UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FINAL DE GRADO

Grau en Enginyeria Electrònica Industrial i Automàtica

SINTETIZADOR DIGITAL QUE ALMACENA LOOP

Memoria

Autor: Pablo Galiano Ruiz
Director: Joan Domingo
Convocatoria: Octubre 2017

Resum

En aquest treball es crearà un sintetitzador digital capaç de emmagatzemar loop. Aquest sintetitzador estarà compost per 29 tecles (cadascuna representant una nota musical) en forma de piano. També s'afegiran quatre efectes de so per tal de poder interactuar i tenir més possibilitats a la hora de crear. A més a més, es podrà guardar una melodia qualsevol i reproduir-la en el moment que es desitgi mentre encara es segueix tocant el sintetitzador.

Tot això serà possible gràcies al microcontrolador Arduino -que serà l'encarregat d'executar el programa creat per tal d'arribar al resultat final establert. Es a dir, tocar un conjunt de notes, emmagatzemar-les a la memòria del propi microcontrolador i posteriorment reproduir-les mentre es segueix tocant una altre melodia- i alguns elements passius. Aquests estaran controlats en tot moment per Arduino. Tot això s'utilitzarà segons el protocol MIDI, per tal de interconnectar amb altes instruments si es desitja en un futur.

A partir dels resultats obtinguts, es mostra la forma de poder construir un sintetitzador capaç d'emmagatzemar loop i posteriorment reproduir-los. S'ha de destacar, que tal i com he plantejat la forma de emmagatzemar la melodia, només es gravaran les notes de forma monofònica. Es a dir, sonaran les notes de una en una. Per aquest motiu es va afegir un regulador del tempo. Capaç de variar la seva velocitat i ajustar-se a les nostres necessitats.

A partir d'aquest primer prototip, les possibilitats d'ampliar-lo en un futur son gairebé infinites. Ja que al ser MIDI un protocol universal, hi ha molta gent inventat i creant coses noves constantment.

Resumen

En este trabajo se va a crear un sintetizador digital capaz de almacenar loops. Éste sintetizador estará compuesto por 29 teclas (cada una de ellas representando una nota musical) en forma de piano. También se añadirán cuatro efectos de sonido y así poder interactuar y tener más posibilidades a la hora de crear. A más a más, se podrá guardad una melodía cualquiera y reproducirla posteriormente en el momento que se desee mientras aún se sigue tocando el sintetizador.

Todo esto será posible gracias al microcontrolador de Arduino –que será el encargado de ejecutar el programa creado y poder llegar al resultado final propuesto. Es decir, tocar un conjunto de notas, almacenarlas en la memoria del propio microcontrolador y posteriormente reproducirla mientras se sigue tocando otra melodía- y algunos elementos pasivos. Éstos estarán controlados en todo momento por Arduino. Todo esto se utilizará según el protocolo MIDI, y así poder interactuar con otros instrumentos MIDI si se desea en un futuro

A partir de los resultados obtenidos, se muestra la forma de poder construir un sintetizador capaz de almacenar un loop y posteriormente reproducirlo. Hay que destacar, que tal y como se ha planteado la forma de almacenar la melodía, solamente se grabaran las notas de forma monofónica. Es decir, sonaran las notas de una en una. Por este motivo se añade un regulador del tempo. Capaz de variar su velocidad y ajustarse a nuestras necesidades.

A partir de este primer prototipo, las posibilidades de ampliarlo en un futuro son inmensas. Ya que al estar construido con el protocolo MIDI, que es universal, hay mucha gente inventando y creando cosas nuevas constantemente.

Abstract

The aim of this project is to create a digital synthesizer able to store a loop. Such synthesizer will have 29 keys (each representing a musical note), resembling a piano. Four sound effects will also be added, so as to provide a wider range of options when creating a musical piece. The user will also be able to save any given melody and play it at any time while still playing the synthesizer.

All of this will be possible thanks to the microprocessor Arduino, responsible for executing the software, which has been written as part of this project. To sum up, the synthesizer will be able to play a predefined set of notes, store that sequence in the microprocessor's memory and after play them while playing another melody and some passive elements. This will all be controlled by Arduino, and under the protocol MIDI, which would still allow to connect other instruments.

From the obtained results, it is possible to see how to build a synthesizer able to store a loop and play it. It has to be highlighted that, because of the way the memory is written, the sequence will only be monophonic, the notes being played one at a time. It is for this reason that a tempo regulator has been added, to control it adjusting it to our needs.

From this first prototype, the upgrading possibilities are almost infinite, given the fact that MIDI is a universal protocol and people are constantly designing new features.

Agradecimientos

A mi familia (Encarna, Juande y Rubén) por el apoyo y confianza en mí en todo momento.

A todos mis amigos de la chupipandi, por animarme y aconsejarme en momentos que pensaba que no sería posible.

Glosario

[1] FM: Frecuencia Modulada

[2] VCO: *voltaje-controlled oscillators.* Osciladores controlados por tensión

[3] VCA: *Voltage-controlled amplifier.* Amplificadores controlados por tensión

[4] MIDI: Musical Instrument Digital Interface. Interfaz Digital de Instrumentos Musicales

[5] SRAM: Static Random Acces Memory. Memoria Estática de Acceso Aleatorio

[6] EEPROM: Electrically Erasable Progammable Read Only Memory. Memoria de solo lectura programable y borrable eléctricamente

[7] ROM: Read Only Memory. Memoria de solo lectura

[8] PWM: Salidas analógicas de Modulación por ancho de pulso

[9] DAC: Conversor Digital-Analógico

[10] ADC: Conversor Analógico-Digital

[11] UART: Universal Asynchronous Reciever-Transmitter. Transmisor-receptor Asíncrono Universal

[12] IDE: Integrated Development Environment. Entorno de desarrollo integrado

[13] DAW: Digital Audio Workstation. Estación de Trabajo de Audio Digital

[14] PBB: Polibromobifenilos o Bifenilos polibromados

[15] PBDE: Polibromodifenil éteres

Índice

1. INTRODUCCIÓN

En el presente Trabajo Final de Grado, se va a crear un sintetizador en forma de teclado capaz de almacenar loop. Es sintetizador estaría destinado para gente que inicia sus primeros pasos en el mundo de la música digital. Ya que se va a intentar que sea lo más fácil e intuitivo a la hora de practicar con él.

1.1. Objetivos del trabajo

La idea premisas iniciales para diseñar y construir nuestro sintetizador son:

- Que sea un sintetizador digital. Para ello se va a utilizar el microcontrolador de Arduino.
- Que sea capaz de realizar alguna variación de sonido (Volumen, modulación, tempo, legato, balance de altavoces,…) y así hacer del sintetizador algo más interactivo.
- Tal y como indica el titulo del proyecto. Que sea capaz de almacenar una secuencia musical para posteriormente reproducirla de forma indefinida.
- Poder seguir tocando la música que se desee mientras el loop sigue sonando.

A partir de estas premisas, se seleccionarán los componentes más adecuados para realizar un instrumento económico y sencillo de utilizar.

2. SINTETIZADOR

2.1. ¿Qué es?

Un sintetizador es un instrumento musical que imita sonidos ya existentes o crea sonidos artificialmente mediante manipulación directa de corrientes eléctricas (sintetizador analógico), mediante la manipulación de una onda digital (sintetizador digital), manipulación de valores discretos usando ordenadores (sintetizador basado en software) o con la combinación de cualquiera de ellos.

Para obtener estos sonidos, se pueden obtener a partir de variaciones de voltaje (sintetizador analógico), o por ordenador (sintetizador digital).

Para modificar las señales creadas por un sintetizador se pueden utilizar diferentes métodos de síntesis:

- Síntesis aditiva: es la superposición o mezcla de ondas simples para crear otras más complejas.
- Síntesis sustractiva: es la filtración de una onda compleja. La señal pasa por un módulo de filtración que modifica su contenido armónico.
- Síntesis por modulación de frecuencias (FM[1]): Es un proceso el cual involucra por lo menos dos generadores de señal para crear o modificar una señal.
- Síntesis granular: Es basado la manipulación de pequeñas muestras de sonido.

2.2. Historia

En 1887, Thaddeus Cahill inventó el telarmonio (Figura 2.1). El cual usaba dinamos para realizar un tipo de síntesis aditiva. Aunque no fue hasta el 1906 que terminó de construirlo.

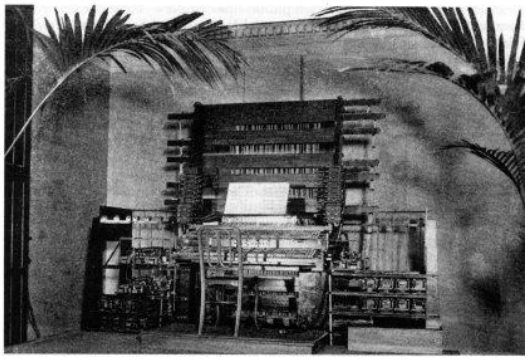


Figura 2.1. Telarmonio (fuente: Wikipedia)

Pero éste instrumento no tuvo el éxito esperado por su elevado peso y su complicada forma de manipularlo. A parte de esto, Thaddeus puso la primera piedra en la evolución de los instrumentos musicales electrónicos.

