Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Creación de un instrumento musical electrónico: Luthier en nuevas tecnologías

Creation of an electronic musical instrument: Luthier in new technologies

Alicia Elizabeta Marrero Ravelo

La Laguna, 13 de marzo de 2023

D. **José Gil Marichal Hernández**, con N.I.F. 78.677.406-H, profesor contratado doctor adscrito al área Teoría de la Señal y Comunicaciones, del Departamento de Ingeniería Industrial de la Universidad de La Laguna, como tutor

D. **Óscar Gómez Cárdenes**, con N.I.F. 78.859.209-Y, doctor en Informática por la Universidad de La Laguna, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

”*Creación de un instrumento musical electrónico: Luthier en nuevas tecnologías*”

ha sido realizada bajo nuestra dirección por Dª. **Alicia Elizabeta Marrero Ravelo**, con N.I.F. 79.084.256-K.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de marzo de 2023.

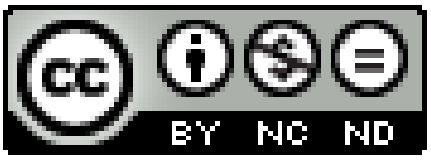
Agradecimientos

A la Universidad de La laguna por haberme provisto de una enseñanza de calidad que me ha permitido llegar hasta este punto.

A José Marichal y Óscar Gómez por haberme permitido participar en esta experiencia única; así como por su tiempo, ayuda y paciencia.

Finalmente a mi familia por apoyarme en el transcurso de mi vida académica.

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional.

Resumen

Los instrumentos musicales han avanzado en la historia de la humanidad. En este momento nos encontramos en un auge de las nuevas tecnologías. Los instrumentos y la música pueden tomar parte en este avance. Algunos ya aparecieron en el siglo pasado pero aún queda mucho camino por descubrir.

En este TFG se propone la tarea de crear un instrumento musical electrónico tomando las herramientas que se han provisto a lo largo del grado en ingeniería informática.

Se ha comenzado con una selección de componentes electrónicos que saciarán el número de entradas de datos necesarios. Posteriormente se seleccionó el microcontrolador a emplear, con la premisa de que fuera capaz de gestionar conexiones tipo Bluetooth.

Los resultados de este TFG son en resumen: .

Palabras clave: instrumento musical, electrónica, Arduino, MIDI, Bluetooth.

Abstract

Musical instruments have progressed in the history of humanity and in this moment we are in a boom of new technologies. Instruments and music can take part in this progress. Some already appeared in the last century but there is still a lot to discover.

In this TFG we propose the task of creating an electronic musical instrument using the knowledge that have been acquired throughout the degree in computer engineering.

It started with a selection of electronic components that would satisfy the number of data inputs required. Subsequently, the microcontroller to be used was selected, with the premise that it should be capable of managing Bluetooth type connections.

The results of this TFG are, in summary: .

Keywords: musical instrument, electronics, arduino, MIDI, Bluetooth.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Objetivos

El objetivo de este TFG es realizar, a través de un microcontrolador y las posibilidades del protocolo de comunicación MIDI, un controlador de MIDI capaz de enviar señales MIDI interpretables a otros dispositivos que trabajen con este protocolo. Se pretende mantener en todo momento una latencia baja en la comunicación para no entorpecer el carácter musical del proyecto. La principal característica de este controlador será emplear gestos con la mano que sean interpretables por el microcontrolador interno gracias a los sensores incorporados pudiendo obtener las notas deseadas. Luego estas notas podrán tener el timbre del instrumento deseado por el receptor. En cuanto a las dimensiones, estas serán reducidas permitiendo usarlo solo con una mano.

Para poder realizar este proyecto se va a trabajar en diferentes campos, todos afines al mismo objetivo. La realización del código de bajo nivel propio en el controlador, la instalación de circuitos y componentes hardware en el microcontrolador con el fin de obtener valores de entrada en el controlador y el diseño de la estructura contenedora del controlador, que se realizará mediante fabricación aditiva 3D.

1.2. Motivaciones

La motivación principal de este trabajo, es realizar un proyecto que desarrolle mi conocimiento sobre los microcontroladores y las amplias posibilidades que ofrecen, apoyado por mi curiosidad hacia el mundo de los instrumentos musicales electrónicos. A la vez que se cuenta con la motivación de realizar un proyecto que aún siendo parte del ámbito de trabajo de un ingeniero informático pueda emplearse para saciar necesidades más expresivas como lo es la música.

1.3. Instrumentos musicales electrónicos

Han existido una serie de instrumentos musicales electrónicos muy variados durante el siglo XIX y XX, como El Ondes-Martenot [1] o El Pianorad [2] , pero el que tomó más relevancia entre ellos sirviendo como referente a día de hoy es el Theremin [3]. Inventado en 1920 por Lev Termen, destaca por contar con dos antenas sobresalientes que controlan de forma independiente el tono y el volumen. Podemos verlo en la figura 1.1 o de forma más ilustrativa en el siguiente vídeo, `https://youtu.be/w5qf9O6c20o` donde además es posible escuchar el sonido que este instrumento produce. Aunque originalmente se planteó para representar musica clásica, el Theremin tuvo un papel importante en el mundo del cine y en la música de los 60s y los 70s hasta que surgió el sintetizador y la popularidad de este último lo reemplazó.

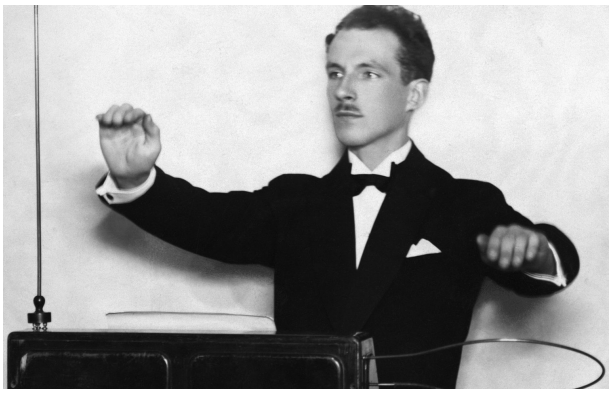


Figura 1.1: Se muestra a Lev Termen en una demostración del Theremin. A su vez se percibe en la imagen ambas antenas, una horizontal y otra vertical.

Los sintetizadores de sonido surgen como unos grandes y costosos sistemas modulares compuestos de osciladores, filtros, cintas, etcétera, que ni siquiera producían el sonido en tiempo real. Solo unos pocos estudios de sonido podían permitirse adquirir este tipo de equipos. A mediados de los años 60s Robert Moog construyó, uno de estos grandes sintetizadores, el que fue conocido como Sintetizador Moog [4]. Pero años después reinventó su sintetizador Moog dando lugar al nuevo MiniMoog. Un sintetizador más parecido a lo que conocemos hoy día. Este contaba con un tamaño más reducido y un carácter portátil como se aprecia en la figura siguiente 1.2. Se perdía con ello su modularidad previa pero su costo se reducía notablemente. Se perdían funcionalidades pero su acceso al publico y popularidad aumentó notablemente. Con él se produció la siguiente obra musical `https://youtu.be/OpxwOXFdV0E`.

Con el auge de sintetizadores, diferentes marcas se lanzaron a la carrera, cada una con su sintetizador y protocolo de comunicación propio. Pero la inmensa mayoría con algo en común, eran monofónicos y su sistema dependía de una combinación de módulos que alteraban el voltaje. Eran aparatos sin memoria ni sistema de procesamiento que obligaban al usuario rehacer el trabajo si alguno de los sonidos no era el deseado.

No se tardó mucho en solucionar esto pues a finales de los años 70 se incorporan a los sintetizadores lo que conocemos como CPU, capaces de procesar diferentes señales y manejar diferentes sonidos, generando por primera vez polifonía en los sintetizadores. Además se añadieron memorias que permitían almacenar grabaciones, sonidos o inclusive

Figura 1.2: A la izquierda una imagen de los primeros sintetizadores Moog. A la derecha una imagen del posterior sintetizador MiniMoog.

configuraciones. A esto último lo conocemos a día de hoy como “PRESETS” . En cuanto a almacenar sonidos, resultó ser clave para el surgimiento de algunos instrumentos como el Sampler[5] o el Mellotrón[6]. Un Sample es un instrumento musical que originalmente fue electrónico, este no generaba sonido propio sino que funcionaba mediante grabaciones de sonidos cargadas en él, posteriormente estas grabaciones se reproducirían para interpretar música. Por otro lado el Mellotrón es un instrumento electromecánico que cuenta en su interior con una serie de cintas magnéticas y lectores asociados. El usuario al presionarlas teclas hace variar el contacto entre los lectores y las cintas para generar el sonido.

Pese a ser más complejos, los sintetizadores seguían produciendo sonidos artificiales, que no podían competir con la naturalidad de los instrumentos convencionales, esto no resultó en un impedimento para los artistas que lograban explotar las ventajas de los sintetizadores. Algunos como Karlzheinz Stockhausen o Juan Amenábar dieron nombre a estilos musicales con caracteres propios y nunca antes explorados como la música concreta o la música electroacústica.

Figura 1.3: Yamaha DX7 el primer sintetizador totalmente digital con un sistema de síntesis de frecuencia.

La pieza clave en el avance de los sintetizadores vino con la síntesis de frecuencia o síntesis FM [7]. Esta forma de trabajar, basada en diferentes operadores sinusoidales que interactuaban entre sí, formaban el equivalente a los tradicionales módulos de los sintetizadores antiguos. Así se dejó a un lado la metodología tradicional y se apostó por un sistema totalmente digital.

El sintetizador pionero en este campo fue el DX7 de la marca Yamaha, el primer sintetizador totalmente digital [8].

1.4. MIDI

El protocolo MIDI [9] (Musical Instruments Digital Interface) surgió oficialmente en 1982. Este protocolo permitió el envío de información musical entre instrumentos, ordenadores y otros dispositivos, lo cual favoreció y facilitó notablemente la producción musical. Inicialmente la única forma de conectar un dispositivo MIDI era mediante un conector de 5 pines denominado DIN 180. Aunque han existido otras formas de enviar la información del protocolo como USB, Bluetooth, Ethernet, o recientemente Wi-Fi. Dada su popularidad, se ha ido ajustando el mismo para adaptarlo a los avances tanto de dispositivos como de medios de transmisión.

Figura 1.4: Cable DIN 180, empleado en conexiones cableadas de dispositivos MIDI.

Cuando se emplea un instrumento MIDI, cada vez que se pulsa una tecla se crea un evento o nota MIDI. Los eventos MIDI incluyen instrucciones que determinan parámetros como la velocidad de la nota, el volumen, si la nota se mantiene a lo largo del tiempo y muchos otros.

MIDI empezó a ser una parte importante de muchos sintetizadores de su época, y a día de hoy sigue siendo un protocolo que se encuentra presente en la mayoría de producciones musicales aunque asociados a controladores de todo tipo. Algunos de estos controladores tienen integrada la producción de sonido mientras que otros requieren de instrumentos virtuales que interpreten los mensajes en formato MIDI.

Figura 1.5: A la izquierda tenemos un dispositivo controlador MIDI, el cual requiere de un ordenador o sintetizador para la producción de sonido. A la derecha tenemos un sintetizador que es capaz de producir sonido a partir de la información enviada por el controlador.

Los mensajes MIDI [10] se componen de un primer byte de estado, que determina si se trata de un mensaje de canal o de un mensaje de modo, y de uno o dos bytes restantes dependiendo del tipo de mensaje. Todos los mensajes se envían y reciben de forma secuencial por lo que se debe leer primero un mensaje para continuar con el siguiente. La estructura en la inmensa mayoría de los casos es la que se muestra en la figura 1.6.

