

Tabla de Contenido

- [1. Introducción](#)
 - [1.1. ¿Que es MiDispositivoMIDI?](#)
 - [1.2. ¿Que es este documento?](#)
 - [1.3. Desarrollo](#)
- [2. Protocolo MIDI](#)
 - [2.1 Introducción](#)
 - [2.2 Tipos de mensajes](#)
- [3. Hardware](#)
 - [3.1. Elementos](#)
 - [3.2. Niveles PCB](#)
 - [3.2.1. MDM_ControlBoard](#)
 - [3.2.2. MDM_LEDPad](#)
 - [3.3. Elementos](#)
 - [3.3.1. LED RGB WS2812B](#)
 - [3.3.2. Multiplexor 74HC4067](#)
 - [3.3.3. Arduino Leonardo Micro Pro](#)
- [4. Software](#)
 - [4.1 Introducción a la programación](#)
 - [4.1.1 Código](#)
 - [4.1.2 Lenguajes de programación](#)
 - [4.1.3 Entornos de desarrollo](#)
 - [4.1.4 Estructuras de control](#)
 - [4.2 Instalación IDE Arduino](#)
 - [4.3 Funcionamiento IDE Arduino](#)
 - [4.3.1 Fundamentos Arduino](#)
 - [4.3.2 Carga y Compilación de Sketches](#)
 - [4.4 Ejercicios](#)
 - [4.4.1 Ejercicio 1](#)
 - [4.4.2 Ejercicio 2](#)
 - [4.4.3 Ejercicio 3](#)
 - [4.4.4 Ejercicio 4](#)
 - [4.4.5 Ejercicio 5](#)
- [5. Secuenciadores](#)
 - [5.1 Configuración Ableton](#)
- [6. Conclusión](#)
 - [6.1 Extensión](#)
 - [6.3 Foro web](#)
 - [6.4 FAQ](#)
- [7. Inventario de vídeos](#)
- [8. Inventario código](#)

1. Introducción

1.1. ¿Que es MiDispositivoMIDI?

MiDispositivoMIDI es un controlador MIDI Open Source basado en Arduino. Puede ser usado con cualquier programa de audio como Ableton o Logic y dispone de las siguientes características:

- 16 botones tipo PAD ON/OFF, dispuestos en una matriz de 4x4.
- 16 LED RGB, situados debajo de cada botón.
- 2 botones laterales configurables.
- Conexión USB con cable MicroUSB.
- Dimensiones de 10x10 cm.

El controlador **viene programado por defecto y se comercializa montado y listo** para ser utilizado. También se le pueden cargar los códigos disponibles en nuestro repositorio de GitHub o programar a medida.



Figura 1. MiDispositivoMIDI en funcionamiento (1/2).



Figura 2. MiDispositivoMIDI en funcionamiento (2/2).

1.2. ¿Que es este documento?

Esta guía pretende orientar a los usuarios del controlador MiDispositivoMIDI Rev A3C. Se cubren aspectos desde lo más general (uso, configuración, componentes,...) hasta lo más avanzado (programación, esquemáticos, circuitos). Además del presente documento, se recomienda consultar:

- **Vídeos de youtube:** En estos vídeos se explican ciertos apartados de la guía, de manera más visual. [Enlace](#).
- **Repositorio:** En dicho [repositorio](#) de GitHub se pueden encontrar diversos códigos que puedes cargar en el controlador, sin tener conocimientos de ningún tipo. Para cargar códigos, ver el vídeo "[Instalación](#)".

1.3. Desarrollo

El controlador **MiDispositivoMIDI** tiene sus orígenes en la *Universidad Politécnica de Madrid (UPM)*, donde dos alumnos de *Ingeniería de Telecomunicación*, decidieron empezar con un controlador MIDI basado en Arduino.

Su primera versión, se basaba en Arduino UNO, empleaba registros de desplazamiento para la iluminación de los LEDs y enviaba mensajes empleando el puerto serial. De dicha versión se hicieron 40 unidades, y se impartió el primer curso, allá por el año 2013.

Tras varias mejoras y cambios de firmware en el Arduino, se llegó a la versión 2.0 del

controlador. Las carcasas eran de madera, y la comunicación con el ordenador ya se establecía a través de MIDI. Se hicieron también 40 unidades, y se impartió un curso con muy buena acogida.

Tras un tiempo parados, pero habiendo escuchado todas las mejoras propuestas por nuestros compañeros, decidimos lanzar la tercera versión de MiDispositivoMIDI. Se trata de un controlador mucho más fino, más ligero, con un Arduino más potente, LEDs que permiten generar miles de colores y fácilmente ampliable. Hemos fabricado una tirada grande de esta tercera versión, y ¡estamos deseando dártela!



Figura 3. Asistentes de la primera edición de MiDispositivoMIDI.



Figura 4. Asistentes de la segunda edición de MiDispositivoMIDI.

2. Protocolo MIDI

2.1 Introducción

El protocolo MIDI (*Musical Instruments Digital Interface*) es un lenguaje que utilizan actualmente muchos instrumentos musicales para comunicarse entre ellos y con un ordenador. Su uso les permite enviar y recibir mensaje además de sincronizarse. Se trata de un protocolo con mucha historia, y data de la década de los 80s. Aún así, se sigue usando hoy en día en gran cantidad de teclados, controladores, cajas de ritmo e incluso para controlar luces.

Por ejemplo, el teclado que mostramos a continuación se puede conectar al ordenador con un cable USB y cuando pulsemos una tecla, enviará un mensaje MIDI al ordenador a través del cable. Evidentemente no va a ser el mismo mensaje si pulsamos la tecla muy fuerte o suave, o si pulsamos una tecla u otra. A continuación lo explicaremos.



Figura 5. Ejemplo de controlador MIDI. Teclado controlador.

2.2 Tipos de mensajes

Antes de nada vamos a recordar que es un **bit** y un **byte**. Un bit es la unidad mínima de información digital, es decir un 0 o un 1. Un byte es un conjunto de 8 bits. Cada mensaje MIDI tiene 3 bytes, es decir 24 bits. En función del valor que tenga, el mensaje significará una cosa u otra. A continuación mostramos los 3 paquetes de 1 byte que contiene cada mensaje MIDI. El primero corresponde al byte de **estado**, y lleva la información relativa al estado y el canal. El estado hace referencia al tipo de mensaje: nota activada, nota desactivada, bend, cambio de programa u otros más. El canal puede tomar valores de 0 a 15, y permite separar diferentes flujos de mensajes.

BYTE 1 (Status Byte)							
Estado				Canal			
X	X	X	X	X	X	X	X

El segundo y tercer byte, llevan información que variará en función del estado. Para el caso de un estado *noteOn*, el BYTE 2 indicará la nota y el BYTE 3 indicará la intensidad de la nota.

BYTE 2 (Data Byte 1)							
X	X	X	X	X	X	X	X

BYTE 3 (Data Byte 2)							
X	X	X	X	X	X	X	X

Particularizando para el ejemplo de un mensaje de nota activada *noteOn*, vamos a ver los bits que se enviarán para un caso concreto.

Ejemplo <i>noteOn</i>	
Nota	DO
Octava	3
Intensidad	Máxima
Canal	0

Por un lado, la tercera octava de DO se corresponde con el valor MIDI 48. La intensidad máxima se corresponde con el valor 127, ya que en MIDI las intensidades varían desde 0 a 127, siendo 0 la menor. Sabiendo que 48 en binario es 0110000, 127 es 1111111, el código de *noteOn* es 1001 y 0 en binario es 0000, tenemos lo siguiente:

noteOn				Canal					Nota					Intensidad							
1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	1	1

Como podemos ver, el bit 9 y el 17 se indican en color rojo. Esto es debido a que estos bits van a ir siempre a 0, sin importar el mensaje enviado. Para otros mensajes, la estructura es similar.