En 1919, el ruso León Teremin, inventó el Theremin. Este instrumento, se toca sin ningún tipo de contacto físico. Su funcionamiento consiste en agitar la mano entre dos antenas que emiten un campo electromagnético. El instrumento detecta la posición relativa de las manos. Con una mano se controla la frecuencia y con la otra la amplitud. Estas señales eléctricas se amplifican para posteriormente reproducirlas a un altavoz.

Más adelante, en 1928, Friedrich Trautwain construye el Trautonium. El trautonium se basaba en la síntesis substractiva. Pero no fue hasta 1952 cuando se desarrolló el primer sintetizador a dos voces. Llamado Mixturtrautonium *(Figura 2.2)*

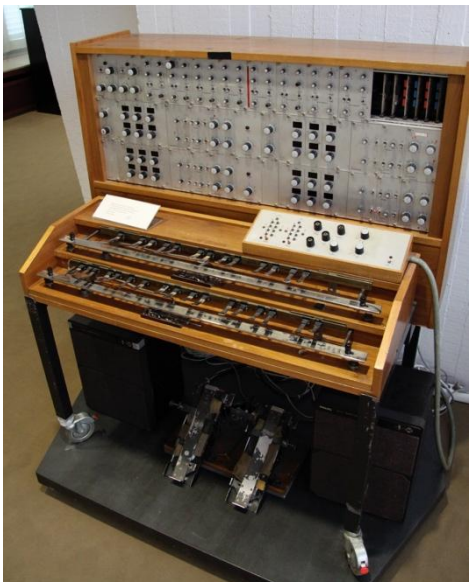


Figura 2.2. *Mixturtrautonium* (Fuente: Wikipedia)

No fue hasta la época de los cuarenta cuando el canadiense Hugh Le Caine inventa el Sackbut electrónico. Éste fue, el primer sintetizador controlado por voltaje y en él se incluía un teclado para tocarlo. Este sintetizador tenía la capacidad de controlar en tiempo real tres componentes del sonido. Volumen, tono y timbre. A más de incluir sensibilidad a la presión.

En 1956 la RCA crea el Mark I. Dicho sintetizador era el más complejo hasta la fecha ya que incluía un secuenciador de sonido. Además, fue el primero en incluir la posibilidad de grabar la música en discos multi-surcos. Años más tarde, desarrollaron el Mark II. Que mantenía las mismas características del anterior pero con muchas más posibilidades de producción.

Pero en 1963, llegó el verdadero sintetizador que revolucionó la forma que se tenía hasta la fecha. Fue el ingeniero Robert Moog y creó el sintetizador Moog *(Figura 2.3)*. Éste incluía los primeros osciladores controlados por tensión (VCO[2]) y amplificadores controlados por tensión (VCA[3]).

Figura 2.3. Sintetizador *Moog* (Fuente: Wikipedia)

Aunque el sintetizador fue un autentico revolucionario en el ámbito, tenía algunos inconvenientes. Uno de ellos es que no era un sintetizador monofónico. Tampoco podía realizar música en vivo. Por lo tanto muchos músicos empezaban a pedirle algo más a este sintetizador.

Y ante esta demanda, en 1978 la empresa Sequencial Circuits inventó el Prophet-5. Un sintetizador polifónico con un microprocesador controlado por un teclado con la posibilidad de grabar las voces en memoria.

En la época de los ochenta, la industria de los sintetizadores estaba marcada por los sintetizadores digitales y los samplers (instrumento encargado de reproducir sonidos grabados mediante teclado o secuenciador). Comparados con los sonidos analógicos, los digitales presentaban numerosas mejoras al respecto. Sonidos más definidos, ataques más claros, calidad tonal con contenido enarmónico y un control del sonido mucho más amplio.

3. MIDI[4]

MIDI es la abreviación de Musical Instrument Digital Interface. Es decir, Interfaz Digital de Instrumentos Musicales.

Se trata de un protocolo de comunicación que apareció en el año 1982, fecha en la que distintos fabricantes de instrumentos musicales electrónicos se pusieron de acuerdo en su implementación. Por tanto, éste sistema da la capacidad de comunicar diferentes instrumentos (sean del fabricante que sea) entre ellos.

3.1. Conexiones

Para poder conectar diferentes dispositivos MIDI, se tendrá que tener un mínimo de tres conexiones en nuestro instrumento como se puede observar en la figura 3.1

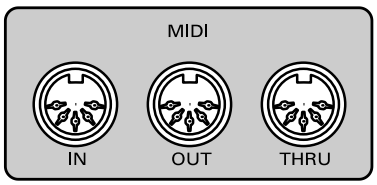


Figura 3.1. Conectores MIDI (Fuente: ite.educacion)

Y cada uno de ellos tiene una función específica:

- MIDI IN: Puerto de entrada de la información MIDI que venga de otro dispositivo.
- MIDI OUT: Puerto de salida de la información MIDI hacia otro dispositivo.
- MIDI THRU: Este puerto simplemente reenvía la misma información que haya entrado en el puerto MIDI IN

Estos conectores serán los encargados de poder transmitir datos MIDI entre ellos. Para poder comunicarse entre ellos, se tendrá que conectar el conector MIDI OUT de un instrumento hasta el conector MIDI IN del siguiente.

3.2. Mensajes

En cuanto a programación se refiere, los mensajes MIDI son un conjunto 8 bits. Es decir, un byte de información. Y existen dos tipos de bytes.

a) Bytes de estado
b) Bytes de información

La diferencia en los mensajes entre ellos es que si en el mensaje el primer bit es un 1, va a ser un byte de estado. Por el contrario, si se tiene un cero, va a ser un byte de datos.

Y los mensajes de estado se dividen en dos grupos:

1. Mensajes de canal: envían a un dispositivo específico
2. Mensajes de sistema: son recibidos por todos los equipos

En la tabla 3.1 se pueden ver todos los mensajes de estado disponibles:

Byte estado	Descripción
1000cccc	Desactivación de nota
1001cccc	Activación de nota
1010cccc	Postpulsación polifónica
1011cccc	Cambio de control
1100cccc	Cambio de programa
1101cccc	Postpulsación monofónica de canal
1110cccc	Pitch
11110000	Mensaje exclusivo del fabricante
11110001	Mensaje de trama temporal
11110010	Puntero posición de canción
11110011	Selección de canción
11110100	*Indefinido*
11110101	*Indefinido*
11110110	Requerimiento de entonación
11110111	Fin de mensaje exclusivo
11111000	Reloj de temporización
11111001	*Indefinido*
11111010	Inicio
11111011	Continuación
11111100	Parada
11111101	*Indefinido*
11111110	Espera activa
11111111	Reseteo del sistema

Tabla 3.1. Mensajes de estado

En la tabla 3.1 se puede observar que los mensajes acabas en c, hace referencia a los mensajes de canal. Por el contario, los que no llevan son mensajes de sistema.

3.3. Instrumentos

Los instrumentos MIDI se pueden clasificar en tres grandes grupos:

1. Controladores: generan los mensajes MIDI.
2. Secuenciadores: graban, reproducen y editan los mensajes MIDI
3. Unidades generadoras de sonido: Reciben los mensajes MIDI y los transforman en señales sonoras.

A parte de estos, ya existen unos valores preestablecidos por MIDI que hace referencia a diferentes tipos de instrumentos musicales. En la tabla 3.2 se pueden observar todos los instrumentos existentes capaces de reproducir en un controlador:

PIANO	CHROM. PERCUS.	ORGAN	GUITAR
01 - Acoustic Grand	09 - Celesta	17 - Drawbar Organ	25 - Nylon String Guitar
02 - Bright Acoustic	10 - Glockenspiel	18 - Percussive Organ	26 - Steel String Guitar
03 - Electric Grand	11 - Music Box	19 - Rock Organ	27 - Electric Jazz Guitar
04 - Honky-Tonk	12 - Vibraphone	20 - Church Organ	28 - Electric Clean Guitar
05 - Electric Piano 1	13 - Marimba	21 - Reed Organ	29 - Electric Muted Guitar
06 - Electric Piano 2	14 - Xylophone	22 - Accoridan	30 - Overdriven Guitar
07 - Harpsichord	15 - Tubular Bells	23 - Harmonica	31 - Distortion Guitar
08 - Clavinet	16 - Dulcimer	24 - Tango Accordian	32 - Guitar Harmonics

BASS	SOLO STRINGS	ENSEMBLE	BRASS
33 - Acoustic Bass	41 - Violin	49 - String Ensemble 1	57 - Trumpet
34 - Electric Bass(finger)	42 - Viola	50 - String Ensemble 2	58 - Trombone
35 - Electric Bass(pick)	43 - Cello	51 - SynthStrings 1	59 - Tuba
36 - Fretless Bass	44 - Contrabass	52 - SynthStrings 2	60 - Muted Trumpet
37 - Slap Bass 1	45 - Tremolo Strings	53 - Choir Aahs	61 - French Horn
38 - Slap Bass 2	46 - Pizzicato Strings	54 - Voice Oohs	62 - Brass Section
39 - Synth Bass 1	47 - Orchestral Strings	55 - Synth Voice	63 - SynthBrass 1
40 - Synth Bass 2	48 - Timpani	56 - Orchestra Hit	64 - SynthBrass 2