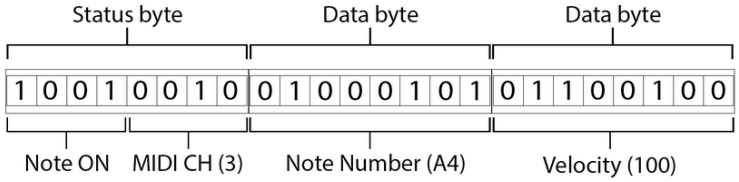


Figura 1.6: Trama de un mensaje del protocolo MIDI.

1.4.1. Mensajes de canal

Los mensajes de canal, implican la gran mayoría de mensajes MIDI. Estos se transmiten a través de canales concretos, de ahí su nombre. MIDI permite trabajar con hasta $2^4 = 16$ canales. Los mensajes de canal afectan solamente a los instrumentos que leen la información en dicho canal. Dentro de esta categoría de mensajes de canal se encuentra varias subcategorías:

- Note On : Este mensaje le indica al dispositivo, que debe iniciar una nota.
- Note Off : Funciona de manera opuesta a la anterior.
- Polyphonic Aftertouch : Envía una nota que cambia el nivel sonoro y el timbre.
- Channel Aftertouch : Similar al anterior pero no sobre la nota sino sobre el canal.
- Pitch Bend : Genera una desafinación a la nota generalmente entre uno o dos semitonos.

- Program Change : Cambia el instrumento o efecto sonoro. Hay 128 posibilidades.
- Control Change : Este mensaje varía el volumen, la reverberación, la modulación, etc. Es la categoría más amplia de mensajes de canal. Tiene 128 posibilidades de cambios en el sonido.

1.4.2. Mensajes de modo o sistema

Este tipo de mensajes comienzan su trama con la secuencia "`1111`". Este mensaje no afecta a ningún canal concreto sino que cuenta con un efecto global. Existen de 16 mensajes de esta categoría y aunque no es necesario profundizar en todos ellos podemos sacar en claro cual es la funcionalidad de estos mensajes.

- Mensajes comunes del sistema: Este mensaje permite posicionar a un dispositivo en determinado fragmento de una determinada pieza. Se emplea con fines de sincronización para momentos concretos.
- Mensajes de tiempo real: Este tipo de mensajes se emplea para coordinar y sincronizar dispositivos. En estos mensajes se establece maestros y esclavos donde es el maestro quien determina los tempos para el resto de dispositivos.
- Mensajes exclusivos del sistema. Son mensajes para aquellos dispositivos MIDI con características peculiares que difícilmente se pueden incluir en el estándar. En la trama de este tipo de mensajes se inicia con un byte "`Ini.SysEx`" (Inicio de sistema exclusivo) y concluye con un byte de estado "`EOX`" (fin de exclusividad). Entre estos dos bytes cada fabricante tiene la libertad de incluir sus características propias.

1.5. Microcontroladores

Un microcontrolador [11] es un circuito programable con varios módulos incluidos, entre ellos los que forman parte de una computadora. El numero y tipo de módulos dependerá del microcontrolador.

El primer microcontrolador, el TMS1000 [12], fue lanzado en 1974 por los ingenieros de Texas Instruments. Este, a diferencia de algunos intentos anteriores de otras compañías, era capaz de funcionar sin la necesidad de circuitos adicionales. Actualmente los microcontroladores incluyen los módulos básicos para su funcionamiento además de varios sistemas que facilitan su uso y dan muchas posibilidades de trabajo como módulos Wi-Fi o Bluetooth. Todo ello de manera muy accesible y a un coste reducido.

Capítulo 2

Estado del arte

2.1. Estado del arte

En 2020 The MIDI Association, mostró oficialmente una actualización del protocolo el cual había permanecido apenas sin cambios desde 1983. Esta versión recibió el nombre de MIDI 2.0 [13]. La principal diferencia entre ambas versiones del protocolo es la capacidad de esta nueva versión de establecer una comunicación bidireccional, lo cual posibilita el intercambio de información funcional entre dispositivos. La versión 1.0 o versión base no ha sido desechada, todo lo contrario, se integró la compatibilidad con esta versión en la actualización de 2020 para así mantener el *environment* que se tenía hasta el momento. Como los autores dicen en su pagina oficial, se continua a día de hoy trabajando en nuevas funcionalidades y mejoras que se integran en la versión 2.0 .

Este cambio de versión favoreció la aparición y actualización de muchas librerías que facilitan las tareas a la hora de trabajar con MIDI y es posible encontrar pequeños proyectos y foros de discusión activos en sitios web como GitHub [14] o la propia página de MIDI Association [9]

En lo referente a los microcontroladores, actualmente en el mercado existe una amplia diversidad de dispositivos, muchos de ellos a un coste asequible, por lo que hay una gran accesibilidad a este recurso.

Además los microcontroladores modernos normalmente integran módulos de comunicación Wi-Fi o Bluetooth que facilita la comunicación y reduce el número de conexiones físicas, ambas, cosas que nos interesan para este proyecto.

En lineas generales la mayoría de microcontroladores se pueden programar con un IDE de software libre. Lo más popular actualmente es la solución provista por Arduino, *Arduino IDE* [15]. Este *IDE*, recibe bastante actualizaciones mensuales, y cuenta con librerías para adaptarse a las diferentes marcas de microcontroladores y a los diferentes componentes que se puedan integrar, proporcionando una interfaz más amigable. Aún así hay otras alternativas a Arduino IDE, pero la más destacable entre ellas es PlatformIO. Este es un IDE de código abierto conocido por permitir cosas como desarrollo remoto o integración con repositorios.

Por otro lado, existen en la actualidad proyectos similares, pues son varias las personas que han empleado la tecnología para darle un uso comparable al objetivo del proyecto. Entre ellos se puede recalcar The Glide [16] el instrumento desarrollado por Keith Groover, que fue presentado 2019 en una TED Talk. Este instrumento fue pensado originalmente como un controlador MIDI diseñado para transmitir mensajes por cable y Bluetooth.

Capítulo 3

Metodología de trabajo

La metodología de trabajo empleada para este TFG ha sido principalmente el desarrollo iterativo o desarrollo incremental. En cada reunión con los tutores, se evalúan los resultados de las iteraciones previas con el fin de recibir *feedback* y mantener en todo momento la línea original del proyecto. De forma externa a las reuniones, mediante el servicio Telegram, se notificaba a los tutores de los más notorios resultados y de las líneas a seguir para futuras iteraciones.

En una de las primeras reuniones, donde se concretaron los objetivos del proyecto, se realizó un esquema 3.1 para hacer una abstracción de los temas a abordar y las lineas de trabajo que se debían seguir para lograr su correcta realización.

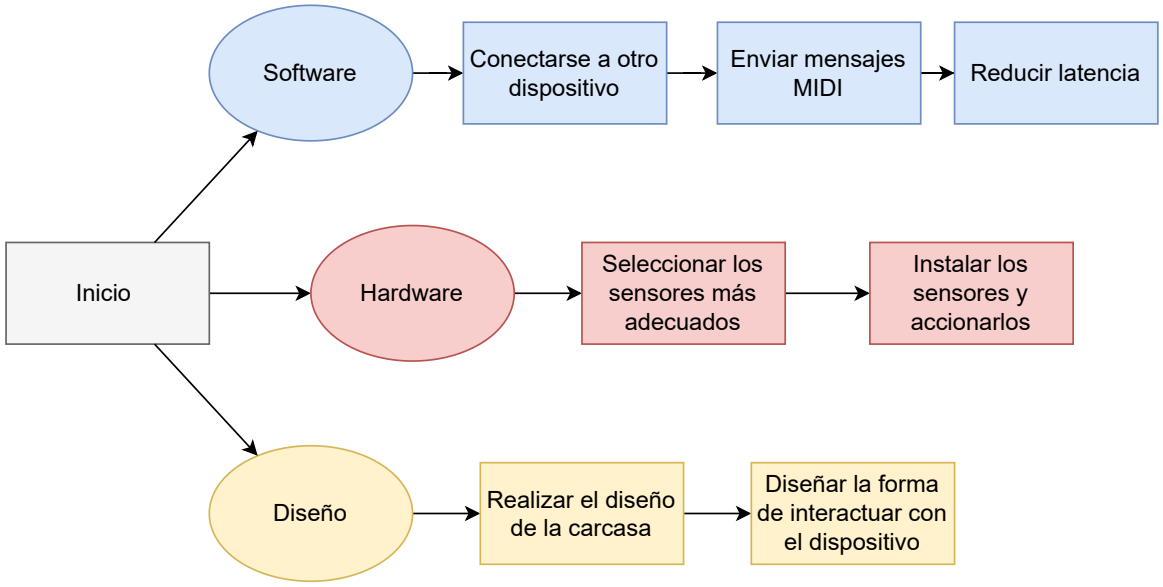


Figura 3.1: Diagrama que muestra las 3 lineas principales de este proyecto y los objetivos que forman parte del camino crítico

En gran medida, el software producido para el microcontrolador, ha sido desarrollado en lenguaje *C++*[17]. Al realizarse en Arduino IDE se mantuvo la extensión de archivos como "*.ino*". Esta es la extensión en la que trabajan muchos microcontroladores entre ellos la Wemos D1 R32. El esquema del cableado se ha realizado con *Fritzing*[18]. El diseño de la carcasa donde se aloja el microcontrolador se ha realizado con *Fusion360*[19]

y se ha procesado para su posterior impresión con *Cura*[20]. La memoria se ha redactado en LaTeX [21], usando el entorno Overleaf [22].

(a) Logotipo del lenguaje de programación C++.

(b) Logotipo de Fritzing.

(c) Logotipo del entorno de desarrollo Arduino

(d) Logotipo del software Fusion360.

(e) Logotipo del software Cura

(f) Logotipo del sistema de composición de textos LaTeX.

(g) Logotipo del entorno Overleaf.

Figura 3.2: Logotipos de algunas de las herramientas utilizadas.

Capítulo 4

Diseño de la interfaz

Una parte crucial antes de proseguir es la realización un buen diseño para el manejo del instrumento con el objetivo de que el usuario pueda fácilmente comprender como usarlo a un nivel básico. Teniendo esto claro se podrán seleccionar los componentes hardware necesarios y elaborar los primeros prototipos de software a un alto nivel. Este apartado tiene una gran libertad creativa pero determinará los siguientes capítulos del proyecto.

Las características que se tuvieron en cuenta a lo largo del proceso de diseño son los siguientes:

- Mantener un factor artístico.
- Reducir los factores que limiten la operatividad.
- Reducir la latencia.

4.1. Mantener un factor artístico

Dado que este proyecto se enfoca en un ámbito musical, se ha priorizado un factor artístico a la hora de diseñar la interacción del usuario con el instrumento MIDI. Se plantea inicialmente que el usuario pueda, mediante gestos con las manos, alterar las variables de control de los mensajes MIDI, por lo que podrá enviar una gran variedad de mensajes limitando el contacto con controlador. La inspiración de este planteamiento recae en el Theremin, donde la nota se genera por interferencias en los campos generados por el instrumento.

