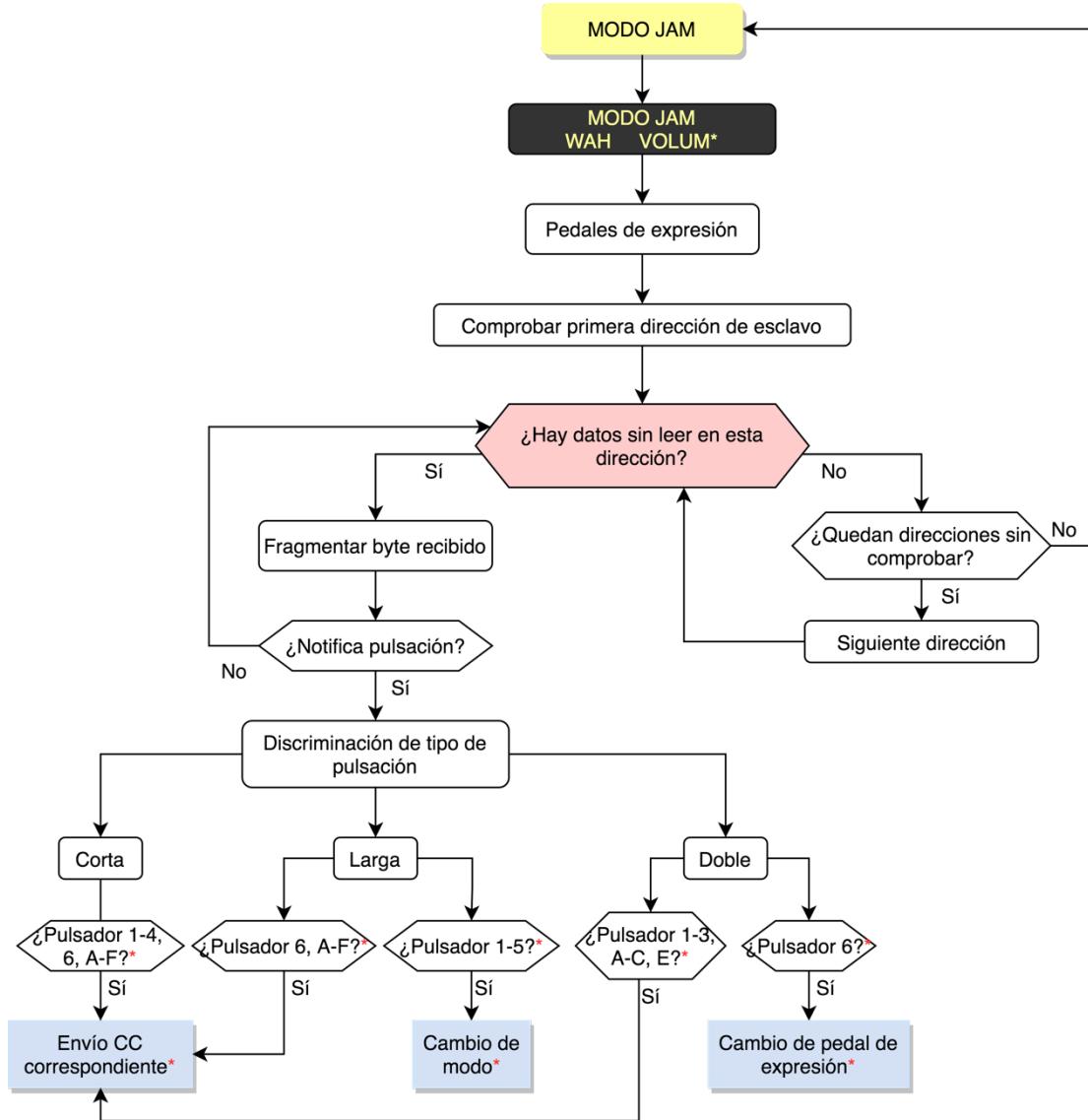


Figura 55. Diagrama de pseudocódigo del Modo Jam.

Modo Sampler

Se representa el código del Modo *Sampler* mediante el diagrama de la Figura 56. El código es prácticamente igual que el de Modo *Preset* y Modo *Jam*. La única modificación o cambio es que se tienen en cuenta pulsaciones dobles de pulsadores diferentes, que son las que activan las funciones especiales que ofrece el modo.

Los pedales de expresión, la carga de comandos CC y la fragmentación del byte recibido funciona de la misma manera ya expuesta cuando se explicó el código del Modo *Preset*.

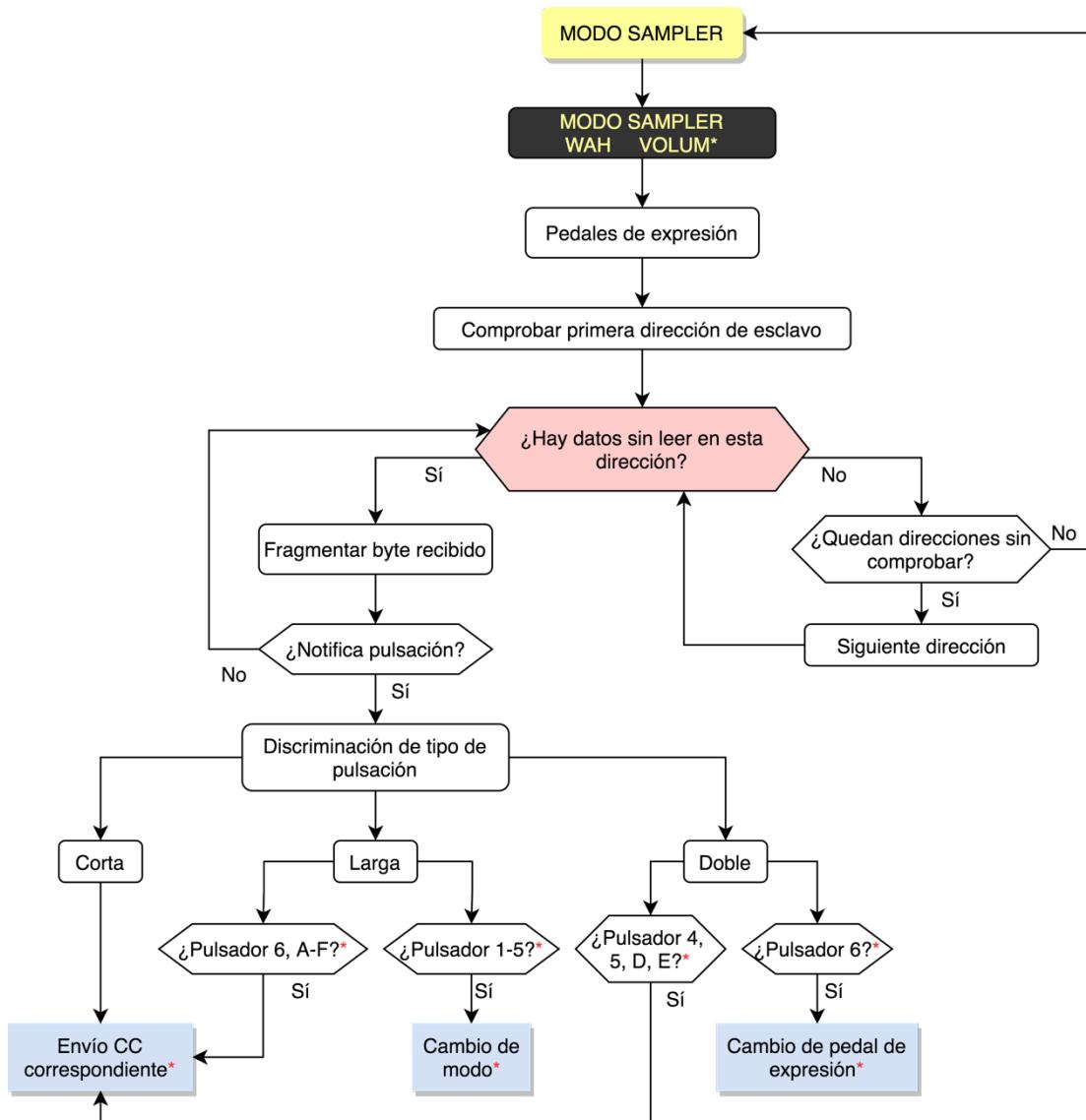
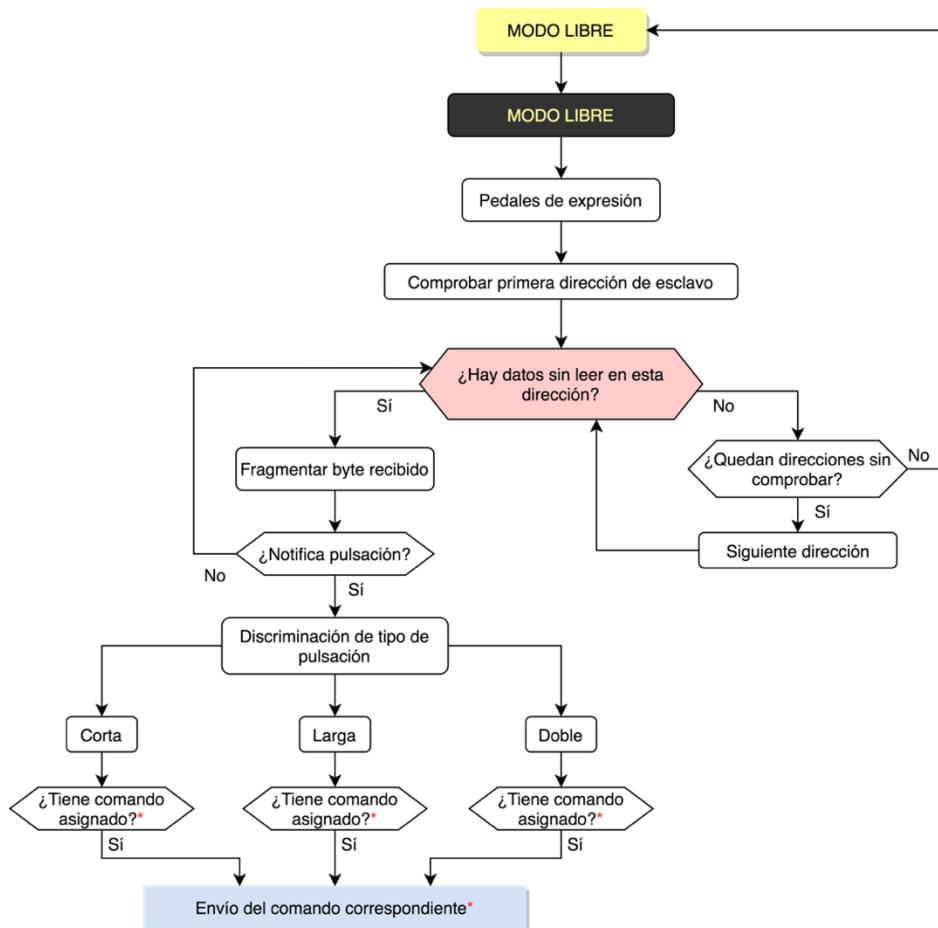


Figura 56. Diagrama de pseudocódigo del Modo Sampler

Modo Libre

El código del Modo libre sigue el esquema de los modos anteriores. Sin embargo, no se conoce qué pulsadores tendrán un comando asociado y cuáles no, ya que es algo que quedará a elección del usuario. Una vez se recibe la pulsación se comprueba si hay un comando que enviar y si lo hay, se envía.

Los comandos están almacenados en 3 *arrays*: uno para las pulsaciones cortas, otro para las largas y otro para las dobles. Cada uno de estos *arrays* tiene dos dimensiones: la primera, índices del 0 al 9 (referidos a los pulsadores) y la segunda, índices del 0 al 2 (referidos a los 3 bytes que se deben almacenar de cada pulsación). Así, para saber el valor de los 3 bytes de una pulsación corta en el pulsador A se consulta en el array de pulsaciones cortas (*shortPuls[x][y]*): *shortPuls[5][0]*, *shortPuls[5][1]* y *shortPuls[5][2]*. En la Figura 57 se muestra el diagrama de pseudocódigo para las pulsaciones del Modo Libre. Los pedales de expresión siguen la lógica ya expuesta en la Figura 54.



Las cajas con el símbolo * redirigen a "¿Hay datos sin leer en esta dirección?" :

- Si son preguntas, cuando la respuesta es "No".
- Si son cajas azules, una vez que hayan realizado la acción.

Figura 57. Diagrama de pseudocódigo del Modo Libre.

SLD

El salvado del estado de los LEDs, como se ha explicado en apartados anteriores, sirve para sincronizar las luces LED con los efectos activos de un *preset* determinado. En este modo de configuración se seleccionará uno de los 8 *presets* con estado de LEDs salvable (1A, 1B, 1C, 1D, 2A, 2B, 2C y 2D) por medio de 2 pulsadores de la fila superior (el 1 y el 2) y cuatro pulsadores de la fila inferior (del A al D). En la Figura 58 se puede observar el diagrama explicativo del código.

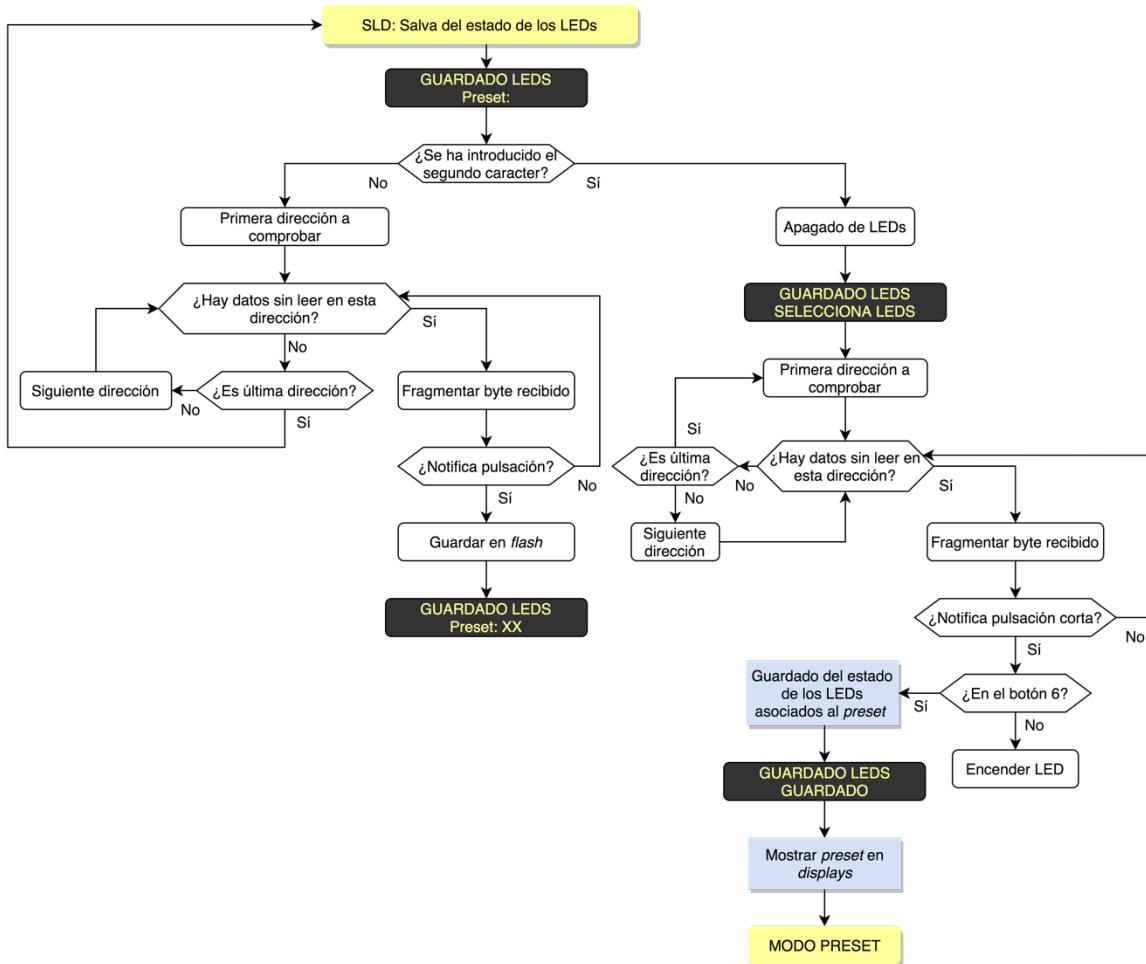


Figura 58. Diagrama de pseudocódigo del modo de configuración SLD.