REED	PIPE	SYNTH LEAD	SYNTH PAD
65 - Soprano Sax	73 - Piccolo	81 - Square Wave	89 - Fantasia
66 - Alto Sax	74 - Flute	82 - Saw Wave	90 - Warm Pad
67 - Tenor Sax	75 - Recorder	83 - Syn. Calliope	91 - Polysynth
68 - Baritone Sax	76 - Pan Flute	84 - Chiffer Lead	92 - Space Voice
69 - Oboe	77 - Blown Bottle	85 - Charang	93 - Bowed Glass
70 - English Horn	78 - Skakuhachi	86 - Solo Vox	94 - Metal Pad
71 - Bassoon	79 - Whistle	87 - 5th Saw Wave	95 - Halo Pad
72 - Clarinet	80 - Ocarina	88 - Bass& Lead	96 - Sweep Pad

SYNTH EFFECTS	ETHNIC	PERCUSSIVE	SOUND EFFECTS
97 - Ice Rain	105 - Sitar	113 - Tinkle Bell	121 - Guitar Fret Noise
98 - Soundtrack	106 - Banjo	114 - Agogo	122 - Breath Noise
99 - Crystal	107 - Shamisen	115 - Steel Drums	123 - Seashore
100 - Atmosphere	108 - Koto	116 - Woodblock	124 - Bird Tweet
101 - Brightness	109 - Kalimba	117 - Taiko Drum	125 - Telephone Ring
102 - Goblin	110 - Bagpipe	118 - Melodic Tom	126 - Helicopter
103 - Echo Drops	111 - Fiddle	119 - Synth Drum	127 - Applause
104 - Star Theme	112 - Shanai	120 - Reverse Cymbal	128 - Gunshot

Tabla 3.2. Instrumentos disponibles en mensajes MIDI (Fuente: protocolossonido)

4. ARDUINO

Arduino es una plataforma de prototipos electrónica de código abierto basada en hardware y software libre.

Figura 4.1. Arduino Mega (Fuente: www.arduino.cc)

Existen diferentes tipos de placas Arduino. Según el proyecto a realizar se determinará por escoger la que más se ajuste a nuestras necesidades. En la tabla 4.1 podemos ver las características de las principales placas Arduino.

Característica de Arduino	UNO	Mega 2560	Leonardo	DUE
Tipo de microcontrolador	Atmega 328	Atmega 2560	Atmega 32U4	AT91SAM3X8E
Velocidad de reloj	16 MHz	16 MHz	16 MHz	84 MHz
Pines digitales de E/S	14	54	20	54
Entradas analógicas	6	16	12	12
Salidas analógicas	0	0	0	2 (DAC)
Memoria de programa (Flash)	32 Kb	256 Kb	32 Kb	512 Kb
Memoria de datos (SRAM[5])	2 Kb	8 Kb	2.5 Kb	96 Kb
Memoria auxiliar (EEPROM[6])	1 Kb	4 Kb	1 Kb	0 Kb

Tabla 4.1. Comparación diferentes placas Arduino.

Por otra parte, el hardware de Arduino está compuesto básicamente por un microcontrolador y elementos pasivos y activos. Un microcontrolador es un circuito

integrado programable capaz de ejecutar las órdenes grabadas en su memoria. Éste está compuesto por varios bloques funcionales, los cuales cumplen una tarea específica. Los microcontroladores tienen en su interior tres principales unidades funcionales:

1. Unidad central de procesamiento.
2. Memoria.
3. Periféricos de entrada y salida.

Algunas de las características de dicho controlador son:

- Velocidad de reloj u oscilador.
- Memoria SRAM.
- Memoria Flash.
- Memoria EEPROM.
- Memoria ROM[7].
- Entradas y salidas digitales.
- Entradas analógicas.
- Salidas analógicas por Modulación por ancho de pulso (PWM[8]).
- DAC[9].
- ADC[10].
- Buses.
- UART[11].
- Otras comunicaciones.

A parte del Hardware comentado anteriormente, Arduino consta de un software basado en un entorno de desarrollo integrado (IDE[12]) y lenguaje de programación basado en Wiring. El microcontrolador de la placa se programa mediante ordenador usando una comunicación serial mediante un convertidor de niveles RS-232 a TTL serial.

A la hora de programar con IDE, el entorno de programación ha sido empaquetado como un programa de aplicaciones. Es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Además incorpora la herramienta para cargar el programa ya compilado en la memoria flash del hardware.

5. HARDWARE

Para construir el sintetizador, he decidido utilizar una placa de Arduino Mega2560. Las características de dicha placa se pueden ver en la tabla 4.1. Para la elección de la placa, me he basado en la capacidad de la memoria SRAM. Ya que es en ésta donde se almacenaran nuestra melodía tocada previamente y reproducida posteriormente. Así que, a más memoria SRAM más tiempo de grabación se obtendrá.

Para saber la capacidad de tiempo de grabación se tendrá, basta con hacer algunos cálculos.

Se sabe que la memoria SRAM de Arduino es de 8kb. También se sabe que en MIDI ocupa 3 bytes cada vez que queremos hacer sonar una tecla. Un byte cuando la tecla es pulsada, otro byte cuando la tecla es soltada y el último la velocidad de la nota. Si nuestro programa ocupa una capacidad del 8% dejando libres un total de 7512 bytes de espacio como nos indica la figura 5.1 del IDE. Por tanto se puede saber podemos la cantidad de notas a grabar

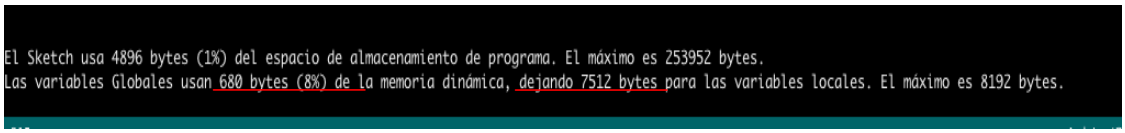


Figura 5.1. Espacio disponible de grabación

$$\frac{7512\ bytes}{3bytes/nota} = 2504\ notas \qquad \textbf{(Eq. 1)}$$

Para el sintetizador diseñado, tendrá forma de teclado y está compuesto por 29 teclas, es decir 2 octavas y media. Para diseñar dicho teclado, se ha construido una matriz de pulsadores agrupados en grupos de ocho. Ya que se utiliza un shift register para reducir el número de entradas en nuestro Arduino y éste shif register está compuesto por ocho salidas de que podremos controlar a partir de Arduino.

También se han añadido cuatro potenciómetros, cada uno con diferentes funciones que permitirán obtener mayores posibilidades a la hora de tocar el sintetizador.

Cuando se quiera comenzar a grabar una melodía, bastará con pulsar el botón de grabación (Record) y se señaliza con el encendido de un led rojo indicando que la grabación ha comenzado. A partir de aquí el sintetizador empezará a grabar la melodía deseada. Cuando se desee acabar con la grabación, pulsaremos el botón de parar la grabación (No Record) y en consecuencia, el led rojo se apagará. Posteriormente, para escuchar nuestra melodía en el momento que se desee, será suficiente con apretar el botón de reproducción (Start) y se visualizará con un led verde indicando que el sintetizador está en modo reproducción. A partir del momento que activemos la reproducción empezará a sonar la melodía grabada con anterioridad. Mientras está sonando la melodía, se puede tocar el sintetizador al mismo tiempo. Con lo que aumenta las posibilidades de crear melodías más complejas. Una vez se quiera parar de escuchar la secuencia que se está reproduciendo, basta con pulsar el botón parar de reproducir (Stop). Una vez activado, se volverá al estado inicial.

5.1. Matriz del teclado

Para hacer el teclado de 29 teclas se ha decidido en construir una matriz de ocho columnas por 4 filas tal y como se puede ver en el plano del apartado 7.

A partir de esta matriz, Arduino va a saber que tecla estamos presionando de la siguiente manera:

1. Se envía desde Arduino al shift register un pulso de activación a cada columna de nuestra matriz de forma individual. Empezando desde la primea salida hasta llegar a la última.
2. Cuando se pulsa una tecla, se cierra el circuito enviando un pulso de activación a la fila que le corresponda.
3. A través de nuestro programa de Arduino identificará la tecla pulsada y en consecuencia sabrá que nota tiene que sonar.

Para poder crear la matriz se utilizarán los siguientes elementos:

- 1 shift register 74HC595
- 29 pulsadores
- 29 diodos

5.1.1. Shift Register 74HC595 [1]

Éste chip, es un Shift Register de 8 bits serial-in, parallel-out y pertenece a una familia de chips que aceptan una entrada de bits en serie y los sacan en ocho pines paralelos. Sólo sirve para escribir señales digitales y no para leerlas. En la figura 5.2 se puede ver el patillaje correspondiente.