4.2. Reducir los factores que limiten la operatividad

Manteniendo presente el apartado anterior parece evidente que se debe mantener reducida la cantidad de elementos que puedan ser conflictivas con un manejo artístico del

controlador. Es por ello que se ha optado por realizar conexiones inalámbricas. La que ha resultado más apropiada para este proyecto es una conexión Bluetooth. A la hora de pensar en el formato final que tendrá el controlador se ha tenido en cuenta que, separar los sensores en múltiples conjuntos separados entre si, podría aumentar el numero de cableado entre dispositivos interponiéndose en la comodidad del usuario.

4.3. Reducir la latencia

Ante este apartado es importante recalcar el término latencia, pues resulta clave para el desarrollo del proyecto. Se entiende por latencia, cualquier retardo en una acción. La latencia es un termino empleado con frecuencia en el mundo de la informática y la tecnología en general. Pero en cuanto a los instrumentos musicales, estos cuentan con una valor de latencia en el que dejan de ser interpretables. Esta latencia límite está en torno a los 30 ms, dependiendo del tipo de instrumento. Este es el apartado más crucial de los tres. Ya que tendremos que tenerlo presente en todo momento a lo largo del desarrollo. Si la latencia llegase a ser más elevada de los 30 ms [23] se perdería directamente el uso como controlador MIDI.

Como se puede ver en la figura 4.1, se cuenta con tres puntos claves en donde afecta la latencia. Estos son la lectura de datos, el procesamiento de datos y el envío de datos. La lectura de datos en sí misma supone un leve aumento de latencia, pero ésta se puede ver aumentada por el número de dispositivos conectados al puerto I2C. En este caso concreto esta variación en la latencia no resulta tan preocupante como las que nos encontramos a continuación. La siguiente es el aumento de la latencia proveniente del procesamiento de datos en el microcontrolador. Los milisegundos que se empleen en software pueden ser críticos, ya que en caso de no estar correctamente optimizado podemos aumentar varios ms. Por esto se debe emplear un lenguaje de programación de bajo nivel. Teniendo claro la forma de actuar en el procesamiento de datos debemos pasar a lo referente al envío. El envío de datos generará un aumento problemático de latencia, se había comentado anteriormente que se realizaría una comunicación inalámbrica, y aunque en comparación con las comunicaciones por cable, el Bluetooth resulte algo más lento, se sigue considerando idóneo para esta situación. Este es el apartado donde más latencia se genera y al no poder reducirla se debe procurar mantenerla baja en los dos primeros puntos.

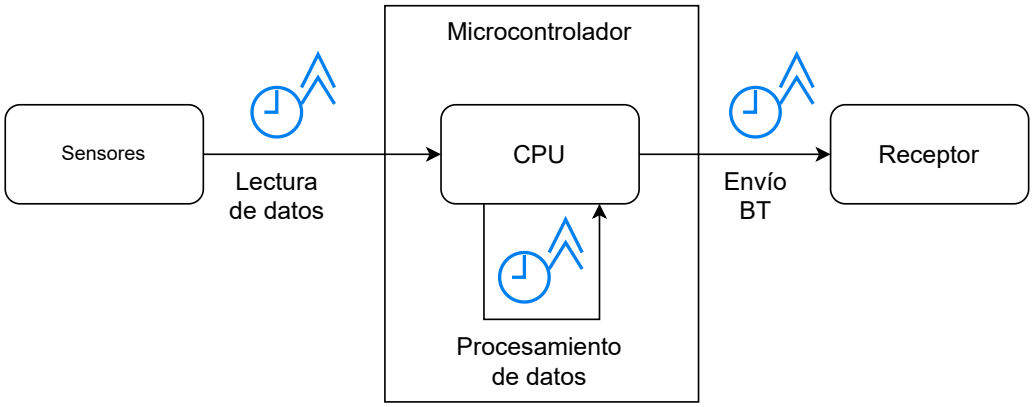


Figura 4.1: Diagrama que muestra los 3 puntos principales donde podemos encontrar un aumento significativo de la latencia.

4.4. Entradas necesarias

Es clave en el diseño saber cuáles son los datos imprescindibles a manipular. En el caso de mensajes MIDI se requiere de una comprensión de los conceptos musicales básicos. Recurriendo a libros de Teoría musical, podemos hallar una primera idea de los factores a recabar [24]. Por ejemplo, el tono, la frecuencia, la intensidad sonora, la duración o el timbre son los esenciales para concretar una nota. Aunque para darle forma a un sonido se requiere de diferentes efectos propios de los mismos instrumentos, como el pedal de un piano, o de un adecuado uso del silencio para lograr un sonido más natural. Hay muchos factores a tener en cuenta y MIDI ofrece un amplio abanico de efectos sonoros y posibilidades para ellos pero como en los controladores habituales, que estos efectos estén disponibles no implica que se deban emplear todos y cada uno de ellos.

Se concluye pues que requerimos de las seis siguientes entradas para las que habrá que habilitar algún sensor hardware:

- Frecuencia: En el lenguaje musical que se emplea en la mayoría de la música occidental actual los sonidos se clasifican mediante escalas. La escala 12 TET 4.2 [25] es la más empleada y por ello la que emplearemos en el proyecto. Consiste en dividir el rango de frecuencias de manera logarítmica en octavas, cada una de estas octavas se componen de 12 partes con las mismas proporciones. Estas partes, con sus respectivas frecuencias, tienen los nombres : Do, Re, Mi, Fa, Sol, La, Si; que junto a los accidentales de la tonalidad (símbolos ♯ y ♭) obtenemos los 12 tonos.
- Tono: Es el espacio que hay entre las diferentes frecuencias y nos permite diferenciar sonidos graves y agudos.
- Intensidad sonora: La intensidad sonora es más comúnmente conocida como volumen.
- Timbre: Es el sonido característico producido por un instrumento, el timbre es la cualidad que nos permite diferenciar por ejemplo, un tambor de un saxofón. MIDI nos da la oportunidad de generar el timbre en el receptor, una vez enviados los otros parámetros.

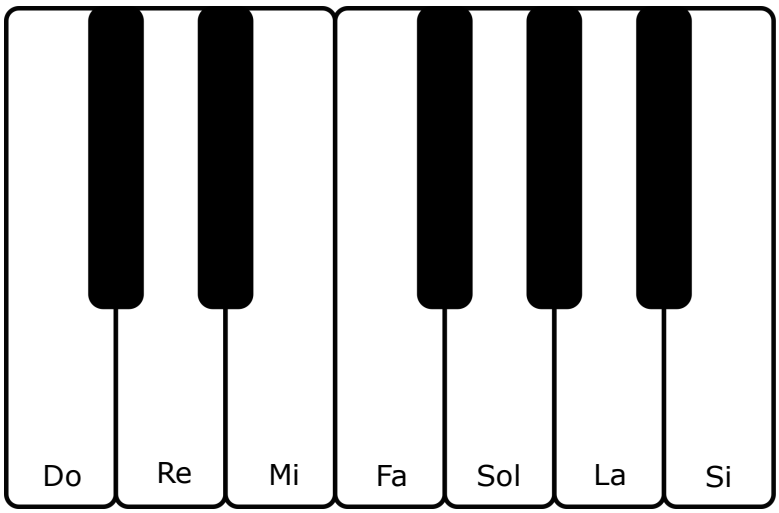


Figura 4.2: Esta imagen muestra sobre la disposición de un piano los 12 sonidos de una escala 12TET. Siendo las teclas negras del piano los accidentales de cada tonalidad. La cual abarca desde un Do hasta un Si.

- Duración: Es el tiempo en el que la nota se prolonga. En teoría musical estaríamos hablando por ejemplo de figuras como fusas o blancas.
- Accionador: Es una de las entradas más importantes que podemos tener, ya que determina cuando se inicia una nota.

Capítulo 5

Hardware

El apartado hardware es crucial para este proyecto, su comprensión y manejo es clave para poder gestionar los datos de entrada provenientes de los sensores y manipularlos de la manera más eficiente y rápida posible, pues recordemos que en caso contrario sufriríamos de un aumento de latencia.

Partimos a continuación con un listado de los elementos hardware que se han empleado además de una descripción de los mismos para su correcta comprensión.

5.1. Wemos D1 R32

Comenzaremos por el microcontrolador que se ha empleado es la Wemos D1 R32 5.1, y es el corazón del hardware ya que su elección repercute en las conexiones hardware. Inclusive influye en algunos detalles del apartado software, que veremos posteriormente, ya que al emplear un bajo nivel en lenguaje de programación, debemos conocer las especificaciones del microcontrolador.

Los detalles del microcontrolador se pueden encontrar en su ficha técnica [26]. Algunos de los datos más destacables de éste son: La intensidad del voltaje de sus pines, tanto para las entradas como para las salidas, esta se maneja con una intensidad de 3.3 V. Cuenta con 512 KB de memoria RAM. Sus pines digitales suman un total de 20 y en su mayoría se encuentran en la parte superior, mientras que los analógicos solo son 6 pines y se encuentran en la parte inferior de este. Y una interfaz para comunicaciones I2C [27]. Estos datos son importantes porque afectaran al tipo de conexiones cableadas que se deben hacer. Pero si debemos recalcar alguna característica del microcontrolador, es que cuenta de forma nativa con conexión Bluetooth. Éste es el apartado que más se ha tenido en cuenta a la hora de seleccionar este dispositivo pues como se comentó en el capitulo 4, buscamos una conexión inalámbrica y tenerla integrada supone un ahorro de costo, tiempo y especialmente de entradas y salidas, pues un módulo externo requeriría de cableado adicional.

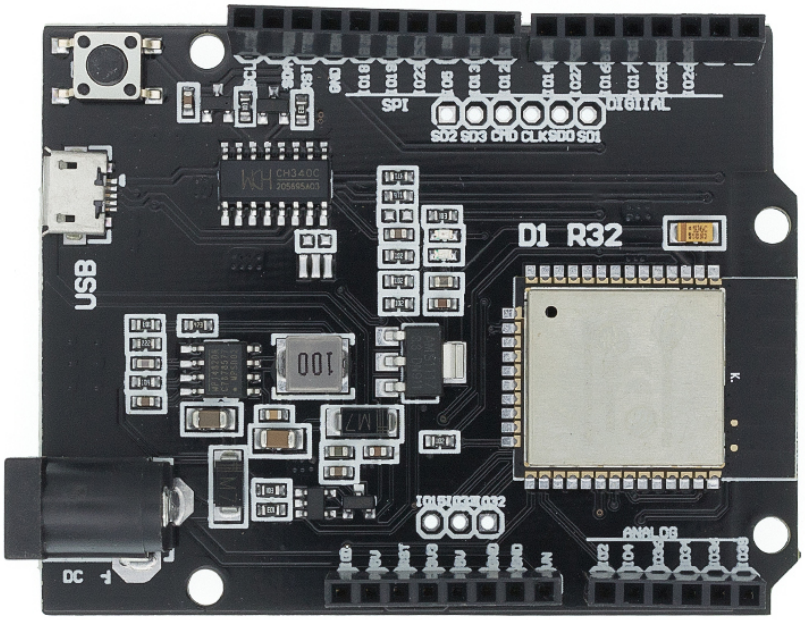


Figura 5.1: Ésta es la apariencia de la WEMOS D1 R32, el microcontrolador empleado en el desarrollo del TFG.

5.1.1. Preparación del microcontrolador

En base a la experiencia obtenida al emplear el microcontrolador a lo largo del TFG, aparecen las siguientes anotaciones que se deberían tener en cuenta para poder trabajar con el microcontrolador sin percances. La primera de estas es la instalación del *driver* CH340 [28], encargado de convertir los datos transportados por USB a datos *serial*. Este *driver* nos permitirá realizar comunicaciones entre nuestro ordenador y el microcontrolador, además de ello, permitirá que software como Arduino-IDE reconozcan el dispositivo como microcontrolador.