Se pueden seleccionar los LEDs encendidos a guardar cuando se ha introducido el segundo carácter. Algo que no se aclara en el diagrama, porque ya quedaba demasiado extenso, es, que para la primera pulsación recibida, el máster solo tiene en cuenta la información recibida de los pulsadores 1 y 2. Para introducir el segundo carácter, solo se consideran las pulsaciones de los pulsadores A, B, C y D.

Cuando se acciona el pulsador aparece el carácter en la pantalla LCD. Así, si se quiere seleccionar el *preset* 2D, primero se pulsa el pulsador 2, y la segunda fila de la pantalla muestra "Preset: 2". Después se pulsa el pulsador D, y la segunda fila de la pantalla muestra "Preset: 2D".

En todo el programa principal, cuando se realiza la salva en la memoria *flash*, se realiza una lectura de la dirección en la que se va a guardar el dato porque, si el dato antiguo es el mismo que el nuevo, no se realiza la guarda. Esto se hace porque en la memoria *flash* es posible hacer todas las lecturas que se deseen, pero las operaciones de escritura están limitadas. Además, la operación de lectura es mucho más rápida que la escritura, por lo que no solo no se pierde mucho tiempo realizando estas comprobaciones, sino que incluso se puede llegar a ahorrar. Una vez realizado el guardado, se vuelve a Modo *Preset*.

En la memoria *flash* se dispone de 512 bytes en los que se pueden almacenar valores del 0 al 255. En nuestro proyecto se utilizan los primeros 210 bytes de la memoria. En la Figura 59, se muestra un esquema de cómo se ha repartido la información en estos bytes.

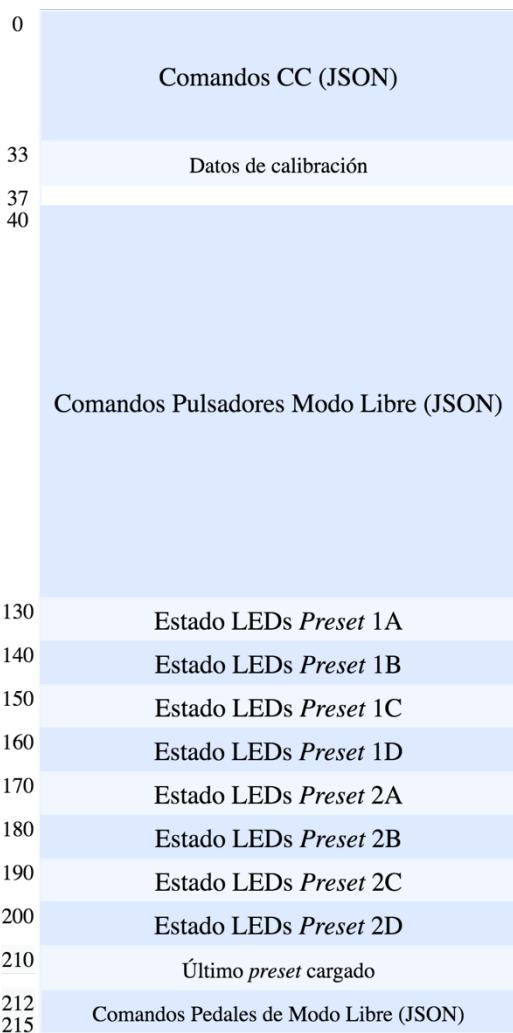


Figura 59. Esquema de contenido de los bytes ocupados en la memoria *flash* del microcontrolador ESP32.

LLD: carga de *presets*

Una vez salvados los *presets* con la opción SLD del menú de configuración, se necesita una opción para poder cargar los *presets*, es decir, mostrar la configuración de luces LEDs guardada y el nombre del *preset* en los *displays* de 7 segmentos.

En el diagrama de la Figura 60, se puede observar que la lógica de este modo es sencilla. Se introduce el nombre del *preset* de la misma manera que en el anterior (SLD). Entonces, se muestra en los *displays* y se realiza la lectura de la memoria de la posición en la que se encuentra la información sobre los LEDs a iluminar y se accionan. Por último, se vuelve a Modo *Preset*.

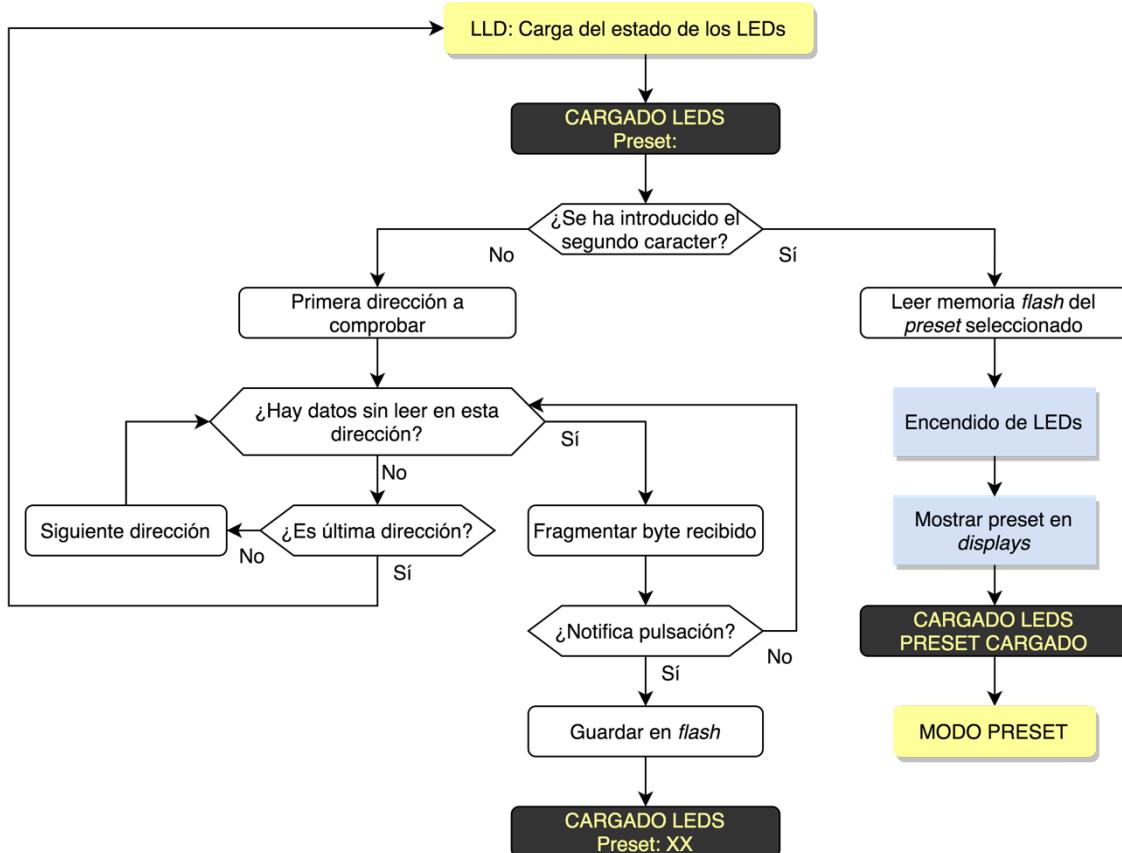


Figura 60. Diagrama de pseudocódigo del modo de configuración LLD.

Hay que destacar, que el número y la letra del *preset* se guardan en la memoria con el objetivo de que, si se apaga el dispositivo, cuando se vuelva a encender cargue el último *preset* que ha sido utilizado. Cuando se navega entre los *presets* con los pulsadores 6 y F, también se realiza este guardado si el *preset* está entre el 1A y el 2D del banco 1.

CC

Este modo de configuración simplemente realiza un guardado en la memoria de los comandos CC asociados a cada control. Estos comandos se cargan en el dispositivo mediante un JSON de configuración que ha sido previamente explicado en el apartado 4.1.1. Sin embargo, si se quiere que los comandos guardados se conserven cuando se apague el dispositivo es necesario seleccionar esta opción en el menú de configuración.

Simplemente se guardan en la memoria *flash* y aparece en la pantalla LCD el mensaje “GUARDADO” para indicar que la operación ha sido realizada con éxito. Por último, se regresa a Modo *Preset*.

ML: Modo Libre

De igual manera que en el modo de configuración CC, esta opción del menú de configuración realiza un guardado, esta vez de los comandos asociados a los pulsadores en Modo Libre. Estos comandos se guardan en *arrays* cuando se recibe el JSON de configuración y ya se encuentran en funcionamiento. Sin embargo, se diferencia entre estar cargados y estar guardados. Si se quieren conservar una vez se reinicie el dispositivo, se deben guardar en la memoria *flash*. Para ello se selecciona esta opción en el menú de configuración.

Cuando se haya realizado el guardado aparecerá en la pantalla LCD la palabra “GUARDADO” y se regresa a Modo *Preset*.

Cabe destacar que, si se carga un JSON, los comandos del anterior JSON cargado se borran. Es decir, si se ha cargado el JSON de configuración de Modo Libre y después se carga el de comandos CC, los comandos de Modo Libre se borran si no se han guardado. Para cargar ambos hay que guardar el primero o cargarlo todo en el mismo JSON.

EXP: Calibración de los pedales de expresión

Se realiza la calibración de los dos pedales de expresión a la vez. En primer lugar, se indica por pantalla que se pongan en posición de mínimo y se guarda el valor mínimo de los sesenta valores recibidos para cada pedal. Después, se indica en la pantalla LCD que se debe poner el valor máximo y se guarda el valor máximo entre los sesenta valores recibidos para cada pedal.

Los valores de mínimo y máximo iniciales son respectivamente 255 y 0 para cada uno de los dos pedales. De esta manera, el primer valor ya sustituirá al valor por defecto.

Con esos dos valores se realiza la recta de calibración con el objetivo de que los valores queden comprendidos entre 0 y 127, ya que en el comando MIDI se dispone de 7 bits para representar el valor. Para ello, se halla la pendiente (m) y el desplazamiento (n) de la recta a partir de los dos valores (mínimo y máximo) para cada pedal. Esto lo hallaremos mediante las ecuaciones 1 y 2, que se muestran a continuación.

$$m = \frac{127}{MÁX - MÍN} \quad (1)$$

$$n = m * MÍN \quad (2)$$

Cabe destacar que el desplazamiento de la recta (n) se halla con valor positivo para el posterior guardado pero la recta será de la forma $y = mx - n$. Para usar solo un byte en guardar el valor de la pendiente, en la que suelen ser importantes los decimales, se guarda el entero resultante de multiplicarla por cien. Se guardará siempre que el resultado no sea mayor que 255, cosa que, si la calibración se ha hecho correctamente, no va a ocurrir. La lógica en su totalidad se explica en el diagrama de pseudocódigo de la Figura 61.

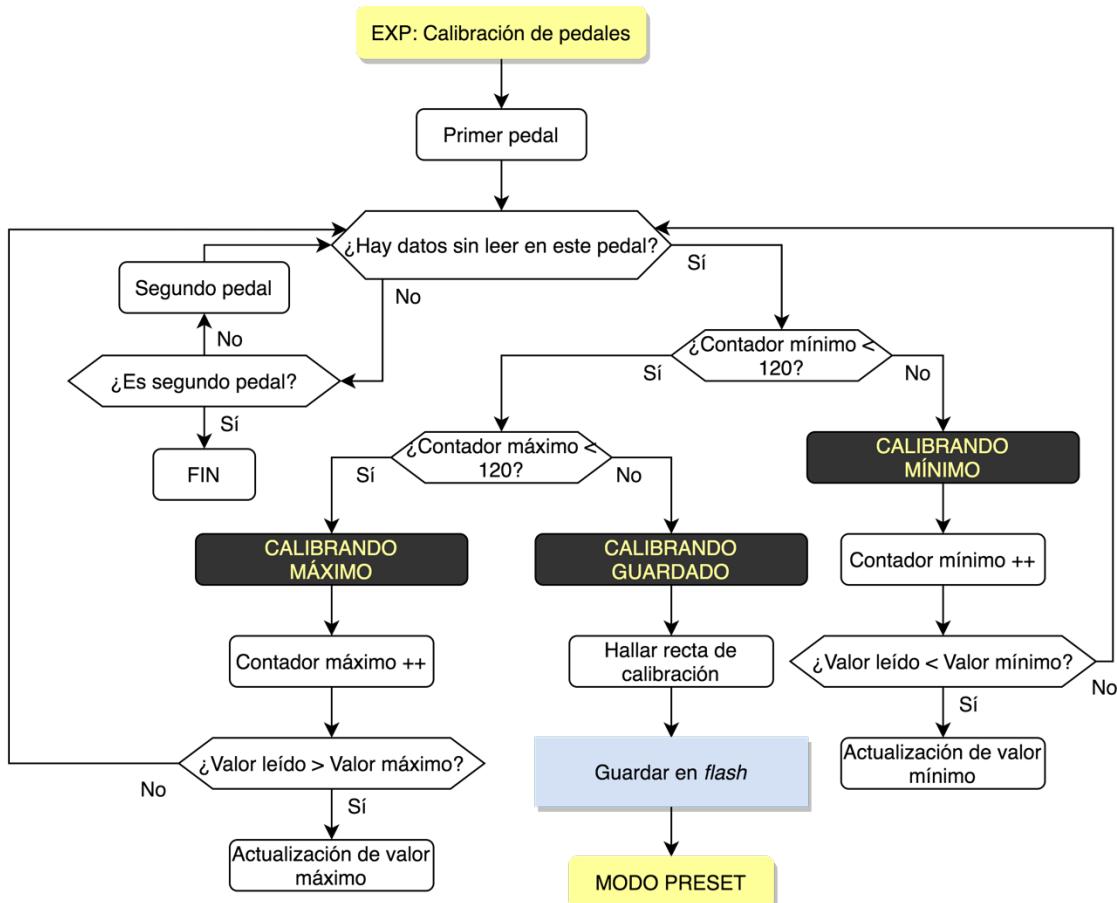


Figura 61. Diagrama de pseudocódigo del modo de configuración EXP.

Cabe destacar que, esta función está siendo llamada una y otra vez en bucle hasta que se finalice la calibración y se redirige al Modo *Preset*.

DSP: Añadir nuevo módulo

La última opción de configuración es la de añadir un nuevo módulo. Se puede utilizar para añadir dispositivos externos que se comuniquen por I2C. Actualmente el programa está preparado para recibir de estos nuevos dispositivos un comando MIDI y enviarlo por cable y Bluetooth. Sin embargo, hay un mundo de posibilidades dependiendo del módulo que se decida añadir al bus I2C.

