First Step Piano

Versión final

Rubén Ochando Sánchez

Grado en Técnicas de Interacción Digital y Multimedia: Aplicaciones interactivas

Nombre Tutor/a de TF
Javier Julià Lundgren

Profesor/a responsable de la asignatura
Carlos Casado Martinez

19 de Junio de 2023

Universitat Oberta de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	*First Step Piano*
Nombre del autor:	*Rubén Ochando Sánchez*
Nombre del consultor/a:	*Javier Julià Lundgren*
Nombre del PRA:	*Carlos Casado Martinez*
Fecha de entrega (mm/aaaa):	*19/06/2023*
Titulación o programa:	Grado en Técnicas de Interacción Digital y Multimedia
Área del Trabajo Final:	*Aplicaciónes interactivas*
Idioma del trabajo:	*Castellano*
Palabras clave	*Partituras, Processing, Arduino*

Resumen del Trabajo

Este proyecto está enfocado en aquellas personas que estén empezando a leer partituras y quieran iniciarse en un instrumento. Con esta premisa, por un lado, se pretende crear una interfaz gráfica que permita al usuario crear, guardar e importar partituras personalizadas que le ayuden a entender las figuras musicales y su relevancia con el instrumento elegido, y por otro lado, gracias a que esta interfaz estará conectada a un piano creado con Arduino, el usuario podrá reproducir las melodías creadas, practicarlas y además podrá reconocer las teclas del piano que debe pulsar gracias a una ayuda lumínica que encontrará asociada a cada tecla.

Todas estas ayudas pretenden favorecer a que el estudiante consiga los resultados deseados sin importar sus limitaciones. De este modo, si el usuario tuviera limitaciones auditivas podría aprender a tocar sus melodías favoritas gracias a la reproducción lumínica de la canción, o si las partituras convencionales le resultaran demasiado complicadas, siempre podría personalizar las suyas propias y adaptar el aprendizaje a su ritmo y conocimientos.

En conclusión, esta aplicación tiene el cometido de acompañar al alumno en su viaje de aprendizaje musical, donde, gracias a las distintas funcionalidades que contiene, podrá usar esta herramienta en la medida que le sea necesario para superar las dificultades que vaya encontrando en el camino. La música es parte del ser humano desde sus orígenes y forma parte de nuestro día a día, hagamos que todo el mundo pueda acceder a ella.

Abstract

This project is focused on those people who are starting to read musical scores and want to start on an instrument. With this premise, on the one hand, it is intended to create a graphical interface that allows the user to create, save and import personalized scores that help them to understand the musical figures and their relevance to the chosen instrument. On the other hand, this interface will be connected to a piano created with Arduino. The user will be able to play the created melodies, practice them, and will also be able to recognize the piano keys to press thanks to a light aid that will be associated with each key.

All these aids are intended to help the student to achieve the desired results regardless of their limitations. In this way, if the user had hearing limitations, they could learn to play their favorite melodies thanks to the light reproduction of the song, or if conventional scores were too complicated, they could always personalize their own and adapt the learning to their rhythm and knowledge.

In conclusion, this application has the task of accompanying the student on his musical learning journey, where, thanks to the different functionalities it contains, he will be able to use this tool as necessary to overcome the difficulties he encounters along the way. Music is part of the human being from its origins and is part of our day to day, let's make it accessible to everyone.

Índice

1. Introducción

1.1. Contexto y justificación del Trabajo

A lo largo de los años se ha descrito a la música de innumerables formas y, en función de la época y las melodías reproducidas, la música ha podido ser amada, detestada, aclamada e incluso censurada. A pesar de todo ello, nadie ha podido describir ese sentimiento que te recorre el cuerpo como si de un rayo se tratara cuando una melodía nos acompaña en la alegría, la tristeza y el sosiego.

Pese a todo lo dicho, solo hay una cosa que podemos tener clara, la música siempre será parte de la cultura del ser humano y, en mi opinión, merece la pena esforzarse en facilitar el entendimiento de esta a todos los públicos, independientemente de los recursos que puedan tener a su alcance o las dificultades físicas de cada usuario.

Principalmente, el mercado de aprendizaje de piano se basa en academias privadas o conservatorios públicos que utilizan como método de enseñanza el método de aprendizaje por medio de libros de texto acompañados de la guía de un profesor. Actualmente, estos métodos de enseñanza están algo distanciados de la digitalización y no aceptan con facilidad ayudas mediante dispositivos digitales.

La mayoría de los usuarios que utilizan ayudas digitales para tocar un instrumento son estudiantes de música que buscan un nuevo reto o tiempo de ocio, usuarios casuales que entienden la música como un hobbie o usuarios que se están iniciando y quieren probar a tocar ese instrumento. Actualmente, es complicado encontrar accesorios digitales de aprendizaje en los centros educativos institucionalizados.

¿No sería genial que existiera una aplicación que permitiera al profesor personalizar las lecciones impartidas, y adaptar el estilo de música y el ritmo de la clase a las necesidades y preferencias del alumno con una interfaz sencilla e intuitiva que sirviera tanto para el estilo clásico, como para el moderno?

En el mercado actual podemos encontrar:

1.Ayudas para el aprendizaje basadas en pegatinas de colores colocadas directamente en las teclas para que el usuario sepa qué nota corresponde a cada tecla.

- **Ventajas**. El coste es muy asequible.
- **Desventajas**. No puedes aplicar la ayuda y quitarla, una vez que aplicas la ayuda es para todas las canciones. No llega a ser un aprendizaje progresivo y encontramos demasiada información en la propia tecla, puede resultar confuso.

2.Existen aplicaciones con vídeos enfocados a las manos del pianista que describen cómo se toca la canción deseada.

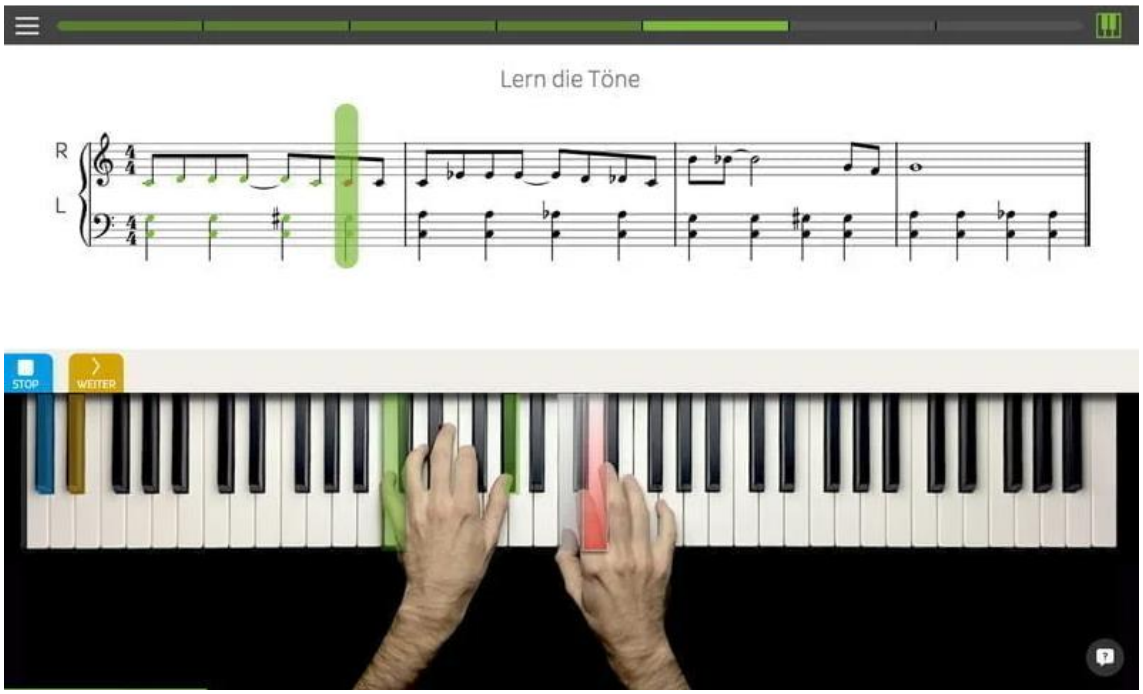
- **Ventajas**. Es una interfaz visual muy práctica.
- **Desventajas**. El aprendizaje se vuelve más complicado cuando tienes que visualizar a la vez la partitura, la mano del ejemplo y tu propia mano. Encontramos limitación de canciones disponibles.

3.También podemos encontrar teclados que tienen preinstaladas entre 50 y 100 canciones y que iluminan las teclas a tocar por el usuario.

- **Ventajas**. Aprendizaje muy intuitivo.
- **Desventajas**. Los teclados con estas funciones están muy limitados y acotados a un tipo específico de dispositivos. Número de canciones limitado,

predefinidas por el fabricante y sin posibilidad de crear nuevas partituras. Coste elevado.

4.También podemos encontrar aplicaciones como “Skoove”. La base de esta aplicación es la interpretación de partituras.



- **Ventajas**. Ofrece aprendizaje técnico.
- **Desventajas**. Clases limitadas por la aplicación y canciones preasignadas. No hay espacio para la adaptación personalizada en función de las necesidades del alumno.

5.Para los más pequeños, existen aplicaciones orientadas al aprendizaje de los instrumentos y a los sonidos de los animales. Un ejemplo es “Easy Music” de la propuesta educativa Montessori.

- **Ventajas**. Es una propuesta muy llamativa y amigable para el público al que va dirigida.
- **Desventajas**. Bajo nivel de aprendizaje. Desarrollo de aprendizaje limitado.

1.2. Objetivos del Trabajo

Una vez que estamos posicionados en el marco de la educación musical, vamos a direccionar el objetivo de este trabajo hacia un aprendizaje basado en la relación que existe entre el profesor y el alumno, y la importancia de poder personalizar las clases por parte del profesor en función de las necesidades específicas de cada alumno.

Así, el objetivo de este trabajo consiste en crear una aplicación desarrollada con el programa “Processing” que permita crear lecciones personalizadas para que los niños se familiaricen con el lenguaje musical de una forma divertida y teniendo en cuenta las necesidades y particularidades de cada alumno.
Desde esta aplicación se podría tanto crear como importar partituras interactivas, lo que permite ampliar el abanico de aprendizaje a todos los estilos y gustos musicales. Decimos que son interactivas porque el alumno podrá reproducir dichas partituras en la aplicación y hacer un seguimiento de ellas en la pantalla del dispositivo utilizado, donde se mostrará la partitura con las notas a ejecutar y la progresión temporal de la misma con la reproducción sonora de cada nota en el momento que debe ser tocada.

Hasta este punto, la interfaz, por un lado va a permitir al profesor crear lecciones pudiendo adaptar la dificultad y el estilo musical, y por otro lado, va a permitir al alumno aprender a tocar sus melodías favoritas al mismo tiempo que descifra las partituras mostradas en pantalla. Además, estas lecciones, podrán ser exportadas e importadas para practicarlas las veces que se requiera.

Pero ¿Con qué va a poder practicar el alumno? Para cerrar el círculo, esta aplicación estará conectada a un dispositivo creado con Arduino que contendrá siete botones, siete luces led y un altavoz (un piano con las siete notas musicales). Este dispositivo permitirá al alumno reproducir la melodía importada o creada, practicarla, y además, como cada luz led estará asignada a un botón, reproducir la melodía de forma lumínica, lo que le mostrará al alumno mediante señales lumínicas qué teclas debe pulsar y en qué tiempo.