EJERCICIO PROPUESTO (INDICADOR NÚMERO)

Se propone escribir el mensaje MIDI completo en binario que se enviaría al pulsar la nota

LA de la tercera octava, con una intensidad de 90.

3. Hardware

3.1. Elementos

MiDispositivoMIDI V3 está conformado por diversos elementos que componen tanto su estructura mecánica como su estructura electrónica. En las siguientes secciones se recogen y analizan distintos aspectos del circuito para que sean comprendidos tanto por iniciados como por usuarios más avanzados de la electrónica.

Los elementos que componen MiDispositivoMIDI V3 son:

QUANTITY.	ITEM
01	Silicone Pad 4x4 10x10cm
01	Wooden CNC Laser Cut MiDispositivoMIDI V3 Cover
01	PCB MDM_V3_ControlBoard
01	PCB MDM_V3_LedPad
01	Arduino Leonardo Micro Pro
16	WS2812B RGB Addressable LED
18	Ceramic Capacitor SMD 1206 10uF
19	Resistor SMD 1206 10kOhm
01	Resistor SMD 1206 470hm
02	Angled Tactile Switch
01	Tactile Switch
01	IC 74HC4067 16to1 Multiplexer
01	DIP Switch 1x4 SMD
01	Male 2x4 Pin Row
01	Male 1x3 Pin Row
01	Female 2x4 Pin Row
01	Female 1x3 Pin Row
04	M2x20 Screw
04	M2 Black Plastic Screw Separator
04	M2 Washer
04	M2 Nut

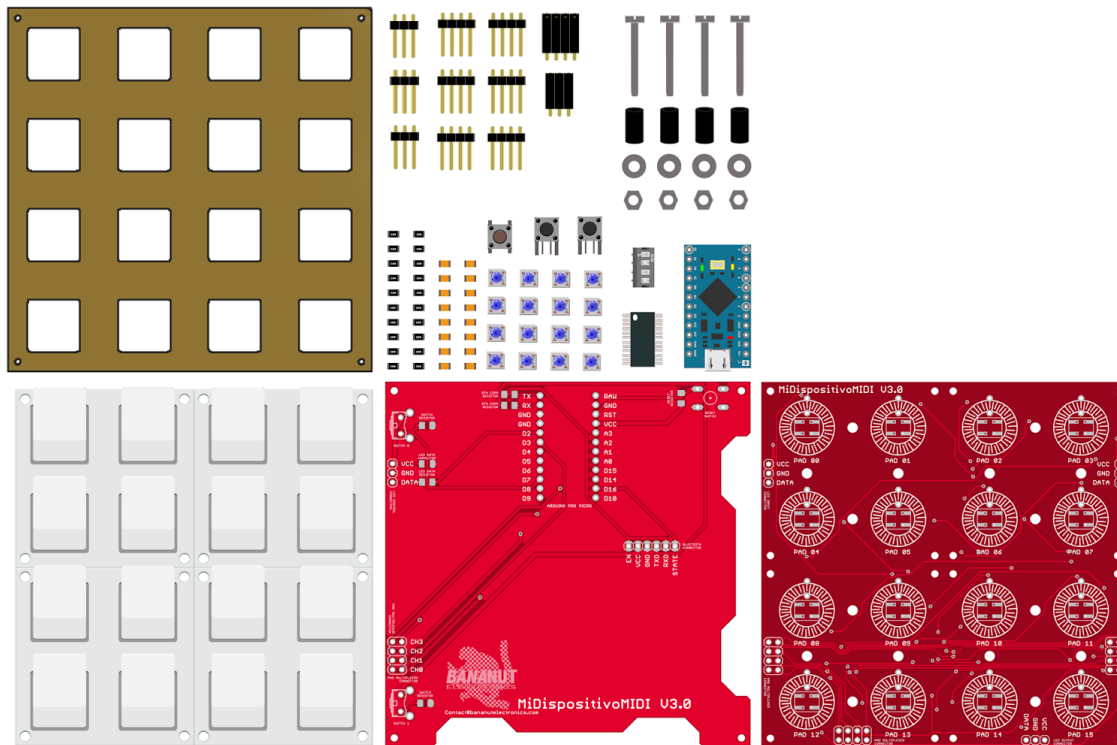


Figura 6. Componentes de MiDispositivoMIDI.

3.2. Niveles PCB

MiDispositivoMIDI V3 está conformado por 2 distintas PCBs. Una PCB (Printed Circuit Board) es una lámina de fibra con pistas conductoras metálicas sobre las que se sueldan los componentes de cualquier producto electrónico. Las PCBs se pueden encontrar en todos los dispositivos independientemente de su complejidad y función.

MiDispositivoMIDI hace uso de dos PCB conectadas a través de pines (sin cables) de modo que se trata de una conexión muy robusta.

3.2.1. MDM_ControlBoard

La placa de control (*MDM_ControlBoard*) contiene la unidad principal de procesamiento del dispositivo con el microcontrolador *Arduino Leonardo* así como los dos botones de control (*Switch 0* y *Switch 1*) y un botón de Reset (*Reset Switch*). Dispone también de un pad en el que se puede soldar un módulo *Bluetooth*. Esta funcionalidad, sin embargo, no ha sido todavía implementada pero se anima a todo aquel que quiera intentarlo a experimentar con un *MiDispositivoMIDI* inalámbrico.

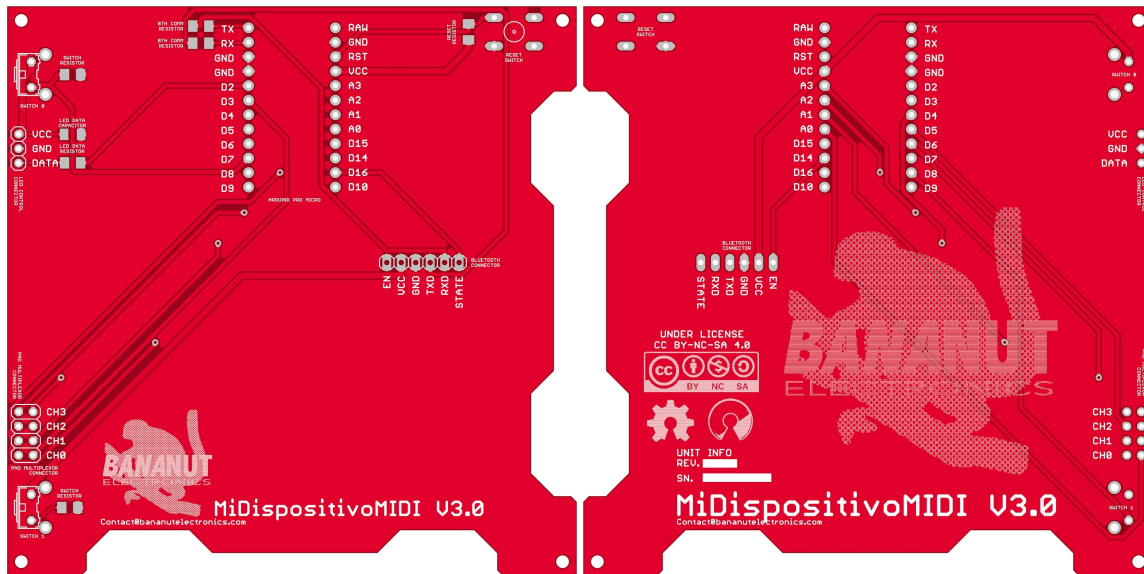


Figura 7. PCB inferior de MiDispositivoMIDI. A la izquierda se muestra la capa superior, a la derecha la inferior.

3.2.2. MDM_LEDPad

La placa del Pad Led 4x4 (*MDM_LEDPad*) conforma junto con el Pad de silicona la parte principal del dispositivo. Consiste en un Pad formado por 16 LEDs WS2812B dispuestos en una fila. El funcionamiento de estos LEDs se describe más adelante en este documento.