Añadir el módulo es muy sencillo. Se accede a la opción DSP y, como se indica en la pantalla LCD, se selecciona la dirección que tendrá el dispositivo esclavo en el bus I2C. Para ello se asociarán números del 0 al 9 a los pulsadores, tal y como se ha hecho ya previamente. El número 0 corresponderá al pulsador 1 y el 9 al pulsador E.

Una vez introducida la dirección, la función que “escucha” al nuevo módulo y envía los comandos correspondientes se ejecuta sin importar el modo de funcionamiento en el que se encuentre el usuario. Esto ofrece la posibilidad de enviar comandos que no sean CC en el Modo *Preset*, el Modo *Jam* y el Modo *Sampler*. También se puede utilizar para añadir dispositivos más adecuados para enviar según qué tipo de comandos. Por ejemplo, se podría conectar una rueda para enviar comandos *Pitch Bend*.

Para no comprometer la estabilidad del bus, el programa actual solo admite un dispositivo adicional, que no se guarda si se apaga el controlador. Es decir, se tendrá que acceder a la opción de configuración DSP para añadir el módulo adicional cada vez que se encienda la pedalera.

5 Resultados

Una vez finalizado el proyecto, se realiza un análisis de los cambios significativos y cuantitativos que proporciona el nuevo dispositivo respecto al controlador original.

En primer lugar, la nueva pedalera está totalmente adaptada a su uso con la aplicación JamUp. Navegando por Internet, la gran parte de los proyectos de modificación de la pedalera Behringer FCB1010 enfocan su uso a unidades multiefectos. Orientar este proyecto al uso de la máquina con este tipo de *software*, no solo se considera atractivo para la comunidad de usuarios existente, sino que puede despertar interés en usuarios potenciales.

El uso con la aplicación es muy intuitivo. Tiene funcionalidad *stompbox* (posibilidad de asignar un pulsador a un efecto). Además, una mejora destacable es que las luces LED asociadas a los pulsadores se mantienen encendidas cuando en efecto está activo. Sin embargo, en el dispositivo original, cuando se pulsa un pulsador la luz LED solo se enciende un momento para indicar que el pulsador había sido accionado. Que la luz permanezca encendida indicando el estado del efecto hace de la pedalera una máquina mucho más cómoda de usar.

Asimismo, se añade una pantalla LCD, en la que se visualiza el modo de funcionamiento en el que se encuentra la pedalera y los efectos asignados a los pedales de expresión. Además, se mantienen los 3 *displays*, que van a mostrar el *preset* y el banco de la aplicación que se está utilizando. En resumen, se intenta trasladar la información de la aplicación a la pedalera. De esta manera es mucho más sencillo interactuar con ella, ya que solo es necesario focalizar la atención en un dispositivo.

Por otro lado, se implementa la conexión con la aplicación mediante Bluetooth BLE, aunque se mantiene también la conexión por el conector DIN de 5 pines. La conexión por Bluetooth supone una mejora reseñable ya que ahorra al usuario utilizar un dispositivo intermedio para conectar la pedalera al móvil o iPad. Así, para utilizar este procesador de efectos solamente hace falta conectar el *jack* de la guitarra al móvil mediante una interfaz de audio tipo iRig.

Asimismo, la configuración también se realiza por Bluetooth de manera muy sencilla. Una vez el usuario se familiariza con el envío de datos al microcontrolador ESP32 por Bluetooth y con la estructura de los JSON, es cuestión de segundos configurar la pedalera, sobre todo para cargar los comandos de la aplicación JamUp.

Otro punto positivo es que se tiene una visión global de la configuración que se va a cargar. En el dispositivo original se introducen uno a uno los comandos mediante

un procedimiento largo y tedioso. Al cabo de un tiempo configurándola es difícil recordar qué comandos se han introducido y a qué pulsador están asociados.

Se tiene también la posibilidad de configurar los LEDs eligiendo cuáles de ellos tienen que estar encendidos en según qué *presets*. En el dispositivo original, el uso de los LEDs es algo confuso y poco personalizable.

Sin embargo, esta pedalera no está limitada a su uso con JamUp. También cuenta con un modo libre con gran flexibilidad, en el que se pueden lanzar 25 comandos diferentes con los pulsadores y 2 comandos con los pedales de expresión. Los comandos de este modo también tienen la posibilidad de ser enviados por Bluetooth.

La configuración de este modo es un poco más compleja que la de los comandos de JamUp dado que no se quiere limitar a comandos ni canales determinados. Por tanto, debe contener más información. De todas formas, es, sin duda, mucho más sencilla y rápida que la configuración original y se realiza por Bluetooth.

Por otra parte, el nuevo dispositivo es un sistema distribuido. Esto implica un gran número de ventajas. En primer lugar, ante posibles fallos o averías. El fallo de un módulo secundario no impide el uso de la pedalera. Por esta razón, suscita mayor confianza a la hora de utilizar este controlador en una actuación en directo. Asimismo, a la hora de solucionar una avería es mucho más fácil aislar dónde se encuentra el problema. Y, finalmente, es mucho más fácil extraer el módulo para repararlo o sustituirlo por uno nuevo. Este tipo de reparaciones resultan también mucho más económicas. Igualmente, se pueden añadir módulos adicionales en cualquier momento. El requisito es que tengan un conector RJ11 al que poder conectar un cable telefónico que lo una con el conector sobrante que se encuentra en el dispositivo.

Este sistema modular estimula el continuo desarrollo del proyecto, ya que es muy fácil añadir nuevos módulos, intercambiarlos o introducir mejoras en los ya integrados. Así, el usuario puede adaptar el dispositivo a sus necesidades en cada momento, consiguiendo cada vez una mayor personalización de la pedalera.

6 Presupuesto

Al igual que en el resto del trabajo, se trata por separado cada módulo. Para cada uno, se tiene en cuenta:

- Materiales del montaje de prueba. Antes de construir la placa que finalmente se integra en el dispositivo, se realizan ensayos de cada módulo en una placa de pruebas.
- Materiales del montaje final.
- Mano de obra. Se consideran tanto las horas de trabajo del diseño y construcción de módulos, como las de programación y pruebas de estos.

Hay que destacar que gran parte de los materiales se compran en páginas del mercado asiático, como Aliexpress, con el objetivo de conseguir que el desarrollo del proyecto resulte lo más económico posible.

6.1 Módulo Pulsador LED

En la Tabla 4 se muestra una relación de los materiales necesarios para construir el Módulo Pulsador LED asociados a su precio por unidad y precio total. Se realiza una separación entre el material que se adquiere para realizar las pruebas previas y que no se utiliza en el dispositivo final (“placa desarrollo”) y el que se compra para la placa final que se integra en el dispositivo.

Hay que destacar, que para la fase de desarrollo se adquiere material para construir 10 de estas placas porque se pretende hacer pruebas de I2C con todos los pulsadores antes de integrarlo. Finalmente se comprueba que es mejor realizar este tipo de pruebas con las placas finales integradas conectadas ya por cable telefónico, por lo que con haber comprado material para desarrollar un Módulo Pulsador LED hubiese sido suficiente.

Aún así, se observa que la fase de desarrollo no tiene un coste significativo dentro del proyecto, ya que entre el precio de las 10 placas integradas y el total (sumándole los materiales necesarios para las pruebas previas) no alcanza los 4 euros de diferencia.

Tabla 4. Materiales necesarios para la construcción de los diez Módulos Pulsador LED

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
Attiny85	Ambas	1,60 €	10	16,00 €
Resistencia 120 Ohm	Ambas	0,03 €	10	0,30 €
Resistencia 10 kOhm	Ambas	0,03 €	10	0,30 €
Resistencia 4,7 kOhm	Ambas	0,03 €	20	0,60 €
Pulsador	Desarrollo	0,40 €	10	4,00 €
Diodo LED rojo	Desarrollo	0,08 €	10	0,80 €
Bloques terminales	Final	0,30 €	12	3,60 €
Conector hembra RJ11	Final	0,67 €	20	13,40 €
Placa PCB 4x6 cm	Final	0,40 €	5	2,00 €
PRECIO 10 PLACAS (€)				36,20 €
PRECIO TOTAL (€)				41,00 €

Además del precio material, hay que añadir la mano de obra. Se estima que el coste por hora de un ingeniero junior es de 15 euros. Para la construcción y programación de este módulo se dedican unas 60 horas de trabajo, como se puede observar en la Tabla 5. El tiempo dedicado a la interconexión de módulos se tiene en cuenta en el presupuesto de mano de obra del Módulo Principal.

Tabla 5. Coste de la mano de obra para el desarrollo y construcción del Módulo Pulsador Led.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo Pulsador LED	60	15	900,00 €

6.2 Módulo Pulsador

En la Tabla 6 se muestra una lista de los precios de los materiales necesarios para la fase de desarrollo y posterior construcción de Módulo Pulsador.

Tabla 6. Materiales necesarios para la construcción de los dos Módulos Pulsador.

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
Attiny85	Ambas	1,60 €	2	3,20 €
Resistencia 10 kOhm	Ambas	0,03 €	2	0,06 €
Resistencia 4,7 kOhm	Ambas	0,03 €	4	0,12 €
Pulsador	Desarrollo	0,40 €	2	0,80 €
Bloques terminales	Final	0,30 €	2	0,60 €
Conector hembra RJ11	Final	0,67 €	4	2,68 €
Placa PCB 4x6 cm	Final	0,40 €	1	0,40 €
PRECIO 2 PLACAS FINALES (€)				7,06 €
PRECIO TOTAL (€)				7,86 €

Como el desarrollo y programación son muy similares de las del Módulo Pulsador LED, no se le dedica demasiado tiempo. Se estima que unas 8 horas, como queda recogido en la Tabla 7.

Tabla 7. Coste de la mano de obra para el desarrollo y construcción del Módulo Pulsador.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo Pulsador	8	15	120,00 €

6.3 Módulo Pedal de Expresión

En la Tabla 8 se observa el coste de los dos Módulos Pedal de Expresión que tiene el dispositivo. Se tienen en cuenta los materiales necesarios tanto para la realización de la fase de pruebas y como para la construcción de las placas integradas en el dispositivo final.

Tabla 8. Materiales necesarios para la construcción de los dos Módulos Pedal de Expresión.

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
Attiny85	Ambas	1,60 €	2	3,20 €
Resistencia 4,7 kOhm	Ambas	0,03 €	4	0,12 €
Resistencia 120 Ohm	Ambas	0,03 €	2	0,06 €
Resistencia 10 kOhm	Desarrollo	0,03 €	2	0,06 €
Diodo LED rojo	Desarrollo	0,08 €	2	0,16 €
Potenciómetro	Desarrollo	0,41 €	2	0,82 €
Bloques terminales	Final	0,30 €	2	0,60 €
Conector hembra RJ11	Final	0,67 €	4	2,68 €
Placa PCB 4x6 cm	Final	0,40 €	1	0,40 €
PRECIO 2 PLACAS FINALES (€)				7,06 €
PRECIO TOTAL (€)				8,10 €

Como se muestra en la Tabla 9, se estima que en este módulo se emplean unas 25 horas de trabajo.

Tabla 9. Coste de la mano de obra para el desarrollo y construcción del Módulo Pedal de Expresión.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo Pedal de Expresión	25	15	375,00 €

6.4 Módulo *Display*

En la Tabla 10 se encuentra el listado de materiales necesarios para la construcción de este módulo. Se puede observar que todo lo adquirido para la fase de desarrollo sirve para la integración final.

Tabla 10. Materiales necesarios para la construcción del Módulos Display.

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
Attiny85	Ambas	1,60 €	1	1,60 €
Resistencia 4,7 kOhm	Ambas	0,03 €	2	0,06 €
Resistencia 470 Ohm	Ambas	0,03 €	24	0,72 €
Displays Ánodo común	Ambas	1,65 €	3	4,95 €
Registro de desplazamiento	Ambas	0,55 €	3	1,65 €
Conector hembra RJ11	Ambas	0,67 €	2	1,34 €
Placa PCB 9x15 cm	Final	4,00 €	1	4,00 €
PRECIO TOTAL (€)				14,32 €

Respecto a la mano de obra, como se observa en la Tabla 11, se estima que se emplean unas veinte horas. Gran parte fueron dedicadas a su construcción.

Tabla 11. Coste de la mano de obra para el desarrollo y construcción del Módulo Display.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo Display	20	15	300,00 €

6.5 Módulo LCD

Este módulo, como se observa en la Tabla 12, es el más sencillo. Simplemente hay que añadirle el pequeño arreglo que se hace para instalar el conector RJ11.

Tabla 12. Materiales necesarios para la construcción del Módulo LCD.

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
Pantalla LCD	Ambas	2,53 €	1	2,53 €
Conector hembra RJ11	Ambas	0,67 €	1	0,67 €
Placa PCB 4x6 cm	Ambas	0,40 €	1	0,40 €
PRECIO TOTAL (€)				3,60 €

En cuanto a la mano de obra, como se muestra en la Tabla 13, se estima que se dedican unas 4 horas. La razón de haber invertido tan poco tiempo es que se adquirió una pantalla que ya venía preparada para su uso con I2C a la que solo se

le tuvo que instalar el conector RJ11. Además, es sencillo ponerla en funcionamiento, ya que existen librerías específicas para ese tipo de pantalla.

Tabla 13. Coste de la mano de obra para el desarrollo y construcción del Módulo LCD.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo LCD	4	15	60,00 €

6.6 Módulo principal

En la Tabla 14 se observa el listado de materiales adquiridos para la construcción del Módulo Principal. Como ocurría también en el Módulo *Display*, para la fase de desarrollo no se adquiere ningún material adicional.

Tabla 14. Materiales necesarios para la construcción del Módulo Principal.

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
ESP32	Ambas	4,75 €	1	4,75 €
Conector DC hembra	Ambas	0,55 €	1	0,55 €
Convertidor Niveles Lógicos	Ambas	3,60 €	1	3,60 €
Conector hembra RJ11	Final	0,67 €	4	2,68 €
Placa PCB 9x15 cm	Final	4,00 €	1	4,00 €
PRECIO TOTAL (€)				15,58 €

Este módulo es en el que más tiempo se invierte. No por su construcción, que es bastante sencilla, sino por su programación. Además, se consideran, en este cálculo de horas, las dedicadas a la interconexión entre módulos mediante el protocolo I2C, con el que surgieron ciertos problemas.

Por otro lado, es un módulo que cuenta con toda la lógica de cada uno de los modos de funcionamiento que tiene el dispositivo. Para cada uno de ellos, no solo se tiene en cuenta su programación, sino también la fase de diseño de la funcionalidad del dispositivo.

Por estas razones, como se observa en la Tabla 15, se estima que se dedicaron unas 200 horas.

Tabla 15. Coste de la mano de obra para el desarrollo y construcción del Módulo Principal.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo Principal	200	15	3.000,00 €

6.7 Presupuesto total

A la hora de calcular el presupuesto material total del proyecto, además de los componentes que se encuentran en los distintos módulos, hace falta adquirir otros materiales que no son atribuibles a un módulo en concreto. Por ejemplo, los distintos cables, tanto los que se utilizan en la placa de pruebas, como el cable que se utiliza en los módulos finales o el cable telefónico que une los módulos. En la Tabla 16 aparece un listado de los materiales adicionales necesarios para el desarrollo y construcción del proyecto.

Tabla 16. Presupuesto de material adicional para el desarrollo y construcción del proyecto.