Este último aspecto es crucial para la experiencia de usuario basada en la accesibilidad, ya que hay muchos alumnos que tendrán diferentes dificultades de aprendizaje. Con este dispositivo, incluimos la posibilidad de aprender partituras de forma lumínica, un feedback en tiempo real de si se han pulsado las teclas adecuadas, y la posibilidad de practicar las melodías independientemente de que la interfaz esté activada o no en pantalla.

En resumen, el objetivo de este proyecto es desarrollar una aplicación basada en la experiencia de usuario personalizada y accesible, que permita a un alumno dar sus primeros pasos en el mundo del lenguaje musical mientras aprende a leer partituras disfrutando de sus canciones favoritas.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

1.3.1 Dimensión sostenibilidad.

Debido a que este TFG se basa en la creación de una aplicación, la sostenibilidad de este producto estaría totalmente ligada a la del dispositivo en la que se utilice, que, en primera instancia, estaríamos hablando de un ordenador. Por tanto, estaríamos ante un proyecto basado en energía eléctrica que se podría extraer de energías renovables. Este hecho relacionaría este desarrollo con el ODS 7 – Afforfable and clean energy, el ODS 9 - Industry, innovation and infrastructure; el ODS 11 – Sustainable cities and communities, el ODS 12 - Responsible consumption and production el ODS 13 - Climate action.

El único impacto negativo que tendría este proyecto es el plástico utilizado para crear el dispositivo piano con Arduino. Este aspecto podría solventarse reciclando el producto al final de su tiempo de vida y uso. El resto de los recursos que se utilizan por parte del usuario forman parte de los recursos medios que cualquier alumno suele tener en casa. Ya que esta aplicación no exige altos rendimientos técnicos: un ordenador básico con una salida con puerto USB será más que suficiente.

1.3.2 Dimensión comportamiento ético y de responsabilidad social(RS).

Debido a que esta aplicación podría ejecutarse en cualquier ordenador personal, este proyecto pretende que, esta herramienta para iniciarse en mundo del lenguaje musical sea accesible para todos los usuarios que tengan acceso a un ordenador, ya sea en casa, o en un centro educativo.

Creo que, el hecho de que se pueda utilizar desde cualquier ordenador tiene un impacto social positivo e inclusivo para cualquier alumno que quiera aprender a tocar un instrumento, sin distinción de su nivel adquisitivo o el instrumento que quiera a prender a tocar.

Además, las razones y las motivaciones sobre los que se ha construido la aplicación tienen su base en la adaptación y personalización de las lecciones de música en función de las necesidades del alumno. No todos los alumnos aprenden con la misma facilidad, no todos tienen los mismos gustos musicales, no todos tienen las mismas capacidades físicas, y no todos van a progresar de la misma forma.

Gracias a las posibilidades de aprendizaje que ofrece esta aplicación, los métodos con los que podemos abordar las necesidades del alumno son enormes. Con la misma herramienta, un alumno que aprenda a tocar una canción descifrando una partitura convencional en pantalla podrá avanzar tan rápido como un alumno que tenga menos capacidades y decida aprenderla mediante su interpretación y reproducción lumínica en el dispositivo de “Arduino”.

1.3.3 Dimensión diversidad, género y derechos humanos.

Este Proyecto no hace distinciones de género, ya que cualquier persona con un ordenador puede disfrutar de esta aplicación. Por otro lado, este TFG sí que está altamente influenciado por la accesibilidad por parte del usuario y la posibilidad de adaptación para alumnos con diversidad funcional.

La dificultad de las lecciones será adaptada en todo momento por el profesor y la forma de interpretar las partituras que el alumno debe tocar podrá discernir entre la interpretación de una partitura convencional, la representación lumínica por medio de luces led asociados a las siete notas musicales, o la interpretación sonora de las frecuencias al reproducir la partitura tanto en la aplicación como en el dispositivo.

La finalidad de este desarrollo es que cualquier alumno que desee aprender a tocar un instrumento, encuentre la manera de aprender que mejor se adapte a sus necesidades personales. Por estos motivos, diría que este TFG está relacionado con el ODS 5 – Gender equality y el ODS 10- Reduced inequalities.

1.4. Enfoque y método seguido

El enfoque que se ha seguido en la realización de este proyecto se ha basado en la metodología Scrum. Por tanto, desde un principio, se ha adaptado el desarrollo de esta aplicación a las condiciones del proyecto y se ha buscado tener una respuesta eficaz y rápida a los inconvenientes que han ido surgiendo.

De la misma forma, el desarrollo ha sido dividido en pequeños sprints, que se han ido abordando y revisando en función de las fechas planificadas en la tabla que podemos ver en el siguiente apartado.

1.5. Planificación del Trabajo

Calendario de tareas en función de los progresos realizados:

Prioridad	Tarea	Fecha Inicio	Fecha fin	Progreso
1	Inicio y creación de aplicación.	26/03/2023	26/03/2023	**Hecho**
2	Interfaz para crear partituras.	26/03/2023	09/04/2023	**Hecho**
3	Interfaz de importación y exportación de partituras.	09/4/2023	23/04/2023	**Hecho**
4	Visualización de la partitura en pantalla.	09/04/2023	23/04/2023	**Hecho**

5	Reproducción de la partitura en frecuencias.	09/04/2023	30/04/2023	**Hecho**
6	Creación del piano con “Arduino”.	09/04/2023	07/05/2023	**Hecho**
7	Conectar el dispositivo con “Processing” y crear una interfaz capaz de mandar órdenes a “Arduino”.	09/04/2023	07/05/2023	**Hecho**
8	Reproducción lumínica de la partitura	30/04/2023	14/05/2023	**Hecho**
9	Animación que ayude a saber en pantalla en que parte de la partitura nos encontramos.	30/04/2023	28/05/2023	**Hecho**
10	Últimas pruebas, limpieza de código y finalización de la memoria.	26/03/2023	04/06/2023	**Hecho**

Hasta ahora, el progreso de las tareas es el adecuado en función de las fechas propuestas, pero las prioridades marcan los desarrollos que se abordarán en primer lugar y los que podrían quedar fuera del desarrollo en el caso de que el tiempo propuesto no sea suficiente. Se prioriza tener una aplicación estable en la que se puedan crear partituras en función de las necesidades del usuario y que, además, sean visualizables en lenguaje musical para que se puedan descifrar con el instrumento que el usuario elija. Por tanto, es totalmente indispensable que el desarrollo llegue hasta el punto número cinco para que el proyecto goce de coherencia y consistencia.

1.6. Breve sumario de productos obtenidos

- Aplicación First Step Piano
- Dispositivo a semejanza de un piano de siete teclas creado con Arduino.

1.7. Breve descripción de los otros capítulos de la memoria

- **Materiales y métodos**. En este apartado especificaremos los aspectos técnicos utilizados para llevar a cabo el desarrollo de la aplicación.

- **Resultados.** En este apartado describiremos las distintas funcionalidades que ofrece esta aplicación.

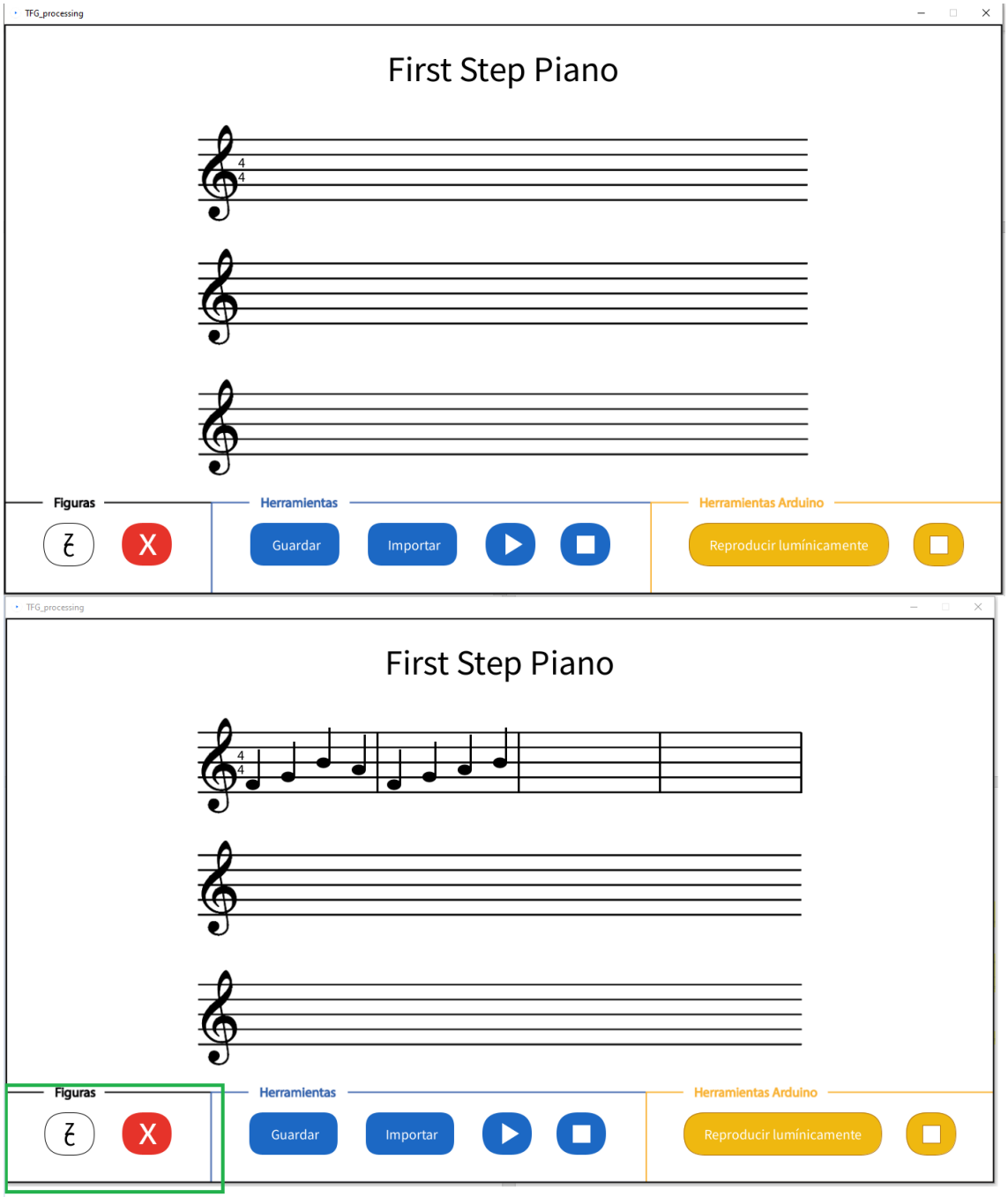
- Conclusiones y trabajos futuros . En este apartado detallaremos hasta donde hemos llegado y hasta donde podemos llegar.

