UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

Dispositivo de realización de "Backup on-the-go" basado en Raspberry Pi

Trabajo Fin de Grado

Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen

AUTOR/A: Popov , Denislav Miroslavov

Tutor/a: Canet Subiela, María José

CURSO ACADÉMICO: 2021/2022

Resumen

Este trabajo presenta un dispositivo basado en Raspberry Pi, alimentado a batería, que permite realizar copias de seguridad. Además, se ha diseñado un software propio, que permite realizar las copias, de tal manera que sea lo más minimalista posible. Este software dispone, principalmente, de un modo que permite seleccionar el dispositivo de almacenamiento de entrada (tarjeta SD) y el dispositivo de almacenamiento de salida (disco duro). De este modo, el proceso de realización del "backup" se resume en 3 pasos: selección de entrada, selección de salida, y realización de "backup". Además de la anteriormente nombrada característica principal, este dispositivo dispone de la posibilidad de ser usado en forma de NAS, para así poder acceder a los archivos mediante una red local.

Palabras clave

Raspberry Pi, copia de seguridad, NAS, desarrollo de software, fotografía.

Abstract

This project presents a Raspberry Pi based device, powered by battery, which allows to perform backups. In addition, a proprietary software has been designed, which allows to perform the backups, in such a way that it is as minimalist as possible. This software has, mainly, a mode that allows to select the input storage device (SD card) and the output storage device (hard disk). Thus, the backup process can be summarized in 3 steps: input selection, output selection and backup. In addition to the above mentioned main feature, this device has the possibility to be used as a NAS, so that the files can be accessed via a local network.

Key words

Raspberry Pi, backup, NAS, software development, photography.

Índice

Índice de figuras

1. Introducción

Desde que el mundo de la fotografía pasó del formato analógico al digital ha habido una evolución en los dispositivos de almacenamiento en los que las cámaras graban sus imágenes. En las videocámaras ha habido múltiples formatos, como pueden ser: cintas magnéticas, discos ópticos, discos duros o tarjetas de memoria, pero en las cámaras fotográficas siempre ha predominado el uso de tarjetas de memoria, siendo la tarjeta SD la más utilizada hasta el día de hoy.
Este proyecto surge de una necesidad. Una buena práctica en el mundo de la fotografía (así y como en cualquier entorno de creación de obras digitales) es la creación de copias de seguridad del proyecto cuanto antes. Hoy en día la forma más común de hacer dichas copias durante una jornada de trabajo es la utilización de un ordenador portátil, junto a un lector de tarjetas y discos duros externos. Este método es totalmente válido, pero hay ciertas ocasiones en las que la utilización de un ordenador portátil puede ser incómoda a la par que peligrosa para el equipo, esto se puede ver en eventos deportivos de bicicleta de montaña (Figura 1) o competiciones de enduro o motocross (Figura 2), donde el fotógrafo se suele ubicar en sitios en los cuales no hay espacio para utilizar cómodamente un ordenador portátil o es un ambiente lleno de polvo y suciedad.

Figura 1- Enduro en bicicleta @EnduroWorldSeries

Figura 2- Enduro en moto @EnduroMagazine

Como método alternativo se propone crear un dispositivo autónomo capaz de realizar las copias de los archivos sin necesitad de un ordenador portátil.

1.1 Objetivos

Como se ha comentado antes, se propone crear un dispositivo que sea capaz de realizar las copias de los archivos. Este dispositivo debe de tener unas características principales de diseño, que son:

- Portabilidad, uso de batería.
- Tamaño reducido.
- Facilidad de uso.
- Rapidez de puesta en marcha y uso.

1.2 Metodología

Para elaborar el prototipo funcional se ha seguido una metodología constituida por varias fases:

- Investigación: Se hace una búsqueda de información relativa a las necesidades y opciones de desarrollo del dispositivo.
- Desarrollo: Se empieza a desarrollar todo el material necesario para el funcionamiento del dispositivo.
- Testeo: Se realizan pruebas de funcionamiento. En caso de no pasar las pruebas se vuelve a pasar por las fases de investigación y desarrollo hasta llegar a pasar un test que muestre el correcto funcionamiento del dispositivo, alcanzando todos los requisitos de diseño.

1.3 Etapas

Este TFG pasa por varias etapas desde su inicio hasta su final, estas son:

- Búsqueda del proyecto.
- Primera investigación para validar la posibilidad de su realización.
- Investigación sobre las tecnologías disponibles para su desarrollo.
- Adquisición de los materiales necesarios.
- Montaje del dispositivo.
- Desarrollo del software.
- Testeo del dispositivo.
- Realización de la memoria.

1.4 Problemas

Durante la realización de este proyecto se ha tenido que lidiar con diferentes problemas, los más relevantes han sido; la falta de stock de dispositivos Raspberry Pi 4 debido a la crisis de chips y la falta de experiencia en Raspberry Pi Os, Python y Bash.

2. Desarrollo del proyecto

2.1 Análisis de requisitos y elección de componentes

Se requiere que el dispositivo tenga las principales características de portabilidad, facilidad y rapidez de uso. Para alcanzar estos requisitos hay que tener en cuenta características del hardware y el software que se va a utilizar.

2.1.1 Hardware

Se elige Raspberry Pi 4 para la creación del dispositivo, un Raspberry Pi se define como un *mini-pc* basado en ARM. Esta elección es debida al pequeño tamaño del dispositivo, los puertos de conexión disponibles, su potencia y su sistema operativo Raspberry Pi OS, basado en Linux.
En concreto se elige el modelo *Raspberry Pi 4 B* (Figura 3) con las siguientes características (Tabla 1):

Procesador	ARM Cortex-A72
Frecuencia de reloj	1,5 GHz
GPU	VideoCore VI (con soporte para OpenGL ES 3.x)
Memoria	2 GB LPDDR4 SDRAM
Conectividad	Bluetooth 5.0
	Wi-Fi 802.11ac
	Gigabit Ethernet
Puertos	GPIO 40 pines
	2 x micro HDMI(v2.0)
	2 x USB 2.0
	2 x USB 3.0
	CSI (cámara Raspberry Pi)
	DSI (pantalla tácil)
	Micro SD Conector
	Conector de audio Jack 3.5
	USB-C (alimentación)

Tabla 1- Especificaciones Raspberry Pi 4 B

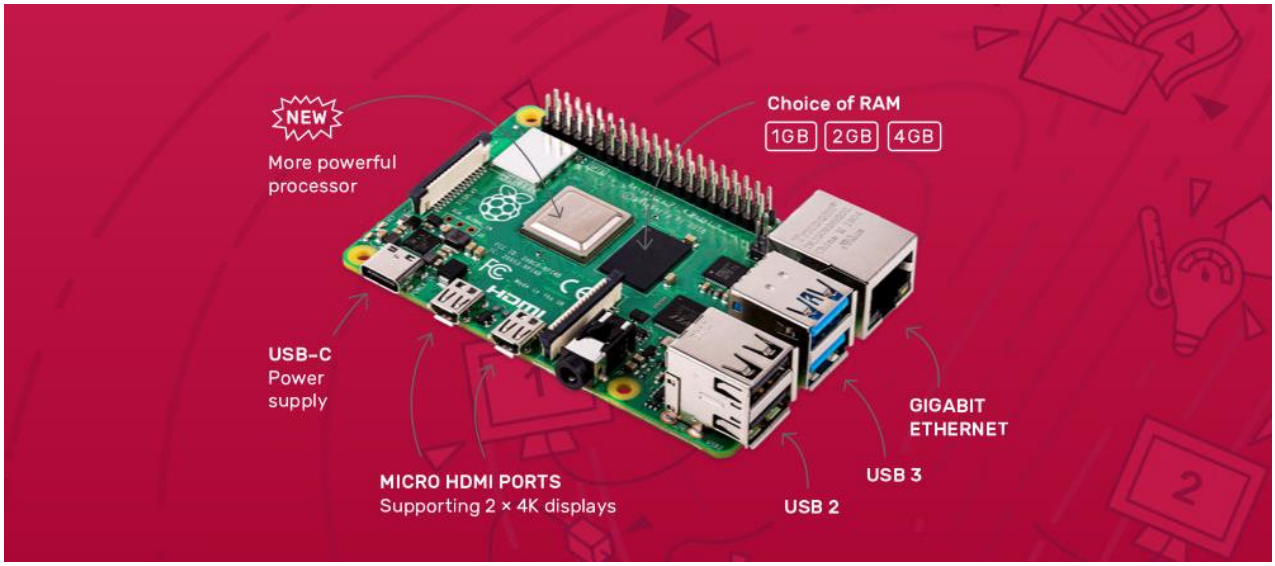

Figura 3- Raspberry Pi 4 B

Para que el usuario interactúe con el dispositivo se decide utilizar una pantalla táctil, ya que esto permite tener una interfaz de usuario flexible y facilitar el uso. En concreto se utiliza una pantalla de 3.5 pulgadas y resolución de 480x320 píxeles con tecnología resistiva. Esta pantalla se conecta mediante los pines de conexión GPIO de la Raspberry Pi. Además, se le instalan cuatro

disipadores de aluminio para una mejor refrigeración (Figura 4). Todo esto va encapsulado en una caja fabricada en ABS (Figura 6)

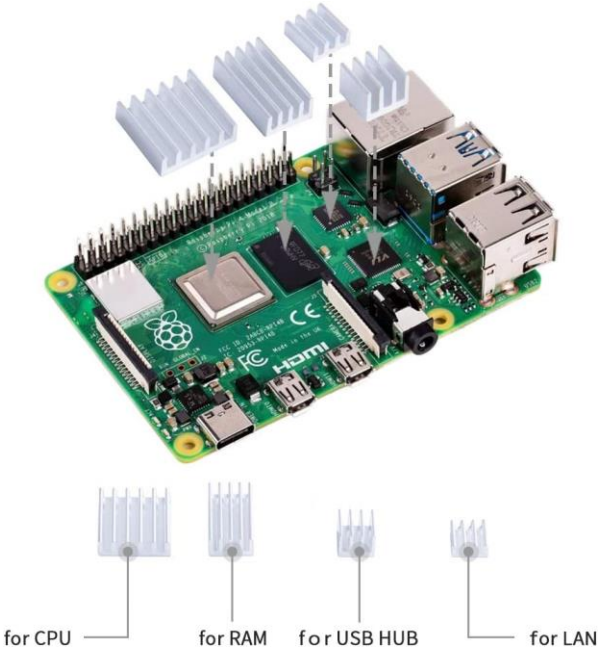

Figura 4- Disipadores de aluminio

Figura 5- Elementos de montaje

Figura 6- Dispositivo montado

La Raspberry Pi 4 cuenta con una entrada de alimentación USB de tipo C, gracias a esto podemos alimentar el dispositivo con cualquier *powerbank* o adaptador de corriente con una potencia mínima de 10W.

2.1.2 Software

Analizando los requisitos de funcionamiento se valora que:

- Para facilitar el uso del dispositivo, el software debe tener una apariencia minimalista, es decir, una interfaz sencilla de usar y entender.
- Al inicializar el dispositivo, todo debe cargar directamente para poder realizar la copia de las imágenes desde la tarjeta de memoria, con el mínimo trabajo de configuración posible.

Se decide utilizar el lenguaje Python para el desarrollo del software. Aunque no sea el lenguaje más rápido, se considera válido debido a las bibliotecas con las que cuenta y la cantidad de

documentación existente relativa a otros proyectos realizados sobre la plataforma de Raspberry Pi.

Aunque la principal funcionalidad del dispositivo sea la copia de las imágenes de una tarjeta de memoria a otro dispositivo de almacenamiento, se considera añadir funcionalidades extra como:

- Copia íntegra del dispositivo de almacenamiento (en lugar de la carpeta DCIM seleccionada por defecto).
- Cambio de idioma de la interfaz entre inglés y español.
- Modo oscuro y modo claro.
- Punto de acceso WIFI para ver o descargar los archivos desde otro dispositivo.
- Ventana de instrucciones de uso.
- Muestra de la temperatura de la CPU.

2.2 Diseño de la interfaz de usuario

Partiendo de las características de minimalismo y facilidad de uso, se opta por la utilización de un único “layout” para realizar todas las funciones, haciendo uso de ventanas emergentes en ciertas ocasiones.
Se comienza la fase de diseño con un esbozo de la interfaz de usuario, teniendo en cuenta solo la función principal de copia de imágenes de la tarjeta de memoria hacia otro dispositivo de almacenamiento.

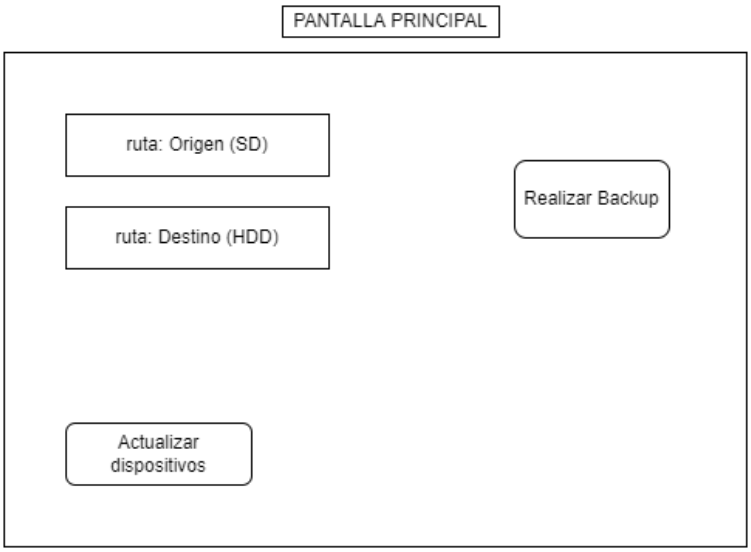

Figura 7- Primer diseño de interfaz

En el primer diseño (Figura 7), se pueden ver dos campos para introducir datos y dos botones. En el primer campo se introduce el origen de la copia, por ejemplo, una tarjeta SD (conectada por USB, mediante un adaptador). En el segundo campo se introduce, por ejemplo, un disco duro externo con interfaz USB. Uno de los botones iniciaría la copia y el otro actualizaría de forma manual los dispositivos de almacenamiento conectados en caso de que no se actualicen automáticamente al introducirlos o expulsarlos.
Este primer diseño nos sirve simplemente como base para empezar a pensar en la correcta organización del *layout*.
Los elementos de selección de dispositivo y el botón para realizar la copia se consideran en una posición correcta, pero se debe de crear algún método para albergar todas las funciones y no tener una gran cantidad de botones ni información innecesaria en la pantalla.
Se opta por crear una zona a modo de barra inferior, que albergue diferentes botones. Además, para poder expulsar rápidamente algún dispositivo de almacenamiento se colocan dos botones destinados a expulsar el dispositivo de origen o de destino (Figura 8).

