



UNIVERSIDAD  
**PABLO**<sub>de</sub>  
**OLAVIDE**  
SEVILLA

**TRABAJO FIN DE GRADO**

**STREAM DECK CON ARDUINO**

Código TFG: 21-22-A10

**OpenDeck**

Autor: Alejandro Govantes Pola

Tutor: Prof. Dr. Francisco Martínez Álvarez

Sevilla, abril de 2022



# ÍNDICE

## 1. INTRODUCCIÓN

- 1.1 Motivación
- 1.2 Procedimiento del plan de trabajo

## 2. ESTADO DEL ARTE

- 2.1 Introducción
- 2.2 Creación del contenido
- 2.3 OBS
- 2.4 Stream Deck de Elgato
- 2.5 Plataforma Arduino
  - 2.5.1 Placas E/S
  - 2.5.2 Módulos
  - 2.5.3 Accesorios
  - 2.5.4 Arduino IDE
- 2.6 Nextion
  - 2.6.1 Pantalla táctil Nextion
  - 2.6.2 Nextion Editor

## 3. COMPONENTES

- 3.1 Arduino Leonardo
- 3.2 Nextion 3.5" NX4832T035
- 3.3 Cable jumper
- 3.4 Cable USB a micro USB
- 3.5 Carcasa mediante impresión 3D

## 4. DESARROLLO

- 4.1 Ideas fundamentales
- 4.2 Panel de botones
  - 4.2.1 Edición de los botones
  - 4.2.2 Programación en Nextion Editor
- 4.3 Codificación en Arduino IDE
  - 4.3.1 Configuración previa del Arduino IDE
  - 4.3.2 Explicación del código

## 5. IMPLEMENTACIÓN

## 6. FUNCIÓN DE LOS BOTONES

## 7. TESTEO

## 8. PROGRAMA DE PLANTILLAS DEL OPENDECK

## 9. MANUAL DE USUARIO

## 10. CONCLUSIONES

## 11. BIBLIOGRAFÍA

## 12. CÓDIGO ARDUINO

# ILUSTRACIONES

1. Vista general de la herramienta OBS
2. Distintos apartados que componen OBS
3. Algunos de los atajos de teclado disponibles en los ajustes de la aplicación
4. Primer modelo de Stream Deck de Elgato
5. Versión 1.8.5 del IDE Arduino
6. Raspberry Pi 3 Model A+
7. Arduino Leonardo
8. Diagrama del Arduino Leonardo
9. Nextion 3.5" NX4832T035
10. Ilustración del cargador recomendado para el uso de la pantalla e imagen de la placa de prueba
11. Cables jumper
12. Cable USB a micro USB
13. Parte superior del modelo 3D de la carcasa del OpenDeck
14. Base del modelo 3D de la carcasa del OpenDeck
15. Materiales disponibles para impresión 3D en INNOVA3D
16. Vista general de GIMP
17. Archivo->Nuevo...
18. Opción de relleno y modificación del color asignado
19. Selecciono rectángulo, los clono y les doy el color y la posición deseados
20. Redimensión de los objetos y colocación dentro del marco
21. En este caso, añado un cuadrado y le cambio el color a verde
22. Panel de botones ya editados y colocados en la matriz
23. Vista general de Nextion Editor
24. Implementación de los 12 botones del OpenDeck
25. Botón de activación/desactivación de la webcam
26. Configuración de la pestaña Herramientas en Arduino IDE
27. Pines de la pantalla
28. Definición de los botones de acción
29. Asociación de los botones a su pantalla, id y nombre
30. Declaración de los eventos Touch
31. Funciones de los botones
32. Setup y loop
33. Botón de doble acción
34. Botón de estado único
35. Colocación de los jumpers en el Arduino Leonardo
36. Conexión mediante los jumpers entre el Arduino Leonardo y la pantalla Nextion
37. OpenDeck ya montado a falta de la carcasa 3D
38. Instalación de la microSD en la pantalla
39. Carga de datos en la pantalla
40. Pantalla mostrando el panel de botones cargado

41. Arduino y jumpers dentro de la carcasa
42. OpenDeck con la carcasa puesta
43. Orificio para conectar el OpenDeck a la corriente
44. Orificio para conectar la SD en la pantalla
45. Soportes para evitar el deslizamiento
46. Botón para activar o desactivar la webcam
47. Botón para apagar o encender la escena
48. Botón para comenzar o finalizar la grabación
49. Botón para comenzar o finalizar la retransmisión
50. Botón para activar o desactivar el sonido
51. Botón para activar o desactivar el micrófono
52. Botón para activar o desactivar la música
53. Botón para cambiar entre las distintas escenas
54. Vista general del programa con JFrame
55. Panel de botones de Streamer #2
56. Panel de botones de Profesor
57. Panel de botones de Productor TV
58. Galería de botones
59. Imagen para explicar la colocación de los jumpers en el Arduino Leonardo
60. Atajos de activación/desactivación de la WebCam

## **RESUMEN**

En este Trabajo Fin de Grado voy a desarrollar una herramienta que permite la interacción directa con el PC mediante una serie de botones implementados en una pantalla, haciendo posible de esa manera la utilización del software OBS sin tener que manipular directamente la computadora. Se ha realizado con el hardware Arduino Leonardo y una pantalla táctil de Nextion.

Lo primero que hice fue estudiar las distintas posibilidades que tenía el hardware de Arduino para conocer si podría realizar lo que tenía en mente. Una vez comprobé que efectivamente se podían cumplir las expectativas, me puse manos a la obra e investigué cuál sería la mejor opción para incorporar a dicho hardware una pantalla táctil que me permitiese interactuar correctamente con el sistema.

Una vez tenía ya definidos estos dos componentes, el siguiente paso consistía en configurar mediante el software Nextion Editor el panel de botones a mi gusto y, posteriormente, codificar con el software Arduino para que dichos botones realicen las funciones que yo deseo en el PC.

Cuando lo tenía todo correctamente configurado, lo puse a prueba en distintos dispositivos para comprobar que todo funcionaba tal y como se esperaba. Este proceso supuso una serie de cambios en el código y en el diseño de los botones para obtener un mejor resultado y, por tanto, una mejor experiencia de usuario.

## AGRADECIMIENTOS

Agradecer antes de nada al tutor que me ha acompañado durante todo este proyecto, Dr. Francisco Martínez Álvarez. Desde un primer momento tuve claro que tendría por su parte todo el apoyo y motivación necesarios para conseguir mi objetivo.

A mi familia por haberme permitido estudiar lo que más me gusta y por haber confiado en mí desde el primer momento. No me olvido de Fido.

A Celia por su paciencia y estar ahí siempre. Por ser mi mayor apoyo durante este proceso y hacer que todo lo complicado pareciese sencillo.

A mis amigos, que sin saberlo me han acompañado en mis peores momentos y gracias a los cuales me fué mucho más fácil salir adelante. Quiero nombrar a Rubén, Alberto, Manu, Selu, Pablo, Javi y Álvaro.

A Óscar Serradilla por su experiencia en el mundo de la creación de contenido, por ofrecerme su punto de vista y ayudarme a mejorar y perfeccionar las ideas.

A mis compañeros de la carrera, aunque a varios los conocí ya casi acabando pero todos ellos me han ayudado y me han hecho este camino más ameno. Pablo, Carlos, Jose, Patri, Jesús, Dani, Noelia, Oleg, Laura, María y Adrián.

A los profesores que me han enseñado a formarme como mejor han sabido para acabar siendo el ingeniero que tanto tiempo llevo deseando ser.

*Puedes ser lo que quieras, solo conviértete en lo que crees que podrías ser.*

**Freddie Mercury**



# 1. INTRODUCCIÓN

## 1.1 Motivación

En la actualidad nos encontramos en el momento más álgido de los creadores de contenido, también conocidos como streamers o youtubers. Dentro de esta moda existen una gran variedad de plataformas que permiten crear contenido pero, la más popular en cuanto a la realización de directos es Twitch. Los streamers que realizan sus retransmisiones desde esta plataforma usan el software OBS ya que es totalmente gratuito y, además, es muy sencillo de utilizar. Más adelante me extenderé más sobre este software y la plataforma citada anteriormente.

Algunas de las mayores motivaciones que me han llevado a realizar este trabajo son las siguientes:

- *El producto más popular (Stream Deck de la marca Elgato) no baja de los 120€ en su versión original. Posteriormente se puso a la venta una versión mini de 6 botones a 80€, precio muy superior al que me ha costado desarrollar mi propia herramienta de streaming.*
- *Al igual que principalmente ha sido pensado para su uso en OBS, puede ser utilizado en cualquier otra aplicación en la cual existan atajos de teclado. Se podrían asociar los botones del OpenDeck a dichos atajos para interactuar sin ningún tipo de problema con dicho software.*
- *Quería realizar un Trabajo Fin de Grado distinto a lo que se suele realizar en esta carrera. Se me ocurrió que algo más relacionado con el hardware y con un uso tan útil podría ser una buena opción para desarrollar mi trabajo. También teniendo en cuenta que podía hacerlo con la libertad de elegir el diseño y el tipo de estética que tendría el dispositivo, adaptarlo a mis gustos en definitiva.*

## 1.2 Procedimiento del plan de trabajo

Para la realización de este trabajo he seguido las siguientes fases:

- **Fase 1:** Estuve investigando durante bastante tiempo en las distintas herramientas utilizadas actualmente para la realización de vídeos y directos. El resultado final fue que claramente el producto de Elgato de Stream Deck tenía ocupado todo este mercado. Consideraba interesante fijarse en qué les hacía ser los mejores en esto para tomar ideas y mejorar algunas otras.
- **Fase 2:** Estudiar con qué tecnología desarrollar este proyecto. Mis dos principales opciones fueron Arduino o Raspberry Pi, debido al precio de ambas y a las innumerables posibilidades que ofrecían una y otra. Posteriormente trataré con mayor profundidad por qué me acabé decantando por la primera de las opciones.
- **Fase 3:** Una vez encargué el Arduino Leonardo y la pantalla Nextion, era el momento de ponerse a trabajar en el diseño de los botones. Utilicé el software Nextion Editor para personalizar la disposición de los mismos y el aspecto que iban a tener.
- **Fase 4:** Una vez tenía la matriz de botones creada y cargada en la pantalla, comencé a codificar en Arduino para que dichos botones realizaran las acciones que yo deseaba. Tras la realización del código, tocaba unir el Arduino con la pantalla táctil para poder probar que todo funcionase correctamente.
- **Fase 5:** Configuré el OBS para que la pulsación de los botones completara una determinada acción en el software. Una vez estaba todo listo, probé el dispositivo y solucioné algunos errores.
- **Fase 6:** Tras todos los pasos anteriores, ya quedaba la realización de esta memoria del proyecto.

## **2. ESTADO DEL ARTE**

### **2.1 Introducción**

Para la correcta realización de este trabajo he realizado una investigación bastante amplia sobre el mundo de la creación de contenido y las distintas tecnologías y dispositivos que se manejan en la actualidad. Era importante conocer bien en qué estoy trabajando y qué es lo que se busca exactamente en todo esto.

El segundo paso era investigar sobre el hardware que podría utilizar para obtener mi objetivo pensando en un bajo presupuesto y buscando la máxima cantidad de posibilidades al mismo tiempo.

### **2.2 Creación de contenido**

Es cierto que Youtube existe desde el año 2005, pero podríamos decir que hasta 2009-2010 no comenzó realmente a cambiar el panorama hasta convertirse en lo que conocemos hoy en día. En sus inicios, esta y otras plataformas pensadas para la realización de vídeos se utilizaban únicamente para contenidos multimedia muy básicos.

Más tarde esos vídeos se convirtieron en algo totalmente diferente, algunos de ellos contaban con una producción detrás a la altura de grandes empresas destinadas al entretenimiento. Tanto es así que desde los primeros años en que todo esto comenzó a crecer, los creadores de contenido más populares contaban con un sueldo proporcionado tanto por la propia plataforma como por alguna empresa destinada al patrocinio y gestión de este tipo de usuarios.

Algunas de estas empresas fueron: Machinima, ZoomingTV, Freedom!, Thoughtful Media o el propio Youtube. Muchas de estas compañías obtuvieron una gran cantidad de popularidad y de beneficios en los años de mayor éxito de esta plataforma. Algunas como Machinima ya no existen debido a que muchos de los creadores de contenido comenzaron a ganar dinero sin necesidad de contar con ninguno de estos partners.

La historia de los partners dejó de tener tanta importancia entre la mayoría de creadores desde hace unos 5 años por lo comentado anteriormente. Es cierto que no se han extinguido por completo estas empresas ya que muchos canales de gran relevancia siguen asociados a algún partner, pero suelen ser casos especiales como músicos o creadores relacionados con el mundo del cine. Ahora el partner más utilizado es el propio que ofrece Youtube ya que es el más seguro en cuanto a tardanza de pagos y a los beneficios obtenidos.

En España, Youtube comenzó a tener éxito a partir de 2011 gracias principalmente a los Youtubers que dedicaban sus vídeos al gameplay. El gameplay consiste tal y como su propio nombre indica, en vídeos que se realizan jugando a videojuegos. Algunos de los creadores que despertaron el interés del público español en esta plataforma fueron Willyrex, Outconsumer o RicharBetaCode.

Hoy en día, el consumo de videojuegos es un mercado importante en el mundo del entretenimiento, generando más dinero que las películas y la música juntas. Esto hace que esta industria naciente sea perfecta para el comercio. Justin Kan, Emmett Shear, Michael Seibel y Kyle Vogt probablemente no lo pensaron cuando fundaron Justin Tv a principios de 2007, de donde nació la actual plataforma Twitch.

Twitch se creó durante el frenesí de Justin.tv. Se trataba de un servicio de video con muchos géneros diferentes. De repente, sus creadores reconocieron el crecimiento del contenido relacionado con los videojuegos y decidieron destinarlo a una plataforma dedicada.

Así nació Twitch, la plataforma de streaming en directo propiedad de la filial de Amazon, Twitch Interactive. Se lanzó en junio de 2011 como una alternativa a la plataforma de streaming Justin.tv. Twitch se centra principalmente en la transmisión de videojuegos y en ayudar al crecimiento de los deportes electrónicos(esports). Su popularidad superó a su hermano Justin.tv en octubre de 2013 con 45 millones de usuarios únicos, y en febrero de 2014 fue reconocida como la cuarta fuente de tráfico de Internet en los Estados Unidos de América, todo eso dió lugar a que en agosto de 2014 Justin.tv cerrase.

Como se indica directamente en su sitio web, la misión de Twitch es: "Construir la plataforma de video comunitaria líder en el mundo y también crear comunidades para juegos, mejorar la cultura de los videojuegos y de las artes creativas".

Según sus propios datos internos, la edad media de los usuarios de Twitch ronda los 21 años, el 99% de ellos ve videojuegos en directo, el 61 % visualiza los streamings chateando con la comunidad, el 38 % ve otro tipo de vídeos y el 25 % hace streaming de su propio juego. Todo esto da como resultado que el 58% de los usuarios pasan más de 20 horas al mes en Twitch, una métrica asombrosa que demuestra la importancia de este nuevo tipo de entretenimiento.

Hay varios servicios de suscripción disponibles en Twitch, como Twitch Prime y Turbo. Pulse, la red social de juegos, también ha estado en Twitch desde la primavera de 2017. Hay muchas formas de monetizar todas las transmisiones en tu canal de Twitch. Los afiliados y socios pueden monetizar sus videos a través de anuncios, suscripciones de usuarios o contenido patrocinado por la marca en su canal, que aparece en los espacios publicitarios del streamer. La comunidad también puede contribuir a través de donaciones.

Algunas de las estadísticas más relevantes sobre esta plataforma en el año 2021 fueron las siguientes[1]:

- Twitch cuenta con más de 26 millones de usuarios registrados.*
- Más del 33% de los usuarios registrados realizan una retransmisión al mes.*
- Más del 40% de los usuarios tiene una edad entre 16 y 24 años, dato importante con el que me extenderé más tarde.*
- Existen más de 50.000 partners de Twitch en la actualidad.*
- Los ingresos publicitarios de la plataforma ascendieron a los 800 millones de dólares.*

Los creadores de contenido más populares antes y, especialmente durante la cuarentena debida al Covid-19, comenzaron a trasladar su contenido de Youtube a Twitch. Este cambio se debió al descontento y el desgaste de muchos de estos creadores con la anterior plataforma líder. En Twitch encontraron esa novedad que les permitía estar en contacto directo con su público, el cuál agradeció estas retransmisiones en vivo durante esos momentos en los que nos vimos obligados a permanecer encerrados en casa por motivos de seguridad.

Rápidamente esta plataforma de origen estadounidense comenzó a introducir mejoras que permitían una mayor interacción entre el streamer y su audiencia. Entre esas mejoras, las más llamativas desde un punto de vista económico, serían las donaciones y las suscripciones mensuales. Mediante las donaciones, la audiencia agradece a sus streamers favoritos el esfuerzo y el trabajo realizados. En cuanto a las suscripciones, pasa algo similar a las plataformas de streaming más populares (Netflix, HBO, Amazon Prime, etc.), los usuarios comenzaron a suscribirse para estar al tanto de todo el contenido que se hacía y para contar con ciertos beneficios en sus canales preferidos.

Actualmente, Twitch se puede considerar sin ningún tipo de duda en un rival para las televisiones y para otras plataformas de streaming como las mencionadas anteriormente. Tanto es así que algunas de las mayores audiencias registradas en el último año en esta plataforma han alcanzado los siguientes números:

- **Ibai (España)** - 1.538.645 espectadores
- **elxokas (España)** - 1.185.959 espectadores
- **TheGrefg (España)** - 1.066.622 espectadores
- **Riot Games (Estados Unidos)** - 854.781 espectadores
- **dota2ti\_ru (Rusia)** - 812.818 espectadores

Sorprende que muchos de los más escuchados sean españoles o, al menos, hispanohablantes. Algunas de estas audiencias superan el 10% de Share si hablamos en términos televisivos, un número muy cercano a las dos cadenas más vistas de la televisión española en la actualidad (Antena 3 y Telecinco). Esto

posicionaría a varias de estas retransmisiones por encima del resto de cadenas españolas durante este último año. Es por eso que muchas de estas cadenas han comenzado a mostrar interés en esta plataforma y en este tipo de contenido ya que encuentran un mundo de posibilidades dentro del mismo.

Una de las características que hacen que esta plataforma esté compitiendo directamente con las cadenas de televisión es que cuenta con todo tipo de streamers. Dentro de las distintas categorías de directos más vistas se encuentran los Just Chatting, en los que el streamer simplemente está hablando e interactuando con sus seguidores. También destacan los contenidos relacionados con los videojuegos, con la música o el arte. Sin duda estamos ante una “nueva” televisión en la que podemos encontrar cualquier tipo de contenido de forma totalmente gratuita.

Esta ha sido una de mis grandes motivaciones a la hora de realizar este trabajo, se trata de una herramienta que puede ser utilizada por los creadores de contenido en una plataforma que actualmente es la líder absoluta en este tipo de contenido.

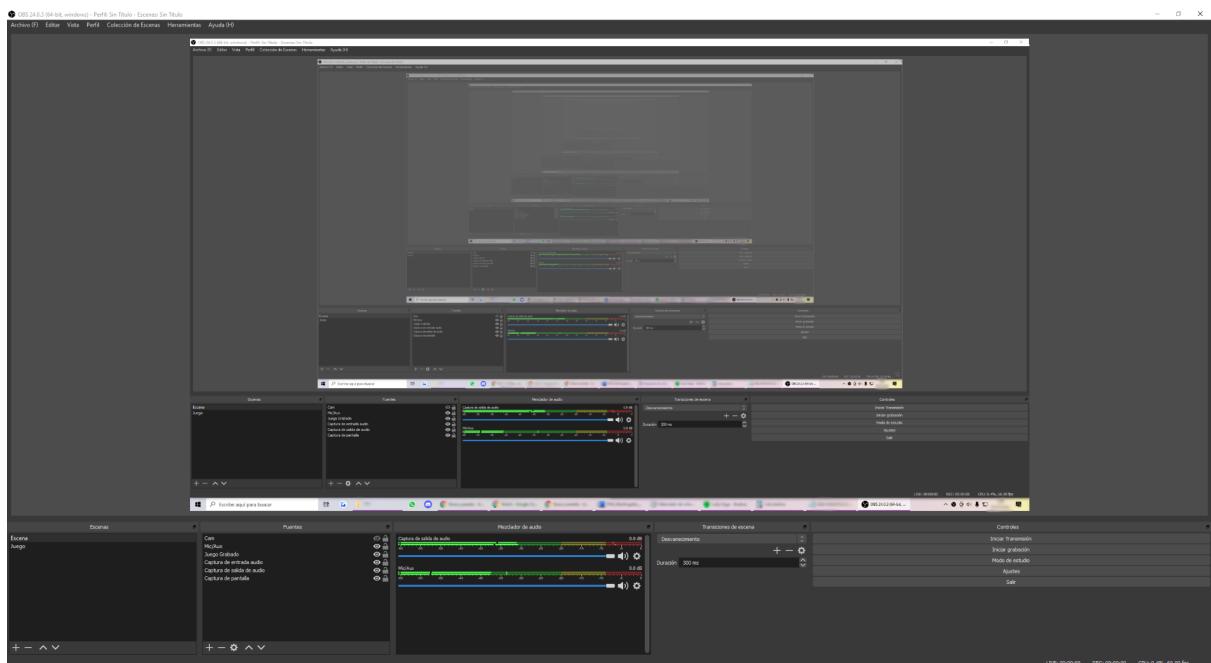
### 2.3 OBS

Es una aplicación libre y de código abierto utilizada para grabar y transmitir vídeo mediante streaming. Fue lanzada en 2012 y está programada en C y C++. Actualmente sigue recibiendo actualizaciones todos los meses y cuenta con más de 40 idiomas.