2. Materiales y métodos

2.1. Processing

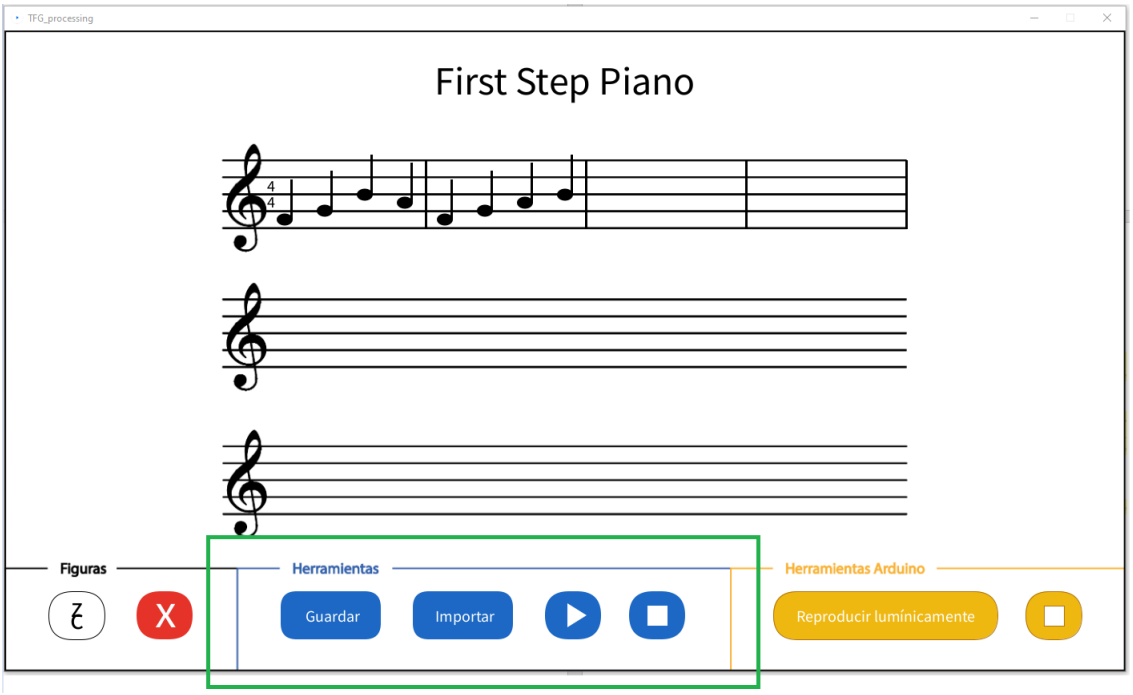
Con la ayuda del programa Processing 3.5.4, he creado un proyecto para realizar todos los desarrollos que conciernen a la creación de la aplicación y la gestión de las partituras.

Una vez que tenemos una mesa de trabajo donde construir la aplicación, he diseñado con Ilustrator una plantilla con pentagramas en blanco para la interfaz de creación de partituras. Con ello, he desarrollado una interfaz donde se pueden crear partituras desde cero añadiendo figuras negras y silencios donde se desee.



Siempre y cuando el usuario haga clic dentro de espacios en pantalla asignados a los pentagramas, el algoritmo registrará cada posición x e y, y en función de dicha posición dentro del pentagrama detectará en que pentagrama ha ubicado la figura el usuario y a que nota pertenece. Una vez recocidos los valores, guardará dicha información en una variable de tipo tabla que el algoritmo pintará en pantalla, y además, podrá ser posterior mente modificada, guardada o borrada.

Así mismo, la interfaz cuenta con un menú de color azul con el que podemos guardar, exportar las partituras creadas en un archivo CSV y reproducir la melodía creada en los altavoces que tengamos configurados en nuestro ordenador parar dicha reproducción. De este modo, si hacemos clic en el botón "Exportar", como resultado de la siguiente partitura, tendríamos un archivo ".csv" como este:

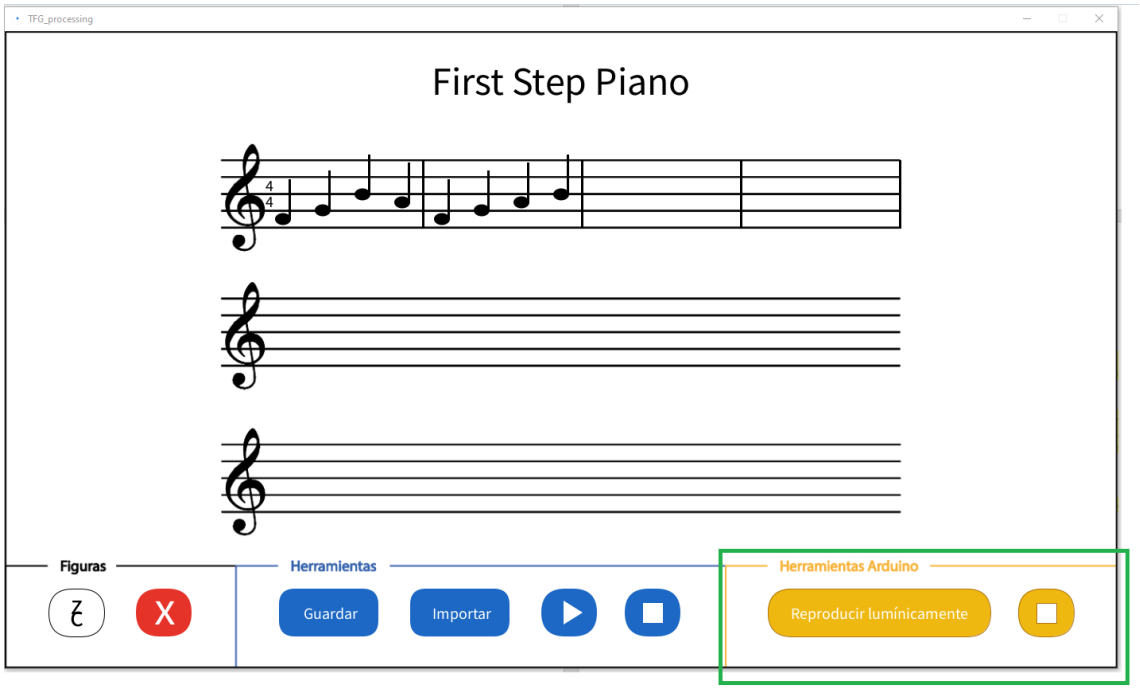


Nota	Inicio intervalo	Fin intervalo
FA	0	1
SOL	1	2
SI	2	3
LA	3	4
FA	4	5
SOL	5	6
LA	6	7
SI	7	8

Ahora que ya tenemos la una partitura guardada o importada el usuario podría reproducir la melodía en pantalla, lo que originaría que el programa iniciase una variable clase que gestiona el tiempo y leyese fila por fila la información de la

tabla. Si el segundo en el que se encuentra el programa está dentro de algún intervalo informado en la tabla, buscará la nota asignada y reproducirá el archivo de audio de la nota que corresponda.

Una vez que tenemos la interfaz terminada en Processing el siguiente paso ha sido conectar la funcionalidad de esta interfaz al dispositivo creado en Arduino mediante el puerto de conexión del dispositivo. De esta forma, ya podríamos utilizar el menú de color amarillo que permiten al usuario reproducir de forma lumínica la melodía creada y parar su reproducción para practicarla.



En el apartado “Anexos” de este mismo documento podremos leer con detalle el algoritmo que se ejecuta en Proccessing para realizar estos procesos. En lo que respecta a Arduino, en el siguiente apartado explicaremos cómo se ha realizado dicha conexión.

2.2. Arduino

2.2.1 Hardware

Siguiendo con las especificaciones del desarrollo, he avanzado en la creación del dispositivo que vamos a utilizar a modo de piano con 7 teclas/notas, 7 luces led y un altavoz (Piezo). El resultado es un dispositivo más que funcional que permite asociar cada botón y luz a un evento en la herramienta de software que nos proporciona Arduino. A continuación detallamos los aspectos técnicos del dispositivo creado.

Imagen del dispositivo:

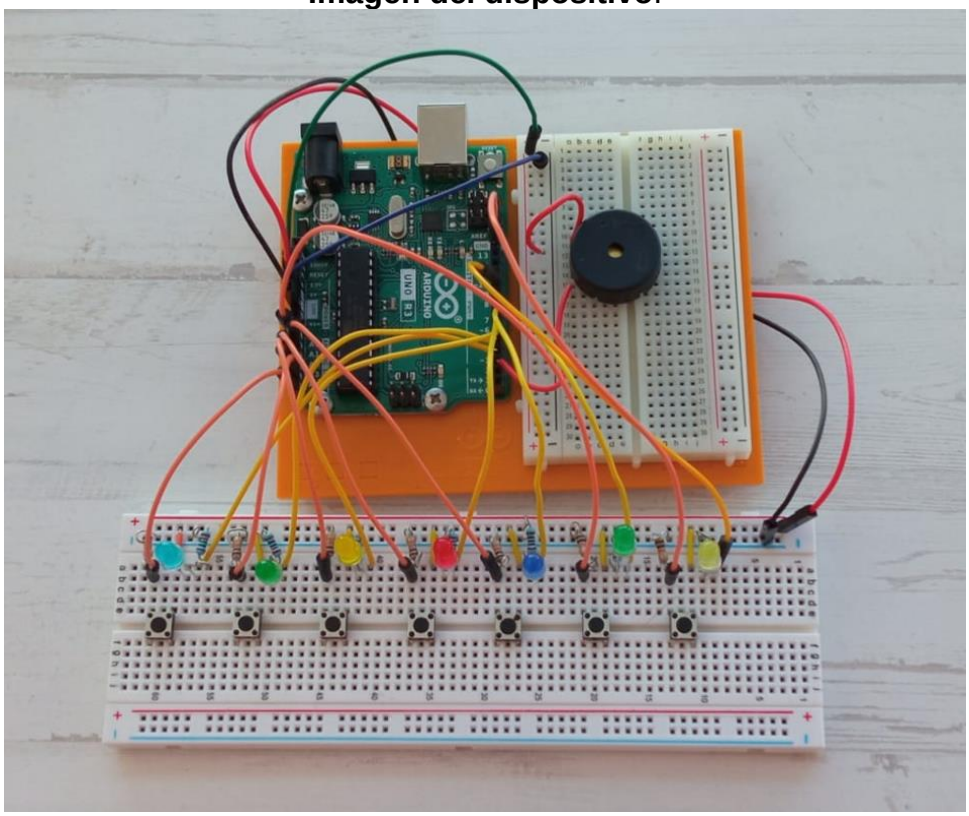
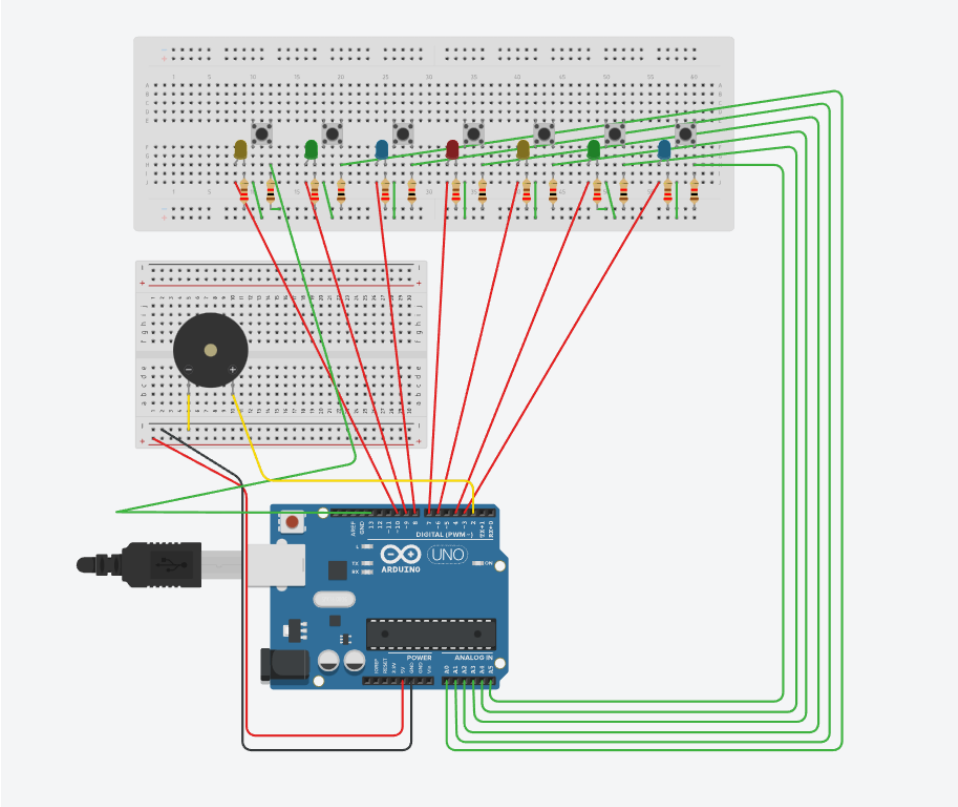
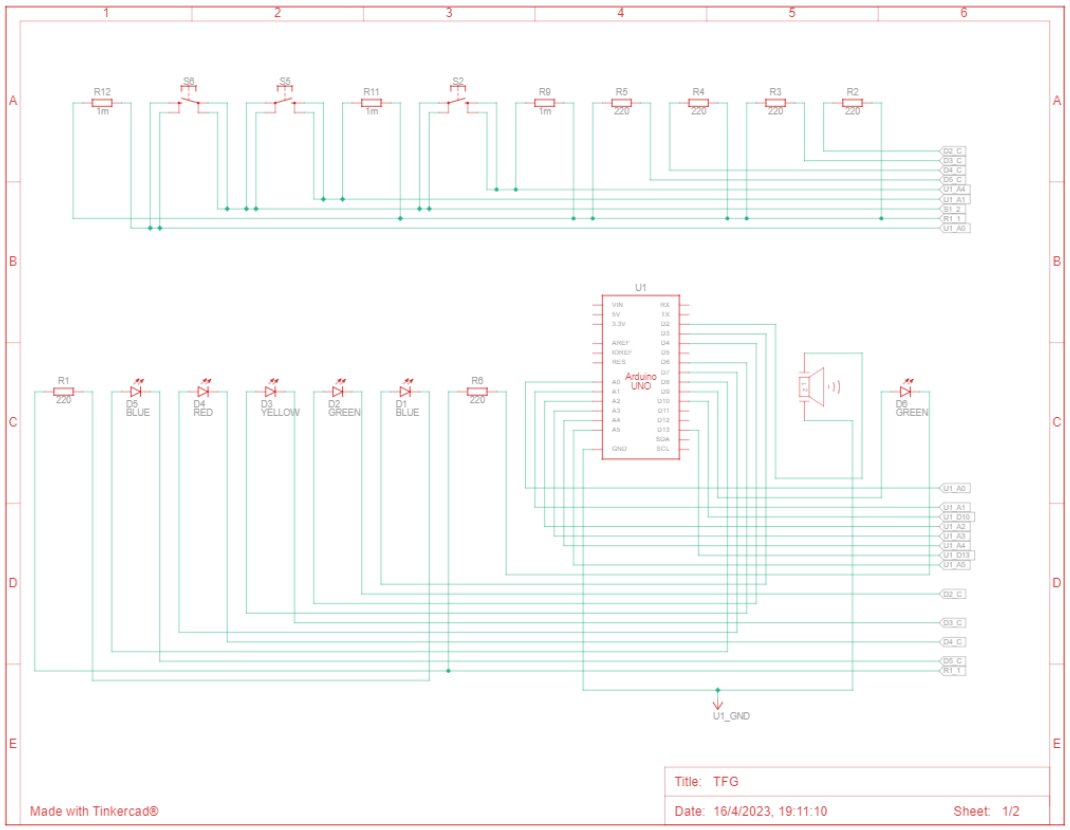
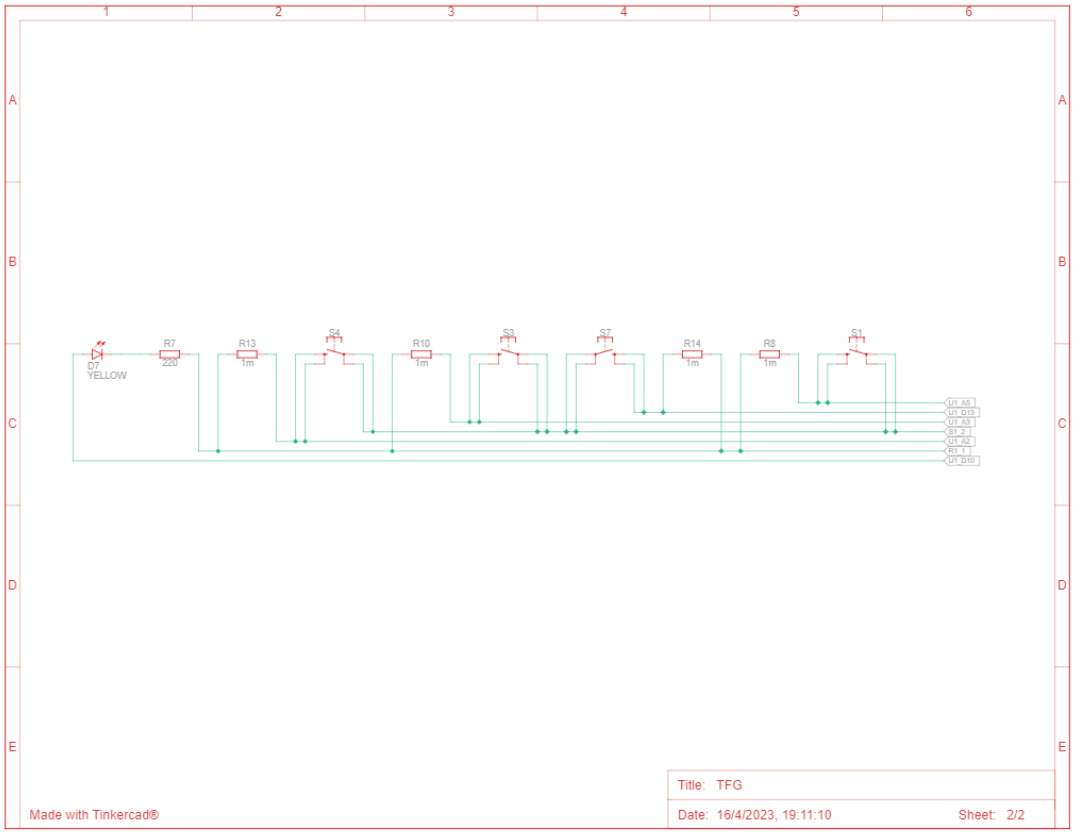


Diagrama:



Vista esquemática:





Lista de componentes:

Nombre	Cantidad	Componente
U1	1	Arduino Uno R3
S1 S2 S3 S4 S5 S6 S7	7	Pulsador
D1 D5	2	Azul LED
D2 D6	2	Verde LED
D3 D7	2	Amarillo LED
D4	1	Rojo LED
R1 R2 R3 R4 R5 R6 R7	7	220 Ω Resistencia
R8 R9 R10 R11 R12 R13 R14	7	1 mΩ Resistencia
PIEZO1	1	Piezo

2.2.2 Software:

Respecto al software utilizado para integrar el dispositivo con el ordenador mediante un puerto USB, es el programa proporcionado por Arduino "*Arduino IDE (2.0.1)".* Gracias a este programa podemos atrapar todas las señales lanzadas por el dispositivo cuando presionamos un botón, y así, transmitir dicha información al programa Processing mediante la conexión al puerto USB donde se encuentre el dispositivo conectado. Esta comunicación es bidireccional, de tal modo que también podemos lanzar desde Processing eventos que el programa de Arduino reconoce y a consecuencia envía voltaje a las luces led o al altavoz.

Todo esto es gracias a un bucle de reconocimiento de mensajes que, conforme Processing lee, línea por línea, el csv que contiene la partitura creada por el usuario, manda un mensaje con dicha nota al dispositivo, que en función de la nota a reproducir emite la frecuencia por el piezzo (altavoz) y enciende la luz led asociada a la tecla de la nota.

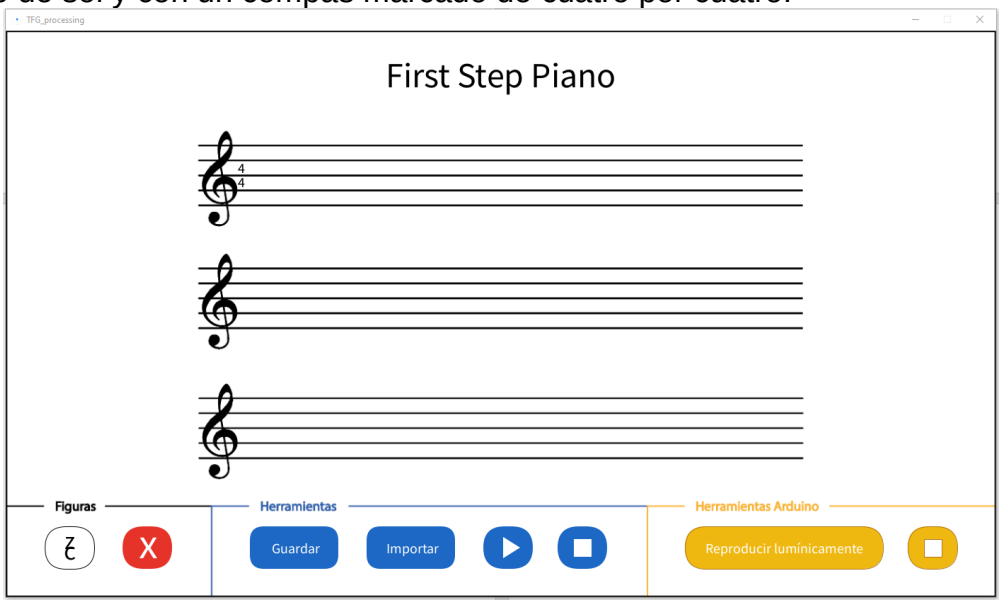
En el apartado "Anexos" de este mismo documento podremos leer con detalle el algoritmo que se ejecuta en Arduino para realizar estos procesos. A nivel funcional, en el siguiente apartado explicaremos con detalle las distintas casuísticas de uso que puede realizar un usuario con esta aplicación.