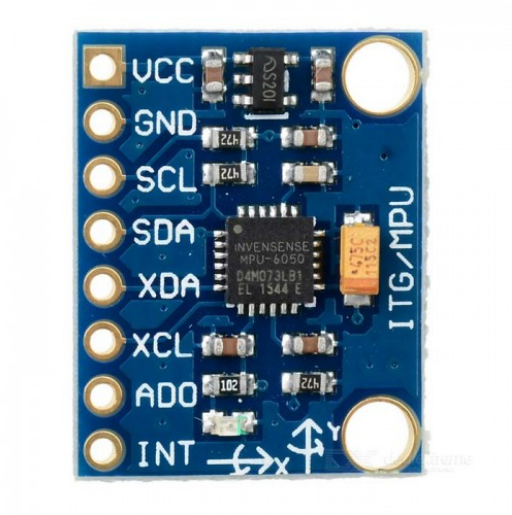
Figura 5.2: Cable USB a micro usb, necesario para conectar la wemos d1 r32 mediante usb a un ordenador.

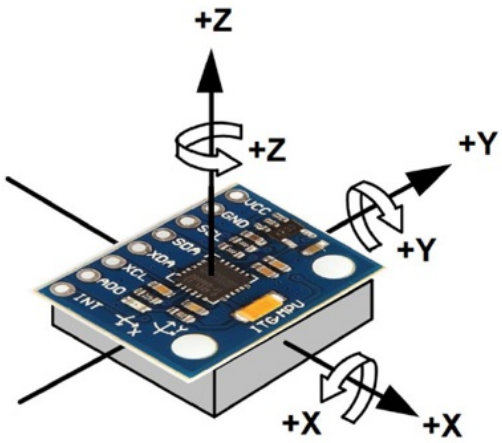
Otro factor necesario será contar con un cable USB a micro USB como el de la figura 5.2, que pueda transmitir información y no solo voltaje. Pues con el podremos poner la Wemos en funcionamiento y cargarle el software deseado.

5.2. Módulo MPU-6050

El módulo MPU-6050 5.3, es un medidor inercial que combina un giroscopio y un acelerómetro, cada uno de ellos en los tres ejes, x, y, z. Su *datasheet* [29] es compartida con el modelo MPU6000 pues comparten algunas similitudes. Aún así esta es una ficha técnica tan completa como cualquier otra de la que podemos obtener algunos datos muy relevantes como sus estructuras, ejemplos de su funcionamiento y demás, aunque a recalcar para nuestro caso son los voltajes de entrada y salida que van de 2.375V a 3.46V. Es decir que podemos alimentarlo con nuestro microcontrolador.

Entre las entradas y salidas destacamos las siguientes pues son las que vamos a emplear





(a) Esquema del MPU6050 que nos ayuda a comprender como se sitúan los ejes del dispositivo y hacia que dirección aumentan los valores de estos

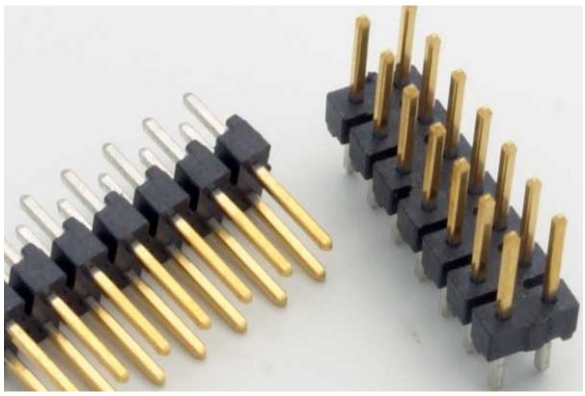
Figura 5.3: MPU6050 y junto a un esquema para su correcta comprensión

- VCC o colector de voltaje
- GND o tierra
- SCL, la señal de reloj
- SDA, la señal de datos

El módulo MPU6050 nos provee de la velocidad angular, pero lo que requerimos de él es obtener el ángulo del módulo. Para ello se emplea la siguiente formula, por la que calculamos el ángulo a partir del ángulo inicial del modulo y su desplazamiento en el tiempo.

$$\theta_x = \theta_{x_0} + \omega_x \Delta t$$

En el caso de este módulo, para su correcto uso requerimos soldarle una serie de pines a los que posteriormente conectaremos nuestros cables dupont 5.4.



(a) Tira de pines que se deben soldar al módulo.

(b) Tira de cables dupont, son los cables que se han empleado para interconectar el hardware.

Figura 5.4: Tiras de pines y cables dupont.

5.3. Joystick

El siguiente componente a tener en cuenta es el joystick 5.5, el cual cumple perfectamente el papel de accionador de nota en el controlador MIDI, pero además lo hace con la ventaja de proporcionarnos dos funciones más que podemos explotar en el proyecto, estos son sus sensores analógicos de eje x y eje y, los cuales se pueden combinar o usar por separado.

Figura 5.5: Módulo de joystick empleado en el proyecto, se pueden apreciar en su lado izquierdo los pines que vamos a emplear.

En el caso del joystick a diferencia del anterior módulo vamos a requerir de cablear todos sus pines los cuales son los siguientes.

- GND o tierra, el polo negativo del módulo
- +5V, es el pin por el que suministramos energía al módulo. En concreto 5V.
- VRX, este pin nos da la información del eje X mediante un voltaje proporcional.
- VRY, este pin nos da la inforamción del eje Y mediante un voltaje proporcional.
- SW, es el pin de *Switch* y nos manda un valor digital de 0 o 1 si el joystick ha sido pulsado. En nuestro caso, la placa Wemos cuenta con resistencias internas que permiten activar el modo de funcionamiento *INPUT PULLUP* [30], el cual activa un sistema interno que garantiza que la señal esté en un estado conocido.

El joystick es una buena herramienta ya que es un accionador más complejo que un botón. Sin embargo, en este caso nos produce un inconveniente, los valores de voltaje con los que trabaja el joystick son 5 voltios mientras que los pines de entrada y salida de datos analógicos en la placa funcionan con 3 voltios. Aún así podemos alimentarla porque contamos con un par de salidas de 5 voltios, pero el análisis de datos puede causar problemas porque las entradas son de 3 voltios. Esto da como resultado que las entradas analógicas esperen 3 voltios y reciban más de eso, obteniendo valores fuera del rango de valores normal. El normal varía entre [0, 1023] pero cuando el joystick envía voltajes superiores a 3 voltios obtenemos valores que pueden superar dicho rango. La codificación de los procedimientos de lectura de los pines se ha diseñado teniendo en cuenta esto, y produciendo valores correctos en todo el rango pero con menos resolución.

5.4. Montaje

Una vez comprendidos los elementos hardware que componen el proyecto podemos pasar a su montaje. Todos los componentes cuentan con pin de tierra por lo que los vinculamos a una misma salida de tierra. A continuación conectamos las alimentaciones de los dispositivos. El MPU puede funcionar con 3V por lo que lo llevamos al pin de 3V de la placa. Por otro lado el joystick será preferible alimentarlo en una de las dos salidas de 5V de la placa.

Pasando a las particularidades, el módulo MPU-6050 lo conectaremos mediante los puertos SDA y SCL, respectivamente a los pines con mismo nombre en la placa. En cuanto al joystick, su pulsador irá a los pines digitales, más concretamente al 27. Por otro lado los analógicos X e Y irán a los pines analógicos 4 y 36.

Con todo ello a continuación podemos ver el esquema, ver 5.6, hecho con la herramienta Fritzing [18]. Se ha añadido una *protoboard*[31] exclusivamente en el esquema para facilitar la comprensión del mismo.

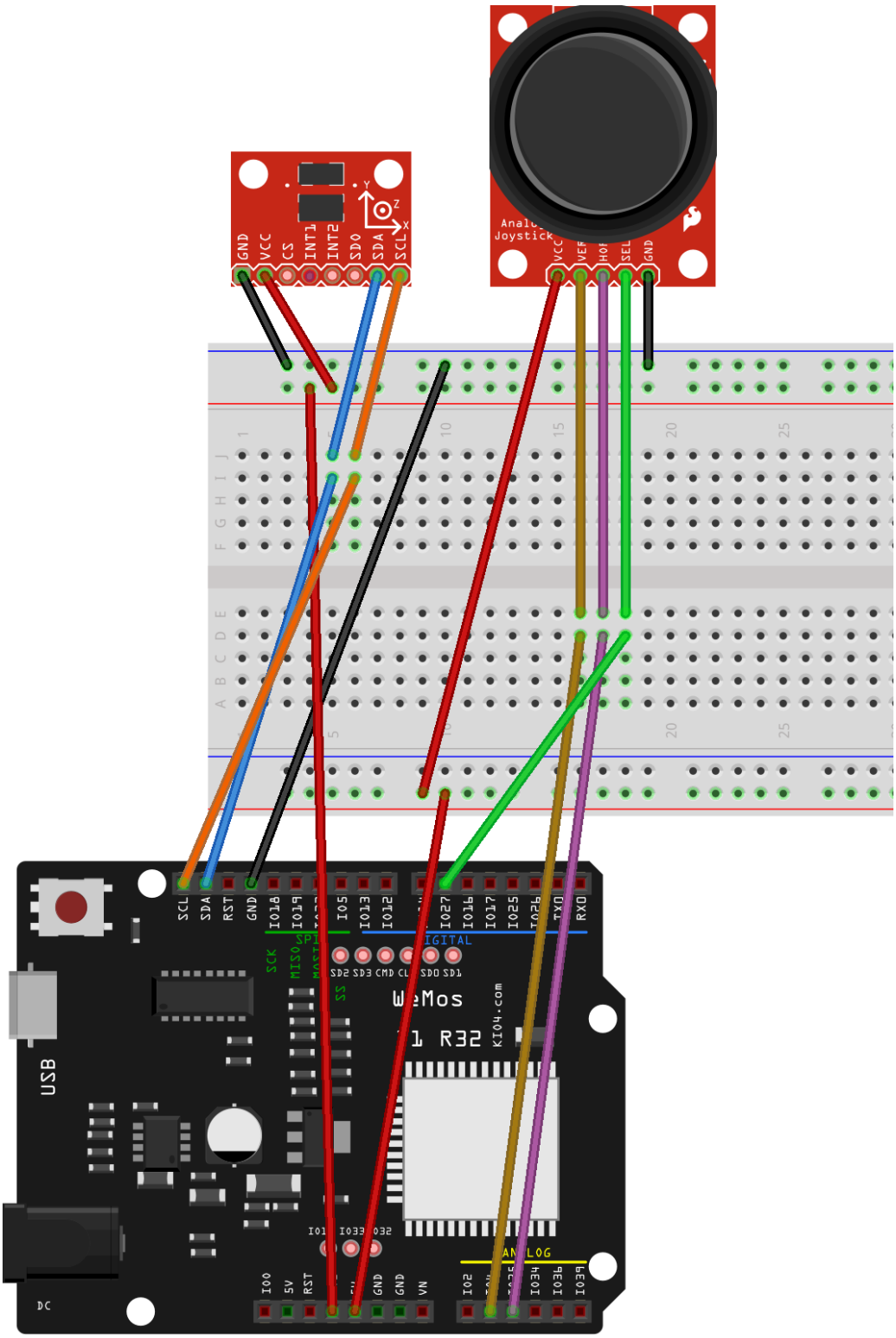


Figura 5.6: Esquema de cableado realizado en Fritzing.

