Ciclo Formativo de Grado Superior

Mantenimiento Electrónico

CURSO 2020-21

PROYECTO:

DUNGEON MASTER'S DECK

Autor:

Álvaro García Hernantes

Profesor Tutor del Proyecto:

Pedro Alonso

Índice general

Tabla de contenido

Abstract

As a player of tabletop role-playing games, I have spent many afternoons playing with my friends and more than once, it has happened to us that we had to cut the action suddenly because something was not well prepared or just because something was wrong. With this tool, we want to create a way for the Dungeon Master to be more efficient during the games, creating a more immersive environment for then players.

The idea to solve this problem was easy: to create a macro keyboard very role-playing focused where the DM could create his own simple macros and make his life easier during the games.

With the help of all of our electronics and programing knowledge we were able to create a robust hardware and a friendly and easy to use desktop app software were the DM can configure his macros and use them during the games with just a click without worrying about anything else.

In conclusion, we have created a very useful tool for the tabletop roleplaying games that are played in person to create a better role-playing environment and help the players play better games. Our plan is to port this app to work on other devices like MacOS or Linux based devices (currently our DMD work just on Windows based devices). We also want to implement this to online games with bot macros to help the players that are forced to play online due to the current pandemic or the distance between them.

Resumen

Como jugador de juegos de rol de mesa, he pasado muchas tardes jugando con mis amigos y más de una vez nos ha pasado que hemos tenido que cortar la acción de repente porque algo no estaba bien preparado o simplemente porque no estaba bien. Con esta herramienta queremos hacer que el Maestro de Mazmorras sea más eficiente durante sus sesiones, creando un ambiente más inmersivo para sus jugadores.

La idea para resolver este problema era simple, crear un teclado de macros muy centrado en el rol donde el DM pudiese crear sus propias macros, haciéndole así la vida más fácil durante las partidas.

Con la ayuda de nuestros conocimientos de electrónica y programación hemos sido capaces de crear un hardware robusto y una aplicación de escritorio amigable y fácil de usar donde el DM puede configurar sus macros y usarlas durante sus partidas sin preocuparse de nada mas.

En conclusión, hemos creado una herramienta muy útil para los juegos de rol de mesa que se juegan en persona para crear un mejor ambiente de roleo y ayudar a los jugadores a que tengan mejores partidas. Nuestro plan es trasladar esta aplicación de escritorio a otros sistemas basados en MacOS o Linux (actualmente nuestro DMD solo funciona en dispositivos basados en Windows). Además también queremos implementar esta herramienta a partidas online mediante bots para ayudar también a los jugadores que están obligados a jugar online, ya sea por la pandemia que estamos viviendo o por la distancia entre los jugadores.

1. Introducción y Objetivos

1.1. Motivación

Soy un gran apasionado de los juegos de rol de mesa, sobre todo del conocido Dragones y Mazmorras (DnD) y he pasado innumerables tardes jugando con mis amigos a este tipo de juegos. A pesar que jugar una partida siempre es divertido, hay determinadas situaciones en las que hay que cortar la acción o sacar a los jugadores de la inmersión del juego por "fallos del directo" (no tener a mano ciertas reglas, que se olvide algo de la preparación para la inmersión como la música o el mapa…). Realmente estos fallos que cortan la acción se pueden solucionar de una forma muy simple y de aquí es de donde surge la motivación para este proyecto: no cortar la acción en una partida de rol para hacerla más entretenida y más inmersiva a los jugadores. Esto lo conseguimos haciéndole la vida más fácil al Maestro de Mazmorras (Dungeon Master en inglés, siendo que a partir de ahora nos referiremos a esta persona como DM) creando un teclado con muchas acciones rápidas que facilitan cosas tanto en la preparación de la partida como durante la misma haciendo todo más fluido e intuitivo.

1.2. Objetivos

El objetivo del proyecto es simple, crear una herramienta que ayude al DM de dos formas: para hacerle la vida más fácil y para mejorar sus partidas, mejorando así también la experiencia de los jugadores. Esto lo conseguimos creando una herramienta que permita hacer

ciertas acciones o instrucciones para realizar una tarea concreta o también conocido como "macros" en este caso desarrollaremos un teclado de macros, pero como nuestro propósito es mejorar la acción de las partidas de rol, será un poco distinto de un teclado de macros convencional.

1.3. Organización de documentos e información

Junto con este documento, se incluye una carpeta llamada DMD. Dentro de esta carpeta podemos encontrar 3 directorios; Software, Hardware y Anexos.

Dentro de Software encontraremos 3 carpetas; Arduino, Python y Final Versión. En la carpeta de Arduino se encuentran todos los códigos escritos para la plataforma de Arduino realizados durante el proyecto siendo el archivo DMD_REV_1.1.ino el código de la versión final de Arduino entregado en este proyecto. En la carpeta de Python se encuentran todos los códigos escritos para la aplicación de escritorio de DMD además de algunas carpetas y archivos temporales de Python. Añadir también que existe una carpeta llamada Others donde se incluyen un montón de pruebas de código e ideas descartadas (como la posibilidad de incluir funcionalidades con Spotify). Por último, en la carpeta Final Version se encuentra la última build del programa de escritorio en su versión portable, para poder ejecutarla en cualquier dispositivo Windows.

En cuanto a la carpeta de Hardware, nos encontramos con la carpeta de 3D_Model que incluye todos los modelos 3D utilizados para crear la carcasa donde se coloca el DMD y sus cortes para su posterior impresión 3D. Luego están las carpetas de Rev_1.0 y Rev 1.1 que contienen el diseño de la PCB de su respectiva revisión además de los archivos de taladros y todo lo necesario para su fabricación. Además, también podremos encontrar la carpeta FT232RL_Breakout_Board_Rev_1.0 y se trata del diseño de una PCB de prototipado para el integrado con el mismo nombre.

Por último, tenemos la carpeta de Anexos, en la que podremos encontrar un BOM del proyecto, el CV de los promotores, los datasheet de los componentes más importantes y demás archivos necesarios a lo largo de este documento.

Junto con las 3 carpetas principales también habrá 2 archivos, una presentación para la defensa y un pequeño clip de video en el que se muestra el DMD en una partida real.

2. Estado del arte

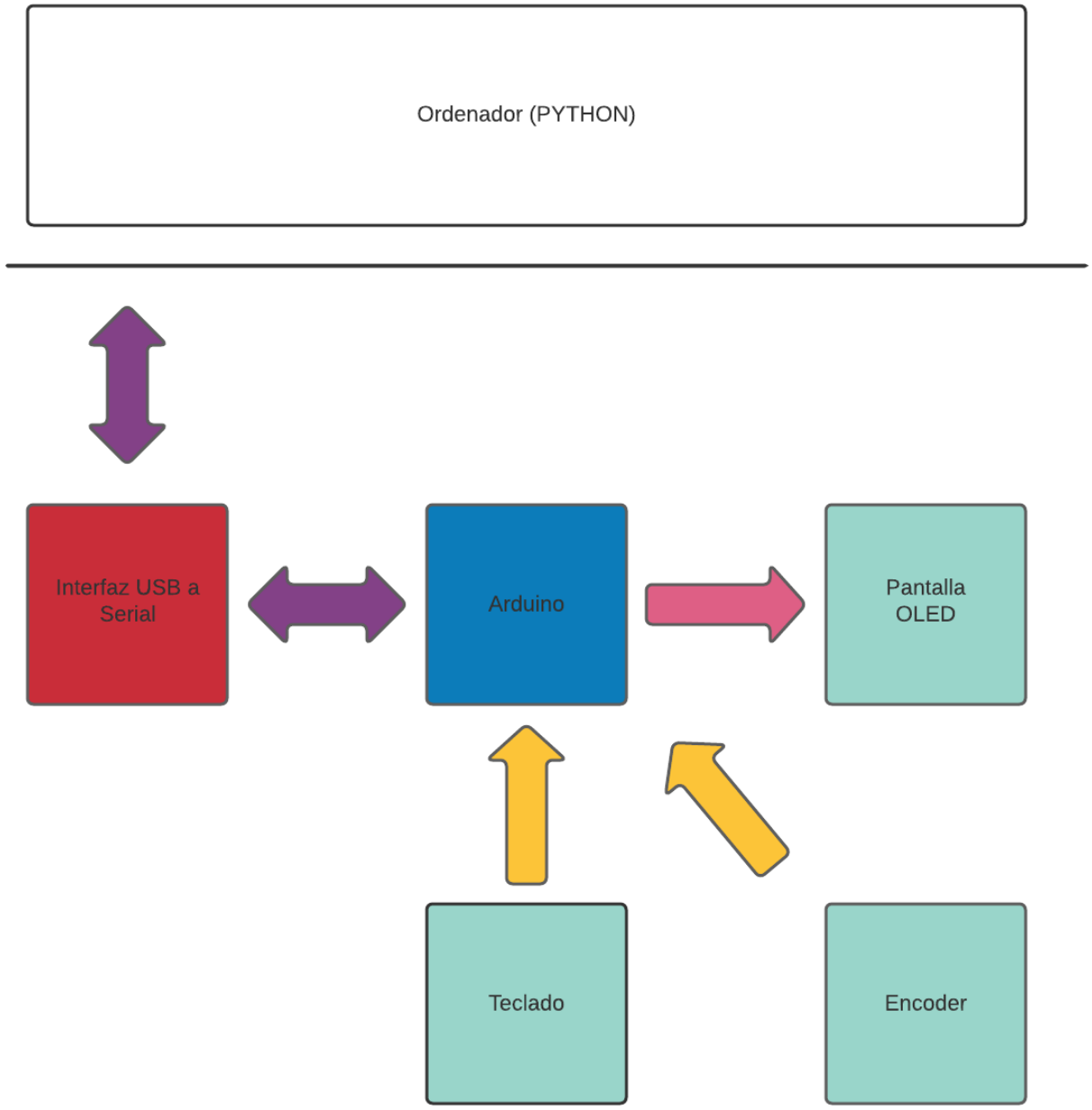
Existen muchos teclados de macros en el mercado, no son ninguna novedad y llevan existiendo entre nosotros desde hace muchos años, habiéndolos de todas las formas y colores. El público al que se enfocan este tipo de productos (o target) suele ser gente interesada por el mundo de la tecnología que acostumbra a realizar muchas acciones iguales en periodos cortos de tiempo como podría ser un editor de video para trabajar con el programa de edición de video o un técnico de sonido para hacer lo mismo con su programa de edición de audio.