Cuenta además con resistencias para los switches y un multiplexor 74HC4067 que permite el control de todos los pads a través de solo 5 pines.

Una peculiaridad atractiva de *MiDispositivoMIDI* es la posibilidad de conectar hasta cuatro *MDM_LEDPad* con un solo *MDM_ControlBoard*. Esto abre la posibilidad a montar un *MiDispositivoMIDI* de 8x8 o 4x16 Pads.

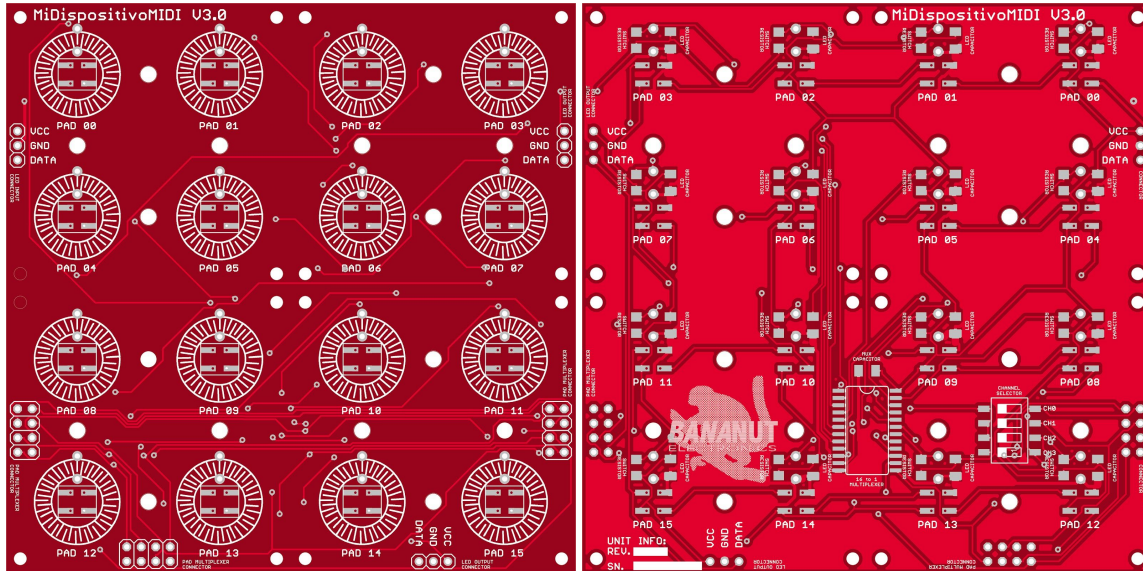


Figura 8. PCB superior de MiDispositivoMIDI. A la izquierda se muestra la capa superior, a la derecha la inferior.

Debido a la robustez de las PCBs, se han usado también para conformar la parte mecánica del dispositivo.

3.3. Elementos

Es esta sección se analiza el funcionamiento de distintos elementos relevantes de *MiDispositivoMIDI V3*.

3.3.1. LED RGB WS2812B

Los LED WS2812B popularizados por *Sparkfun* son una maravilla de la técnica que simplifica mucho el uso de LEDs RGB gracias a la posibilidad de controlarlos y organizarlos en tiras que pueden ser controladas tan solo a través de un pin (una sola línea de datos).

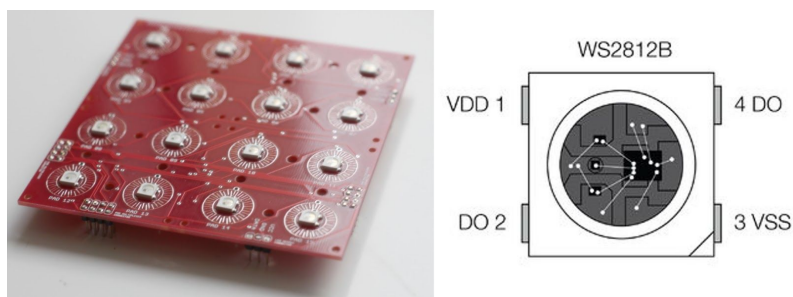


Figura 9. A la izquierda la PCB superior con sus LEDs. A la derecha el esquemático de un LED.

Su funcionamiento se basa en la integración en un solo integrado de 3 LEDs (RGB), una etapa de alimentación y un registro serie. De este modo, tan solo alimentando la tira

entera con 5V, se puede definir el color de cualquiera de los LEDs de manera individual con una precisión de 3x8bit, es decir, se pueden lograr un total de más de 16 millones de colores distintos.

Su control a través de Arduino se hace a través de la librería Adafruit_Neopixel, que a través de funciones ofrece una interfaz simplificada para trabajar con estos ICs.

3.3.2. Multiplexor 74HC4067

Un multiplexor es un circuito integrado (IC) que dispone de 2^X puertos de entrada, un puerto de salida y X puertos de selección. Su funcionamiento consiste en ofrecer a su salida el valor existente en el puerto de entrada seleccionado a través de los puertos de selección, tal y como se recoge en esta tabla genérica para un multiplexor 4to1:

ENA	SEL[1:0]	SEL[DEC]	OUTPUT
0	XX	X	GND
1	00	0	INPUT_0
1	01	1	INPUT_1
1	10	2	INPUT_2
1	11	3	INPUT_3

El Multiplexor 74HC4067 es un multiplexor de 16 canales, con lo cual a través de solo 5 pines (4 de selección y 1 de salida) se pueden controlar los 16 botones de cada Pad. La lectura se hace de forma secuencial, es decir, se establece un valor de SEL[3:0] y se lee la señal a la salida. Se lee por lo tanto solo el valor de uno de los Pads a cada momento, pero dado que la lectura de cada pad tiene una duración aproximada de 5ms, la sensación de uso es similar a que todos los Pads fuesen leídos simultáneamente en todo momento.

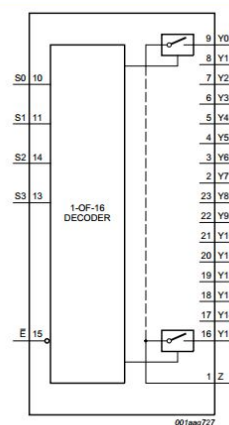


Figura 10. Esquemático de un multiplexor.

En código, esta secuenciación y uso del multiplexor se modelan de la siguiente forma:

```
/* Function void readPADs()
 * Reads the PADs through the MUX.
 * Reads a number of LEDPAD_EXT Channels starting from 0
 */
void readPADs(){
  for(int i=0; i<LEDPAD_NUM; i++){ // PAD Button Iterator
    for(int h=0; h<4; h++){
      digitalWrite(muxPin[h], ((i>>(3-h)) & 0x01)); //MUX SetUp
    }
    delay(5); // Time por MUX SetUp
    for(int j=0; j<LEDPAD_EXT; j++){ // PAD Channel Iterator
      padRead[j][i] = analogRead(padPin[j]); //Assign Reads to PAD matrix
    }
  }
}
```

En el código se escribe en cada pin de muxPin[3:0] el valor correspondiente en notación binaria de un número entre 0 y 15 (Que abarca los 16 Pads de la botonera).

3.3.3. Arduino Leonardo Micro Pro

El Arduino Leonardo Micro Pro es un modelo de Arduino de potencia reducida pero con un microprocesador ATMEGA3204 que reúne una serie de características muy interesantes para la aplicación de MiDispositivoMIDI:

- Conector microUSB.
- Capacidad de funcionar como HID (*Human Interface Device*) sin necesidad de cambiar el firmware. Puede ser reconocido por el ordenador como un controlador MIDI comercial, sin necesidad de realizar configuraciones.
- Tamaño reducido

Existen también otras versiones de Arduino, como el Arduino UNO o el Arduino MEGA2504. Cada modelo de Arduino cuenta con una serie de características ligeramente distintas de otras (tamaño, potencia, periféricos especiales, Bluetooth) por lo que antes de desarrollar cualquier sistema es necesario informarse de cuál es el objetivo del proyecto y cuál de los modelos de Arduino encaja mejor con las especificaciones.