Capítulo 6

Software

Una vez adquiridos los componentes hardware se recurrió a los test y ejemplos que acompañan a las librerías de cada módulo para comprender como comunicarse con ellos y posteriormente se continuó con la etapa programación.

6.1. Puesta a punto de las herramientas software

Tras la instalación de el driver CH340 y el entorno de desarrollo Arduino IDE en nuestro equipo, nos encontramos con que se debía instalar también un *plugin* para que el entorno de Arduino IDE pudiese convertir las instrucciones a las placas de la marca Wemos. Para ello hay que acceder a las preferencias del IDE y en el apartado de *Additional boards manager URLs* incluir un gestor de placas Wemos, para este proyecto se ha optado por el gestor de *Espressif,* [32] el cual cuenta con actualizaciones recurrentes y una documentación adecuada.

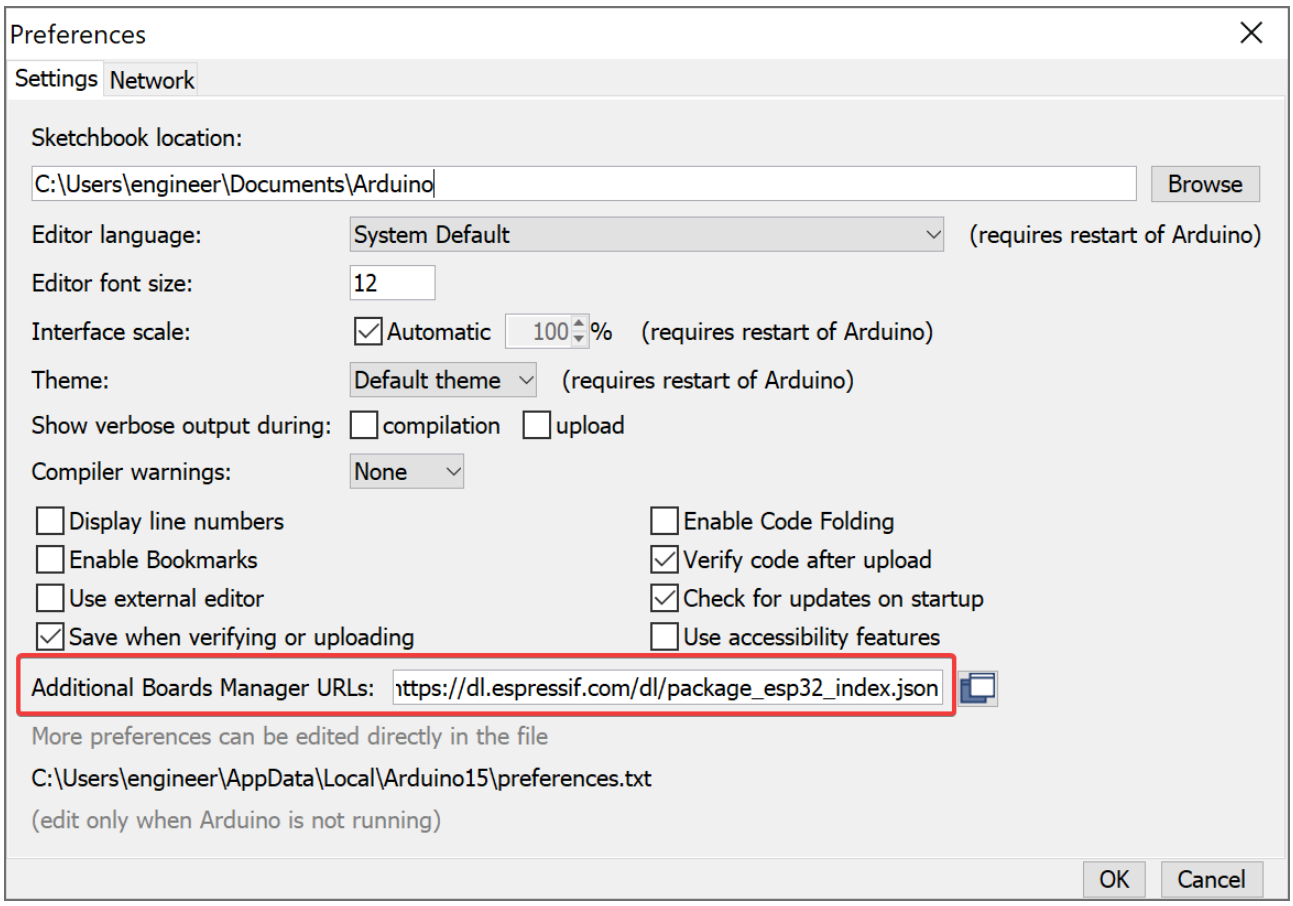


Figura 6.1: Apartado de preferencias del Arduino IDE donde podemos especificar que gestores de placas vamos a emplear.

Una vez que el gestor del microcontrolador está en el IDE habrá que buscar la placa Wemos D1 R32 en el apartado *Tools/Boards*. Si no apareciera el microcontrolador es que el IDE requiere de un reinicio para cargar adecuadamente los paquetes instalados.

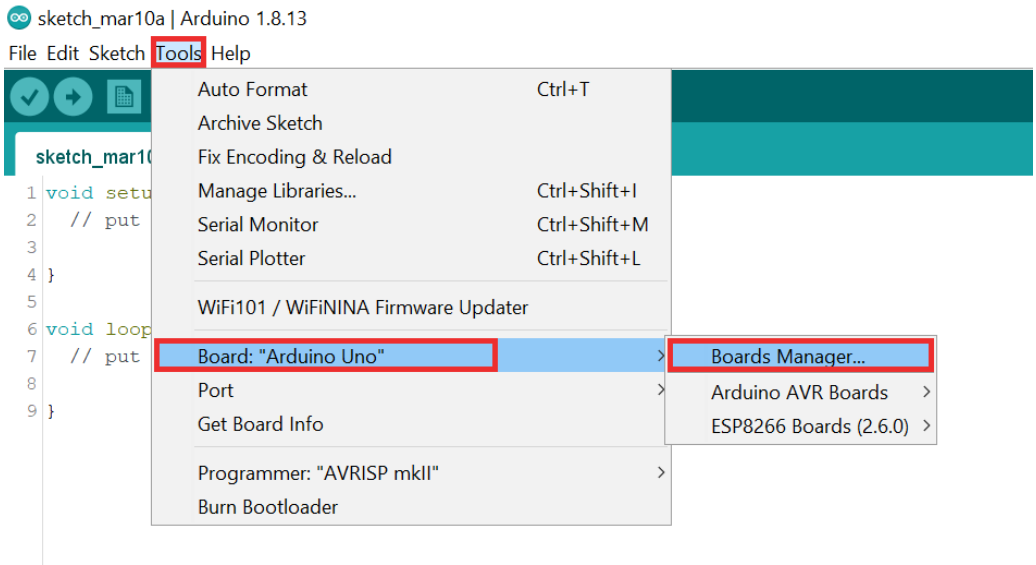


Figura 6.2: Pestaña de Tools en la cual señalaremos el microcontrolador a usar dentro del apartado Boards.

Con esto realizado ya se podrá comenzar a ver los ejemplos que se proporcionan para comprender de forma práctica como programar el microcontrolador.

6.2. Librerías

Para facilitar el trabajo a la hora de producir el software se recurrió en varias ocasiones a varias librerías que han resultado esenciales. Además todas ellas contaban con documentación y ejemplos que han facilitado las tareas de desarrollo. Las librerías son las siguientes:

- Arduino Midi Library, de FortySevenEffects[33]. Esta librería añade métodos de comunicación MIDI tanto de entrada como de salida a las placas Arduino y similares.
- El módulo para MPU-6050 de la librería I2Cdevlib, de jrowberg [34]. Esta librería nos provee de interfaces más amigables a diferentes dispositivos que se comunican por I2C.
- Arduino BLE-MIDI Transport, de lathoub[35]. Esta librería implementa una capa de transporte para las mensajes MIDI. Se basa en la librería Arduino Midi Library.

Además existen librerías por defecto que se incluyen en el gestor del microcontrolador. La que vamos a nombrar de este tipo de librerías es Wire.h una librería que facilita las comunicaciones I2C.

Tras descargar e instalar las librerías en el IDE solo requeriremos de una llamada `include` para incluirlas en nuestro código.

6.3. Código

En este apartado se analizarán poco a poco los fragmentos del código implementado y explicando sus funciones además de justificándolos si fuese preciso.

En estas primeras líneas se pueden ver todos los `include` que se han hecho en el código, como se puede apreciar son las librerías de las que se hablaba en el apartado anterior. Hay que puntualizar que la versión para la Wemos D1 R32 unifica las librerías de MIDI y de Bluetooth, ambas de los mismos autores.

```
1 #include <BLEMIDI_Transport.h> //Libreria para el transporte BT y MIDI
2 #include <hardware/BLEMIDI_ESP32_NimBLE.h> // Ajustar el BT a la placa
3 #include "MPU6050.h" //Facilita la lectura del acelerometro MPU6050
4 #include "Wire.h" //Libreria que facilita la comunicacion I2C
```

Código Fuente 6.1: Cuatro primeras líneas del software. Todos de ellos `include` de librerías.

A continuación nos encontramos con la declaración de `BLEMIDI_CREATE_DEFAULT_INSTANCE`, la cual crea una interfaz MIDI a través de Bluetooth y la vincula al *serial port* del microcontrolador. Con esto hecho pasamos a inicializar el módulo MPU6050. Creamos un objeto de clase MPU6050, y le añadimos el nombre sensor. Asociado a esto creamos una serie de variables que nos serán de ayuda a la hora de capturar los valores provistos por el MPU6050. Estos son los valores en crudo porque recordemos que debemos pasar estos datos por una función para obtener la posición angular del módulo.

```
1 BLEMIDI_CREATE_DEFAULT_INSTANCE();
2
3 MPU6050 sensor; //Sensor MPU6050 - Giroscopio y Acelerometro
4 short ax, ay, az; // Valores RAW del acelerometro  en los ejes x y z
```

Código Fuente 6.2: Se crea la instancia BLEMIDI y el objeto de clase MPU6050.

En las siguientes lineas, aún en el preámbulo del código, encontramos la configuración de los pines del joystick. Estas líneas no son cruciales pues podríamos poner el valor del pin en donde fuera necesario, pero esta forma de plantearlo nos permite cambiar el parámetro que deseemos de un pin a otro solo alterando el valor del pin en la declaración de la variable. Esto nos da mayor flexibilidad. Los pines que hemos declarado son los siguientes, el *switch* del *joystick* se encuentra en el número 26, y los pines de los ejes X e Y son 36 y 4 respectivamente.

```
1 //Joystick
2 const int SW_pin = 26;  // IO26
3 const int X_pin = 36;   // IO36 Eje de la escala
4 const int Y_pin = 4;    // IO4 Eje del pitch
```

Código Fuente 6.3: Declaración de los pines del joystick.

Para las siguientes líneas de código tenemos que empaparnos del ámbito musical. Primero encontramos la variable que indica el canal. MIDI cuenta con 16 canales y a no ser que nuestro receptor solo esté escuchando uno concreto, podremos seleccionar cualquiera de ellos. Sino habrá que indicar al receptor que escuche el canal donde estamos transmitiendo. Continuamos con la velocidad de nota o el volumen, un número estándar de este suele ser 170 pero se puede variar sin ningún problema. Continuamos con las escalas, el conocido como DO central, es el DO que se encuentra en la cuarta octava. Es por ello que se selecciona aquí el valor 4.