Es considerada la herramienta líder en este sector debido a que es gratuita y resulta muy sencilla de utilizar. Ya que es de código abierto, puede modificarse y adaptarse a las necesidades de cada uno casi sin límite.

Permite la grabación de vídeo y la retransmisión en directo del mismo desde la propia aplicación con tan solo unos clicks. Lo que me parece muy interesante de

este software es que cualquier persona con interés puede desde su casa ser director, productor o realizador mediante el uso de esta app.

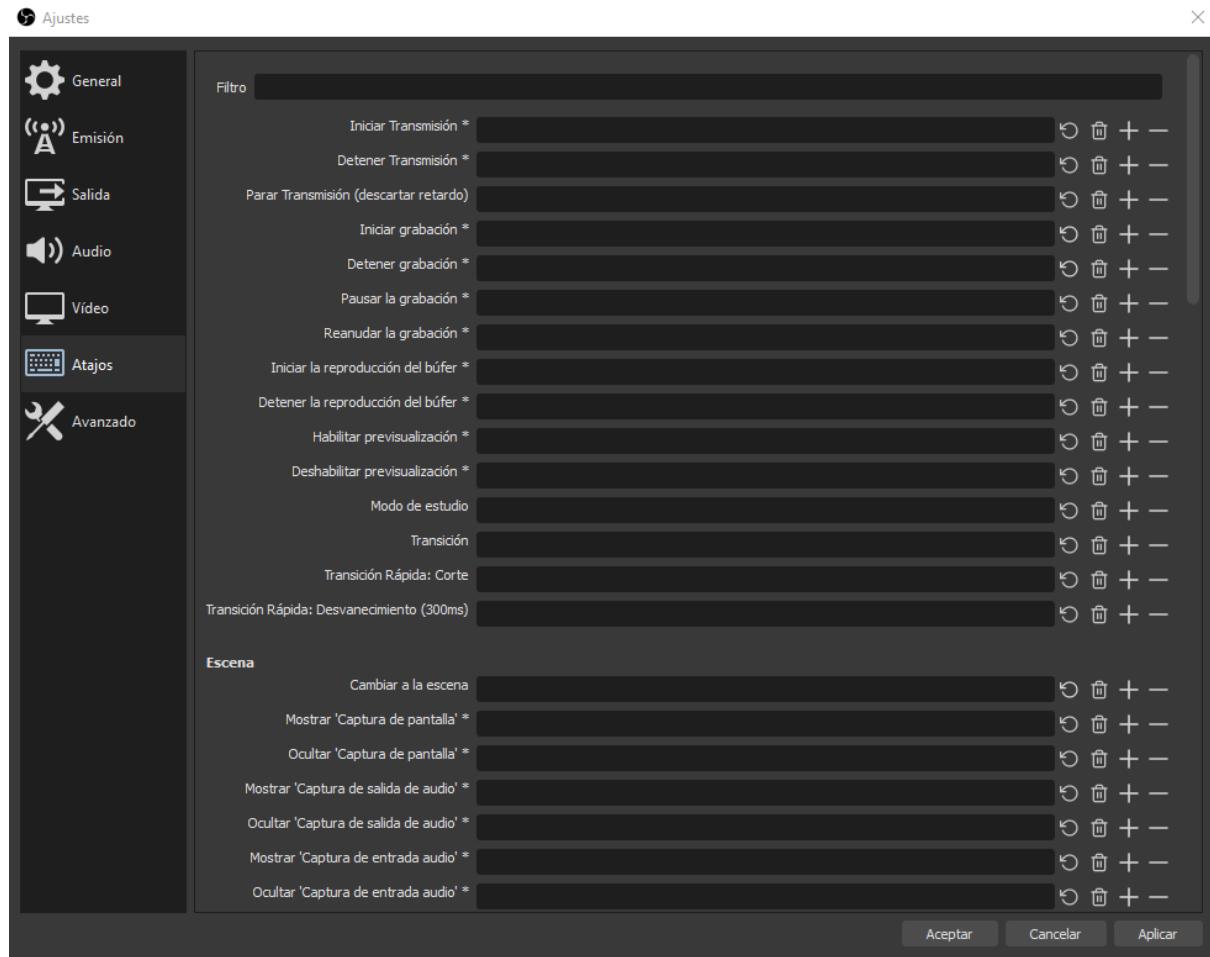


1. Vista general de la herramienta OBS



2. Distintos apartados que componen OBS

Ahí es donde entra en acción mi proyecto, dentro de las infinidad de posibilidades que ofrece OBS se encuentran las funcionalidades más usadas y básicas como el control del sonido, el micrófono o el cambio de planos. Todas estas funciones pueden asociarse mediante los atajos de teclado integrados en la aplicación a nuestros propio teclado para hacer más sencillo el manejo y manipulación de todos estos parámetros.



### *3. Algunos de los atajos de teclado disponibles en los ajustes de la aplicación*

Estos atajos se utilizan con mucha frecuencia ya que resultan muy cómodos para alguien que necesita constantemente estar cambiando parámetros de OBS. Algunos ejemplos pueden ser el cambio entre escenas, activar/silenciar el micrófono, activar/desactivar la webcam, etc.

Lo que hacen los dispositivos más conocidos es presentar una matriz de botones mediante los cuales, estos cambios comentados anteriormente pueden llevarse a cabo con la sola pulsación de una tecla. Esto permite no tener que estar todo el tiempo accediendo a la aplicación ni estar tocando el teclado o el ratón.

Resulta de gran utilidad para los creadores de contenido que cuentan con una gran cantidad de escenas (que son la mayoría) y que realizan retransmisiones casi a diario. Evidentemente todas estas aplicaciones se pueden llevar a otros terrenos según lo que se busque hacer, hablaré sobre eso en posteriores apartados.

## 2.4 Stream Deck de Elgato

Tal y como comentaba en el apartado 1.1, la tecnología más utilizada para la interacción con el software OBS es el dispositivo Stream Deck de la empresa alemana Elgato. El primer dispositivo y, el que se hizo más popular, consiste en una matriz de quince botones mediante los cuales se busca facilitar y agilizar el manejo de la aplicación durante la grabación y retransmisión de los vídeos.



4. Primer modelo de Stream Deck de Elgato

Como ya he comentado, existe un gran inconveniente para aquellas personas a las que les interesen las funcionalidades de este dispositivo, su precio. A continuación voy a mostrar un listado de las versiones más vendidas de Stream Deck:

- ❖ **Primera versión (15 botones)** | 120-150 euros
- ❖ **Versión mini (6 botones)** | 80 euros
- ❖ **Versión renovada (15 botones)** | 150 euros
- ❖ **Versión XL (32 botones)** | 250 euros

Tal y como decía, estos precios resultan excesivos para alguien que no obtenga un beneficio lo suficientemente alto con la creación de contenido o para personas que simplemente se toman como un hobbie todo esto.

Reconozco que los precios son bastante elevados pero también entiendo la utilidad que pueden tener para aquellos que le puedan dar un uso muy frecuente. Sin duda, si buscas algo que te ayude a la hora de realizar tus vídeos o directos, con esta opción no puedes equivocarte. Elgato es una empresa que lleva muchos años introducida en este mundo tanto por la venta de sus famosas capturadoras de vídeo, sus cromas, sus Stream Deck y otros muchos periféricos que han ido sacando al mercado debido al creciente éxito de la marca.

Una vez he investigado lo suficiente sobre la empresa número 1 en este tipo de dispositivos y he indagado en los intereses de los streamers en la actualidad, estoy preparado para comenzar a buscar las mejores opciones para poder realizar mi propio Stream Deck, en mi caso se llama OpenDeck.

## 2.5 Plataforma Arduino

Arduino es una empresa de desarrollo de hardware y software libre. Consta de una comunidad internacional que diseña y fabrica placas de desarrollo de hardware para crear dispositivos digitales e interactivos que detectan y controlan objetos del mundo real. Arduino se enfoca en introducir y promover el uso de la electrónica y la programación de sistemas embebidos en proyectos multidisciplinarios. Los productos vendidos por la empresa se distribuyen como hardware y software gratuitos, bajo la Licencia pública general de GNU (GPL) y la licencia pública general reducida de GNU (LGPL), lo que permite a cualquier individuo fabricar placas Arduino y distribuir el software. Las placas Arduino están disponibles en el mercado como placas ensambladas o en forma de kits.

Estrictamente hablando, el proyecto "Arduino" comenzó en 2005 como un proyecto centrado en los estudiantes del Instituto Ivrea de Diseño de Interacción (IDII) en Italia. En esos años, los estudiantes usaban microcontroladores BASIC Stamp, que

costaban 100 dólares, un precio bastante elevado para el estudiante promedio. Previo al 2005, y especialmente durante el 2003, Hernando Barragán creó la plataforma de desarrollo Wiring en el proyecto de tesis de maestría del IDII, bajo la supervisión de Massimo Banzi y Casey Reas, quienes son conocidos por trabajar con lenguajes de procesamiento. El objetivo de este proyecto es crear herramientas sencillas y de bajo coste para la elaboración de proyectos digitales por parte de personas sin altos conocimientos técnicos o perfiles de ingeniería. El proyecto Wiring es una placa de desarrollo de hardware que incluye una placa de circuito impreso (PCB) con un microcontrolador ATmega168, un entorno de desarrollo integrado (IDE) basado en procesamiento y una biblioteca de funciones para facilitar la programación del microcontrolador. En 2005, Massimo Banzi trabajó con David Mellis (otro estudiante del IDII) y David Cuartielles para agregar soporte de cableado al microcontrolador ATmega8, que era más económico que el microcontrolador original (Atmega168). Pero en lugar de seguir desarrollando Wiring, se separaron del proyecto y lo rebautizaron como Arduino.

### 2.5.1 Placas E/S

En la actualidad, existen una gran variedad de placas Arduino con las que podemos desarrollar infinidad de proyectos. Los principales modelos voy a comentarlos a continuación[2]:

- **Arduino UNO:** utiliza el microcontrolador ATmega328 para el manejo de USB en lugar del utilizado en generaciones anteriores. Esto hace que posea un ratio de transferencia mucho más rápido y con más memoria. No necesita de driver para Linux o Mac. Detalles técnicos: voltaje de 7-12V, 14 pines digitales E/S, 6 entradas analógicas, 32k de memoria Flash y reloj de 16MHz de velocidad.
- **Arduino Leonardo:** esta placa se diferencia de otras que usan un chip USB-Serial separado en el microprocesador tiene comunicación USB incorporada, esto elimina la necesidad de un procesador secundario. Detalles técnicos: voltaje de funcionamiento 5V, voltaje de E/S de 7-12V, 20 pines

digitales E/S, 12 entradas analógicas, 32k de memoria Flash y reloj de 16MHz de velocidad.

- **Arduino DUE**: es la primera placa Arduino basada en un microcontrolador de núcleo ARM de 32 bits, es la placa adecuada para proyectos potentes a gran escala. Detalles técnicos: 54 pines digitales E/S, 12 entradas analógicas.
- **Arduino Yún**: es una placa pensada para combinar la potencia de Linux junto con la sencillez característica de Arduino. El chip está conectado al módulo Linux, lo que hace muy sencilla la comunicación entre ambos y permite delegar procesos pesados a la máquina Linux integrada en la placa. Detalles técnicos: microprocesador Atheros AR9331, alimentación de 3.3V, memoria RAM de 64 MB DDR2, memoria Flash de 32MB.
- **Arduino TRE**: primera placa Arduino fabricada en los Estados Unidos. Gracias a su procesador Sitara AM335x de 1GHz, se obtiene hasta 100 veces más rendimiento con el TRE que con Arduino Uno o Leonardo. Detalles técnicos: memoria Flash de 32KB, 14 pines digitales E/S, 7 entradas analógicas.
- **Arduino Micro**: basada en el microcontrolador ATmega32U4, y fue desarrollada en conjunto con Adafruit. Gracias a su tamaño, es muy fácil de colocar a la hora de montar nuestro proyecto. Detalles técnicos: 20 pines digitales E/S, 12 entradas analógicas, posee un micro USB que le permite no necesitar de un procesador secundario.
- **Arduino ROBOT**: se trata de la primera placa de Arduino sobre ruedas. En este caso, consta de dos procesadores en cada una de sus dos placas. Una de ellas controla los motores y la otra lee los sensores y decide qué hacer. Ambas pueden ser programadas con el IDE de Arduino y sus respectivos microcontroladores están basados en el ATmega32U4. Cuenta con numerosos pines asignados a los sensores y su programación es similar al proceso necesario con la Arduino Leonardo.

- **Arduino Esplora:** es una placa de microcontrolador que deriva del Arduino Leonardo. A diferencia de todas las placas Arduino anteriores, Esplora ofrece muchos sensores integrados listos para usar en la interacción. Está diseñado para personas que quieren comenzar con Arduino sin aprender primero la electrónica. Esplora tiene salidas de luz y sonido integradas, así como una variedad de sensores de entrada, incluidos joysticks, controles deslizantes, sensores de temperatura, acelerómetros, micrófonos y sensores de luz. También tiene el potencial de expandir sus capacidades con dos conectores de entrada y salida Tinkerkit y un zócalo para una pantalla LCD TFT a color. Detalles técnicos: voltaje de funcionamiento de 5V, memoria Flash de 32k, frecuencia de reloj de 16 MHz.
- **Arduino Mega:** está basada en ATmega2560. Es una de las placas con más éxito de todas, por lo tanto, si necesitas inspiración para alguno de tus proyectos seguro que puedes encontrar mucha ayuda que te puede ser de gran utilidad. Detalles técnicos: consta de 54 pines de E/S, 16 entradas analógicas, 4 puertos seriales de hardware, una frecuencia de reloj de 16MHz y una conexión USB.
- **Arduino Ethernet:** posee un puerto ethernet y está basado en Arduino Uno. Permite la conexión a la red o a Internet.
- **Lilypad Arduino:** de entre todas las placas existentes, puede que esta sea la más curiosa de todas. Es una placa diseñada para poder ser llevada en la ropa o en textiles electrónicos. La idea es que pueda ser cosida a la tela y, de manera similar, montarse con baterías, sensores y dispositivos de interacción con hilo conductor.
- **Arduino Nano:** tal y como su propio nombre indica, se trata de una placa muy pequeña. Eso no impide que sea completa y compatible con otras placas de prueba basadas en ATmega328. Tiene una funcionalidad bastante similar al Arduino Duemilanove, pero con un paquete distinto. Solo carece de conector de alimentación CC y funciona con un cable USB Mini-B en lugar de un cable USB estándar. Detalles técnicos: voltaje de funcionamiento de 5V, 8

pines de entrada analógica, memoria Flash de 32k, 22 pines digitales de E/S y con un peso de tan solo 7 gramos.

- **Arduino Pro:** esta placa está basada en el ATmega168 o bien en el ATmega328.
- **Arduino Pro Mini:** evidentemente está basada en su versión estándar, tiene un ATmega168.
- **Arduino Fio:** es perfecta para proyectos inalámbricos ya que cuenta con un zócalo para un módulo XBee y puede ser alimentado directamente por una batería LiPo. Es compatible con Funnel e incluye un circuito de carga a través de USB. Posee 14 pines digitales de E/S, es compatible con baterías de polímero de litio, tiene un reset button, un On/Off Switch y Status/Charge/ RSSI LEDs.

### 2.5.2 Módulos

Debido a la popularidad de las placas Arduino, existen muchísimos módulos compatibles o que han sido directamente diseñados para estas placas. Voy a destacar algunos de los módulos más conocidos dentro del mercado:

- **Arduino GSM Shield.** conecta su Arduino a Internet mediante la red inalámbrica GPRS. Simplemente se debe instalar este módulo a una placa Arduino compatible, conectar una tarjeta SIM de un operador que ofrezca cobertura GPRS y tras seguir las instrucciones, comenzará a tener acceso a la red. También permite hacer y recibir llamadas de voz utilizando el conector de audio y micrófono integrado. Además permite enviar y recibir mensajes SMS.
- **Arduino Ethernet Shield:** permite la conexión a Internet de la placa Arduino. Tan solo necesita ser conectado a la placa Arduino, conectarse a la red y

seguir las instrucciones correspondientes para que todo comience a funcionar.

- **Arduino WiFi Shield:** módulo que permite la conexión a Internet de forma inalámbrica.
- **Arduino Wireless SD Shield:** permite agregar fácilmente comunicación inalámbrica a su Arduino usando los módulos XBee. También se puede utilizar con Wifi 802.11 b/g y el RN-XV WiFly disponible en productos relacionados. Tiene una ranura para una tarjeta microSD y podemos usar la biblioteca SD para almacenar y recuperar datos. Si se usa, el pin CS no se podrá usar, ya que lo usará la biblioteca. También contiene un práctico interruptor que nos permite elegir si comunicarnos con el módulo o la placa vía USB sin utilizar incómodos jumpers.
- **Arduino Wireless Proto Shield:** permite que la placa Arduino se comunique de forma inalámbrica utilizando el módulo inalámbrico. Puede usar cualquier módulo del mismo tamaño de XBee. El módulo puede comunicarse hasta 30 metros en interiores o 90 metros en exteriores. Se puede usar como un reemplazo serie-USB o puede ponerlo en modo de comando y configurarlo para varias opciones de redes broadcast y mesh. El módulo divide cada pin del XBee en almohadillas y no tiene toma SD. Posee un interruptor integrado que permite al módulo inalámbrico la comunicación con un microcontrolador o con el convertidor de USB a serie.
- **Arduino USB Host Shield:** es un módulo que puede ser montado a las placas Arduino Uno, Dualmilaneo y Mega para que funcionen como un USB Host y, de esa manera, puedan controlar dispositivos esclavos como pueden ser memorias USB, teclados, ratones, etc.
- **Arduino Motor Shield:** está basado en L298, que se trata de un controlador dual de puente completo diseñado para controlar cargas inductivas como relés, solenoides y motores. Permite manejar dos motores DC con tu placa Arduino, controlando la velocidad y la dirección de cada uno de forma

independiente. Además, permite medir la absorción de corriente del motor de cada motor, entre otras muchas posibilidades. Es compatible con TinkerKit, lo que permite crear proyectos de forma rápida conectando los módulos TinkerKit a la placa.

- **Arduino Proto Shield:** proporciona espacio suficiente para montar circuitos personalizados. Permite soltar las partes necesarias para crear un proyecto o utilizar la zona de prototipos con una pequeña protoboard sin soldar para testear las ideas que tenemos hasta el momento desarrolladas. Cuenta con conexiones a los pines de E/S de Arduino.

### 2.5.3 Accesorios

- **Pantalla TFT LCD.** Las pantallas LCD convencionales, como las que se encuentran en las computadoras, muestran elementos de imagen controlados directamente: se puede aplicar voltaje a un segmento sin afectar a otros segmentos. Esto no puede suceder en pantallas grandes con muchos píxeles porque se requieren millones de conexiones: cada uno de los tres colores (rojo, verde y azul) de cada píxel está conectado en la parte superior e inferior. Para evitar esto, los píxeles se direccionan en filas y columnas, lo que reduce la cantidad de conexiones de millones a miles. Si todos los píxeles de una fila reciben un voltaje positivo y todos los píxeles de una columna reciben un voltaje negativo, se aplica y cambia el píxel en la intersección con el voltaje más alto. La desventaja de esto es que todos los píxeles en la misma columna reciben una fracción del voltaje aplicado, al igual que todos los píxeles en la misma fila, por lo que se oscurecen incluso si no se convierten por completo. La solución a este problema es dar a cada píxel su propio transistor de conmutación, lo que permite el control individual de cada píxel. La baja corriente de fuga del transistor significa que el voltaje aplicado al píxel no se pierde cuando la imagen se actualiza en la pantalla. Cada píxel es un pequeño condensador con una capa transparente de óxido de indio y estaño en el frente, una capa transparente en la parte posterior y una capa de aislamiento de cristal líquido en el medio.

- **USB/Serial Light Adapter.** Tiene un ATmega8U2 programado con la idea de ser un convertidor de USB a serie. En el sistema operativo Windows necesita del archivo .inf. Consta de un conector mini USB integrado.
- **Mini USB/Serial Adapter.** La placa convierte la conexión USB en una TX y RX de 5V que puede ser conectada directamente a la placa Arduino Mini o a otros microcontroladores.
- **Arduino ISP.** Permite cargar sketches y grabar el gestor de arranque de cualquier placa base AVR, incluyendo Arduino. Esto permite la posibilidad de recuperar el gestor de arranque en caso de algún fallo o también se puede utilizar dicho gestor en un nuevo microcontrolador ATmega en la placa Arduino.

#### 2.5.4 Arduino IDE

Arduino Integrated Development Environment (IDE) es una aplicación multiplataforma escrita en el lenguaje de programación Java. Se utiliza para escribir y transferir programas a placas compatibles con Arduino, pero también se puede utilizar con placas de terceros que utilizan núcleos de terceros.

El código fuente del IDE se publica bajo la Licencia Pública General GNU versión 2. El IDE de Arduino admite los lenguajes C y C++ utilizando reglas de estructura de código especiales. El IDE de Arduino proporciona una biblioteca de software para el proyecto Wiring que proporciona muchos procesos de E/S comunes. El código escrito por el usuario requiere solo dos funciones básicas, el bosquejo del gestor de arranque y el bucle principal, compilado y repetido junto con el atajo main(), y también incluye la cadena de herramientas GNU. El IDE de Arduino utiliza el programa avrdude para convertir el código ejecutable en un archivo de texto codificado hexadecimal que se carga en la placa Arduino mediante el programa de transferencia de firmware integrado.