Uno de los elementos más diferenciadores de los modelos de Arduino es el pinout. El pinout es la definición de los pines del Arduino, es decir, cual es la capacidad del procesador para comunicarse, leer o manejar periféricos (Elementos externos. Cubre desde motores a LEDs, SDcards, etc).

Arduino tiene definidas distintas categorías de pines, según su funcionalidad. Las funcionalidades principales son:

- Pin 5V: Pin de alimentación de periféricos que entrega una tensión de 5V (Salida del regulador 7805 de la placa) con respecto a GND y una corriente máxima de
- Pin 3.3V: Pin de alimentación de periféricos que entrega una tensión de 3.3V (con respecto a GND y una corriente máxima de
- Pin VIN/RAW: Pin de alimentación de entrega la tensión de alimentación del Arduino (La tensión que el Arduino recibe, por ejemplo la alimentación USB o la entregada por unas baterías)
- Pin GND: Pin de retorno GND común al Arduino (Debe, en principio, ser la referencia de tensión única del sistema).
- Pin Reset: Pin de Reset del atmega328 activo a nivel bajo. Si este pin se conecta a GND, el Arduino se reinicia.
- Pin Digital (DX): Pin para la lectura y generación de señales digitales. Además, pueden tener asociadas otras funcionalidades como comunicación I2C, SPI, o capacidad para generar señales PWM.
- Pin Analógico (AX): Pin para la lectura (No generación) de señales analógicas de 0 a 5V con una precisión de 10b (En la lectura, dará un valor de 1023 para señales de 5V y de 0 para señales de 0V). Son además pines digitales.

En general se debe tener en cuenta en el sistema:

- Nunca cortocircuitar las señales de GND, VIN, 5V y 3.3V ya que provocaría el reinicio y degradación del Arduino llegando en la mayoría de los casos a romperlo.
- Utilizar el pin de 5V para alimentar periféricos de bajo consumo (LEDs, potenciómetros, sensores, etc) y una alimentación externa para alimentar actuadores de alto consumo (motores, servomotores, etc).
- Dejar el pin de Reset desconectado o conectado a nivel alto (5V) a través de un pull-up.

A continuación se muestra gráficamente el pinout del Arduino Leonardo Micro que usa MiDispositivoMIDI V3:



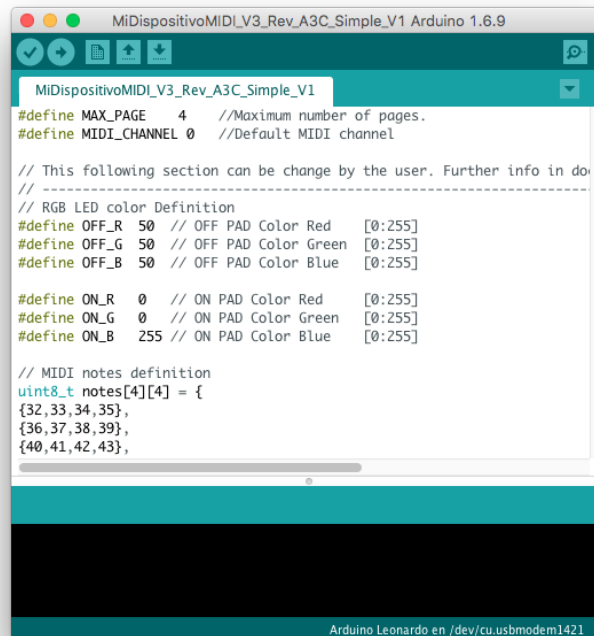
En el caso de MiDispositivoMIDI, se hace uso de los pines exclusivamente como pines de tipo digital (Para controlar tanto los LEDs como el multiplexor o la lectura de los Switches) como de tipo analógico (Para la lectura del valor de los Pads a la salida del multiplexor). Los pines 0(RX) y 1(TX), se usan para a comunicación Serial (por USB) de modo que no se deben conectar a ningún otro punto.

4.1 Introducción a la programación

MiDispositivoMIDI V3.0. User Guide Page 17

4.1.1 Código

Un código es un texto, que contiene un conjunto de instrucciones que pueden ser ejecutadas en un ordenador. En nuestro caso el ordenador será el controlador. Se observa a continuación un código de ejemplo.



```
MiDispositivoMIDI_V3_Rev_A3C_Simple_V1
#define MAX_PAGE 4 //Maximum number of pages.
#define MIDI_CHANNEL 0 //Default MIDI channel

// This following section can be change by the user. Further info in do
// -----
// RGB LED color Definition
#define OFF_R 50 // OFF PAD Color Red [0:255]
#define OFF_G 50 // OFF PAD Color Green [0:255]
#define OFF_B 50 // OFF PAD Color Blue [0:255]

#define ON_R 0 // ON PAD Color Red [0:255]
#define ON_G 0 // ON PAD Color Green [0:255]
#define ON_B 255 // ON PAD Color Blue [0:255]

// MIDI notes definition
uint8_t notes[4][4] = {
{32,33,34,35},
{36,37,38,39},
{40,41,42,43},
{44,45,46,47}}
```

Figura 12. Ejemplo de un código en el IDE de Arduino.

Cuando se escribe un código y se quiere cargar en un ordenador o microcontrolador para que pueda ser ejecutado, se requiere de un compilador. Las principales tareas que realiza el compilador son: verificar que no hay errores de sintaxis y generar un código entendible por la máquina en ceros y unos. Más adelante se explica cómo pasar de un código (texto) a algo entendible por el Arduino (código máquina).

4.1.2 Lenguajes de programación

Existen diversos lenguajes de programación, cada uno con sus ventajas e inconvenientes. Java, C, C++ o Python son algunos de los lenguajes de programación más conocidos y usados. A pesar de que tienen sus diferencias, resultan todos muy similares. El controlador MiDispositivoMIDI está pensado para ser programado en C.

4.1.3 Entornos de desarrollo

Un entorno de desarrollo es un programa que hace más fácil escribir código. Por ejemplo, permite detectar y corregir errores según se va escribiendo. Es decir, es algo así como el Word pero en la programación, ya que hace mucho más fácil escribir código.

Los entornos de desarrollo también tienen un compilador, que permite pasar de código en texto a algo entendible por el Arduino.

Arduino tiene su propio entorno de desarrollo, llamado Arduino IDE. Es la herramienta que permite compilar y cargar códigos al Arduino de vuestro controlador. Más adelante se explicará cómo descargarlo e instalarlo.

4.1.4 Estructuras de control

En programación estructurada, que es al grupo al que pertenece C, el código se ejecuta de manera secuencial. Esto significa que se va ejecutando desde el principio hasta el final, línea por línea. Cada línea o conjunto de líneas son una instrucción, como puede ser encender un LED o enviar una nota MIDI.

Dado que muchas veces no solo basta con ejecutar de principio a final, en la programación existen las **estructuras de control**, que son sentencias que permiten modificar el flujo de ejecución del programa en función de determinadas variables. Se explican las fundamentales: *if*, *while* y *for*.

El código **if**, permite ejecutar una sección de código si se cumple una determinada condición. Se muestra un ejemplo:

```
IF (variable < 30){  
    EJECUTA ACCIÓN  
}
```

Significa que si una determinada variable llamada variable es menor que 30, se ejecuta la acción encerrada entre llaves {}. Es habitual en lenguajes como C, el uso de las llaves como se indica. La variable podría ser una temperatura y la acción encender una calefacción. Es decir, si la temperatura es menor que 30, ejecutar la acción de encender la calefacción.