Pero nuestro producto es distinto, ya que pretende llevar las macros a un terreno completamente nuevo y a un público que "generalmente" no está muy interesado en este tipo de cosas y no sabe aplicarlas a sus tareas. Por eso hemos hecho esto de forma que sea muy accesible a cualquier jugador de juegos de rol de mesa y de una forma algo simplificada, siendo así un producto nuevo en su campo.

3. Descripción del Hardware

Para crear este teclado hemos tratado de integrar la expresión más simple (y la única que necesitamos para nuestro propósito) de un Arduino en una misma PCB con el resto de periféricos, quedando así un circuito compacto que es lo que buscamos para un teclado. De cara el usuario nuestro hardware cuenta con una interfaz USB para conectarlo al ordenador (Puerto USB B) 9 teclas para las macros, un encoder y una pantalla.

3.1. Diagrama de bloques

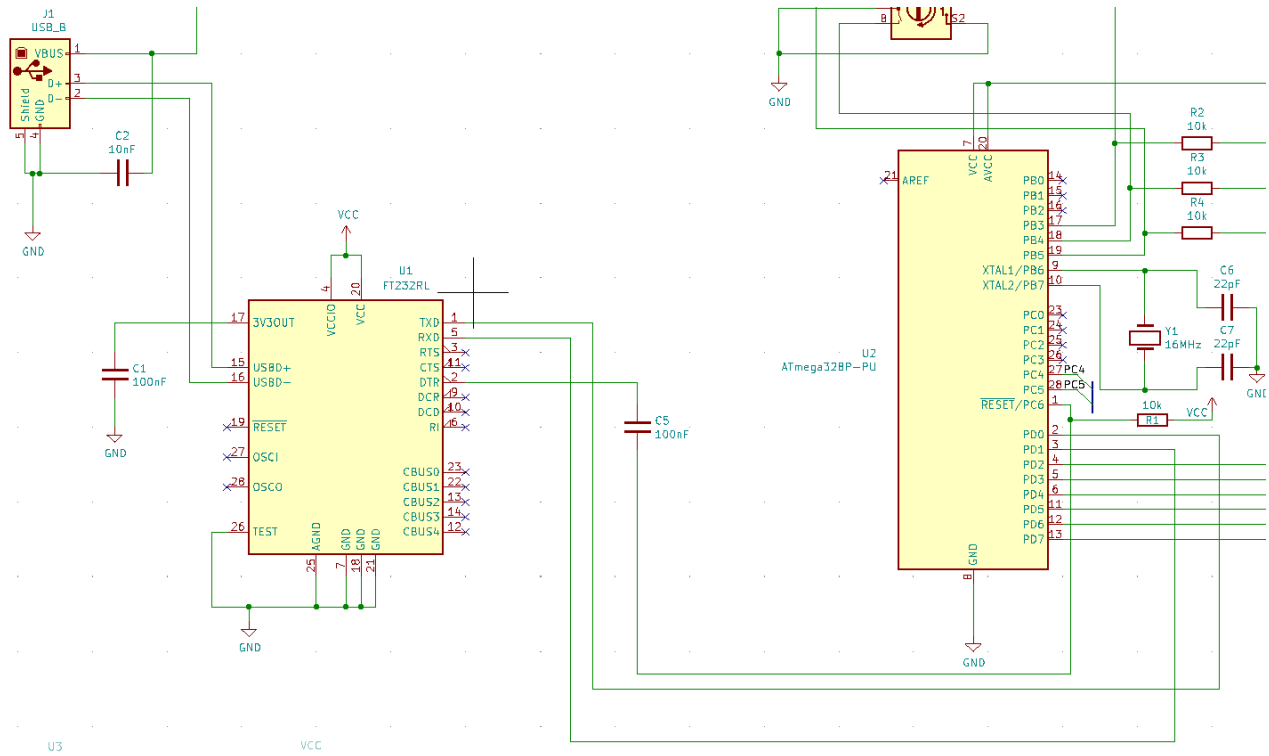


3.2. Bloque 1 – Arduino e interfaz USB/Serial

Esta parta ha sido probablemente la parte más difícil y más importante del hardware ya que siempre he trabajado en un entorno de Arduino cerrado y no he tenido que preocuparme de cómo funcionaba esa pequeña PCB. Para poder hacer esto he tenido que identificar cuáles eran las partes fundamentales para hacer funcionar un Arduino en su mínima expresión. Son dos partes muy diferenciadas; el ATMega 328P y el FT232 RL. Para hacer funcionar el primero necesitamos tan solo un cristal de cuarzo de 16MHz (Y1) y dos condensadores de 22pF (C6 y C7) conectados a este para hacerlo oscilar. Esto junto con una tensión de 5V hace

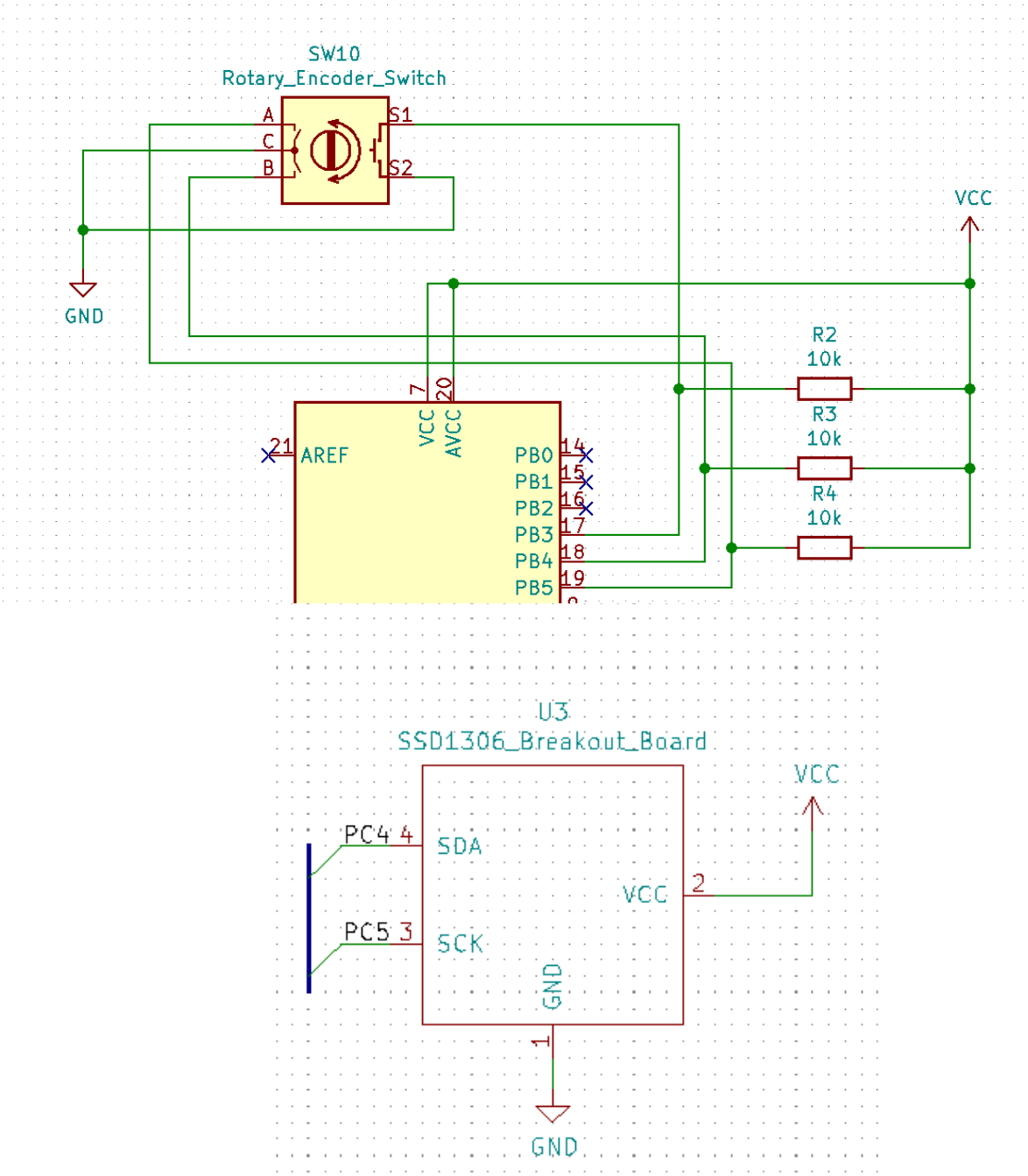
que este pequeño microcontrolador pueda ejecutar el código que tenga cargado. Y aquí es donde viene el segundo componente principal; el FT232 RL que actúa como traductor entre el ordenador y el ATMega para poder así cargarle los códigos a través de una interfaz USB como lo harías en tu Arduino convencional. Los componentes esenciales que necesita este integrado para funcionar son nulos, con la simple alimentación que proporciona el puerto USB ya funciona (aunque es cierto que se añaden algunos condensadores para eliminar ruido y conseguir una señal más limpia). Solo debemos añadir un condensador de 100 nF entre la patilla DTR del FT232 y la patilla RESET del ATMega junto con una resistencia de pull-up en esta misma patilla para que el primero pueda reiniciar el ATMega y hacerle entrar en el modo programación, pudiendo así cargar código. Si no existiese este condensador deberíamos de reiniciar el ATMega de forma manual cada vez que quisiéramos cargar un código nuevo.