##### 5. Versión 1.8.5 del IDE Arduino

Para programar el Arduino Leonardo, al igual que otras placas Arduino, puedes programar desde diferentes plataformas como macOS, Windows y Linux. Esto es gracias al entorno de desarrollo Arduino IDE que está disponible para estas plataformas.

Para obtener más información sobre cómo comenzar a programar con esta placa, tan solo tienes que buscar documentación oficial, cursos creados por empresas o tutoriales realizados por aficionados. Tan solo hay que seleccionar la placa adecuada en el menú IDE de Arduino para cargar el programa. Esto significa abrir el IDE de Arduino, ir a Herramientas > Placas > Seleccionar Leonardo... y empezar a disfrutar de los proyectos que ha creado usted mismo o que puedes encontrar en la red.

## 2.6 Nextion

Nextion es una solución de interfaz hombre-máquina (HMI) que combina un procesador integrado y una pantalla táctil habilitada para memoria con el software Nextion Editor para el desarrollo de proyectos de GUI HMI. De este editor hablaré más adelante ya que es con el que configuré y personalice los botones del OpenDeck.

Con el software Nextion Editor, puede desarrollar rápidamente GUI de HMI arrastrando y soltando elementos (gráficos, texto, botones, controles deslizantes, etc.) e instrucciones de texto ASCII para codificar elementos de interacción en la pantalla.

La pantalla HMI de Nextion está conectada a la MCU periférica a través del puerto serial TTL (5V, TX, RX, GND) para proporcionar notificaciones de eventos que la MCU periférica puede manipular y la MCU periférica puede fácilmente Actualizar fácilmente el progreso y el estado de regreso a Nextion pantalla utilizando texto ASCII.

### 2.6.1 Pantalla táctil Nextion

Los componentes fáciles de usar, la programación de eventos táctiles y una GUI personalizada en pantalla le permiten crear proyectos de forma rápida y rentable. Los monitores de la serie Nextion TTL son la solución HMI más rentable con una curva de aprendizaje corta y baja. Consulte los tutoriales y guías del editor de Nextion.

### 2.6.2 Nextion Editor

El software Nextion Editor proporciona una forma sencilla de crear interfaces de usuario táctiles sorprendentes e intuitivas, incluso para principiantes. Agregar una imagen estática como fondo, definir la funcionalidad de su elemento y se creará una interfaz gráfica de usuario simple en minutos. Los componentes simples de arrastrar

y soltar y las instrucciones de texto ASCII reducirán en gran medida el esfuerzo involucrado en el desarrollo de un proyecto HMI.

### 3. COMPONENTES

#### 3.1 Arduino Leonardo

Una vez estudiadas las opciones más populares e interesantes dentro del mundo de las placas Arduino y las Raspberry Pi, finalmente opté por el Arduino Leonardo. Una de las grandes dudas a la hora de realizar este proyecto ha sido decidirme por un tipo o por otro y es cierto que durante una parte importante de este Trabajo Fin de Grado, todo el OpenDeck estuvo siendo ideado para ser desarrollado en una Raspberry Pi 3 Model A+.



6. *Raspberry Pi 3 Model A+*

Este modelo de Raspberry me permitía disponer de la potencia necesaria para todo lo que tenía en mente en un tamaño de placa muy reducido. Otro de los puntos a favor era que se trata de un tipo de placas muy conocidas, eso significa que existen multitud de bibliotecas, tutoriales y complementos desarrollados por los usuarios. Sin embargo, después de tanto estudiar esta llamativa opción, acabé eligiendo el Arduino por los siguientes motivos:

- *Más económico que la Raspberry(10 euros vs 30 euros)*
- *Sencillo de programar(IDE Arduino)*
- *Fácil de utilizar por el usuario*
- *Potencia sobrada para el proyecto a realizar*
- *Multitud de accesorios y módulos compatibles con Arduino*

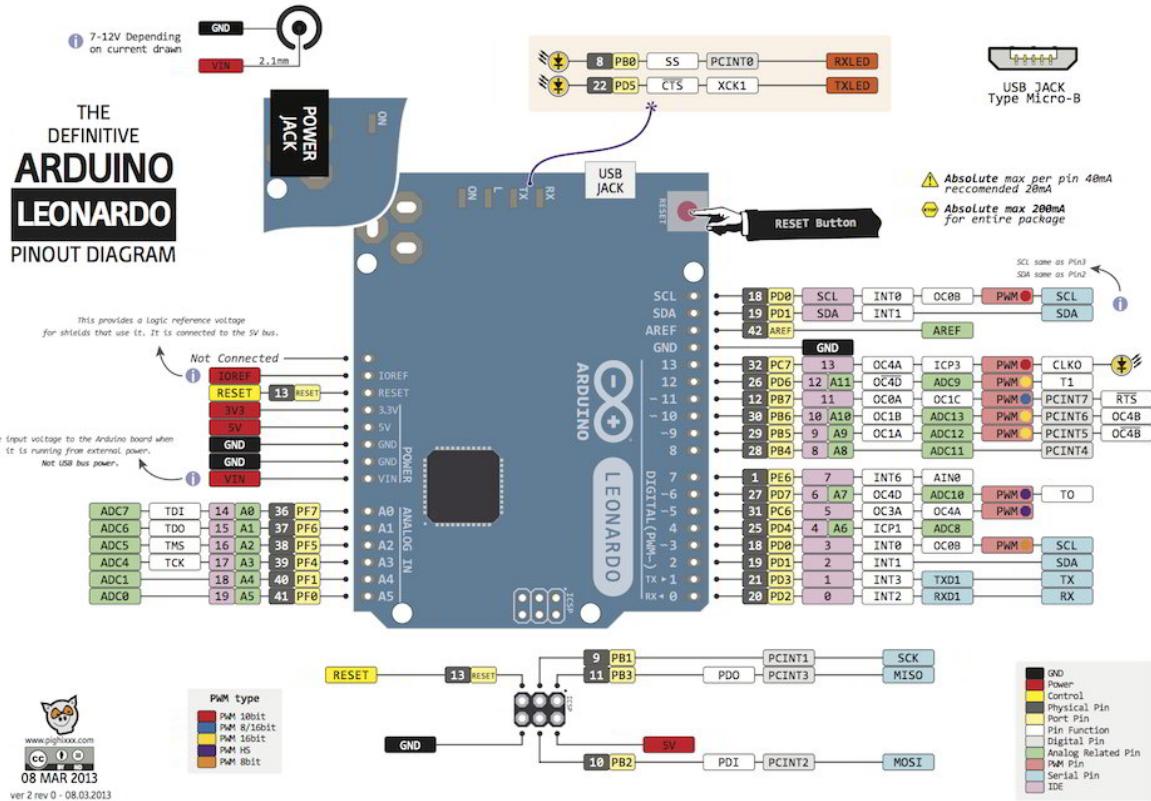
Todo esto me acababa llevando a la conclusión de que la mejor opción para este proyecto era el Arduino Leonardo. Mi idea desde un primer momento era muy clara, necesito un dispositivo que me permita crear una alternativa al Stream Deck pero mucho más barato y con una potencia y tamaño similares. Esta placa me aporta todo lo necesario para cumplir con esos objetivos, además de que cumple con los dos requisitos más importantes de todos desde mi punto de vista, es muy económica y no necesita de ningún sistema operativo o configuración especial para poder empezar a trabajar con ella. En definitiva, barata y fácil de usar tanto para el desarrollador como para el usuario final.



7. Arduino Leonardo

En comparación con algunos de sus hermanos, esta placa de microcontrolador programable tiene una de las funciones más potentes de esta serie.

Por supuesto, esta placa oficial de la Fundación Arduino es compatible con una gran variedad de dispositivos electrónicos. De esta manera puedes combinar la placa Leonardo como quieras con tantos componentes para crear los diseños más variados que puedas imaginar. Es muy similar al Arduino Uno, incluso en apariencia. Pero no deben confundirse, ya que existen diferencias significativas entre ellas...



8. Diagrama del Arduino Leonardo

Una de las características clave del Arduino Leonardo que se debe destacar es su pinout, que son los pines o conexiones que tiene. Como se puede observar en la imagen de arriba[8], no es lo mismo que la placa UNO Rev3. Existen varias diferencias entre cantidad, límites y buses.

Las especificaciones de esta placa pueden resumirse como las siguientes:

- Microcontrolador:** Atmel ATmega32U4 a 16 Mhz, uno de los más populares dentro de la marca Arduino.
- Memoria SRAM:** 2.5 KB.
- EEPROM:** 1 KB.
- Flash:** 32 KB, pero hay que restarle 4 KB usados para el bootloader. Otro tipo de memoria Flash muy utilizada por otras placas comentadas anteriormente.

- Voltaje de operación: 5v.*
- Voltaje de entrada (recomendado): 7-12v.*
- Voltaje de entrada (límite máximo): 6-20v.*
- Pines E/S digitales: 20, de los cuales 7 son PWM.*
- Pines de entradas analógicas: 12 canales.*
- Intensidad de corriente por pin E/S: 40mA.*
- Intensidad de corriente para el pin 3.3v: 50mA.*
- Peso y dimensiones: 68.6×53.3mm y 20 gramos.*

Teniendo en cuenta el reducido tamaño de esta placa, posee unas especificaciones más que interesantes. Es por eso que se trata de una de las más conocidas, cumple con los requisitos necesarios para la mayoría de proyectos desarrollados por los usuarios, además de ser muy fácil de manipular gracias a sus dimensiones.

Como es habitual con las placas Arduino oficiales, hay una gran cantidad de esquemas, datos y documentación sobre el tema, e incluso es posible construir una placa a partir de ésta, ya que es de código abierto. La información existente tanto en webs oficiales como en toda la red es abrumadora, prácticamente cualquier idea que tengas en mente puede ser encontrada en un foro o página particular.

¿Por qué Arduino Leonardo y no Arduino UNO? Aquí la respuesta.

Lo ideal sería compararla con la placa más parecida, la Arduino UNO Rev3. Si comparas el Arduino Leonardo con el UNO, verás muchas similitudes, pero estas diferencias también son importantes a la hora de decidirse entre una u otra.

Físicamente parecen tener el mismo tamaño y el mismo número de pines. También están dispuestos de la misma manera. La fuente de alimentación es la misma, incluso la frecuencia proporcionada por el generador de frecuencia. El A0-A5 también se puede configurar digitalmente usando la función pinMode (número de pin, modo). Entonces, ¿cuál es la diferencia entre ellas?

Bueno, una de las principales diferencias entre las dos placas de desarrollo es el microcontrolador. El UNO está basado en el ATmega328, mientras que el Arduino Leonardo en su última versión está basado en el ATmega32U4. Para ATmega328 no hay comunicación USB incorporada, por lo que se requiere un adaptador para este puerto serie.

La ATmega32U4 ya implementa comunicación USB, por lo que no necesita un segundo chip. A nivel de usuario real, marcará la diferencia. Al conectar la placa Arduino UNO, se especifica un puerto COM virtual para la comunicación. En Leonardo, el PC reconoce el disco como un dispositivo USB, como un ratón o un teclado. Esto hace posible el uso de funciones de estos dispositivos. El que el ordenador lo reconozca como un dispositivo USB también fue uno de los motivos importantes por los que elegí esta opción, hace más fácil y rápido el uso de esta placa con respecto a otras alternativas que tenía presentes.

Por supuesto, al tener una MCU(Microcontroller Unit) diferente, algunos datos de la memoria también son diferentes. De la memoria flash de 32KB del Arduino UNO (de los cuales 0.5 KB están reservados para el cargador de arranque) se pasa a los 32 KB y 4 KB utilizados por el bootleader en Leonardo. En caso de SRAM aumenta de 2 KB a 2.5 KB, en caso de EPROM es igual en ambos casos.

Otra diferencia es el canal de entrada analógica. El Arduino UNO tiene solo 6 canales, mientras que el Arduino Leonardo tiene 12 canales. Esto se aplica a A0-A5, y los pines 4, 6, 8, 9, 10 y 12 corresponden a los canales A6-A11.

Cuando se trata de PWM, Leonardo tiene más de UNO. Además de las de UNO, se agrega otra tarjeta en el pin 13. Las otras dos tarjetas son iguales, es decir, en los pines 3, 5, 6, 9, 10 y 11.

Se encuentran más diferencias en la comunicación I2C. Ambos pueden usar TWI, pero la diferencia es el pinout de la línea de datos en serie o SDA y SCL o la de reloj. En el UNO, están en los pines A4 y A5. Pero en Leonardo se encuentran en el 2 y 3 respectivamente. Ligeramente diferentes, pero suficientes para hacer que los sombreros o escudos de UNO no sean totalmente compatibles con Leonardo.

Respecto a la comunicación SPI, el Arduino UNO dispone de los pines 10, 11, 12 y 13 para señales SS, MOSI, MISO y SCK respectivamente. Esto no se aplica a Leonardo, ya que tiene un conector ICSP dedicado, un conector macho de 6 pines en el extremo de la tarjeta.

También se han realizado algunos cambios en las interrupciones externas. En el UNO tienes dos pines, el pin 2 (interrupción 0) y el pin 3 (interrupción 1). En el caso del Arduino Leonardo, se amplían a 5 pines. Estos son los pines 3, 2, 0, 1 y 7 para las interrupciones 0, 1, 2, 3 y 4 respectivamente. Otro cambio entre las dos placas que mucha gente olvida es el tipo de cable USB necesario para conectar las dos placas a la computadora. Mientras que el UNO usa un cable A-B, el Leonardo requiere un cable A-microB.

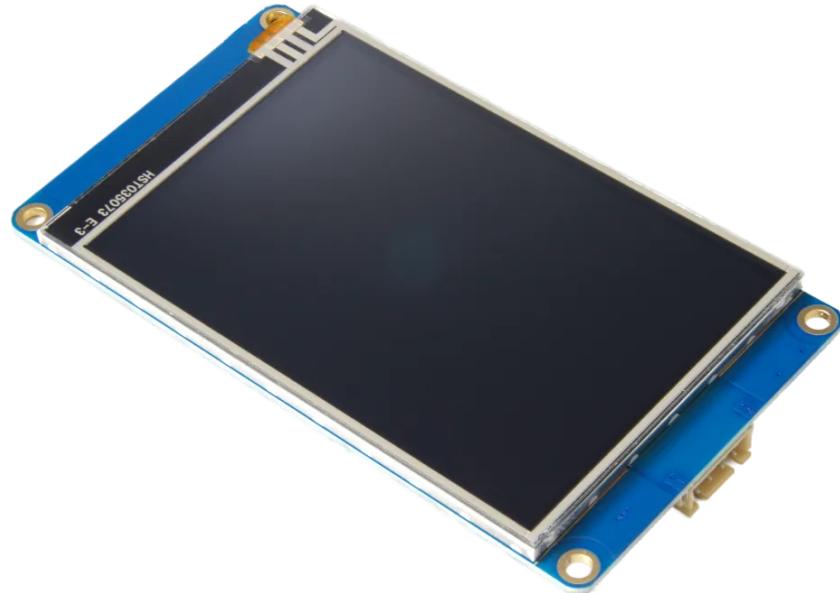
Después de analizar todos estos datos, acabé seleccionando el Leonardo y puedo asegurar que hice bien. Ha cumplido con todas mis expectativas y más.

### 3.2 Nextion 3.5" NX4832T035

Otro de los componentes principales del proyecto se trata de la pantalla Nextion. Me estuve informando durante bastante tiempo sobre las distintas pantallas que podrían ser usadas con mi placa Leonardo. Primero estuve sopesando la opción de usar pantallas no táctiles porque resultan más económicas y son más fáciles de encontrar en el mercado pero, finalmente opté por la opción táctil. Considero que la interacción que le aporta una pantalla táctil al usuario es algo muy beneficioso para el proyecto.

Dentro de las distintas posibilidades que fuesen compatibles con Arduino Leonardo, la que más me convenció fue la Nextion por tres motivos fundamentales para el proyecto. El primero de ellos es que es una pantalla muy económica, el segundo es que resulta muy fácil de utilizar ya que simplemente hay que conectarla al arduino mediante 4 jumpers(más tarde hablaré de ellos) y a funcionar y, por último, cuenta con un software del que ya he informado anteriormente(Nextion Editor) que ofrece todas las posibilidades que yo necesitaba para este trabajo.

Todo esto estuvo acompañado de que realmente, tras buscar mucho, el precio de las pantallas táctiles según desde dónde sean compradas tampoco es tan elevado. En este proyecto me he encontrado con un problema que, a pesar de no ser grave, no me ha resultado demasiado agradable pero al fin y al cabo es totalmente lógico y entendible. Estoy hablando de la subida de precio que han sufrido las pantallas táctiles debido a la pandemia de la Covid-19. Siempre me ha gustado este tipo de dispositivo, eso me hace saber que antes de la pandemia, esta pantalla de la que estamos hablando costaba unos 15 euros(muy buen precio en mi opinión vistas otras opciones en el mercado) y tras todo lo sucedido, su precio se ha duplicado. Me gustaría pensar que en cuanto pase todo lo que estamos viviendo, los precios volverán a su origen.



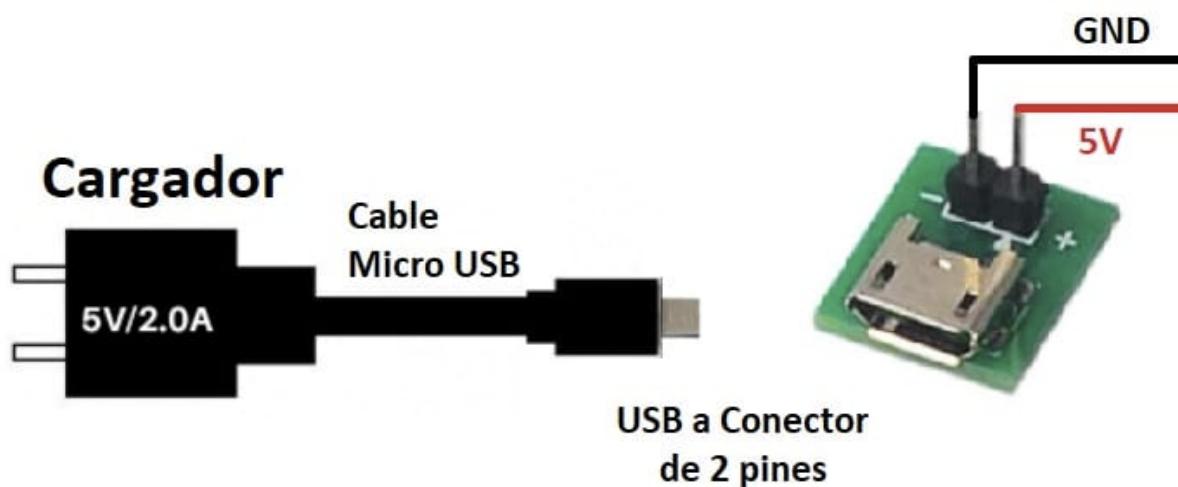
9. *Nextion 3.5" NX4832T035*

Las especificaciones del dispositivo son las siguientes:

- *Voltaje de Operación:* 4.75V (mín.), 5V (típico) y 7V(máx.)
- *Corriente de Operación:*
  - *100% de brillo:* 145mA
  - *Modo Sleep:* 15 mA
- *Dimensiones:* 73.44mm x 48.96mm
- *Resolución:* 480×320 píxeles
- *Color:* 16 bit 565, 65K , 65536 colores

- *Memoria Flash*: 16 MB
- *Memoria RAM*: 3584 byte
- *Interfaz* : 4 pines (+ 5V, TX, RX, GND)
- *Puerto Serial*: TTL
- *Interfaz USB*: No
- *Puerto SD*: Si, Formato FAT32 , Micro SD 32G máximo
- *Temperatura de Trabajo*: -20°C a 70°C

Para poder probar y cargar información fácilmente en la pantalla, se incluye una pequeña placa de prueba de la fuente de alimentación y el cable de conexión que permiten probar tanto el funcionamiento de la pantalla como si el suministro eléctrico es el adecuado.



10. Ilustración del cargador recomendado para el uso de la pantalla e imagen de la placa de prueba

### 3.3 Cable jumper

Este es el elemento fundamental para la conexión entre la pantalla y la placa Arduino Leonardo. Los cables de puente comúnmente en muchos kits electrónicos, desde algunos para robots, hasta Arduino y más. Además, son cables muy prácticos para multitud de proyectos electrónicos. No solo se puede usar en PCB como pines GPIO en Raspberry Pi, sino también en diseños de placas de pruebas.

Un jumper o puente es un componente electrónico que permite que los terminales abran o cierren un circuito. Los jumpers suelen estar soldados a una PCB como la GPIO Raspberry Pi (macho) o la entrada y salida de una placa Arduino (hembra). Gracias a una pequeña pieza de plástico con una placa conductora en su interior, se puede conectar fácilmente a estos terminales para hacer un puente.



11. Cables jumper

Probablemente hayas visto esto en las placas base de muchas computadoras: el botón de reinicio, el botón de encendido o el propio jumper que se usa para borrar el BIOS está conectado a la placa base. En el pasado, también se usaban para configurar voltajes, es decir, modos de funcionamiento de periféricos, celdas de memoria M/S(Memory Stick), modos de emulación de algunas tarjetas de expansión, etc. Estos puentes suelen estar dispuestos en una o más filas, y los conectores utilizados suelen conectar 2 de estos pines, aunque los hay de más

pines. Son una alternativa más económica y compacta a los interruptores DIP. Sin embargo, también tienen limitaciones, como algunas inscripciones confusas o solo su función específica en las instrucciones de PCB.

Como mencioné antes, estos pines o terminales están conectados por una pequeña pieza o un puente. Pero para ello, los terminales deben ser contiguos. Si no están presentes o si están ubicados muy separados en diferentes circuitos o partes del tablero, se debe usar un jumper.