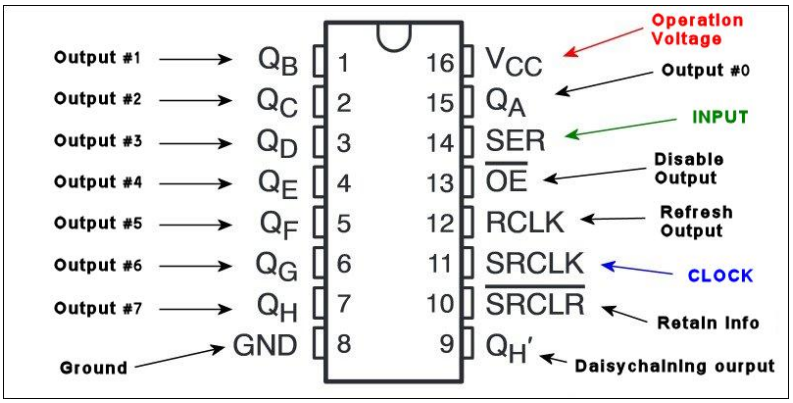


Figura. 5.2. Patillaje de Shit Register 74HC595. (Fuente:Idogendel)

El shift register funciona mediante comunicación serie síncrona. Es decir, que se usa un pin para enviar los bits en serie (pin data) y se usa el pin del reloj para indicar cuándo hay que leer el bit.

Cuando los ocho bits se han leído en el registro, el latch escribe estos bits en los pines de salida del shift register y los mantiene hasta que se reciban nuevos datos. En la figura 5.3 se puede ver el funcionamiento explicado.

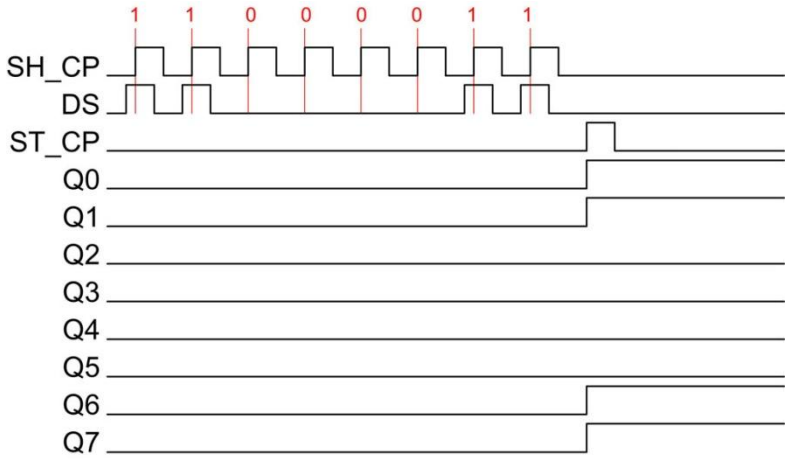


Figura 5.3. Diagrama de funcionamiento 74HC595 (Fuente: Protostack)

5.1.2. Pulsadores

Los pulsadores son los encargados de simular las teclas de nuestro teclado. Para ello se utiliza unos pulsadores con membrana como el mostrado en la figura 5.4 y así conseguir una mayor sensibilidad a la hora de tocar nuestras teclas y simular al máximo en teclado real.

Figura 5.4. Pulsador con membrana (Fuente:Adafruit)

El conexionado de este tipo de pulsadores lo podemos ver en la figura 5.5:

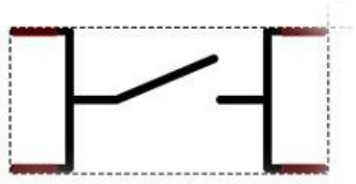


Figura 5.5. Conexionado pulsador (Fuente: felixmaocho)

5.1.3. Diodos

Estos diodos son una medida de protección contra posibles cortocircuitos. Éstos serían posibles si se tocan dos teclas de la misma fila como se puede observar en la figura 5.6

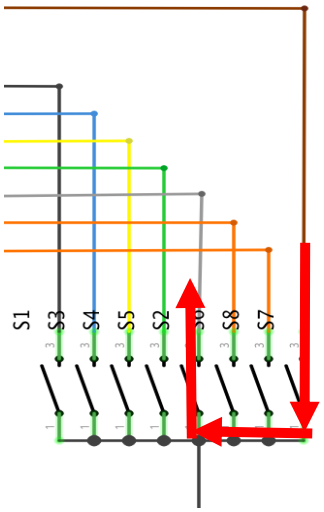


Figura 5.6. Representación de cortocircuito

Para evitar los cortocircuitos, se instala un diodo 1N4007 y así evitar corrientes inversas por los pulsadores como se observa en el plano del apartado 7.

5.2. Potenciómetros

Los potenciómetros instalados son los encargados de modificar algunas características de nuestro sintetizador.

Para ello, se eligen unos potenciómetros de un valor de 10 kΩ y lineales. Para así facilitar a la hora de programarlos en el programa de Arduino.

El conexionado de los potenciómetros los podemos ver en la figura 5.7 y la funcionalidad de cada uno de ellos en la tabla 5.1

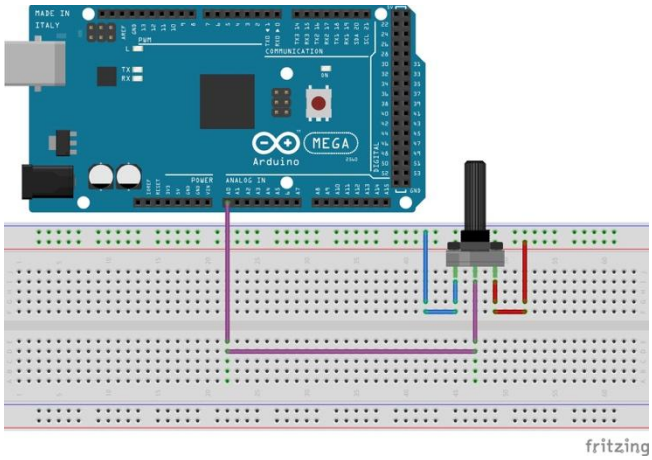


Figura 5.7. Conexionado de potenciómetro

Potenciómetro	Función
Potenciómetro 1	Volumen
Potenciómetro 2	Modulación
Potenciómetro 3	Volumen de grabación
Potenciómetro 4	Tiempo de la grabación

Tabla 5.1. Asignación de potenciómetros

5.3. Leds

Para mayor visualización y mientras se está usando el sintetizador, se instalan diferentes leds indicativos. La función de cada led la podemos ver en la tabla 5.2

Color	Indica
Rojo	Grabación pista
Verde	Reproducción grabación

Tabla 5.2. Representación de leds

Para saber el valor de resistencia de protección del diodo led, se mira en las especificaciones del propio led la tensión e intensidad en sentido directo. Y estos valores se pueden observar en la tabla 5.3

Color Led	Tensión directa (V)	Intensidad directa (mA)
Rojo	1,8	20
Verde	2,6	20

Tabla 5.3. Tensiones e intensidades directas del led

Con estos datos, se podrán calcular las resistencias de cada led con la ley de Ohm.

Led Rojo: $$R = \frac{V - V_F}{I} = 160\ \Omega$$ **(Eq. 2)**

Led Verde: $$R = \frac{V - V_F}{I} = 110\ \Omega$$ **(Eq. 3)**

Por tanto, cogiendo unos valores de resistencias estándares según la E-24 se obtendrán unos valores:

Led Rojo:	180 Ω
Led Verde:	120 Ω

5.4. USB-MIDI

El cable de la figura 5.8 va a ser el encargado de transmitir los datos MIDI a nuestro DAW[13].

Figura 5.8. Cable MIDI-USB (Fuente: Amazon)

Una vez conectado el cable a nuestro ordenador, se crea una unidad virtual MIDI para que los datos que entran por el puerto USB los reconozca como datos MIDI y no como otro tipo de datos.

5.5. Interruptor encendido/apagado

El interruptor que se puede ver en la figura 5.9, se va a encargar de encender o apagar nuestro instrumento.

Figura 5.9. Interruptor encendido/apagado. (Fuente:eBay)

Este tipo de interruptor es muy sencillo de entender. Su funcionamiento es, posición del interruptor 0 apretado. No deja pasar la corriente, por lo tanto, el instrumento permanecerá apagado. Por lo contrario, apretarlo y dejarlo con la posición marcada con 1 hacia abajo, dejara pasar la corriente y como resultado el instrumento se encenderá.

5.6. Baterías

Para alimentar nuestro teclado, se van a ofrecer dos alternativas.

1. Por pilas AA de 1,5V
2. Mediante cable USB desde el ordenador.

Se sabe que Arduino permite una alimentación de entrada entre 9 V y 12 V. Por tanto, sabiendo que cada pila proporciona 1,5 V

$$Nº\ de\ pilas = \frac{9\ V}{1{,}5\ V} = 6\ pilas \qquad \textbf{Eq(4)}$$

Por tanto, se van a necesitar un mínimo de 6 pilas para poder alimentar nuestro sintetizador.

Para poder introducir las pilas, se compra un portapilas con capacidad de 6 pilas como el de la figura 5.10.

Figura 5.10. Portapilas capacidad de 6 pilas. (Fuente: Electrodh)

6. SOFTWARE

Como ya es sabido, el sintetizador construido es digital y no analógico. Por tanto, todo el peso de este proyecto recae principalmente en nuestro código creado en el IDE de Arduino que se puede ver en el anexo.

Nuestro código de Arduino va a crear notas MIDI asignadas a cada tecla de nuestro teclado. Para ello se necesitará, siempre, un software musical (DAW) capaz de identificar estas notas MIDI. Sin él, no podremos escuchar reproducidas las notas tocadas. En este caso, se ha utilizado GarageBand.

