3

INTRODUCCIÓN AL DESARROLLO DE INTELIGENCIA ARTIFICIAL CON TARJETAS RASPBERRY PI 4

La Raspberry Pi 4 es una computadora pequeña de placa única de la familia Raspberry Pi, la cual ha ganado popularidad en la comunidad de desarrolladores, así como en procesos industriales por su tamaño compacto, bajo costo y versatilidad.

Equipada con un procesador ARM Cortex-A72 de cuádruple núcleo, memoria RAM de hasta 8 GB y múltiples interfaces, es una herramienta potente para una amplia gama de aplicaciones.

Cuenta con almacenamiento microSD, soporte para conexiones Wi-Fi, Ethernet Gigabit y Bluetooth, así como la posibilidad de conectar unidades de almacenamiento externo a través de puertos USB 2.0 y 3.0, lo que le permite manejar grandes volúmenes de datos.

Si se conecta una cámara externa al equipo, es posible desarrollar aplicaciones las cuales integren visión por computadora y Machine Learning, útiles para proyectos de reconocimiento facial, detección de objetos y clasificación de imágenes. Aunque con una velocidad de latencia considerablemente más baja que cuando se utiliza una GPU.

A continuación, vamos a identificar todos los componentes necesarios para realizar el ensamblado del hardware con sus accesorios:

- Raspberry Pi 4 Modelo B.
- Fuente de alimentación USB-C con switch.
- Cable HDMI a mini-HDMI.
- Carcasa protectora para Raspberry Pi 4 Modelo B.
- Memoria microSD.

Si no se dispone de un ordenador con una tarjeta gráfica NVIDIA, el código funcionará al eliminar la parte relacionada con CUDA. Esto provocará que se ejecute en la CPU en lugar de la GPU, lo que resultará en un rendimiento inferior.

El primer paso es conectar de manera directa el ventilador a los pines que contengan 5v y a tierra como se muestra en la Figura 3.1. Adicionalmente se colocan los aditamentos en color negro sobre los componentes esenciales de la Raspberry los cuales permitirán disipar de una mejor manera el calor.

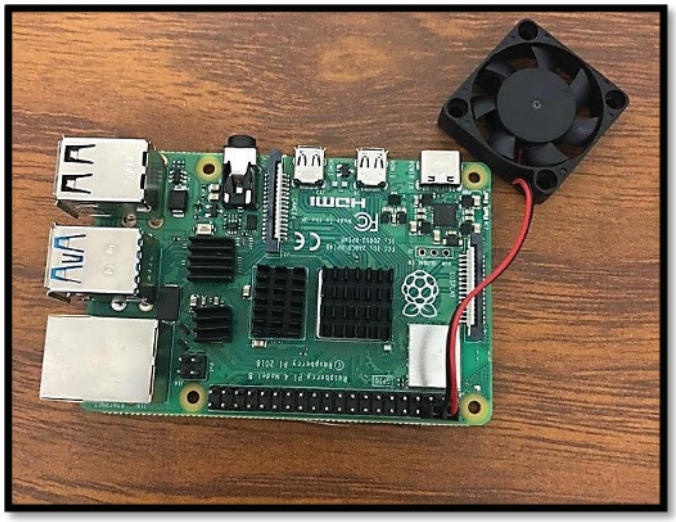

Figura 3.1. Colocación del ventilador sobre la placa de la Raspberry Pi 4

Para la configuración correcta, se muestra la siguiente guía donde se muestra la localización de los pines correspondientes, Figura 3.2.