Este sería el circuito eléctrico final para este bloque del hardware:



3.3. Bloque 2 – Pantalla OLED y Encoder

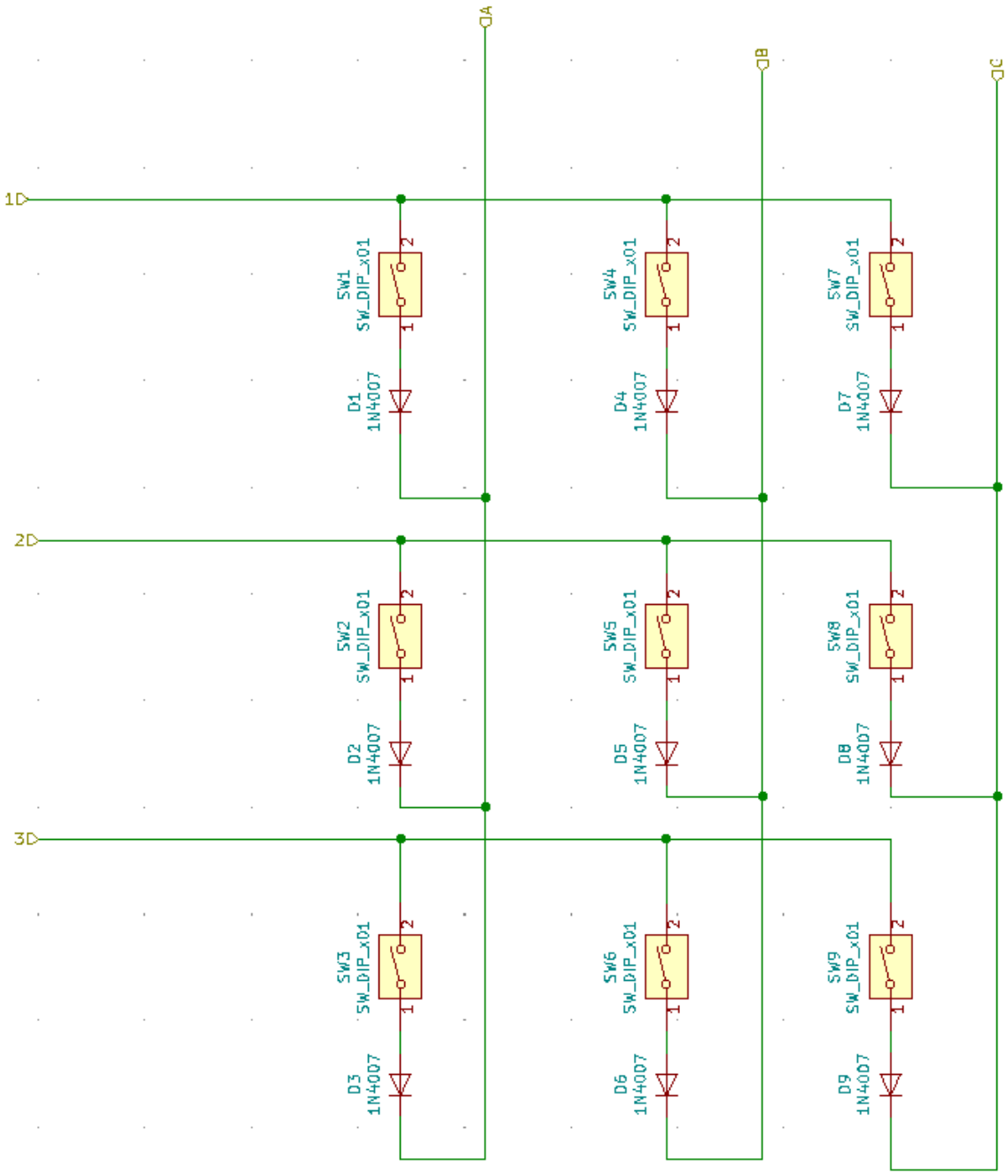
La parte de la pantalla OLED es bastante sencilla ya que al ser un dispositivo con protocolo I2C no necesita nada más allá de su alimentación de 3.3V (que obtenemos gracias al FT232 sin necesidad de ningún tipo de conversor buck o regulador de tensión) y conectar correctamente SDA y SCL al microcontrolador. En el caso del encoder no es tan fácil y directo de hacer funcionar como la pantalla OLED pero apenas necesita una conexión a tierra y unas pocas resistencias de pull-up que finalmente se han puesto por hardware y no por software mediante el ATMega porque ya que incluirlas es un gasto mínimo y así simplificamos y aseguramos el correcto funcionamiento del software en un futuro. El esquema eléctrico quedaría tal que así:



3.4. Bloque 3 – Matriz de switches del teclado.

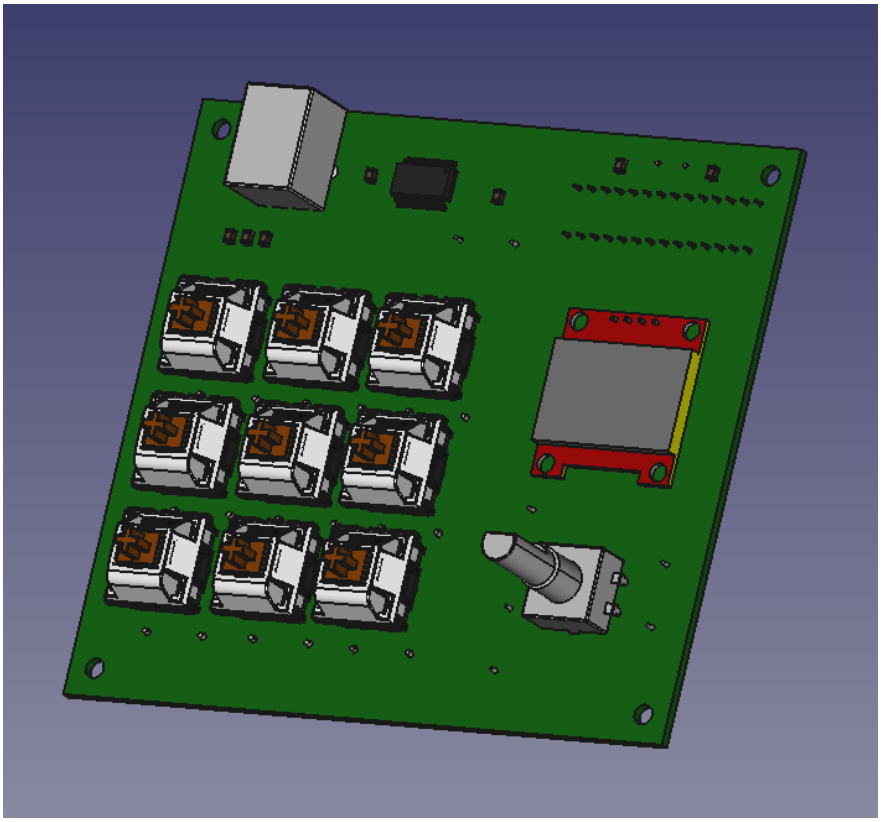
La forma fácil de identificar cada pulsador seria conectando cada uno a una patilla del ATMega junto con una resistencia de pull-up/pull-donw (depende de la lógica que quisiéramos seguir) o incluso activando las resistencias de pull-up que nos proporciona el ATMega. El problema de esto es que utiliza una patilla para cada pulsador. Cuando tienes 2-3 no hay ningún problema, pero en el momento que queremos hacer un teclado enseguida nos empiezan a hacer falta patillas. Por eso se colocan los pulsadores divididos en filas y columnas (con un diodo de por medio para evitar falsas lecturas cuando se pulsan 2 o más a la vez) que, junto con un algoritmo de software bastante simple (más detalles en el apartado de software) se consigue controlar más pulsadores con un numero considerablemente inferior de patillas.