Otra cosa que se ha de tener en cuenta, es que un ordenador no identifica automáticamente una entrada MIDI. Por ello, hay que crear una entrada de MIDI virtual. Así, nuestro ordenador reconocerá los datos de entrada por nuestro puerto USB como datos MIDI gracias al cable USB-MIDI de la figura 6.7.

Una vez tenidas estas consideraciones, y acabado todo el conexionada hardware, se va crear el código que será el encargado de dar vida al sintetizador.

6.1. Código Arduino

Antes de empezar a escribir el código, hay que tener en cuenta que para que Arduino pueda trabajar con MIDI es necesaria descargar dicha librería e introducirla en nuestro compilador. Para poder hacerlo hay que indicarlo tal y como se muestra a continuación

```
#include <MIDI.h>
```

6.1.1. Escanear Matriz

Para poder reconocer que tecla ha sido pulsada en cada momento, se crea una matriz de ocho columnas por cuatro filas. Las columnas serán las encargadas de ir enviando pulsos con nuestro Arduino de forma correlativa a cada columna y de uno en uno gracias a las propiedades del registro de desplazamiento como se puede ver a continuación:

```
int bits[] = { B00000001, B00000010, B00000100, B00001000, B00010000, B00100000, B01000000, B10000000 };
```

Al pulsar una tecla, el circuito se cerrará enviando un pulso a la fila que corresponda. Aquí Arduino habrá identificado en que fila ha sido pulsada.

Para ello, lo primero que se hace en nuestro código es decirle a Arduino en que pin está conectado cada fila.

```
const int Fila1 = 2;

const int Fila2 = 3;

const int Fila3 = 4;

const int Fila4 = 5;
```

Posteriormente se crea un bucle para ir observando en todo momento que columna es presionada de la siguiente manera. Así, se sabrá que fila y que columna se presiona, es decir, que tecla exactamente se ha presionado:

```
for (int col = 0; col < 8; col++) {

  scanColumn(bits[col]);
```

Definido el bucle capaz de identificar que tecla se presiona, se procede a relacionar la tecla que se ha presionada y se ha soltado con la nota que le pertenezca. Porque como ya es sabido, MIDI envía un byte cuando se presiona y otro byte cuando se suelta. Por tanto, se tendrá dos funciones para ello. Una para cuando se presiona y otra para cuando se suelta.

Para hacer sonar nuestro sintetizador cuando la tecla es presiona, se procede a crear cuatro variables int. Una para cada fila de nuestra matriz que estarán continuamente leyendo Arduino.

```
int grupo1 = digitalRead(Fila1);

int grupo2 = digitalRead(Fila2);

int grupo3 = digitalRead(Fila3);

int grupo4 = digitalRead(Fila4);
```

A partir de aquí, se crea una función para hacer sonar el sintetizador. Y lo primero que se hace es mirar si el estado de cualquiera de las filas ha cambiado.

```
if (grupo1 != 0 || grupo2 != 0 || grupo3 != 0 || grupo4 != 0)
```

Si ha cambiado, hay que saber cuál de ellas ha sido. Por tanto, dentro de esta principal condición, se crea una condición de un orden inferior para cada una de las filas. En ésta condición, se mira si ha habido algún cambio en la primera. Si se afirma, se sabrá que columna y que fila se pulsa. Y así hasta escanear las cuatro filas que se tienen.

```
If (grupo1 ¡= 0 && !teclaPresion[col]) {

    teclaPresion[col] = true;

  }
```

```
        if (grupo2 != 0 && !teclaPresion[col + 8]) {
          teclaPresion[col + 8] = true;
        }
        if (grupo3 != 0 && !teclaPresion[col + 16]) {
          teclaPresion[col + 16] = true;
        }
        if (grupo4 != 0 && !teclaPresion[col + 24]) {
          teclaPresion[col + 24] = true;     }
```

Y por el contrario, para saber si una tecla ha sido soltada, se crea una condición similar a la de tecla presionada.

```
if (grupo1 == 0 && teclaPresion[col]) {
     teclaPresion[col] = false;
   }
   if (grupo2 == 0 && teclaPresion[col + 8]) {
     teclaPresion[col + 8] = false;
     }
   if (grupo3 == 0 && teclaPresion[col + 16]) {
     teclaPresion[col + 16] = false;
      }
   if (grupo4 == 0 && teclaPresion[col + 24]) {
```

```
teclaPresion[col + 24] = false;

}
```

6.1.2. Sonido al pulsar tecla

Una vez sabido que tecla se ha pulsado, se procede a relacionar cada tecla pulsada a una señal MIDI. Para esto se crea las variables:

a. uint8_t notaMidi[29]
b. Boolean teclaPresion[29]
c. int velocidadNota = 127

En el primero se utiliza una variable del tipo uint8_t. Esto significa que la variable "notaMidi" es una variable de 1 byte. Tal y como se ha explicado anteriormente, MIDI utiliza datos de 1 byte cada vez que envía información. A parte de esto, se le especifica que hay 29 "uint8_t notaMidi". Es decir, uno por cada tecla.

La segunda variable es una boolean. Es decir, esta activa o no. Y nos dirá si una tecla ha sido presionada o no.

Y por último, la variable int velocidadNota nos dará el sonido de la tecla pulsada. Porque como ya es sabido, cuando una tecla es presionada en MIDI envía la información de que la nota ha sido pulsada, la nota musical y la velocidad. En el caso de que la tecla haya sido pulsada, la velocidad será igual a 127 según las especificaciones de MIDI. Y al ser soltada un cero.

Como se ha comentado anteriormente, en el protocolo MIDI existen unas tablas que dan valor a cada una de las notas musicales existentes como se puede ver en la figura 7.1.

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

Figura 7.1. Relación entra nota musical y valor MIDI. (Fuente: UPV)

Por tanto, se le dará el valor deseado a cada tecla (notaMidi) la nota que se desee. En nuestro caso, el teclado empieza por un sol y se elige la tercera octava. Que correspondería al valor 55.

```
notaMidi[0] = 55;

notaMidi[1] = 56;

notaMidi[2] = 57;

notaMidi[3] = 58;

notaMidi[4] = 59;

notaMidi[5] = 60;

notaMidi[6] = 61;

notaMidi[7] = 62;

notaMidi[8] = 63;

notaMidi[9] = 64;

notaMidi[10] = 65;

notaMidi[11] = 66;

notaMidi[12] = 67;
```

```
notaMidi[13] = 68;
notaMidi[14] = 69;
notaMidi[15] = 70;
notaMidi[16] = 71;
notaMidi[17] = 72;
notaMidi[18] = 73;
notaMidi[19] = 74;
notaMidi[20] = 75;
notaMidi[21] = 76;
notaMidi[22] = 77;
notaMidi[23] = 78;
notaMidi[24] = 79;
notaMidi[25] = 80;
notaMidi[26] = 81;
notaMidi[27] = 82;
notaMidi[28] = 83;
```

A partir de aquí y como ya se ha comentado anteriormente, se tiene que enviar el byte MIDI de nota activada. Para ello, bastará con añadir una línea dentro de la función de tecla presionada.

```
noteOn(0x91, notaMidi[col], velocidadNota);
```

Y la siguiente línea para tecla soltada.

```
noteOn(0x91, notaMidi[col], 0);
```

A parte de esto, es necesario incluir módulo que relaciona todo lo comentado con MIDI gracias a su librería.

```
void noteOn(int cmd, int pitch, int velocity) {
  Serial.write(cmd);
  Serial.write(pitch);
  Serial.write(velocity);
}
```

6.1.3. Potenciómetros

6.1.3.1. Volumen y modulación

Para el correcto funcionamiento de los potenciómetros, se crean las siguientes variables y se inicializan. La función de cada potenciómetro se puede ver en la tabla 6.1.

```
int volumen = 0;
int lastVolumen = 0;
int modulation = 0;
int lastModulation = 0;
int volumen2 =0;
int volumenBase=0;
```

El proceso de lectura de los potenciómetros es de la siguiente manera:

- Se asocia una variable al pin analógico de entrada de Arduino.
- Se hace un cambio del límite superior e inferior. Ya que la lectura de un potenciómetro en Arduino va de 0 a 1023. Y MIDI solo acepta de 0 a 127.
- Se crea una condición para saber si el valor del potenciómetro ha cambiado
- Se envía el valor del comando de MIDI (176), el valor establecido para que realice la función que se desea y el nombre de la variable con la que se está trabajando.
- Traducir esta información en información MIDI a partir del módulo MIDImessage.

```
//VOLUMEN
    volumen = analogRead(0);
     volumen = map(volumen, 0, 1023, 0, 127);
     if (volumen != lastVolumen)  {
       MIDImessage(176, 7, volumen);
       lastVolumen = volumen;

 //      MODULACIÓN
      modulation = analogRead(1);
      modulation = map(modulation, 0, 1023, 0, 127);
      if (modulation != lastModulation)
      {
       MIDImessage(176, 1, modulation);
      }
      lastModulation = modulation;
    void MIDImessage(byte command, byte data1, byte data2) {
      Serial.write(command);
      Serial.write(data1);
      Serial.write(data2);
      }
```

En el caso del volumen de la grabación, se programa de forma distinta. Ya que la grabación ya no es un mensaje MIDI. Sino una matriz guardada dentro de la memoria de Arduino. Por tanto, solamente es necesario crear las variables y ajustar el límite superior

```
//VOLUMEN GRABACIÓN
 volumen2 = analogRead(2);
volumenBase = map(volumen2, 0, 1023, 0, 127);
```

6.1.4. Velocidad Melodía

Para saber en qué tiempos se tiene que ajustar nuestro potenciómetro, se mira la tabla 7.1 y se decide que se trabajará a una velocidad entre 40 y 200 pulsaciones por minuto.