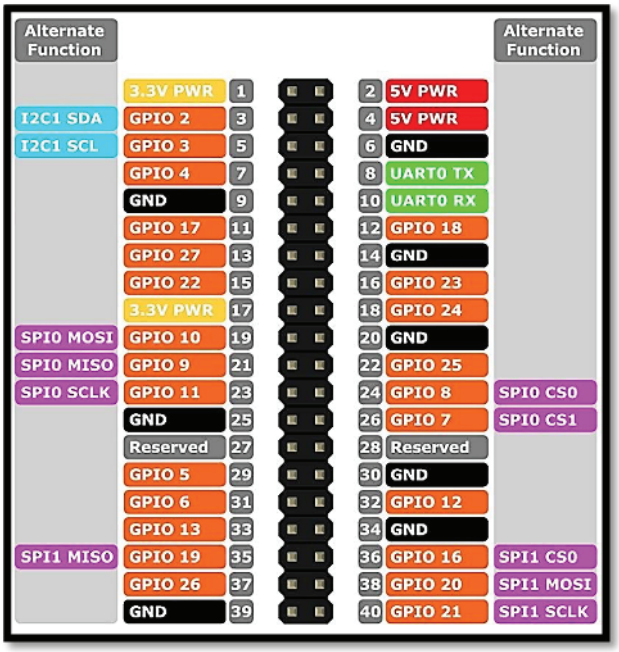

Figura 3.2. Manual de pines de las Raspberry Pi

Una vez conectado el ventilador, lo ensamblamos en la parte superior de la carcasa con ayuda de los tornillos proporcionados por el fabricante. En la parte inferior colocamos a la Raspberry cuidando la alineación de las perforaciones de la tarjeta con las de la carcasa, como se muestra en las Figuras 3.3 y 3.4.

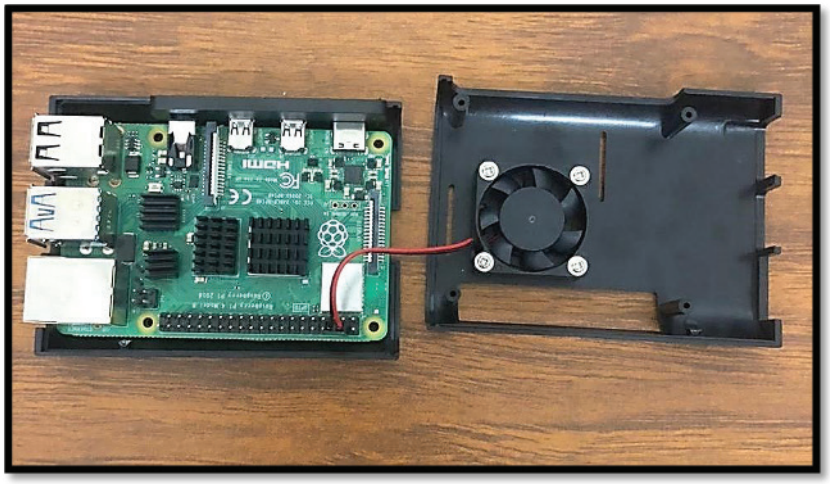

Figura 3.3. Raspberry Pi 4 y ventilador montados en la carcasa

Ensamblar y cerrar la carcasa con ayuda de más tornillos.

Figura 3.4. Raspberry Pi 4 ensamblada

Una vez cerrada la carcasa, conectar el cable de alimentación, así como también el mini HDMI para la pantalla, Figura 3.5.

Figura 3.5. Cables de la Raspberry Pi 4

Por último, se inserta la tarjeta microSD a la Raspberry con los pines hacia arriba, como se muestra en la Figura 3.6.

Figura 3.6. Inserción de la tarjeta microSD sobre la Raspberry Pi 4

3.1 ADVERTENCIAS DE USO DE RASPBERRY PI

Por su importancia para evitar accidentes y daños al equipo, a continuación, transcribimos las advertencias de fábrica para la Raspberry Pi.

- La fuente de alimentación debe proporcionar 5V y una corriente nominal mínima de 3A.
- No debe usarse con una frecuencia de reloj superior a la nominal.
- No colocarlo en una superficie conductora mientras se encuentra en funcionamiento.
- No exponer a ningún tipo de fuente de calor.
- Utilizar la Raspberry en un ambiente bien ventilado y no cubrirlo durante su uso.
- Cualquier periférico o equipo utilizado con la Raspberry debe cumplir con las normas aplicables en el país de uso.
- Evitar la exposición de la placa a fuentes de luz de alta intensidad (por ejemplo: flash de xenón o láser).

3.2 INSTALACIÓN DEL SISTEMA OPERATIVO

En los siguientes apartados, se exponen los pasos necesarios para poner en marcha una Raspberry Pi 4 pues aprenderemos a instalar y configurar el sistema operativo Raspbian OS, del cual hablaremos en el siguiente capítulo.