Note Numbers

Octava	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
-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 6.3: Valores de las notas, desde 0 hasta 127, y sus respectivas octavas. Para inicializar las variables tomaremos de referente el valor 60, el DO de la cuarta octava.

Aunque algunos instrumentos no tienen disponibles sonidos para todas las octavas, como por ejemplo la percusión. Para acceder a ellos habrá que ajustar la escala, probablemente reduciéndola. Para concluir con estas lineas se encuentran la nota actual y la última nota. La nota actual se recalculará en cada variación de los valores de entrada pero hemos dejado el valor del DO central en esta variable para mantener la coherencia con la variable octava. La ultima nota es una variable que nos permite silencia la última nota que haya sonado, en comparativa a un instrumento convencional, requerimos que el sonido se detenga cuando deja de pulsarse el accionador. Y dos valores de tiempo que usaremos como herramientas de control en las funciones de calcular las octavas.

```
1 //Base musical
2 int canal = 10;
3 int d = 170; // Velocidad de nota o volumen
4 int octava = 4; // Octavas que tienen el mínimo a -1 y el máximo a 9
5 int note = 60;
6 int last_note = 60;
7 unsigned long tiempo1 = 0;
8 unsigned long tiempo2 = 0;
```

Código Fuente 6.4: Líneas de configuración de los parámetros musicales.

Como suele ser habitual en la programación enfocada a Arduino y similares, llegamos al `setup`. Esta parte recoge específicamente la configuración del dispositivo. Comenzamos iniciando el salida serial para que a través del IDE la placa pueda enviar mensajes y podamos recogerlos, es algo similar a los usuales *print*. Luego inicializamos la librería Wire, ya que como habíamos comentado, el MPU es un dispositivo que funciona mediante I2C.

Continuamos iniciando un canal de escucha MIDI. Esto lo hacemos ya que algunos dispositivos no solo reciben mensajes MIDI sino también los envían. De esta forma podemos recibir los mensajes que pudiesen enviarnos. Al no especificar ningún canal está a la escucha en los 16.

Posteriormente iniciamos el objeto sensor, el cual es del tipo MPU6050, y tanto si se inicia correctamente como si no seremos notificados de ello. Tras iniciarse se le asignan valores de calibración adecuados para la posición que tendrá en la carcasa. Por último encontramos una sentencia `pinMode` aquí indicamos el funcionamiento de un pin, como ya habíamos mencionado el pin del switch del joystick funcionará en modo `INPUT_PULLUP` para evitar ruido en la señal. Con esto termina el segmento de configuración.

```
1  void setup()
2  {
3      Serial.begin(38400);      //Inicialización de salida de serial a 115200 baudios
4      Wire.begin();              //Iniciando I2C mediante la libreria Wire
5
6    //Arranque del MIDI
7      MIDI.begin(/*MIDI_CHANNEL_OMNI*/);  // Escucha los mensajes de entrada
8      tiempo1 = millis();
9
10   // //Sensores
11     sensor.initialize();       //Iniciando el sensor MPU6050
12     if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
13     else Serial.println("Error al iniciar el sensor");
14
15     sensor.setXAccelOffset(-671);
16     sensor.setYAccelOffset(-804);
17     sensor.setZAccelOffset(1452);
18     sensor.setXGyroOffset (-378);
19     sensor.setYGyroOffset (-66);
20     sensor.setZGyroOffset (-61);
21
22   // //Joystick
23     pinMode(SW_pin, INPUT_PULLUP);
24 }
25
```

Código Fuente 6.5: Bloque de código `setup`.

Una vez vista la función `setup` podemos continuar a la función `loop` que, como su nombre indica, este segmento de código se repite constantemente. El bloque empieza con un recálculo de la octava, ya que el usuario puede mediante el joystick variar la escala en cualquier momento. Posteriormente profundizaremos en la función pero de momento avanzaremos a la siguiente línea. En la línea 46, se calcula la nota base, es decir el nuevo DO base. Partíamos desde el DO central o de cuarta escala, pero al recalcular anteriormente la escala debemos recalcular el DO que será la primera nota de la escala actual.

Con la nueva escala seleccionada y el nuevo DO correspondiente seleccionados podemos avanzar hasta el resto de notas de la escala, como se puede apreciar en la Figura 6.6.

Pasamos al `if` que se encuentra en la línea 7, accederemos al contenido de este bloque si el interruptor del joystick ha sido accionado. En ese momento pasamos a silenciar la última nota que hubiésemos enviado. En caso de que no haya ninguna nota sonando no ocurrirá nada. A continuación calculamos la nota que se quiere enviar a partir de la nota correspondiente a la inclinación respecto a la nota base de la escala. Con la nota

calculada procedemos a enviarla y a actualizar la variable `last_note`, para que podamos silenciar la nota posteriormente. Nos encontramos un bucle `while` en el que entraremos si seguimos con el botón pulsado, y nos permitirá prolongar la nota en el tiempo además el bloque de su interior permite proporcionar el efecto *Pitch Bend* a la nota que está sonando. Al dejar de presionar el botón salimos del `while` y del `if`, terminando el bucle y dando pie a que se repita el bloque `loop`.

```
1  void loop()
2  {
3    octava =  updateOctave(octava);
4    int minNote = 12 + (octava * 12);
5
6
7    if (digitalRead(SW_pin) == 0) {
8      MIDI.sendNoteOff(last_note, 0, canal);
9      note = minNote + getNoteFromMPU6050();
10     MIDI.sendNoteOn(note, d, canal);
11     last_note = note;
12     while(digitalRead(SW_pin) == 0 ){
13         int pitch = map(analogRead(Y_pin), 0, 4095, -8192, 8191);
14         MIDI.sendPitchBend(pitch, canal);
15     }
16   }
17 }
```

Código Fuente 6.6: Bloque de código `loop`.

Para concluir con el análisis del código veremos las dos funciones que empleamos, estas han sido llamadas anteriormente en el bucle `loop`, más concretamente en las líneas 45 y 52. La primera de estas funciones llamada `updateOctave` recibe la octava actual por parámetros y a partir de lo leído en el eje X del joystick eleva o reduce la octava y la devuelve. Siempre con un rango entre [-1, 9]. La otra función, `getNoteFromMPU6050` toma los valores en crudo del módulo MPU y calcula la aceleración angular a partir de una formula matemática. El valor de esta formula está aproximadamente entre [18000, -13000], es por ello que recurrimos a la `function_map` para que nos devuelva un numero proporcional pero en un rango menor [0, 11]. Este nuevo valor se lo sumaremos a la nota base para generar la nota final. Por ejemplo si partíamos de un DO en cuarta, que tiene un valor de 60. Y la función `map` nos devuelve un 2 al sumarlo pasaríamos al valor 62, es decir un RE.

```
1  int updateOctave(int octava){
2    tiempo2 = millis();
3    if((analogRead(X_pin) < 2000) && (octava < 9) &&
     ↪  (tiempo2>tiempo1+1000)){octava = octava + 1; tiempo1 = millis();}
4    if((analogRead(X_pin) > 3000) && (octava > -1) &&
     ↪  (tiempo2>tiempo1+1000)){octava = octava - 1; tiempo1 = millis();}
5    return octava;
6  }
7
8  int getNoteFromMPU6050() {
9    sensor.getAcceleration(&ax, &ay, &az); //Lee los valores del MPU6050
10   int valueNote = map(ay, 18000, -13000, 0, 11); //Mapea los valores y devuelve
     ↪  entre 0 y 11 dependiendo del valor
11   return valueNote;
12 }
```

Código Fuente 6.7: Funciones `updateOctave` y `getNoteFromMPU6050`.

Capítulo 7

Carcasa en 3D

En este capítulo se comentará del proceso de diseño y elaboración de los modelos 3D para la carcasa que contendrá el microcontrolador y los componentes, es decir el instrumento como conjunto. La finalidad de la carcasa no es solo contener los componentes sino agruparlos y ordenarlos de manera reducida y cómoda que facilite el uso del controlador MIDI tal y como lo sería en cualquier otro controlador MIDI.

7.1. Planificación

El primer paso a realizar es medir el tamaño del mayor componente, el microcontrolador. La Wemos cuenta con un registro de sus dimensiones en el apartado *Specifications* de su ficha técnica 70x55x13mm. Esto facilitará la tarea.

Además, al continuar evaluando con atención la placa, se podrá observar que ésta viene, como es habitual, con unos orificios en sus esquinas que permitirán ensamblarla cómodamente 7.1. Para aprovecharlos se deben añadir al diseño unas barras que permitan soportar el microcontrolador.

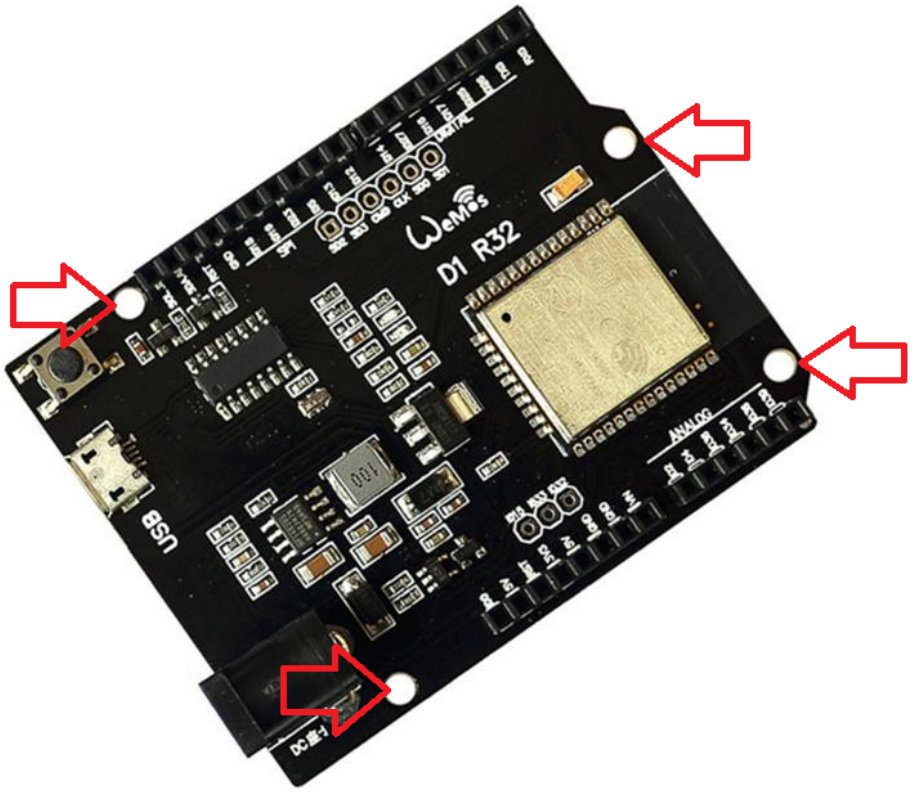


Figura 7.1: Podemos observar los orificios que facilitan el ensamblaje.

La forma más rápida de avanzar es buscar en librerías online de modelos 3D por carcasas para la Wemos D1 R32. Pero tras buscar tanto en *Google,* como en páginas más concretas como *Thingiverse*[36] o *Printables by Josef Prusa*[37] no ha sido posible encontrar nada que solucione esta etapa. Lo único que se pudo aprovechar es un modelo [38] de *Thingiverse* desarrollado por *RomanHujer* que se aproxima a lo que se buscaba, ya viene con las medidas para la Wemos por lo que se podrá usar como base.

7.2. Diseño 3D

A continuación se accederá a Fusion360 y se indicarán una serie de pasos para poder trabajar con el modelo encontrado en la web.