Tempo	Pulsaciones por minuto
Largo	40-60
Adagio	60-70
Andante	70-90
Moderato	90-110
Allegro	110-140
Vivace	140-160
Presto	Más de 160

Tabla 7.1. Pulsaciones por minuto

A continuación se pasa las pulsaciones por minuto a segundos. Así se sabrá el tiempo que tiempo entre notas vamos a tener

$$1\ pulso * \frac{60\ s}{40\ pulso} = 1.5\ s \qquad \textbf{(Eq. 5)}$$

$$1\ pulso * \frac{60\ s}{200\ pulso} = 0.3\ s \qquad \textbf{(Eq. 6)}$$

```
//// BPM

 BPM_Controller = analogRead(3);

BPM = map(BPM_Controller, 0, 1023, 1500, 300);
```

Para poder variar la matriz guardada en la memoria Arduino, se va a modificar la señal creada a partir de retrasos (timer). De esta manera, se controlará el tiempo entre las notas guardadas y se podrá poner el ritmo que se desee en la melodía.

```
        if (playing == true) {
          int currentMillis = millis();
          if (currentMillis - previousMillis >= BPM) {
            previousMillis = currentMillis;
            noteOn(0x91, colM[k], volumenBase);
            k++;
  }
```

6.1.5. Grabación

El procedimiento para poder guardar las notas en la memoria, es el siguiente:

1. Se crean las variables encargadas de la grabación y se dice en que pin están conectados los pulsadores correspondientes.

```
int startB = 11; //Reproducir grabación

int ledRec = 23;

int recordBState = 0;

int norecordBState = 0;

boolean recording = false;
```

```
boolean recordStart = false;

int norecordB = 6; //Parar de grabar

int j = 0;

int y = 0; //variable tamaño matriz de grabacion

uint8_t colM[200];

uint8_t dataRec;
```

Donde:

- int j: es cada nota que es guardad en la matriz.
- int y: tamaño máximo de la matriz guardada.
- uint8_t colM[200]: matriz de grabación en la que se pueden grabar 200 notas. Este valor se puede modificar hasta una máximo de 2504 notas tal y como se muestra en la ecuación 1. Pero se cree que con un conjunto de 200 notas se tiene más que suficiente para realizar una base musical.
- uint8_t dataRec: variable que guarda la nota que ha sido pulsada.

2. Se activa el modo grabación pulsando el pulsador y se crea una función específica para ello. Además, se activará en led rojo para indicar que la grabación ha comenzado.

```
        if (recordBState == HIGH && playing == false) {
          recording = true;
          digitalWrite(ledRec, HIGH);
          j = 0;
          y = 0;
 }
```

3. Cada vez que se pulsa una tecla, se guarda en la variable dataRec. Y una vez se ha pulsado el botón de grabar, empezará a garbar en la matriz "colM" de 200 posiciones cada tecla que es pulsada.

```
dataRec = notaMidi [col]

if (recording == true) { //Empezamos a grabar

  colM[j] = dataRec;
```

4. Cuando se haya decidido que ya no se quiere grabar más, se presionará el pulsador correspondiente, la melodía quedará guardada y el led se apagará.

```
        if (norecordBState == HIGH) {
          recording = false;
          digitalWrite(ledRec, LOW);
          j = 0;
 }
```

6.1.6. Reproducir grabación

Una vez se ha grabado las notas deseadas, se procede a reproducir lo grabado. Para ello, se dispone de un pulsador que activará la reproducción de la melodía y otro pulsador que parará de reproducir la melodía. El proceso para poder realizar la acción de reproducir es la siguiente:

1. Se indica en que pines están conectados los pulsadores y se crean las variables.

```
int startB = 11; //Reproducir grabación

int stopB = 7; // Parar reproducción

int ledPlay = 24;
```

```
int startBState = 0;

int stopBState = 0;

boolean playing = false;

boolean playStart = false

int k = 0;
```

2. Se crea la función que indique cuando ha sido pulsado el botón de reproducir comience a reproducir la matriz de notas guardadas en la memoria. También se podrá visualizar que se está reproduciendo la melodía mediante un led verde.

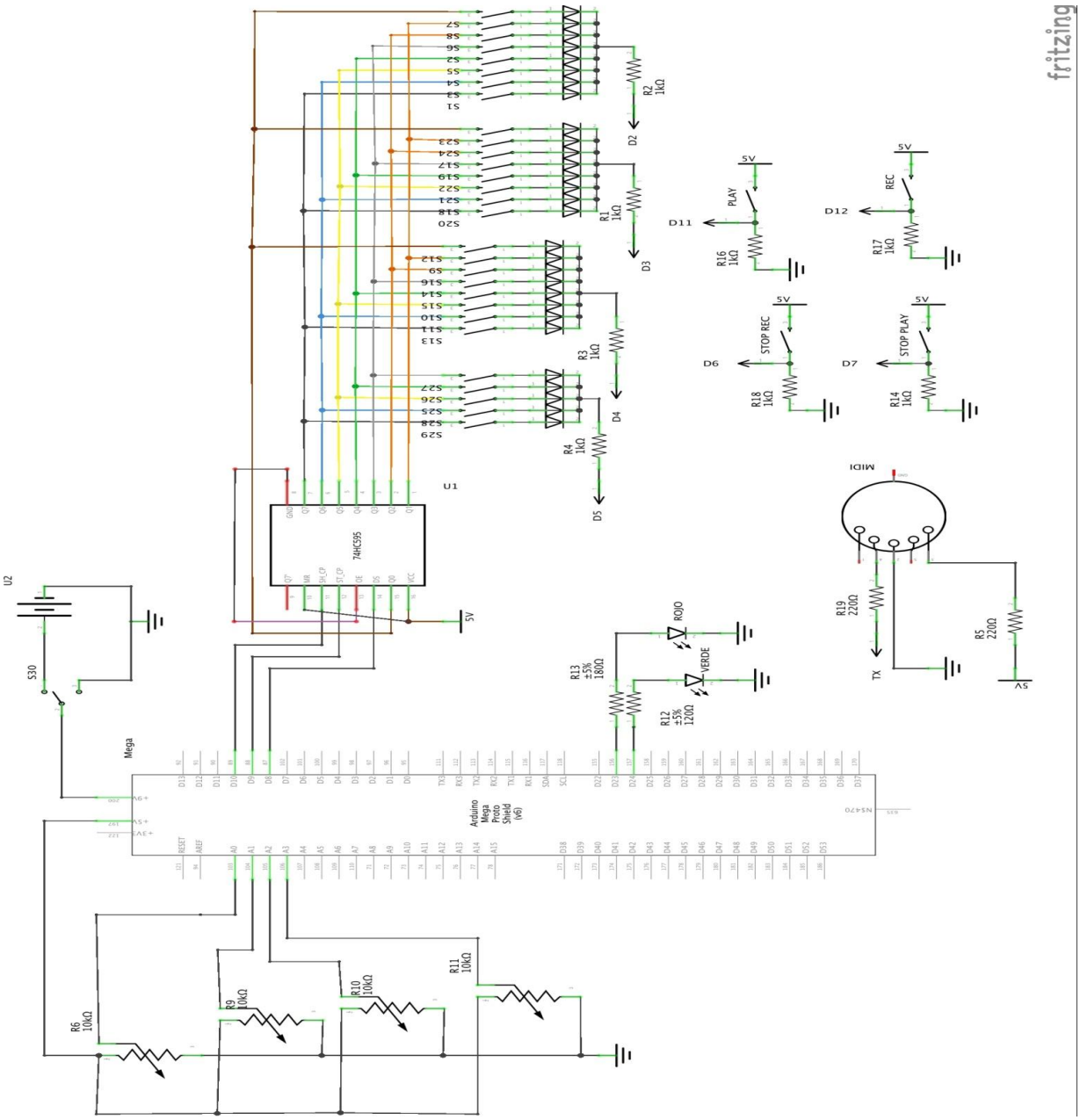
```
if (startBState == HIGH && recording == false) {
  playing = true;
  digitalWrite(ledPlay, HIGH);
  int k = 0;
 }
```

3. Para dejar de reproducir el conjunto de notas en cualquier momento, bastará con pulsar el botón de stop.

```
   if (stopBState == HIGH) {
    playing = false;
    for (int m = 55; m <= 85; m++) {
     noteOn(0x91, m, 0);
    }
    digitalWrite(ledPlay, LOW);
    j = 0;
  } }
```

7. PLANOS

En este apartado se puede ver el esquemático del circuito propuesto para desarrollar el sintetizador.



Pablo Galiano Ruiz	ESQUEMA ELECTRÓNICO
03/10/2017	UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH Escola d'Enginyeria de Barcelona Est

8. IMPACTO MEDIO AMBIENTAL

Una vez diseñado nuestro sintetizador, se tiene en cuenta el impacto medioambiental que éste pueda tener.