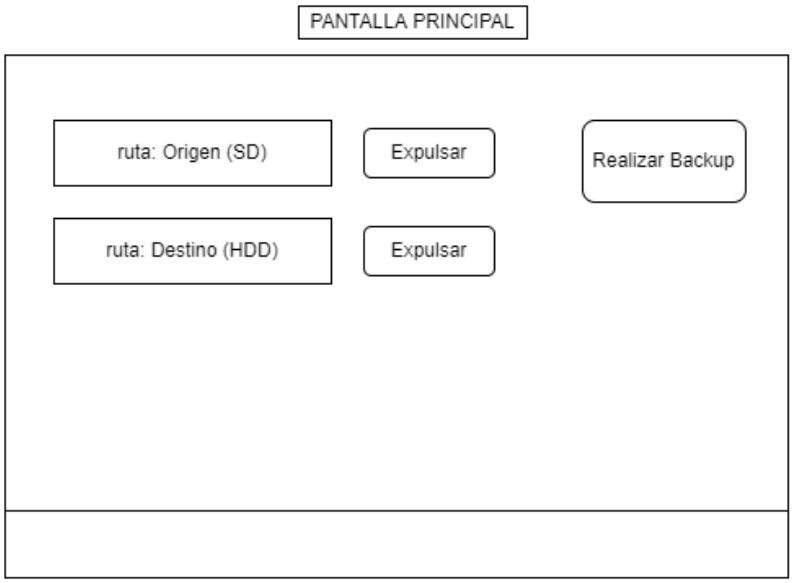

Figura 8- Barra inferior y botones de expulsión

La siguiente tarea que realizar es la colocación de elementos para funciones complementarias, como:

- Actualizar dispositivos
- Cambiar idioma
- Cambiar entre tema claro y oscuro
- Mostrar instrucciones
- Apagar el dispositivo
- Activar/desactivar punto de acceso wifi
- Etc…

Para poder albergar todas esas funciones se decide utilizar botones para mostrar las instrucciones y activar el wifi, además se añade un “combo-box” para elegir el idioma entre español e inglés y otro botón auxiliar que muestra un menú desplegable con todas las funciones restantes (Figura 9). Junto al combo-box para la selección de origen y destino hay un texto mostrando el estado del dispositivo seleccionado.

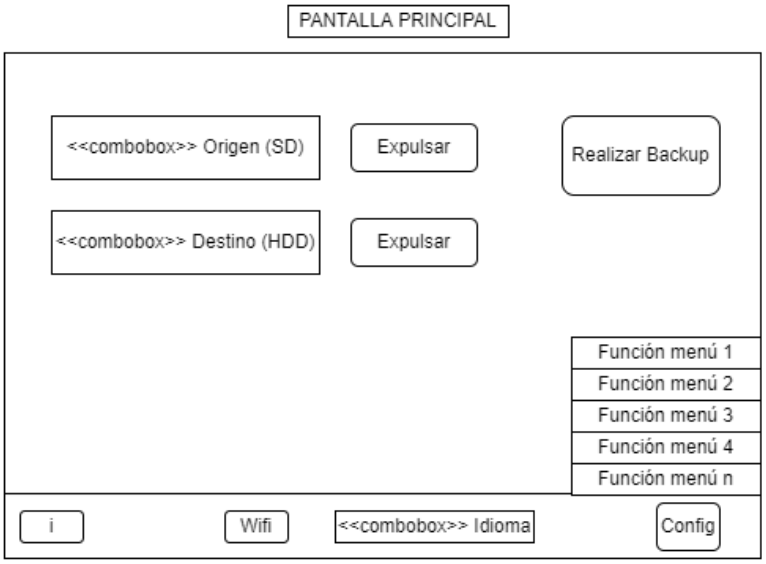

Figura 9- Botones inferiores y menú desplegable

Como una función secundaria del dispositivo es poder copiar todo lo que contiene el dispositivo de origen se decide añadir otro *combo-box* que permita seleccionar entre la carpeta DCIM que contiene las imágenes y la carpeta RAÍZ del dispositivo de almacenamiento elegido. Por defecto está siempre seleccionada la opción DCIM (Figura 10).

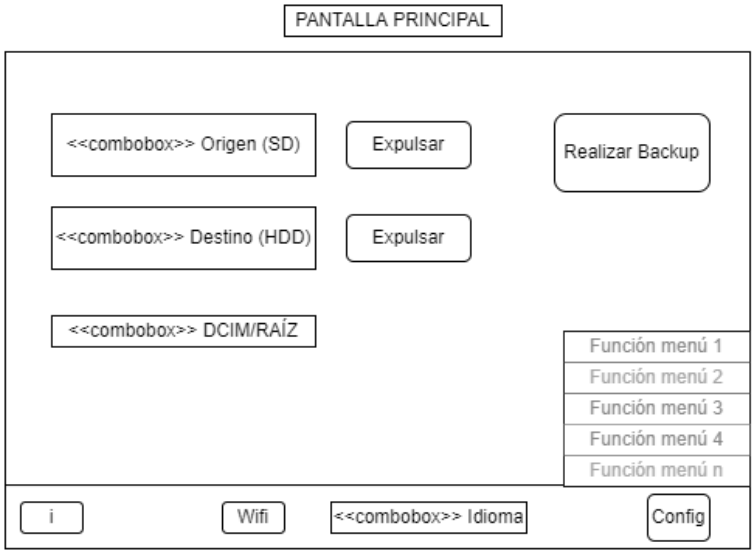

Figura 10- Combo-box seleccionar DCIM o RAÍZ

En el momento de pulsar el botón de mostrar las instrucciones se abre una ventana emergente que contiene el texto de las instrucciones (Figura 11).

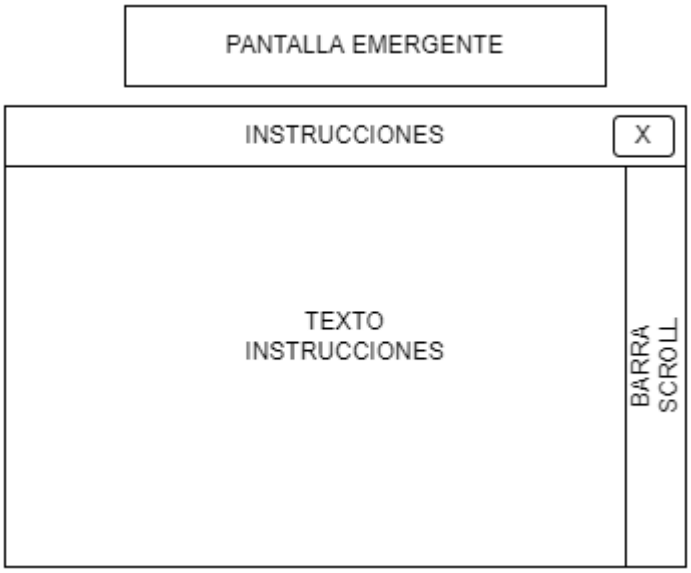

Figura 11- Ventana de instrucciones

Cuando se inicia una copia aparece una ventana emergente mostrando el progreso de la copia (Figura 12).

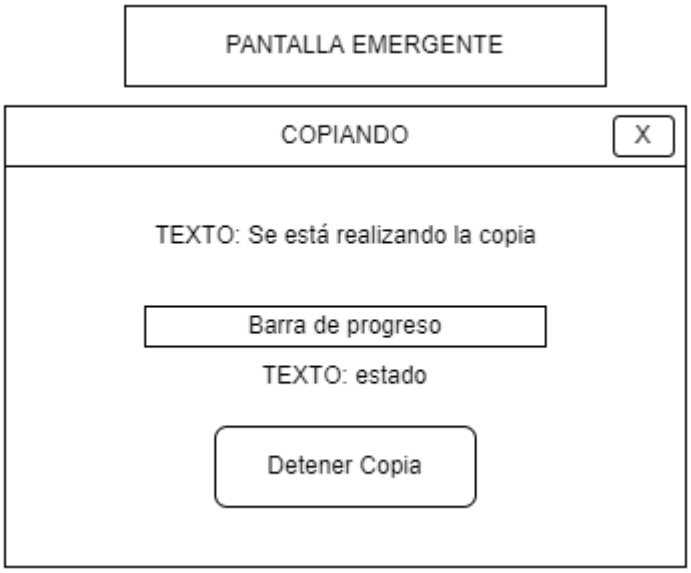

Figura 12- Ventana de progreso

En caso de que hubiese algún fallo o hubiera que notificar algo al usuario aparecería una ventana emergente específica para dicha función, como por ejemplo preguntar si realmente se desea apagar el dispositivo (Figura 13).

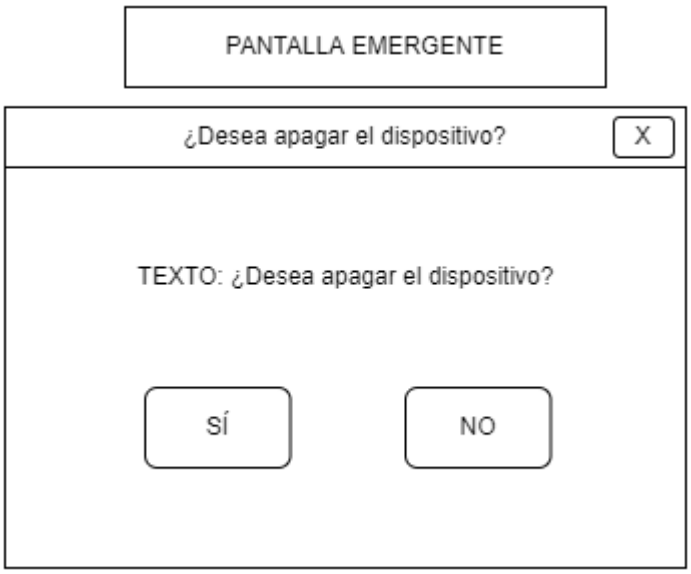

Figura 13- Ventana emergente para info. Usuario

2.3 Desarrollo del software

Para el desarrollo del software que utiliza el dispositivo se hace uso del lenguaje de programación Python y se utiliza Tkinter para la elaboración de la GUI (interfaz gráfica de usuario). Se ha elegido Python frente a otros lenguajes de programación debido a que es un lenguaje bastante usado actualmente y se ha considerado oportuno utilizarlo para tener más experiencia con él. Tkinter se ha elegido frente a otras bibliotecas gráficas porque se considera un estándar para la creación de GUI's en Python. Además, se modifican ficheros en Bash y se utiliza RaspAP, un software externo para la apertura del punto de acceso wifi.

Se desarrolla de tal forma que el usuario pueda interactuar con la interfaz y realizar diferentes acciones:

- **Seleccionar origen:** Elige desde qué dispositivo conectado se copiarán las imágenes o archivos.
- **Seleccionar destino:** Elige a qué dispositivo conectado se copiarán las imágenes o archivos.
- **Expulsar:** Expulsa el dispositivo de almacenamiento conectado en el origen o destino. En caso de pulsar el botón durante la realización de una copia salta un aviso mostrando que hay que esperar a que se realice la copia.
- **Copiar:** Inicia la copia de las imágenes o archivos.
- **Copiar desde:** Permite seleccionar si la copia desde el dispositivo de origen se realiza desde la carpeta DCIM o desde la carpeta raíz del dispositivo de origen.
- **Instrucciones:** Abre una ventana emergente con las instrucciones de uso del dispositivo.
- **AP Wifi:** Establece un punto de acceso wifi para poder visualizar o descargar los archivos desde un dispositivo móvil.
- **Cambiar de idioma:** Permite seleccionar entre los idiomas inglés y español.
- **Cambiar tema:** Cambia entre el modo oscuro y el modo claro de la interfaz.
- **Mostrar temperatura de la CPU:** Muestra en pantalla la temperatura actual de la CPU.
- **Refrescar dispositivos:** En caso de que no se actualice automáticamente la lista de dispositivos conectados permite realizar una actualización manual de los mismos.
- **Reiniciar App:** Permite reiniciar la aplicación sin reiniciar todo el dispositivo.
- **Minimizar:** Quita el modo de pantalla completa para poder interactuar con la interfaz de Raspberry Pi OS en caso de que fuese necesario.
- **Reiniciar dispositivo:** Reinicia todo el dispositivo.
- **Apagar dispositivo:** Apaga el dispositivo.

La aplicación se aloja en una carpeta del escritorio de la Raspberry Pi. Esta carpeta está comprendida por:

- **Res:** Carpeta que contiene *resources*, en este caso los iconos utilizados en formato .png.
- **Theme:** Carpeta que contiene la configuración del tema visual utilizado.
- **Azure.tcl:** Archivo del tema visual utilizado.
- **Backup.py:** Archivo que contiene el código de la App desarrollada.
- **Instrucciones.txt:** Archivo de texto que contiene las instrucciones de uso en español.
- **Instructions.txt:** Archivo que contiene las instrucciones de uso en inglés.

2.3.1 Funcionamiento principal

El flujo de la función principal de copia de imágenes o archivos se muestra en la Figura 14:

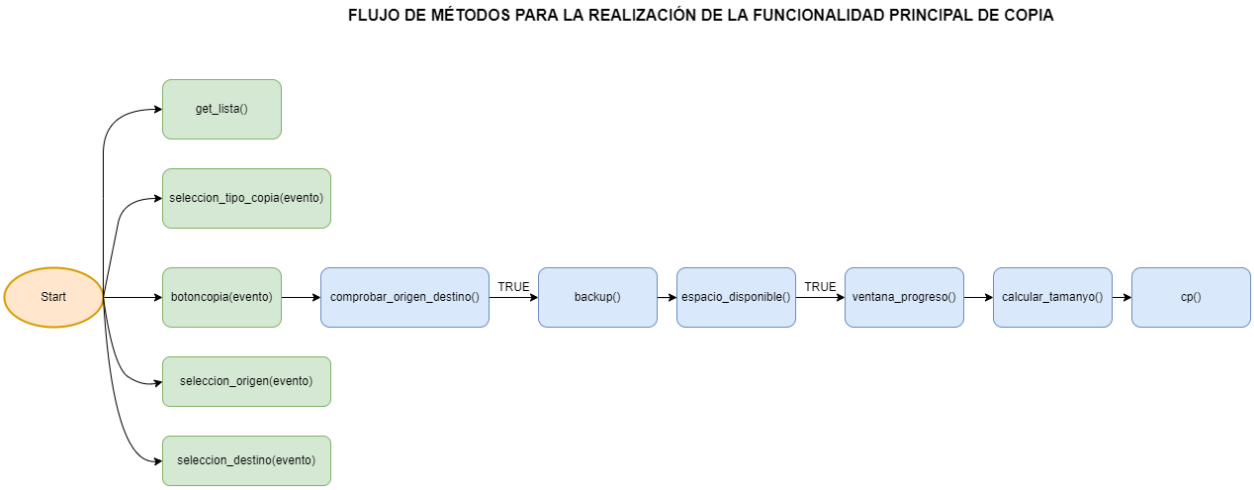

Figura 14- Flujo de métodos para la realización de la funcionalidad principal de copia

En esta figura se aprecian los métodos utilizados para la realización de la copia de las imágenes o archivos. Se excluyen de este diagrama los métodos encargados de la inicialización de la interfaz gráfica y sus diferentes elementos, salvo el método ventana_progreso(), encargado de crear una nueva ventana emergente.