Estos cables son muy comunes para conectar muchos módulos y dispositivos al GPIO de Raspberry Pi o para conectar componentes a la placa Arduino(en este caso la pantalla Nextion), para conectar el botón de reinicio y el botón de encendido a la placa base de la computadora, para proyectos integrados. En lugar de soldar, es fácil de montar y desmontar, etc.

Estos cables solo se pueden mantener en su lugar insertando los extremos con contactos macho que se cruzan con los pines hembra. Dicho esto, se conectan y desconectan muy rápido.

Existen varios tipos de jumpers que se diferencian según su forma: los hay macho y hembra, pero en el mercado encontrarás cables con extremos uniformes o heterogéneos. Es decir:

- ❖ **Hembra-hembra**, en ambos extremos.
- ❖ **Hembra-macho**.
- ❖ **Macho-macho**, en ambos extremos.

Depende del método de conexión: suelen tener extremos aislados, los tipos más comunes pueden ser macho o hembra, también existen versiones especiales con pinzas de cocodrilo en la parte superior. Este tipo de abrazadera se puede usar para leer resultados o para conectar componentes temporalmente cuando un conector específico no está disponible.

### 3.4 Cable USB a micro USB

Hablamos del componente que permite conectar la placa Arduino al PC o para conectar a la corriente la pequeña placa de prueba que viene con la pantalla Nextion.

El cable es un alambre formado por diferentes conductores aislados entre sí y cubiertos por una funda protectora. Por otro lado, USB es un acrónimo de Universal Serial Bus: un estándar en informática que establece protocolos y conectores en el bus para conectar dispositivos.

Así, un cable USB es un conector que permite conectar distintos componentes a través del Universal Serial Bus. Esta interfaz fue creada en la segunda mitad de la década de 1990 por varias empresas, incluidas Microsoft, Intel, IBM y NEC, para garantizar la compatibilidad de los dispositivos.



12. Cable USB a micro USB

Un PC con un puerto USB, puede conectar una variedad de periféricos: desde un teclado a un ratón, pasando por una impresora, un escáner, un altavoz o un mando

de videojuegos. Incluso puede conectar discos duros externos, tarjetas sintonizadoras de TV y otros dispositivos.

Digamos que queremos copiar una foto tomada con una cámara digital a nuestro ordenador. Para ello necesitamos un cable USB con un conector en un extremo que necesitamos conectar a la cámara. Por lo general, la cámara ya viene con este cable. Por lo tanto, necesitamos conectar un extremo del cable de datos a la cámara y el otro extremo al puerto USB del PC. El dispositivo puede reconocer automáticamente esta conexión y brindarnos la capacidad de reconstruir la imagen correspondiente.

Además de transferir datos, los cables USB también se utilizan para alimentar dispositivos, cargar baterías o simplemente mantenerlos en funcionamiento. Hoy en día, los monitores suelen tener uno o más puertos USB, para que los usuarios puedan cargar cómodamente teléfonos y tablets conectándose directamente, en lugar de las torres de ordenadores que suelen estar ubicadas en lugares donde hay menos espacio para el acceso de los usuarios.

Si bien es cierto que la simplicidad de la conexión es uno de los supuestos del estándar USB, los intereses de los fabricantes de hardware y el deseo de muchos de destacarse de la competencia a toda costa han llevado a la fabricación de una gran variedad de conectores. El tipo de conexión puede ser confuso para alguien que no está muy informado sobre el asunto. ¿Necesito un cable USB "micro" o "mini" para conectar esta tableta gráfica? ¿Es tipo A o tipo B?

Algunos de los tipos de USB más comunes son los siguientes:

- **Tipo A:** Este es el tipo más utilizado. De hecho, este es el que solemos encontrar al entrar en un ordenador de sobremesa y portátil. Cuando decimos "USB" en el lenguaje cotidiano, solemos pensar en este tipo de conector.
- **Tipo B:** tiene un conector "cuadrado" que se ha utilizado en la mayoría de los modelos de impresoras, aunque ahora se usa con menos frecuencia.

- **Tipo C:** es el tipo más utilizado en la actualidad. Ofrece velocidades de transferencia de datos más altas y puede transferir más energía que sus predecesores. Los últimos dispositivos exitosos han adoptado este tipo de conector USB, incluidos los dispositivos móviles más novedosos, que lo usan para cargar la batería o para transferir datos, todo a través del cable USB.
- **Mini USB:** antes de la llegada de los cables USB C, era el tipo más común en teléfonos móviles y cámaras digitales.
- **micro USB:** todavía se usa para cargar algunos teléfonos, especialmente teléfonos de gama baja y media.

### 3.5 Carcasa mediante impresión 3D

Desde un primer momento tuve claro que el resultado final del OpenDeck iba a contar con una carcasa impresa en 3D. Esta elección se debía a que quería que tuviese un buen acabado y no se viesen a simple vista los componentes comentados anteriormente.

Una vez tuve claro esto, necesitaba pensar en cómo conseguir un modelo 3D que fuese compatible con mi pantalla, el Arduino y los demás cables que componen todo el proyecto. Afortunadamente, como ya he comentado en numerosas ocasiones, este proyecto es una muestra más de que todos estos elementos(Arduino, streamings, proyectos independientes, etc.) están en su mejor momento. Puedes encontrar cualquier tipo de solución en Internet y, en muchos casos, de forma totalmente gratuita.

Antes de mostrar el diseño 3D que elegí para este proyecto, voy a hablar sobre la historia y la evolución que ha tenido esta tecnología de la impresión 3D. Me parece de gran utilidad e importancia ya que se está comprobando que con los avances, este tipo de impresiones acabará siendo de gran relevancia en muchos sectores.

Cuando las personas buscan en Google la historia de las impresoras 3D, aparece información que se refiere a la década de 1980, cuando apareció la tecnología. Sin embargo, el embrión, su origen, se remonta a muchos años atrás.

En el siglo XIX aparecieron las primeras ideas sobre la reproducción automática de objetos tridimensionales. Creado en 1859 por el escultor francés François Willème, se dice que es el primer intento en la historia de un escáner 3D. Lo hizo configurando 24 cámaras para tomar la misma foto desde diferentes ángulos. Joseph E. Blanther patentó el desarrollo de mapas tridimensionales en 1892 utilizando una técnica de estratificación similar a la utilizada cientos de años después por las impresoras 3D. Estos fueron los primeros comienzos, las semillas de la historia de la impresión 3D. Pero para hablar de la impresión 3D en sí, todavía tenemos que esperar casi cien años.

La década de los 80 supondría un gran avance en la historia de la impresión 3D. Suponía un momento difícil porque se vivía en una época en la que la gente todavía piensa en 3D (el boom de la fotografía digital aún no ha llegado). En 1981 se presentó la primera solicitud de patente para impresión 3D. El proyecto estaba dirigido por Hideo Kodama, investigador del Instituto de Investigación de Nagoya. Kodama tiene la intención de crear piezas sólidas a través del endurecimiento de una tina de fotopolímero con luz UV.

El plan de Kodama nunca llegó a buen término, y él no fue el único que sufrió el mismo destino. En Francia, Alain Méhauté trabajó en Alcatel, creó copias de piezas geométricas con formas fractales. Dada la dificultad de la tarea, Méhauté trató de encontrar una manera de fabricar piezas complejas de forma más rápida y sencilla.

Le Méhauté decidió compartir sus problemas con su amigo Olivier de Witte, que trabajaba en la sucursal de Alcatel. De Witte es un experto en el uso de láseres y descubrió que algunos líquidos se pueden curar con láseres.

Con la idea de la polimerización por láser, Le Méhauté y De Witte recurrieron a su amigo Jean-Claude André, que trabaja en el Centro Nacional Francés de Investigaciones Científicas (CNRS). Aunque Andrei expresó su apoyo al proyecto, el

CNRS nunca lo aprobó. La razón era que creían que no había suficientes áreas de aplicación.

Los primeros años de la década de 1980 estuvieron marcados por proyectos fallidos o infructuosos. Pero estaban abiertos a otros que tuvieran ideas más elaboradas o más dinero para seguir sus pasos.

A partir de 1984, y especialmente a finales de la década de 1980, tienen lugar los acontecimientos que impulsaron el nacimiento de la impresión 3D como industria. Durante esos años se crearon las primeras patentes y empresas. Nació oficialmente la tecnología de impresión 3D SLA, SLS y FDM.

1984 fue un año caluroso en la historia de esta creciente tecnología. Ese fue el año en que Chuck Hull inventó la impresión estereoscópica (SLA). Este método de impresión 3D se basa en la síntesis láser en capas. Unos años más tarde, en 1986, Hull patentó su invento y fundó 3D Systems, la primera empresa en el campo de la impresión tridimensional. 3D Systems también sería el primero en ofrecer impresoras 3D SLA comerciales, específicamente el modelo SLA-1.

Unos años más tarde, aparecieron los primeros sistemas de impresión 3D alternativos. Era 1987 y Carl Deckard, científico de la Universidad de Texas, inventó un sistema que podía sintetizar polvos plásticos y convertirlos en sólidos. Así nació la impresión 3D SLS. Al igual que el invento de Hull, la máquina de Deckard funcionaba con un láser, pero sintetizaba polvo en lugar de plástico líquido. La primera máquina diseñada por Deckard para usar este sistema se llamó Betsy.

Sin embargo, es cierto que este método de impresión existe desde hace mucho tiempo, pero la primera impresora 3D SLS comercial apareció recientemente en 2006. Como tal, el camino en la historia de las impresoras 3D acaba de comenzar.

Las impresoras 3D FDM son las más famosas en la actualidad y ocupan una gran parte del mercado. Muchas personas pueden pensar que es la primera impresora 3D con este sistema, pero en realidad es de una empresa de terceros.

Es bien sabido que la tecnología FDM se basa en la deposición de una fibra fundida. El filamento se calienta para que pueda pasar a través de la boquilla y colocarse en plataforma o cama caliente.

1988 también es importante para la historia de las impresoras 3D, ya que S. Scott Crump y Lisa Crump patentaron la tecnología FDM. En 1989 fundaron Stratasys, hoy en día una de las empresas más importantes del sector en lo que a impresoras 3D profesionales se refiere.

La historia de la idea de esta tecnología es curiosa. Crump padre trató de hacer una rana de juguete para su hija usando una pistola de pintura llena de cera caliente y termoplástico. Cansado de no poder obtener los resultados que quería, ideó cómo hacer el mismo proceso con la superposición de capas de termoplástico.

La década de 1990 fue una década de empresas que jugaron un papel muy importante en el desarrollo de la impresión 3D en diferentes áreas de especialización. 3D Systems Corporation o Stratasys se unirían a otras empresas en la década de 1990, como Z-Corporation, ARCAM y Object Geometries.

En ese momento, la impresión 3D hizo avances en áreas como la odontología, la fabricación de prótesis y la medicina. A finales de siglo, unos científicos del Wake Forest Institute for Medical Research lograron crear los primeros órganos en el laboratorio mediante una técnica basada en la regeneración de órganos, utilizando las células de uno mismo. Este es el primer paso en el campo de la bioimpresión 3D.

Con la llegada del siglo XXI, todos estamos preocupados por el efecto Y2K (también conocido como efecto 2000, error del milenio o problema informático del año 2000) , que es un error de software causado por la forma en que los programadores almacenaban y programaban las fechas, ya que el software en cuestión solo estaba pensado para ejecutar en los años 19XX. Todo esto se supone que conducirá al colapso global. Pero era mentira, tan solo nos hizo ver que se iniciaba un siglo en el que el desarrollo tecnológico sería primordial.

Las impresoras 3D siguen siendo un producto caro y centradas en ámbitos profesionales, pero el alcance cada vez se amplía a más áreas: arquitectura, joyería, automoción, diseño de personajes decorativos, etc.

Sin embargo, todavía faltaba algo para que esta tecnología llamara más la atención. El año 2005 marcó un hito importante en la historia de la impresión 3D. El mismo año, Adrian Bowyer fundó el proyecto RepRap con el objetivo de crear una impresora 3D abierta y autorreplicante que pudiera imprimir sus propias piezas para crear nuevos modelos. Sin embargo, la iniciativa Fab@Home de la Universidad de Cronwell sería la que lograría lanzar con éxito la primera impresora 3D de código abierto. Todo esto representa un gran paso adelante en el suministro de impresoras 3D a los usuarios o a los centros de aprendizaje.

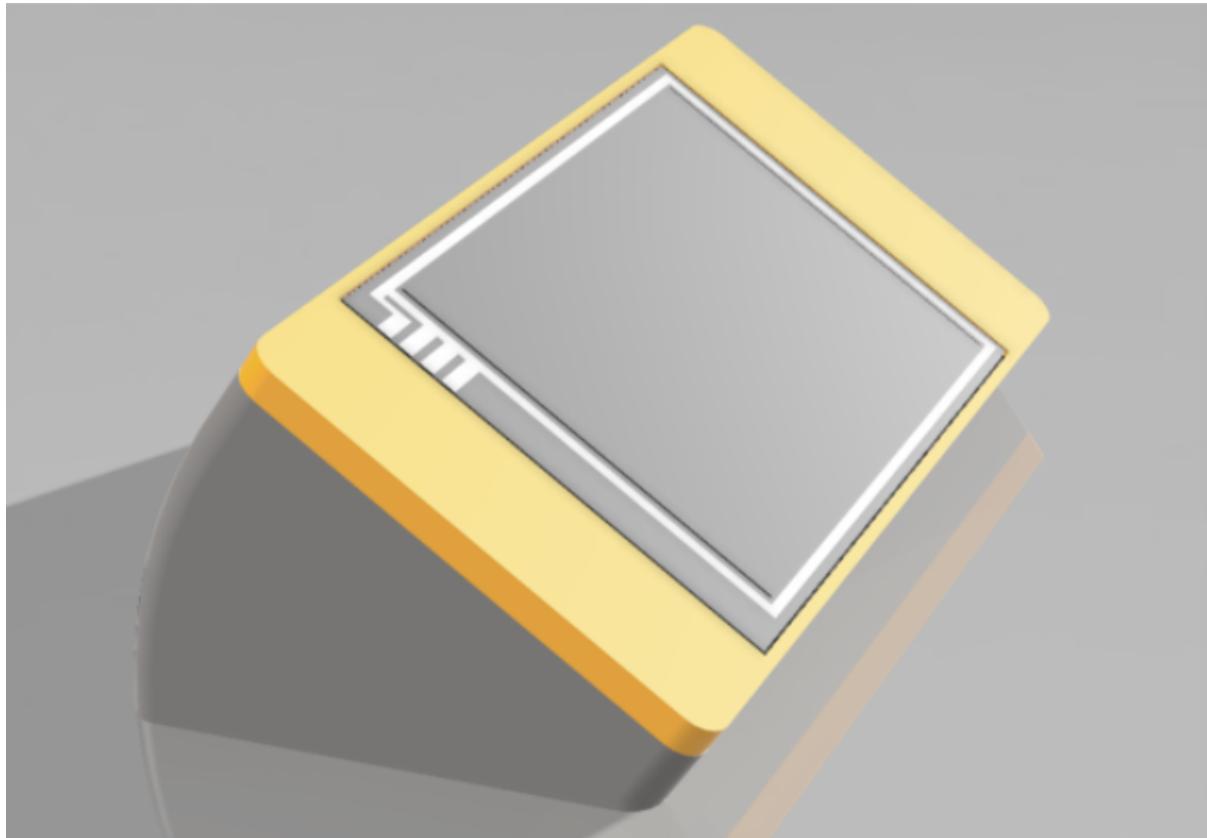
Los años 2010 están marcados por la llegada definitiva a casa de las impresoras 3D. Aunque todavía nos queda mucho camino por recorrer, la mayoría de la gente ya es consciente de que estas máquinas existen y que hoy en día no son tan caras ni tan inútiles como se creía hace años.

Gran parte de la razón de esto radica en empresas como Makerbot, que lanzó el primer kit de impresora 3D al mercado en 2009. Estos kits son muy populares en la actualidad (especialmente entre los fabricantes de China), lo que permite a los usuarios comprar impresoras 3D baratas a cambio de montarlas y calibrarlas ellos mismos.

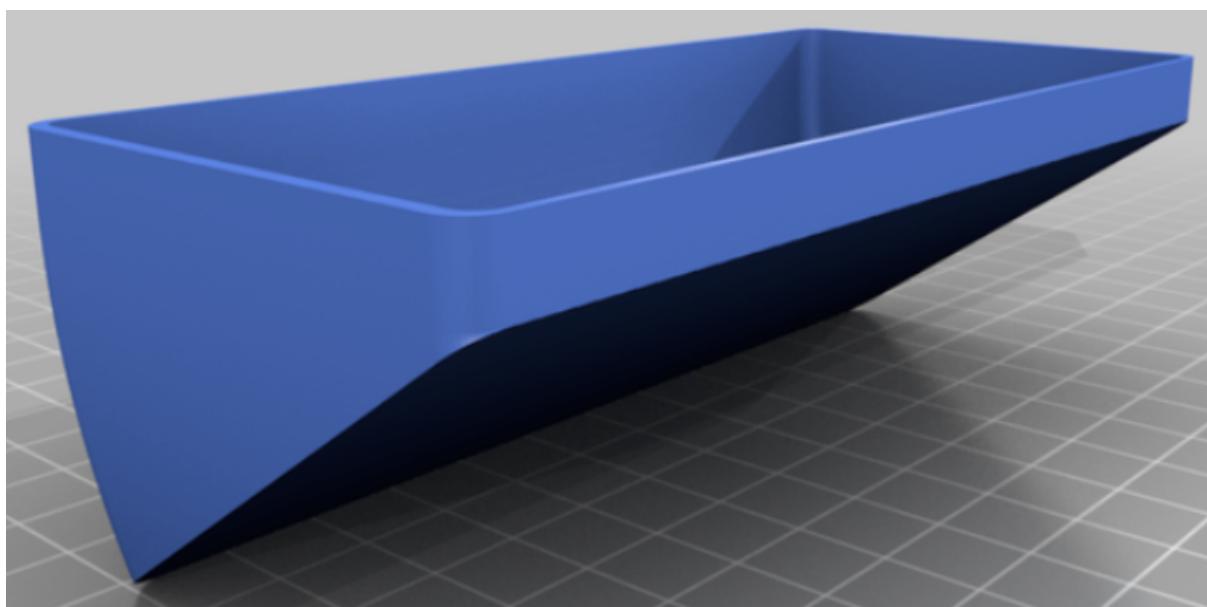
Muchas de estas impresoras 3D económicas han atraído a una gran cantidad de nuevos usuarios en todo el mundo. Algunos de los modelos más famosos son la Anet A8 o la Creality Ender 3, entre otros muchos.

Sin embargo, la historia de las impresoras 3D apenas comienza a escribirse. A menudo vemos ejemplos de lo que estas máquinas pueden hacer, ya sea imprimir casas, un automóvil o réplicas de joyas de oro y plata. Todo esto nos hace pensar que el futuro de esta tecnología solo puede ir a mejor.

Tras repasar rápidamente y de forma muy resumida la historia de la impresión 3D, voy a mostrar el diseño que acabé seleccionando para que fuese la carcasa de mi proyecto.



13. Parte superior del modelo 3D de la carcasa del OpenDeck



14. Base del modelo 3D de la carcasa del OpenDeck

Elegí este modelo ya que constaba únicamente de dos piezas muy fáciles de colocar y con el espacio espacio suficiente para introducir los distintos componentes necesarios. Más adelante le añadiría algunas modificaciones a la carcasa como, por ejemplo, goma en la base para que no deslice el dispositivo al utilizarse y también un soporte para un lápiz táctil.

Tuve que realizar una serie de modificaciones del modelo mostrado en la imagen 14 ya que no iban a caber todas las partes del trabajo dentro de la carcasa. Estas modificaciones las hice con el software Vectary[4], que es gratuito y de un uso muy sencillo. Cuando muestre el resultado final de la carcasa se podrán observar estos cambios.

Ahora tocaba decidir a qué empresa podría encargar la impresión. Tras una búsqueda de algunas de las más conocidas en España, opté por INNOVA3D[3]. Se trata de una empresa situada en Valencia y que hace envíos a toda España por unos precios muy interesantes.

También me llamó la atención todas las posibilidades que ofrecía esta opción en cuanto a materiales y colores con los que realizar la impresión.

Características de los materiales de impresión 3D (Fused Deposition Modeling / FDM)						
Material	Resistencia	Nivel de detalle	Resistencia a la temperatura	Flexibilidad	Coste de producción	Resistencia a rayos UV
PLA	★	★★★	★	★	★	★
PLA HR-870*	★★★	★★	★★★	★	★★★	★
ABS	★★★	★★	★★★	★	★★	★
ASA	★★★	★	★★★	★	★★★	★★★
PETG	★★	★★★	★★	★★	★★	★★
FLEXIBLE	★★★	★	★	★★★	★★★	★

#### 15. Materiales disponibles para impresión 3D en INNOVA3D

Elegí el material PLA HR 870 en color negro para ambas piezas. Los motivos para elegir ese material son los siguientes:

- Es un material muy resistente***, esa es una de las características que debía tener el dispositivo. A pesar de ser un producto delicado, me gustaba la idea de que en caso de recibir un golpe por el motivo que sea, no le pasase nada y siguiese funcionando con total normalidad.
- El nivel de detalle es muy aceptable*** para lo que estoy buscando, no necesito un material que permita una precisión máxima ya que se trata de una carcasa y me gustaba la idea de que tuviese cierta textura rugosa para impedir el deslizamiento.
- Excelente resistencia a la temperatura***. Es fundamental teniendo en cuenta que vivo en Sevilla y no me gustaría que en caso de subir mucho las temperaturas, el plástico comenzase a deformarse. También, al tratarse de un dispositivo que va a ser usado cerca de un PC es ideal para que no se vea afectado por el calor desprendido por el mismo.
- No es apenas flexible***, no necesito que lo sea ni me interesa en absoluto. La idea es que sea un aparato con cierta robustez pero ligero.
- Muy buena relación calidad/precio***. A pesar de ser de los más caros en la tabla mostrada arriba[15], seguía resultando bastante barato y me aseguraría unos acabados de muy buena calidad.