En el caso del **while**, se trata de una estructura de control en forma de bucle. Ejecuta la acción mientras se cumpla una determinada condición.

```
WHILE (variable < 30){  
    EJECUTA ACCIÓN.  
}
```

En el ejemplo mostrado, se ejecutaría la acción **mientras** la variable tenga un valor menor que 30. En el momento en el que la variable supere ese valor, se deja de ejecutar el bucle. La diferencia entre el *while* y el *for* es que el *while* se repite siempre que se cumpla la condición, mientras que el *if* solamente se ejecuta una vez.

Por último, el **for** se puede pensar como una especie de *while*, con la salvedad que se sabe el número de veces que se va a ejecutar la acción. El número de veces se especifica en el código.

```
for (i=0; i<5; i++){
```



```
EJECUTA ACCIÓN;  
}
```

Para los *for* se usa una variable auxiliar, la *i*. Se inicializa *i*=0 y mientras la *i* sea menor que 5 se ejecuta la acción que está dentro del *for*. En cada iteración se aumenta la *i* en una unidad, eso es lo que significa "*i*++". Por lo tanto, el ejemplo anterior se ejecutará la acción entre llaves 5 veces.

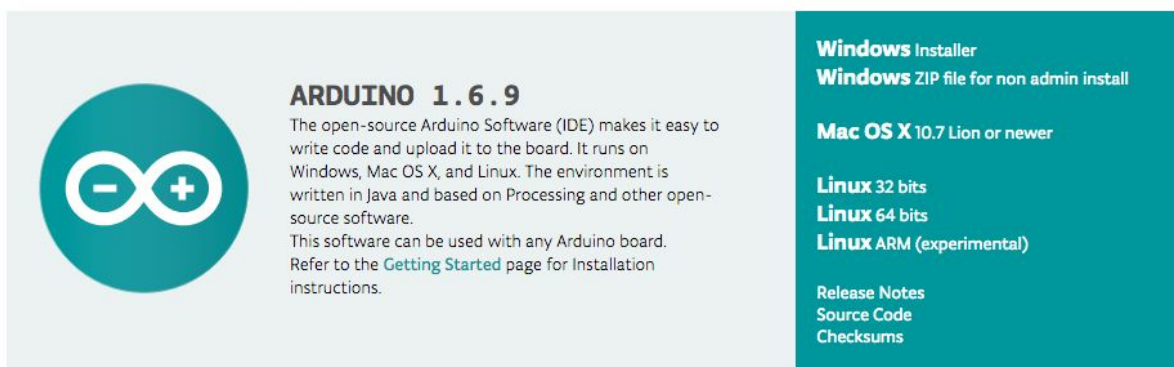
4.2 Instalación IDE Arduino

El IDE de Arduino es un entorno de desarrollo muy sencillo pensado para escribir código y cargarlo al Arduino. El objetivo de esta sección es enseñar cómo usarlo, para que podáis cargar los códigos que os demos, simplemente abrir el código y cargarlo (ver repositorio de código). A las personas con más conocimientos de programación, las invitamos a que creen sus propios códigos y los compartan.

Para descargarlo, basta con acceder al siguiente enlace y seleccionar vuestro sistema operativo. Se da la opción de donar a la causa, pero si no queréis, hacer *click* en "Just Download"

<https://www.arduino.cc/en/Main/Software>

Download the Arduino Software



ARDUINO 1.6.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

- Windows Installer
- Windows ZIP file for non admin install
- Mac OS X 10.7 Lion or newer
- Linux 32 bits
- Linux 64 bits
- Linux ARM (experimental)
- Release Notes
- Source Code
- Checksums

Figura 12. Web de Arduino

El siguiente paso es descargar dos librerías necesarias: una librería MIDI, y una librería para el control de los LEDs desarrollada por Adafruit. Los enlaces son los siguientes.

<https://github.com/arduino-libraries/MIDIUSB>

https://github.com/adafruit/Adafruit_NeoPixel

Para descargar cada librería, seleccionar “Clone or download” y después “Download ZIP”.



Figura 13. Descarga de una librería en GitHub.

Una vez descargadas las dos librerías en el IDE de Arduino, abrir el programa de Arduino. En Mac puede dar problemas si os dice que proviene de un desarrollador no identificado. Darle permiso en preferencias.



Figura 14. Ejemplo del código básico de Arduino.

Ahora solo falta decirle al programa donde tiene que localizar las librerías necesarias. Para ello, “Programa”/”Incluir librería”/”Incluir librería ZIP”.

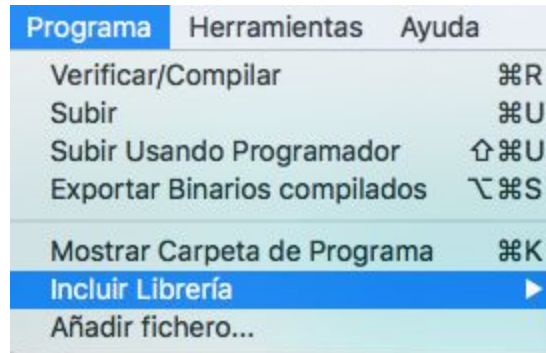


Figura 15. Menú para incluir una librería en Arduino.

Una vez seleccionadas las dos, ya estáis en condiciones de cargar cualquier código que os proporcionemos. Diferentes códigos realizarán diferentes funciones, por lo que os aconsejamos leer la documentación de cada código antes de cargarlo, para saber qué hace.

Los códigos de Arduino tienen la extensión “*.ino” y se encuentran almacenados en una carpeta con el mismo nombre que código.

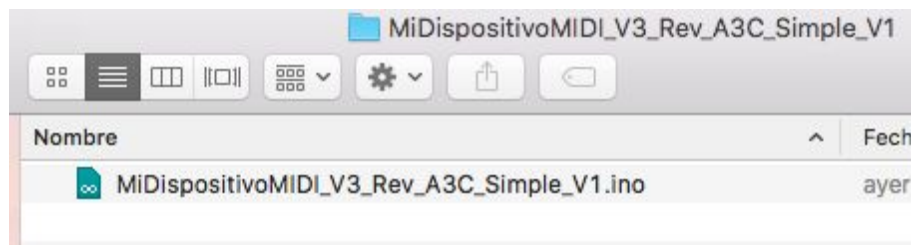


Figura 16. Forma de almacenamiento de un código en Arduino.

4.3 Funcionamiento IDE Arduino

4.3.1 Fundamentos Arduino

El lenguaje de programación de Arduino es C con ciertas particularidades. Un código de Arduino debe tener siempre dos zonas diferenciadas: *setup* y *loop*.



Figura 17. Estructura del código de Arduino.

En el **setup** se realiza la configuración del Arduino. Por ejemplo, se dice que pines son de entrada y cuales son de salida. Por otro lado el **loop** es el código que se estará ejecutando continuamente. Recordad que como se explicó antes, en la programación el código se ejecuta línea por línea, por lo que si no existiera un **loop** que estuviera continuamente ejecutándose, llegaría un momento en el que se habría ejecutado todo y el Arduino ya no realizaría nada.

4.3.2 Carga y Compilación de Sketches

Antes de cargar un código de Arduino, es necesario verificar que se tiene seleccionada la placa correspondiente. Esto es debido a que existen diferentes modelos de Arduino, y un código se compila para un modelo determinado. En el caso de MiDispositivoMIDI es necesario seleccionar "Arduino Leonardo".



Figura 18. Configuración del puerto de comunicación con Arduino.

También es necesario seleccionar el puerto al que está conectado Arduino. Si usamos Arduino conectado por USB, que es el caso, es necesario seleccionar el siguiente puerto. Es posible que en Windows aparezca otro nombre.