3. Resultados

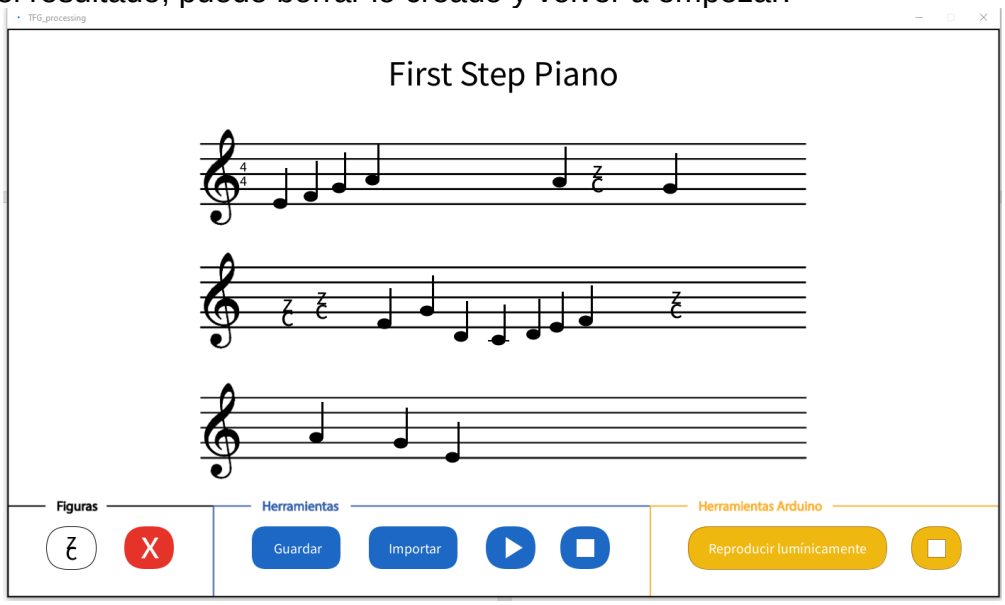
Una vez que la interfaz gráfica está iniciada y tenemos el dispositivo/piano conectado a un puerto USB de nuestro ordenador podemos realizar las siguientes acciones.

3.1 Creación de la partitura.

Al iniciar la aplicación, el usuario va a encontrar tres pentagramas vacíos en clave de sol y con un compás marcado de cuatro por cuatro.

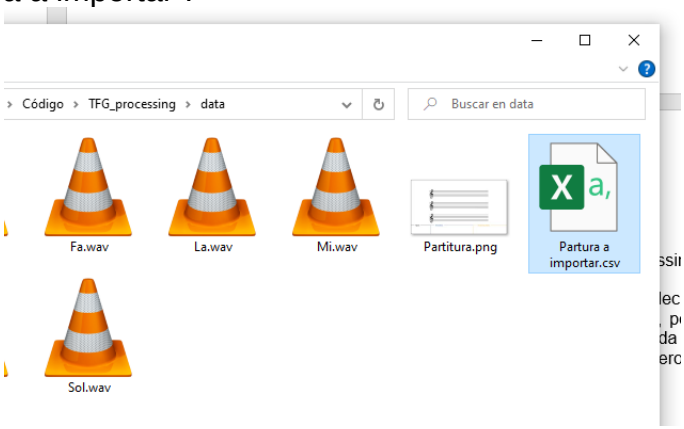


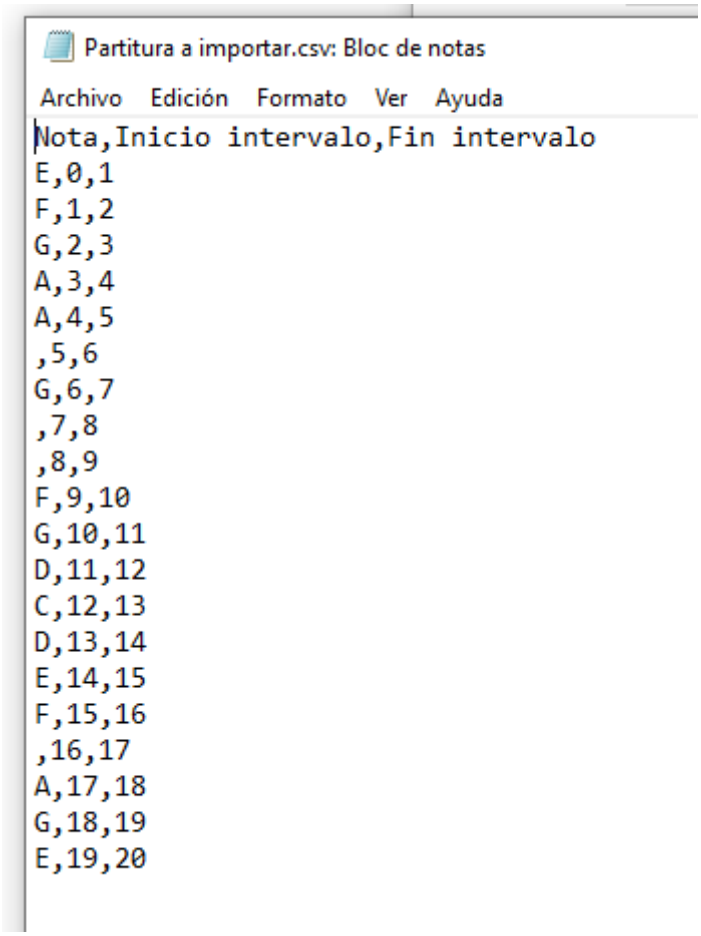
A partir de ahí, y teniendo en cuenta el menú de figuras situado en la esquina inferior izquierda, el usuario podrá rellenar dichos pentagramas a su antojo, utilizando figuras de negra y silencio de negra. Además, si no está conforme con el resultado, puede borrar lo creado y volver a empezar.



3.2 Exportación, importación, reproducción y stop en Processing

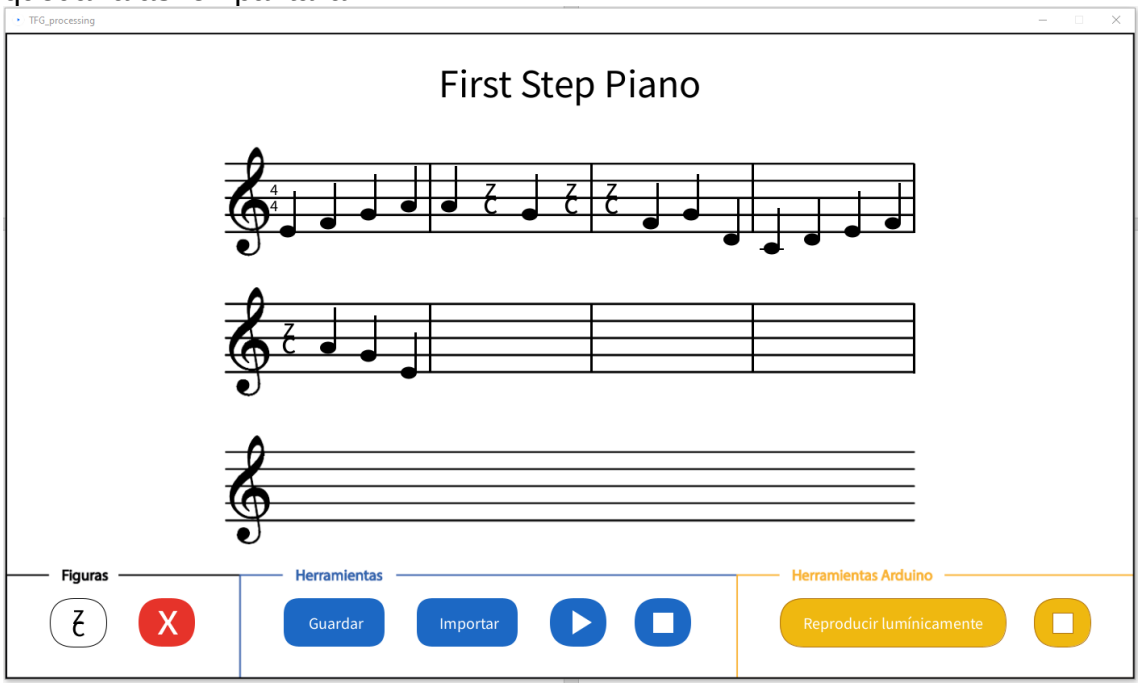
Siguiendo el ejemplo de las capturas anteriores, si el usuario decidiera que está satisfecho con el trabajo y no quiere perder la partitura, podría pulsar el botón de guardar y la aplicación detectaría la posición de cada figura y a qué notas corresponden, y guardaría dicha información en un fichero tipo CSV con el nombre “partitura a importar”.





Una vez guardada y exportada, el usuario dispondría de la partitura creada cuando desee y podrá importarla para seguir trabajando en ella o reproducirla.

Cuando el usuario hace clic en el botón importar la aplicación lee el archivo .csv, identifica la secuencia de notas e intervalos de tiempos que contiene y, además, escribe en la interfaz la partitura de forma ordenada. Al importar, quedaría así en pantalla.



Ahora el usuario podría seguir trabajando en ella añadiendo más figuras y guradando su trabajo, y además podría reproducirla tanto en el ordenador como en el dispositivo de Arduino.

3.3 Reproducción lumínica y sonora, y stop en Processing

De la misma forma, una vez que tenemos guardada o importada la partitura, el usuario podrá reproducir la melodía en el dispositivo creado con Arduino. Al hacer clic en “Reproducir lumínicamente”, la aplicación leerá las figuras pintadas en la pantalla, creará una tabla partitura con la información de nota e intervalo de ejecución, y nota por nota le hará saber a la placa de Arduino qué frecuencia debe reproducir y qué led tiene que iluminar.

3.4 Practicar la melodía en el piano.

Por último, ya sabemos que el usuario podrá reproducir sus melodías favoritas en el piano Arduino las veces que desee pero, además, podrá ir pulsando los botones a modo de teclas para practicar mientras se reproduce la melodía, y cuando no se esté reproduciendo, al pulsar las “teclas”, las frecuencias relacionadas con los sonidos de cada nota se irán reproduciendo como si de un piano eléctrico se tratase.

4. Conclusiones y trabajos futuros

Una vez terminada la aplicación en función de las metas y plazos planteados, tenemos como resultado una aplicación totalmente funcional que puede servir como herramienta de aprendizaje para cualquier usuario que se esté iniciando en el lenguaje musical.

Es necesario reiterar que, el desarrollo de esta aplicación nació con la idea de acercar la música y su aprendizaje a aquellos usuarios con capacidades funcionales reducidas, discapacidades visuales o auditivas, y con economías reducidas. En este aspecto, la aplicación dispone de una interfaz de creación de partituras sencilla y compatible con cualquier ratón o joystick que el usuario utilice, y además, permite reproducir la partitura de forma sonora o de forma lumínica, por lo que la flexibilidad y adaptación de la aplicación hacia el usuario está muy presente.

Este resultado es a causa de que el eje principal del desarrollo de esta aplicación siempre ha sido facilitar la interacción del usuario lo máximo posible, ofreciéndole flexibilidad de aprendizaje y variedad de uso. A consecuencia, tenemos ante nosotros una aplicación totalmente funcional con la que empezar a estudiar la música desde cero y con la flexibilidad de poder hacerlo a nuestro ritmo y necesidades.

Por otro lado, aunque la aplicación ya puede ser usada por cualquier usuario, siempre hay espacio para la mejora, y en este aspecto, estas son las propuestas de trabajo a futuro:

- Incorporar más figuras musicales a las opciones de creación.
- Posibilidad de cambiar la clave de la partitura y el compás.
- Añadir la posibilidad de crear varias páginas de pentagramas.
- Mejorar la interfaz de exportación e importación, dando la posibilidad de buscar el archivo en nuestro ordenador.
- Mejorar el dispositivo que reproduce las partituras de forma lumínica añadiéndole las “teclas” que le faltan o incluso creando dispositivos que referencien a otros instrumentos como una guitarra o un oboe.