Lo primero es importar el archivo de extensión STL [39] para ello es necesario acceder a "*Mesh*", o Malla y en el subapartado "*CREATE*", seleccionar la opción "*Insert Mesh*". Tal como se muestra en la figura 7.2. Una vez realizado esto se contará con la malla en el entorno de Fusion360.

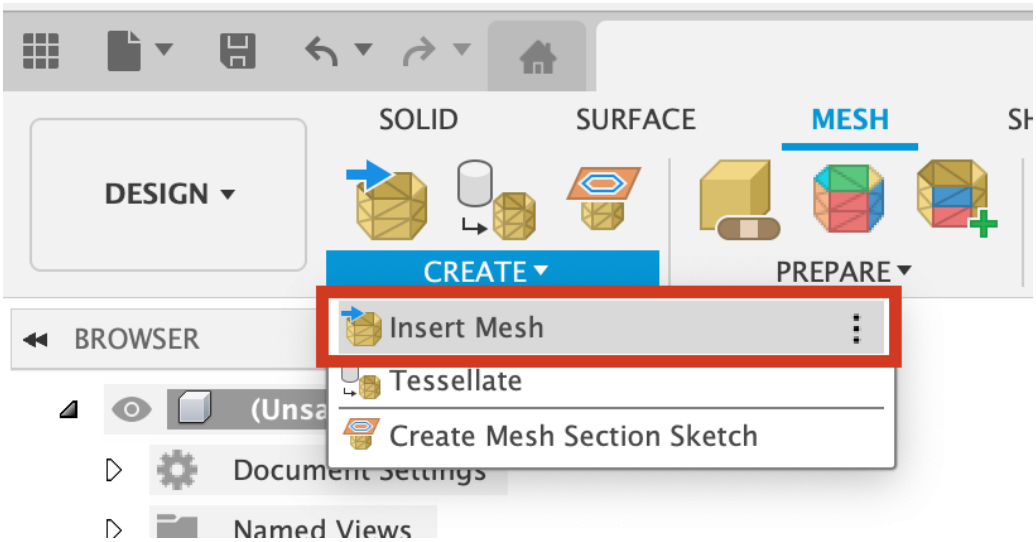


Figura 7.2: Imagen que muestra como insertar una malla en Fusion360.

No obstante Fusion360 trabaja con lo objetos *SOLID* por lo que para poder modificar tanto aristas, bordes como caras, se deberá previamente convertir la malla a solido. El procedimiento consiste en acceder a al apartado *MODIFY* de la sección *MESH* donde se haya la opción "*Convert Mesh*".

Una vez esclarecido este asunto, es posible modificar el archivo web original. Para ello se sustituyen las barras de soporte para dar paso a orificios en los que introducir tornillería y se realiza una disminución en el grosor de las paredes de la carcasa dejandolás en 2cm .

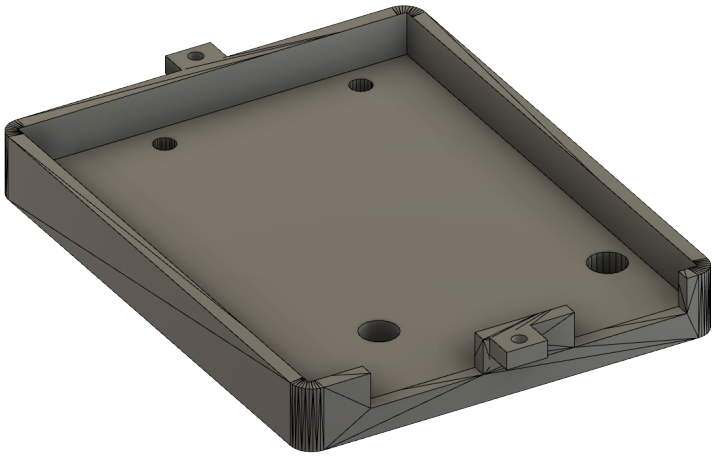


Figura 7.3: Imagen que muestra la parte inferior de la carcasa realizada en Fusion360.

El resultado de la carcasa cuenta con dos partes, la inferior, una modificación del archivo web, y la superior creada desde cero respetando las medidas de la parte inferior. En la parte superior de la carcasa se han dispuesto unos orificios para atornillar el joystick además de un estilo de rejilla en la parte superior de la misma para facilitar el flujo de

aire al interior a la vez que se hace un ahorro en material de impresión. El resultado de ensamblar ambas partes es el siguiente

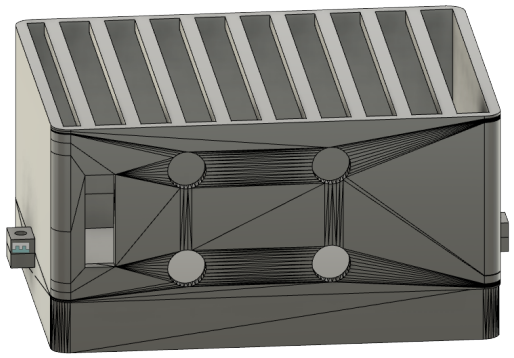


Figura 7.4: Imagen que muestra ambas partes de la carcasa ensambladas en el entorno Fusion360

7.3. Laminado 3D

Una vez se cuenta con el diseño, el siguiente paso es realizar el proceso de laminación. La laminación o laminado es un proceso por el cual un programa de laminado, denominado en ocasiones como laminador, crea una archivo que almacena las instrucciones necesarias para la impresión de un elemento 3D a partir de su archivo de objeto (por ejemplo el anteriormente mencionado STL). El archivo resultante de la laminación se adapta al lenguaje propio de la impresora, además contiene datos de configuración de la impresión que han sido establecidos por el usuario. Estos datos de configuración pueden ser el espaciado entre capas o la velocidad de impresión, entre otros. El programa que emplearemos para laminar la carcasa es el anteriormente nombrado Ultimaker Cura.

Para emplearlo se deben cargar en el laminador los archivos a imprimir, cura permite cargar ambos o cargarlos de forma individual. En este caso los cargaremos a la vez. Esto implica que se imprimirán ambos por el mismo archivo de impresión.

Al cargar los archivos se obtiene una imagen similar a esta 7.5. Una vez todo esté configurado al gusto del usuario se debe seleccionar *Slice* para comenzar el laminado. Este proceso retorna el tiempo que se calcula que tardará la impresión y el archivo que deberemos pasar a la impresora 3D.

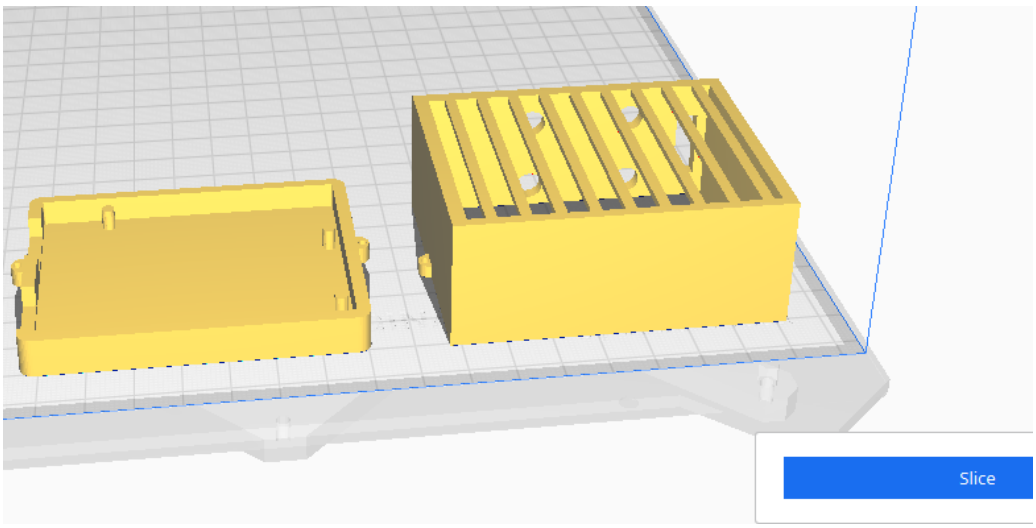


Figura 7.5: Imagen que muestra ambas partes de la carcasa ensambladas en el entorno Fusion360

7.4. Impresión

Una vez exportados los archivos de impresión se deben llevar a la impresora, la forma de hacer esto varía dependiendo de la impresora pudiendo ser por USB, micro SD u otros. La impresora 3D empleada ha requerido de un tarjeta SD para transmitir los archivos de impresión.

Para este proyecto, el tiempo se estimaba aproximadamente en 5 horas. Dos para elaborar la parte inferior y tres para la parte superior. Este tiempo no solo depende del objeto 3D sino que afectan factores como la impresora, el material que se desea emplear o la configuración del grosor para el material depositado por la impresora, entre otros.

Figura 7.6: Fotografía de la placa Wemos D1 R32 en la parte inferior de la carcasa con dos de sus punto de anclaje atornillados.

Tras obtener ambas piezas solo resta montar la placa y los demás componentes, es posible ver como la placa encaja perfectamente en la figura 7.6.

Capítulo 8

Conclusiones y líneas futuras

8.1. Conclusiones

Estoy orgullosa del trabajo realizado y aunque no esperaba los altibajos que pudiese encontrarme, no por ello he dejado de disfrutar el proceso en todas sus partes. El principal problema ha sido llevar el tiempo en contra durante la convocatoria, pero aún así ha salido adelante un proyecto que cumple las expectativas que tenía puestas en él.

Creo que el proyecto cuenta con ciertas limitaciones a nivel electrónico que podrían solventarse de muchas maneras. Claro está que no llega a un controlador MIDI de mercado, pero ese no era el objetivo. Quisiera no poner énfasis en como se limita al compararlo con los elementos disponibles en el mercado sino en las cualidades que tiene como un proyecto que está disponible para saciar las ganas de descubrimiento tanto mías como de otras personas, enfocándolo en las posibilidades que tiene de escalar en muchas direcciones diferentes. Tanto de manera artística como en lo referente al apartado técnico.

Espero que además sirva como recurso para otras personas que quieran realizar proyectos de una índole similar. Pues este TFG demuestra que las posibilidad son muy diversas y que con los recursos adecuados podemos dar con el resultado que buscamos o incluso uno mejor.

8.2. Líneas futuras

Algunas de las líneas futuras que he barajado durante el desarrollo serían:

- Integrar un batería para poder desvincular completamente el instrumento del ordenador y así eliminar el cableado restante.
- Diseñar e imprimir una carcasa 3D de mayor ergonomía para que pueda ajustarse a la mano con mayor comodidad.

- Incorporarle algún elemento que proporcione una retroalimentación al usuario, como leds o efecto de vibración, tras su conexión.

Capítulo 9

Conclusions and future lines

9.1. Conclusions

I am proud of the work I have done and although I did not expect the ups and downs that I might encounter, I have not stopped enjoying the process in all its parts. The main problem was that the weather was against me during the call for proposals, but even so, a project that fulfils the expectations I had placed in it has gone ahead.

I think the project has certain limitations at the electronic level that could be solved in many ways. Of course it falls short of a commercially available MIDI controller, but that was not the aim. I don't want to emphasise how it is limited when compared to what is available on the market, but rather the qualities it has as a project that is available to satisfy my own and other people's desire for discovery, focusing on the possibilities it has to scale in many different directions. Both artistically and technically.

I hope that it will also serve as a resource for other people who want to carry out projects of a similar nature. This TFG shows that the possibilities are very diverse and that with the right resources we can achieve the result we are looking for or even better.