2.3.2 Detalles de los métodos utilizados para la funcionalidad principal

Se procede a analizar los diferentes métodos (Figura 14) encargados del funcionamiento de la función de copia de imágenes o archivos. Se matiza que solo se va a analizar los métodos que son llamados a la hora de realizar una copia. Los métodos adicionales, encargados de otras funciones, quedan excluidos de este apartado.

- **get_lista():** Este método se encarga de crear una variable “lista” que contiene los diferentes dispositivos de almacenamiento conectados al dispositivo (Figura 15).

```
#-------------------------------------------------------------------------------
# Método para obtener una lista de los dispositivos de almacenamiento usb montados
#-------------------------------------------------------------------------------

def get_lista():
    global lista
    peticion="/media/"+str(user).rstrip("\n")+"/"
    lista = os.listdir(peticion)
    return lista
```

Figura 15- get_lista()

- **selección_tipo_copia():** Este método ante el evento de selección desde el ComboBox cambia la ruta de copia del dispositivo de origen, seleccionando la carpeta DCIM o la carpeta raíz del dispositivo (Figura 16).

```
#-------------------------------------------------------------------------------------------
# Método para seleccionar una copia desde la carpeta DCIM o desde la raíz del
# dispositivo
#-------------------------------------------------------------------------------------------
def seleccion_tipo_copia(event):

    try:
        global src
        if combo_copia.get()=="DCIM":

            #print(str(user))
            src = "/media/"+str(user).rstrip("\n")+"/"+origen+"/DCIM"
            return src

        else:

            src="/media/"+str(user).rstrip("\n")+"/"+origen
            return src
        #print(src)
    except:

        messagebox.showinfo(title=Error, message=seleccionarPrimeroOrigen)
```

Figura 16- seleccion_tipo_copia()

- **selección_origen():** Selecciona mediante un ComboBox, entre los dispositivos conectados, cuál será el de origen de los archivos a copiar (Figura 17).

```
#-------------------------------------------------------------------------------------------
# Método de seleccion de origen (p.e. tarjeta SD)
#-------------------------------------------------------------------------------------------
def seleccion_origen(event):
    global origen
    origen = combo_origen.get()
    seleccion_tipo_copia(event)
    #print(src)
    try:
        totalOrigen, usadoOrigen, disponibleOrigen = shutil.disk_usage(src)
        if (usadoOrigen // (10**9))<=1:
            usoOrigen= str(usadoOrigen // (10**6)) + "MB"
        else:
            usoOrigen=str(usadoOrigen // (10**9))+"GB"

        if (disponibleOrigen // (10**9))<=1:
            dispOrigen= str(disponibleOrigen // (10**6)) + "MB"
        else:
            dispOrigen=str(disponibleOrigen // (10**9)) + "GB"


        #estado = usadoEn + combo_copia.get()+" : "+(usoOrigen) + libre + (dispOrigen)
        #estadoOrigen.config(text=estado)
        cambiar_estado_origen(usoOrigen, dispOrigen)

    except:
        estadoOrigen.config(text=noHayDCIM)
```

Figura 17- seleccion_origen()

- **selección_destino():** Selecciona mediante un ComboBox, entre los dispositivos conectados, cuál será el dispositivo de destino de los archivos copiados (Figura 18).

```
#-------------------------------------------------------------------------------------------
# Método de seleccion de destino (p.e. HDD)
#-------------------------------------------------------------------------------------------
def seleccion_destino(event):
    global destino
    destino = combo_destino.get()
    user=os.popen("whoami").read()
    #print(user)
    dest = "/media/"+str(user).rstrip("\n")+"/"+destino
    totalDestino, usadoDestino, disponibleDestino = shutil.disk_usage(dest)

    if (totalDestino // (10**9))<=1:
        tDestino= str(totalDestino // (10**6)) + "MB"
    else:
        tDestino= str(totalDestino // (10**9))+"GB"

    if (disponibleDestino // (10**9))<=1:
        dispDestino= str(disponibleDestino // (10**6)) + "MB"
    else:
        dispDestino= str(disponibleDestino // (10**9)) + "GB"


    #estado = "Total: " + (tDestino) + libre + (dispDestino)
    #estadoDestino.config(text=estado)
    cambiar_estado_destino(tDestino, dispDestino)
```

Figura 18- seleccion_destino()

- **comprobar_origen_destino():** Método utilizado para comprobar si están seleccionados el origen y el destino de la copia (Figura 19).

```
#-------------------------------------------------------------------------------------------
# Método para comprobar si hay origen y destino
#-------------------------------------------------------------------------------------------
def comprobar_origen_destino():

    try:
        print("El origen es "+origen)
        origen


    except:
        print("No hay ningun origen seleccionado")
        permitir_copia=True
        messagebox.showinfo( title=noHayOrigen, message=noHayOrigen)
        return False



    try:
        print("El destino es "+destino)
        destino


    except:
        print("No hay ningun destino seleccionado")
        messagebox.showinfo( title=noHayDestino, message=noHayDestino)
        return False
```

Figura 19- comprobar_origen_destino()

- **espacio_disponible():** Método que comprueba si en el destino hay espacio suficiente para guardar los archivos a copiar (Figura 20).

```
#-------------------------------------------------------------------------------
# Método que comprueba si hay espacio disponible en el destino para copiar
# los archivos del origen en él
#-------------------------------------------------------------------------------
def espacio_disponible():
    totalOrigen, usadoOrigen, disponibleOrigen = shutil.disk_usage(src)
    dest = "/media/"+str(user).rstrip("\n")+"/"+destino
    totalDestino, usadoDestino, disponibleDestino = shutil.disk_usage(dest)

    if usadoOrigen>disponibleDestino:
        return False
    else:
        return True
```

Figura 20- espacio_disponible()

- **ventana_progreso():** Crea una ventana emergente en la cual se muestra el progreso de la copia. Dependiendo del idioma seleccionado crea la ventana con los textos en inglés o en español (Figura 21).

```
#-------------------------------------------------------------------------------------------
# Método que crea la ventana que muestra el progreso de la copia, en inglés o español
#-------------------------------------------------------------------------------------------
def ventana_progreso():
    if idioma=="Español":
        global ventanaProgreso
        ventanaProgreso = Toplevel(main_window)
        ventanaProgreso.title("Copiando")
        ventanaProgreso.geometry(f'{"300"}x{"160"}+{int(90)}+{int(90)}')
        #ventanaProgreso.geometry("300x160")
        ventanaProgreso.minsize(300, 160)

        global progressbar2
        progressbar2 = ttk.Progressbar(ventanaProgreso, mode="determinate", length=280)
        progressbar2.grid(row=1, sticky=tk.EW, pady=5, padx=10)
        progressbar2.step(0)

        global texto
        texto = Label(ventanaProgreso, text="Se está realizando la copia")
        texto.grid(row=0, sticky=tk.EW, pady=10, padx=25)

        global textoporc
        textoporc = Label(ventanaProgreso, text="Copiado: 0%")
        textoporc.grid(row=2 , column=0)

        global boton_detener_backup
        boton_detener_backup = ttk.Button(ventanaProgreso, text='Detener Copia',
         command=detener_backup, width=12)
        boton_detener_backup.grid(row=4, column=0, padx=30, pady=8, sticky=tk.S)
    else:
        ventanaProgreso = Toplevel(main_window)
        ventanaProgreso.title("Backup")
        ventanaProgreso.geometry(f'{"300"}x{"160"}+{int(90)}+{int(90)}')
        #ventanaProgreso.geometry("300x160")
        ventanaProgreso.minsize(300, 160)

        #global progressbar2
        progressbar2 = ttk.Progressbar(ventanaProgreso, mode="determinate", length=280)
        progressbar2.grid(row=1, sticky=tk.EW, pady=5, padx=10)
        progressbar2.step(0)

        #global texto
        texto = Label(ventanaProgreso, text="Backup is in progress")
        texto.grid(row=0, sticky=tk.EW, pady=10, padx=25)

        #global textoporc
        textoporc = Label(ventanaProgreso, text="Copied: 0%")
        textoporc.grid(row=2 , column=0)

        #global boton_detener_backup
        boton_detener_backup = ttk.Button(ventanaProgreso, text='Cancel Backup',
         command=detener_backup, width=12)
        boton_detener_backup.grid(row=4, column=0, padx=30, pady=8, sticky=tk.S)
```

Figura 21- ventana_progreso()

- **calcular_tamanyo():** Método que calcula el tamaño de la carpeta de destino, actualizando la barra de tareas creada en la ventana emergente (Figura 22).

```
#-------------------------------------------------------------------------------------------
# Método que calcula el tamaño de la carpeta destino, actualizando la barra de progreso
#-------------------------------------------------------------------------------------------
def calcular_tamanyo():
    print("Iniciamos calcular_tamnyo()")
    global control
    control = False
    size=0
    size2=0
    prog=0

    while control == False:

        try:

            size=0
            for ele in os.scandir(src):
                if ele.is_dir():
                    for dire in os.scandir(ele.path):
                        size+=os.path.getsize(dire.path)
                else:
                    size+=os.path.getsize(ele)



            size2=0
            for ele in os.scandir(dst):
                if ele.is_dir():
                    for dire in os.scandir(ele.path):
                        size2+=os.path.getsize(dire.path)
                else:
                    size2+=os.path.getsize(ele)


            prog = size2*100/size
            size2=0
            size=0
            progressbar2['value'] = prog


            if idioma=="Español":
                estado = "Copiado: " + str(round(prog)) + "%"
            else:
                estado = "Copied: " + str(round(prog)) + "%"

            textoporc.config(text=estado)

            if prog<99.99:
                size2=0


            else:
                print("el progreso es : "+ str(prog))
                prog=0
                control=True


                texto.config(text=backupRealizado)

                progressbar2['value']=0

                print(dst)
                del size
                del size2

                print("control es: "+str(control))

                print("---------------------\n")
                messagebox.showinfo(title=hecho, message=backupRealizado)
                ventanaProgreso.destroy()


        except:
            #print("No existe la carpeta todavía")
            1==1
```

Figura 22- calcular_tamanyo()

- **cp():** Método que haciendo uso de Copytree() copia todo desde la ruta de origen *src* a la ruta de destino *dest.* Por defecto realiza la copia mediante el método Copy2(), el cual copia los metadatos de los archivos (Figura 23).

```
#-------------------------------------------------------------------------------------
# Método que utiliza Copytree() para realizar la copia de todos los archivos.
# Usa el método copy2(), con el cual se copian los metadatos
#-------------------------------------------------------------------------------------
def cp(src: str, dest: str):
    print("cp iniciado")
    #print("Despues de barra")

    copytree(src, dest)
    print("cp finalizado")
```

Figura 23- cp()

- **botoncopia():** Método callback del botón de copia. Cuando se activa llama a *comprobar_origen_destino()* y si la respuesta es verdadera llama al método *backup()* (Figura 24).

```
#-------------------------------------------------------------------------------------
# Método callback del botón de realizar la copia
#-------------------------------------------------------------------------------------
def botoncopia():

    if comprobar_origen_destino()!=False:
        background(backup)
```

Figura 24- botoncopia()

- **backup():** Método para realizar la copia. Hace varias comprobaciones para asegurar que se puede realizar la copia y crea la carpeta de destino con un nombre siguiendo el patrón de << “Copia” + fecha y hora de la copia >>. En este método se hace la llamada a *espacio_disponible(), ventana_progreso(), calcular_tamnyo() y cp()* (Figura 25).

```
#-------------------------------------------------------------------------------------------
# Método para realizar la copia
#-------------------------------------------------------------------------------------------
def backup():
    try:
        global permitir_copia
        if permitir_copia==True:
            permitir_copia=False
            if origen!=destino:
                if os.path.exists(src):

                    if espacio_disponible():
                        print("\n#########################")
                        ventana_progreso()
                        print("abrimos ventana progreso en Backup()")
                        fecha_hora = strftime("%d_%m_%Y %H_%M_%S")
                        carpetanueva= "/Copia "+ fecha_hora
                        global dst
                        dst = "/media/"+str(user).rstrip("\n")+"/"+ destino + carpetanueva
                        background(calcular_tamanyo)
                        print("background calcular tamaño()")
                        global proc
                        proc = Process(target=cp, args=(src, dst), daemon=True)
                        proc.start() #copytree(src,dst)
                        proc.join()
                        #print(str(src), str(dst))
                        print(proc)

                        if proc.exitcode==0:
                            #messagebox.showinfo(title="Hecho", message="Se ha realizado
                            #  el Backup")
                            permitir_copia=True
                            return

                        else:
                            print("El backup se ha cancelado")
                            permitir_copia=True
                            messagebox.showinfo(title=cancelado, message=seHaCancelado)


                    else:
                        permitir_copia=True

                        messagebox.showinfo(title=Error, message=noHayEspacio)


                else:
                    permitir_copia=True

                    messagebox.showinfo(title=Error, message=noHayDCIM)


            else:
                permitir_copia=True

                messagebox.showinfo(title=Error, message=mismoOrigenDestino)


        else:

            messagebox.showinfo(title=Error, message=esperaCopia)

    except:
        print("algo ha ido mal al copiar")
        permitir_copia=True
        messagebox.showinfo(title=Error, message=haIdoMal)
```

Figura 25- backup()

2.4 Configuración de la Raspberry Pi

2.4.1 Inicio automático de la App

La App desarrollada se encuentra alojada en una carpeta llamada “Backup_TFG” en el escritorio de la Raspberry Pi. Para hacer que la app se ejecute al iniciar el sistema operativo se modifica el archivo ***/home/pi/.bashrc*** (Figura 26) y se añade al final del mismo las siguientes líneas de código :

cd /home/pi/Desktop/Backup_TFG/
python backup.py

Estas líneas se encargan de entrar a la carpeta *Backup_TFG* y ejecutar el archivo *backup.py*.