Para instalar el sistema operativo en la Raspberry Pi 4, se debe descargar e instalar el software *Raspberry Pi Imager* en nuestra computadora desde el sitio web oficial de Raspberry Pi.

https://www.raspberrypi.com/software/

Vamos a instalar la imagen del sistema operativo Raspbian OS que se encuentra en la carpeta **Raspberry Pi 4 SO**, la cual, se incluye en los recursos del libro. El archivo lleva por nombre *2023-05-03-raspbios-bullseye-armhf-full.img.xz*.

Posteriormente debemos extraer la tarjeta microSD de la Raspberry e insertarla en un adaptador SDHC o similar, esto para poder ingresarla a nuestra computadora. Si empleamos SDHC, debemos asegurarnos de que el adaptador no esté en modo LOCK, Figura 3.7.

Figura 3.7. Ejemplo de adaptador SDHC

Se ejecuta el software *Raspberry Pi Imager* con la tarjeta microSD dentro del computador, el proceso se muestra de las Figuras 3.8, a la Figura 3.9.

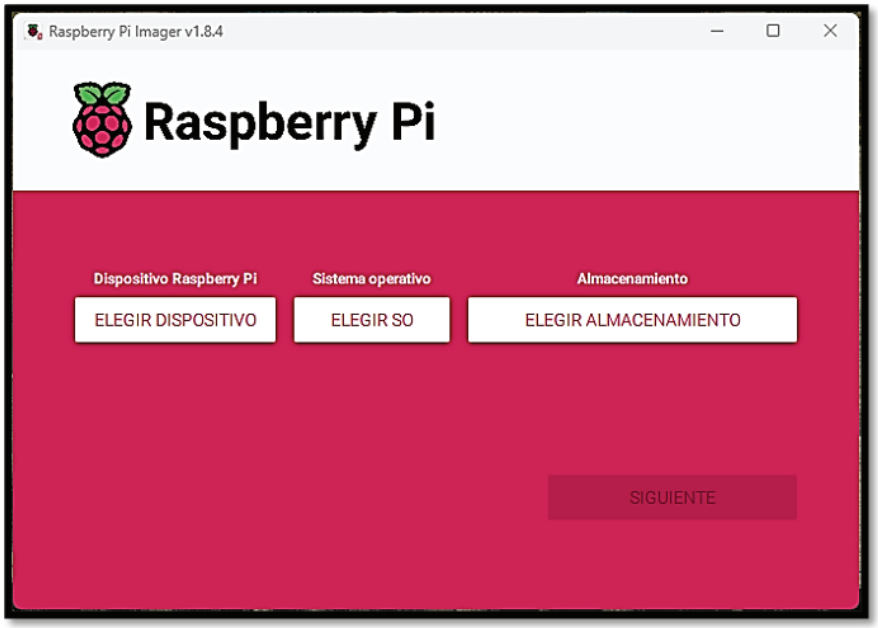

Figura 3.8. Pantalla de bienvenida del software Raspberry Pi Imager

Y en el botón *Elegir dispositivo*, elegimos la opción *Raspberry Pi 4*.