Material	Placa	Precio por unidad (€)	Cantidad	Precio total (€)
Protoboard	Desarrollo	5,50 €	3	16,50 €
Cable de puente para placa de pruebas (8m)	Desarrollo	3,00 €	1	3,00 €
Fuente alimentación	Final	8,00 €	1	8,00 €
Cable telefónico (1m)	Final	0,21 €	4	0,84 €
Conectores macho RJ11	Final	0,10 €	32	3,20 €
Crimpadora	Final	9,95 €	1	9,95 €
Cable PCB wrapping o "de envoltura"	Final	4,83 €	1	4,83 €
Tornillos con tuerca para sujeción	Final	2,50 €	1	2,50 €
Arduino UNO	Ambas	3,90 €	1	3,90 €
Pedalera FCB1010 (segunda mano)	Final	60,00 €	1	60,00 €
PRECIO PLACAS FINALES (€)				93,22 €
PRECIO TOTAL (€)				112,72 €

Cabe destacar la adquisición de un Arduino UNO, que es muy útil a la hora de localizar problemas. Al final, este es el microcontrolador con lenguaje Arduino del que más información se dispone en la red. En muchas ocasiones, para la resolución de problemas, se parte del módulo funcionando con el Arduino UNO. Además, se utiliza para programar los Attiny85.

Otro gasto destacable es la crimpadora. Se necesita cable telefónico con una longitud muy concreta para unir los distintos módulos sin que sobre cable. Se compra cable telefónico, conectores RJ11 y la crimpadora para fabricarlos manualmente.

A continuación, en la Tabla 17 se suman los costes materiales de los distintos módulos y el material adicional adquirido para el desarrollo del proyecto.

Tabla 17. Desglose del coste material del proyecto.

Material	Precio dispositivo final (€)	Precio total (€)
Módulos Pulsador LED	36,20 €	41,00 €
Módulos Pulsador	7,06 €	7,86 €
Módulos Pedal de Expresión	7,06 €	8,10 €
Módulo Displays	14,32 €	14,32 €
Módulo LCD	3,60 €	3,60 €
Módulo Principal	15,58 €	15,58 €
Material adicional	93,22 €	112,72 €
COSTE MATERIAL DEL DISPOSITIVO FINAL (€)		177,04 €
COSTE MATERIAL DEL PROYECTO (€)		203,18 €

Si se realiza una comparación entre el coste material del proyecto final (sin fase de pruebas) y el coste material total se puede concluir que la diferencia es de unos 25 euros. El mayor coste de esta fase de desarrollo fueron las placas de pruebas o *protoboards* (16,50€).

A este coste material, hay que sumarle el coste de la mano de obra total, que aparece desglosado por módulos en la Tabla 18.

Tabla 18. Coste total de la mano de obra de todo el proyecto desglosado por módulos.

Módulo	Horas de trabajo	Coste por hora (€)	Coste total (€)
Módulo Botón Led	60	15	900,00 €
Módulo Botón	8	15	120,00 €
Módulo Pedal de Expresión	25	15	375,00 €
Módulo Display	20	15	300,00 €
Módulo LCD	4	15	60,00 €
Módulo Principal	200	15	3.000,00 €
HORAS TOTALES			317
PRECIO TOTAL (€)			4.755,00 €

Por último, se suma el coste material al coste de la mano de obra y se obtiene el coste total del proyecto, que queda recogido en la Tabla 19.

Tabla 19. Coste total del proyecto

COSTE MATERIAL DEL PROYECTO (€)	177,04 €
COSTE DE MANO DE OBRA DEL PROYECTO	4.755,00 €
COSTE TOTAL DEL PROYECTO	4.932,04 €

7 Conclusiones

El punto de partida es un dispositivo robusto y económico con una lógica muy limitante y una configuración “tediosa” a la que muchos usuarios no consiguen hacer frente. Se realiza una investigación por foros especializados para conocer las modificaciones de código abierto compartidas por los usuarios e identificar las necesidades de estos. Con esta información se plantean los requisitos del proyecto.

Se propone la adaptación de la pedalera a su uso con la aplicación JamUp, a la que se conecta por Bluetooth (para evitar el uso de otros dispositivos especiales). Se considera que los resultados respecto a este punto son satisfactorios. La conexión es muy estable, es decir, no hay pérdida de mensajes, interrupciones, ni fallos. Además, se dispone de suficientes pulsadores y pedales para enviar comandos con cierta flexibilidad y comodidad. Su distribución también es un punto a favor, ya que la asociación entre funciones disponibles y pedales resulta intuitiva para el usuario.

Poder asociar un pulsador con LED a un efecto era uno de los puntos centrales del proyecto. Para ello, no se encontró dificultad técnica, pero sí de diseño, ya que había que adaptar el resto de las funcionalidades de la pedalera, reservando las pulsaciones cortas de estos 8 pulsadores con LED asociados a los efectos.

Asimismo, se busca centralizar la interacción con el *software* en la nueva pedalera. Es decir, no obligar al usuario a que esté pendiente de 2 dispositivos diferentes (la *tablet* y el controlador) durante una interpretación. Para ello, se intenta trasladar la mayor información posible al controlador. Se utilizan 3 *displays* de 7 segmentos para mostrar el banco y el *preset* en curso. Además, como se ha comentado, los LEDs se utilizan para mostrar los efectos activos y distinguir las pulsaciones efectuadas. Por último, la pantalla LCD muestra los efectos en curso en los dos pedales de expresión y el modo de funcionamiento en el que se encuentra la pedalera. Esto implica una mejora significativa en cuanto a comodidad de uso del dispositivo.

De igual modo, se incluye la opción de configurar el estado de los LEDs de 8 *presets*. Así, cuando se cambie a uno de estos *presets* se encenderán los LEDs cuyo estado ha sido guardado, mostrando los efectos que están activos. Esto hace que la experiencia de uso sea muy parecida a la de las pedaleras multiefectos, cuya información está centralizada en un solo dispositivo.

Por otro lado, otro de los objetivos es facilitar la configuración. Se considera que se ha cumplido satisfactoriamente. La configuración actual del dispositivo es mucho más sencilla y rápida y, además, se realiza de una manera cómoda por Bluetooth.

Cabe destacar que mediante el fichero JSON se tiene una visión global de lo que va a realizar el dispositivo y se pueden revisar las funciones que se han cargado en la pedalera.

Durante el desarrollo del proyecto se plantea simplificar la configuración del Modo Libre. Sin embargo, esto supone restarle flexibilidad al modo. Como el resto del dispositivo está tan orientado a su uso con la aplicación, se prioriza tener cierta libertad en este modo. De todas formas, aunque resulte un poco más tediosa que la carga de comandos para la aplicación JamUp, una vez te familiarizas con la estructura de JSON y los comandos MIDI, es mucho más completa y sencilla que la del dispositivo original.

Se decide construir el controlador siguiendo una estructura distribuida. Resulta interesante porque, además de proporcionar ventajas a la hora de localizar o reparar averías, da la oportunidad de añadir nuevos módulos que acepten este protocolo. Asimismo, atendiendo a la naturaleza del proyecto, es ideal para seguir incorporando mejoras y funcionalidades manteniendo lo ya desarrollado en funcionamiento.

Aunque a priori dividir el trabajo por módulos simplifica el abordaje del proyecto, se ha tenido que hacer frente muchos problemas a la hora de implantar el protocolo para la interconexión módulos. Si bien en este proyecto no existen limitaciones reales en cuanto a número de módulos a conectar al bus, es un protocolo pensado para la conexión de dispositivos en la misma placa. Por esta razón se intentó que los diferentes módulos estuviesen lo más cerca posible y que no hubiese mucha longitud de cable entre ellos. Además, se añadieron las resistencias de 4700 ohmios para dar estabilidad a los niveles de las señales SDA y SCL. De esta manera se consiguió que funcionase la interconexión de los 15 módulos actualmente instalados en la pedalera.

El foro oficial de Arduino es bastante optimista en cuanto al rendimiento del I2C y expone que el funcionamiento es correcto en buses de hasta 5 o 6 metros. Sin embargo, en este proyecto se limita la agregación de nuevos módulos a un solo dispositivo, ya que la longitud total del bus aumenta, y esto puede desestabilizar las señales. Se considera que puede disminuir la calidad de la señal el gran número de conectores del bus y que su construcción podría ser más lograda. Realizar mejoras como la sustitución de los actuales módulos por circuitos impresos repercutiría positivamente en la estabilidad de las señales y podría implicar que fuese posible aumentar la longitud del bus, es decir, añadir más módulos.

Por otra parte, aunque en el proyecto se tomen ciertas decisiones en pos de utilizar este dispositivo en directo, se estima que es necesario que se realicen más pruebas antes de confiar en la pedalera para una actuación en directo. Sería muy beneficioso

que distintos músicos pudiesen hacer uso del controlador durante cierto periodo de tiempo y que fuesen ellos los que planteasen fallos y posibles mejoras.

En resumen, se considera que el controlador MIDI desarrollado cumple los requisitos planteados. En cambio, es importante recalcar que el objetivo del proyecto es ofrecer a la comunidad de usuarios de la pedalera FCB1010 una modificación que pueden realizar ellos mismos, adaptándola a sus necesidades. Se propone una estructura distribuida, un diseño atractivo y unas nuevas características que se piensa que podrían resultar beneficiosas.

Sin embargo, el proyecto queda totalmente abierto a distintas modificaciones. A continuación, se plantean algunas que se piensa que podrían encajar:

- Incorporación de batería. En este proyecto se elimina la necesidad del cable DIN de 5 pines para la conexión MIDI con el teléfono móvil o el iPad. El único cable saliente de la pedalera es el de alimentación. Si el usuario busca independencia, una posible mejora es incorporar una batería que sustituya a la fuente de alimentación.
- Cambio de microcontrolador en el Módulo Pedal de Expresión. Como se comenta en el apartado 4.1.1 “Funcionamiento del dispositivo”, el Attiny solo puede realizar lecturas analógicas de 10 bits. Para el correcto funcionamiento del comando *Pitch Bend* se necesita un microcontrolador que realice lecturas analógicas de 14 bits.
- Memoria SD. El espacio de memoria no volátil con el que se cuenta es muy reducido. Hay módulos de memorias SD que se pueden conectar por I2C. Esto supondría añadir un módulo más a la pedalera y permitiría, por ejemplo, poder guardar el estado de los LEDs de más *presets* o guardar distintas configuraciones para el Modo Libre.
- Añadir varios *presets* al Modo Libre. Si se realiza la modificación de la tarjeta SD ya no se tendrá una limitación tan grande de espacio para guardar variables. Esto permite crear varios *presets* en el Modo Libre, cada uno podrá contar con hasta 27 comandos: 25 en los pulsadores y 2 en los pedales de expresión.
- LEDs de colores. Se pueden utilizar para varias funciones según las necesidades del usuario. Por ejemplo, se puede asociar un color de LED a un modo. De esta manera será mucho más visual el modo en curso.
- *Switch* (conmutador) exterior para cambiar de modo. Se toma la idea de esta mejora del dispositivo “FCBInfinity”, una de las modificaciones de la pedalera celebradas en la comunidad. Se habla brevemente de ella en el apartado 2.4 “Pedalera Behringer FCB1010” cuando se tratan los proyectos de modificación de la pedalera encontrados. Este incorpora un pulsador al lado del pulsador 1 para cambiar rápidamente entre los distintos modos. Una de las ventajas es que este nuevo pulsador permitiría no tener que reservar las pulsaciones largas de los pulsadores de la fila superior para

cambiar entre los modos, teniendo así más pulsaciones asignables en Modo Libre.

- Mensajes de error: se considera positivo implementar un mayor control de los fallos y añadir mensajes de error explicativos en la pantalla LCD.
- Mejorar el control de *presets*. Si se necesita poder acceder a los *presets* marcando su número y su letra, se podría añadir la posibilidad de enviar una combinación de comandos MIDI. Por ejemplo, si desde el *preset* 1A se marca el 1C, el programa envía dos comandos PRESET+ para llegar hasta el 1C.
- Modo híbrido. Podría resultar interesante incorporar un modo híbrido entre Modo *Preset* y Modo Libre. Pudiendo asignar a ciertas pulsaciones del Modo *Preset* un comando cualquiera.

En definitiva, se concluye que los resultados de este proyecto son satisfactorios, aunque con margen de mejora. Se confía en que el proyecto resulte inspirador para los miembros de la comunidad creada alrededor de la pedalera Behringer FCB1010 y los incite a seguir este diseño e incorporar las mejoras que consideren oportunas. El verdadero sentido de este proyecto es, que a partir del momento en que se comparta, tome vida propia y sirva de lienzo para todo lo que sus usuarios quieran añadir. Porque... ¿qué otra filosofía podría tener un proyecto sobre un protocolo como el MIDI?

8 Referencias

- [1] Wikipedia, «Saxofón,» 24 marzo 2021. [En línea]. Disponible: <https://es.wikipedia.org/wiki/Saxof%C3%B3n#Historia>. [Consultado en febrero de 2021].
- [2] Construmática, «Ley de Faraday,» [En línea]. Disponible: https://www.construmatica.com/construpedia/Ley_de_Faraday. [Consultado en febrero de 2021].
- [3] J. A. Álvarez, «El estándar MIDI cumple 30 años - una breve historia de sus inicios y evolución,» 28 enero 2013. [En línea]. Disponible: <http://www.futuremusic-es.com/el-estandar-midi-cumple-30-anos-una-breve-historia-de-sus-inicios-y-evolucion/>. [Consultado en febrero de 21].
- [4] BBC News, «MIDI, la tecnología que le abrió la puerta a la música digital,» 8 diciembre 2012. [En línea]. Disponible: https://www.bbc.com/mundo/noticias/2012/12/121205_midi_tecnologia_musica_np. [Consultado en febrero de 2021].
- [5] Behringer , «FCB1010: Manual de uso,» [En línea]. Disponible: https://mediadl.musictribe.com/media/sys_master/hda/h8e/8850102091806.pdf. [Consultado en enero de 2021].
- [6] Practical Usage, «The Behringer FCB1010 Pedal Board,» 22 marzo 2011. [En línea]. Disponible: <http://practicalusage.com/the-behringer-fcb1010-pedal-board/>. [Consultado en marzo 2021].
- [7] O. Miramontes, «Los sistemas complejos como instrumentos de conocimiento y transformación del mundo,» 1999. [En línea]. Disponible: <https://www.fisica.unam.mx/personales/mir/mundo.html>. [Consultado en abril de 2021].
- [8] T. Erik, «Selfmade I2C 7-Segment Display with ATTiny85,» 26 diciembre 2018. [En línea]. Disponible: <https://www.hackster.io/taunoerik/selfmade-i2c-7-segment-display-with-attiny85-a637d0>. [Consultado en septiembre de 2020].
- [9] L. Llamas, 1 abril 2018. [En línea]. Disponible: <https://www.luisllamas.es/esp32/>. ESP32, el "hermano mayor" del ESP8266 con Wi-Fi y Bluetooth. [Último acceso: 20 febrero 2020].
- [10] N. Bags, «arduino-esp32 BLE-MIDI,» 23 abril 2019. [En línea]. Disponible: <https://github.com/neilbags/arduino-esp32-BLE-MIDI>. [Consultado en noviembre de 2020].

9 Anexos

9.1 Manual de usuario

Este manual pretende servir de guía a un usuario cuyo interés es únicamente poder utilizar el controlador MIDI que se ha desarrollado, no comprender el por qué de su diseño ni su lógica interna.

9.1.1 Conexión Bluetooth

Para conectar el teléfono móvil al controlador MIDI por medio de Bluetooth utilizamos la aplicación nRF Connect, que permite conectar dos dispositivos mediante el protocolo Bluetooth BLE.