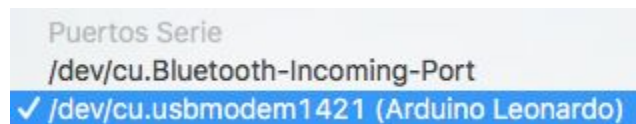


Figura 19. Puerto USB configurado en MAC.

4.4 Ejercicios

A continuación se explican paso a paso varios ejercicios, que tienen como fin guiar al usuario en los fundamentos de la programación de MiDispositivoMIDI. Van desde el manejo de los LEDs hasta el envío de mensajes MIDI.

4.4.1 Ejercicio 1

Si abrimos el código “*MiDispositivoMIDI_V3_Rev_A3C_Exercise1.ino*”, nos encontraremos con un primer ejemplo que nos permitirá saber cómo se programan los LEDs, para saber **cual** encendemos y de qué **color**.

<u>Código</u>	MiDispositivoMIDI_V3_Rev_A3C_Exercise1.ino
<u>Descripción</u>	Entender el funcionamiento de los LEDs, para poder encender un determinado LED del color que queramos. Se consideran 16 LEDs (4x4) y 3 posibles colores (rojo, verde y azul) con diferentes intensidades configurables
<u>Objetivos</u>	Saber iluminar el LED que queramos en un determinado color. Entender que cada color tiene 256 intensidades posibles. Ver de forma práctica la formación de un color a través de sus componentes rojo, verde y azul (RGB de aquí en adelante).
<u>Ejercicios</u>	<ol style="list-style-type: none">1. Modificar los colores, tanto puros (RGB) o mezcla de varios.2. Iluminar varios LEDs a la vez.

Como se puede ver en la siguiente sección de código, la matriz “ledPad” es la responsable de almacenar los colores para cada LED. El ejemplo que se muestra ilumina el primer LED, ubicado en el borde superior a la izquierda, de color verde.

```
ledPad[0][0] = 0;    //0 corresponds to the intensity of red color  
ledPad[0][1] = 255;  //1 corresponds to the intensity of green color  
ledPad[0][2] = 0;    //2 corresponds to the intensity of blue color
```

Como podemos ver, “ledPad” es una matriz de dos dimensiones, por lo que hay que especificar dos valores almacenados entre corchetes. El primer valor, siempre a cero en el ejemplo se refiere al LED. Como tenemos 16 LEDs, el valor irá desde 0 a 15. Si queremos encender el segundo LED, tendremos que poner el valor 1. La correspondencia es la siguiente:

0	1	2	3
4	5	6	7
8	9	10	11

12	13	14	15
----	----	----	----

El segundo elemento de “ledPad”, que en el va desde 0 a 2, indica las componentes de color rojo, verde y azul. La correspondencia es la siguiente:

Color	Código
Rojo	0
Verde	1
Azul	2

Y el valor que se asigna a cada elemento de “ledPad”, es la intensidad de cada color desde 0 a 255. Por lo tanto, si quisiéramos un color azul en el primero botón (referido como 0), el código sería el siguiente:

```
ledPad[0][0] = 0;    //0 corresponds to the intensity of red color  
ledPad[0][1] = 0;    //1 corresponds to the intensity of green color  
ledPad[0][2] = 255;  //2 corresponds to the intensity of blue color
```

Si usáramos un valor de 200 en vez de 255, el color sería también azul pero de menor intensidad. Se trataría de un azul claro.

Recordar que después de hacer modificaciones en la matriz “ledPad”, es necesario llamar a la función **updateLEDs()**. Esta función simplemente lleva el contenido que hayamos introducido en la matriz al controlador.

4.4.2 Ejercicio 2

En el apartado anterior hemos visto cómo iluminar un determinado LED del color que queramos. A continuación vamos a crear animaciones con los LEDs, haciendo que vayan cambiando con el tiempo, tanto de color como cual está encendido. Para ello se va a experimentar con el uso de los bucles **for**, explicados anteriormente, y la función **delay()**, que permite esperar un tiempo determinado indicado en milisegundos. Para ello se emplea el código “MiDispositivoMIDI_V3_Rev_A3C_Exercise2.ino”.

Código	MiDispositivoMIDI_V3_Rev_A3C_Exercise2.ino
Descripción	Crear animaciones animaciones con los LEDs, haciendo que los colores y LEDs iluminados varíen con el tiempo.
Objetivos	Entender el uso de bucles for y la función delay() . Confirmar haber entendido el Ejercicio 1

Ejercicios	<ol style="list-style-type: none">1. Modificar el bucle for, para verificar que se ha entendido su funcionamiento. ¿Qué pasaría si se cambia i<16 por i<20?2. Modificar el valor del <i>delay</i>, viendo cómo afecta a la animación final.
-------------------	---

La siguiente parte de código es la que es de interés. Fijémonos en que una parte ya la conocemos del ejercicio anterior.

```
for (int i = 0; i < 16; i++) {  
    ledPad[i][0] = 0;  
    ledPad[i][1] = 255;  
    ledPad[i][2] = 0;  
  
    delay(500);  
    updateLEDs();  
}
```

Lo que hace el bucle **for**, es repetir lo que hay dentro del código mientras que el valor de la **i** sea menor que 16. Además, cada vez que se ejecuta lo que hay dentro del **for**, se suma a la **i** una unidad. Justo eso es lo que simboliza **i++**, coge el valor de la **i** y súmale uno. Por lo tanto esta sección de código se ejecutará 16 veces, y la **i** tomará valores desde **0** a **15**.

Teniendo claro lo anterior, nos damos cuenta que lo que se hace con “ledPad” es exactamente a lo que hacíamos en el ejercicio 1, con la salvedad que en vez de iluminar un LED fijo, se ilumina el LED **i**. Como la **i** varía entre 0 y 15, vamos a iluminar primero el primero, después el segundo, y así hasta el 15 y último. Entre cada uno, se espera 500 milisegundos, que es lo que significa *delay(500)*.

No hay mejor forma de entenderlo que *cacharreando*. Por lo tanto, os animamos a que cambiéis diferentes valores:

- Cambiar **i<16**
- Cambiar valor *delay()*

Para los que vayan más avanzados, se sugiere intentar iluminar los LED al revés (desde el último al primero), hacer animaciones en las que intervengan varios LEDs o probar a dibujar una espiral.

4.4.3 Ejercicio 3

A continuación vamos a ver cómo podemos enviar mensajes desde el Arduino al ordenador, y como visualizarlos. Para probarlo, vamos a enviar un mensaje si se pulsa uno de los dos botones localizados a los laterales de MiDispositivoMIDI. Se emplea el código “MiDispositivoMIDI_V3_Rev_A3C_Exercise3.ino”.

Código	MiDispositivoMIDI_V3_Rev_A3C_Exercise3.ino
---------------	---

Descripción	Lectura del estado de dos botones, pulsado o no pulsado. Envío del estado a través del puerto serial al ordenador.
Objetivos	<ol style="list-style-type: none">1. Entender el uso de la estructura if-else.2. Usar el puerto Serial para comunicar al Arduino con el ordenador3. Usar la función <i>digitalRead()</i>.
Ejercicios	<ol style="list-style-type: none">1. Hacer que cuando el botón no está pulsado también se envíe un mensaje.2. Algo más complicado. Hacer si que se pulsa el botón, solamente se envíe una vez el mensaje de que ha sido pulsado.