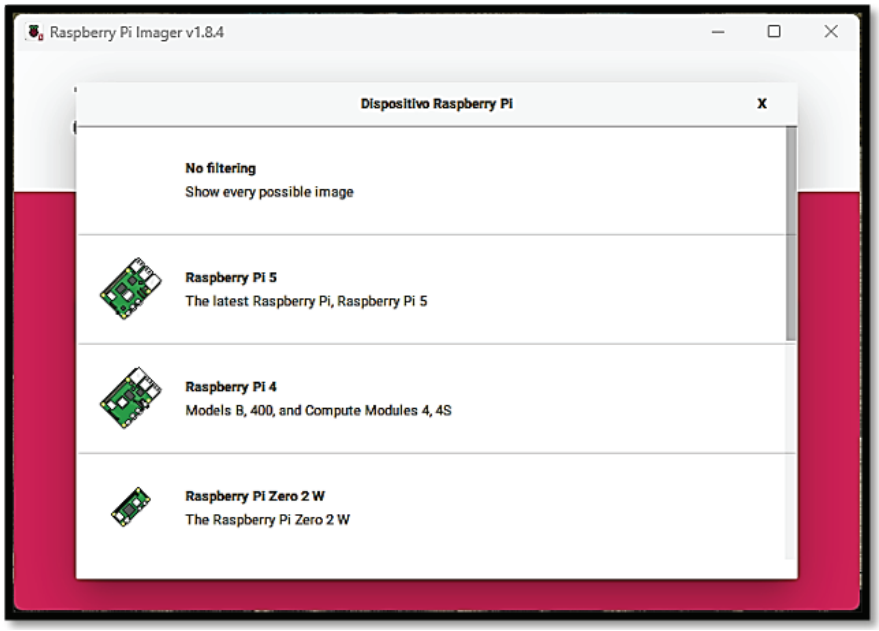

Figura 3.9. Selección del dispositivo

Posteriormente, en el botón *Elegir SO* se busca la opción *sistema personalizado* (use custom), Figura 3.10.

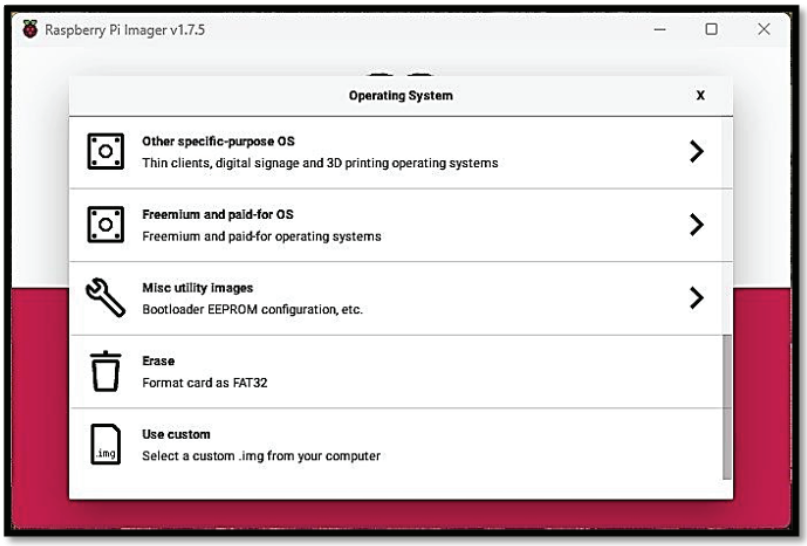

Figura 3.10. Selección del sistema operativo

Y buscamos la imagen del sistema operativo que ya obtuvimos, con ayuda del explorador de archivos de nuestra computadora, Figura 3.11.

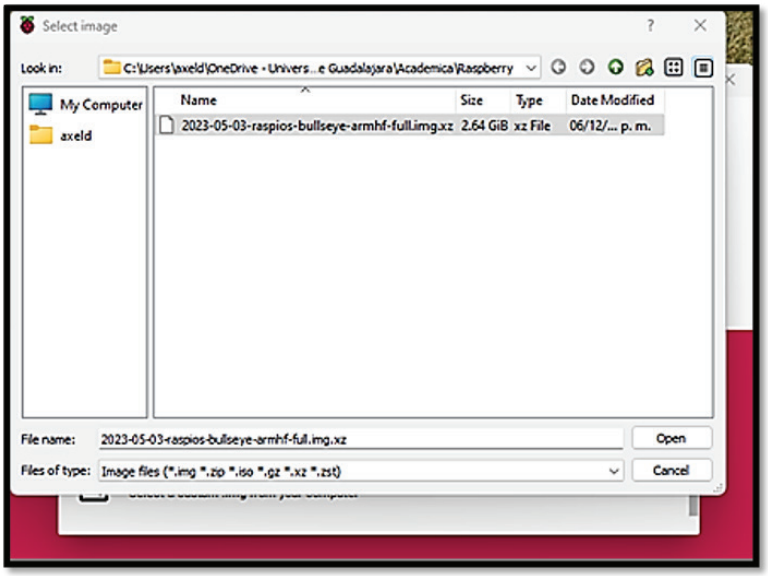

Figura 3.11. Buscar la imagen del sistema

A continuación, se elige la tarjeta *microSD* como destino en el botón *Elegir almacenamiento,* como se observa en la Figura 3.12.