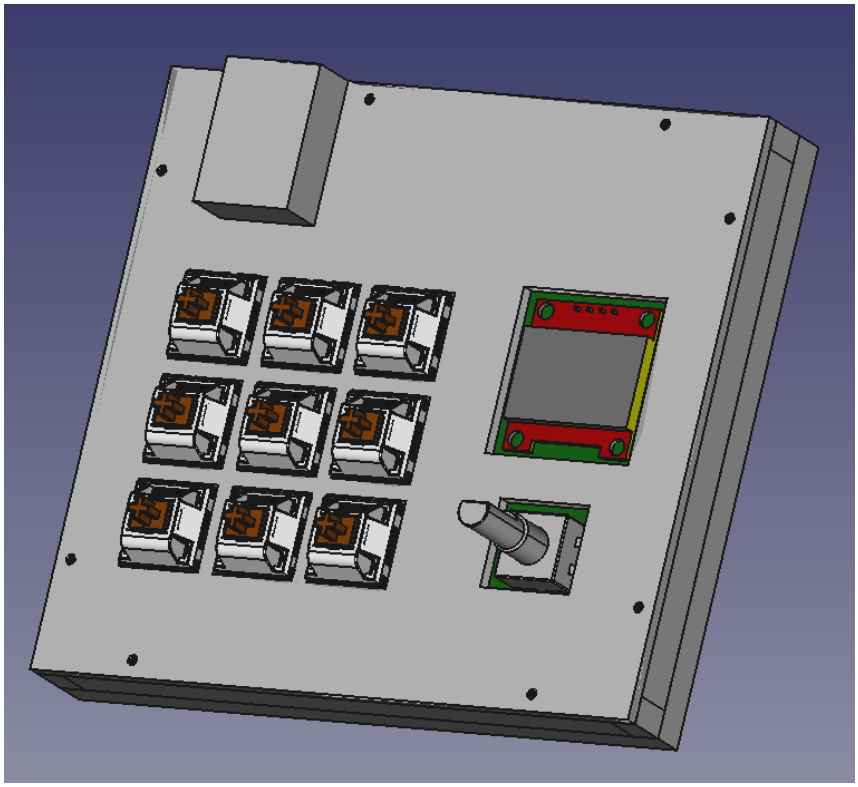
La disposición de estos switches queda tal que así:



3.5. Bloque 4 – Modelo 3D y encapsulado

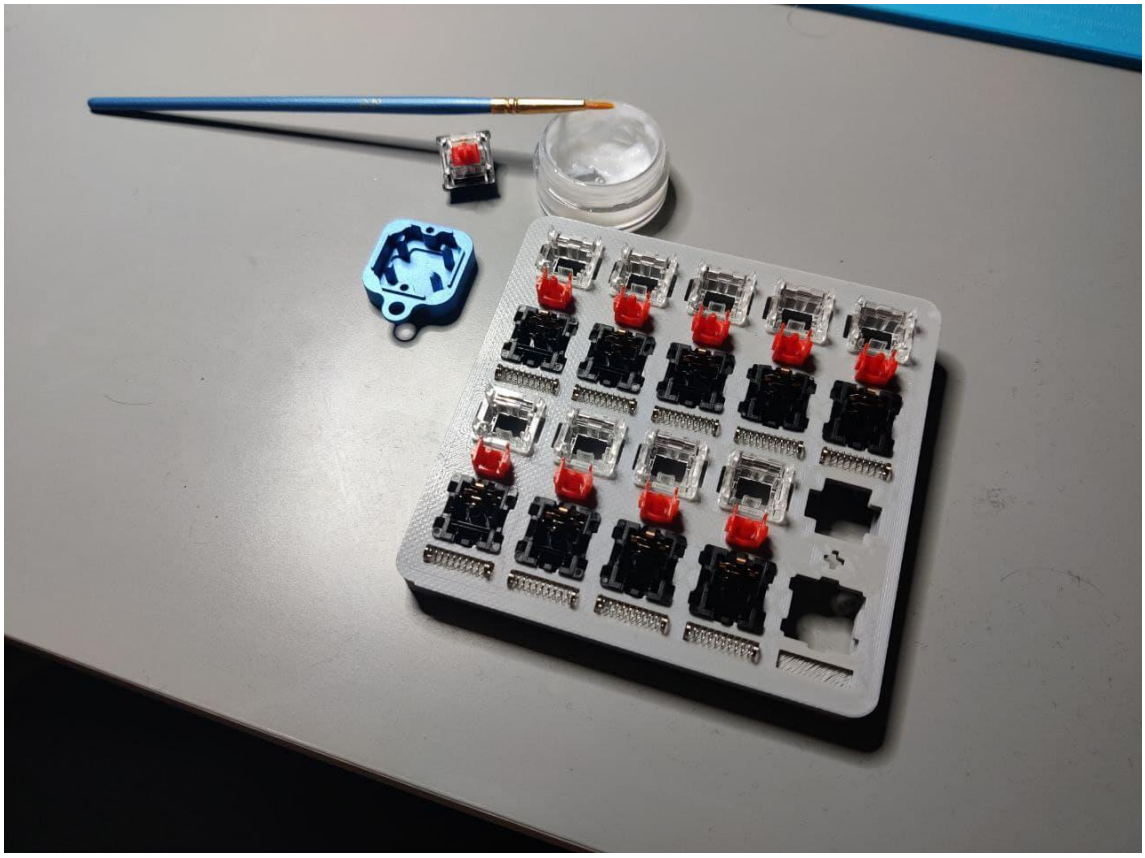
Para crear el modelo 3D en el que reposaría nuestra PCB exporte el modelo 3D con todos sus componentes previamente colocados y trabaje con él en FreeCAD, creando un encapsulado como este:





3.6. Bloque extra – Switches mecánicos y lubricación

Como ya he dicho soy un gran entusiasta de los juegos de mesa, pero también lo soy de los teclados mecánicos y de todas sus variantes así que quise darle un toque de calidad en lo que a los switches mecánicos se refiere. El primer prototipo usa unos switches Razer Green tipo 'Clicky' de tres puntas. Son de una calidad bastante pobre además de ser tipo clicky considerados los peores tanto por la comunidad de teclados mecánicos como por mí por ser los más ruidosos y por ser el único de los tres tipos que no puede ser lubricado, desmejorando mucho la experiencia de cada pulsación. Además, el hecho de que tenga 3 puntas en vez de 5 hace que sea del tipo 'plate mounted' o montado encima de la placa de sujeción de los teclados, haciendo que al estar montados en una PCB en mi caso queden mal alineados y de un aspecto poco cuidado. En la segunda revisión (aparte de cambiar la orientación de los switches para mejorar el sonido una vez puestas las teclas) decidi colocar unos Gateron Red 'linear' de 5 puntas con muelles de 45g. Al ser linear pueden ser lubricados sin ningún problema, en este caso con Krytox 205G0. El proceso de lubricación mejora mucho tanto el tacto como el sonido que producen.



4. Descripción del Software

Desarrollar el hardware de este proyecto ha sido sin duda la parte que más tiempo a consumido en el desarrollo y también la más complicada hablando tanto de hardware como de software. Está dividido en 2 partes muy diferenciadas. La primera se trata del código del programa de Arduino; no ha sido un código fácil pero debido a mi conocimiento del lenguaje y mi experiencia previa con el mismo en los pasados años no ha sido excesivamente complicado realizar un código robusto para el teclado. La otra parte bien diferenciable del código es la aplicación de escritorio para Windows, hecha en Python 3.9; debido al desconocimiento casi total del lenguaje ha sido un aprendizaje de cero y ha sido la parte más ardua de todo esto. Consta de partes muy diferenciadas e importantes como la creación de la GUI (Graphical User Interface, o Interfaz Gráfica de Usuario) o los distintos scripts para realizar tareas (como modificar el archivo de configuración o abrir/modificar los distintos recursos).