Todos los componentes que se han utilizados para crear el sintetizador, han seguido la normativa ROHS, que restringe ciertas sustancias peligrosas en aparatos eléctricos y electrónicos. Ésta normativa garantiza que los componentes usados están fabricados restringiendo el uso de plomo, mercurio, cadmio, cromo, PBB[14] y PBDE[15]. Cosa que reduce los residuos durante y al finalizar su vida útil.

También se ha intentado cumplir con la ley "Directiva de Residuos de Aparatos Eléctrico y Electrónicos" (WEEE[16]). En él se pretendo promover el reciclaje, reutilización y recuperación de los residuos de estos equipos para reducir su contaminación.

Por ejemplo, en nuestro sintetizador de ha reutilizados las teclas de un viejo teclado para darle un nuevo uso.

Conclusiones

Una vez finalizado el montaje del sintetizador se pueden sacar varias conclusiones al respecto.

Primero de todo el aprendizaje adquirido a la hora de realizar todo el conjunto del proyecto me ha hecho aprender a salir de problema que nunca antes me habían sucedido. Para superar cada uno de ellos he aprendido a dividirlos en pequeños apartados para llegar una solución final que creo que ha sido la óptima.

Hablando del propio proyecto, se puede llegar a la conclusión de que se han cumplido los objetivos propuestos en el apartado 1.1.

Hay que destacar que al haberlo diseñado con el protocolo MIDI, abre una amplia gama de posibilidades. Tanto a la hora de ampliar el sintetizador como a la hora de poder tocar con otros instrumentos compatibles con este sistema.

Uno de los aspectos a mejor sería la de poder almacenar más de un loop y de esta manera poder ir haciendo capas con diferentes acordes. De esta manera el usuario tendría gran cantidad de posibilidades de crear música. Incluso de hacer una canción por sí solo. Para ello habría que modificar el código Arduino de tal manera que se puedan leer las dos matrices, donde se almacenan las notas tocadas, y después reproducirlas. Por falta de tiempo no he podido aplicarlo. Pero es un aspecto a considerar de cara al futuro. Otra posible solución sería la de instalar una tarjeta SD. Dónde se podrían almacenar gran cantidad de loops sin estar preocupado de la escasa memoria SRAM de Arduino.

También se pensó en añadir la posibilidad de cambiar de instrumento. Y de hecho instale dos pulsadores capaces de cambiar a una guitara y un órgano. El problema venia cuando una vez grabada la melodía deseada, al reproducir las notas se solapaban. Ya que son notas que hasta que no sueltas la tecla no dejan de sonar. Y para grabar, se trabaja en todo momento con la activación de la nota y no la desactivación. Y por este motivo sonaban las notas por el orden tocado pero sin tener un fin en su sonido. Y por este motivo decidí suprimirlos. Aunque es un aspecto interesante en el que en un futuro habría que tener en cuenta.

Bibliografía

MIDI

- SpikenzielLabs (2011): Serial – Midi Converter: http://www.spikenzielabs.com/SpikenzieLabs/Serial_MIDI.html

- MIDI Association: GM1 Sound Set https://www.midi.org/specifications/item/gm-level-1-sound-set#instrument

- Crear Música (2010): Clases de MIDI http://www.creamusica.com.ar/Clase_de_midi.htm

- Sarah Green (2007): Beginner's Guide to MIDI http://www.softpianola.co.uk/midi.html

- AG (2015): MIDI Note Player https://www.arduino.cc/en/Tutorial/Midi

- Doxigen (2012): Arduino MIDI Library version 3.2 http://arduinomidilib.sourceforge.net/a00001.html

SHIT REGISTER

- Carly Maw y Tom Igoe (2006): Serial to Parallel Shifting-Out with a 74HC595 https://www.arduino.cc/en/Tutorial/ShiftOut

TIMER

- SM(2015): Blink without Delay https://www.arduino.cc/en/Tutorial/BlinkWithoutDelay

MATRIZ

- HeadFUZZ (2006) How to build an Open Source MIDI Keyboard https://headfuzz.co.uk/?q=midihack2

- Duane B (2012): Five Dollar Synthesiser http://rcarduino.blogspot.com.es/2012/10/five-dollar-synthesiser.html

LIBROS

- Cook, Mike (2015). Arduino Music and Audio Projects.

- Edstrom, Brent (2016). Arduino for Musicians: A Complete Guide to Arduino and Teensy Microcontrollers

TRABAJO FINAL DE GRADO

Grau en Enginyeria Electrònica Industrial i Automàtica

SINTETIZADOR DIGITAL QUE ALMACENA LOOP

Presupuesto

Autor: Pablo Galiano Ruiz
Director: Joan Domingo
Convocatoria: Octubre 2017

Presupuesto Económico

Componentes

COSTE DE COMPONENTES					
	Descripción	Cantidad	Precio Unidad (€)	Precio Total (€)	
Resistencias	10 kΩ, 5% tolerancia		8	0,03	0,24
	220 Ω, 5% tolerancia		2	0,02	0,04
	180 Ω 5% tolerancia		1	0,03	0,03
	120 Ω, 5% tolerancia		1	0,03	0,03
Potenciómetros	10 kΩ, lineal		4	0,83	3,32
Pulsadores	Pulsador goma, 6mm		33	0,68	22,44
Diodos	1N4007 1A/1000V		29	0,04	1,16
MIDI	Conector MIDI Hembra		1	0,83	0,83
	Cable MIDI-USB		1	8,59	8,59
	Tuerca		2	0,02	0,04
	Tornillo M3x08		2	0,03	0,06
Leds	Rojo 5mm		1	0,08	0,08
	Verde 5mm		1	0,08	0,08
Zócalo	Zócalo Cir. Impreso 16 pines		1	0,09	0,09
Tira de pines	Tira de pines 36C 2,54m		1	1,86	1,86
Arduino	Arduino Mega2560		1	36,18	36,18
Cables	Rojo 0,28mm		1	0,13	0,13
	Negro 0,28mm		1	0,13	0,13
	Verde 0,28mm		1	0,13	0,13
Placa Circuito impreso	Placa Circuito Impreso 30x18 cm		1	5,13	5,13
Madera	Tablero Pino 200x60x1,8 cm		2	14,95	29,90
Knobs	Pomo Plástico Negro. Diámetro 6mm		4	0,17	0,68
Shit Register	74HC595, 16 pines, Texas Instruments		1	0,59	0,59
Interruptor	Interruptor faston pequeño		1	0,53	0,53
Conector alimentación	Conector alimentación 2,1mm		1	0,99	0,99
TOTAL					113,38
				IVA 21%	23,79
				TOTAL	**137,07**

Personal

Si se ha trabajado una media de 5 horas al día durante seis meses. Y se estima un gasto de 10€/h. Se va a tener un gasto de personal según se puede ver en la tabla A.1 según el tiempo trabajado en los diferentes puntos en los que se ha hecho el sintetizador

Costes personal	
	Tiempo (h)
Diseño de hardware	95
Montaje hardware	205
Programación	315
Esquemas Electrónicos	20
Memoria	25
Total	660
Precio por horas (€/h)	10
Sueldo (€)	**6600**

Tabla A.1. Gastos de personal

Por tanto, el gasto mensual por el personal contratado serían de 1100€/mes.

Total presupuesto

Una vez se ha calculado el presupuesto de los componentes que lo forman y del personal que hace falta para prepararlo, se sumaran ambas cantidades para saber el coste total de nuestro sintetizador.

Concepto	Precio (€)
Componentes	137,07
Personal	6600
TOTAL	**6737,07**

Rentabilidad

Para saber la cantidad de sintetizadores hay que vender para tener un beneficio del 10%, habrá que hacer un sencillo cálculo donde:

$$Beneficio = 0{,}1 * 137{,}07 = 13{,}707€ \quad \textbf{(Eq. 7)}$$

Por tanto:

$$Ventas = \frac{6737{,}07}{13{,}707 + 137{,}07} = 44{,}68\ unidades \quad \textbf{(Eq. 8)}$$

Es decir, que hasta que no hayan vendido un total de 45 unidades, no se empezará a tener los beneficios deseados del 10%.