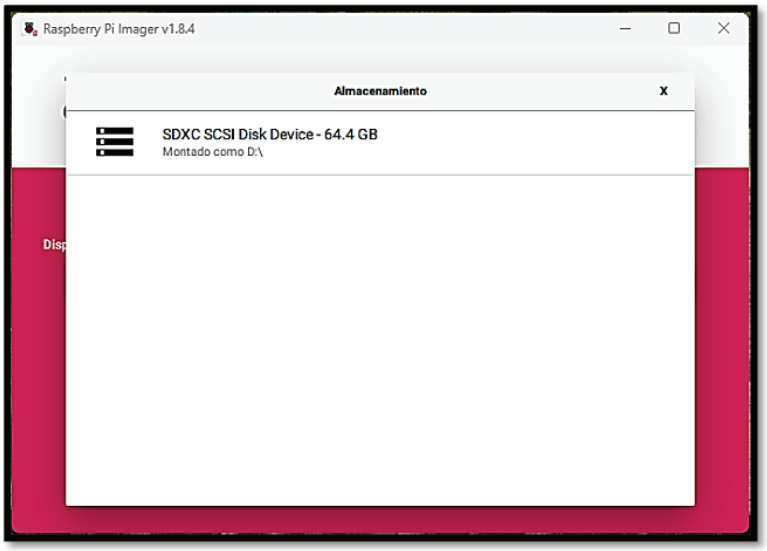

Figura 3.12. Selección del almacenamiento

Una vez hecho esto damos clic en el botón *Siguiente*, el cual desplegará un cuadro de diálogo como el siguiente, allí debemos dar clic sobre el botón *Editar ajustes,* Figura 3.13.

Figura 3.13. Posibilidad de personalizar ajustes

A continuación, se despliega la ventana de configuración, en la cual se requiere activar las casillas *Establecer nombre de anfitrión, Establecer nombre de usuario y contraseña* y *Configurar LAN inalámbrica*. Se recomienda dejar por defecto los parámetros de la primera casilla. El activar la segunda casilla, nos va a permitir asignarle a la Raspberry un nombre de usuario y una contraseña, los cuales debemos tener presentes puesto que se requieren en pasos posteriores, y el activar la tercera casilla nos va a permitir configurar la red inalámbrica a la cual se conectará el equipo una vez pueda arrancar.

Es recomendable tomar nota del nombre del anfitrión que tendrá la Raspberry, puesto que la necesitaremos en un futuro.

Cada vez que se mencione el nombre de usuario o del anfitrión en los ejercicios que realizaremos en el texto, se pondrán en cursiva, haciendo alusión a que es necesario que adaptes esos nombres por los que le hayas dado tú, como se muestra en la Figura 3.14.

Figura 3.14. Ajustes generales del sistema operativo Raspbian OS

A continuación, nos desplazamos a la pestaña *Servicios*, para habilitar la casilla de verificación *Activar SSH*. Nuevamente dejamos los parámetros que nos dá por defecto, Figura 3.15.