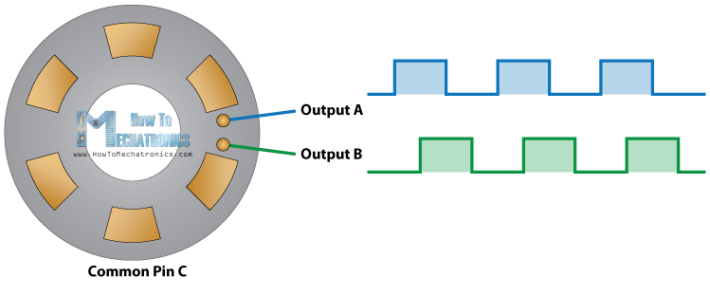
4.1. Bloque 1 – Arduino

Código de referencia: DMD_REV_1.1.ino

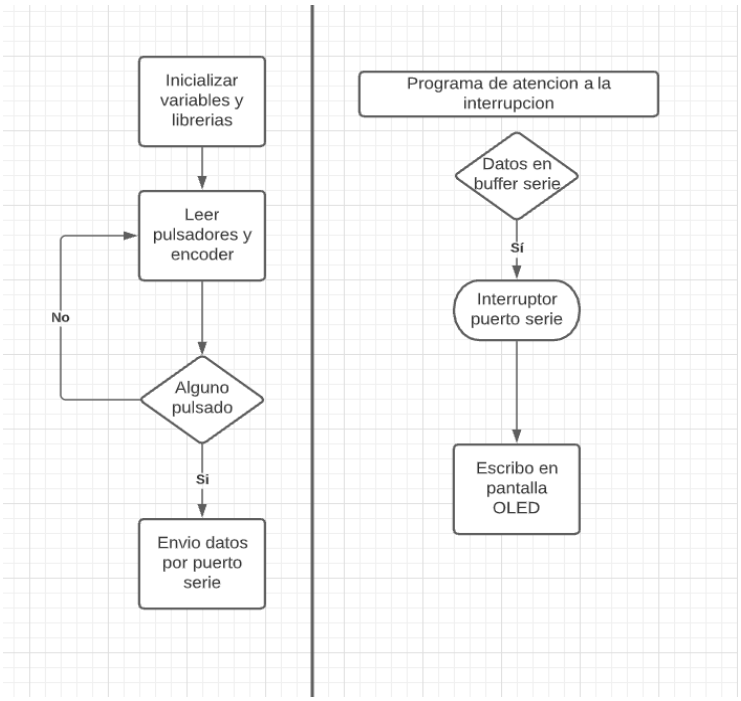
Este programa es relativamente simple; se dedica a comprobar el estado tanto de los pulsadores como del encoder (que esencialmente son interruptores) constantemente. En el caso de los pulsadores, al estar colocados en una matriz de 3x3 (como bien cite anteriormente en el apartado de hardware). Para poder hacer esto se generalmente se utiliza un algoritmo en el que, si dividimos la matriz en filas y columnas, unas de ellas se convierten en entradas y las otras en salida (dependiendo de la disposición de los diodos). En mi caso particular, las filas serán salidas y las columnas entradas. Estas columnas tendrán siempre activadas una resistencia de pull-up (en este caso está hecho por software por si en un futuro es necesario cambiar la lógica del algoritmo) por lo que normalmente estarían leyendo un "1" lógico. En el caso de las filas, al ser las salidas colocaríamos en todas ellas un "1" lógico de nuevo, pero solo hasta que el algoritmo empezase a funcionar. Una vez explicado esto, podemos pasar al siguiente paso; poner una por una las filas a "0" mientras el resto permanecen a "1"; aclarar que es muy importante que en ningún momento haya dos filas a "0" con lo cual el orden de actuación es primero poner la fila anterior a "1" y luego poner la siguiente a "0" y no al revés. Sabiendo esto, ahora es cuando entra la variable del usuario quien decide pulsar un botón de los 9. Estos cambios en la lógica de las líneas suceden a una velocidad abrumadora y según la disposición de la matriz al estar el circuito cerrado a través del pulsador, una de las líneas será capaz de poner un "0" en una de las columnas. Sabiendo que columna ha recibido el "0" y que fila estaba a "0" podemos mapear de que botón se trata y ejecutar otras partes del código

en consecuencia. A cada pulsador le corresponde una letra, la cual después de ser activado el interruptor correspondiente es enviada por puerto serie a una velocidad de 115200 baudios.

El encoder es mucho más sencillo, consta de un disco con puntos de contacto eléctrico igualmente espaciados que van conectados a GND (patilla C del encoder) y que gira a medida que nosotros giramos el encoder. Dependiendo del orden en el que los puntos A y B nos den los 0V de GND podemos averiguar el orden de giro del encoder y trabajar en consecuencia. Además el encoder cuenta con un interruptor que actúa como pulsador cuando apretamos el mando.



Por ultimo también he creado una función más para cuando se active la interrupción del puerto serie; serialEvent(). Esta función se ejecuta cada vez que hay información disponible en el buffer del puerto serie. La función recoge los datos del buffer (en este caso son todo tiradas de dados con el tipo de dado correspondiente) que vienen en un String en el formato DXXrZZ siendo XX el número del tipo de dado y ZZ la tirada que se ha realizado. Luego el programa modifica y da formato a estos String para representarlos en la pantalla OLED.



4.2. Bloque 2 – Python GUI

Código de referencia: app.py

La interfaz de usuario es una parte muy delicada del programa ya que va a ser la experiencia directa que tenga el usuario con nuestra aplicación de escritorio. Para crearla he utilizado una extensión de Python llamada Tkinter para hacer interfaces gráficas. No es la más potente ni la que más opciones tiene, pero para hacer nuestro trabajo nos sobra.

Con la función createApp() creamos la primera ventana y ventana principal de nuestro programa, que consta de 9 botones que referencian a las 9 teclas del teclado, un desplegable con todos los puertos de comunicación que hay actualmente en nuestro sistema y un botón de inicio para empezar a correr los scripts. Cuando pulsamos un botón para configurarlo, llamamos a la función selectFunction(key) que nos crea una ventana con las distintas configuraciones que le podemos dar a ese botón además de almacenar en memoria el botón que hemos pulsado para más tarde guardar todo en el archivo de configuración. Dependiendo de la función que elijamos, llamara a una función u otra. Si seleccionamos música nos abrirá otra ventana con un cuadro de texto en el que podremos pegar un link con una canción de YouTube, si seleccionamos un dado nos abrirá otra ventana con un desplegable con los tipos de dados para seleccionar el que queramos y, por último, si seleccionamos recurso nos abrirá el selector de archivos de Windows para que indiquemos en la ruta del archivo que queremos abrir. Además, todas estas ventanas cuentan con un botón de aceptar para guardar los cambios una vez hechos y escribirlos en el archivo de configuración. Una vez todo configurado podemos proceder a pulsar el botón de Iniciar que guardara todos los datos necesarios y se los cederá a los scripts para que empiece a funcionar, dando paso en orden a 3 funciones distintas; initializeDMD (), downloadMusic () y runDMD ().

4.3. Bloque 3 – Python Scripts

Código de referencia: dmd.py

El primer script que se lanza es initializeDMD (). Este script va a revisar la carpeta donde se encuentra alojado el programa en busca de archivos con la extensión '.webm' que es la extensión con la que la extensión de Pafy descarga el audio de las canciones de YouTube. Todos los archivos que encuentre los va a borrar para posteriormente proceder a descargar los nuevos archivos.

El siguiente script es downloadMusic () que como su nombre bien indica, se dedica a revisar el archivo de configuración y a descargar las canciones que sean necesarias. Además, según va descargando estas canciones va modificando los nombres de los archivos y guardándolos

en el archivo de configuración, añadiéndole un 0 al principio del nombre a cada archivo para saber que ese archivo ya ha sido tratado y controlado por el script.

Por ultimo nos encontramos con el más simple y a la vez más importante de los scripts; runDMD (). Este script inicia la comunicación puerto serie con el Arduino del teclado y entra en un bucle infinito revisando el buffer del puerto serie. Comparando los String que entran por el puerto serie con los que tenemos configurados, identificaremos que botón se ha pulsado y realizaremos determinadas tareas en consecuencia. Cada vez que se pulse una tecla el script la reconocerá e ira al archivo de configuración a leer la información de la misma. Este archivo de configuración guarda el tipo de botón que es (música, dado o recurso), la información del mismo y el nombre del archivo si es necesario. De esta tarea se encarga la función checkMode () que lee el archivo de configuración y lanza otras funciones en consecuencia.

Para la música tenemos la función playMusic () que primero matará la tarea anterior de VLC (programa necesario como reproductor multimedia por defecto para el correcto funcionamiento del programa) y luego abrirá el archivo dictado por el archivo de configuración (además de guardar este nombre como último archivo abierto para eliminar ciertos bugs).

En caso de que sea un dado tenemos la función de rollDice () que nos da un numero aleatorio en función del dado que queramos tirar y nos crea el String del formato DXXrZZ para mandarlo por puerto serie al Arduino.

Por ultimo tenemos la función openResource () que nos abre el archivo de la ruta que nosotros hayamos configurado. Es el más simple de todos.

Además, hay 3 funciones más relacionadas con el encoder, play_pause (), volumeUp () y volumeDown () que hacen específicamente eso. Son bastante similares entre sí; primero utilizan una extensión de Python llamada pygetwindow que se utiliza para agarrar la ventana que necesitemos y convertirla en ventana activa en función de su nombre, para más tarde pulsar los atajos del teclado respectivos mediante software gracias a la extensión keyboard de Python.

Código de referencia: config.py

En otro plano diferente encontramos los scripts para leer y escribir en el archivo de configuración. El primer script y el más importante es update (key, mode, data) que necesita los argumentos con la información de la tecla, el modo y el dato para escribirlos.

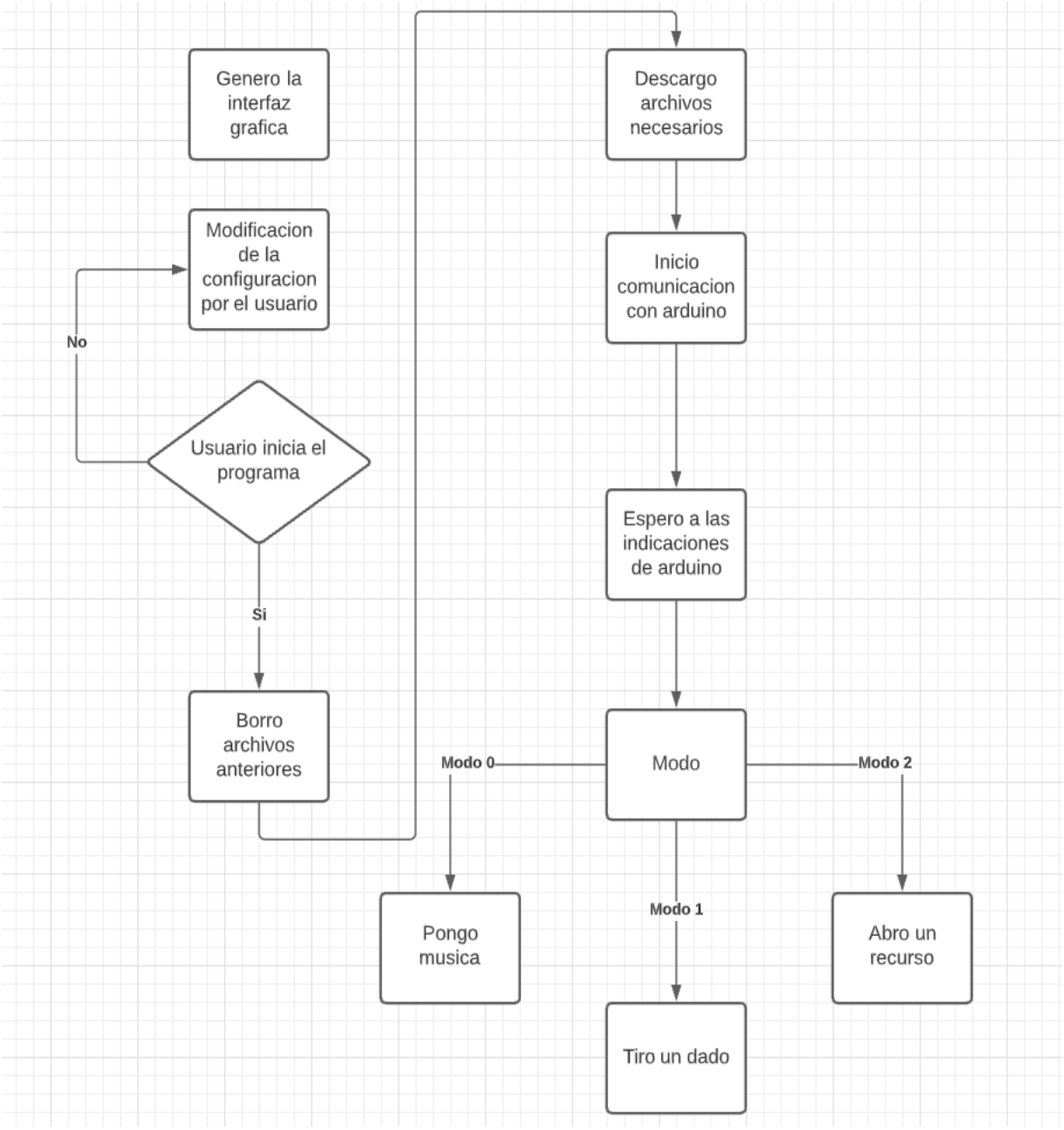
El siguiente script es read (key) que nos proporciona toda la información de una tecla en concreto.