1. Seleccione “conectar” en el dispositivo llamado “MIDI”.
2. Pulse en la segunda pestaña llamada “Cliente”.
3. Pulse en el ícono de recibir información constantemente.

La Figura 1 muestra cómo realizar estos pasos.

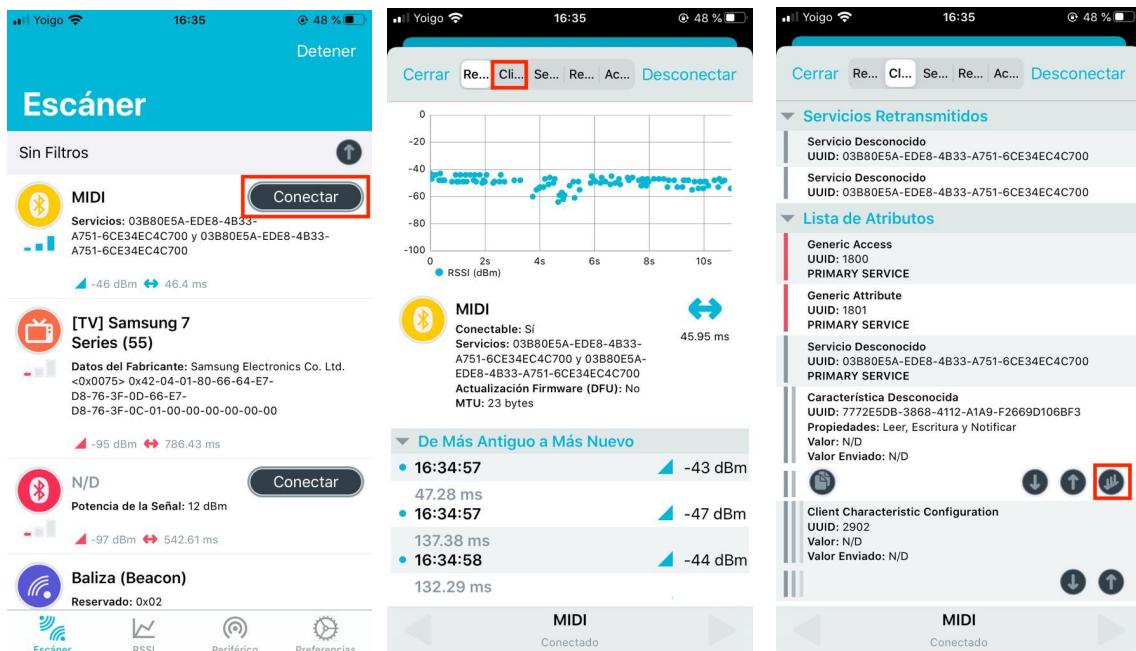


Figura 1. Pasos para recibir comandos en el teléfono móvil por Bluetooth BLE.

Por otro lado, es necesario que activar el control por MIDI en la aplicación JamUp siguiendo los siguientes pasos.

1. Acceda a la configuración de la aplicación en el ícono de la esquina inferior derecha (Figura 2).



Figura 2. Acceso al menú de configuración en la aplicación JamUp.

2. Seleccione *MIDI Setting* para acceder a la configuración MIDI (Figura 3).

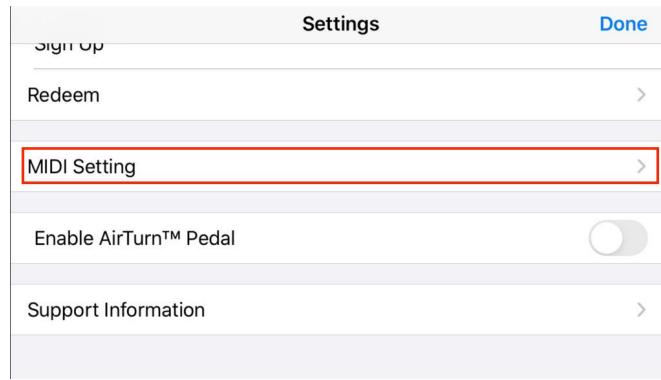


Figura 3. Acceso al menú de configuración MIDI en la aplicación JamUp

3. Habilite el control MIDI en la primera opción (*Enable MIDI Control*) y seleccione el canal MIDI (*MIDI channel*) 1 o, si solo va a trabajar con estos comandos MIDI puede seleccionar “All channel” (todos los canales). El resultado final aparece en la Figura 4.



Figura 4. Habilitación del control MIDI y selección del canal.

9.1.2 Funcionamiento del dispositivo

Este controlador MIDI cuenta con 4 modos: Modo *Preset*, Modo *Jam*, Modo *Sampler* y Modo Libre. Los tres primeros sirven para utilizar el dispositivo con la aplicación JamUp.

El Modo Libre es independiente a la aplicación. En este modo se pueden asignar comandos a las 3 pulsaciones de los pulsadores con LED de la fila de abajo y a dos pulsaciones (corta y doble) de la fila de los pulsadores con LED de arriba. Las pulsaciones largas de la fila de arriba se reservan para cambiar de modo. Esto significa que hay 25 comandos asignables a pedales y 2 comandos asignables a los dos pedales de expresión.

El dispositivo se enciende al conectarlo a la red eléctrica a través de su fuente de alimentación. Por defecto se inicia en Modo *Preset*. La funcionalidad de este modo se mantiene en Modo *Jam* y Modo *Sampler*. Estos dos modos solo añaden algunos comandos para controlar funciones específicas. En la Figura 5 aparecen las funciones del Modo *Preset* asignadas al tipo de pulsación de cada pulsador.

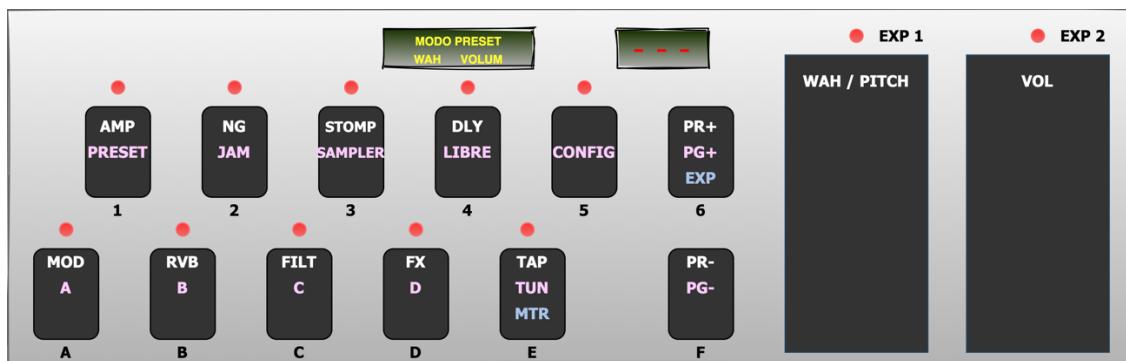


Figura 5. Pulsaciones asociadas a comandos del Modo Preset.

En la Figura 6 aparecen las funciones del Modo *Jam* asignadas al tipo de pulsación de cada pulsador. Como vemos solo se añaden al Modo *Preset* seis pulsaciones dobles más.

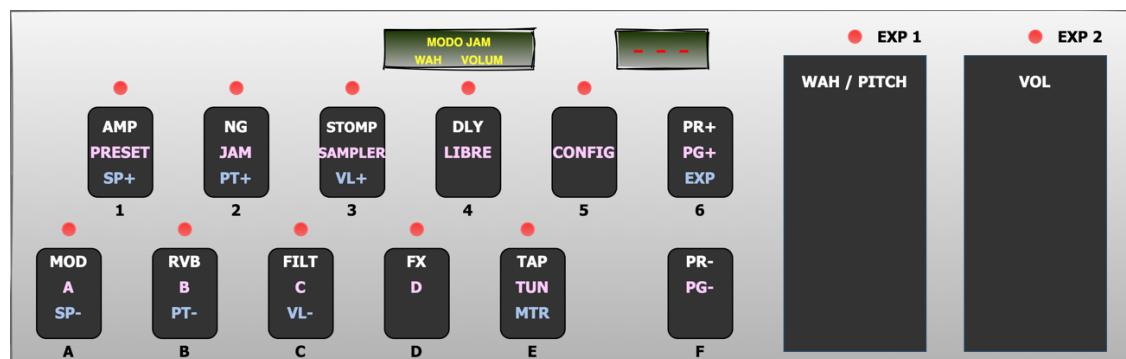


Figura 6. Pulsaciones asociadas a comandos del Modo Jam.

En la Figura 7 aparecen las funciones del Modo *Sampler* asignadas al tipo de pulsación de cada pulsador. Como vemos solo se añaden al Modo *Preset* 3 pulsaciones dobles y una corta.

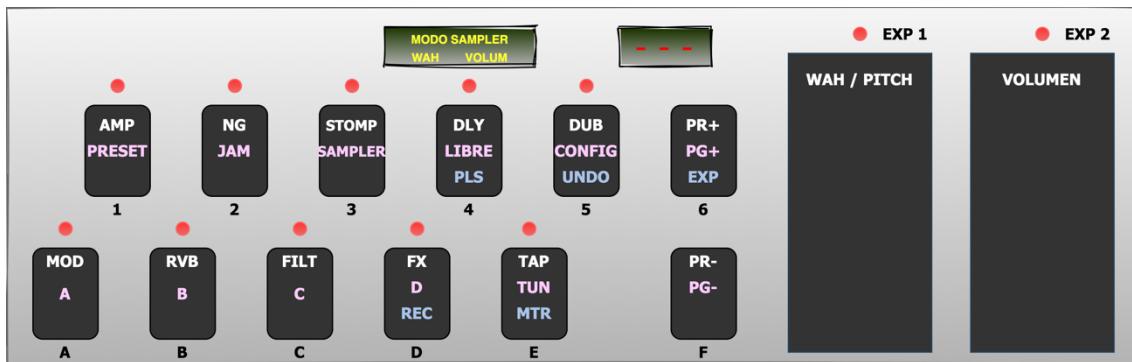


Figura 7. Pulsaciones asociadas a comandos del Modo Sampler.

9.1.3 Configuración del dispositivo

Las configuraciones SLD, LLD y CC se utilizan para Modo *Preset*, Modo *Jam* y Modo *Sampler*.

SLD: salvado del estado de los LEDs

Se pueden guardar los LEDs que se deben iluminar por estar el efecto activo en 8 *presets* (1A, 1B, 1C, 1D, 2A, 2B, 2C y 2D), todos asignados al banco 1.

1. En la pantalla LCD, se muestra la opción de configuración (“guardado LEDs”) y “Preset: ” en la fila inferior. La pedalera espera el primer carácter.
2. Seleccione el primer carácter (1 ó 2). Si quiere guardar el estado de los LEDs del *preset* 2C, en este paso debe pulsar el pulsador 2.
3. En la pantalla aparece el carácter introducido. La pedalera espera el segundo carácter.
4. Introduzca el segundo carácter con los pulsadores de la fila inferior (de la A a la D). Si desea guardar el estado de los LEDs del *preset* 2C accione el pulsador C.
5. Aparece en la pantalla LCD el *preset* introducido.
6. Aparece en la pantalla LCD “Seleccione LEDs” y se apagan todos los LEDs de la pedalera.
7. Encienda los LEDs que deben estar activos cuando se cambie al *preset* seleccionado mediante pulsaciones en los pulsadores correspondientes.
8. Realice una pulsación larga en el pulsador 6 para guardar.
9. Aparece “GUARDADO” en la pantalla LCD. La pedalera pasa a Modo *Preset* y queda cargado el estado de los LEDs que se acaba de guardar. En los displays aparece el *preset* también.

Estos pasos se encuentran ilustrados en el diagrama de uso de la Figura 8. El código de colores para los distintos tipos de pulsaciones es blanco para pulsaciones cortas, rosa para pulsaciones largas y azul para pulsaciones dobles.

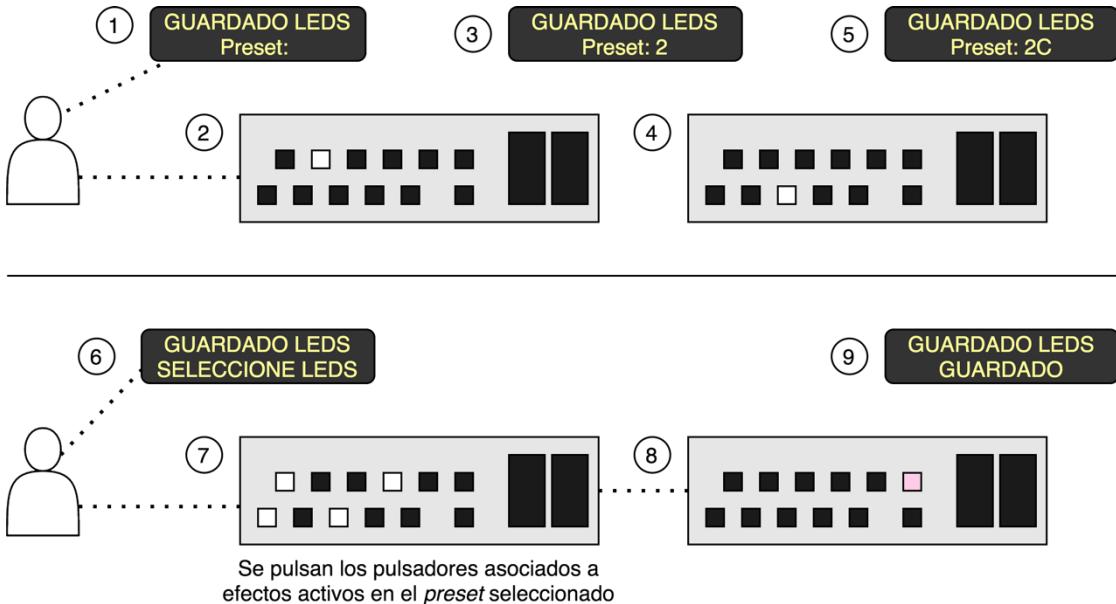


Figura 8. Diagrama de uso de la opción de configuración del SLD (salvado del estado de los LEDs).

LLD: carga del estado de los LEDs

Mediante esta opción 1 del menú de configuración, puede introducir el *preset* que se desea cargar, las luces LEDs asociadas a este *preset* se encienden y su nombre aparece en los *displays* de 7 segmentos.

1. En la pantalla LCD, se muestra la opción de configuración ("cargado LEDs") y "Preset: " en la fila inferior. La pedalera espera el primer carácter.
2. Seleccione el primer carácter (1 ó 2). Si quiere cargar el estado de los LEDs del *preset* 2C, en este paso debe pulsar el pulsador 2.
3. En la pantalla aparece el carácter introducido. La pedalera espera el segundo carácter.
4. Introduzca el segundo carácter con los pulsadores de la fila inferior (de la A a la D). Si desea cargar el estado de los LEDs del *preset* 2C, accione el pulsador C.
5. Aparece en la pantalla LCD el *preset* introducido.
6. Aparece "PRESET CARGADO" en la pantalla LCD.
7. La pedalera pasa a Modo *Preset*, en los *displays* aparece el *preset* seleccionado y se encienden los LEDs correspondientes.

Estos pasos se encuentran ilustrados en el diagrama de uso de la Figura 9.