Con todos estos aspectos estudiados, realicé mi pedido y en poco tiempo ya tenía la carcasa conmigo. Más adelante, en este documento mostraré el aspecto que tiene y las distintas modificaciones que le haré para conseguir que tenga el mejor acabado posible.

## 4. DESARROLLO

### 4.1 Ideas fundamentales

Tal y como llevo comentando desde el comienzo de todo este proyecto, creo que es buena idea tener en cuenta el que es con grandísima diferencia el producto número uno en este campo. Estoy hablando como ya he dicho anteriormente del Stream Deck de Elgato.

Considero de gran importancia el estudio de este dispositivo ya que el éxito que acompaña a esta herramienta tiene que deberse a una serie de fundamentos que se han realizado correctamente. Para ello, he estudiado las distintas versiones comentadas en el apartado 2.4 y he sacado una serie de conclusiones que resultan cruciales a la hora de plantear este trabajo. Estas serían las que considero más notables:

- Básicamente, el dispositivo usa botones para **no tener que estar constantemente interactuando directamente con el programa de streaming** que utilice el usuario.
- Puedes **olvidarte de los atajos de teclado** ya que van a estar registrados dentro del propio aparato.
- El Stream Deck **realiza las acciones fundamentales** a la hora de crear este tipo de contenido(cambiar de escena, reproducir contenido multimedia, configurar el audio, etc.).
- He comprobado que es **necesaria una respuesta visual en el panel de botones** para que el usuario sienta que al tocar un determinado botón se ha producido una acción en concreto.
- Resumir lo que hacen los grandes estudios televisivos en **un aparato pequeño, fácil de usar y de entender desde un primer momento**.
- Puede utilizarse con cualquier software o sistema que cuenta con atajos de teclado.
- **Diseño de botones llamativo e intuitivo.**

→ Permitir acciones más allá de las básicas comentadas hasta ahora.

→ Personalización de botones.

Es fundamental tratar uno de los apartados más importantes que me han llevado y motivado a realizar este trabajo. La accesibilidad y la sencillez para la realización de ciertas tareas pensando en aquellos usuarios con algún tipo de problema, ya sea alguna discapacidad o lesión temporal.

Personalmente, pienso en muchas ocasiones en todas aquellas personas con dificultad para la realización de tareas que para el resto de nosotros son de lo más simples y habituales. Afortunadamente en el mundo de los videojuegos y de la tecnología en general, se está avanzando mucho y se está teniendo cada vez más en cuenta este tipo de problema tanto en empresas pequeñas como en otras que son auténticos gigantes de la industria.

En lo que se refiere a la tecnología en general, gracias a los grandísimos avances que se han ido produciendo con el paso de los años, cada vez contamos con más herramientas y funcionalidades disponibles para aquellas personas que las necesiten. Existe ya de todo tanto en dispositivos físicos como en software. Podemos encontrar aplicaciones que faciliten la visualización de los distintos iconos y letras en una pantalla o, en el apartado físico, teclados adaptados a personas ciegas, joysticks para un manejo más sencillo a través del PC, etc.

Gracias a que la tecnología no va a dejar de estar presente y cada vez va a usarse más y más, estoy seguro de que este tipo de soluciones van a seguir desarrollándose hasta el punto de convertirse en algo fácil de encontrar y hasta cierto punto económico (espero). No me cabe duda que muchas organizaciones se interesan en estos temas únicamente porque pueden conseguir un gran beneficio gracias a ello, pero también estoy seguro de que existen personas detrás de todo esto que se preocupan de verdad por estos problemas y van a seguir luchando y ayudando a aquellos que lo necesiten.

Al fin y al cabo, da igual la intención con la que se haga algo de este tipo, en definitiva, lo importante es que se haga y se tengan en cuenta todos los distintos tipos de usuarios interesados en algo. En este caso, al tratarse de algo ya tan expandido y que está presente en todo el mundo, los usuarios interesados en este tipo de soluciones pueden contarse por millones.

En cuanto al mundo de los videojuegos hace ya varias décadas que se cuenta con ciertas facilidades para este tipo de usuarios. Desde dispositivos que permiten hablar con el ordenador y que este haga la operación que le pidas, aparatos que facilitan la interacción directa con el sistema que sea o modificaciones en audio o imagen para que los distintos componentes de un juego sean detectables para estas personas.

Evidentemente queda un largo camino por delante ya que en mi opinión personal, creo que deberían establecer unos requisitos mínimos en todos los desarrollos para que al salir al mercado, estuviesen disponibles para todos aquellos jugadores que lo deseen adquirir. Creo que esto puede acabar llegando gracias a la visualización que tienen estos temas a día de hoy y también debido a que grandes empresas tienen muy en cuenta esto y dan un ejemplo muy bueno de cómo hacer bien las cosas en este mundillo.

Todas estas conclusiones que he sacado después de analizar los distintos dispositivos que ofrece Elgato, hacen que ellos sean los líderes en este sector.

Antes de comenzar mi trabajo, necesitaba conocer todos estos puntos para saber que lo que iba a realizar iba a estar a la altura de lo exigido por los usuarios que ya conocen la mejor herramienta en el mercado.

#### 4.2 Panel de botones

Una vez listo para comenzar de una vez por todas con el proyecto, lo primero que decidí hacer fueron los botones del OpenDeck. Era un apartado al que tenía muchas ganas de echarle el guante ya que permite una libertad absoluta de creatividad.

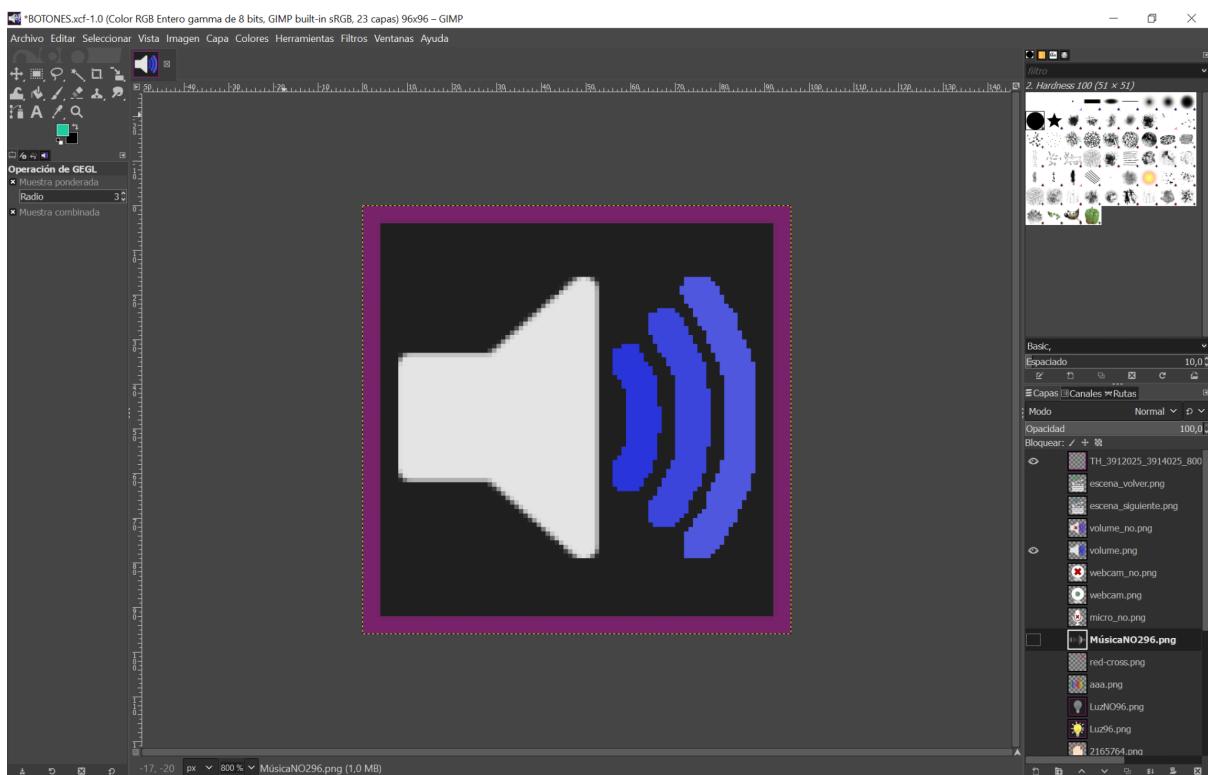
Estuve observando los botones más comunes y reconocidos del Stream Deck para hacerme una idea de cómo pueden gustarle a los usuarios y cómo hacerlos entendibles. La empresa alemana utiliza siempre un fondo azul o negro para sus botones, y los iconos mantienen un color blanco.

Yo he preferido hacerlo a mi manera para que mis botones sean únicos y diferentes a dicho dispositivo.

#### 4.2.1 Edición de los botones

Para conseguir eso he estado utilizando el software GIMP, es una aplicación gratuita y sencilla basada en Photoshop que me va a permitir ajustar los detalles y obtener la idea que yo tenga en mente.

Dentro de esta aplicación, he creado una plantilla con el tamaño que tendrán los botones, en este caso será de 96x96 píxeles. Este tamaño permite que la definición sea la suficiente como para que todo se represente bien y los colores se distingan con facilidad, además de que el archivo de la imagen ocupa realmente poco. Decidí que el fondo de todos los botones fuese un gris bastante oscuro, casi negro. Cada botón va a tener un marco morado que lo separe y diferencie del resto de botones del panel.



16. Vista general de GIMP

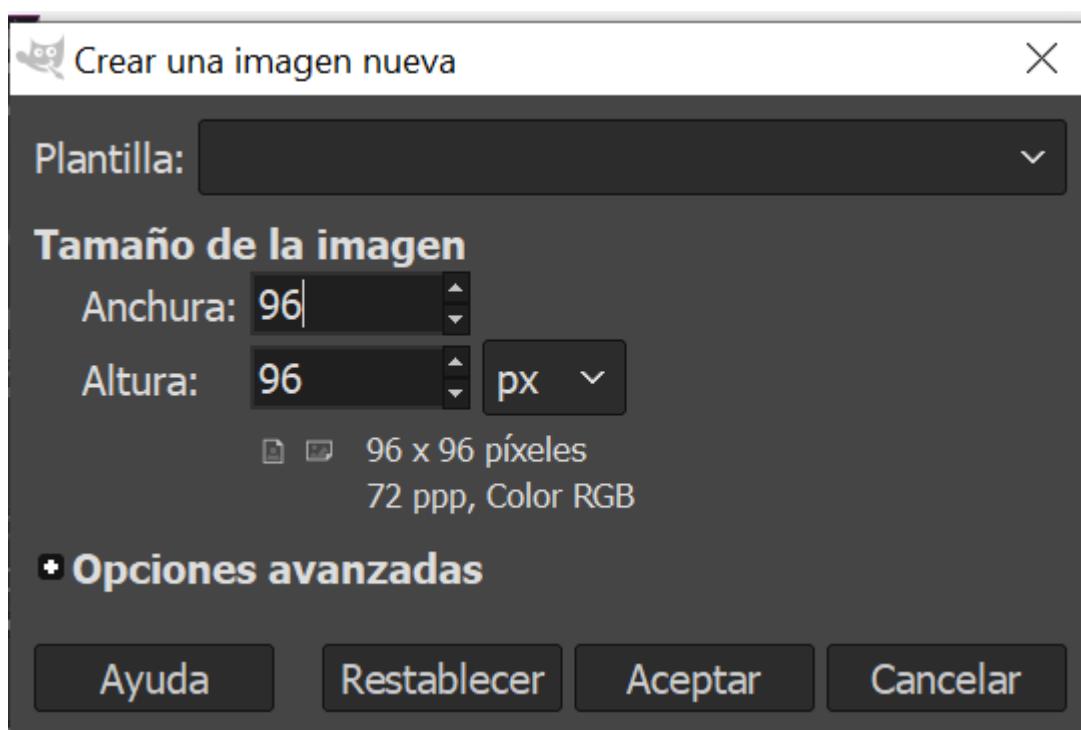
En cuanto al diseño de cada botón, he buscado que todos tuvieran colores distintos unos de otros para que sea más fácil distinguirlos. He considerado fundamental que se comprenda la acción que se va a realizar al pulsar cada uno de ellos con tan solo echar un simple vistazo a la matriz de botones.

Evidentemente, a lo largo de este trabajo han ido modificándose en numerosas ocasiones por distintos motivos. En muchos casos no me gustaba cómo quedaban ciertos colores o no parecían representar claramente la acción a la que estaban asignados.

Como ya he dicho, uno de los grandes puntos a favor de la pantalla que adquirí para este proyecto, es que tiene una gran variedad de posibilidades en cuanto a la representación del color se refiere. Esto me permite jugar con los colores y hacer cualquier diseño que tenga en mente, se van a poder distinguir incluso colores muy similares como los que se observan en la imagen de arriba.[16]

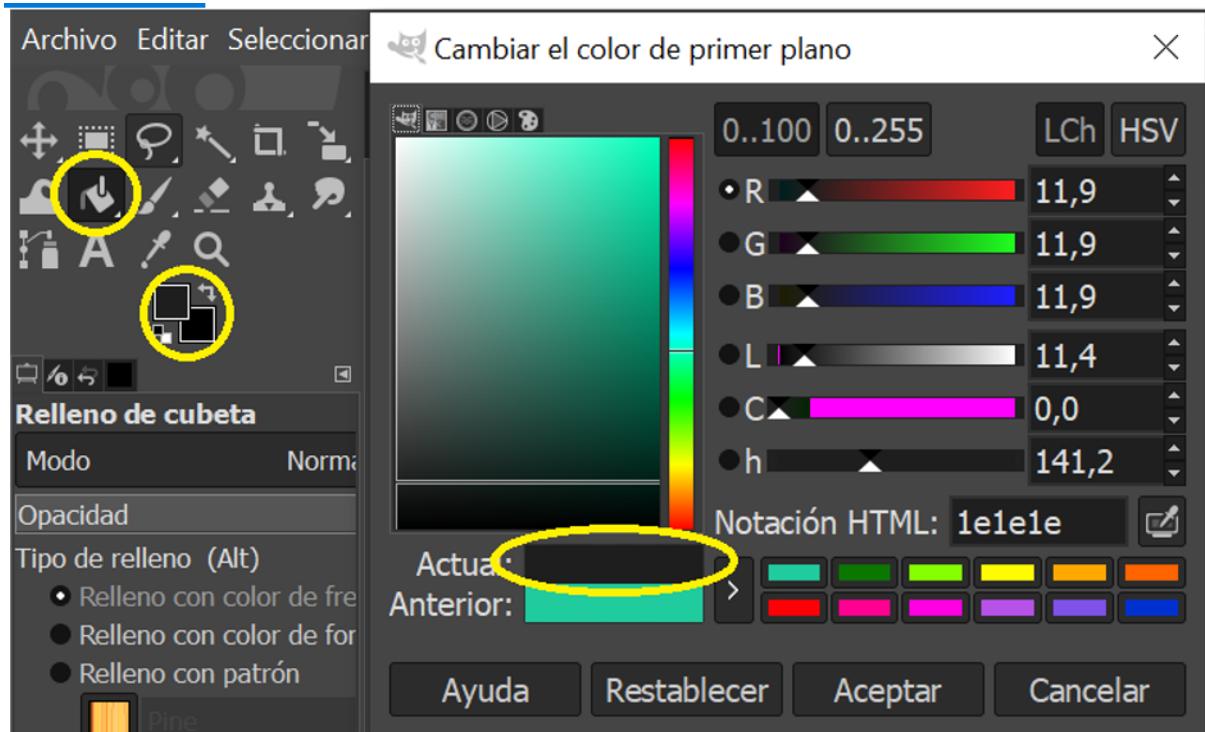
Voy a tratar de mostrar paso a paso cómo he ido realizando cada botón y de esa forma también muestro el software que he utilizado. Este sería el proceso:

1. El primer paso consistía en crear la plantilla con el tamaño adecuado y el fondo gris oscuro.



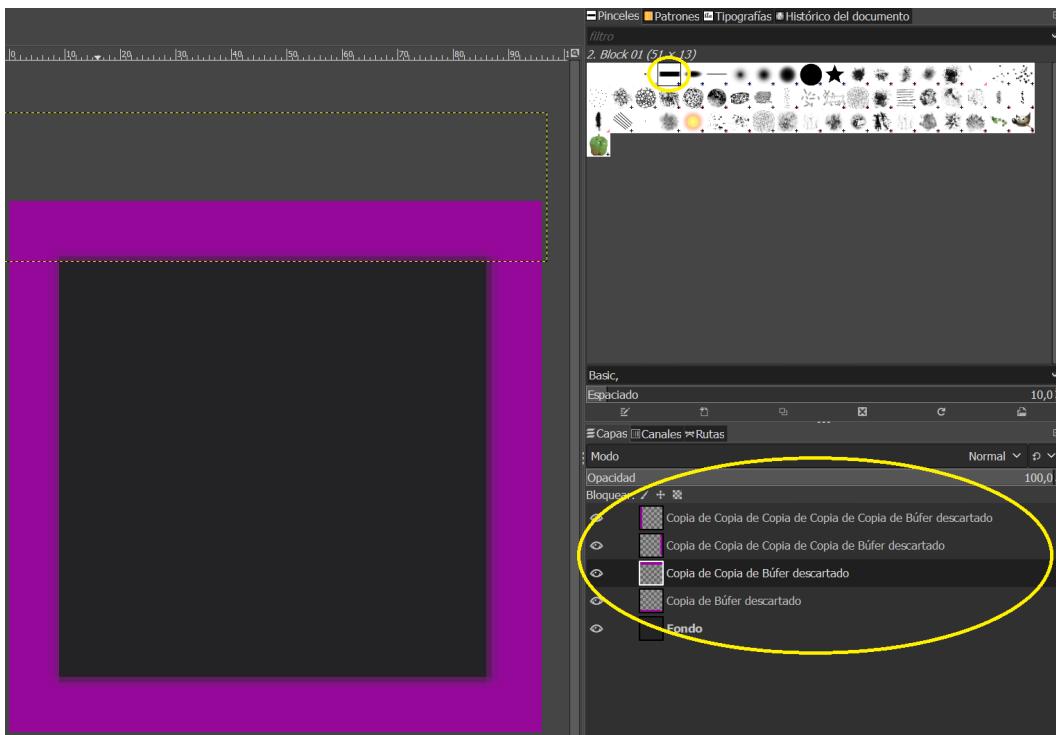
17. Archivo->Nuevo...

2. Para cambiar el color de fondo utilizo la típica herramienta del cubo que permite modificar todo de color en un solo click. Una vez que he seleccionado la opción de rellenar, cambio el color elegido.



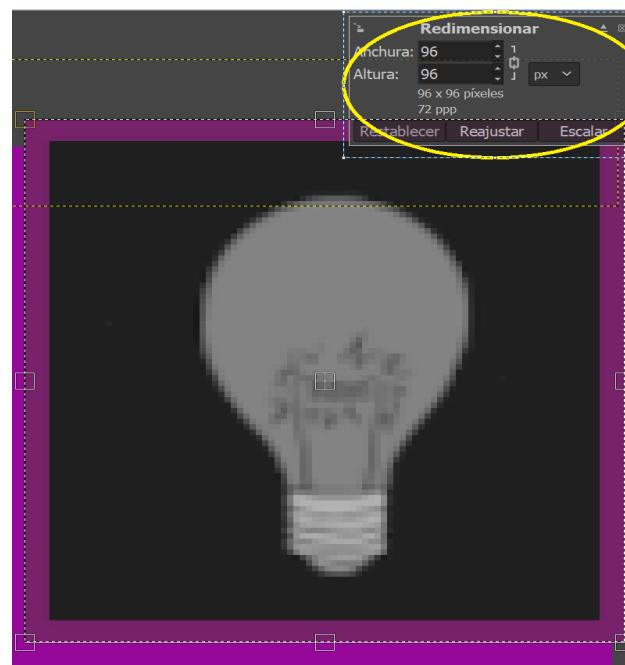
18. Opción de relleno y modificación del color asignado

3. El siguiente paso es la creación del marco morado en los bordes. Para ello, selecciono un rectángulo y le cambio el color por el deseado. Una vez esté a mi gusto, comienzo a replicar el rectángulo tantas veces como paredes tenga el botón y lo voy situando de manera que luzca como un marco.



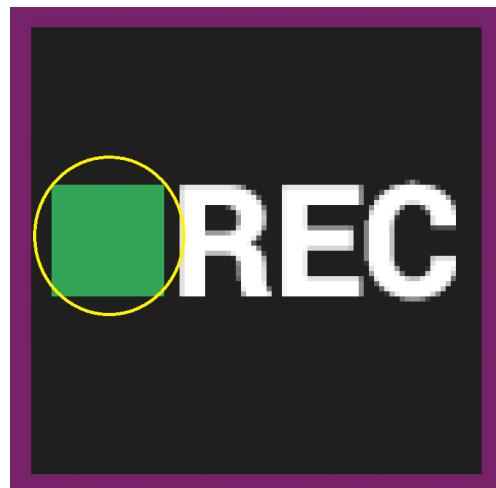
19. Selecciono rectángulo, los clono y les doy el color y la posición deseados

4. Una vez llegado este punto ya depende del botón que desee creo. Ajusto cada ícono al tamaño correspondiente para que se ajuste a los 96x96 píxeles y centro dentro del marco.