Después tenemos updateName (key, name) que nos guarda el nombre del archivo descargado con el pyfy para poder agarrar la ventana o cerrar la tarea en un futuro.

Como último script tenemos latest_file (audioName) que nos guarda el nombre de la última canción activa para evitar bugs.

Código de referencia: setup.py

Por último, nos encontramos con un pequeño script de apenas 4 líneas que nada tiene que ver con el programa pero que es necesario para poder construir la versión ejecutable mediante la ayuda de la librería py2exe.

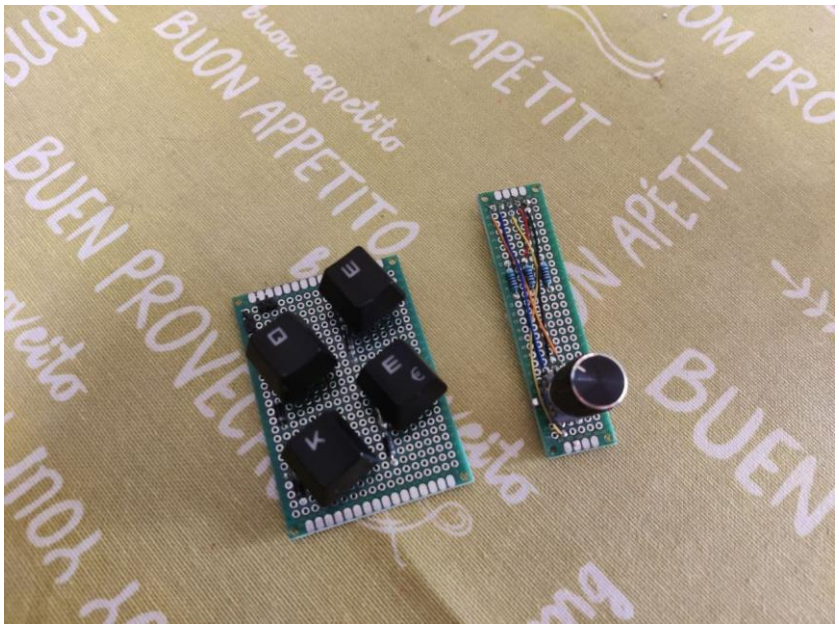


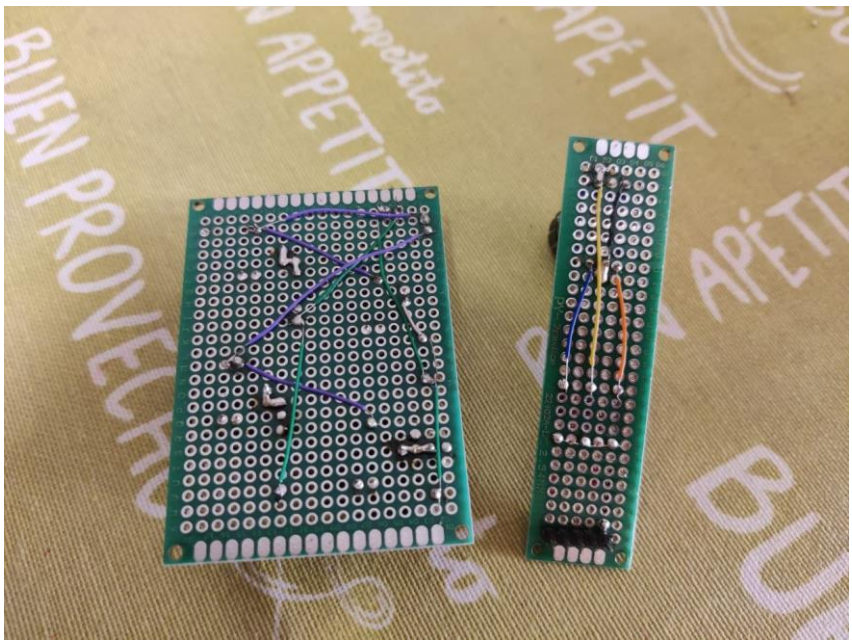
5. Experimentos y pruebas

A la hora de desarrollar el primer prototipo funcional (antes de desarrollar la pcb) he tenido que hacer ciertas pruebas y experimentos para asegurarme de que todo funcionaba y he construido varias "breakout boards" para poder hacer esto.

5.1. Prueba 1 - Botones y Encoder

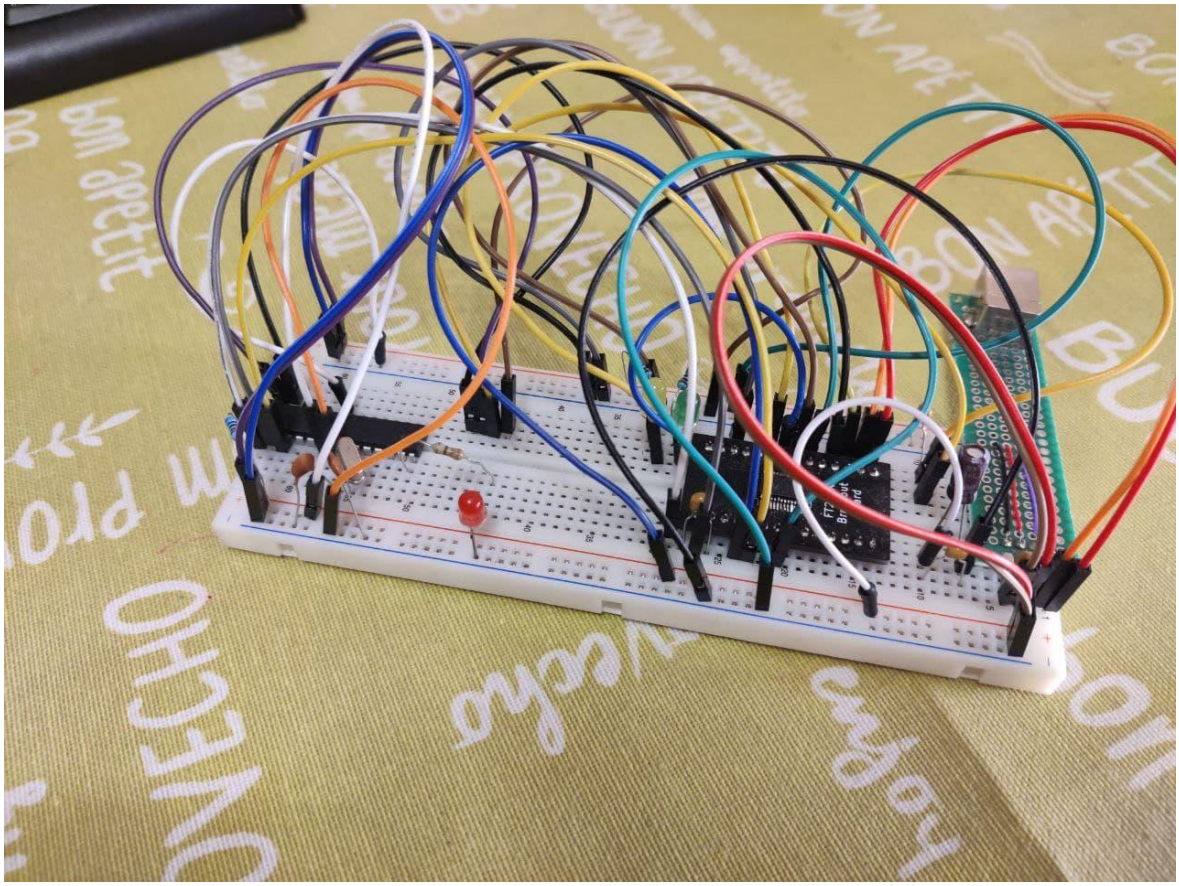
Todos estos componentes no se podían pinchar en una protoboard convencional así que acabe haciendo un par de placas de desarrollo para ambos periféricos y poder probarlos, el resultado fue satisfactorio y funcionaba tal y como esperaba.