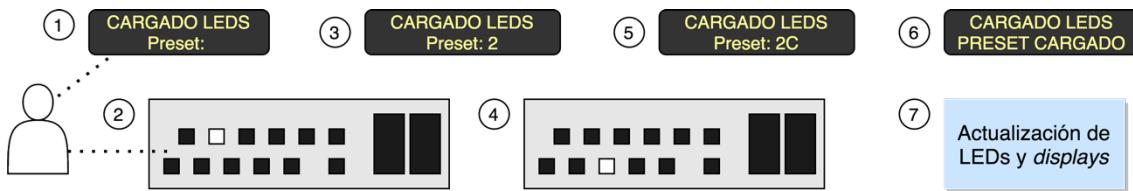


Figura 9. Diagrama de uso de la opción de configuración del LLD (cargado del estado de los LEDs).

CC: carga de comandos CC

Se debe configurar en número de CC asociado a cada una de las acciones que se pueden realizar mediante comandos MIDI en la aplicación. Para ello se utiliza un fichero JSON que se envía mediante *Bluetooth* del teléfono móvil al microcontrolador ESP32.

A continuación, en la Figura 10, se copia el JSON completo de comandos CC que puede recibir el dispositivo.

```
{
    "AMP": "12",
    "NG": "13",
    "STOMP": "14",
    "DLY": "15",
    "MOD": "16",
    "RVB": "17",
    "FILT": "18",
    "FX": "19",
    "PR+": "20",
    "PR-": "21",
    "PG+": "22",
    "PG-": "23",
    "A": "24",
    "B": "25",
    "C": "26",
    "D": "27",
    "TUN": "28",
    "MTR": "29",
    "TAP": "30",
    "VOL": "31",
    "WAH": "32",
    "PTCH": "33",
    "SP+": "34",
    "SP-": "35",
    "PT+": "36",
    "PT-": "37",
    "VL+": "38",
    "VL-": "39",
    "PLJ": "40",
    "REC": "41",
    "UNDO": "42",
    "PLS": "43",
    "DUB": "44"
}
```

Figura 10. JSON de configuración de comandos CC.

Como se observa, tiene estructura clave-valor. La clave no se puede cambiar, pero sí el comando asignado a ella. Por ejemplo, si quiere asociar el CC7 a activar/desactivar el amplificador, tiene que asociar a la clave “AMP” el número “7”. En JamUp es necesario realizar esta misma asociación en el menú de configuración MIDI.

A continuación, se muestra un listado de las acciones controlables por comandos por la aplicación JamUp. Entre paréntesis está la clave con la que se tienen que enviar en el JSON.

- Efectos:
 - AMP (AMP): apagar/encender amplificador.
 - NG (NG): apagar/encender puerta de ruido.
 - STOMP (STOMP): apagar/encender *stomp*.
 - DLY (DLY): apagar/encender *delay* (retardo).
 - MOD (MOD): apagar/encender modulación.
 - RVB (RVB): apagar/encender *reverb* (reverberación).
 - FILT (FILT): apagar/encender efecto de filtro.
 - FX (FX): apagar/encender efecto seleccionado.
- Preset:
 - PRESET + (PR+): cambiar al siguiente *preset*.
 - PRESET - (PR-): cambiar al anterior *preset*.
 - PAGE - (PG-): cambiar a la página anterior.
 - PAGE + (PG+): cambiar a la página siguiente.
 - PRESET A (A): cambiar a la página de *preset A*.
 - PRESET B (B): cambiar a la página de *preset B*.
 - PRESET C (C): cambiar a la página de *preset C*.
 - PRESET D (D): cambiar a la página de *preset D*.
- Pedales de expresión
 - Volumen (VOL): establecer el volumen de salida.
 - Cry Wah (WAH): cambiar la profundidad del *cry-wah* (filtro paso banda de frecuencia central variable).
 - Pitch shifter (PTCH): modificador la altura de la nota.
- Utilidades
 - METRONOME (MTR): apagar/encender metrónomo.
 - TAP TEMPO (TAP): hacer *tap* (pulsaciones) para establecer el tempo del metrónomo.
 - TUNER (TUN): apagar/encender afinador.

- *Jam:*

- PLAY/STOP (PLJ): parar y continuar.
- SPEED + (SP+): aumentar la velocidad de la pista de fondo.
- SPEED – (SP-): disminuir la velocidad de la pista de fondo.
- PITCH + (PT+): incrementar el *pitch* de la pista de fondo.
- PITCH – (PT-): disminuir el *pitch* de la pista de fondo.
- VOLUME + (VL+): incrementar el volumen de la pista de fondo.
- VOLUME – (VL-): disminuir el volumen de la pista de fondo.

- *Sampler*

- REC (REC): empezar/parar grabación.
- UNDO (UNDO): rehacer el último bucle.
- PLAY (PLS): reproducir/parar.
- DUB (DUB): empezar/parar bucle.

El JSON puede tenerse guardado en un fichero de texto o en la sección de notas del móvil y desde ahí cambiar los valores asociados a las claves. Otra opción es generar el JSON en una aplicación como “Jayson”. Esto se haría de la siguiente manera:

1. Abra la aplicación y se cree un archivo nuevo en el icono “+”.
2. Seleccione “*dictionary*” (diccionario) para crear un archivo de estructura clave - valor.
3. Seleccione una nueva entrada clave – valor.

Se puede seguir la guía de la Figura 11 para saber dónde se encuentran las opciones en la aplicación.

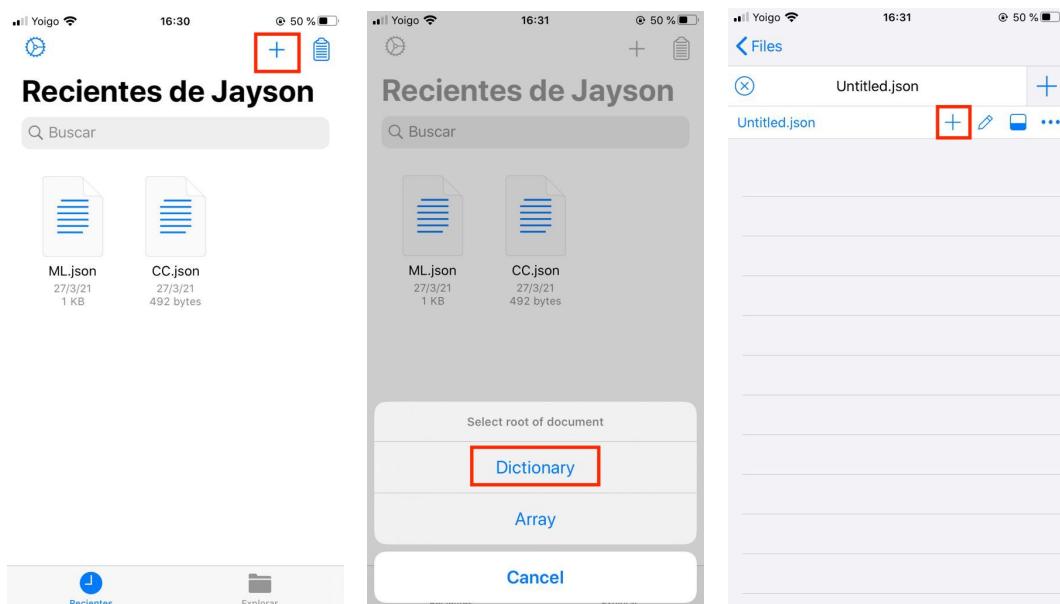


Figura 11. Pasos 1 (izq.), 2 (centro) y 3 (dcha.) de la explicación de generación de un JSON con la aplicación “Jayson”.

1. Seleccione que el valor va a ser un texto. Complete la clave y el valor y finalmente pulse “*save*” (guardar). Como ejemplo, se va a añadir a activar/desactivar amplificador el comando CC12.
2. Seleccione el icono de los tres puntos de la parte derecha de la pantalla y la opción de “*share*” (compartir).
3. Pulse en “*share value*” (compartir valor).

En la Figura 12 se muestra cómo seguir estos pasos en la aplicación.

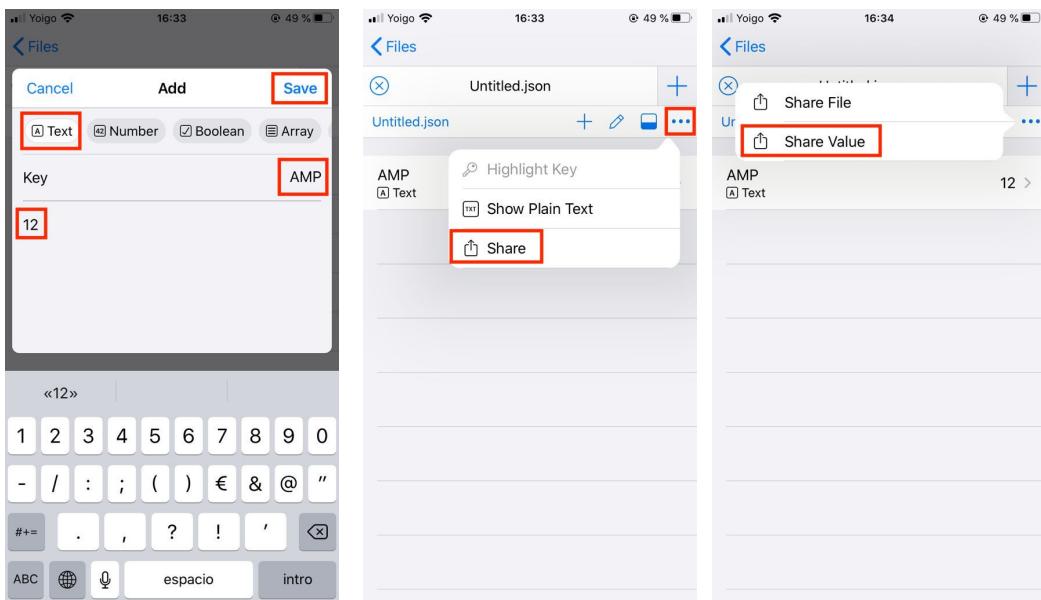


Figura 12. Pasos 4 (izq.), 5 (centro) y 6 (dcha.) de la explicación de generación de un JSON con la aplicación "Jayson".

4. Finalmente, seleccione “*copy*” (copiar) como se muestra en la Figura 13. De esta manera ya tiene en el portapapeles el fichero JSON de la forma en la que hay que enviarlo listo para pegar.

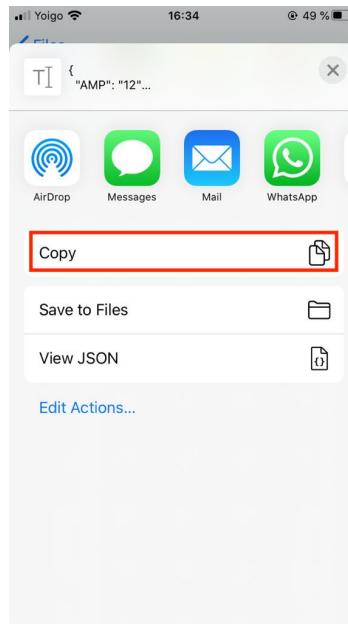


Figura 13. Paso 7 de la explicación de generación de un JSON con la aplicación "Jayson".

Para enviar el JSON desde el teléfono móvil se utiliza la aplicación nRF Connect:

1. Una vez se tiene el JSON en el portapapeles y activado el Bluetooth del móvil, acceda a esta aplicación y seleccione “conectar” en el dispositivo llamado “MIDI”, como se muestra en la Figura 14.
2. Seleccione la segunda pestaña llamada “Cliente”.
3. En la característica, seleccione la opción de enviar datos.

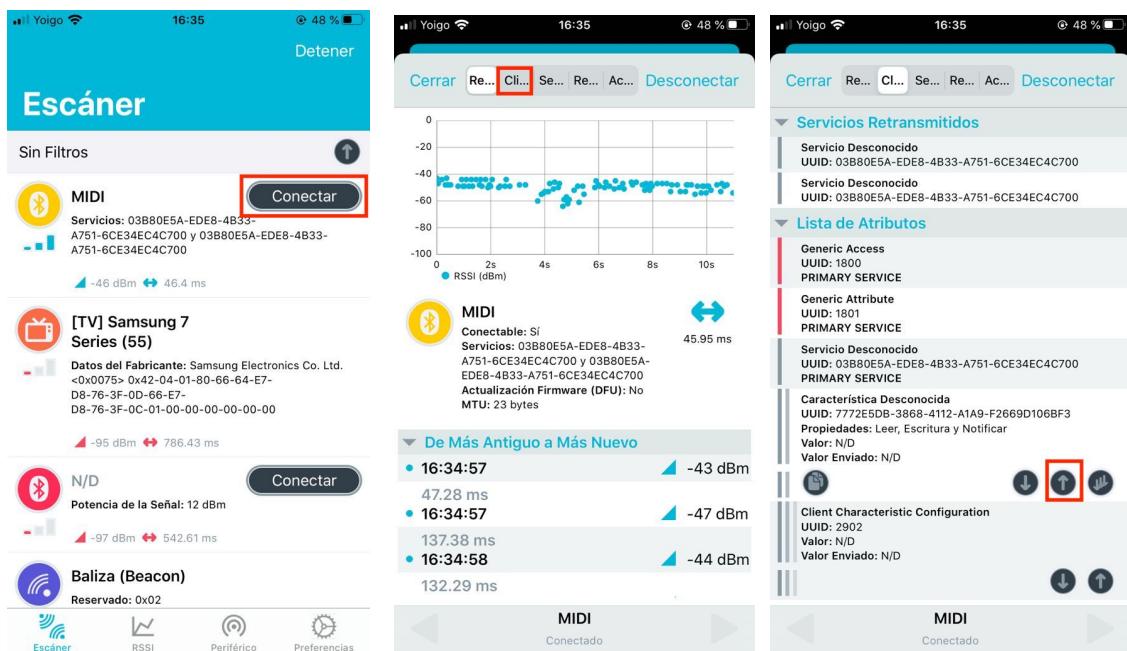


Figura 14. Primeros pasos para conectar la pedalera al teléfono móvil y enviar un JSON.

4. Seleccione UTF8 en la casilla que se puede ver en la Figura 15.
5. Copie el JSON y seleccione “Escritura”.

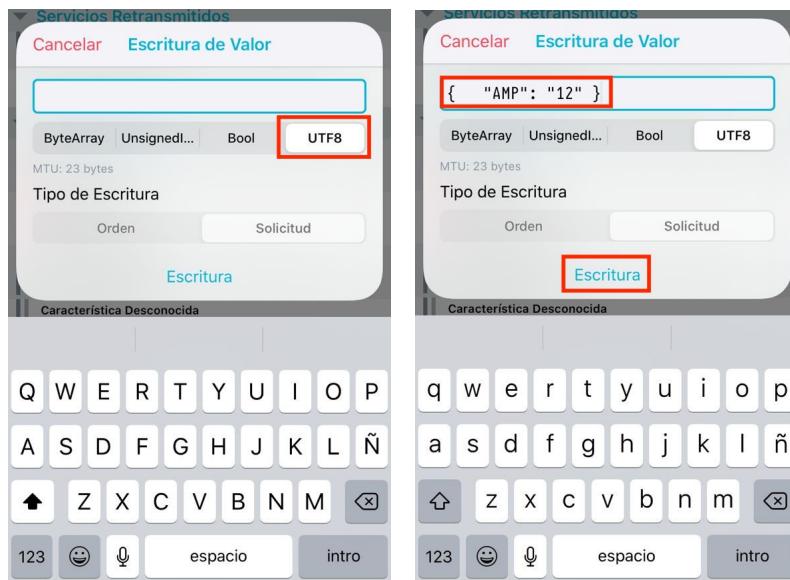


Figura 15. Pasos a seguir para el envío del JSON de configuración.