20. Redimensión de los objetos y colocación dentro del marco

5. Por último cambio los colores de los objetos o añado detalles para mejorar el botón.



21. En este caso, añado un cuadrado y le cambio el color a verde

A continuación voy a mostrar un ejemplo de los botones que componen el panel para explicar por qué los he hecho así y qué función están destinados a tener cada uno de ellos. Esta fue una de las configuraciones que hice para el panel, finalmente sufrió modificaciones:



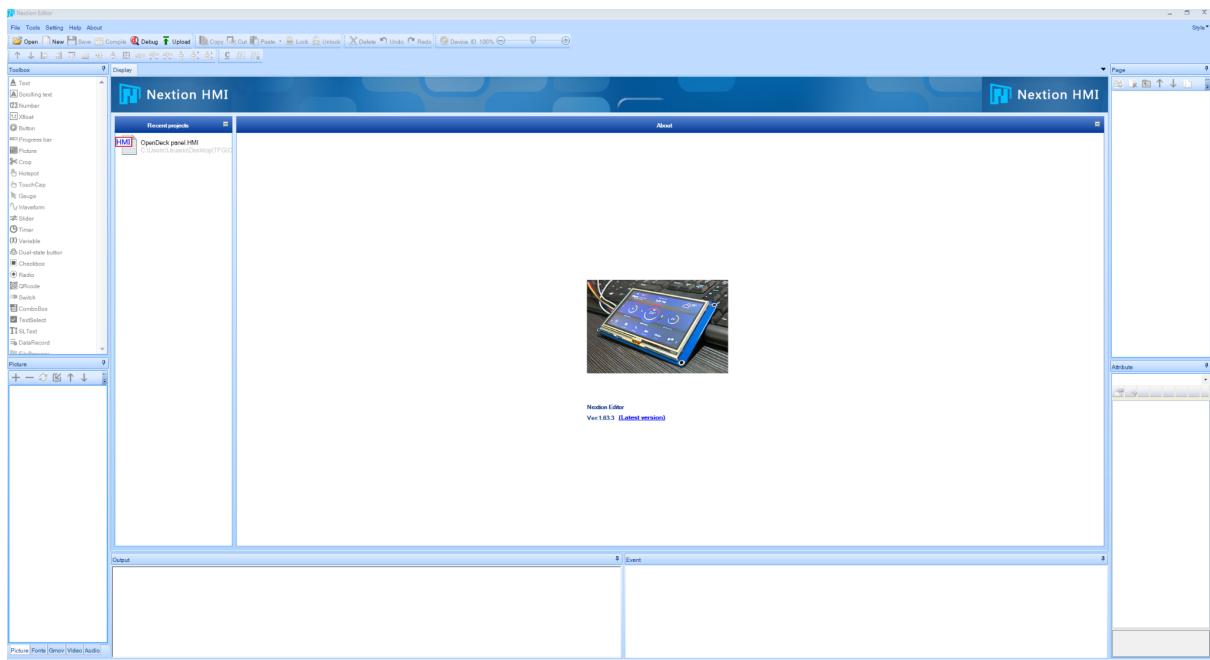
22. Panel de botones ya editados y colocados en la matriz

Como se puede observar, mi idea es que sea un panel de botones muy colorido para que resulte llamativo visualmente. Una vez se van pulsando algunos de estos

botones, también les irá cambiando la apariencia para que sea más intuitivo. Me parecía fundamental que se diferenciarán claramente todos ellos para que no se confundieran las acciones que realiza cada uno.

#### 4.2.2 Programación en Nextion Editor

Una vez ya están creados los botones a mi gusto, el siguiente paso consiste en la programación de dichos botones en el software oficial de la pantalla Nextion. Tuve que estar formándome y viendo vídeos para entender cómo funcionaba este programa y poder ponerme manos a la obra.



23. Vista general de Nextion Editor

El primer paso que tengo que hacer es seleccionar dentro del Nextion Editor la pantalla con la que voy a trabajar, de esa manera se cargará un espacio con las dimensiones correspondientes a dicho dispositivo.

Cuando tenía ante mí el espacio con el que iba a trabajar, me tocaba calcular cuántos botones iba a situar dentro de la pantalla. Viendo las distintas versiones del Stream Deck, me parecía que quince botones serían demasiados y que seis serían

muy pocos. Por lo tanto, acabé decidiendo que doce botones serían los idóneos y los dispuse en tres filas y cuatro columnas.



24. Implementación de los 12 botones del OpenDeck

En la imagen se ve que algunos se llaman btX y otros bX, eso depende de si son botones de un solo estado o de más de uno.

Tras incluir dentro del espacio los botones, importé todas las imágenes correspondientes para poder llenar cada uno de ellos con uno o dos iconos. Los botones de doble estado tendrán asociadas dos imágenes, una para cada estado.

La posibilidad de incluir botones que tienen más de un estado es crucial para el proyecto ya que ayuda al usuario y hace el dispositivo mucho más interesante. Por ejemplo, en el caso del ícono que hace referencia a una cámara web encendida, cuando lo pulsemos cambiará a una imágenes que representa esa misma cámara pero apagada.



25. Botón de activación/desactivación de la webcam

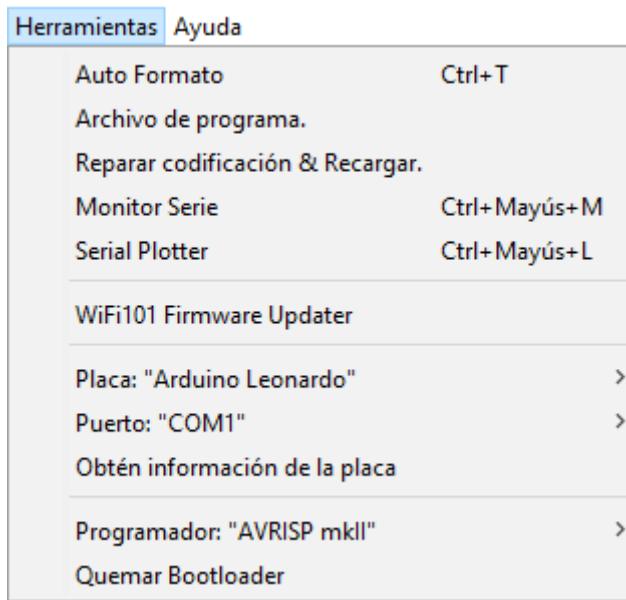
#### 4.3 Codificación en Arduino IDE

El último paso antes de juntar todos estos apartados es la programación del Arduino. Afortunadamente, en la carrera he tenido la oportunidad de utilizar este software en la asignatura Sistemas de Información en Tiempo Real. Eso me facilitó el comenzar a programar ya que conocía el entorno y las cosas principales las tenía aún en mente.

La mayor diferencia es que en este caso iba a estar trabajando con una pantalla táctil y unos botones que tenían que hacer una serie de acciones.

##### 4.3.1 Configuración previa del Arduino IDE

El primer paso fundamental es seleccionar en la pestaña Herramientas la placa de Arduino con la que voy a trabajar, el puerto del PC al que está conectado y el programador que voy a utilizar, en este caso AVRISP mkII. La elección de este se debe a que es compatible con el Arduino Leonardo que es con el que voy a estar programando.



26. Configuración de la pestaña Herramientas en Arduino IDE

Lo último que debe realizarse antes de comenzar a programar es instalar las librerías necesarias para que el software pueda compilar correctamente el Arduino con el que vamos a trabajar. Esto ha sido un pequeño problema ya que depende de la versión del Arduino IDE que utilices o de algunas configuraciones que tengas en el PC. Por suerte, investigando y buscando el problema que tenía, me descargué un paquete que incluía una gran variedad de librerías para evitar cualquier futuro inconveniente.

En cuanto al código completo, se podrá acceder a él en el último apartado de este Trabajo Fin de Grado.

#### 4.3.2 Explicación del código

Lo primero es establecer en la cabecera las librerías que se van a utilizar para este proyecto. Aquí nos topamos con uno de los principales motivos por lo que acabé eligiendo el Arduino Leonardo antes que cualquier otro. La librería Keyboard.h solo es compatible con este Arduino, por lo tanto, si quería que mis botones simulasen pulsaciones de teclas de teclado, debía elegir este debido a su compatibilidad con esta librería.

Establezco los pines del Arduino que voy a utilizar para la pantalla. En este caso, he elegido los pines 10 y 11.

```
SoftwareSerial HMISerial(10, 11);
```

#### *27. Pines de la pantalla*

Ahora llegamos a la definición de los botones de acción. Es imprescindible explicar una característica en la que me he basado para la realización del trabajo, las teclas de función del teclado. Y es que existe una curiosidad sobre estas teclas que muchos usuarios quizás desconocen y es que no solo existen las teclas F1-F12, sino que también podemos llegar a utilizar de la F13 hasta la F24.

```
#define BUTTON_KEY1 KEY_F13  
#define BUTTON_KEY2 KEY_F14  
#define BUTTON_KEY3 KEY_F15  
#define BUTTON_KEY4 KEY_F16  
#define BUTTON_KEY5 KEY_F17  
#define BUTTON_KEY6 KEY_F18  
#define BUTTON_KEY7 KEY_F19  
#define BUTTON_KEY8 KEY_F20  
#define BUTTON_KEY9 KEY_F21  
#define BUTTON_KEY10 KEY_F22  
#define BUTTON_KEY11 KEY_F23  
#define BUTTON_KEY12 KEY_F24
```

#### *28. Definición de los botones de acción*

Esta característica me viene como anillo al dedo para el proyecto ya que mi idea era que al utilizar cualquier botón del OpenDeck, no se viesen afectadas otras funciones del PC. Me explico, en el caso de haber utilizado las teclas F1-12, cuando se pulsase el botón asociado a la tecla F11, esto produciría en muchos software que la ventana se pusiese a pantalla completa. También en el caso de pulsar F5 podríamos provocar algunas modificaciones no deseadas.

De esta manera, me aseguraba que al utilizar cualquiera de los doce botones, únicamente iba a verse afectado el programa con el que OpenDeck estuviese interactuando

Ahora sigo configurando los botones de acción que encontraremos en la pantalla táctil. Recordar que los botones btX tienen doble acción y que los bX son de un único estado. Específico en qué pantalla está cada botón(0), cuál es su id asociada(13-24) y cuál es el nombre que tiene en el Nextion Editor(bt0-bt6, bt11, b7-b10).

```
NexDSButton bt0 = NexDSButton(0, 13, "bt0");
NexDSButton bt1 = NexDSButton(0, 14, "bt1");
NexDSButton bt2 = NexDSButton(0, 15, "bt2");
NexDSButton bt3 = NexDSButton(0, 16, "bt3");
NexDSButton bt4 = NexDSButton(0, 17, "bt4");
NexDSButton bt5 = NexDSButton(0, 18, "bt5");
NexDSButton bt6 = NexDSButton(0, 19, "bt6");
NexButton b7 = NexButton(0, 20, "b7");
NexButton b8 = NexButton(0, 21, "b8");
NexButton b9 = NexButton(0, 22, "b9");
NexButton b10 = NexButton(0, 23, "b10");
NexDSButton bt11 = NexDSButton(0, 24, "bt11");
```

#### *29. Asociación de los botones a su pantalla, id y nombre*

Lo siguiente es la declaración de los eventos Touch, que incluye todos los botones que quiero que tomen los eventos.

```
NexTouch *nex_listen_list[] =
{
    &bt0, &bt1, &bt2, &bt3, &bt4, &bt5, &bt6, &bt11,
    &b7, &b8, &b9, &b10,
    NULL
};
```

#### *30. Declaración de los eventos Touch*

Establezco todas las funciones, una por cada botón del panel. Estas funciones se usarán luego en los botones para que realicen las acciones correspondientes.

```

void bt0_pulsar(void *ptr);
void bt1_pulsar(void *ptr);
void bt2_pulsar(void *ptr);
void bt3_pulsar(void *ptr);
void bt4_pulsar(void *ptr);
void bt5_pulsar(void *ptr);
void bt6_pulsar(void *ptr);
void b7_pulsar(void *ptr);
void b8_pulsar(void *ptr);
void b9_pulsar(void *ptr);
void b10_pulsar(void *ptr);
void b11_pulsar(void *ptr);

```

### *31. Funciones de los botones*

Programo el setup que se ejecutará una única vez cuando se conecte el Arduino. En el setup asocio los botones a su correspondiente función. Posteriormente incluyo un loop para que esté todo constantemente ejecutándose y escuchando atentamente el puerto de comunicaciones por si se pulsa un botón.

```

void setup() {
    Keyboard.begin();
    Serial.begin(9600);
    bt0.attachPop(bt0_pulsar, &bt0);
    bt1.attachPop(bt1_pulsar, &bt1);
    bt2.attachPop(bt2_pulsar, &bt2);
    bt3.attachPop(bt3_pulsar, &bt3);
    bt4.attachPop(bt4_pulsar, &bt4);
    bt5.attachPop(bt5_pulsar, &bt5);
    bt6.attachPop(bt6_pulsar, &bt6);
    b7.attachPop(b7_pulsar);
    b8.attachPop(b8_pulsar);
    b9.attachPop(b9_pulsar);
    b10.attachPop(b10_pulsar);           void loop() {
    b11.attachPop(bt11_pulsar, &bt11);   nexLoop(nex_listen_list);
    HMISerial.begin(9600);              delay(5);
}
}

```

### *32. Setup y loop*

Por último, mostraré cómo está programado cada botón según si es de un único estado o si es de dos acciones. En el caso de un botón de doble acción, el código sería el siguiente:

```

void bt0_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt0.getValue(&EstadoBoton);

    if(EstadoBoton) {
        Keyboard.press(BUTTON_KEY1);
        delay(500);
        Keyboard.release(BUTTON_KEY1);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY1);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY1);
    }
}

```

*33. Botón de doble acción*

En el caso de pulsar el botón 0, teclea F13. En el caso de que no se encuentre activado el botón, teclará SHIFT+F13. Esto se utiliza para diferenciar los dos estados del botón, F13 para una cosa y SHIFT+F13 para otra.

Cuando se trata de botones de un único estado, no importa el estado del botón ya que no hay que diferenciar acciones, solo realizará una.

```

void b8_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY9);
    delay(500);
    Keyboard.release(BUTTON_KEY9);
}

```

*34. Botón de estado único*

Como ya he comentado, en este caso cuando se pulse el botón 8, se tecleará la tecla F correspondiente independientemente de si el botón ya había sido pulsado antes. Como no necesita diferenciar dos acciones, tan solo se va a utilizar la tecla F que tenga asociada el botón.

Estas configuraciones se siguen repitiendo en el código hasta completar la configuración de cada uno de los botones. El punto positivo es que aunque los

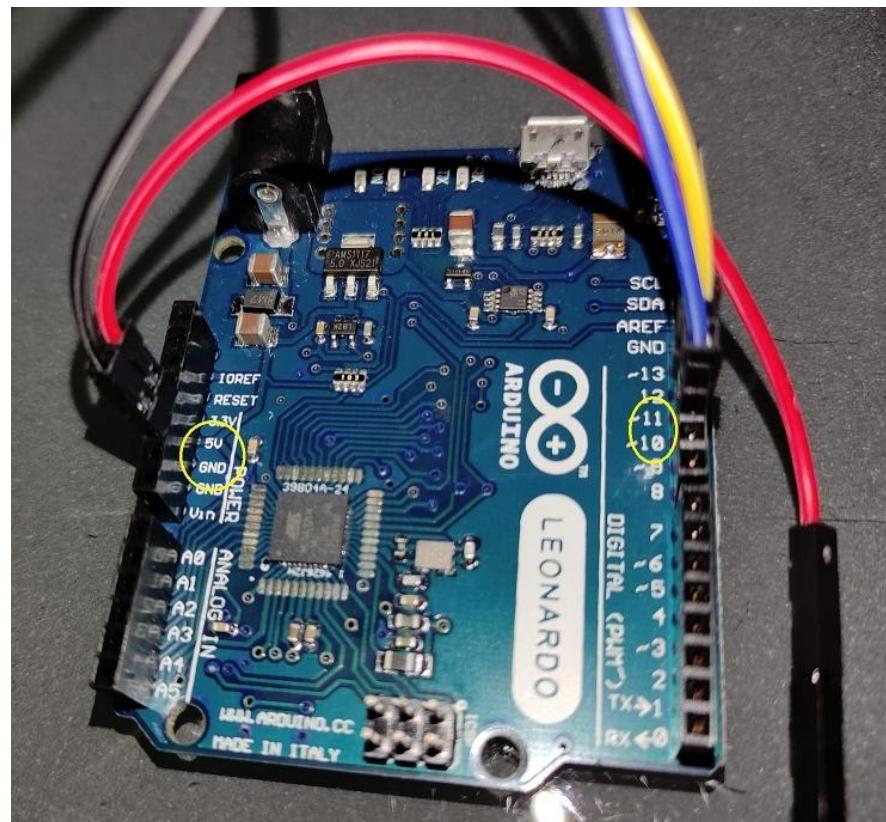
botones tengan acciones diferentes asociadas, el código es prácticamente el mismo en todos ellos.

## 5. IMPLEMENTACIÓN

Después de haber realizado y explicado el código del Arduino IDE, es momento de unir todo con lo que se ha estado trabajando y comprobar que las conexiones entre todos los componentes funcionan.

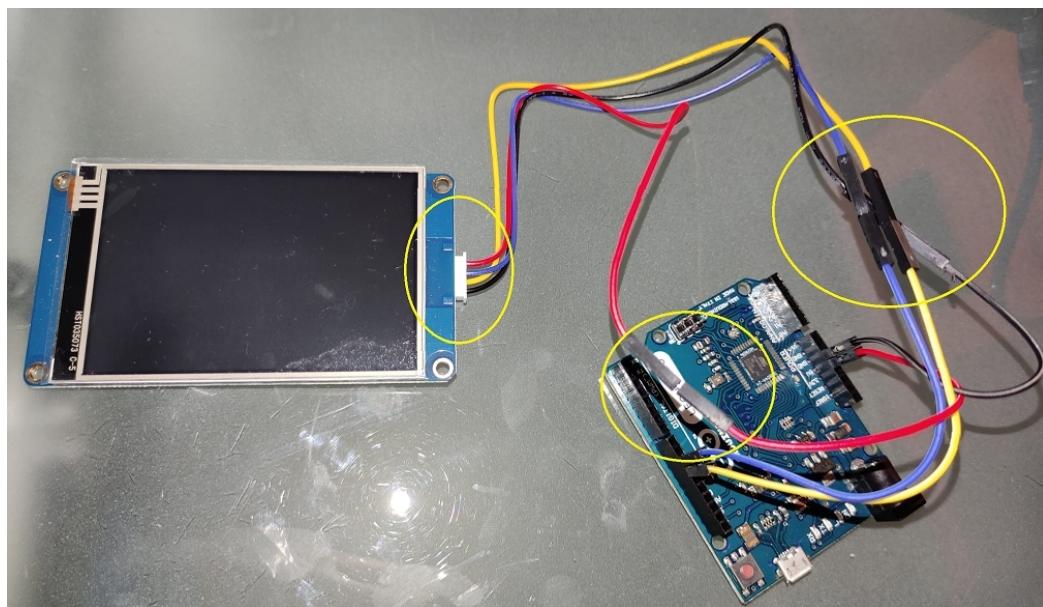
Durante este proceso iré mostrando imágenes de los pasos que he ido siguiendo para poner todo a punto:

1. Lo primero que hice fue poner los jumpers en los pines correspondientes del Arduino. Debía colocarlos en la toma de tierra, 5V y los pines 10 y 11 que ya enseñé anteriormente en el código.



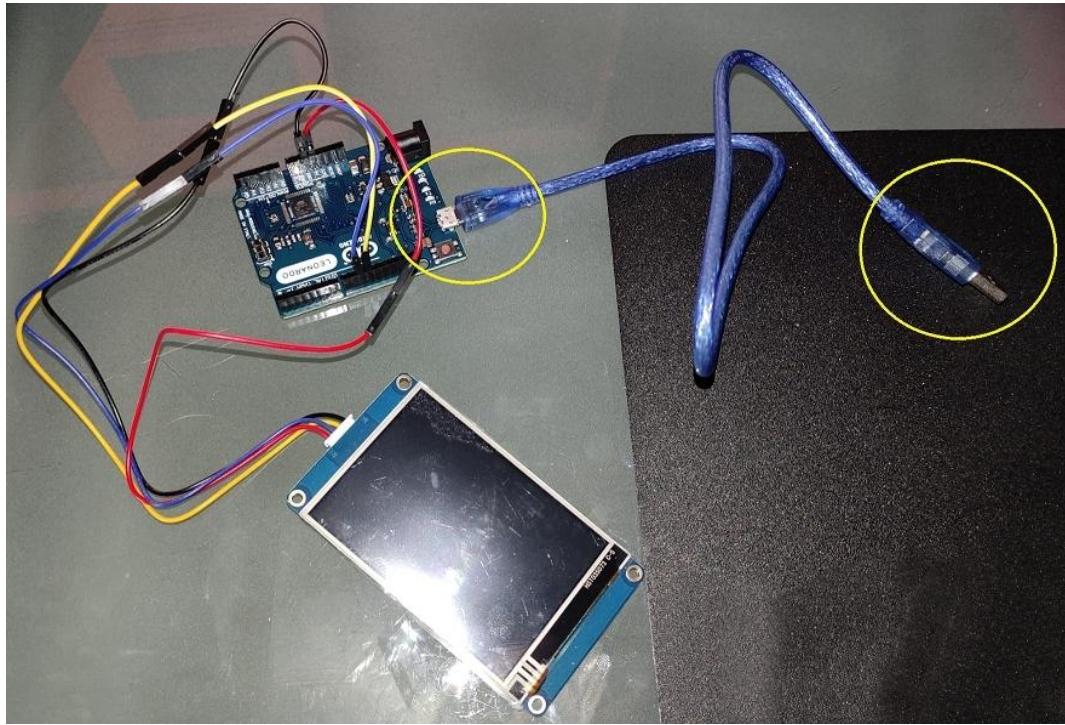
35. Colocación de los jumpers en el Arduino Leonardo

2. El siguiente paso es conectar estos jumpers a los de la pantalla Nextion. Para ello, estoy utilizando cables de puente macho-macho y de 10 cm de longitud. El motivo para usarlos de esta medida en vez de 20 cm que es lo habitual, es para que no ocupe demasiado espacio una vez esté todo montado.