La primera diferencia que vemos con respecto a los códigos anteriores, es que se añaden dos líneas al apartado de *setup()*. Simplemente se configura los botones de controlador como entrada (*INPUT*).

```
pinMode(SW_0_PIN, INPUT);  
pinMode(SW_1_PIN, INPUT);
```

La siguiente parte importante, es en la que se usa la función ***digitalRead()***. Esta función devuelve un 1 si el botón está pulsado y un 0 si no lo está. Como entrada, le tenemos que decir cuál es el pin que queremos leer. En el siguiente código, la primera variable almacenará el estado del primero pulsador, y la segunda el estado del segundo pulsador.

```
uint8_t switch0 = digitalRead(SW_0_PIN);  
uint8_t switch1 = digitalRead(SW_1_PIN);
```

Una vez hemos almacenado el valor, miramos si es igual a uno. En el caso de que sea igual a uno, el botón está pulsado. Si está pulsado, enviamos un mensaje por el puerto Serial. Esto es lo que hace el siguiente fragmento de código. Ver si la variable *switch1* vale 1, y si es cierto, envía por el puerto Serial el mensaje "Button 0 is pushed".

```
if (switch0 == 1){  
    Serial.println("Button 0 is pushed");  
}
```

Para ver los mensajes que envía el Arduino a nuestro ordenador, basta con hacer click en la **lupa** ubicada en la barra de herramientas del Sketch de Arduino.



Figura 20. Barra de herramientas de Arduino. Puerto serial.

Verificar que el *baudrate* sea de **9600 baudios**, aunque es la opción que está por defecto. Dicho valor es simplemente a la velocidad que se comunica nuestro Arduino con el ordenador. Valores más altos implicarán una mayor velocidad de comunicación.

4.4.4 Ejercicio 4

Para este ejemplo emplearemos el código “*MiDispositivoMIDI_V3_Rev_A3C_Exercise4.ino*” y un secuenciador o **DAW** (*Digital Audio Workstatio*). Un DAW es un programa que permite entre otras cosas, recibir mensajes MIDI, interpretarlos y reproducir sonidos en consecuencia. Algunos de los más típicos son Ableton Live, Cubase, Nuendo, Logic o Reason. En el caso de no disponer de uno de estos programas mencionados, se recomienda alguna alternativa libre de código abierto.

<u>Código</u>	MiDispositivoMIDI_V3_Rev_A3C_Exercise4.ino
<u>Descripción</u>	Experimentar con el envío de mensajes MIDI como noteOn y noteOff, viendo los diferentes parámetros que componen el mensaje. Se requerirá un software en el ordenador que permita interpretar los mensajes (Ableton Live, Logic, Cubase, Reason o similar)
<u>Objetivos</u>	Entender el concepto de canal MIDI, velocidad, y nota. Familiarizarse con el uso de noteOn y noteOff. Ver el efecto que tiene el envío de un mensaje MIDI en un secuenciador.
<u>Ejercicios</u>	<ol style="list-style-type: none"> 1. Modificar los valores de velocidad 2. Modificar las notas enviadas 3. Crear una melodía sencilla 4. Para los más avanzados, experimentar con la recepción de mensajes MIDI. Ver librería “MIDIUSB.h” y su documentación asociada.

Pero antes de entrar con secuenciadores, vamos a primero entender cómo funciona el código para enviar MIDI. Se va a probar con los mensajes de *noteOn* y *noteOff*, que permitirán enviar la pulsación de una nota, con una determinada intensidad. Un mensaje *noteOn*, tiene 3 parámetros: el **canal** MIDI, la **nota** que se quiere enviar y la **intensidad**. En la siguiente sección de código, se puede ver cómo se envía un mensaje de nota activada por el canal 0, la nota 48 y con una intensidad de 127. Recordar que la intensidad va desde 0 a 127, por lo que 127 es el valor de máxima intensidad.

```
noteOn(0, 48, 127);
MidiUSB.flush();
delay(500);
```

Como hemos visto, el fragmento de código anterior envía la nota 48 por el canal 0 y con una intensidad máxima (127). Pero claro, ¿que nota es la 48?. Aquí está la tabla de correspondencias:

Octava	Números											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B

-2	0	1	2	3	4	5	6	7	8	9	10	11
-1	12	13	14	15	16	17	18	19	20	21	22	23
0	24	25	26	27	28	29	30	31	32	33	34	35
1	36	37	38	39	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55	56	57	58	59
3	60	61	62	63	64	65	66	67	68	69	70	71
4	72	73	74	75	76	77	78	79	80	81	82	83
5	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107
7	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127	-	-	-	-

Como se puede ver, los valores de las notas musicales del mensaje *noteOn*, tienen también valores comprendidos entre 0 y 127. En la tabla anterior hemos visto la correspondencia con las notas musicales, empleando el cifrado americano. Las correspondencias son las siguientes:

<u>Cifrado americano</u>	<u>Nomenclatura latina</u>
A	LA
B	SI
C	DO
D	RE
E	MI
F	FA
G	SOL

Volvamos al código. Antes hemos visto que el mensaje *noteOn* permite enviar un mensaje de nota activada. Dicha nota seguirá sonando hasta que se envíe un mensaje de *noteOff*. En las siguientes líneas se muestra el envío de nota desactivada. Se puede ver como el valor de la intensidad toma el valor de 0 (es decir, sin pulsar). Se está enviando un mensaje de nota desactivada por el canal 0, en la nota 48 (la misma que anteriormente se envió que estaba pulsada) y con la intensidad 0. Un mensaje *noteOff* siempre tendrá una intensidad de 0.

```
noteOff(0, 48, 0);  
MidiUSB.flush();  
delay(500);
```

Por lo tanto el código del ejemplo envía un mensaje de nota activada, espera 500 ms y envía un mensaje para la misma nota pero diciendo que se ha desactivado.

Como se ha indicado anteriormente, es necesario tener en el ordenador instalado un programa que reciba estos mensajes MIDI, los procese y emita sonidos en consecuencia. Para el caso de Ableton Live, basta con configurar en *Live/Preferencias/MIDI* lo siguiente:



Figura 21. Configuración de un controlador en Ableton Live.

Si ahora añadimos una pista MIDI con un instrumento virtual cargado, y tenemos conectado a nuestro *MiDispositivoMIDI*, podremos escuchar como se está continuamente enviando una nota, parando y volviendo a enviar. Para los que no tengáis Ableton, no tengáis secuenciador o no sepáis configurarlo, contactad con nosotros y daremos soporte específico.

4.4.5 Ejercicio 5

<u>Código</u>	MiDispositivoMIDI_V3_Rev_A3C_Exercise5.ino
<u>Descripción</u>	Código muy similar al "...Simple". Es completamente compatible con Ableton. Envía MIDI y gestiona los colores de los LED en función de si el botón está pulsado o no.
<u>Objetivos</u>	Entender por encima el código y realizar modificaciones en la sección del código <i>"This following section can be changed by the user"</i> .
<u>Ejercicios</u>	<ol style="list-style-type: none">1. Cambiar las notas que envía cada botón.2. Cambiar las intensidades de cada botón.3. Cambiar los colores de los LED sin pulsar4. Cambiar los colores de los LED al pulsar

Se deja libertad al usuario para que realice las modificaciones que considere oportunas. Se recomienda modificar sólo la sección de código indicada para ello.


```
45 // This following section can be change by the user. Further info in doc.
46 // -----//
47 // RGB LED color Definition //
48 #define OFF_R 50 // OFF PAD Color Red [0:255] //
49 #define OFF_G 50 // OFF PAD Color Green [0:255] //
50 #define OFF_B 50 // OFF PAD Color Blue [0:255] //
51 //
52 #define ON_R 0 // ON PAD Color Red [0:255] //
53 #define ON_G 0 // ON PAD Color Green [0:255] //
54 #define ON_B 255 // ON PAD Color Blue [0:255] //
55 //
56 // MIDI notes definition //
57 uint8_t notes[4][4] = { //
58 {32,33,34,35}, //
59 {36,37,38,39}, //
60 {40,41,42,43}, //
61 {44,45,46,47}, //
62 }; //
63 uint8_t velocity[4][4] = { //
64 {127,127,127,127}, //
65 {127,127,127,127}, //
66 {127,127,127,127}, //
67 {127,127,127,127}, //
68 }; //
69 // -----//
```

Figura 22. Sección de código modificable.