6. Los valores quedan cargados en las variables correspondientes y comienzan a funcionar. En cambio, si quiere que estos valores se mantengan una vez se reinicie el dispositivo debe acceder a la opción CC del menú de configuración, que simplemente guarde estos valores en la memoria del ESP32.

Los comandos CC que se introducen en la pedalera tienen que coincidir con los de la aplicación. Para introducir los comandos en la aplicación hay que acceder al menú de configuración MIDI como se explicó al comienzo del manual. Después, se selecciona “*Midi Control Assignments*” (asignaciones de comandos MIDI) como se muestra en la Figura 16.

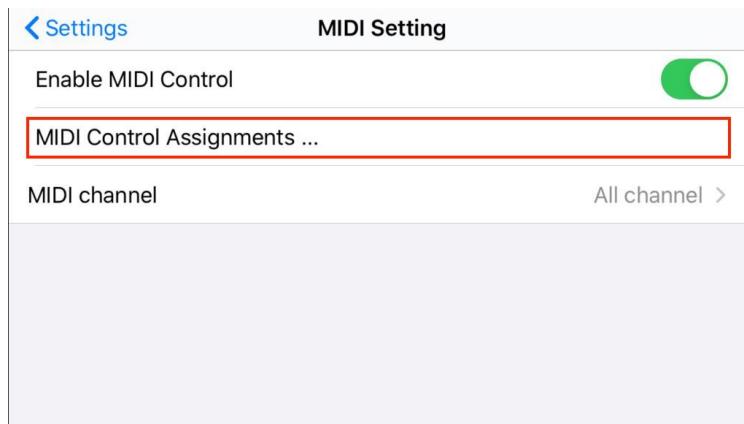


Figura 16. Acceso a las asignaciones del canal MIDI en la aplicación JamUp.

Aquí se encuentran todos los comandos ya asignados, que se pueden modificar pulsando en ellos. Si lo que se quiere es añadir un nuevo comando seleccione “Add New Control Assignment” (añadir nueva asignación de comando), como se muestra en la Figura 17.

MIDI Assignments	
Done	Edit
Add New Control Assignment...	
ASSIGNED CONTROLS	
Toggle amplifier	CC 12
Toggle noise gate effect	CC 13
Toggle delay effect	CC 14
Toggle stomp effect	CC 15
Toggle modulation effect	CC 16

Figura 17. Añadir nueva asignación de comando CC en la aplicación JamUp.

Se selecciona, de entre la lista, la función a la que se quiere asignar un comando. Si se quiere asignar a activar/desactivar el amplificador el CC12 se selecciona lo que se indica en las Figuras 18, 19 y 20.



Figura 18. Acceso a los controles de amplificadores y efectos en la aplicación JamUp.

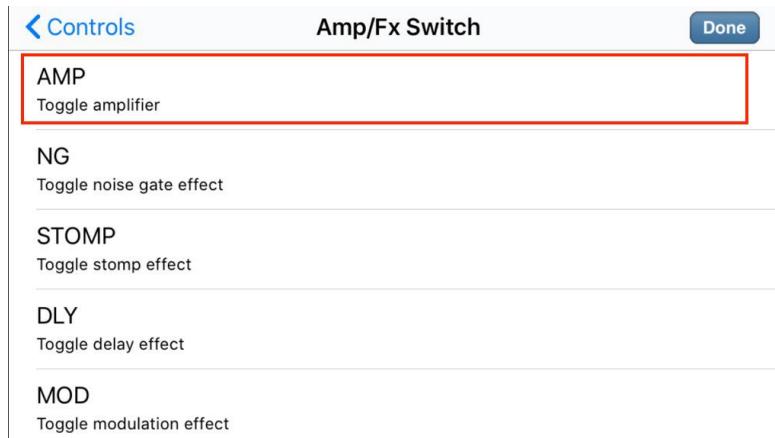


Figura 19. Selección de la acción de activar/desactivar amplificador.



Figura 20. Selección del comando CC 12 en la aplicación JamUp.

ML: configuración del modo libre

De igual manera que en el apartado anterior, para asociar los comandos MIDI a los distintos pulsadores se utiliza un fichero JSON. Cada pulsador tiene 3 claves, una para cada tipo de pulsación (S* para cortas, L* para largas y D* para dobles). En esta ocasión, los pulsadores se numeran del 0 al 9 empezando por el que antes llamábamos pulsador 1 hasta el F que sería el 9. Así, la clave de la pulsación larga del pulsador 4 será L3. Cada clave lleva asociados 3 valores: uno para cada byte del comando MIDI a completar, ya que ahora el tipo de comando no está limitado a CC.

El primer byte es el de estado, que indica qué función activar y en qué canal. Los 4 bits que se marcan como nnnn o N en hexadecimal son los de canal. N va de 0 al 15 para canales del 1 al 16. En la Tabla 1 se muestran los valores del byte de estado de los distintos tipos de mensajes MIDI junto al significado que tienen los valores del segundo y tercer byte para cada uno. Se recuerda que el byte más significativo del segundo y tercer byte es 0, lo que quiere decir que sus valores estarán comprendidos entre 0 y 127.

Tabla 1. Tipos de mensajes MIDI

Función	Binario	Hex	Decimal	Byte 2	Byte 3
Note Off	1000 nnnn	8 N	128 + N	Altura	Velocidad
Note On	1001 nnnn	9 N	144 + N	Altura	Velocidad
Aftertouch polifónico	1010 nnnn	A N	160 + N	Altura	Presión
Cambio de control	1011 nnnn	B N	176 + N	Nº de control	Nuevo valor
Cambio de programa	1100 nnnn	C N	192 + N	Nº programa	
Aftertouch de canal	1101 nnnn	D N	208 + N	Presión	
Pitch bend	1110 nnnn	E N	224 + N	MSByte	LSByte
Sistema Exclusivo	1111 nnnn	F N	240 + N		

A los pulsadores se les puede asociar cualquier pulsador del listado. En cambio, a los pedales de expresión se le pueden asignar 4 comandos: *Aftertouch* de Canal, *Aftertouch Polifónico*, *Control Change*, y *Pitch Bend*. El movimiento del pedal en los dos primeros modifica el valor del byte de presión. En el tercero, el byte de valor. En cambio, en el comando Pitch Bend se utilizan los dos bytes para mandar el valor de la entrada con el objetivo de alterar el sonido de una manera más fluida.

En la Figura 21 se muestra el fichero de configuración de un modo libre que haría lo siguiente:

- Canal 3 – Pulsador A – Pulsación corta: Cambio de Programa al 27.
- Canal 10 – Pulsador 3 – Pulsación larga: *Note On* de C5 con una velocidad de 64.
- Canal 16 – pulsador C – Pulsación doble: Cambio de Control 8 a un valor de 16.
- Canal 5 – Pedal de expresión 1 – *Aftertouch* polifónico. Nota D4.

```
{
  "S5": [
    196,
    27,
    0
  ],
  "L2": [
    155,
    72,
    64
  ],
  "D7": [
    191,
    8,
    16
  ],
  "EXP1": [
    164,
    62,
    0
  ]
}
```

Figura 21. Ejemplo de JSON de configuración del Modo Libre.

Este texto, de igual manera que en el apartado anterior, se puede tener guardado en un fichero de texto o en las notas del móvil, pero también se puede generar con la aplicación “Jayson”. Para ello:

1. Acceda a la aplicación y creamos un fichero nuevo de tipo “Dictionary” (diccionario), que quiere decir que se asigna un valor a una clave.
2. Añada una entrada en el ícono “+”.
3. Seleccione que el tipo de la entrada sea “Array” y añada la clave teniendo en cuenta el tipo de pulsación (S para corta, L para larga y D para doble) y el pulsador (del 0 al 9). En el ejemplo de la Figura 22 se introduce el ejemplo anterior de la pulsación corta en el pulsador A (S5) y canal 3 para realizar un Cambio de Programa al 27.

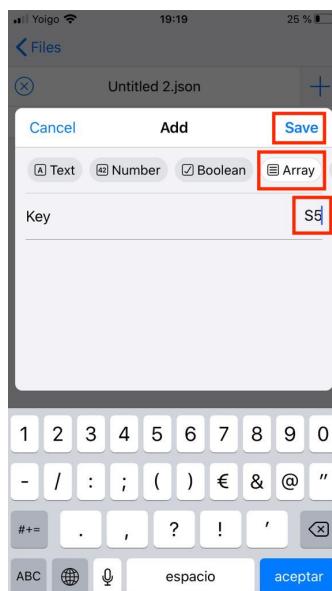


Figura 22. Creación de una clave del JSON para una pulsación corta en el pulsador A.

4. A continuación, añada los valores del *array*, para lo que debe seleccionar la clave, “S5” en el caso de ejemplo.
5. Seleccione el ícono “+” para añadir el primer valor del *array*.
6. Seleccione que se va a introducir un número (“Number”), escriba el número (196) y pulse en “save” (guardar). En la Figura 23 se muestran las capturas explicativas.

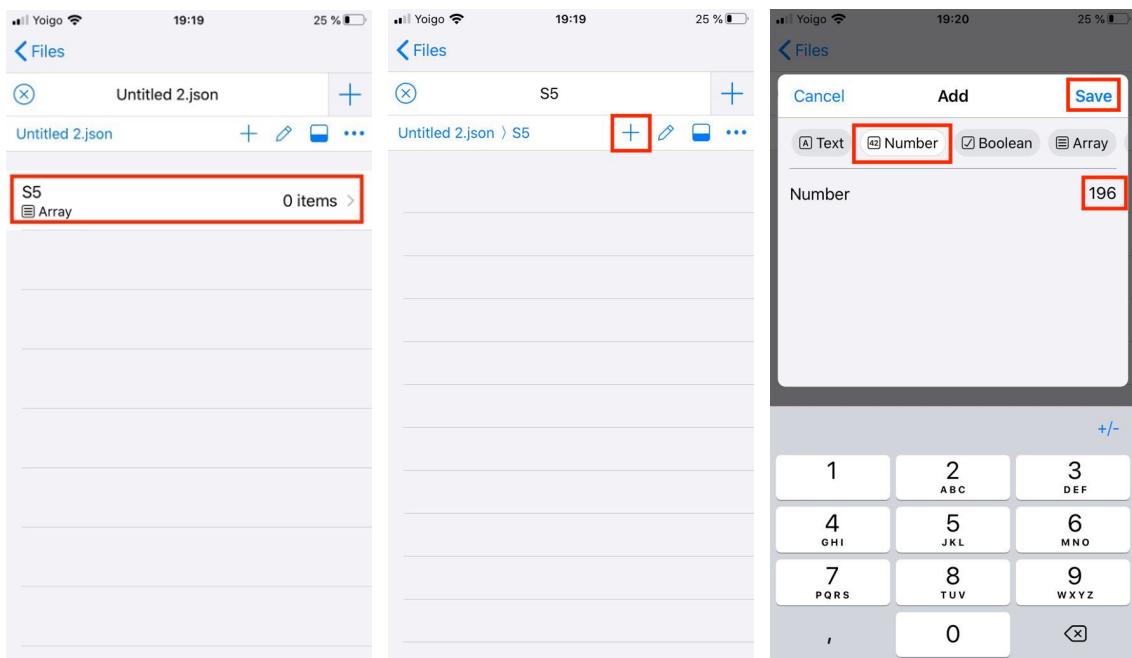


Figura 23. Introducción de los valores del array con clave S5.

- Realice lo mismo para los otros dos valores del array. Cada una de las claves queda como la que se muestra en la Figura 24.

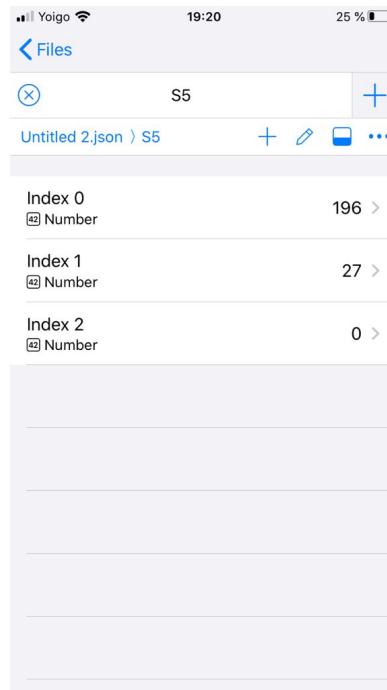


Figura 24. Valores del array de la clave "S5" del JSON de configuración de Modo Libre.

EXP: calibración de pedales de expresión

Es recomendable calibrar los pedales de expresión con cierta asiduidad. El dispositivo, por defecto siempre utiliza la última calibración realizada. Para ello, en el menú de configuración, al que como se ha dicho antes se accede mediante una pulsación larga en el pulsador 5, se selecciona el pulsador 4 para acceder a EXP (calibración de pedales de expresión), como aparece indicado en la pantalla LCD.

Primero, la pantalla indica “MÍNIMO”, por lo que debe bajar los pedales de expresión hasta el mínimo. A los pocos segundos, la pantalla indica “MÁXIMO”, por lo que debe hacer lo contrario, debe ponerlos en el máximo. A los segundos aparecerá “GUARDADO”. Los pedales han sido calibrados.

DSP (añadir dispositivo a bus I2C).

Pulsando el pulsador A se accede a la última opción del menú, donde se puede añadir un nuevo módulo al bus I2C. A continuación, se detallan los pasos a seguir :

1. La pantalla LCD muestra la opción de configuración en la fila superior (NUEVO MÓDULO) y, en la fila inferior el texto “Direcc (0-99)”. La pedalera espera que introduzca una dirección del 0 al 99. Las direcciones del bus I2C ocupadas actualmente por los distintos módulos son: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 y 16. No se puede asignar ninguna de estas direcciones al nuevo módulo.
2. Introduzca el primer carácter de la dirección. Los pulsadores introducen los números del 0 al 4 en la fila superior y del 5 al 9 en la fila inferior. Los dos pulsadores sin LED no introducen ningún número.
3. La pantalla LCD muestra el dígito introducido en la pantalla.
4. Introduzca ahora el segundo carácter.
5. La pantalla LCD muestra la dirección introducida.
6. Aparece en la pantalla “MÓDULO AÑADIDO”, la dirección se guarda y se redirige a Modo Preset.

En la Figura 25 se muestra el diagrama de uso de esta opción de configuración, que ilustra los pasos que se acaban de detallar.