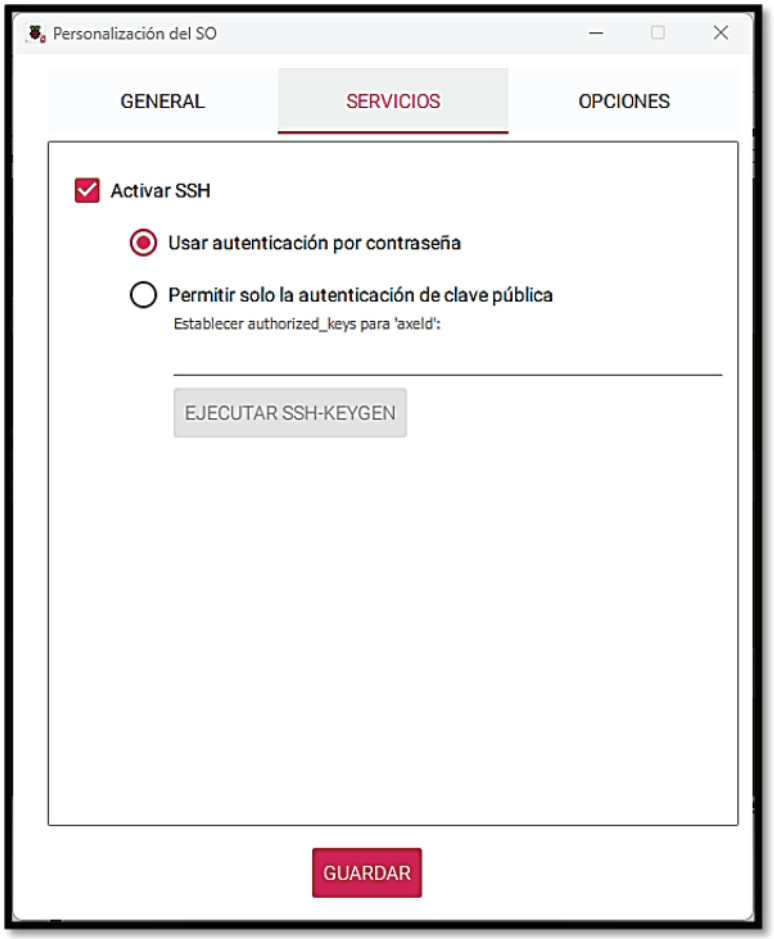

Figura 3.15. Ajustes de servicios del sistema operativo Raspbian OS

Y en la pestaña *Opciones*, mantenemos activadas las 3 casillas, Figura 3.16.

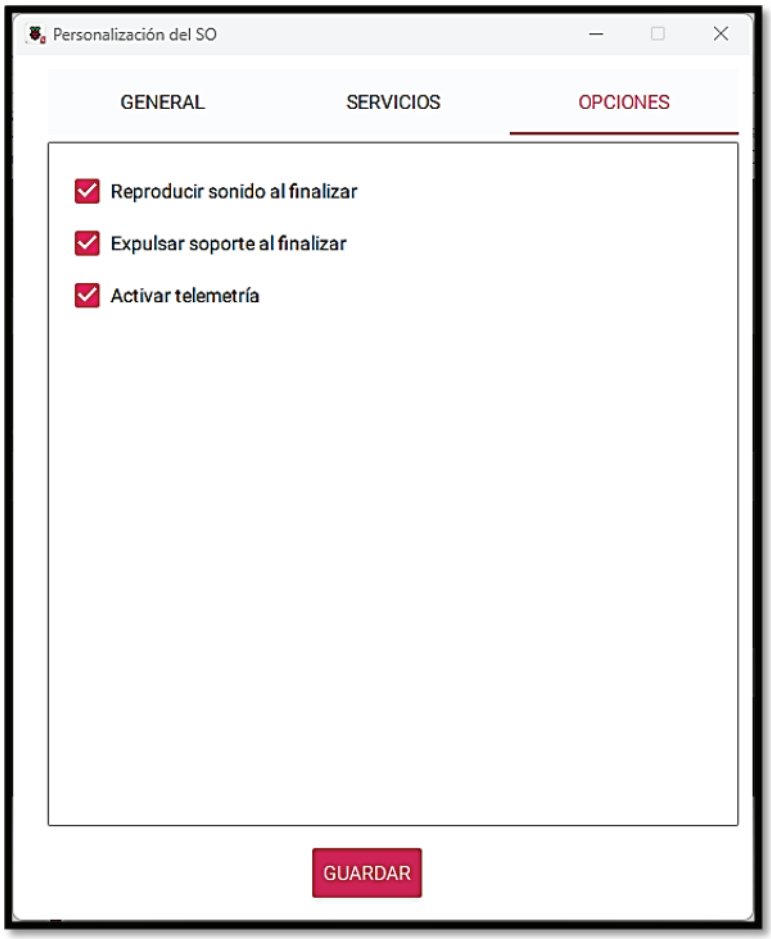

Figura 3.16. Ajustes opcionales del sistema operativo Raspbian OS

Finalmente hacemos clic en el botón *Guardar*, esto nos llevará de vuelta al cuadro de diálogo de la imagen 3.13, en la cual, ahora damos clic sobre el botón *Sí*, y esperamos a que el programa grabe la imagen del sistema operativo en la tarjeta de memoria. Si el programa nos pregunta que si queremos continuar, nuevamente presionamos en *Sí*.