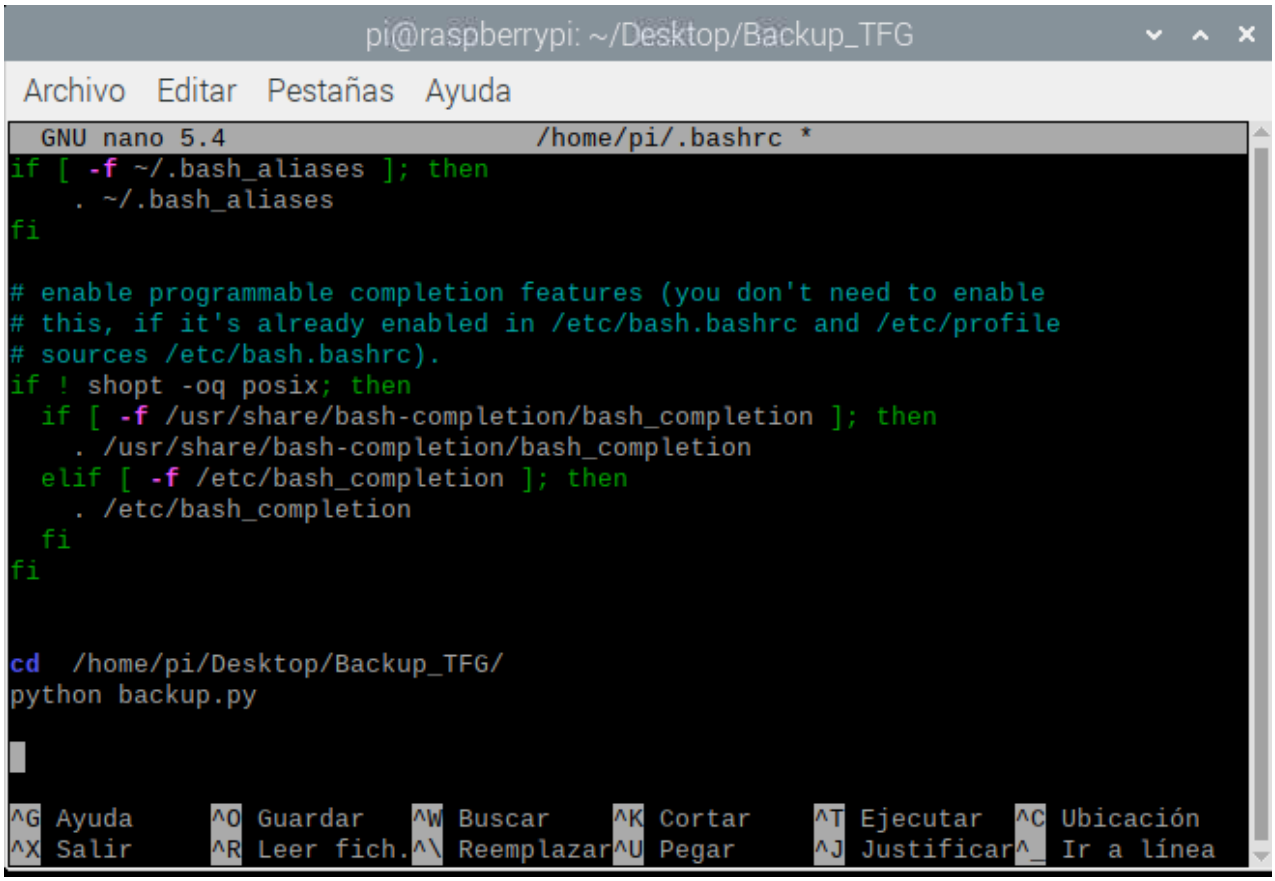

Figura 26- archivo de inicio

2.4.2 Configuración RaspAP

Para la creación de un punto de acceso wifi se utiliza el software RaspAP.
Se instala el software mediante la siguiente serie de comandos:

(Actualiza y reinicia el dispositivo)

sudo apt-get update
sudo apt-get full-upgrade
sudo reboot

(Abre configuración de Raspberry Pi para cambiar el país de utilización del wifi)

sudo raspi-config

(Llama al instalador rápido de RaspAP)

Curl -sL https://install.raspap.com | bash

Una vez instalado el software, el punto de acceso wifi queda configurado con los siguientes parámetros:

- **Dirección IP:** 10.3.141.1
- **Usuario:** admin
- **Contraseña:** secret
- **Rango DHCP:** 10.3.141.50 – 10.3.141.255
- **SSID:** raspi_webgui
- **Password:** ChangeMe

El siguiente paso es cambiar el nombre de la red y la contraseña. Para esta tarea se debe acceder a la webgui de RaspAP, mediante el navegador web a la dirección de localhost. Una vez en la configuración se accede al apartado de *Hotspot* y se cambia el nombre de la red a ***Backup DMP*** (Figura 27) y la contraseña se cambia a ***1234567#*** (Figura 28).

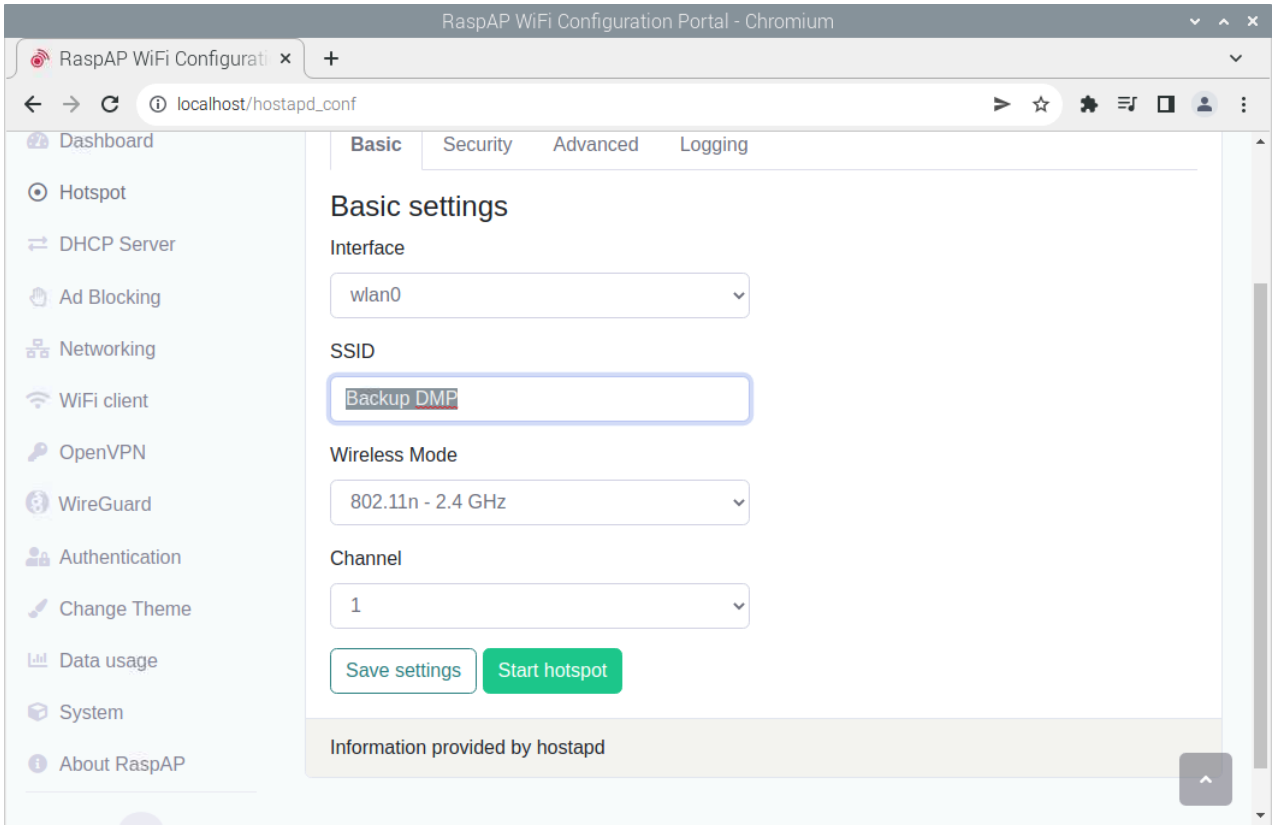

Figura 27- Configuración SSID

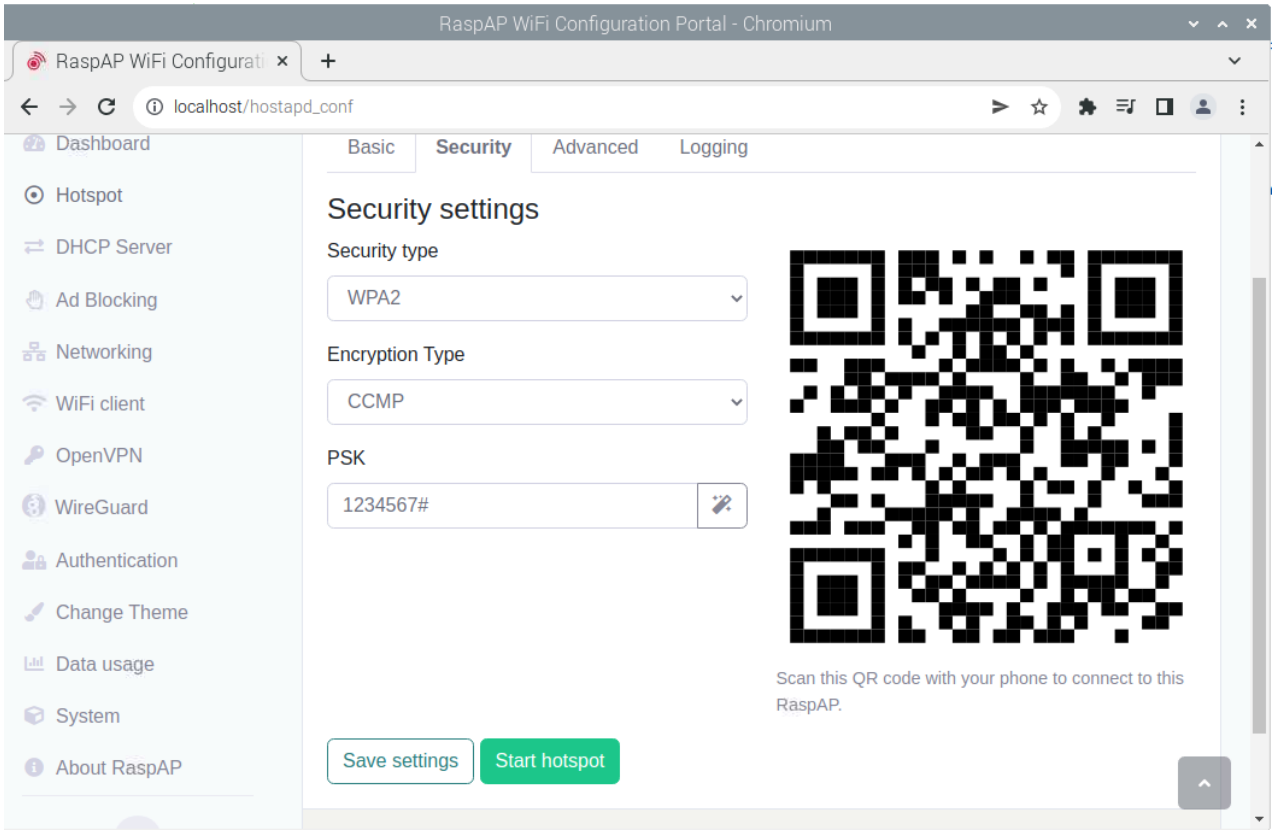

Figura 28- Configuración contraseña Wifi

2.4.3 Configuración de Samba

Para poder visualizar y descargar los archivos de los dispositivos de almacenamiento desde otro dispositivo mediante la red wifi creada se utiliza el protocolo Samba. Dicho protocolo es una implementación del protocolo de archivos compartidos de Microsoft Windows CIFS (anteriormente conocido como SMB) para sistemas tipo UNIX.
Para configurar Samba se modifica el archivo de configuración smb.conf. Para realizar esta tarea se utiliza el comando:

sudo nano /etc/samba/smb.conf

Una vez abierto el archivo con el comando anterior se procede a comentar la configuración predeterminada de los dispositivos y se añade la siguiente:

[Dispositivos conectados]
Comment = Dispositivos conectados
Path = media/pi/
Browsable = yes
Writeable = yes
Only guest = no
Create mask = 0777
Directory mask = 0777
Public = no

Con el comando ***smbpasswd*** cambiamos la contraseña del usuario ***pi*** a ***1234***.

Con todo esto Samba queda configurado como:

- **IP:** 10.3.141.1
- **Usuario:** Pi
- **Contraseña:** 1234

2.4.4 Configuración notificaciones y barra de tareas

Se desactivan las notificaciones del sistema operativo, innecesarias para este uso (Figura 29):

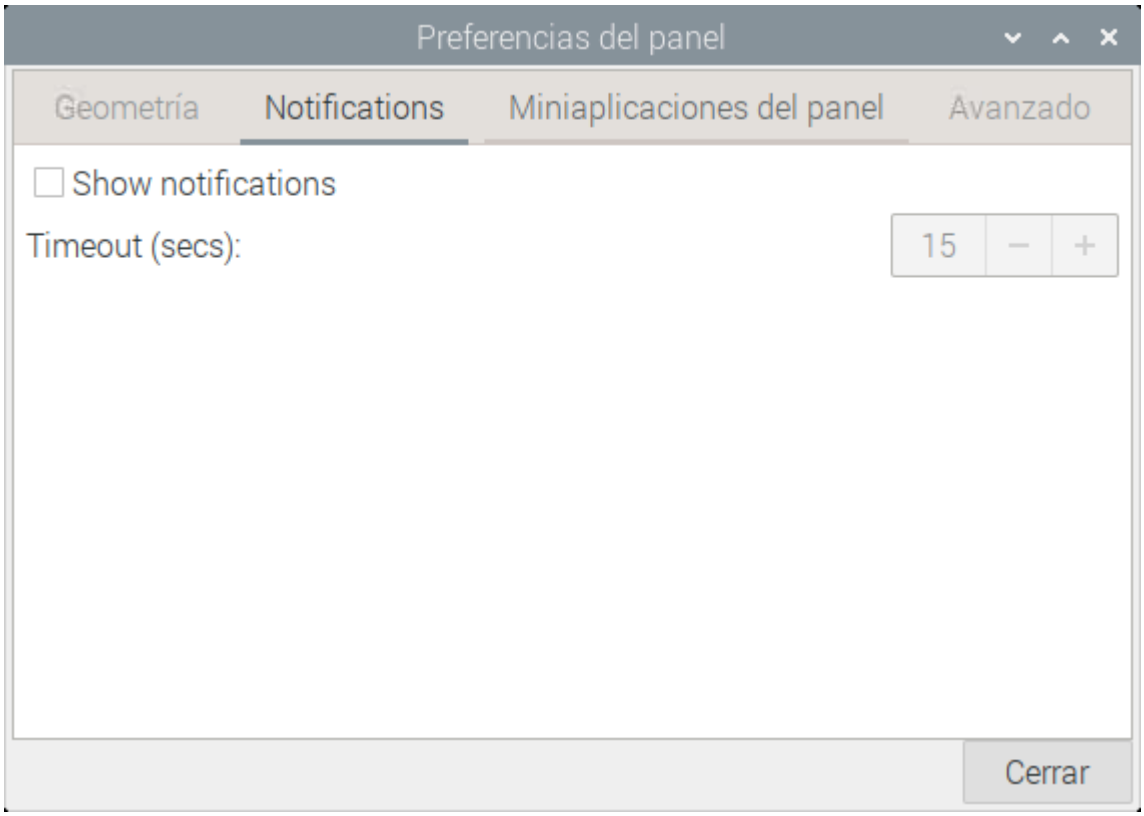

Figura 29- Desactivar notificaciones

Se oculta la barra de tareas automáticamente cuando no está en uso (Figura 30):