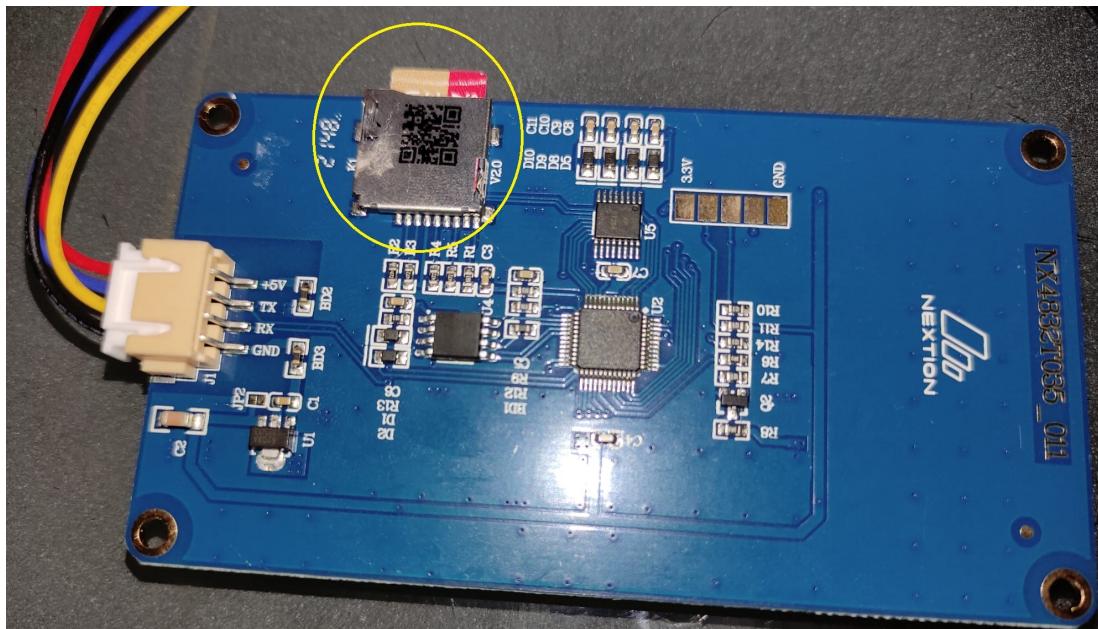
36. Conexión mediante los jumpers entre el Arduino Leonardo y la pantalla Nextion

3. Ahora debo conectar el cable USB al Arduino para poder enchufarlo a un PC y que así tanto el Arduino como la pantalla reciban energía. Mi idea es utilizar un cable fino y no demasiado largo para que no resulte muy engorroso.



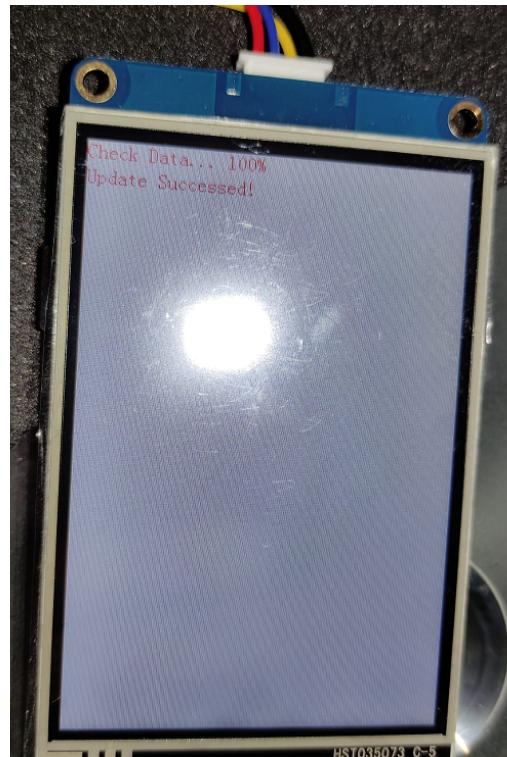
37. OpenDeck ya montado a falta de la carcasa 3D

4. Antes de instalar la carcasa, hay que cargar en la pantalla el panel de botones que hice en el Nextion Editor. Es muy sencillo, guardo en el editor una vez que tengo la matriz de botones deseada y se genera un archivo. Ese archivo lo meto en una microSD vacía y la introduzco en la pantalla.



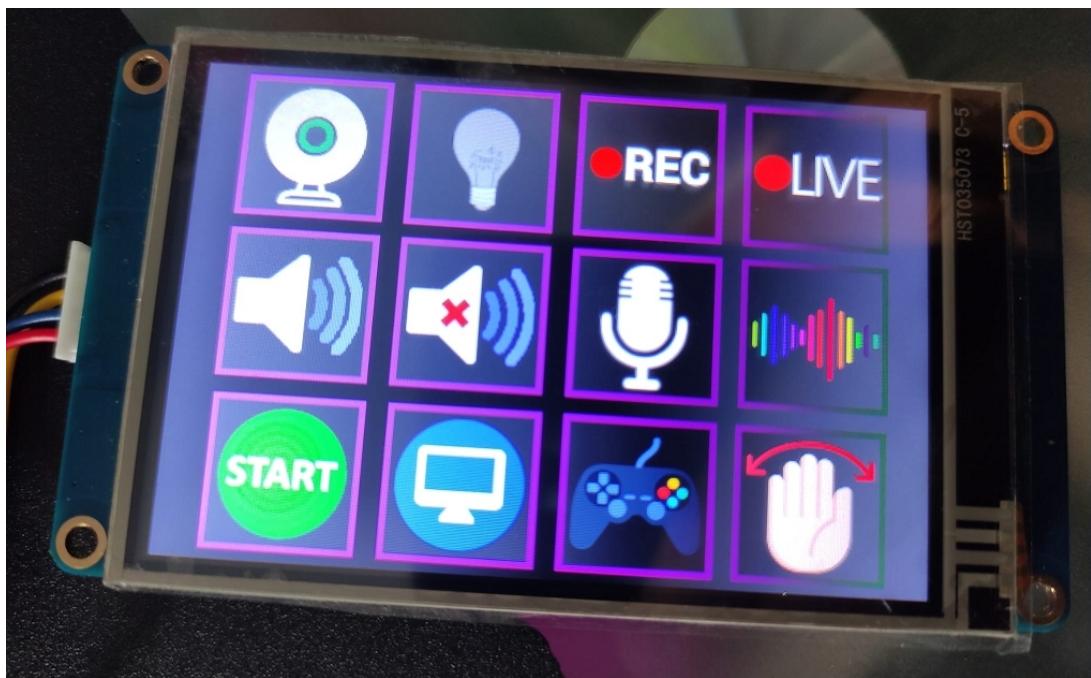
38. Instalación de la microSD en la pantalla

5. Con la microSD conectada, se van a cargar los datos en la pantalla una vez esté conectado el USB a un PC. Será un proceso rápido, de apenas unos 8-10 segundos. Se vería en la pantalla algo así:



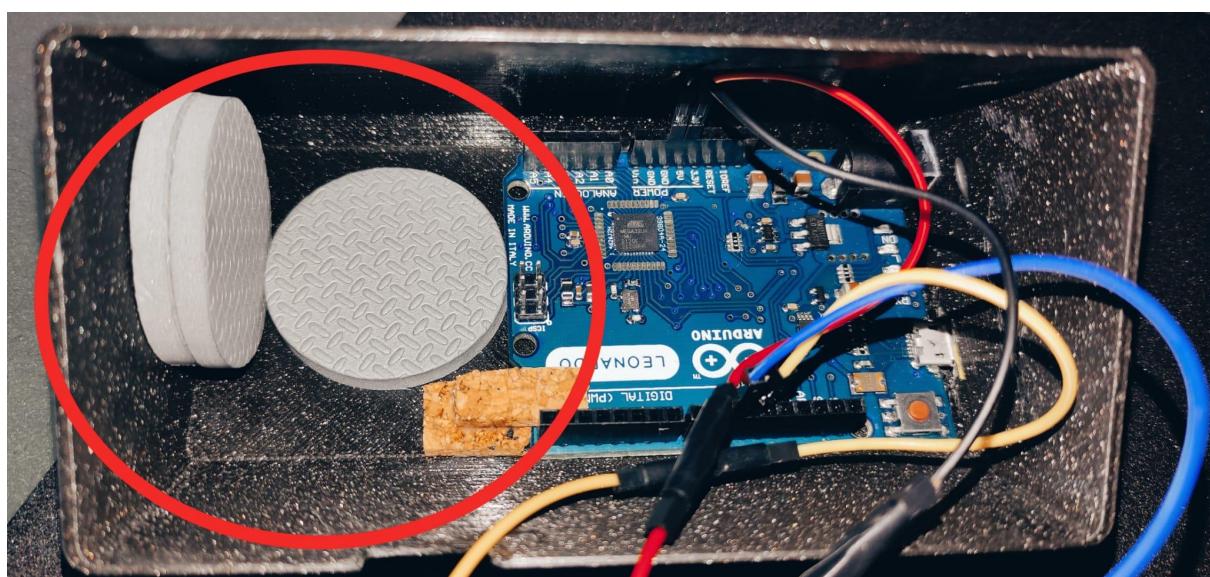
39. Carga de datos en la pantalla

6. Una vez se ha completado la carga, retiro la microSD y vuelvo a conectar el USB para comprobar que ya está cargado el panel de botones en la pantalla.



40. Pantalla mostrando el panel de botones cargado

7. Una vez llegados a este punto, tan solo resta colocar todos los componentes dentro de la pequeña carcasa. Tras esto, podría darse por finalizada la implementación de todas las partes que componen este trabajo. El círculo rojo está marcando algunas de las modificaciones finales que le realicé a la carcasa para mejorar el diseño y funcionamiento. La idea de incluir esos detalles es para evitar que el Arduino Leonardo se mueva a la hora de conectar el cable por el orificio derecho que mostraré dentro de un momento.



*41. Arduino y jumpers dentro de la carcasa*

He intentado que el resultado final fuese lo más elegante posible. El material del que está hecha la base sobre la que se sostiene la tapadera con la pantalla contiene brillo mediante purpurina para aportar más contraste a la carcasa en general.



*42. Opendeck con la carcasa puesta*

A continuación mostraré las distintas modificaciones que le he ido realizando a la carcasa del OpenDeck con la idea de mejorar el dispositivo y hacerlo lo más sencillo y útil para el usuario posible.

Le realicé un orificio con la ayuda de un soldador en la parte baja derecha de la base. Este hueco servirá para conectar el Arduino a la corriente sin necesidad de tener que interactuar con nada del interior de la carcasa.



43. Orificio para conectar el OpenDeck a la corriente

Otro de los orificios realizados fue uno que se encuentra debajo de la pantalla. De esta manera se puede introducir y retirar la tarjeta SD sin necesidad de retirar la pantalla del dispositivo. Todos estos cambios los hice con la intención de que el dispositivo quede mucho más recogido y no haya que interactuar con ninguno de los componentes más de lo necesario.



*44. Orificio para conectar la SD en la pantalla*

Para mejorar la estabilidad del OpenDeck, decidí añadirle unos pequeños soportes de goma en la base. Con estos soportes, quería conseguir que no se deslizase con tanta facilidad a la hora de interactuar con el aparato. Buscaba que no desencajese con respecto al resto del diseño del dispositivo.



*45. Soportes para evitar el deslizamiento*

## 6. FUNCIÓN DE LOS BOTONES

Antes de comenzar a mostrar el funcionamiento del OpenDeck, se debe recordar que es fundamental cada vez que conectemos el dispositivo, subir el código del Arduino IDE al Arduino Leonardo. Una vez esté cargado todo en el Arduino, empezará a funcionar.

Como ya he estado comentando durante todo el trabajo, para desarrollar este sistema me he estado centrando en su uso para el software OBS. Lo importante es que a pesar de estar testado en este programa, se puede utilizar en cualquier otro que cuente con atajos de teclado personalizables.

Los pasos exactos a seguir uno a uno que deben seguirse para que el OpenDeck pueda empezar a usarse los detallaré en el manual de usuario más adelante. Ahora me voy a centrar exclusivamente en en qué es lo que puede llegar a hacer y cómo va actuando el dispositivo según se va interactuando con el mismo. Esto implica que se va a dar por hecho que ya se tiene cargado el código de Arduino y convenientemente configurado el OBS para el uso de este aparato.

Una vez tenemos el OpenDeck conectado al PC, se enciende la pantalla y podemos comenzar a interactuar con ella. Voy a explicar paso a paso qué hace la interacción con cada uno de los 12 botones que componen el dispositivo del que trata este proyecto dentro del software OBS.

Intentaré ilustrarlo mediante imágenes para que sea más sencillo de entender:



46. Botón para activar o desactivar la webcam

Se entiende que lo más lógico es que la cámara esté en un principio desactivada para evitar cualquier tipo de inconveniente. Es por ese motivo que en el panel de botones, el primero que se observará es el de la izquierda, de esa manera, al pulsar en dicho botón se activará la cámara y se modificará la imagen por la que podemos ver en la derecha.



47. Botón para apagar o encender la escena

Este botón permite que la escena en la que se encuentre el usuario se oculte de manera que se ponga todo en negro. Se suele utilizar para ocultar cualquier cosa al

espectador o simplemente para empezar un directo o vídeo a oscuras y luego mostrar la escena deseada. La imagen de la izquierda es la que se mostrará predeterminadamente, al pulsarla se verá reemplazada por la de la derecha. Esta segunda imagen volverá a mostrar la escena en la que se encuentre el usuario.



48. Botón para comenzar o finalizar la grabación

Uno de los botones más usados junto con los dos siguientes. Esto se debe a que al momento de grabar algo, siempre está la duda de si se ha pulsado la tecla correcta o el atajo asignado. Dentro del software OBS, están muy juntas las opciones de comenzar grabación y comenzar retransmisión, eso en ocasiones puede ser problemático.



49. Botón para comenzar o finalizar la retransmisión

Como ya he comentado, este botón resuelve una de las equivocaciones más comunes a la hora de realizar grabaciones o directos. Consideraba fundamental que se detectase fácilmente la diferencia entre estos cuatro botones y las distintas variantes de los mismos.



50. Botón para activar o desactivar el sonido

Botón que permite activar o desactivar el sonido del escritorio del PC. En muchos casos simplemente se busca que se escuche la música del directo o el micrófono del creador de contenido y nada más. También en ciertas escenas, este sonido es el menos interesante y quizás lo más importante es lo que se escuche en un juego o aplicación determinados.



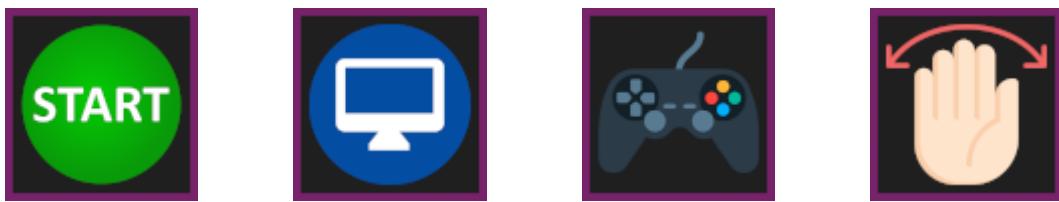
51. Botón para activar o desactivar el micrófono

Esta es una de las acciones más populares, el muteo o activación del micrófono del streamer. De forma predeterminada, el primer botón será el de la izquierda para que no se cuele ningún sonido o conversación no deseados. Una vez pulsado el botón, se modificará por la segunda imagen, lo que permitirá reactivar el sonido del micrófono.



52. Botón para activar o desactivar la música

En muchos casos el creador de contenido incluye una lista de reproducción o cualquier tipo de sonido que se usa para dotar al directo o al vídeo de cierta atmósfera deseada. Este botón permite la activación del sonido de fondo y, mientras está siendo pulsado, se mostrará la imagen que podemos ver a la derecha. Esto lo programé así para que podamos observar distintas formas de representar botones de doble acción.



53. Botón para cambiar entre las distintas escenas

La utilidad de estos cuatro botones es la de cambiar de una escena a otra con tal sólo la pulsación del botón deseado. Estas serían las cuatro escenas elegidas para este proyecto:

1. El primer botón a la izquierda, cambia a la escena utilizada para el comienzo del directo, puede tratarse de algún tipo de vídeo en bucle o imagen.
2. El segundo botón, muda a la escena donde se mostrará el contenido relacionado con el PC, ya sea una captura de la pantalla o algún programa.
3. El tercer botón, hace la transición a la escena utilizada para la retransmisión o grabación de un videojuego
4. El último botón, transita a la escena usada para despedir el vídeo o directo. Se suelen utilizar imágenes o vídeos pero también se muestra en ocasiones un recuadro donde se pueden ver los comentarios del chat en directo. Esto permite al streamer leer los últimos mensajes de sus seguidores y despedirse de ellos.

## 7. TESTEO

Para terminar de comprobar que todo cumplía con las expectativas esperadas, realicé una serie de pruebas básicas que pusiesen en funcionamiento el OpenDeck. Conecté el dispositivo tanto en mi ordenador portátil como en el PC de mesa, de esa forma me aseguraba que funcionaba indistintamente del ordenador que se utilizase.

Lo más importante es tener una versión de OBS actualizada para que no haya problemas a la hora de usar los atajos de teclado. Una vez tienes configurados dichos atajos y cuentas con la versión del software adecuada, todo funciona sin ningún tipo de inconveniente.

Lo que hice fue ponerme en la situación de un creador de contenido que va a grabar un video o va a realizar un directo. Comencé a probar todos los botones uno a uno y comprobando que realizaban lo que les correspondía. Testeé también el comienzo y detención de una grabación o un directo para ver si todo se producía con normalidad.

Tras realizar estas pruebas, di por finalizado este proceso. Afortunadamente, OpenDeck no me ha fallado en ningún momento, todo ha funcionado desde un principio y no he tenido que hacer muchas modificaciones más allá de los cambios estéticos que ha podido ir sufriendo el dispositivo.

## 8. PROGRAMA DE PLANTILLAS DEL OPENDECK

Una vez cumplí con los objetivos principales de este proyecto, se me ocurrió que podría hacer un pequeño programa que permitiese descargar algunas plantillas de botones y su respectivo código. De esta manera, el usuario que quiera utilizar el OpenDeck y no desea editar botones o no tiene el conocimiento necesario para hacerlo, tan solo tiene que descargar los archivos y realizar la instalación correspondiente. Decidí hacerlo con el entorno de desarrollo NetBeans y usando JFrame.

NetBeans IDE proporciona herramientas de clase mundial para desarrollar aplicaciones Java móviles, de escritorio, corporativas y web. Este es el primer IDE compatible con las últimas versiones de JDK, Java EE y JavaFX. Proporciona un modelo inteligente para ayudarlo a comprender y administrar sus aplicaciones, incluido el soporte incorporado para tecnologías populares como Maven.

En cuanto a JFrame, es una de las herramientas más conocidas para crear aplicaciones GUI independientes. Aparece y actúa como una ventana que normalmente aparece en la pantalla del PC, como una ventana de notificación o una ventana de alerta.

Al igual que JPanel, es parte del conjunto de herramientas Swing, pero su clase principal es java.awt.Frame. Esto significa que es una versión extendida del marco de API de Java más antiguo en Java Abstract Window Toolkit (AWT). Lo que hace que JFrame sea mejor que Frame y eso se debe a que proporciona una opción para cerrar u ocultar la ventana mediante el método setDefaultCloseOperation(int).

Tiene constructores y métodos para colocar elementos como cuadros de texto, botones, bordes, barras de título, etc. Edita y personaliza sus características así como fuente, tamaño, color y alineación. Cada función tiene su propia sintaxis para la personalización.

Tiene dos subsecciones, una barra de menú y una tabla de contenido. Los elementos de JFrame se denominan activos y la mayor parte del contenido se encuentra en el panel de contenido.

JFrame permite realizar las siguientes funciones: activar, cerrar, abrir, minimizar o maximizar una ventana. También utiliza un detector de ratón para que la plataforma pueda responder a las acciones del mismo.

JFrame puede contener muchos marcos y JPanels, pero todos ellos dependen de la presencia del mainframe. No solo el método Listener, sino también los métodos get, set y add se pueden usar para crear una gran cantidad de funciones para el JFrame.