5. Glosario

Arduino: es una compañía de desarrollo de software y hardware libres, así como una comunidad internacional que diseña y manufactura placas de desarrollo de hardware para construir dispositivos digitales y dispositivos interactivos que puedan detectar y controlar objetos del mundo real.

CSV: Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

Illustrator (AI): es un editor de gráficos vectoriales que sirve, entre otras cosas, para la ilustración como rama del arte digital, técnica o de diseño gráfico.

Joystick: es un periférico de entrada que consiste en una palanca que gira sobre una base e informa su ángulo o dirección al dispositivo que está controlando.

Negra: es una figura musical que equivale a ¼ del valor de la figura redonda.

Scrum: Metodología basada en dividir el proyecto en pequeños bloques (sprints) que se planifican y revisan continuamente.

Silencio de negra: en música un silencio o pausa es un signo que representa gráficamente la duración de una determinada pausa en una pieza musical.

Sprints: Periodos de tiempo determinados en función de las necesidades de un proyecto.

Piezo: Componente utilizado en el hardware de Arduino que mediante vibraciones que se expande en el aire es capaz de emitir frecuencias.

Processing: es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

USB: el Bus Universal en Serie (BUS) (en inglés: Universal Serial Bus), más conocido por la sigla USB, es un bus de comunicaciones que sigue un estándar que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.

6. Bibliografía

- Pegatinas para piano: https://www.amazon.es/Pegatinas-Branger-Principiantes-Transparentes-Extra%C3%ADbles/dp/B08535NYY1/ref=sr_1_19?adgrpid=73951625250&gclid=CjwKCAjw-rOaBhA9EiwAUkLV4sflk4GIHpwFf5nBFP650uqHC3LV07801wBcvck4OUioQ0NEbQVkgBoCPB4QAvD_BwE&hvadid=601346237962&hvdev=c&hvlocphy=9047043&hvnetw=g&hvqmt=e&hvrand=17725858914713492426&hvtargid=kwd-354615531563&hydadcr=27993_2290694&keywords=accesorios+para+piano&qid=1666031741&qu=eyJxc2MiOiIyLjg5IiwicXNhIjoiMi41MCIsInFzcCI6IjAuOTIifQ%3D%3D&sr=8-19

- Aplicación de aprendizaje: https://www.youtube.com/watch?v=haAPVubbfJU

- Piano con luces en las teclas: https://musicalsanfrancisco.es/teclado-aprendizaje-yamaha-ez-30/

- Las mejores aplicaciones para la educación musical infantil : https://www.educaciontrespuntocero.com/recursos/apps-musica-educacion-infantil/

- Las 15 mejores Apps para aprender a tocar el piano: https://latouchemusicale.com/es/15-apps-aprender-tocar-piano/#elementor-toc__heading-anchor-0

7. Anexos

Como una imagen vale más que mil palabras y un video vale por mil imágenes, adjunto un link en el que explico de forma práctica los desarrollos anteriormente comentados.

Link de presentación: https://youtu.be/yD0_48XGEnU

Código Processing

```
import processing.serial.*;                //Importamos la bilbioteca para mandar información al
dipositivo de Arduino
import ddf.minim.*;                        //Importamos la biblioteca para reproducir el audio más
tarde
Minim minim;                               // Esta variable nos servirá para reproducir la música
AudioPlayer player;                        // En esta variable cargará el sonido de Do

AudioPlayer playerDo;                      // En esta variable cargará el sonido de Do
AudioPlayer playerRe;                      // En esta variable cargará el sonido de Re
AudioPlayer playerMi;                      // En esta variable cargará el sonido de Mi
AudioPlayer playerFa;                      // En esta variable cargará el sonido de Fa
AudioPlayer playerSol;                     // En esta variable cargará el sonido de Sol
AudioPlayer playerLa;                      // En esta variable cargará el sonido de La
AudioPlayer playerSi;                      // En esta variable cargará el sonido de Si

Serial SerialPort;
Table xPartitura;
color ColorAzulMenu = color(28, 104, 196);
color ColorFondo = color(56,161,198);
color Arduino = color(239,184,16);
int i = 0;
Boolean PartituraIniciadaArduino;
Boolean PartituraIniciada;
float TiempoNota;
Float Intervalo;
int TiempoEjecucionAnterior;
PImage PartitureImage;
Table NotesInPartiture;
int[] NotasInPartiture_px = new int[200];
int[] NotasInPartiture_py = new int[200];
int[] NewArray = new int[200];
int[] NewArray2 = new int[200];
int NotesInPartitureCounter;
Boolean StartCreativeProcess;
int FirstPentagramaY = 160;
int SecondPentagramaY= 335;
int ThirdPentagramaY = 518;
int PentagramaY;
String actualNote;
int NotaImportadapx;
int NotaImportadapy;
String NotaActual;
Boolean PartituraImportada;
Boolean SilencioActivo;
int PentagramaActual;


///------Variables para controlar el tiempo y actuar a consecuencia en distintas partes del programa
Clock clock;
Timer timer;
```

```
void setup() {
  SerialPort = new Serial(this, "COM9", 9600);   // Configuramos el puerto por el que vamos a mandar
información a ARDUINO
  size(1400,800);                                // Configuramos el tamaño de la interfaz

  //FONDOS
  PartitureImage = loadImage("Partitura.png");
  PartitureImage.resize(1400,800);

  //TABLA QUE PINTAMOS EN PANTALLA
  NotesInPartiture = new Table();
  NotesInPartiture.addColumn("Nota", Table.STRING);
  NotesInPartiture.addColumn("Inicio intervalo", Table.INT);
  NotesInPartiture.addColumn("Fin intervalo", Table.INT);

  //TIEMPO
  clock = new Clock();
  timer = new Timer(0, 90000);                   // Establecemos 90 segundos máximo en el contador del
tiempo
  timer.start (true);

  // Mostramos en la consola el tamaño de la partitura inicial e inicializamos variables
  xPartitura = loadTable("Partitura a importar.csv", "header");
  println(xPartitura.getRowCount() + " total rows in table");
  println(xPartitura.getColumnCount() + " total rows in table");
  PartituraIniciadaArduino = false;
  StartCreativeProcess = false;
  PartituraImportada = false;
  SilencioActivo = false;
  PentagramaActual = FirstPentagramaY;

  ////Cargamos la música
  minim = new Minim(this);                    // Inicializamos la variable.
  playerDo = minim.loadFile("Do.wav");        // Cargamos el sonido Do
  playerRe = minim.loadFile("Re.wav");        // Cargamos el sonido Re
  playerMi = minim.loadFile("Mi.wav");        // Cargamos el sonido Mi
  playerFa = minim.loadFile("Fa.wav");        // Cargamos el sonido Fa
  playerSol = minim.loadFile("Sol.wav");      // Cargamos el sonido Sol
  playerLa = minim.loadFile("La.wav");        // Cargamos el sonido La
  playerSi = minim.loadFile("Si.wav");        // Cargamos el sonido Si

  PartituraIniciada = false;
  TiempoEjecucionAnterior=1;


}

void draw() {
  background(ColorFondo);
  image(PartitureImage,0,0);
  fill(0);
  noStroke();
  if (StartCreativeProcess){
    for(int n=0; n<NotesInPartitureCounter; n++){
      if (NotesInPartiture.getString(n,0).equals("")){         // Si la celda de la tabla es igual a
blanco pintamos un silencio
        textSize(35);
        text ("z", NotasInPartiture_px[n]+1,NotasInPartiture_py[n] );
        textSize(40);
        text ("c", NotasInPartiture_px[n]+1 ,NotasInPartiture_py[n]+20 );
      }else{                                                   // En caso contrario pintamos la figura
en pantalla
        ellipse(NotasInPartiture_px[n],float(NotasInPartiture_py[n]),20,15);
        rect(NotasInPartiture_px[n] +7,NotasInPartiture_py[n] - 50,3,50);
        if (NotesInPartiture.getString(n,0).equals("C") ){     // Si la nota es un DO añadimos una
barra al simbolo
          rect(NotasInPartiture_px[n]-15,NotasInPartiture_py[n] ,30,2);
        }
      }                                                        // Si se ha guardado la partitura
iniciamos la barra que
      if (PartituraImportada){                                 // señaliza que nota está reproduciendo
        if (NotesInPartitureCounter>2){
          rect(525,FirstPentagramaY+2,3,86);
          rect(725,FirstPentagramaY+2,3,86);
          rect(925,FirstPentagramaY+2,3,86);
          rect(1125,FirstPentagramaY+2,3,86);
        }
        if (NotesInPartitureCounter>16){
          rect(525,SecondPentagramaY,3,86);
          rect(725,SecondPentagramaY,3,86);
          rect(925,SecondPentagramaY,3,86);
          rect(1125,SecondPentagramaY,3,87);
        }
        if (NotesInPartitureCounter>32){
          rect(525,ThirdPentagramaY+1,3,86);
          rect(725,ThirdPentagramaY+1,3,86);
          rect(925,ThirdPentagramaY+1,3,86);
          rect(1120,ThirdPentagramaY+1,3,86);
          rect(1125,ThirdPentagramaY+1,5,86);
        }
      }
    }
  }
```

```
//// Configuramos los elementos de la interfaz
  //MENU
  textSize(50);
  fill(0);
  text ("First Step Piano", 700, 80);
  textSize (25);
  textAlign(CENTER);

  //Tiempo del compas (4x4)
  textSize (20);
  text ("4", 333, 221);
  text ("4", 333, 201);

  //Botón figuras negras/silencios
  fill(300,300,300);
  stroke(0,0,0);
  rect(55, 700, 70, 60, 200);
  fill(0,0,0);
  if (SilencioActivo){
  ellipse(90,742,15,10);
    rect(90 +6,742 - 30,1,30);
  } else{
    textSize (35);
    text ("z", 90, 732);
    textSize (37);
    text ("c", 90, 747);
  }
  noStroke();

  //Botón eliminar figuras
  fill(230,51,42);
  rect(165, 700, 70, 60, 200);
  fill(300,300,300);
  textSize (60);
  text ("x", 201, 745);

  //Exportar PARTITURA
  fill(ColorAzulMenu);
  rect(345, 700, 125, 60, 20);
  fill(300,300,300);
  textSize (20);
  text ("Guardar", 410, 738);

  //Importar PARTITURA
  fill(ColorAzulMenu);
  rect(510, 700, 125, 60, 20);
  fill(300,300,300);
  textSize (20);
  text ("Importar", 575, 738);


  //Iniciar PARTITURA  PROCESSING
  fill(ColorAzulMenu);
  rect(675, 700, 70, 60, 200);
  fill(300,300,300);
  triangle(703, 715, 727, 730, 703, 745);

  //Parar PARTITURA   PROCESSING
  fill(ColorAzulMenu);
  rect(780, 700, 70, 60, 200);
  fill(300,300,300);
  rect(803, 718, 25, 25, 0);

  //Iniciar PARTITURA  ARDUINO
  stroke(175,116,21);
  fill(Arduino);
  rect(960, 700, 280, 60, 200);
  fill(300,300,300);
  textSize (20);
  text ("Reproducir lumínicamente", 1100, 738);

  //Parar PARTITURA  ARDUINO
  fill(Arduino);
  rect(1275, 700, 70, 60, 200);
  fill(300,300,300);
  rect(1298, 718, 25, 25, 0);

  //Controlamos las reproducciones de las partituras
  clock.update();                                    // Actualizamos la clase clock
  timer.update(clock.getDeltaMillis());              // Controlamos el tiempo que ha pasado
desde que hemos iniciado el programa
```