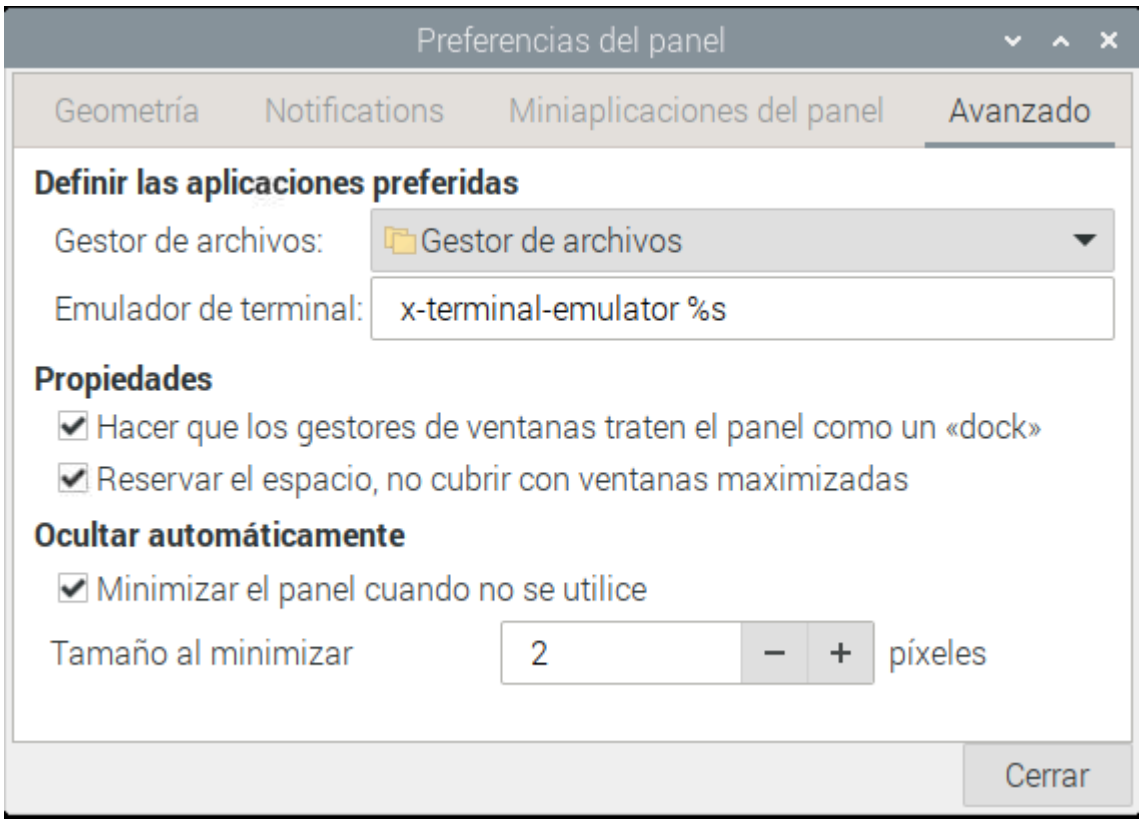

Figura 30- Ocultar barra de tareas

Por último, se desactivan las ventanas emergentes al insertar dispositivos de almacenamiento (Figura 31):

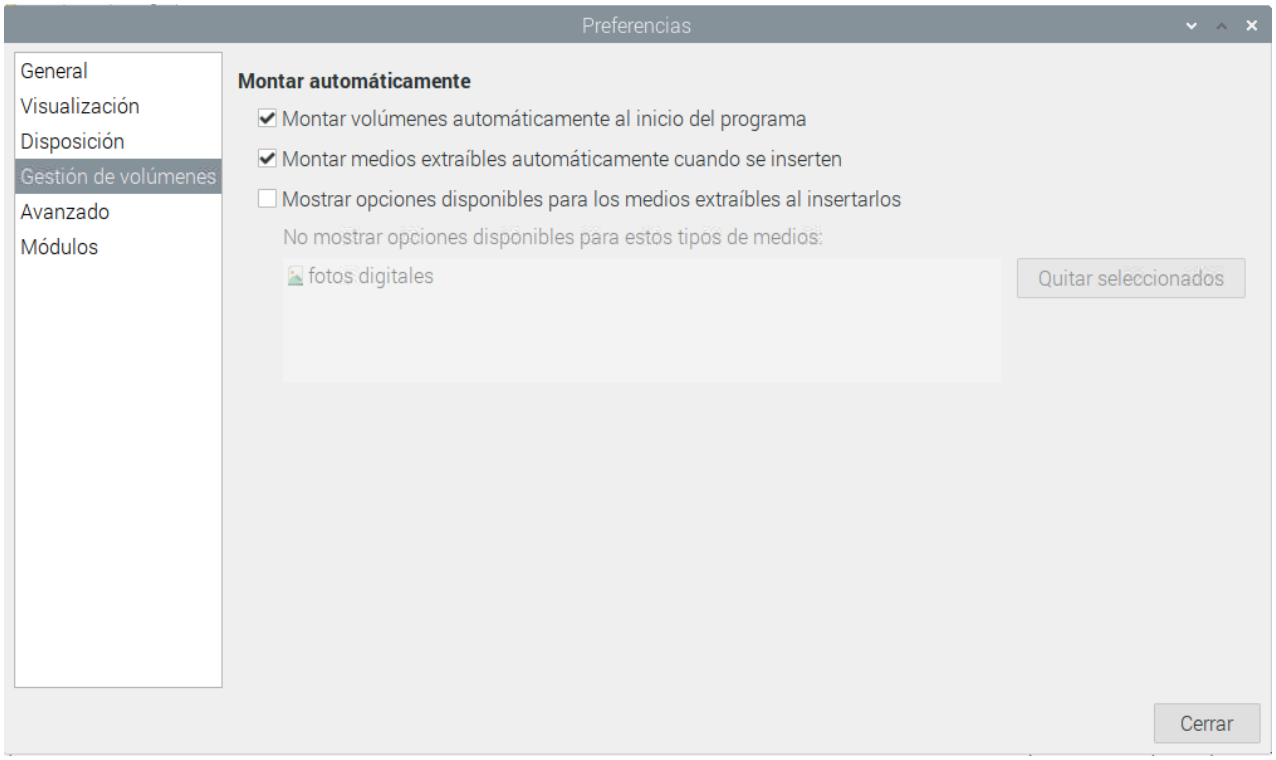

Figura 31- Ocultar opciones al insertar dispositivo

3. Guía de uso

Una vez encendido el dispositivo se presenta la ventana principal de la App (Figura 32):

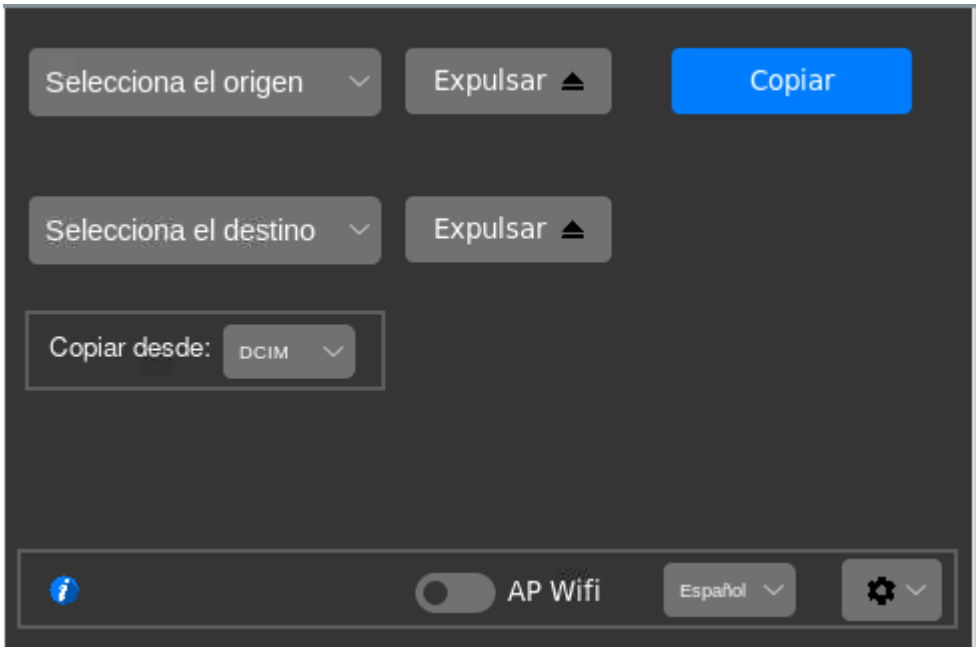

Figura 32- Ventana principal App

3.1 Realizar copia

Para realizar una copia desde la carpeta de fotos DCIM del dispositivo de origen únicamente debemos seleccionar el origen, el destino y pulsar sobre el botón "**Copiar**" (Figura 33, Figura 34). Una vez iniciada la copia se visualiza una ventana con una barra de progreso Figura 35) y al terminar el proceso de copia un mensaje notificando el éxito de la copia (Figura 36).

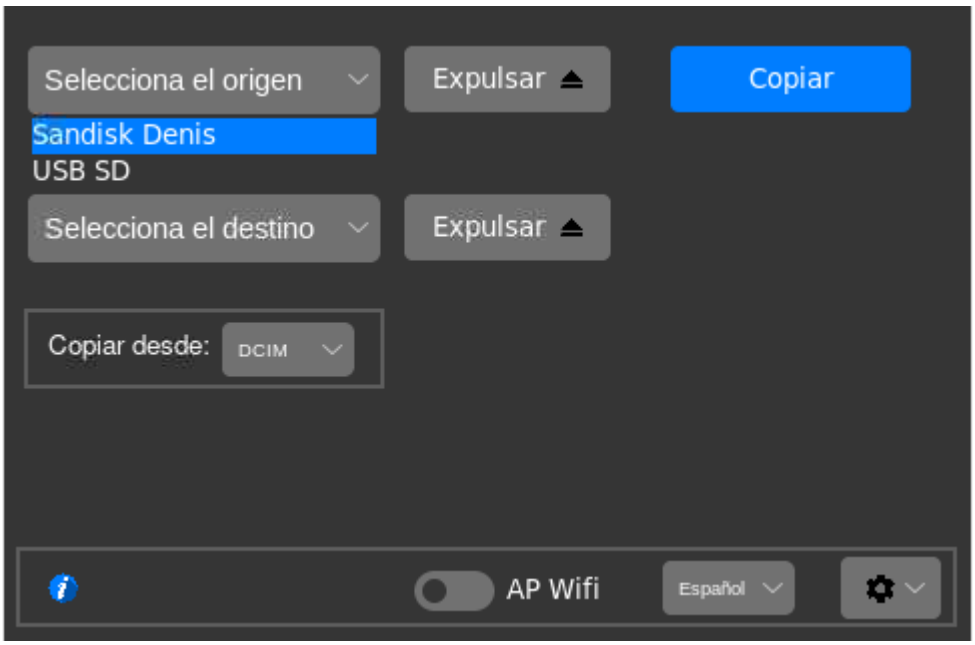

Figura 33- Combobox dispositivos de origen

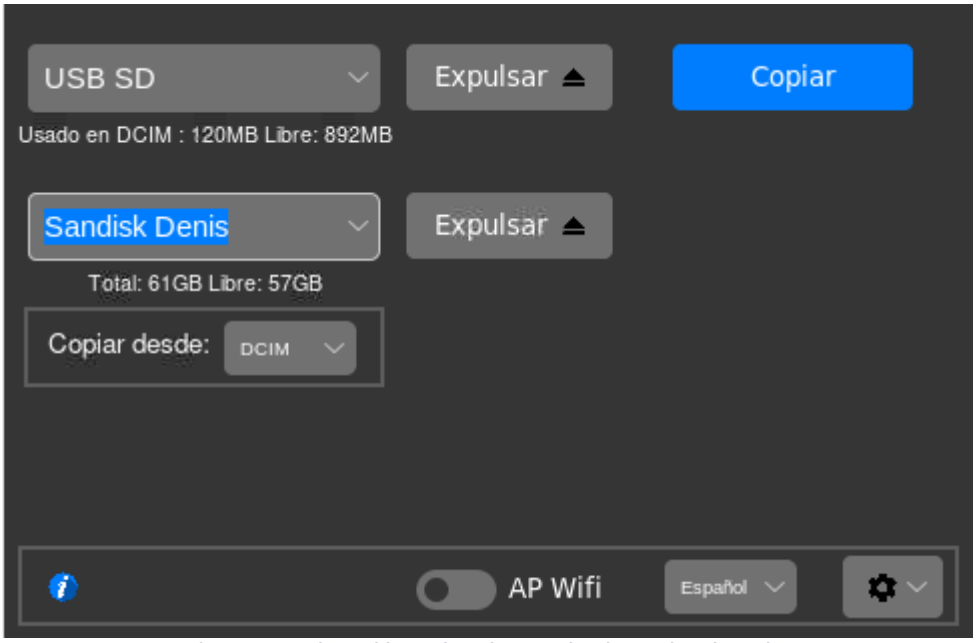

Figura 34- Dispositivos de origen y destino seleccionados

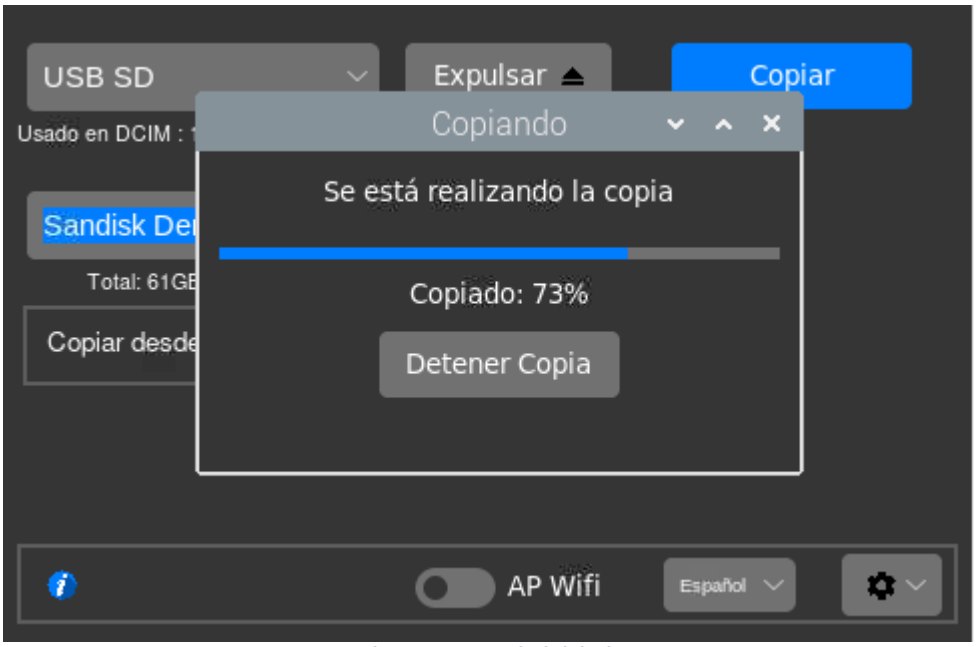

Figura 35- Copia iniciada

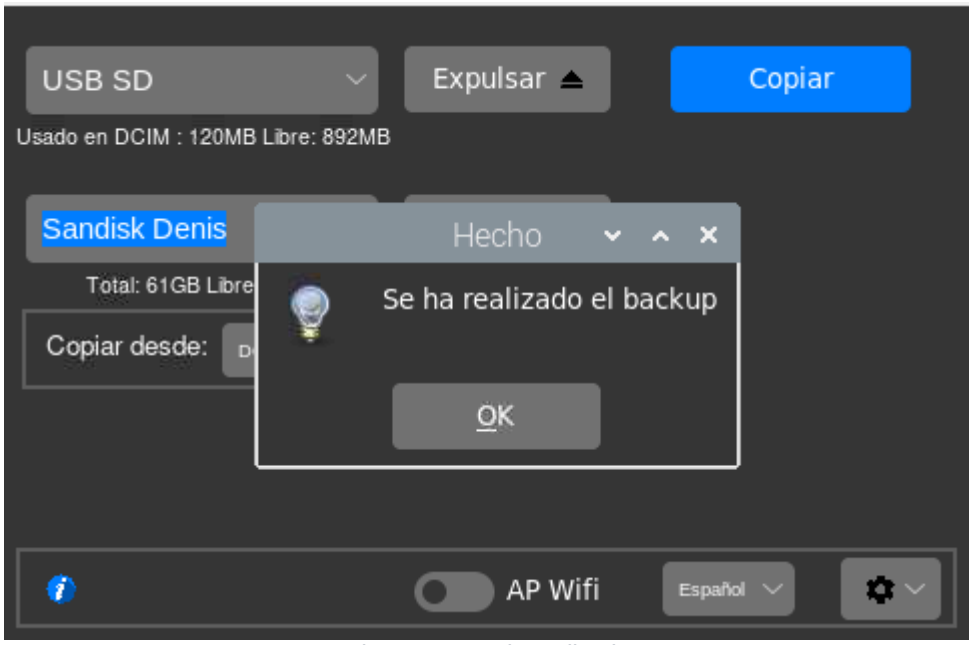

Figura 36- Copia realizada

En caso de querer copiar todo el contenido del dispositivo de origen al dispositivo de destino se selecciona ***"copiar desde raíz"*** (Figura 37):