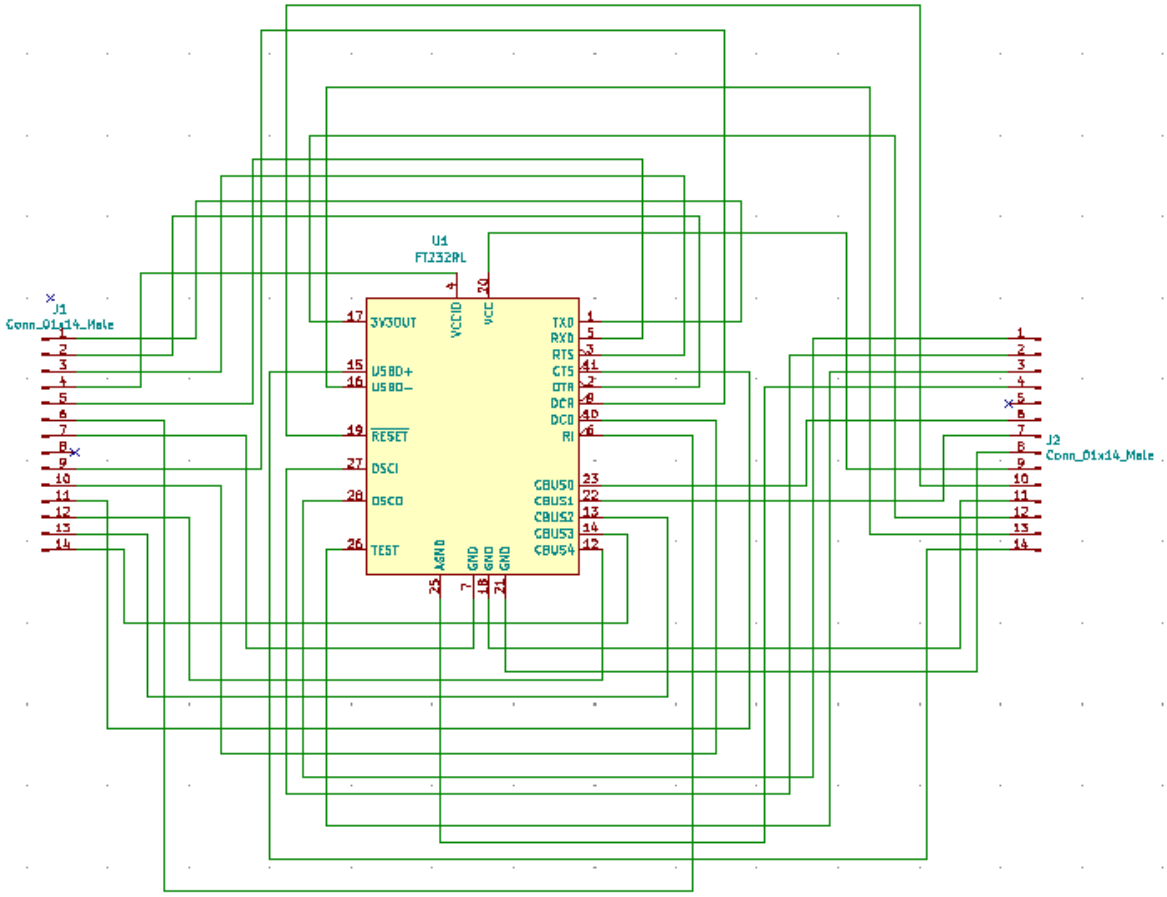
5.2. Prueba 2 - Hackduino

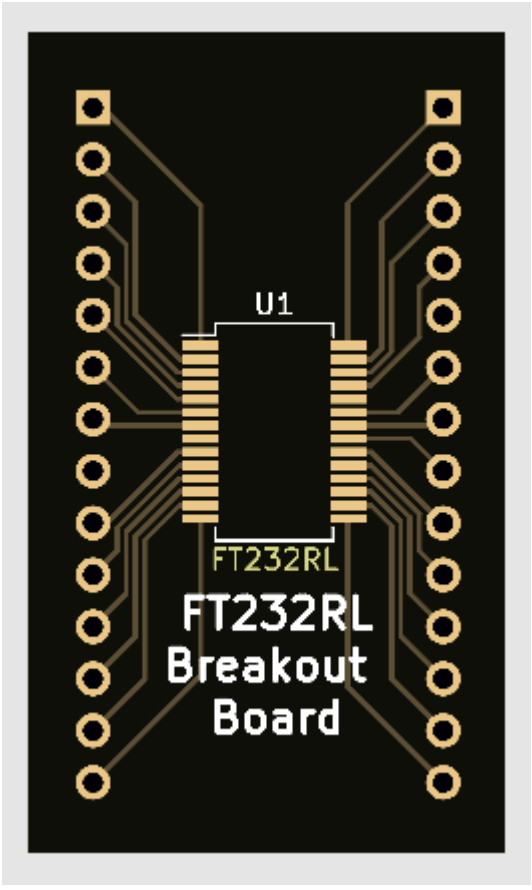
El hackduino es como se le llama comúnmente a un sistema basado en Arduino que no viene directamente de la propia empresa, en nuestro caso, construir un simple Arduino con interfaz UART para nuestro proyecto. Para poder usar el FT232 he tenido que crear una PCB para transformar sus patillas SMD a THT ya que este integrado solo se encuentra en formato SMD. Después tuve que montar un ATMega y un FT232 en una protoboard con todos sus componentes y conexiones necesarias junto con un puerto USB B también hecho en una breakout board hasta dar con el circuito que cumplía mis necesidades (el antes mencionado). Aquí una muestra de la placa de prototipo con el hackduino.