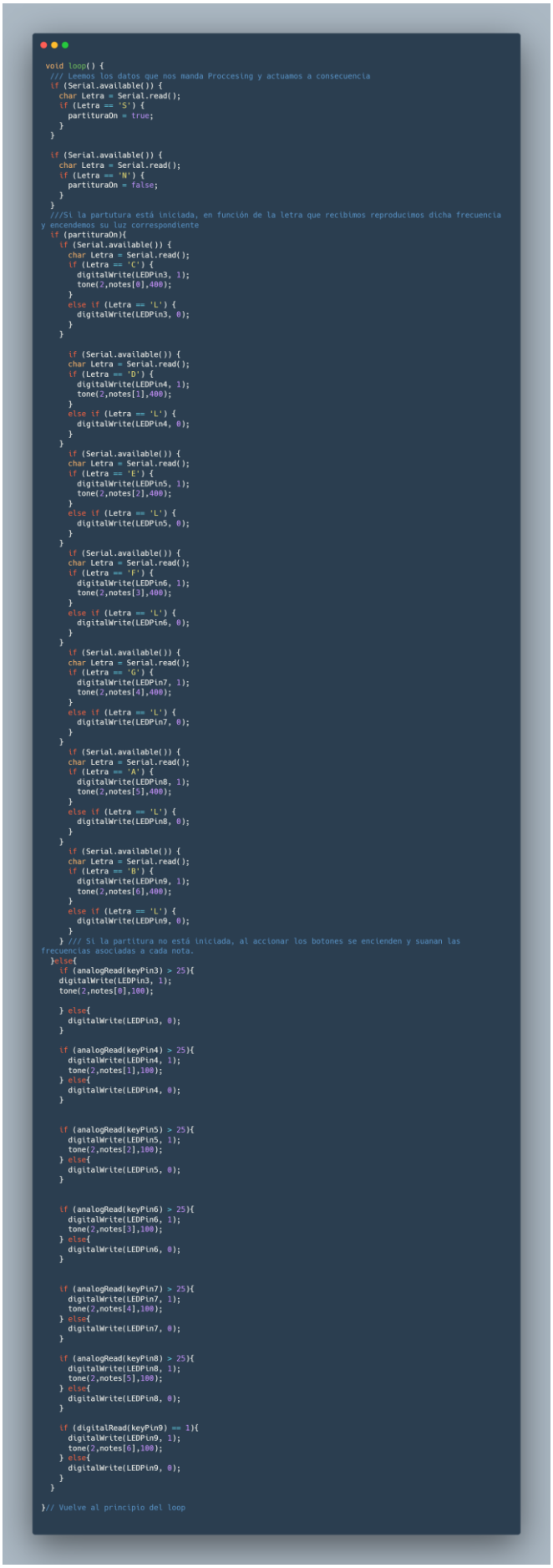
```
 /// Si el usuario ha hecho click en iniciar y el tiempo está dentro de la duración de la canción
  if ( PartituraIniciada && (int(timer.getTimeSec())< xPartitura.getRowCount())){
    if(PartituraImportada){
     //Marcamos la nota que se está reproduciendo
     fill(149,193,31);
     if (int(timer.getTimeSec())<=15){PentagramaActual = FirstPentagramaY;}
     if ((int(timer.getTimeSec())>=16)){PentagramaActual = SecondPentagramaY;}
     if (int(timer.getTimeSec())>=32){PentagramaActual = ThirdPentagramaY;}
       noStroke();
       rect(NotasInPartiture_px[int(timer.getTimeSec())],PentagramaActual-15,4,140);
       stroke(0,0,0);
     }
     // Configuramos el intervalo de transmisión de datos a ARDUINO
     Intervalo = xPartitura.getFloat(int(timer.getTimeSec()),2) -
xPartitura.getFloat(int(timer.getTimeSec()),1);
     if (Intervalo <= 1){
       TiempoNota = (Intervalo/2)-0.1;
     }else{
       TiempoNota = (Intervalo/2)-0.25;
     }

     if (TiempoEjecucionAnterior != int(timer.getTimeSec())){  // Si cambiamos de segundo paramos las
reproducciones y las rebobinamos


playerDo.pause();playerSi.pause();playerLa.pause();playerSol.pause();playerFa.pause();playerMi.pause();
playerRe.pause();

playerDo.rewind();playerSi.rewind();playerLa.rewind();playerSol.rewind();playerFa.rewind();playerMi.rew
ind();playerRe.rewind();

     }
     else{
       /// Si está está dentro del intervalo la reproducimos
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("C") ){playerDo.play();}
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("B") ){playerSi.play();}
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("A") ){playerLa.play();}
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("G") ){playerSol.play();}
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("F") ){playerFa.play();}
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("E") ){playerMi.play();}
       if (xPartitura.getString(int(timer.getTimeSec()),0).equals("D") ){playerRe.play();}
       println ("Nota : " + xPartitura.getString(int(timer.getTimeSec()),0));
     }

     TiempoEjecucionAnterior = int(timer.getTimeSec());

   }else {PartituraIniciada = false;}


  /// Si el usuario ha hecho click en iniciar en arduino y el tiempo está dentro de la duración de la
canción
  if ( PartituraIniciadaArduino && (int(timer.getTimeSec())< xPartitura.getRowCount())){
     if(PartituraImportada){
       //Marcamos la nota que se está reproduciendo
       fill(149,193,31);
       if (int(timer.getTimeSec())<=15){PentagramaActual = FirstPentagramaY;}
       if ((int(timer.getTimeSec())>=16)){PentagramaActual = SecondPentagramaY;}
       if (int(timer.getTimeSec())>=32){PentagramaActual = ThirdPentagramaY;}
       noStroke();
       rect(NotasInPartiture_px[int(timer.getTimeSec())],PentagramaActual-15,4,140);
       stroke(0,0,0);
     }
     // Configuramos el intervalo de transmisión de datos a ARDUINO
     Intervalo = xPartitura.getFloat(int(timer.getTimeSec()),2) -
xPartitura.getFloat(int(timer.getTimeSec()),1);
     if (Intervalo <= 1){
       TiempoNota = (Intervalo/2)-0.1;
     }else{
       TiempoNota = (Intervalo/2)-0.25;
     }

     if (((timer.getTimeSec() - int(timer.getTimeSec()))> TiempoNota)){          /// Si la nota no está
dentro del intervalo no la mandamos a ARDUINO
       SerialPort.write("L");
     }else {
       SerialPort.write(xPartitura.getString(int(timer.getTimeSec()),0));        /// Si está está
dentro del intervalo la mandamos a ARDUINO
       println ("Nota : " + xPartitura.getString(int(timer.getTimeSec()),0));
     }

   }else {
     SerialPort.write("N");                    // Si la partutura no está activa se lo hacemos saber a
ARDUINO
   }

   if (timer.getFinish()) {                    // Si han pasado 90 segundos cambiamos la pantalla al
MENU
     timer.reset();                            // Restablecemos el tiempo
     timer.start(false);                       // Paramos el Timer
     timer.update(clock.getDeltaMillis());     // Actualizamos el timer
     PartituraIniciadaArduino = false;         // Reestablecemos la variable que marca si la partitura
está en marcha
   }
}
```

```
 void mousePressed() {
  ///Click en Iniciar partitura arduino
  if ((mouseX > 960) & (mouseX < 1240) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(960, 700, 280, 60, 200);
    ////Controlamos el tiempo
    timer.reset();                              // Restablecemos el tiempo
    timer.start(true);                          // Empezamos a contar hasta los 5 segundos propuestos
    timer.update(clock.getDeltaMillis());       // Controlamos el tiempo que ha pasado desde que hemos
iniciado el programa

    SerialPort.write("S");                      // Hacemos saber a ARDUINO que la partitura está
iniciada
    PartituraIniciadaArduino = true;
  }

  ///Click en Parar partitura arduino
  if ((mouseX > 1275) & (mouseX < 1345) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(1275, 700, 70, 60, 200);
    timer.reset();                              // Restablecemos el tiempo
    timer.start(false);                         // Empezamos a contar hasta los 5 segundos propuestos
    timer.update(clock.getDeltaMillis());
    PartituraIniciadaArduino = false;
  }

  ///Click en Importar partitura
  if ((mouseX > 510) & (mouseX < 635) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(510, 700, 125, 60, 20);
    xPartitura = loadTable("Partitura a importar.csv", "header");         // Importamos la partitura
    println(xPartitura.getRowCount() + " total rows in table");
    println(xPartitura.getColumnCount() + " total Column in table");
    pintarNotasImportadas();                                              // Rellenamos la tabla que
pinta las figuras
    PartituraImportada = true;
  }

  ///Click en Exportar partitura
  if ((mouseX > 345) & (mouseX < 470) & (mouseY > 700) & (mouseY < 760)){

    fill(100,100,100);
    rect(345, 700, 125, 60, 20);
    saveTable(NotesInPartiture, "data/Partitura a importar.csv");         // Guardamos la tabla como
fichero csv
  }

    ///Click en Iniciar partitura
  if ((mouseX > 675) & (mouseX < 745) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(675, 700, 70, 60, 200);
    ////Controlamos el tiempo
    timer.reset();                                                        // Restablecemos el tiempo
    timer.start(true);                                                    // Empezamos a contar hasta
los 5 segundos propuestos
    timer.update(clock.getDeltaMillis());                                 // Controlamos el tiempo que
ha pasado desde que hemos iniciado el programa
    PartituraIniciada = true;

  }

  ///Click en Parar partitura
  if ((mouseX > 780) & (mouseX < 850) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(780, 700, 70, 60, 200);
    timer.reset();                                                        // Restablecemos el tiempo
    timer.start(false);                                                   // Empezamos a contar hasta
los 5 segundos propuestos
    timer.update(clock.getDeltaMillis());
    PartituraIniciada = false;

  }

  ///Click en figura Negra/Silencio
  if ((mouseX > 60) & (mouseX < 130) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(55, 700, 70, 60, 200);
    SilencioActivo = !SilencioActivo;
  }
    ///click en borrar figuras
  if ((mouseX > 165) & (mouseX < 235) & (mouseY > 700) & (mouseY < 760)){
    fill(100,100,100);
    rect(165, 700, 70, 60, 200);
    NotesInPartiture.clearRows();                                          // Limpiamos la tabla que
pinta las figuras
    NotesInPartitureCounter=0;
    NotasInPartiture_px = NewArray;
    NotasInPartiture_py = NewArray2;
    PartituraImportada = false;
    StartCreativeProcess = false;
  }

  // Detectamos si el usuario ha puesto una figura en el primer pentagrama
  if ((mouseX > 350) & (mouseX < 1100) & (mouseY > 160) & (mouseY < 265)){
    PentagramaY = FirstPentagramaY;
    getNote();
    insertValuesToTable();
  }

  // Detectamos si el usuario ha puesto una figura en el segundo pentagrama
  if ((mouseX > 350) & (mouseX < 1100) & (mouseY > 335) & (mouseY < 440)){
    PentagramaY = SecondPentagramaY;
    getNote();
    insertValuesToTable();
  }

    // Detectamos si el usuario ha puesto una figura en el tercer pentagrama
  if ((mouseX > 350) & (mouseX < 1100) & (mouseY > 518) & (mouseY < 623)){
    PentagramaY = ThirdPentagramaY;
    getNote();
    insertValuesToTable();
  }
}
```

```
 void getNote() {                // En función de la posición del click determinamos la nota a guardar

  println (PentagramaY);
  if (SilencioActivo){actualNote = "";println ("Silencio");
  }else{
    if ((mouseY > PentagramaY+101) & (mouseY < PentagramaY +150)){actualNote = "C";println ("DO");}
    if ((mouseY > PentagramaY+39) & (mouseY < PentagramaY +50)){actualNote = "B";println ("SI");}
    if ((mouseY > PentagramaY+50) & (mouseY < PentagramaY +60)){actualNote = "A";println ("LA");}
    if ((mouseY > PentagramaY+60) & (mouseY < PentagramaY +71)){actualNote = "G";println ("SOL");}
    if ((mouseY > PentagramaY+71) & (mouseY < PentagramaY +81)){actualNote = "F";println ("FA");}
    if ((mouseY > PentagramaY+81) & (mouseY < PentagramaY +91)){actualNote = "E";println ("MI");}
    if ((mouseY > PentagramaY+91) & (mouseY < PentagramaY +100)){actualNote = "D"; println ("RE");}
  }
}

void insertValuesToTable(){
    //Con cada clic dentro de un pentagrama añadimos una linea a la tabla
    NotesInPartiture.addRow();
    NotesInPartiture.setString(NotesInPartitureCounter,"Nota",actualNote);
    NotesInPartiture.setInt(NotesInPartitureCounter,"Inicio intervalo",NotesInPartitureCounter);
    NotesInPartiture.setInt(NotesInPartitureCounter,"Fin intervalo",NotesInPartitureCounter +1);

    NotasInPartiture_py[NotesInPartitureCounter]= mouseY;
    NotasInPartiture_px[NotesInPartitureCounter]= mouseX;
    NotesInPartitureCounter+=1;
    StartCreativeProcess = true;
}

void pintarNotasImportadas(){
    //Pintamos en pantalla la partitura importada
    NotesInPartiture.clearRows();
    NotesInPartitureCounter=0;
    NotaImportadapx = 350;
    NotesInPartiture = xPartitura;
    NotasInPartiture_px = NewArray;
    NotasInPartiture_py = NewArray2;
    for(int k=0; k<xPartitura.getRowCount(); k++){

      if ((k==16)||(k==32)){NotaImportadapx = 350;}

      NotaActual = xPartitura.getString(k,0);

      //En función de la psosición numérica determinamos un pentagrama y un espacio para pintar las
figuras
      if (k<=15){
        if (NotaActual.equals("C") ){NotasInPartiture_py[k]= FirstPentagramaY + 107;}
        if (NotaActual.equals("B") ){NotasInPartiture_py[k]= FirstPentagramaY + 45;}
        if (NotaActual.equals("A") ){NotasInPartiture_py[k]= FirstPentagramaY + 55;}
        if (NotaActual.equals("G") ){NotasInPartiture_py[k]= FirstPentagramaY + 65;}
        if (NotaActual.equals("F") ){NotasInPartiture_py[k]= FirstPentagramaY + 76;}
        if (NotaActual.equals("E") ){NotasInPartiture_py[k]= FirstPentagramaY + 86;}
        if (NotaActual.equals("D") ){NotasInPartiture_py[k]= FirstPentagramaY + 96;}
        if (NotaActual.equals("") ){NotasInPartiture_py[k]= FirstPentagramaY + 44;}
      }

      if ((k>=16)&&(k<=31)){
        if (NotaActual.equals("C") ){NotasInPartiture_py[k]= SecondPentagramaY + 107;}
        if (NotaActual.equals("B") ){NotasInPartiture_py[k]= SecondPentagramaY + 45;}
        if (NotaActual.equals("A") ){NotasInPartiture_py[k]= SecondPentagramaY + 55;}
        if (NotaActual.equals("G") ){NotasInPartiture_py[k]= SecondPentagramaY + 65;}
        if (NotaActual.equals("F") ){NotasInPartiture_py[k]= SecondPentagramaY + 76;}
        if (NotaActual.equals("E") ){NotasInPartiture_py[k]= SecondPentagramaY + 86;}
        if (NotaActual.equals("D") ){NotasInPartiture_py[k]= SecondPentagramaY + 96;}
        if (NotaActual.equals("") ){NotasInPartiture_py[k]= SecondPentagramaY + 42;}
      }

      if (k>=32){
        if (NotaActual.equals("C") ){NotasInPartiture_py[k]= ThirdPentagramaY + 107;}
        if (NotaActual.equals("B") ){NotasInPartiture_py[k]= ThirdPentagramaY + 45;}
        if (NotaActual.equals("A") ){NotasInPartiture_py[k]= ThirdPentagramaY + 55;}
        if (NotaActual.equals("G") ){NotasInPartiture_py[k]= ThirdPentagramaY + 65;}
        if (NotaActual.equals("F") ){NotasInPartiture_py[k]= ThirdPentagramaY + 76;}
        if (NotaActual.equals("E") ){NotasInPartiture_py[k]= ThirdPentagramaY + 86;}
        if (NotaActual.equals("D") ){NotasInPartiture_py[k]= ThirdPentagramaY + 96;}
        if (NotaActual.equals("") ){NotasInPartiture_py[k]= ThirdPentagramaY + 42;}
      }

      NotasInPartiture_px[k]= NotaImportadapx;
      NotaImportadapx +=50;
      NotesInPartitureCounter+=1;
    }
    StartCreativeProcess = true;
}
```