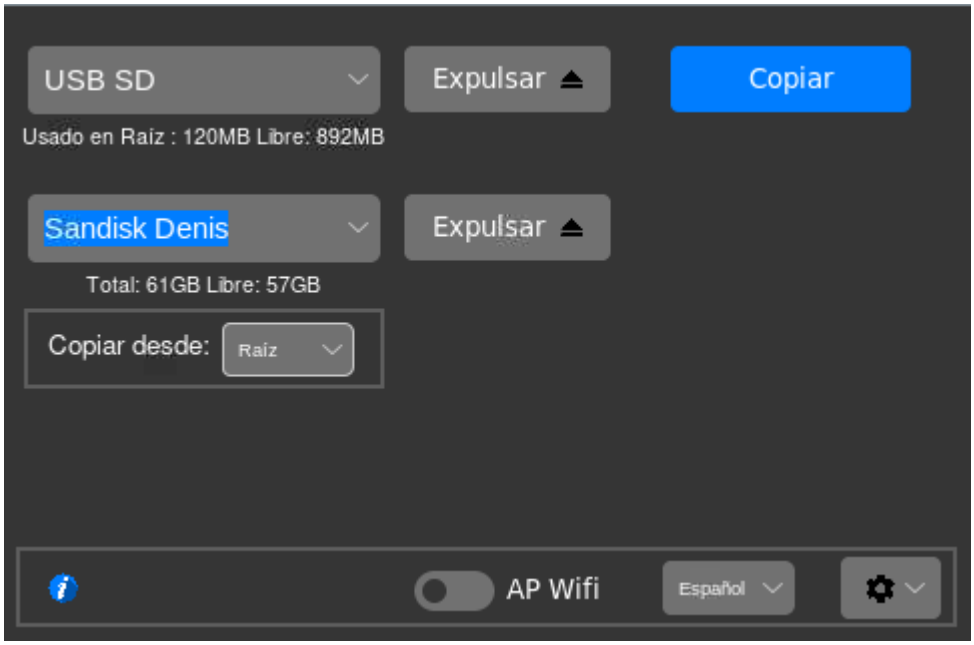

Figura 37- Copiar desde Raíz

3.2 Ver instrucciones de uso

Para ver las instrucciones de uso se pulsa el botón azul “ *i* ” situado en la esquina inferior izquierda (Figura 38).

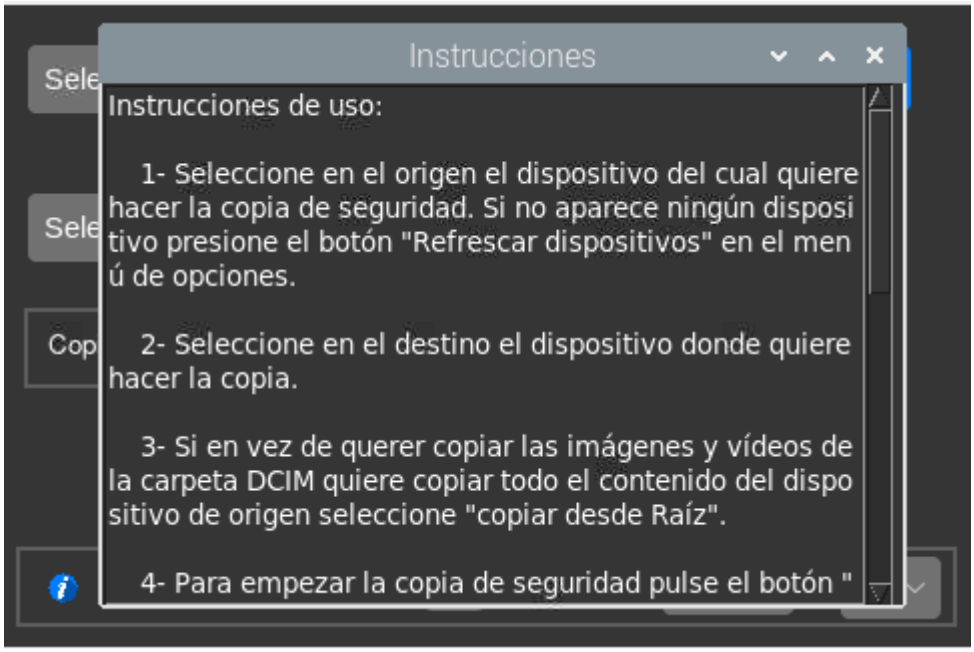

Figura 38- Instrucciones de uso

3.3 Cambiar de idioma

Para cambiar el idioma de la interfaz de usuario se puede seleccionar el mismo desde el desplegable situado en la zona inferior central de la pantalla (Figura 39, Figura 40, Figura 41).

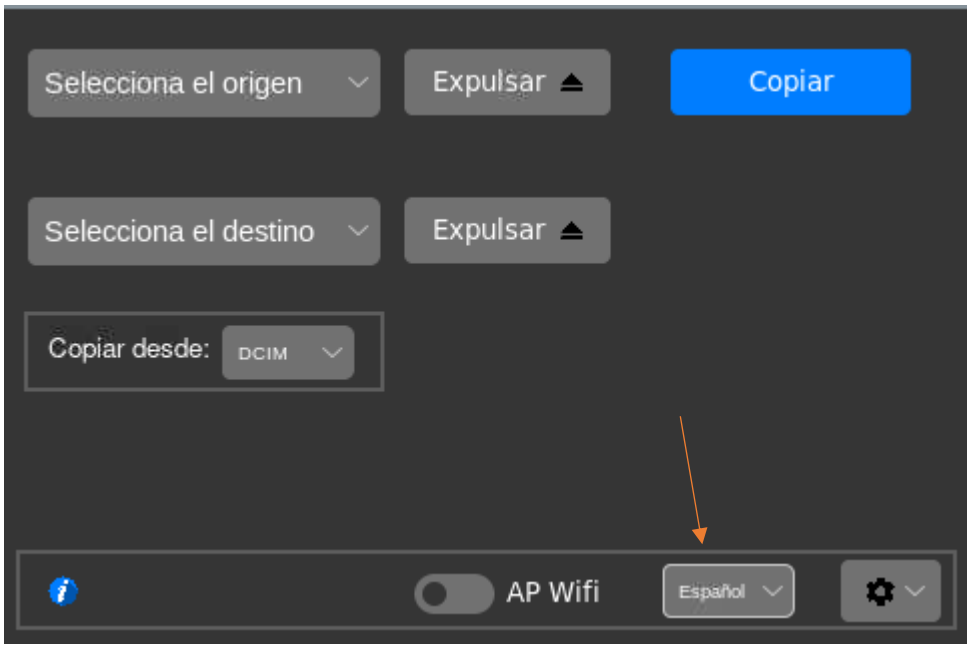

Figura 39- Cambiar idioma de la interfaz

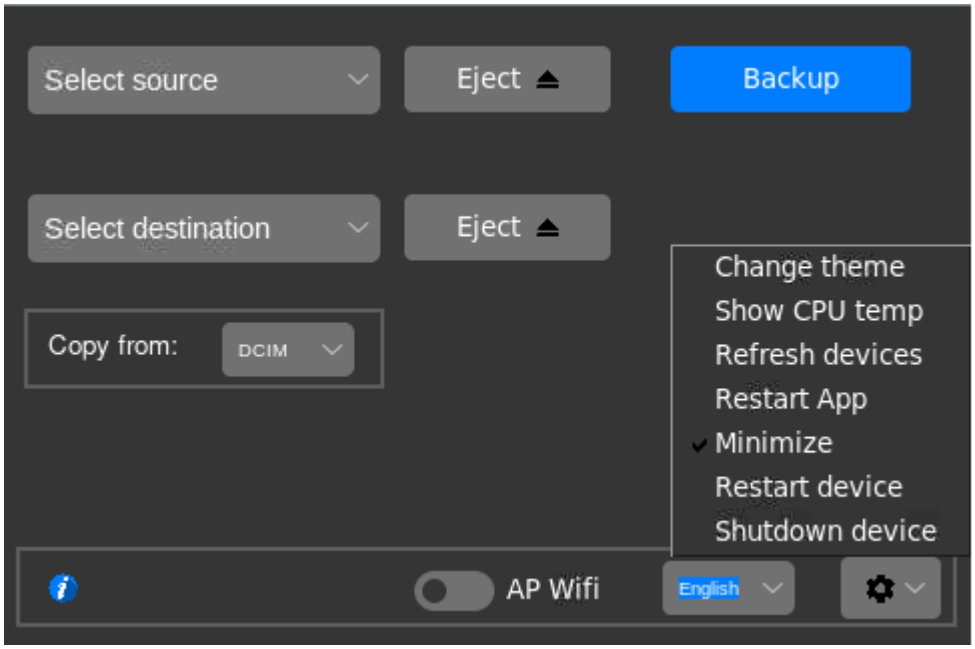

Figura 40- Interfaz en inglés

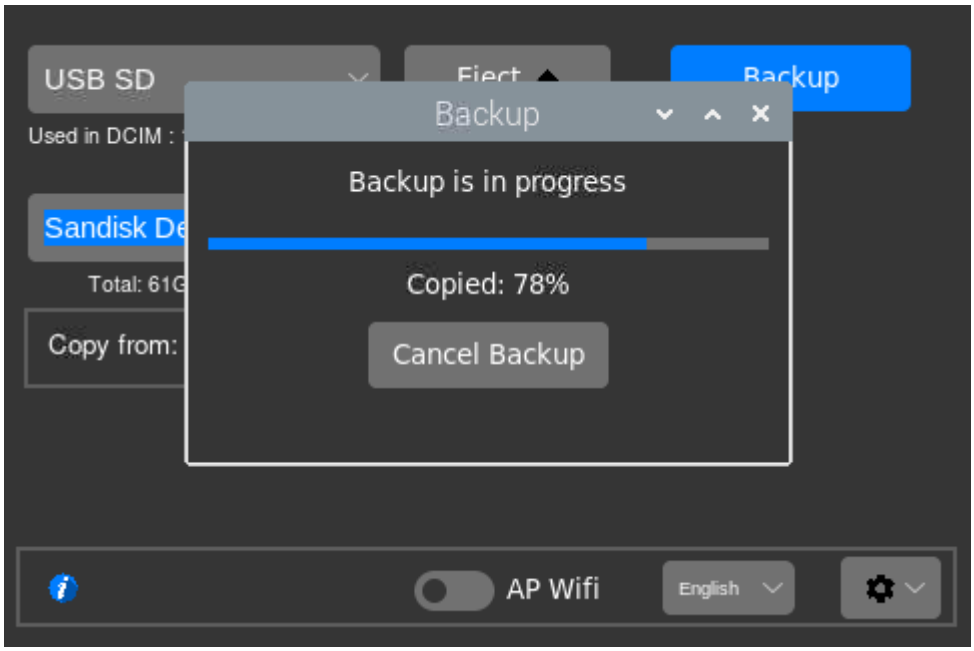

Figura 41- Interfaz en inglés 2

3.4 Cambiar tema

Para cambiar entre el tema claro y oscuro de la interfaz se ha de pulsar sobre el icono de rueda dentada ⚙ y seleccionar la opción “Cambiar tema” (Figura 42, Figura 43).