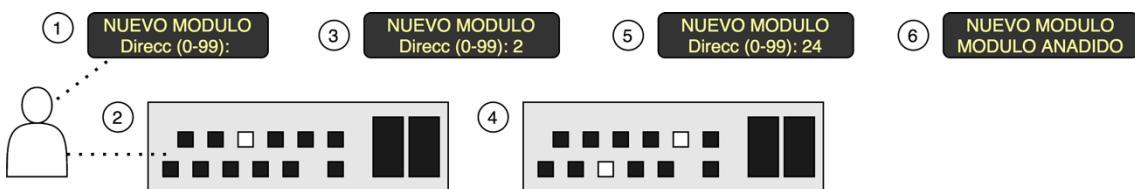
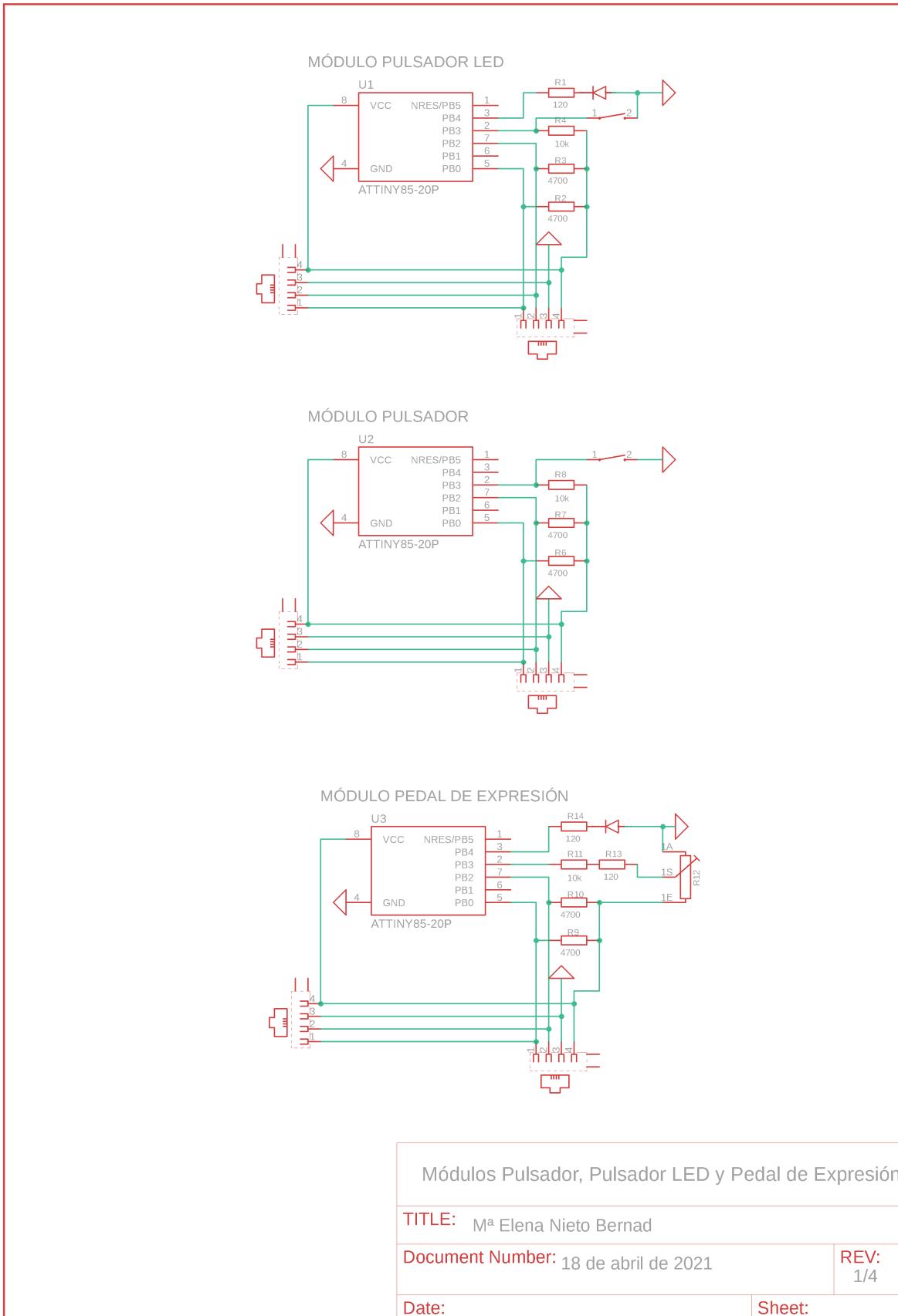


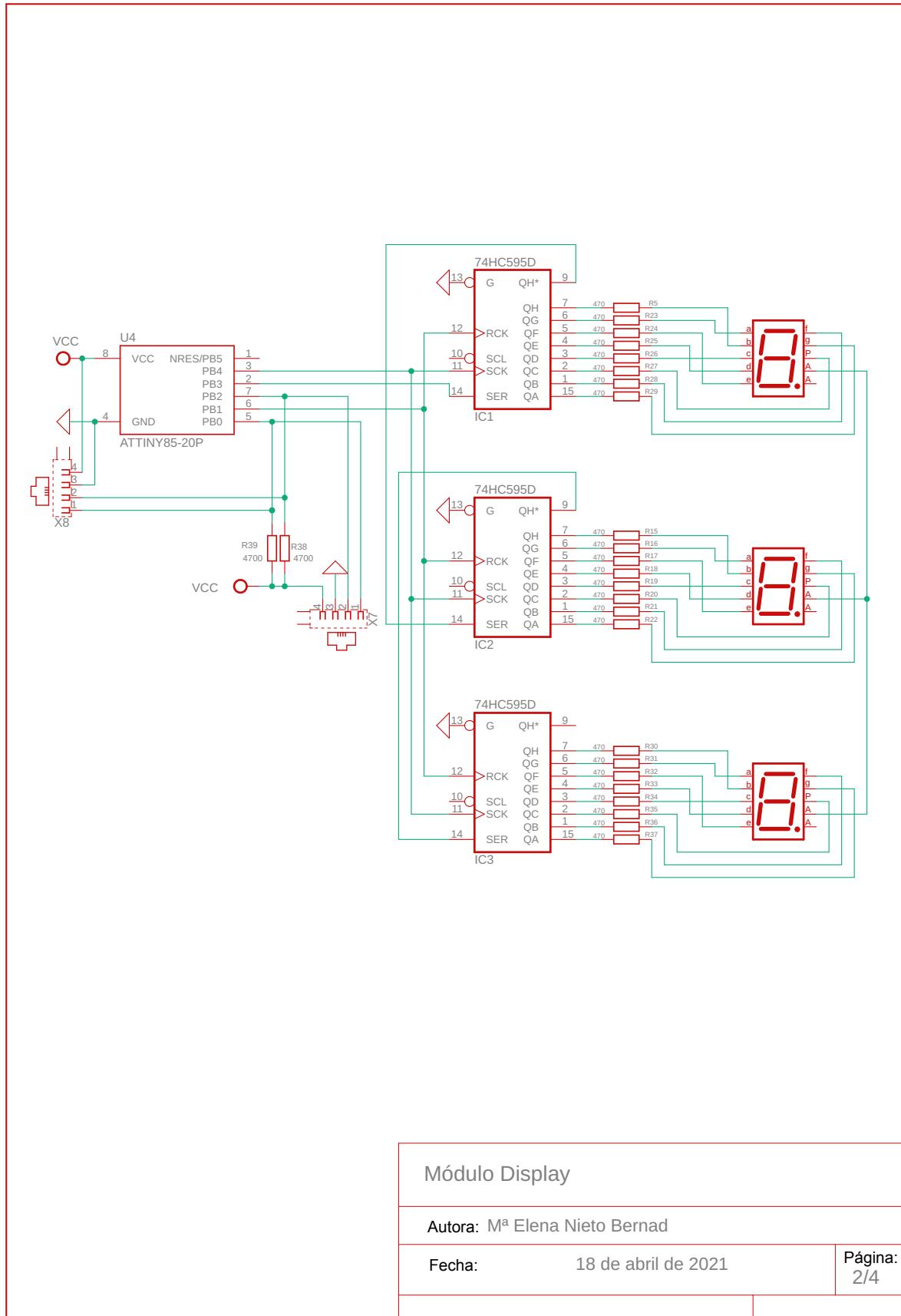
Figura 25. Diagrama de uso de la opción de configuración del DSP (añadir dispositivo al bus I2C).

El controlador está preparado para recibir comandos MIDI del nuevo módulo y enviarlos por Bluetooth y por el conector DIN de 5 pines. Asimismo, está limitado para solo poder añadir un módulo adicional. Si se apaga el dispositivo, hay que añadir el módulo de nuevo en esta opción de configuración.

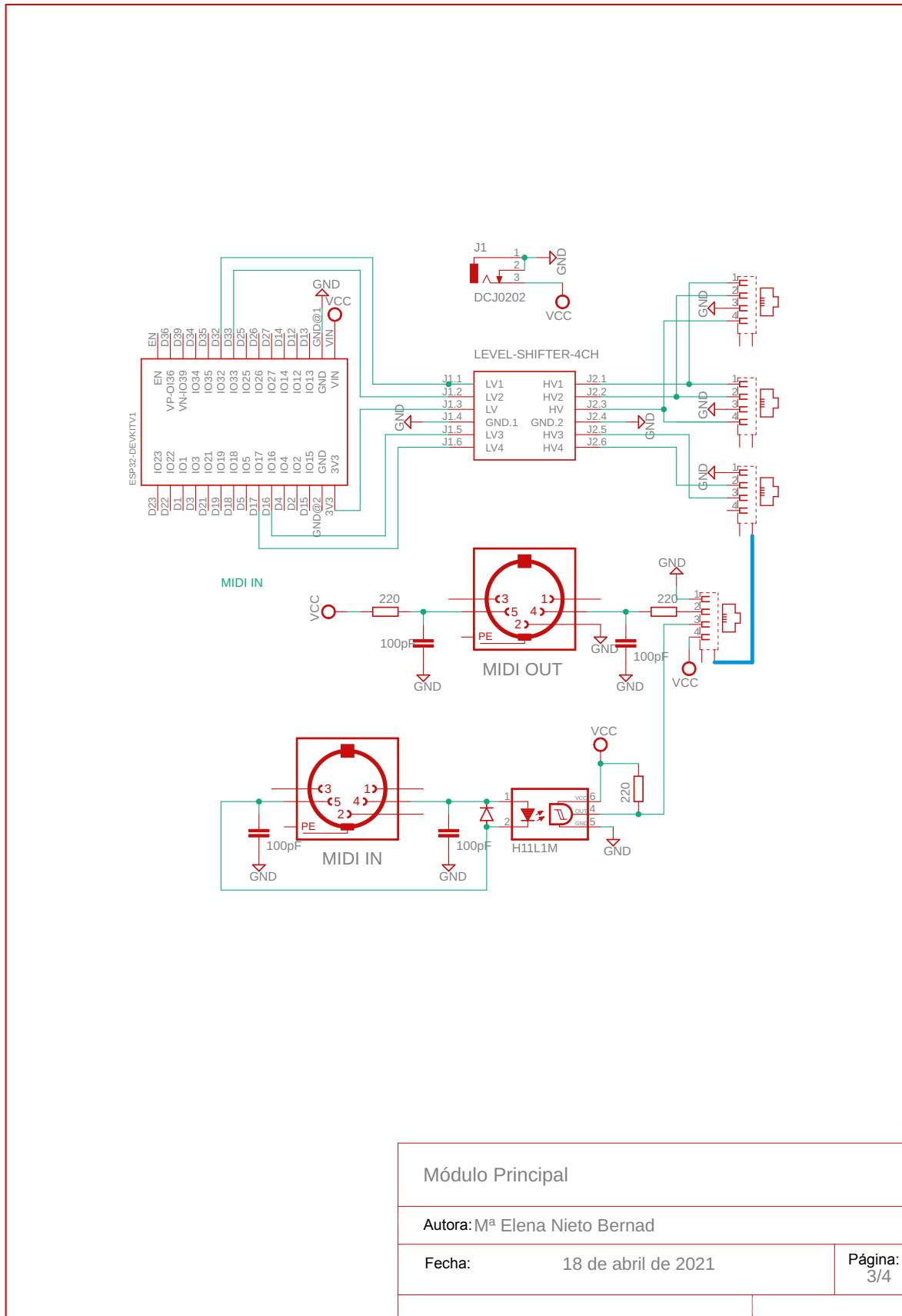
9.2 Esquemas electrónicos

9.2.1 Módulos Secundarios

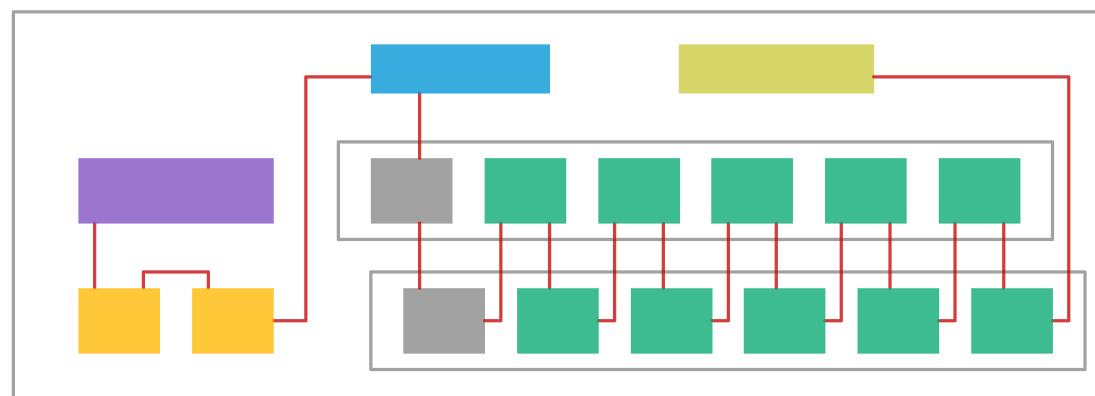




9.2.2 Módulo Principal



9.2.3 Conexión



- Módulo Principal
- Módulo Pedal de Expresión
- Módulo Pulsador
- Módulo Pulsador LED
- Módulo Display
- Pantalla LCD

Distribución de módulos

TITLE: M^a Elena Nieto Bernad

Document Number:
28 de abril de 2021

REV:
4/4

Date: Sheet:

9.3 Índice de Figuras

Figura 1. Pasos para recibir comandos en el teléfono móvil por Bluetooth BLE..	107
Figura 2. Acceso al menú de configuración en la aplicación JamUp.	108
Figura 3. Acceso al menú de configuración MIDI en la aplicación JamUp.....	108
Figura 4. Habilitación del control MIDI y selección del canal.....	108
Figura 5. Pulsaciones asociadas a comandos del Modo Preset.	109
Figura 6. Pulsaciones asociadas a comandos del Modo Jam.	109
Figura 7. Pulsaciones asociadas a comandos del Modo Sampler.	110
Figura 8. Diagrama de uso de la opción de configuración del SLD (salvado del estado de los LEDs).	111
Figura 9. Diagrama de uso de la opción de configuración del LLD (cargado del estado de los LEDs).	112
Figura 10. JSON de configuración de comandos CC.....	112
Figura 11. Pasos 1 (izq.), 2 (centro) y 3 (dcha.) de la explicación de generación de un JSON con la aplicación "Jayson".	114
Figura 12. Pasos 4 (izq.), 5 (centro) y 6 (dcha.) de la explicación de generación de un JSON con la aplicación "Jayson".	115
Figura 13. Paso 7 de la explicación de generación de un JSON con la aplicación "Jayson".	116
Figura 14. Primeros pasos para conectar la pedalera al teléfono móvil y enviar un JSON.	116
Figura 15. Pasos a seguir para el envío del JSON de configuración.	117
Figura 16. Acceso a las asignaciones del canal MIDI en la aplicación JamUp. ...	117
Figura 17. Añadir nueva asignación de comando CC en la aplicación JamUp....	118
Figura 18. Acceso a los controles de amplificadores y efectos en la aplicación JamUp.....	118
Figura 19. Selección de la acción de activar/desactivar amplificador.	119
Figura 20. Selección del comando CC 12 en la aplicación JamUp.	119
Figura 21. Ejemplo de JSON de configuración del Modo Libre.	120
Figura 22. Creación de una clave del JSON para una pulsación corta en el pulsador A.	121
Figura 23. Introducción de los valores del array con clave S5.....	122

Figura 24. Valores del array de la clave “S5” del JSON de configuración de Modo Libre.....	122
Figura 25. Diagrama de uso de la opción de configuración del DSP (añadir dispositivo al bus I2C).....	123

9.4 Índice de Tablas

Tabla 1. Tipos de mensajes MIDI	120
---------------------------------------	-----