TRABAJO FINAL DE GRADO

Grau en Enginyeria Electrònica Industrial i Automàtica

SINTETIZADOR DIGITAL QUE ALMACENA LOOP

Anexos

Autor: Pablo Galiano Ruiz
Director: Joan Domingo
Convocatoria: Octubre 2017

Anexo A

Código Arduino

```
#include <MIDI.h>

#define midiChannel  0xc0;


//POTENCIOMETROS

//Inicializamos el valor de los potenciometro

int volumen = 0;

int lastVolumen = 0;


int modulation = 0;

int lastModulation = 0;


int BPM_Controller = 0;


int volumen2 =0;

int volumenBase=0;


MIDI_CREATE_DEFAULT_INSTANCE();
```

```
//TECLADO

// Pin

// Conexiones de las filas de la matriz

const int Fila1 = 2;

const int Fila2 = 3;

const int Fila3 = 4;

const int Fila4 = 5;


int startB = 11; //Reproducir grabación

int recordB = 12; //Grabar secuencia

int stopB = 7; // Parar reproducción

int norecordB = 6; //Parar de grabar


//Variables de estado

int startBState = 0;

int recordBState = 0;

int stopBState = 0;

int norecordBState = 0;
```

```
// Boolean

boolean recording = false;

boolean playing = false;

boolean playStart = false;

boolean recordStart = false;

int j = 0; //variable de matriz indica tono durante grabacion

int k = 0; //variable matriz indica tono durante reproduccion

int y = 0; //variable tamaño matriz de grabacion


// Timers

int previousMillis = 0;

int BPM;



  // Memory

  uint8_t colM[200];

  uint8_t dataRec;



  //LEDs estado

  int ledPlay = 24;
```

```
  int ledRec = 23;


  // 74HC595


  const int clock = 10;
  const int latch = 9;
  const int data = 8;


  uint8_t notaMidi[29];
  boolean teclaPresion[29];
  int velocidadNota = 127;


   // Envío de pulsos del registro de desplazamiento para poder realizar el escanee de la
  matriz
  int bits[] = { B00000001, B00000010, B00000100, B00001000, B00010000, B00100000,
  B01000000, B10000000 };


// 74HC595
void scanColumn(int value) {
  digitalWrite(latch, LOW);
  shiftOut(data, clock, MSBFIRST, value); //Shift out de 8 bits para el shift register
  digitalWrite(latch, HIGH);
```

```
}


void setup() {
  //POTENCIOMETROS
  MIDI.begin(); // Inicializamos la comunicacion midi


  //CONTROL GRABACION
  pinMode(startB, INPUT);
  pinMode(recordB, INPUT);
  pinMode(stopB, INPUT);
  pinMode(norecordB, INPUT);
  //TECLADO
  // Asigne los botones/teclas de la matriz del escáner al número de nota Midi real.


    notaMidi[0] = 55;
    notaMidi[1] = 56;
    notaMidi[2] = 57;
    notaMidi[3] = 58;
    notaMidi[4] = 59;
    notaMidi[5] = 60;
    notaMidi[6] = 61;
```

```
  notaMidi[7] = 62;
  notaMidi[8] = 63;
  notaMidi[9] = 64;
  notaMidi[10] = 65;
  notaMidi[11] = 66;
  notaMidi[12] = 67;
  notaMidi[13] = 68;
  notaMidi[14] = 69;
  notaMidi[15] = 70;
  notaMidi[16] = 71;
  notaMidi[17] = 72;
  notaMidi[18] = 73;
notaMidi[19] = 74;
notaMidi[20] = 75;
notaMidi[21] = 76;
notaMidi[22] = 77;
notaMidi[23] = 78;
notaMidi[24] = 79;
notaMidi[25] = 80;
notaMidi[26] = 81;
notaMidi[27] = 82;
```

```
notaMidi[28] = 83;


// Configuración pines entradas/salidas
pinMode(data, OUTPUT);
pinMode(clock, OUTPUT);
pinMode(latch, OUTPUT);


pinMode(Fila1, INPUT);
pinMode(Fila2, INPUT);
pinMode(Fila3, INPUT);
pinMode(Fila4, INPUT);


  pinMode(ledPlay, OUTPUT);
  pinMode(ledRec, OUTPUT);




  Serial.begin(31250);
  delay(1000);


}
```

```
void loop() {


  //          //POTENCIOMETROS

  //     //VOLUMEN


   volumen = analogRead(0);

   volumen = map(volumen, 0, 1023, 0, 127);

   if (volumen != lastVolumen) // Si el valor no es igual al ultimo valor es que se ha girado
  el potenciometro. De lo contrario,

                        //el potenciometro sigue igual y no se emite ningun mensaje MIDI

   {

     MIDImessage(176, 7, volumen);

 }        // 176 = CC command (channel 1 control change), 7 = valor establecido por MIDI,
volumen = valor que lee el potenciometro

 lastVolumen = volumen;


 //       MODULACIÓN

 modulation = analogRead(1);

 modulation = map(modulation, 0, 1023, 0, 127);

 if (modulation != lastModulation)

 {
```

```
  MIDImessage(176, 1, modulation);
 }
 lastModulation = modulation;



 ////  BPM
 BPM_Controller = analogRead(3);
 //LA MUSICA SUELE ESTAR ENTRE 40 Y 200 BPM, HACEMOS EL RESPECTIVO CALCULO DE CUANTO DURA UNA PULSACIÓN PARA CADA CASO. ESTA
 //INVERTIDO YA QUE AUMENTAR BPM ES INVERSAMENTE PROPORCIONAL A REDUCIR LA DURACION DE LA NOTA.
 BPM = map(BPM_Controller, 0, 1023, 1500, 300);


  //VOLUMEN GRABACIÓN
  volumen2 = analogRead(2);
  volumenBase = map(volumen2, 0, 1023, 0, 127);


  delay(10); // Añadimos un pequeño retardo para prevenir de pequeñas fluctuaciones
  //BUTTON STATS
  startBState = digitalRead(startB);
  recordBState = digitalRead(recordB);
  stopBState = digitalRead(stopB);
```

```
  norecordBState = digitalRead(norecordB);


  if (startBState == HIGH && recording == false) {
    playing = true;
    digitalWrite(ledPlay, HIGH);
    int k = 0;


  }
  if (stopBState == HIGH) {
    playing = false;
    for (int m = 55; m <= 85; m++) {
      noteOn(0x91, m, 0);
    }
  digitalWrite(ledPlay, LOW);
  j = 0;
}


if (recordBState == HIGH && playing == false) {
  recording = true;
  digitalWrite(ledRec, HIGH);
  j = 0;
```

```
  y = 0;


}
if (norecordBState == HIGH) {
  recording = false;
  digitalWrite(ledRec, LOW);
  j = 0;


}


if (recording == true) {//Empezamos a grabar


   colM[j] = dataRec; //graba en la matriz colM el valor de dataRec. Que sera el mmismo
que la nota presionada
  }
  //TECLADO
  for (int col = 0; col < 8; col++) {
    scanColumn(bits[col]);


    // Variables de cada fila de nuestro teclado
    int grupo1 = digitalRead(Fila1);
    int grupo2 = digitalRead(Fila2);
```

```
int grupo3 = digitalRead(Fila3);
int grupo4 = digitalRead(Fila4);
// Proceso por si se ha pulsado cualquier tecla

if (grupo1 != 0 || grupo2 != 0 || grupo3 != 0
    || grupo4 != 0) {

  if (grupo1 != 0 && !teclaPresion[col]) {
    teclaPresion[col] = true;
    dataRec = notaMidi [col];
    noteOn(0x91, notaMidi[col], velocidadNota);

}

if (grupo2 != 0 && !teclaPresion[col + 8]) {
  teclaPresion[col + 8] = true;
  dataRec = notaMidi [col + 8];
  noteOn(0x91, notaMidi[col + 8], velocidadNota);
}

if (grupo3 != 0 && !teclaPresion[col + 16]) {
```

```
    teclaPresion[col + 16] = true;
    dataRec = notaMidi [col + 16];
    noteOn(0x91, notaMidi[col + 16], velocidadNota);
   }

  if (grupo4 != 0 && !teclaPresion[col + 24]) {
    teclaPresion[col + 24] = true;
    dataRec = notaMidi [col + 24];
    noteOn(0x91, notaMidi[col + 24], velocidadNota);
   }
 }

 //  Proceso si se ha soltado cualquier tecla presionada anteriormente (released)
   if (grupo1 == 0 && teclaPresion[col]) {
     teclaPresion[col] = false;
     noteOn(0x91, notaMidi[col], 0);
     if (playing == false) {
       j++;
       y++;
     }
   }
```

```
  if (grupo2 == 0 && teclaPresion[col + 8]) {
    teclaPresion[col + 8] = false;
    noteOn(0x91, notaMidi[col + 8], 0);
    if (playing == false) {
      j++;
      y++;
    }
  }

  if (grupo3 == 0 && teclaPresion[col + 16]) {
    teclaPresion[col + 16] = false;
    noteOn(0x91, notaMidi[col + 16], 0);
    if (playing == false) {
   j++;
   y++;
  }
}

if (grupo4 == 0 && teclaPresion[col + 24]) {
  teclaPresion[col + 24] = false;
```

```
    noteOn(0x91, notaMidi[col + 24], 0);
    if (playing == false) {
      j++;
      y++;
    }
  }
}

//TECLADO DEL PLAY
if (playing == true) {
  int currentMillis = millis();
  if (currentMillis - previousMillis >= BPM) {
    previousMillis = currentMillis;
        noteOn(0x91, colM[k], volumenBase);
        k++;
       }
```

```
      //tope proteccion
       if (k >= y) {

         // EL ARRAY TIENE UN RETRASO DE 4 NOTAS (O INCLUIDO)
         k = 3;
       }
      }
    }

    void MIDImessage(byte command, byte data1, byte data2) //Command pasa los valores a través del Midi Command estándar
    {
      Serial.write(command);
      Serial.write(data1);
      Serial.write(data2);
    }
  void noteOn(int cmd, int pitch, int velocity) {
  Serial.write(cmd);
  Serial.write(pitch);
  Serial.write(velocity);
}
```

Referencias

Datasheet

[1] ***"SNx4HC595 8-bit Shift Register with 3-State output Registers"***, *Texas Instruments*.

http://www.ti.com/lit/ds/symlink/sn74hc595.pdf