Así se vería la ejecución del programa:



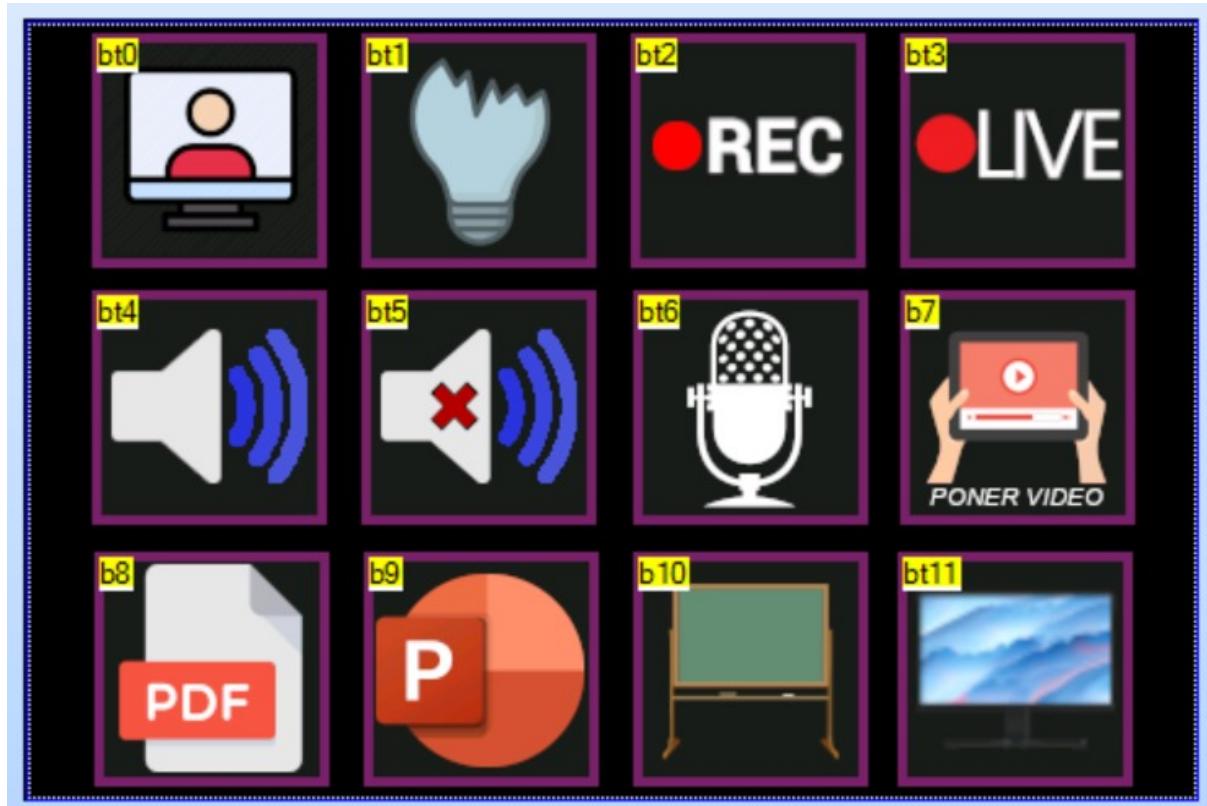
54. Vista general del programa con JFrame

Mi idea era que existieran al menos cuatro plantillas. Una de ellas sería la de Streamer #1, que consiste en el conjunto de botones que se ha estado mostrando durante todo este trabajo. Pero, además de esta plantilla principal, quise añadir una alternativa llamada **Streamer #2**, cuyos botones serían los siguientes:



55. Panel de botones de Streamer #2

También una plantilla para que un docente pueda impartir sus clases mediante el software que desee. Para ello, tenía que crear y editar botones que se ajusten al tipo de contenido que un tutor puede necesitar para su enseñanza. Estos son los botones que yo he editado para un **Profesor**:



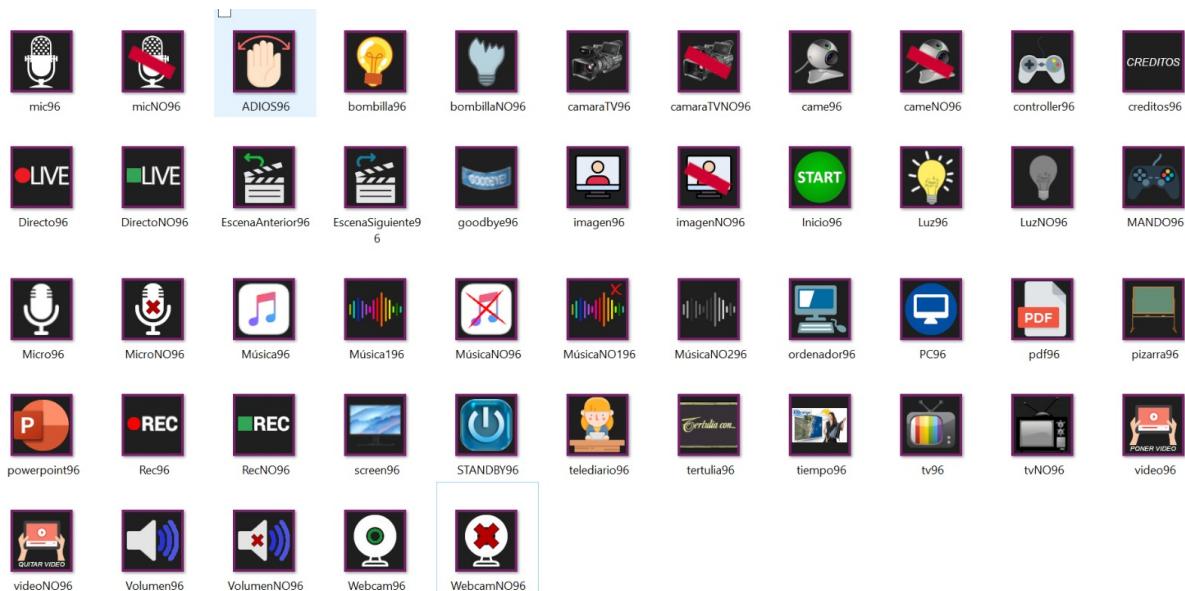
56. Panel de botones de Profesor

Por último, la plantilla que podría utilizar un **productor televisivo**. Creo que este mundo de las retransmisiones en directo mediante internet puede ser el futuro del entretenimiento. Es por eso que considero que esta plantilla puede ser de gran utilidad para que este tipo de contenido se siga practicando y siga creciendo. La plantilla desarrollada sería la siguiente:



57. Panel de botones de Productor TV

Existe una última opción de descarga que consiste en un conjunto de todos los botones observados en las cuatro plantillas que he mostrado y algunos más extra. A continuación muestro el contenido de la **galería de botones**:



58. Galería de botones

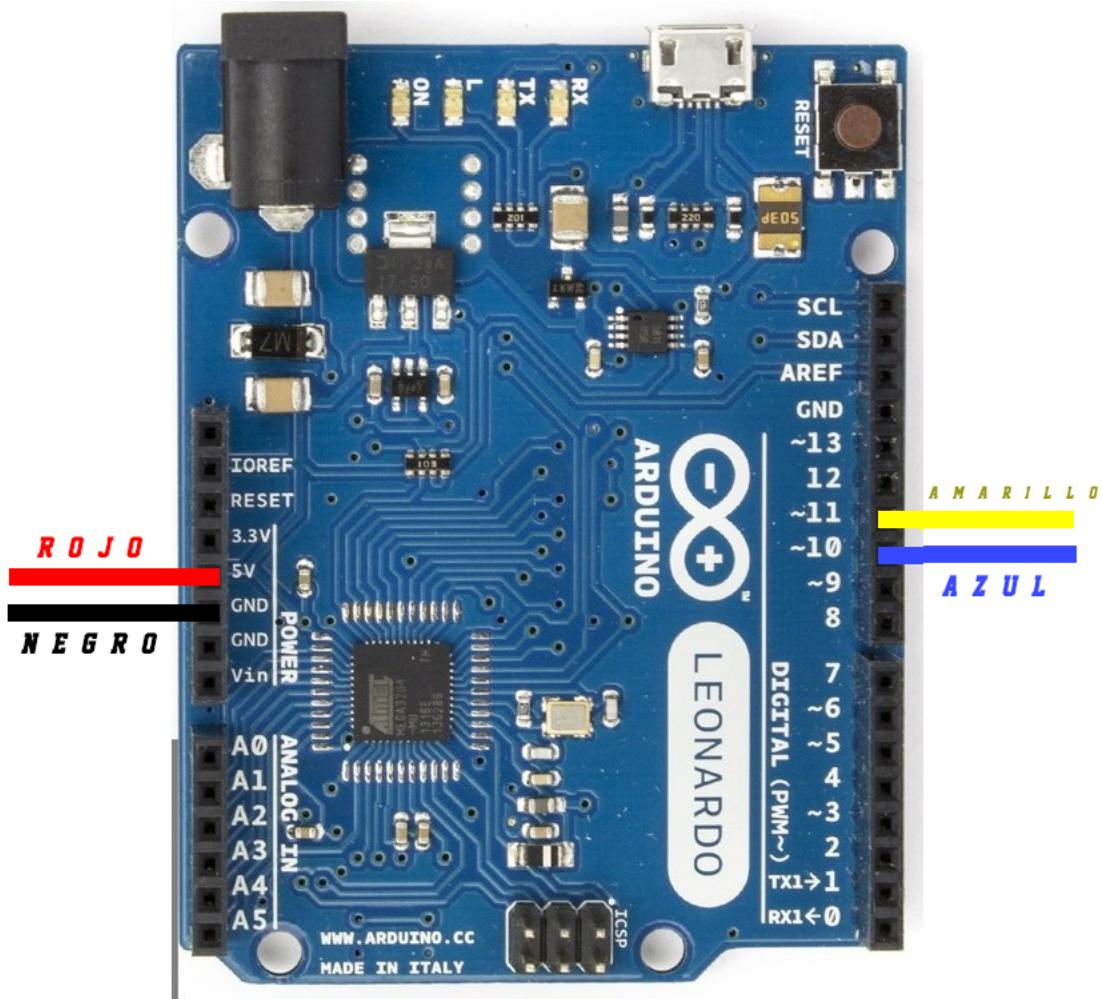
## 9. MANUAL DE USUARIO

Este es uno de los apartados más importantes de todo el proyecto, las instrucciones de los pasos que deben seguir los usuarios para poner en funcionamiento el OpenDeck. Intentaré que cada uno de los apartados a seguir queden totalmente claros y que no haya ninguna duda.

Una de las cosas fundamentales de este dispositivo es que se busca la sencillez en todos los aspectos. Para ponerlo en marcha se necesitan seguir pocos pasos y todos ellos bastante sencillos desde mi punto de vista.

### 1) Carga del panel de botones en la pantalla.

- a) El usuario contará con un archivo **.tft** que deberá introducir en una **microSD** de cualquier tipo, se recomienda formatear la tarjeta con el formato **FAT32(pero no es necesario)**.
- b) Lo más cómodo en este punto es conectar ya la pantalla al **Arduino Leonardo**. Esto se realizará de la siguiente manera para que no nos encontramos con ningún problema, debemos conectar los jumpers de la pantalla a otros **jumpers tipo macho-macho** para ponerlos en los pines correspondientes del Arduino. Los pines correspondientes serían los siguientes: **cable rojo en 5V, cable negro en GND, cable amarillo en 11 y cable azul en 10**. Dejo una ilustración para que no hay duda:



59. Imagen para explicar la colocación de los jumpers en el Arduino Leonardo

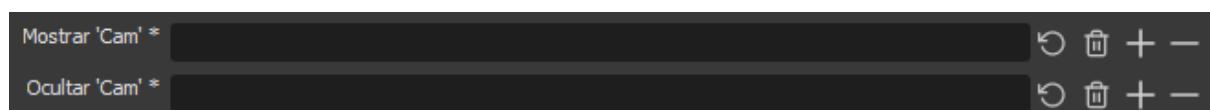
- c) A continuación, se debe **conectar la pantalla a la corriente**, puede hacerse mediante el pequeño **adaptador** que se incluye con el OpenDeck o directamente conectando el **Arduino** a un **PC**. Sinceramente creo que lo más cómodo es hacerlo mediante Leonardo, ya que en el paso anterior hemos conectado ya la pantalla al dispositivo.
- d) Una vez la pantalla haya cargado los datos de la **microSD**, puede desconectarse la pantalla de la corriente. Se **retira la tarjeta** de la pantalla y ya estaría el panel de botones cargado. Cada vez que se quieran **cambiar los botones** que se muestran por pantalla se debe realizar el **mismo proceso**.

## 2) Subida del código al Arduino Leonardo.

- a) Se ejecuta el archivo .ino que contiene el código del Arduino. Una vez se ha abierto y cargado el programa, se debe escoger en la pestaña de **Herramientas** la placa “**Arduino Leonardo**”, el **puerto** correspondiente en el que ha sido conectado el Arduino y el **programador “AVRISP mkII”**.
- b) Tras esta configuración previa, se selecciona la opción **Subir**. Esto **cargará el código del Arduino IDE al Leonardo**.

## 3) Configuración de OBS.

- a) Lo primero es disponer de una **versión actualizada** de Open Broadcaster Software(**OBS**).
- b) Cada **usuario creará** las escenas que crea oportunas y usará los dispositivos, opciones de volumen y distintas configuraciones según lo que necesite.
- c) Una vez esté todo listo, se debe acceder a **Ajustes->Atajos**. Dentro de este apartado se debe asignar **cada acción al botón deseado**, por ejemplo, se selecciona el atajo de Mostrar ‘Cam’ y se pulsa en la pantalla táctil el botón correspondiente para que quede asignado(en este caso le asignará la tecla F13).



60. Atajos de activación/desactivación de la WebCam

## 4) Uso del programa para descarga de ficheros.

- a) Accede al programa y **ejecútalo**.

- b) Se mostrará una **ventana** con las distintas posibles **descargas** según lo que el usuario necesite. Pulse en **descargar** donde desee.
- c) Una vez tenga el archivo **.zip**, debe **descomprimirlo**. Tendrá un fichero **.ino** y un **.tft**.
- d) El **.tft** debe introducirlo en la pantalla Nextion siguiendo el **paso 1**).
- e) El **.ino** debe cargarlo en el Arduino IDE y subirlo al Leonardo tal y como se ha indicado en el **paso 2**).

## 10. CONCLUSIONES

Este proyecto ha significado para mí el poner el broche final a mi carrera. Este desenlace lo he querido hacer realizando un Trabajo de Fin de Grado sobre un tema que conociese y que me llamase la atención, buscando también que no fuese un trabajo distinto a los habituales.

Me gustaba la idea de tratar con bastante hardware, la improvisación y diseño del panel de botones, la impresión 3D de la carcasa del OpenDeck, la creación del logotipo del dispositivo y la programación de los botones en Arduino. Todo esto ha formado parte de un proceso de estudio e investigación que me ha llevado meses de trabajo y varias comeduras de cabeza hasta llegar a definir lo que quería, cómo lo quería y qué iba a necesitar para conseguirlo.

Afortunadamente he contado con mucha información referente a los distintos componentes del OpenDeck ya que he utilizado elementos fáciles de conseguir y conocidos por la comunidad. He intentado darle mi propio toque y punto de vista buscando lo más sencillo, práctico y flexible posible.

Creo con sinceridad que he cumplido con las expectativas que tenía en un principio. He conseguido realizar un documento donde he concentrado toda la información

relativa al OpenDeck y todo lo que le rodea de una forma amena. También he incluído cosas que en un principio no tenía en mente pero que han acabado mejorando este trabajo.

Por último, la inclusión de una aplicación de escritorio con plantillas de botones para que los usuarios que utilicen este dispositivo puedan usar otras configuraciones. Esto último me parece un detalle extra que puede ser muy útil para gente que no esté familiarizada con la edición de imágenes o que simplemente no tengan ganas de ponerse a trastear con todos estos programas y solo quieran usar el OpenDeck lo antes posible.

Espero que toda aquella persona que haya llegado hasta aquí leyendo este documento lo haya disfrutado y ahora tenga una idea más clara de las distintas utilidades que puede tener en la actualidad.

## **11. BIBLIOGRAFÍA**

- [1] <https://brandme.la/blog/10-estadisticas-sobre-twitch-en-2021-que-tu-marca-debe-conocer/>
- [2] <https://www.arduino.cc/>
- [3] <https://innova3d.es/>
- [4] <https://app.vectary.com/p/>

## 12. CÓDIGO ARDUINO

```
// Librerías usadas
#include "Keyboard.h"
#include <SoftwareSerial.h>
#include <Nexion.h>

// Pines asignados a la pantalla
SoftwareSerial HMISerial(10, 11);

// Definimos las teclas F13-F20
#define BUTTON_KEY1 KEY_F13
#define BUTTON_KEY2 KEY_F14
#define BUTTON_KEY3 KEY_F15
#define BUTTON_KEY4 KEY_F16
#define BUTTON_KEY5 KEY_F17
#define BUTTON_KEY6 KEY_F18
#define BUTTON_KEY7 KEY_F19
#define BUTTON_KEY8 KEY_F20
#define BUTTON_KEY9 KEY_F21
#define BUTTON_KEY10 KEY_F22
#define BUTTON_KEY11 KEY_F23
#define BUTTON_KEY12 KEY_F24

// Declaración Botones táctiles
NexDSButton bt0 = NexDSButton(0, 13, "bt0");
NexDSButton bt1 = NexDSButton(0, 14, "bt1");
NexDSButton bt2 = NexDSButton(0, 15, "bt2");
NexDSButton bt3 = NexDSButton(0, 16, "bt3");
NexDSButton bt4 = NexDSButton(0, 17, "bt4");
NexDSButton bt5 = NexDSButton(0, 18, "bt5");
NexDSButton bt6 = NexDSButton(0, 19, "bt6");
NexButton b7 = NexButton(0, 20, "b7");
NexButton b8 = NexButton(0, 21, "b8");
NexButton b9 = NexButton(0, 22, "b9");
NexButton b10 = NexButton(0, 23, "b10");
NexDSButton bt11 = NexDSButton(0, 24, "bt11");
```

```
// Eventos Touch
NexTouch *nex_listen_list[] =
{
    &bt0, &bt1, &bt2, &bt3, &bt4, &bt5, &bt6, &bt11,
    &b7, &b8, &b9, &b10,
    NULL
};

void bt0_pulsar(void *ptr);
void bt1_pulsar(void *ptr);
void bt2_pulsar(void *ptr);
void bt3_pulsar(void *ptr);
void bt4_pulsar(void *ptr);
void bt5_pulsar(void *ptr);
void bt6_pulsar(void *ptr);
void b7_pulsar(void *ptr);
void b8_pulsar(void *ptr);
void b9_pulsar(void *ptr);
void b10_pulsar(void *ptr);
void bt11_pulsar(void *ptr);

// Se ejecutará siempre que se conecte el Arduino
void setup() {

    Keyboard.begin();
    Serial.begin(9600);
    bt0.attachPop(bt0_pulsar, &bt0);
    bt1.attachPop(bt1_pulsar, &bt1);
    bt2.attachPop(bt2_pulsar, &bt2);
    bt3.attachPop(bt3_pulsar, &bt3);
    bt4.attachPop(bt4_pulsar, &bt4);
    bt5.attachPop(bt5_pulsar, &bt5);
    bt6.attachPop(bt6_pulsar, &bt6);
    b7.attachPop(b7_pulsar);
    b8.attachPop(b8_pulsar);
    b9.attachPop(b9_pulsar);
    b10.attachPop(b10_pulsar);
    bt11.attachPop(bt11_pulsar, &bt11);
    HMISerial.begin(9600);

}
```

```
// Estará constantemente ejecutándose y escuchando posibles interacciones
void loop() {
    nexLoop(nex_listen_list);
    delay(5);
}

// Código de botón con doble estado
void bt0_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt0.getValue(&EstadoBoton);

    if(EstadoBoton) {
        //Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY1);
        delay(500);
        //Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY1);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY1);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY1);
    }
}
```

```
// Código de botón con doble estado
void bt1_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt1.getValue(&EstadoBoton);

    if(EstadoBoton) {
        //Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY2);
        delay(500);
        //Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY2);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY2);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY2);
    }
}

// Código de botón con doble estado
void bt2_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt2.getValue(&EstadoBoton);

    if(EstadoBoton) {
        //Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY3);
        delay(500);
        //Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY3);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY3);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY3);
    }
}
```

```
// Código de botón con doble estado
void bt3_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt3.getValue(&EstadoBoton);

    if(EstadoBoton) {
        //Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY4);
        delay(500);
        //Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY4);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY4);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY4);
    }
}

// Código de botón con estado único
void bt4_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY5);
    delay(500);
    Keyboard.release(BUTTON_KEY5);
}

// Código de botón con estado único
void bt5_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY6);
    delay(500);
    Keyboard.release(BUTTON_KEY6);
}
```

```
// Código de botón con doble estado
void bt6_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt6.getValue(&EstadoBoton);

    if(EstadoBoton) {
        //Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY7);
        delay(500);
        //Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY7);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY7);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY7);
    }
}

// Código de botón con estado único
void b7_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY8);
    delay(500);
    Keyboard.release(BUTTON_KEY8);
}

// Código de botón con estado único
void b8_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY9);
    delay(500);
    Keyboard.release(BUTTON_KEY9);
}

// Código de botón con estado único
void b9_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY10);
    delay(500);
    Keyboard.release(BUTTON_KEY10);
}
```

```
// Código de botón con estado único
void b10_pulsar(void *ptr) {
    Keyboard.press(BUTTON_KEY11);
    delay(500);
    Keyboard.release(BUTTON_KEY11);
}

// Código de botón con doble estado
void bt11_pulsar(void *ptr) {
    uint32_t EstadoBoton;
    bt11.getValue(&EstadoBoton);

    if(EstadoBoton) {
        //Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY12);
        delay(500);
        //Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY12);
    } else {
        Keyboard.press(KEY_LEFT_SHIFT);
        Keyboard.press(BUTTON_KEY12);
        delay(500);
        Keyboard.release(KEY_LEFT_SHIFT);
        Keyboard.release(BUTTON_KEY12);
    }
}
```

FIN

hecho por  
ALEJANDRO GOVANTES POLA