5. Secuenciadores

5.1 Configuración Ableton

El código “...Simple” disponible en nuestro repositorio de GitHub (ver inventario de código) puede ser usado con programas como Ableton Live de manera muy sencilla. Dicho código, es también es el que se encuentra cargado por defecto en el controlador cuando es comprado.

El primer paso es conectar MiDispositivoMIDI y abrir Ableton. Ir a la sección de configuración del programa “Preferencias”. Es necesario hacer una pequeña configuración, como en cualquier controladro comercial.

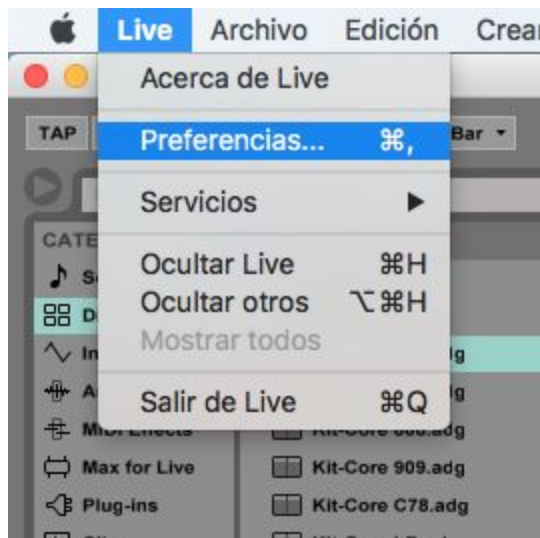


Figura 23. Configuración en Ableton de un controlador MIDI.

Seleccionar la pestaña “MIDI”.



Figura 24. Pestaña de configuración “MIDI Sync”

Verificar que “Pista” y “Remote” está en “sí” para la entrada del Arduino Leonardo. Una vez realizado, ya puedes hacer música con tu MiDispositivoMIDI.



Figura 25. Configuración de “Pista” y “Remote” para el uso de un controlador.

6. Conclusión

6.1 Extensión

MiDispositivoMIDI es un controlador de 16 botones dispuestos en una matriz de 4x4 botones con un LED por botón. Sin embargo, permite ser ampliado de manera modular hasta 4 matrices de 4x4, con diferentes configuraciones, 8x8 o 16x4. Notar que para la ampliación, es necesario un MiDispositivoMIDI V3 y 3 *MDM_LEDPad*. Actualmente no vendemos por separado las placas *MDM_LEDPad*, pero si estás especialmente interesado, contacta con nosotros en bananutelectronics@gmail.com.

6.3 Foro web

Puedes consultar en nuestro foro www.bananutelectronics.com cualquier duda, problema o sugerencias que tengas sobre nuestro controlador. Rogamos que antes de preguntar en el foro, busques si tu problema ya ha sido resuelto.

A los más experimentados programando, les invitamos a que compartan sus códigos con los demás en el foro, para que otros puedan usarlo o colaborar.

6.4 FAQ

¿Cómo uso MiDispositivoMIDI?

Nuestro controlador MiDispositivoMIDI es compatible con cualquier secuenciador del mercado (Nuendo, Ableton Live, Logic,...) ya que el funcionamiento es igual al de cualquier otro controlador comercial. Se entrega montado y programado listo para funcionar.

Tengo problemas ¿qué hago?

El controlador se entrega totalmente probado y configurado listo para funcionar. Si tienes alguna duda o problemas, te agradeceríamos que postearas un comentario en nuestro foro (www.bananutelectronics.com). Te contestaremos en la mayor brevedad posible.

No se programar, ¿es MiDispositivoMIDI para mí?

Nuestro controlador esta pensado para gente con o sin conocimientos de programación. Se entrega listo para funcionar y no se tiene porque saber programar para poder usarlo. Si quieres modificar el comportamiento del controlador, también damos códigos listos para ser cargados. Para los que sepan programar o quieran aprender, también se da material (ver "Ejercicios").

7. Inventario de vídeos

Se pueden encontrar en nuestro canal de [youtube](https://www.youtube.com). En ellos encontrarás desde cómo configurar el controlador hasta cómo poder programarlo a tu medida.

INVENTARIO VÍDEOS
Iniciación. Para quién lo quiera conectar y a funcionar.
Presentación: Características del controlador, componentes, disposición de los mismos, desmontaje y montaje.
Demo: Ejemplos con algunas de las posibilidades del controlador. Se hace uso del controlador tal y como llegaría al ser comprado.
Modificaciones. Para quién quiera programar su controlador al gusto.
Instalación: Uso del programa Simple con Ableton. Explicación de su funcionamiento y configuración con secuenciadores. Dicho programa es el que viene por defecto cargado.
(*)Ejercicio 1: Realización del primer ejercicio de la guía paso a paso. El objetivo es introducir como se realiza el control de los LEDs.
(*)Ejercicio 2: Segundo ejercicio de la guía. Introduce la estructura de control for y el uso de la función delay().
(*)Ejercicio 3: Lectura de botones y envío a través del puerto Serial información de pulsado o no pulsado.
(*)Ejercicio 4: Breve explicación del protocolo MIDI y modificaciones sobre el código para entender cómo modificar la nota y velocidad que se envía.
(*)Ejercicio 5: Modificaciones muy sencillas sobre el código por defecto Simple. Se enseña a cambiar los colores y modificar las notas que envía cada botón.
Los vídeos marcados con (*) están pendientes de realizarse. Los tendremos en el mínimo tiempo posible. Los códigos están disponibles en el apartado “Ejercicios”

8. Inventario código

Se puede acceder a diferentes códigos para MiDispositivoMIDI en nuestro repositorio de [GitHub](#). Animamos a que os los descarguéis y probéis en vuestro controlador. También a los más experimentados a hacer *forks* con nuevas funcionalidades. Para cargar los códigos, deberás descargar el entorno de desarrollo de Arduino (ver vídeo [Instalación](#)). Recuerda que no es necesario saber programar para poder usar los códigos, simplemente aprender a cargarlos. Lee la descripción del código y decide cuál crees que le puede ir mejor a tu controlador.

Nombre código	Descripción
MiDispositivoMIDI_V3_Rev_A3C_Simple_V1	Dicho código se encuentra cargado por defecto en el controlador. Cuando un botón es pulsado, se ilumina el PAD correspondiente y se envía una nota MIDI. Al ser liberado se envía el mensaje de <i>noteOff</i> . Los botones laterales cambian las notas enviadas. Ver vídeo demo .
MiDispositivoMIDI_V3_Rev_A3C_ColorTest	Realiza animaciones con los LEDs. No posee funcionalidad MIDI.
MiDispositivoMIDI_V3_Rev_A3B_StepSeq_V1	Funciona como un <i>step sequencer</i> de 16 notas.
MiDispositivoMIDI_V3_Rev_A3C_Exercise1	Ejercicio que permite familiarizarse con el manejo de los LED RGB. Se proponen varias modificaciones.
MiDispositivoMIDI_V3_Rev_A3C_Exercise2	En este ejercicio se enseña a crear animaciones sencillas con los LED. Se introduce el uso del <i>for</i> y la función <i>delay()</i> .
MiDispositivoMIDI_V3_Rev_A3C_Exercise3	Ejercicio en el que se introduce el uso de la función <i>digitalRead()</i> para leer el estado de los pulsadores laterales.
MiDispositivoMIDI_V3_Rev_A3C_Exercise4	Se explican los fundamentos del MIDI con un sencillo ejemplo.
MiDispositivoMIDI_V3_Rev_A3C_Exercise5	Ejercicio similar al código "Simple". Se propone realizar modificaciones sobre el código, cambiando los colores de los botones, las notas e intensidades que se envían. Ver la sección del código

	<p><i>“this section can be changed by the user”.</i> Dicho código contiene funcionalidades MIDI y puede ser usado con un secuenciador (Ableton, Logic).</p>
--	---