9.2. Future lines

Some of the future lines I have considered during the development would be:

- Integrate a battery so that the instrument can be completely disconnected from the computer and thus eliminate the remaining wiring.
- Design and print a 3D housing with improved ergonomics to fit the hand more comfortably.
- Incorporate some element to provide feedback to the user, such as LEDs or vibration effect, after connection.

Capítulo 10

Presupuesto

10.1. Licencias de software

Nombre del software	Coste	Anotaciones
Arduino IDE	0.00 €	Gratuito. Código libre
C++	0.00 €	Gratuito
Garage Band	0.00 €	Gratuito. Es parte de los sistemas IOS
GitHub	0.00 €	Gratuito. Código libre
Fritzing	8.00 €	
Autodesk Fusion 360	0.00 €	Licencia de estudiante provista por la ULL
UltiMaker Cura	0.00 €	Gratuito
LaTeX (pdfLaTeX)	0.00 €	Gratuito. Código libre
Overleaf	0.00 €	Gratuito. Código libre
Trello	0.00 €	Plan Gratuito
Windows 11	0.00 €	Coste incluido en el precio del equipo
Total:	8.00 €	

Cuadro 10.1: Coste de las licencias de software utilizadas.

10.2. Materiales

Material	Valor	Factor de amortización	Coste
Equipo sobremesa	908.90 €	5 %	45,44 €
Ipad (2022)	579.00 €	5 %	28,95 €
Wemos D1 ESP32	11.50 €	100 %	11.50 €
Joystick XY con pulsador	2,10 €	100 %	2,10 €
Acelerómetro-Giroscópio MPU-6050	3,85 €	100 %	3,85 €
Cables Dupont 20 cm macho-hembra	3,10 €	100 %	3,10
Total:			94,94 €

Cuadro 10.2: Coste de los materiales utilizados

10.3. Costes de personal

Asumiendo un coste por hora de 30€.

Tarea	Horas	Coste
Desarrollo del software	50	1,500.00 €
Investigación *machine learning*	68	2.040.00 €
Elaboración de la memoria	16	480.00 €
Total	208	4.020.00 €

Cuadro 10.3: Costes de personal.

10.4. Presupuesto final del proyecto

Motivo	Coste
Licencias de software	8.00 €
Materiales	94,94 €
Mano de obra	4.020.00 €
Coste	4.122,94 €
Beneficio (6 %)	246,20 €
Presupuesto total del proyecto	4.369,14 €

Cuadro 10.4: Presupuesto final del proyecto.

Apéndice A

Códigos

A.1. Código completo a cargar en el microcontrolador

```
#include <BLEMIDI_Transport.h> //Libreria para el transporte bluethoot de contenido midi
#include <hardware/BLEMIDI_ESP32_NimBLE.h> // Libreria para ajustar la conexion BT a la placa
#include "MPU6050.h" //Libreria que facilita la lectura del acelerometro MPU6050
#include "Wire.h" //Libreria que facilita la comunicacion I2C

BLEMIDI_CREATE_DEFAULT_INSTANCE();  // Crear y vincular, la interfaz MIDI al Serial port por defecto del hardware

MPU6050 sensor; //Sensor MPU6050 - Giroscopio y Acelerometro
short ax, ay, az; // Valores RAW del acelerometro  en los ejes x y z

//Joystick
const int SW_pin = 26;  // IO26
const int X_pin = 36;   // IO36 Eje de la escala
const int Y_pin = 4;    // IO4 Eje del pitch

//Base musical
int canal = 10;
int d = 170; //Velocidad de nota o volumen
int octava = 4; //Octavas que tienen el minimo a -1 y el maximo a 9
int note = 60;
int last_note = 60;
unsigned long tiempo1 = 0;
unsigned long tiempo2 = 0;

void setup()
{
    Serial.begin(38400);     //Inicialización de salida de serial a 115200 baudios
    Wire.begin();             //Iniciando I2C mediante la libreria Wire

  //Arranque del MIDI
    MIDI.begin(/*MIDI_CHANNEL_OMNI*/);  // Escucha los mensajes de entrada
    tiempo1 = millis();

```

```
34    // //Sensores
35      sensor.initialize();      //Iniciando el sensor MPU6050
36      if (sensor.testConnection()) Serial.println("Sensor iniciado
        ↪  correctamente");
37      else Serial.println("Error al iniciar el sensor");
38
39      sensor.setXAccelOffset(-671);
40      sensor.setYAccelOffset(-804);
41      sensor.setZAccelOffset(1452);
42      sensor.setXGyroOffset  (-378);
43      sensor.setYGyroOffset  (-66);
44      sensor.setZGyroOffset  (-61);
45
46    // //Joystick
47      pinMode(SW_pin, INPUT_PULLUP);
48  }
49
50
51  void loop()
52  {
53    octava =  updateOctave(octava);
54    int minNote = 12 + (octava * 12); // In MIDI 60 is equivalent to C4 or Do
55
56
57    // //Si se pulsa el botón se envia una nota y se espera una cantidad de tiempo
58    if (digitalRead(SW_pin) == 0) {
59      Serial.print("Switch:  ");
60      Serial.print(digitalRead(SW_pin));
61
62      MIDI.sendNoteOff(last_note, 0, canal);
63      note = minNote + getNoteFromMPU6050(); //Obtiene la nota a enviar con base
        ↪  en Do cuarta a Si en cuarta
64      Serial.print(note + "\n");
65      MIDI.sendNoteOn(note, d, canal); //nota velocidad canal
66      last_note = note;
67      while(digitalRead(SW_pin) == 0 ){
68          int pitch = map(analogRead(Y_pin), 0, 4095, -8192, 8191);
69          MIDI.sendPitchBend(pitch, canal);
70        }
71    }
72
73    Serial.print(digitalRead(SW_pin));
74
75    Serial.print(" ");
76    Serial.print("Nota: ");
77    Serial.print(minNote);
78    Serial.print("  Octava: ");
79    Serial.print(octava);
80    Serial.print("\n");
81  }
82
83
84  int updateOctave(int octava){
85    tiempo2 = millis();
86    if((analogRead(X_pin) < 2000) && (octava < 9) &&
      ↪  (tiempo2>tiempo1+1000)){octava = octava + 1; tiempo1 = millis();}
87    if((analogRead(X_pin) > 3000) && (octava > -1) &&
      ↪  (tiempo2>tiempo1+1000)){octava = octava - 1; tiempo1 = millis();}
```

```
88    return octava;
89  }
90
91  int getNoteFromMPU6050() {
92    sensor.getAcceleration(&ax, &ay, &az); //Lee los valores del MPU6050
93    //float accel_ang_x=atan(ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);
      ↪  //Calcula la inclinacion de eje X
94    int valueNote = map(ay, 18000, -13000, 0, 11); //Mapea los valores y devuelve
      ↪  entre 0 y 11 dependiendo del valor
95    return valueNote;
96  }
97
```

Glosario

CPU Unidad Central de Procesamiento. 2

FM Frequency Modulation. 4

I2C Inter-Integrated Circuit. 12

IDE Entorno de desarrollo integrado. 7

KB Kilobyte. 15

MIDI Musical Instrument Digital Interface. 1

ms Milisegundo. 12

RAM Random Access Memory. 15

TET Tone equal temperament. 13

V Voltios. 15

Bibliografía

[1] IDIS, "Ondas Martenot." https://proyectoidis.org/ondas-martenot/.

[2] H. Gernsback, "The 'Pianorad' a New Musical Instrument which combines Piano and Radio Principles," 1926.

[3] Jason, "Theremin world." http://www.thereminworld.com/Article/14232/what-s-a-theremin-.

[4] T. B. Valley, "Breve historia de los sintetizadores." https://thebassvalley.com/breve-historia-de-los-sintetizadores/.

[5] MusicalWars, "MIDI, Samplers y Sintetizadores: Conceptos Básicos.." https://es.scribd.com/doc/89608737/MIDI-Samplers-y-Sintetizadores#.

[6] F. J. Díaz, "Instrumentos musicales en los museos de Ureña. Mellotrón." https://acortar.link/a3PEPB.

[7] N. AERONAUTICS and S. ADMINISTRATION, "Patente de la sintesis de frecuencia." https://ntrs.nasa.gov/api/citations/19710014049/downloads/19710014049.pdf.

[8] Yamaha, "Synt-DX7." https://download.yamaha.com/files/tcm:39-331941.

[9] MIDIorg, "MIDI." https://www.midi.org/.

[10] S. J. Puig, "Audio digital y MIDI." https://www.ccapitalia.net/reso/articulos/audiodigital/pdf/08-EspecificacionMIDI.pdf, 1997.

[11] J. Bartlett, "Introduction to microcontrollers," in *Electronics for Beginners*, pp. 201–211, Berkeley, CA: Apress, 2020.

[12] S. Augarten, *State of the art*. TICKNOR AND FIELDS, 1983.

[13] MIDIorg, "MIDI-2.0." https://www.midi.org/specifications/midi-2-0-specifications.

[14] GitHub, "Github official website." https://github.com/.

[15] Arduino, "Arduino IDE official website." https://www.arduino.cc/en/software.

[16] K. Groover, "The Glide." https://www.theglide.com/about-f-a-q.

[17] C++, “Standard CPP Foundation official website.” https://isocpp.org/.

[18] Fritzing, “Fritzing official website.” https://fritzing.org/.

[19] Autodesk, “Fusion official website.” https://www.autodesk.es/products/fusion-360/overview?term=1-YEAR&tab=subscription.

[20] UltiMaker, “Cura official website.” https://ultimaker.com/es/software/ultimaker-cura.

[21] T. L. project, “The latex project official website.” https://www.latex-project.org/.

[22] Overleaf, “Overleaf official website.” https://es.overleaf.com/.

[23] B. Aravena and M. Felipe, “Sistema inalámbrico para transmisión de señales de audio con mínima latencia,” 2016.

[24] A. E. D’Agostino, *Teoría musical moderna*. Melos, 2018.

[25] J. M. Barbour, *Tuning and temperament : a historical survey*. East Lansing : Michigan State College Press, 2ª ed. ed., 1951.

[26] AZ-Delivery, “Technical data sheet Wemos D1 R32.” https://www.halloweenfreak.de/arduino/pdfs/D1_R32_ENG.pdf.

[27] Google, “I2C-Bus website.” https://www.i2c-bus.org/.

[28] N. Q. M. Co., “CH340 website.” http://www.wch-ic.com/products/CH340.html?

[29] InvenSense, “Product Specification MPU6000 and MPU6050.” https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[30] P. Horowitz, *The art of electronics*. Cambridge [etc.]: University Press, 2ª ed., reimp.. ed., 2001.

[31] W. Young, “What is a "breadboard¿.” https://tangentsoft.com/elec/breadboard.html.

[32] Espressif, “Arduino-esp32.” https://github.com/espressif/arduino-esp32.

[33] FortySevenEffects, “Arduino MIDI Library.” https://github.com/FortySevenEffects/arduino_midi_library.

[34] jrowberg, “I2Cdevlib.” https://github.com/jrowberg/i2cdevlib.

[35] lathoub, “Arduino BLE-MIDI Transport.” https://github.com/lathoub/Arduino-BLE-MIDI.

[36] T. official website, “Thingiverse.” https://www.thingiverse.com/.

[37] J. Prusa, “Printables by Josef Prusa.” https://www.printables.com/es.

[38] RomanHujer, “OnStep Shield for ESPDUINO-32.” https://www.thingiverse.com/thing:4750134.

[39] Fabbers, “The STL format.” http://www.fabbers.com/tech/STL_Format.