Código Arduino

```
 //Declaramos las varibales constantes
const int sensorPin = A0; // Asignamos al sensor de temperatura el pin A0 analógico
const float baselineTemp = 17.5; // Marcamos como temperatura base de la habitación 17.5
const int LEDPin3 = 3; // Asignamos el led verde RGB al pin 2 digital
const int LEDPin4 = 4; // Asignamos el led azul RGB al pin 3 digital
const int LEDPin5 = 5; // Asignamos el led rojo RGB al pin 4 digital
const int LEDPin6 = 6; // Asignamos el led verde RGB al pin 2 digital
const int LEDPin7 = 7; // Asignamos el led azul RGB al pin 3 digital
const int LEDPin8 = 8; // Asignamos el led rojo RGB al pin 4 digital
const int LEDPin9 = 9; // Asignamos el led rojo RGB al pin 4 digital

const int keyPin3 = A5; // Asignamos el led verde RGB al pin 2 digital
const int keyPin4 = A4; // Asignamos el led azul RGB al pin 3 digital
const int keyPin5 = A3; // Asignamos el led rojo RGB al pin 4 digital
const int keyPin6 = A2; // Asignamos el led verde RGB al pin 2 digital
const int keyPin7 = A1; // Asignamos el led azul RGB al pin 3 digital
const int keyPin8 = A0; // Asignamos el led rojo RGB al pin 4 digital
const int keyPin9 = 13; // Asignamos el led rojo RGB al pin 4 digital

float notes[]= {262,293,330,349,392,440,494}; // Asignamos las frecuencias de las notas
bool partituraOn; // Variable para saber si se está reproduciendo la partitura o no


void setup() {
  //Inicializamos las variables que se tendrán en cuenta en el programa
  Serial.begin(9600); // abrimos un puerto serie
  pinMode(LEDPin3, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(LEDPin4, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(LEDPin5, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(LEDPin6, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(LEDPin7, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(LEDPin8, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(LEDPin9, OUTPUT); // Señalamos que se trata de información de salida

  pinMode(keyPin3, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(keyPin4, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(keyPin5, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(keyPin6, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(keyPin7, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(keyPin8, OUTPUT); // Señalamos que se trata de información de salida
  pinMode(keyPin9, OUTPUT); // Señalamos que se trata de información de salida

  partituraOn = false;
}
```



```
void loop() {
  /// Leemos los datos que nos manda Proccesing y actuamos a consecuencia
  if (Serial.available()) {
    char Letra = Serial.read();
    if (Letra == 'S') {
      partituraOn = true;
    }
  }

  if (Serial.available()) {
    char Letra = Serial.read();
    if (Letra == 'N') {
      partituraOn = false;
    }
  }
  ///Si la partutura está iniciada, en función de la letra que recibimos reproducimos dicha frecuencia
y encendemos su luz correspondiente
  if (partituraOn){
    if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'C') {
        digitalWrite(LEDPin3, 1);
        tone(2,notes[0],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin3, 0);
      }
    }

      if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'D') {
        digitalWrite(LEDPin4, 1);
        tone(2,notes[1],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin4, 0);
      }
    }
      if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'E') {
        digitalWrite(LEDPin5, 1);
        tone(2,notes[2],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin5, 0);
      }
    }
      if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'F') {
        digitalWrite(LEDPin6, 1);
        tone(2,notes[3],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin6, 0);
      }
    }
      if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'G') {
        digitalWrite(LEDPin7, 1);
        tone(2,notes[4],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin7, 0);
      }
    }
      if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'A') {
        digitalWrite(LEDPin8, 1);
        tone(2,notes[5],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin8, 0);
      }
    }
      if (Serial.available()) {
      char Letra = Serial.read();
      if (Letra == 'B') {
        digitalWrite(LEDPin9, 1);
        tone(2,notes[6],400);
      }
      else if (Letra == 'L') {
        digitalWrite(LEDPin9, 0);
      }
    } /// Si la partitura no está iniciada, al accionar los botones se encienden y suanan las
frecuencias asociadas a cada nota.
  }else{
    if (analogRead(keyPin3) > 25){
    digitalWrite(LEDPin3, 1);
    tone(2,notes[0],100);

    } else{
      digitalWrite(LEDPin3, 0);
    }

    if (analogRead(keyPin4) > 25){
      digitalWrite(LEDPin4, 1);
      tone(2,notes[1],100);
    } else{
      digitalWrite(LEDPin4, 0);
    }


    if (analogRead(keyPin5) > 25){
      digitalWrite(LEDPin5, 1);
      tone(2,notes[2],100);
    } else{
      digitalWrite(LEDPin5, 0);
    }


    if (analogRead(keyPin6) > 25){
      digitalWrite(LEDPin6, 1);
      tone(2,notes[3],100);
    } else{
      digitalWrite(LEDPin6, 0);
    }


    if (analogRead(keyPin7) > 25){
      digitalWrite(LEDPin7, 1);
      tone(2,notes[4],100);
    } else{
      digitalWrite(LEDPin7, 0);
    }

    if (analogRead(keyPin8) > 25){
      digitalWrite(LEDPin8, 1);
      tone(2,notes[5],100);
    } else{
      digitalWrite(LEDPin8, 0);
    }

    if (digitalRead(keyPin9) == 1){
      digitalWrite(LEDPin9, 1);
      tone(2,notes[6],100);
    } else{
      digitalWrite(LEDPin9, 0);
    }
  }

}// Vuelve al principio del loop
```