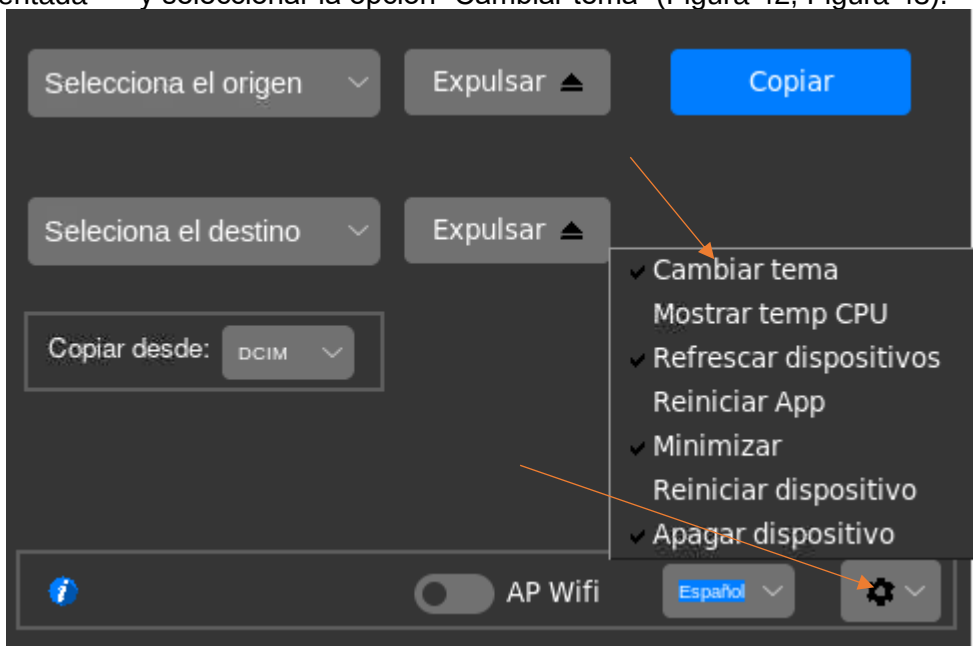

Figura 42- Cambiar tema

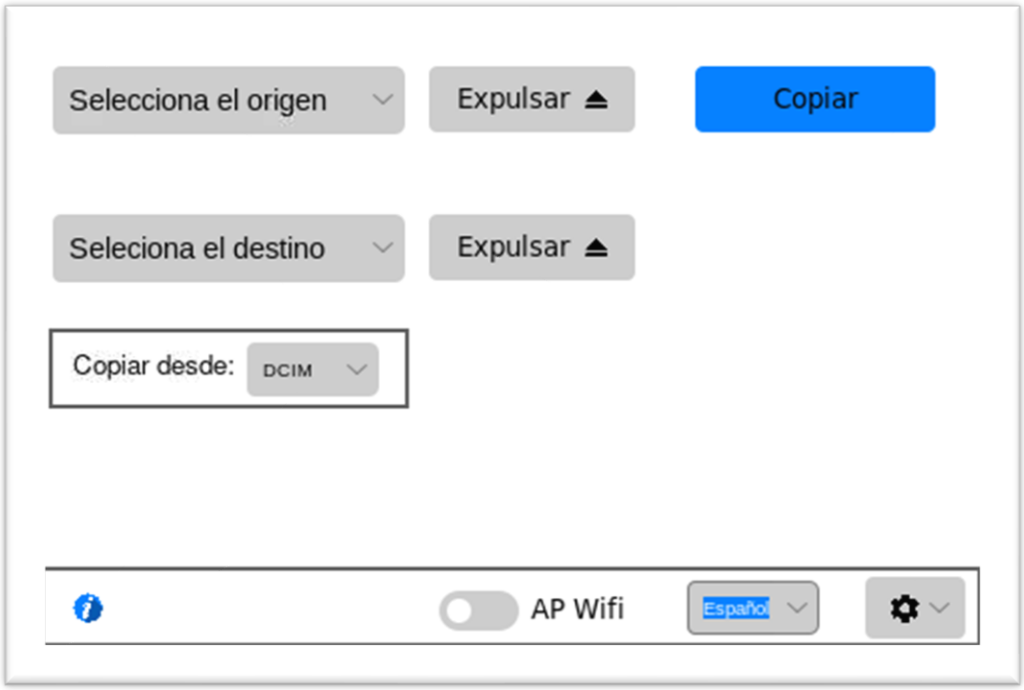

Figura 43- Tema cambiado

3.5 Mostrar temperatura de CPU

Para mostrar la temperatura de la CPU se pulsa sobre la opción “Mostrar temp CPU” en el menú desplegable de la rueda dentada (Figura 44).

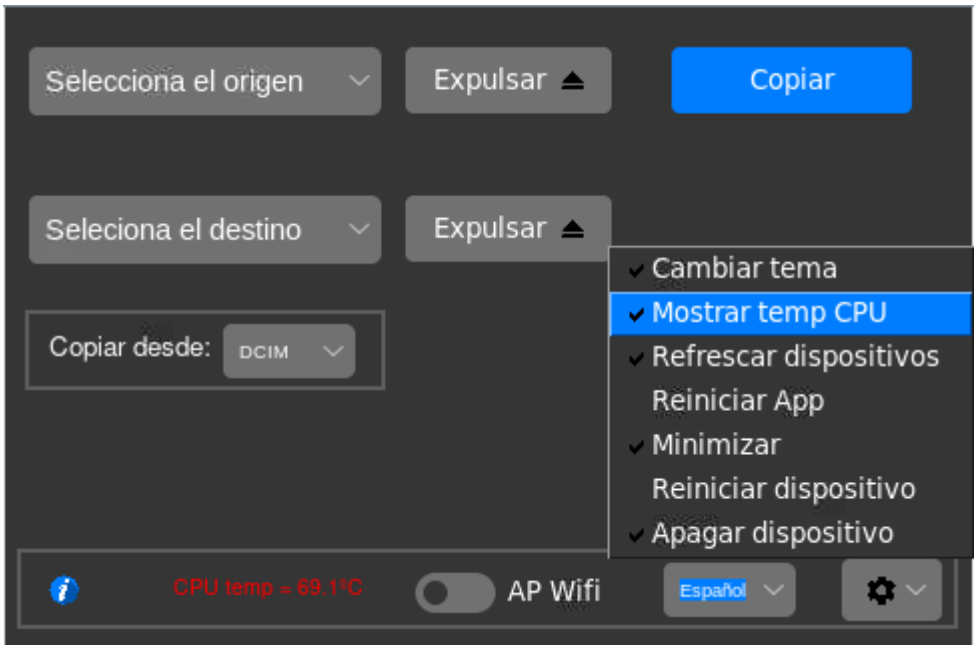

Figura 44- Mostrar temperatura CPU

3.6 Refrescar dispositivos

Esta función permite refrescar de forma manual la lista de dispositivos conectados.
Se activa de igual forma que las opciones anteriormente mencionadas.

3.7 Reiniciar la aplicación

Permite volver a iniciar la aplicación desde cero. Útil en caso de fallo.

3.8 Reiniciar o apagar el dispositivo

Estas funciones permiten reiniciar el dispositivo en caso necesario o apagarlo totalmente (Figura 45).

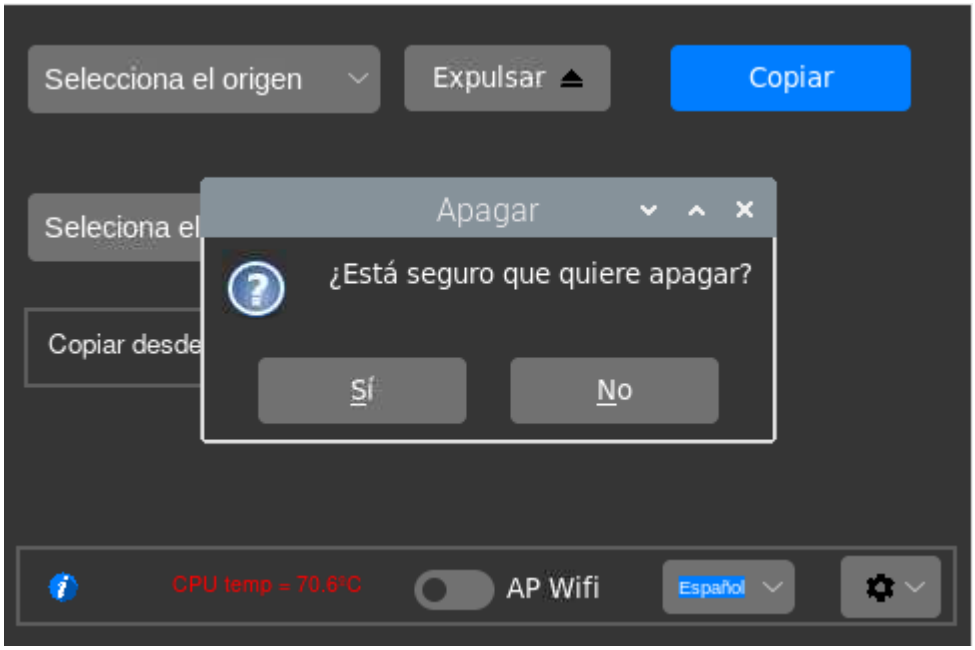

Figura 45- Apagar dispositivo

3.9 Visualizar o descargar archivos desde smartphone

Esta utilidad sirve para visualizar desde un dispositivo móvil los archivos que contienen los dispositivos de almacenamiento conectados. Permite además descargarlos de forma inalámbrica al dispositivo móvil, en caso de necesidad. Esto es útil, por ejemplo, para revisar con un cliente las fotografías que se están realizando y disponer de archivos brutos de ejemplo.
Para hacer uso de esta característica, lo primero que se debe hacer es activar el punto de acceso wifi desde el “switch” “AP Wifi” del dispositivo (Figura 46).

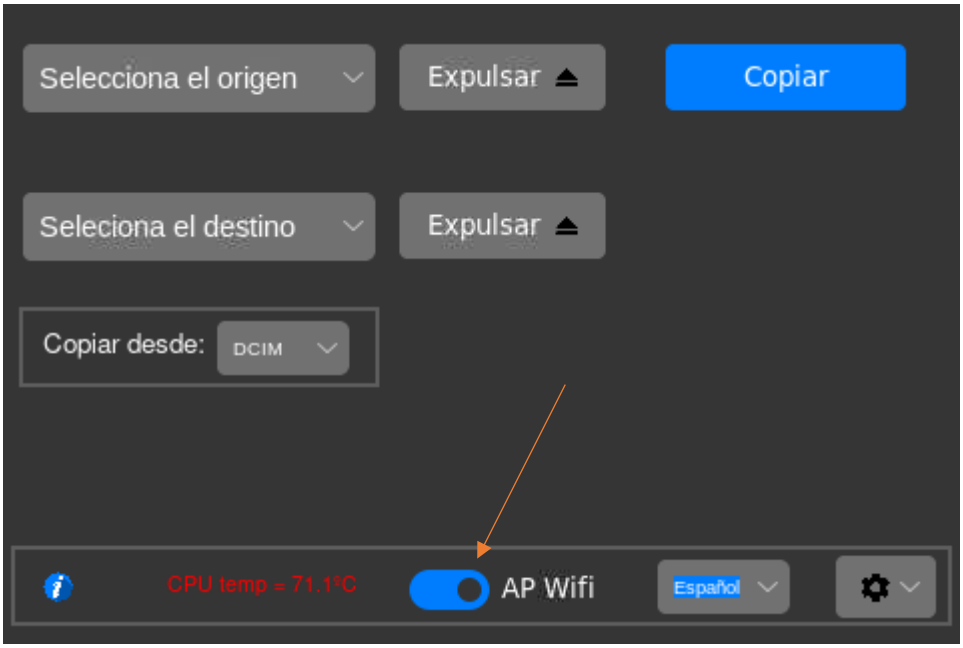

Figura 46- Activar AP Wifi

El siguiente paso es conectarse desde el smartphone a la red wifi creada (Figura 47).

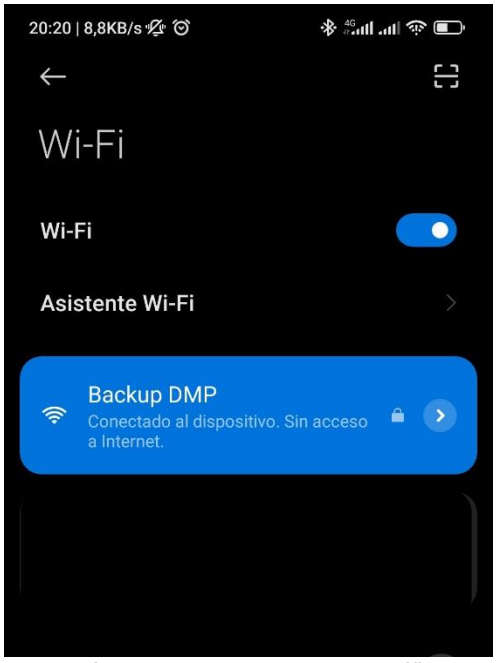

Figura 47- Conectarse a AP Wifi

Una vez conectados a la red Wifi se procede a entrar al apartado “Remoto” del gestor de archivos del smartphone (Figura 48).

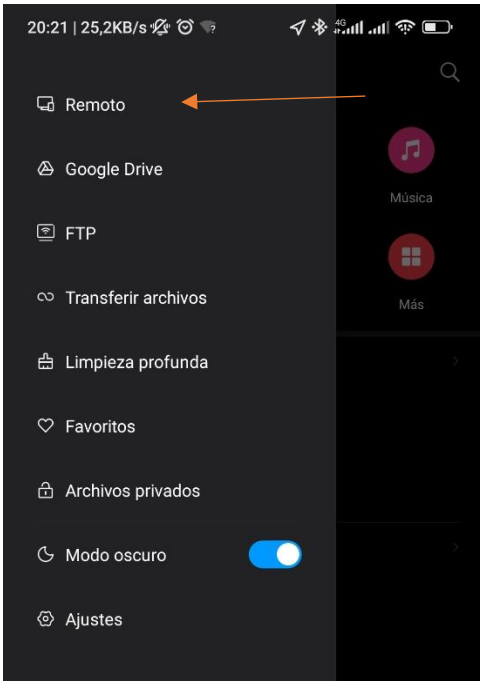

Figura 48- Apartado Remoto del gestor de archivos

El siguiente paso es añadir un dispositivo remoto (Figura 49).

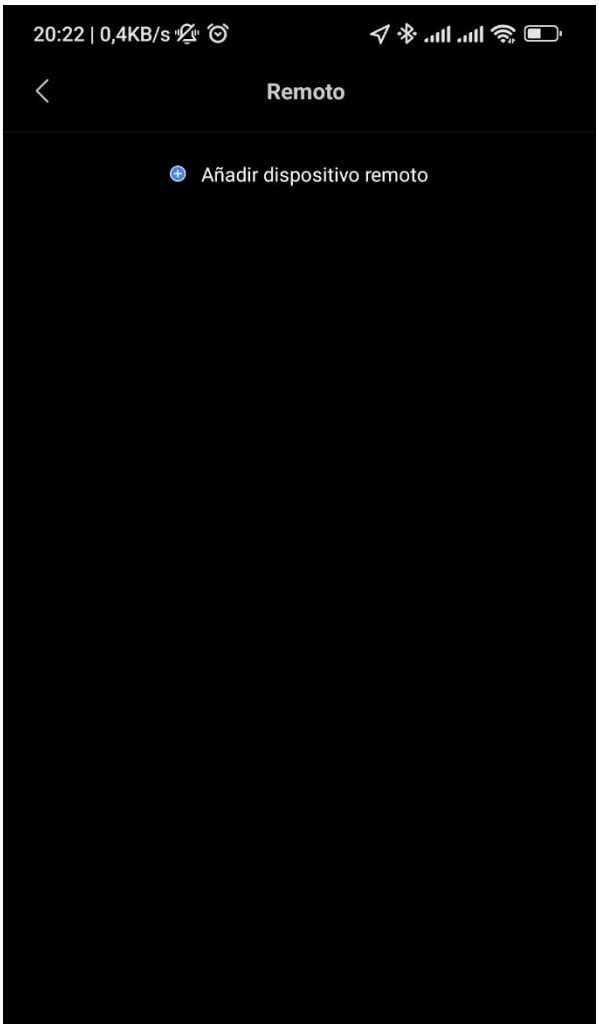

Figura 49- Añadir dispositivo remoto 1

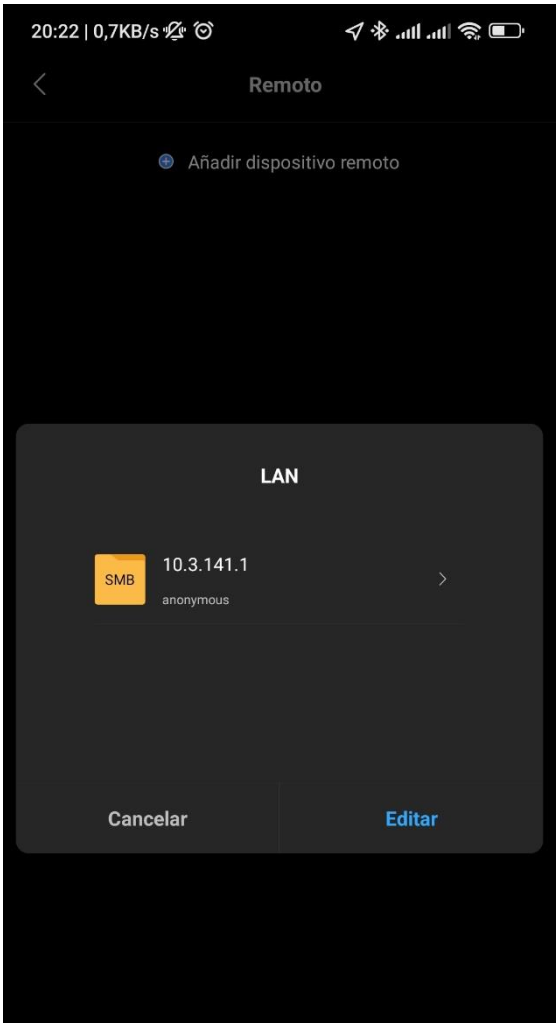

Figura 50- Añadir dispositivo remoto 2

Se puede observar en la Figura 50 que la IP del dispositivo aparece listada.