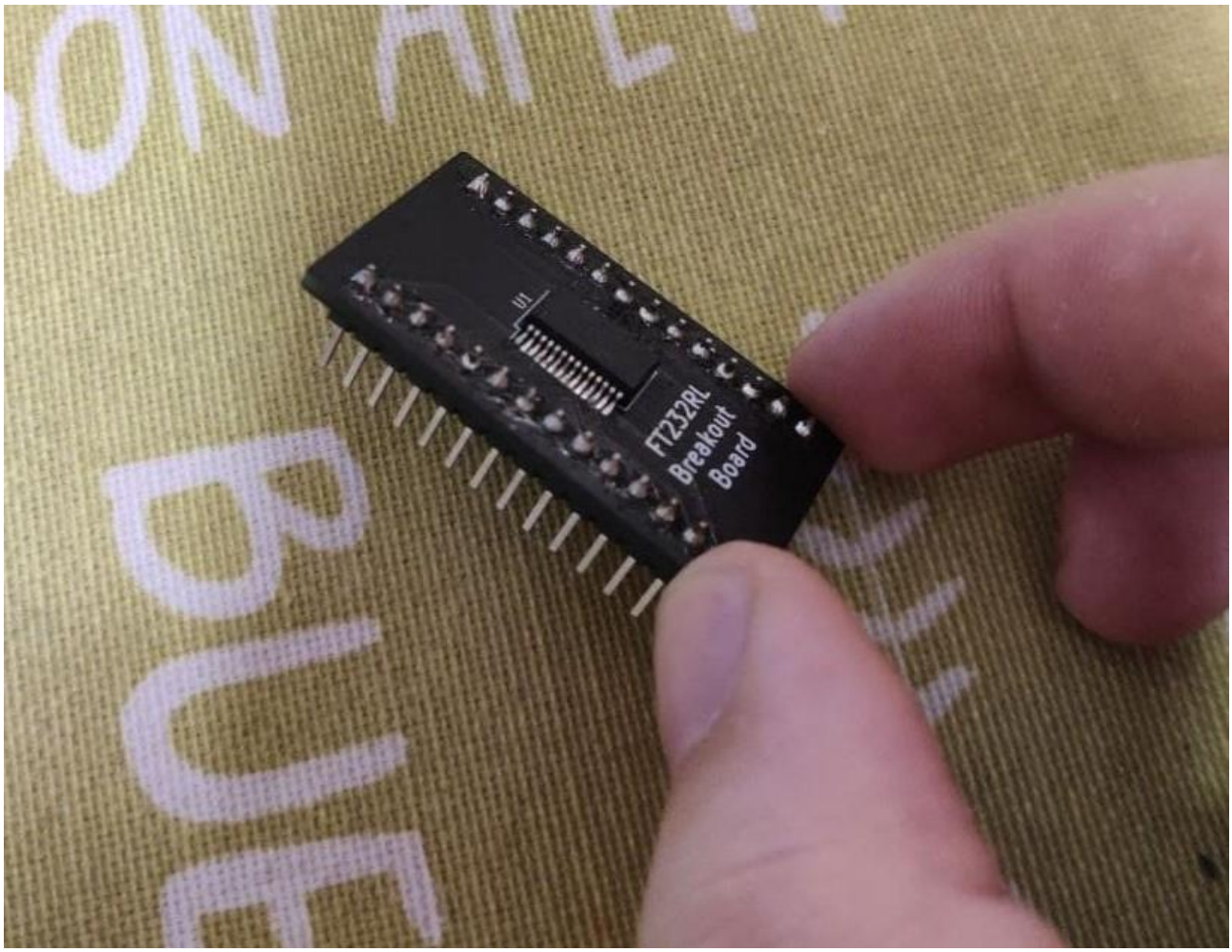


6. Planos y Esquemas

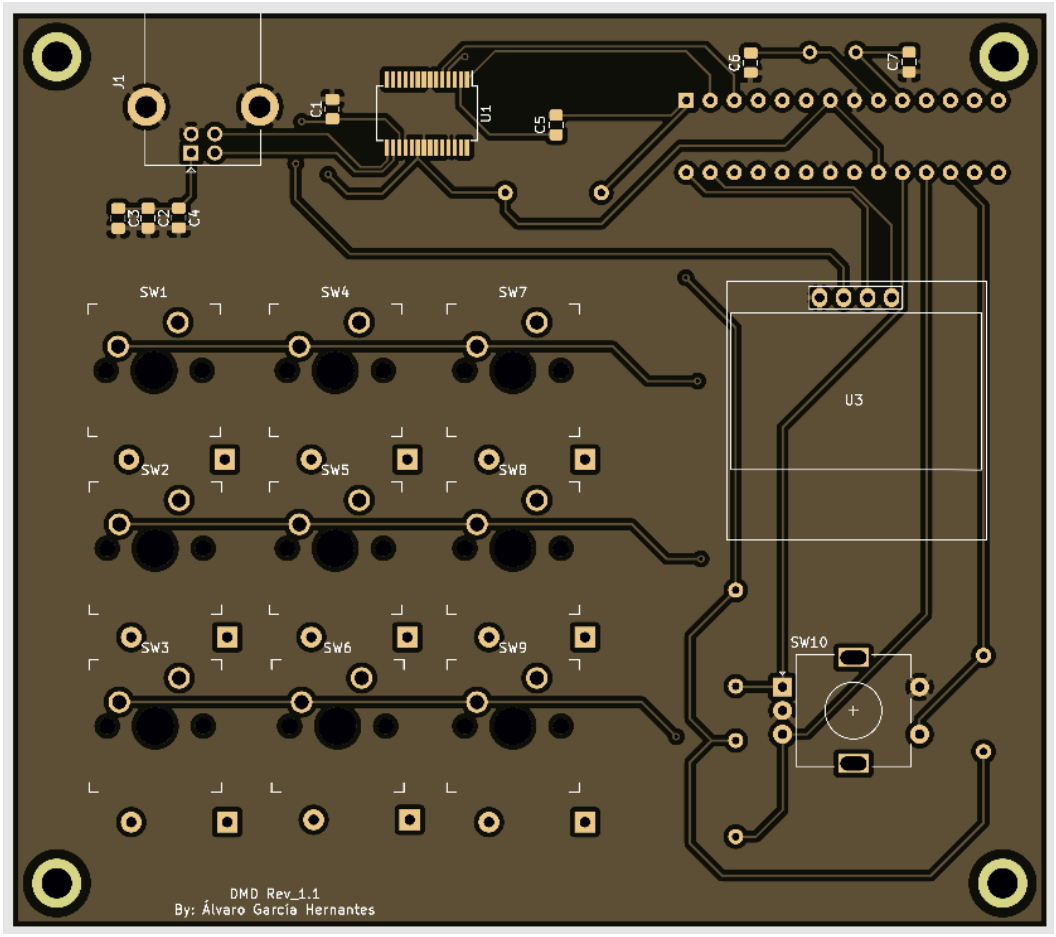
Como bien he mencionado antes, para la realización del prototipo y las pruebas tuve que desarrollar una PCB para el FT232 y poder utilizarlo en una protoboard, adjuntos captura del circuito, la PCB y la placa ya montada.

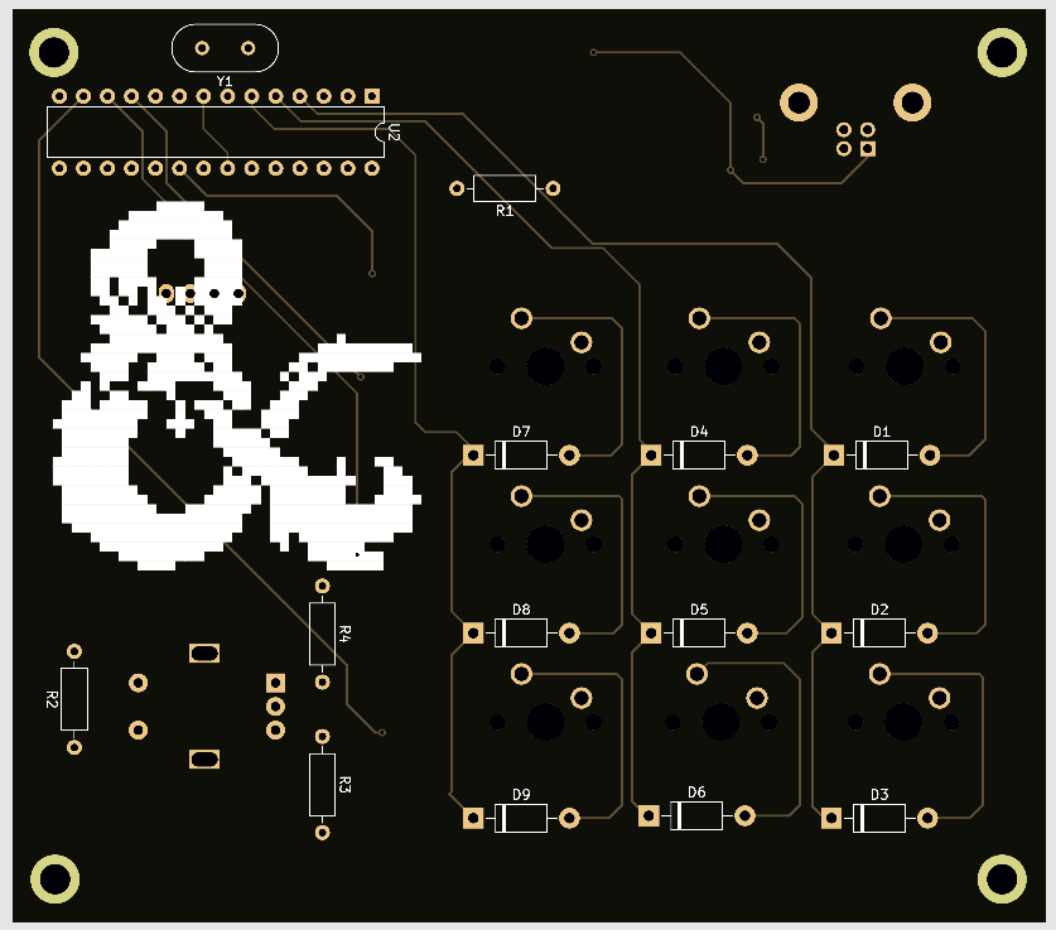

U1
FT232RL
J1
Conn_01x14_Male
3V3OUT
VCCIO
VCC
USBD+
USBD-
RESET
DSCI
DSCO
TEST
AGND
GND
TXD
RXD
RTS
CTS
DTR
DCR
DCD
RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
J2
Conn_01x14_Male


U1
FT232RL
FT232RL
Breakout
Board



Además, también he desarrollado una PCB final que incluía todo lo mencionado anteriormente y que se utiliza en el teclado. Ha habido 2 revisiones de esta PCB y aquí muestro la versión 1.1.


J1
C1
U1
C5
C6
C7
C4
C8
SW1
SW4
SW7
SW2
SW5
SW8
SW3
SW6
SW9
U3
SW10
DMD Rev_1.1
By: Álvaro García Hernantes


Y1
U2
R1
D7
D4
D1
D8
D5
D2
D9
D6
D3
R4
R2
R3

7. Anexos

Aquí se incluyen tanto los Datasheet de los componentes utilizados como una BOM a modo de guía. Todos estos archivos se encuentran en la Carpeta 'Anexos' de la documentación entregada.

7.1. Bill Of Materials (BOM)

Incluido en la carpeta mencionada anteriormente, el documento se llama DMD_BOM.xlsx.

7.2. Datasheet

Se incluyen los Datasheet del FT232 RL y del ATMega 328P entre otros.

7.3. Anexo III Plan de empresa

7.3.1. La idea de negocio.

Esta idea es muy simple y clara; llevar al mundo de los juegos de rol de mesa la tecnología para facilitar y mejorar las partidas.

7.3.2. Breve currículum de los promotores.

Se trata del archivo CV_GarciaHernantes en la carpeta de anexos.

7.3.3. Imagen corporativa (marca, logotipo, página web…).

7.3.4. Público objetivo al que se dirige el producto.

El target de mi producto es la comunidad de juegos de rol de mesa, una comunidad cada vez más amplia que el año pasado llego a superar los 40 millones de jugadores solo en DnD. Además, se calcula que el 39% de los jugadores son mujeres y que sobre un 40% son menores de 25 años.

A pesar de ser un producto enfocado a todos los jugadores de juegos de mesa, sí que es cierto que es probable que llame más la atención a los jugadores de 30 años o menores, ya que estos estarán más familiarizados con los videojuegos y la tecnología.

7.3.5. Análisis DAFO.

7.3.6. Modelo Lean Canvas

8. Bibliografía

https://stackoverflow.com/

https://www.arduino.cc/

https://forum.arduino.cc/

https://www.reddit.com/r/Python/

https://www.reddit.com/r/arduino/

https://www.reddit.com/r/KiCad/

https://www.reddit.com/r/MechanicalKeyboards/

http://smisioto.no-ip.org/