El siguiente paso es hacer el login con los datos de usuario (Figura 52) y ya habría acceso a los dispositivos de almacenamiento conectados al dispositivo de copias (Figura 51).

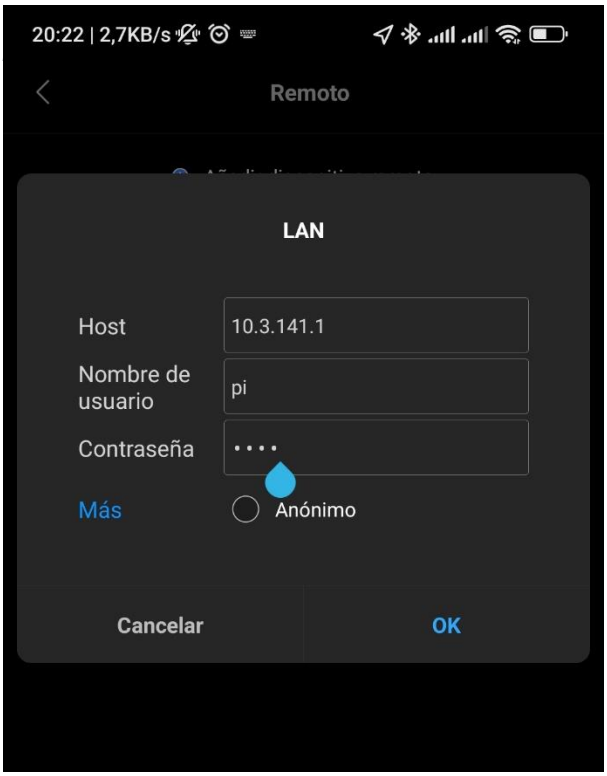

Figura 52- Login

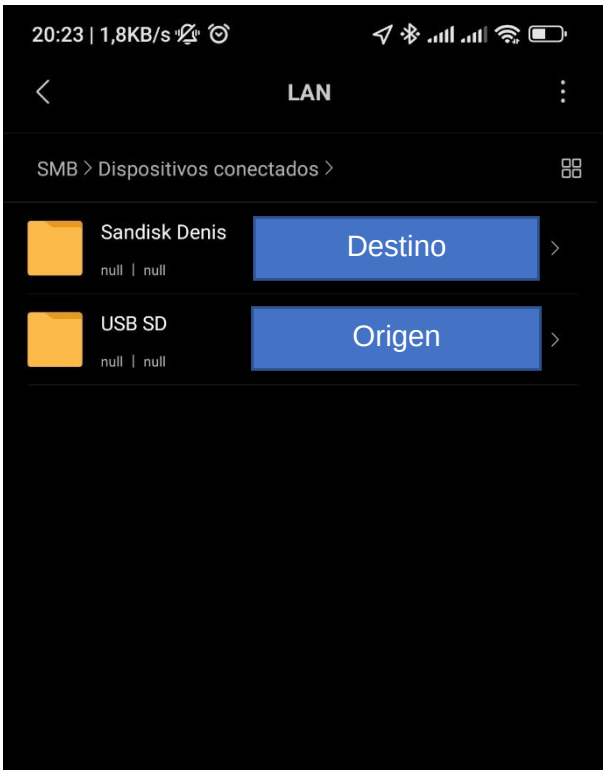

Figura 51- Acceso a dispositivos

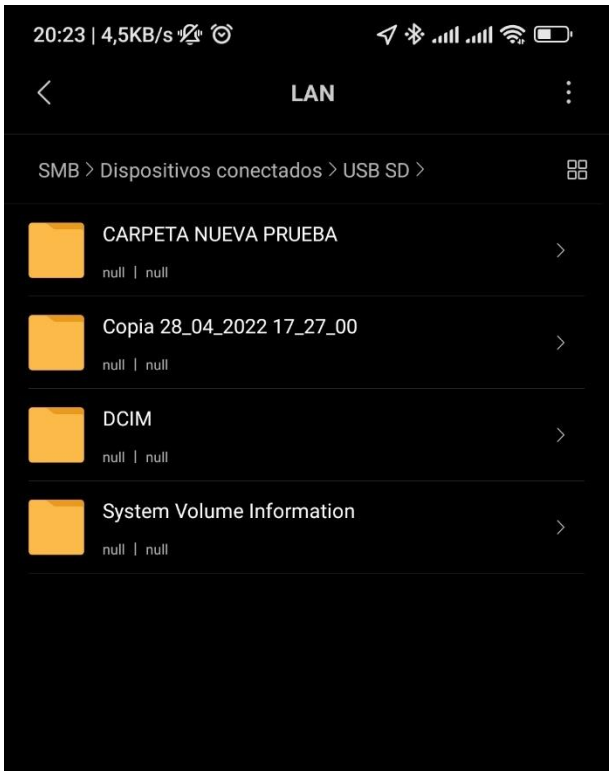

Figura 54- Acceso a USB SD

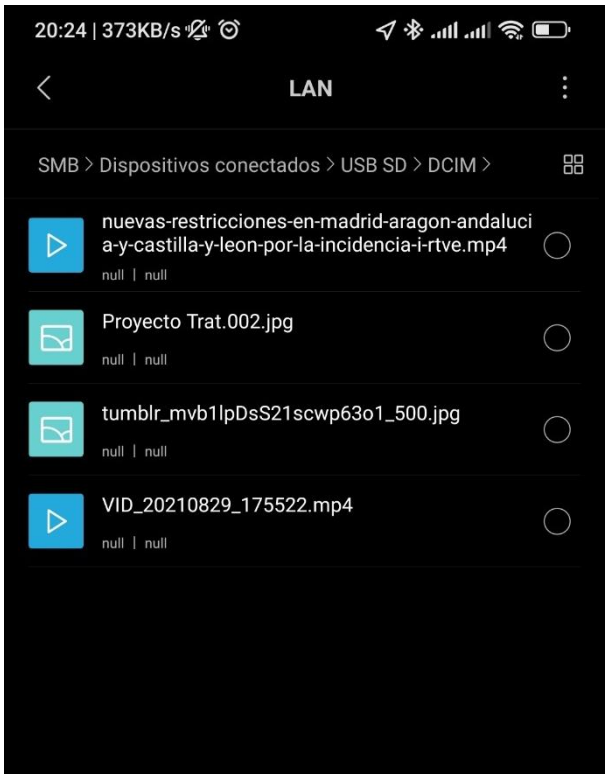

Figura 55- Carpeta DCIM de USB SD

4.Presupuesto

Para la creación de este dispositivo se ha hecho uso de varios componentes, el presupuesto de estos se muestra en la Tabla 2. No se han tenido en cuenta las horas de ingeniería a la hora de hacer el presupuesto, en caso de hacerlo se calcularía basándose en el tiempo dedicado al desarrollo del proyecto.

Componente	Precio
Raspberry Pi 4 2GB	57€
Carcasa	29€
Pantalla Táctil 3.5”	
Disipadores aluminio	
Adaptador tarjeta SD a USB	8€
Powerbank 6Ah 10W	10€
Tarjeta microSD 32GB	8€
TOTAL	**112€**

Tabla 2- Presupuesto creación del dispositivo

Se aprecia que por 112€ el dispositivo tiene un precio contenido, teniendo en cuenta los precios de los accesorios de fotografía, que suelen ser bastante elevados.

5. Conclusiones

Se puede concluir que durante el desarrollo de este trabajo se ha experimentado cómo es el desarrollo de un proyecto de estas características. Se ha pasado por diferentes fases de investigación sobre la materia, aprendizaje del lenguaje Python y Tkinter, así y como de Bash. Además, se ha aprendido sobre la plataforma de Raspberry Pi y el funcionamiento de su sistema operativo Raspberry Pi OS.
Un punto importante que destacar es que se ha logrado salvar todos los problemas acontecidos, a pesar de haber seguido un sistema de autoaprendizaje.
Finalmente, se ha conseguido crear un dispositivo totalmente funcional que cumple con los requisitos de diseño, planteados en las primeras fases del proyecto. Es fácil y rápido de utilizar, es portátil, de tamaño reducido, y es capaz de ser usado en entornos donde un ordenador portátil sería más incómodo y correría mayor riesgo de rotura.

5.1 Posibles mejoras

Aunque el dispositivo que se ha creado es totalmente funcional, hay una serie de mejoras que podrían aplicarse.

5.1.1 Software

Entre las posibles mejoras de software se valoran principalmente tres:

1. **Idiomas:** Mejora de la implementación del cambio de idioma de la interfaz de usuario con un modelo más escalable. Actualmente el sistema de cambio de idioma funciona mediante el uso de comparaciones con if/else, en el caso de que la variable de idioma seleccionada mediante el ComboBox sea “español” se leen unas variables con el texto en español, en caso de que no sea “español” será “inglés” ya que se han implementado solo esas dos opciones y se leerán las variables con el texto en inglés. La mejora de este apartado debería permitir añadir más idiomas de una forma más elegante y fácil que crear árboles if/else o switch-case.
2. **División de archivos:** Dividir todo el código de la App en varios archivos según su funcionalidad. Por ejemplo; GUI, funciones principales, funciones extra… etc. La división del código de esta manera fomenta una programación más limpia y organizada, permite además localizar con mayor facilidad los métodos desarrollados y trabajar de manera más eficiente.
3. **Optimización del código:** Optimizar el código para evitar ciclos innecesarios y conseguir mayor velocidad y estabilidad. Esto aporta una reducción del gasto energético del dispositivo, de los tiempos de carga y ejecución y disminuye las probabilidades de “crash”.

5.1.2 Hardware

En cuanto a las posibles mejoras de hardware se valoran diferentes opciones como:

1. **Carcasa:** Mejora de la carcasa del dispositivo, haciendo uso de una más resistente y estanca.
2. **Refrigeración:** Uso de disipador pasivo de mayor tamaño que permita la estanqueidad de la carcasa y a la vez unas temperaturas óptimas. Esto permitiría utilizar el dispositivo en situaciones ambientales más extremas, como lluvia, barro, arena… etc.
3. **Pantalla:** Uso de una pantalla capacitiva de mayor calidad. Una pantalla con mayor calidad de imagen y mejor respuesta táctil permitiría un uso más agradable.
4. **Batería interna:** Uso de una batería interna para no depender de adaptadores de corriente alterna o baterías externas.

6. Bibliografía

Academy, P. L. (2020). *Python para principiantes.*

Álvarez, A. C. (2016). *Python 3. Curso Practico.* RA-MA S.A.

Carbon. (2022). *Crear imágenes del código.* Obtenido de https://carbon.now.sh/

Codemy. (2020). *Progress bars with Tkinter.* Obtenido de https://www.youtube.com/watch?v=Grbx15jRjQA

Csatlas. (2021). *Import another python file as module.* Obtenido de https://csatlas.com/python-import-file-module/

Dexterindustries. (s.f.). *Run a program on Raspberry Pi at Startup.* Obtenido de https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/

edX. (2021). Python: Aprender a programar.

Geekforgeeks. (2021). *Open a new window with a button in Python Tkinter.* Obtenido de https://www.geeksforgeeks.org/open-a-new-window-with-a-button-in-python-tkinter/

Geekforgeeks. (s.f.). *Add image on a Tkinter button.* Obtenido de https://www.geeksforgeeks.org/python-add-image-on-a-tkinter-button/

J2LOGO. (s.f.). *Programación orientada a objetos en Python*. Obtenido de https://j2logo.com/python/tutorial/programacion-orientada-a-objetos/

Plus2net. (s.f.). *Python Tkinter Combobox.* Obtenido de https://www.plus2net.com/python/tkinter-Combobox.php

Programmerclick. (s.f.). *OptionMenu Tkinter.* Obtenido de https://programmerclick.com/article/194777629/

Python docs. (2022). *Shutil library.* Obtenido de https://docs.python.org/3/library/shutil.html#directory-and-files-operations

Python Guides. (2021). *Tkinter MessageBox.* Obtenido de https://pythonguides.com/python-tkinter-messagebox/

Python Tutorial. (s.f.). *Tkinter grid.* Obtenido de https://www.pythontutorial.net/tkinter/tkinter-grid/

RaspAP. (2022). *RaspAP*. Obtenido de https://raspap.com/

Raspberry para novatos. (2020). *Cómo compartir una carpeta con Samba entre tu Raspberry Pi y otro ordenador de tu red.* Obtenido de https://raspberryparanovatos.com/tutoriales/compartir-carpeta-samba-raspberry-pi-windows/

rdbedne. (2022). *Azure theme.* Obtenido de https://github.com/rdbende/Azure-ttk-theme

Stackabuse. (2018). *Creating and deleting directories with Python.* Obtenido de https://stackabuse.com/creating-and-deleting-directories-with-python/

Stackexchange. (2014). *How to eject USB device on Raspberry Pi.* Obtenido de https://raspberrypi.stackexchange.com/questions/14843/how-to-eject-usb-device-on-raspberry-pi-not-just-unmount

Stackoverflow. (2008). *How to copy a file in Python with progress bar.* Obtenido de https://stackoverflow.com/questions/274493/how-to-copy-a-file-in-python-with-a-progress-bar

Stackoverflow. (2016). *Find size and free space of the filesystem.* Obtenido de https://stackoverflow.com/questions/4260116/find-size-and-free-space-of-the-filesystem-containing-a-given-file

Stackoverflow. (2020). *How can the directory of a USB drive connected to a system be obtained?* Obtenido de https://stackoverflow.com/questions/22615750/how-can-the-directory-of-a-usb-drive-connected-to-a-system-be-obtained

The Pi. (2017). *How to use your Raspberry Pi as a wireless access point.* Obtenido de https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/

Wikipedia. (2022). *Samba.* Obtenido de https://es.wikipedia.org/wiki/Samba_(software)