Una vez finalizado el proceso de instalación y configuración, se extrae la tarjeta microSD del equipo y se inserta nuevamente en la Raspberry, como se mostró en el capítulo anterior. Ya podemos encender el dispositivo. Los elementos que componen la interfaz gráfica del sistema operativo Raspbian OS los veremos a detalle en el siguiente capítulo.

3.3 CONFIGURACIÓN DE LA RASPBERRY PI 4

En los siguientes apartados, se exponen los pasos necesarios para poder realizar una manipulación remota de nuestra Raspberry, así como la instalación de software complementario que nos permitirá desarrollar el proyecto propuesto para este apartado.

3.3.1 Control remoto de la Raspberry Pi 4

Es importante señalar que, para llevar a cabo los primeros pasos que se exponen en este apartado, se requiere tener un monitor conectado a la Raspberry para poder visualizar todo el proceso, así como un teclado y un ratón y opcionalmente un cable ethernet, si se desea mantener una conexión más estable y veloz a internet.

La Raspberry Pi 4 puede monitorearse por control remoto desde una computadora, para lo cual, es necesario descargar e instalar el programa *RealVNC* en el equipo en cuestión, el cual se puede obtener desde el siguiente enlace.

https://www.realvnc.com/es/connect/download/viewer/

3.3.2 Obtención de la dirección IP de la Raspberry

Para poder configurar la conexión remota en la Raspberry se deben ejecutar los siguientes comandos dentro de la terminal de esta. El primer comando que ingresaremos nos abrirá un menú con las configuraciones de la Raspberry, como se muestra en la Figura 3.17.

```
$ sudo raspi-config
```

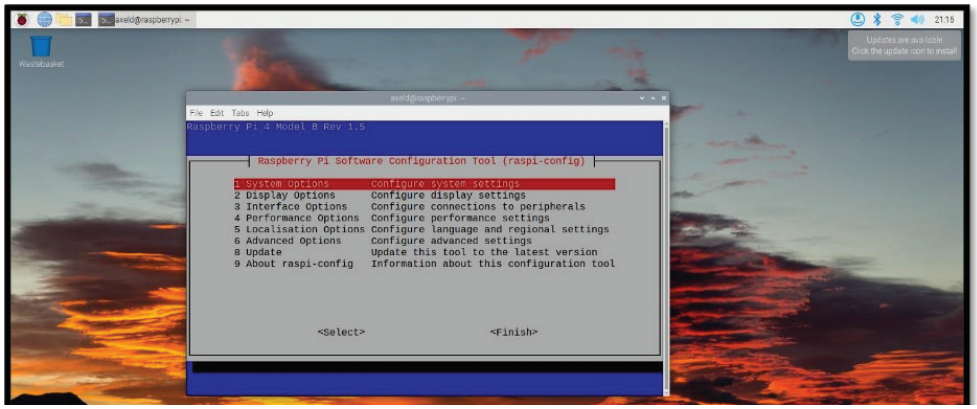

Figura 3.17. Configuraciones de la Raspberry

Dentro de ese menú debemos seleccionar las siguientes opciones, para ello podemos desplazarnos con las flechas del teclado, o con la tecla tabuladora (si desconoces la apariencia de esta tecla, no te preocupes, se muestra en la Figura 3.18).

Figura 3.18. Tecla tabuladora

```
$ 3 Interface Options
$ I3 VNC
$ <Yes>
$ <Ok>
```

Y reiniciar la Raspberry.

Finalmente, para poder controlar de forma remota la Raspberry y omitir tener conectado un monitor, un teclado y un ratón al equipo, se requiere obtener su dirección IP para lo cual se utiliza el comando *ifconfig*, el cual se ingresa en la misma terminal que ya se tenía abierta[5].

La parte que se encuentra subrayada de la siguiente imagen, nos proporciona la dirección IP del dispositivo, la cual, en este ejemplo sería la 192.168.68.65, como se muestra en la Figura 3.19.

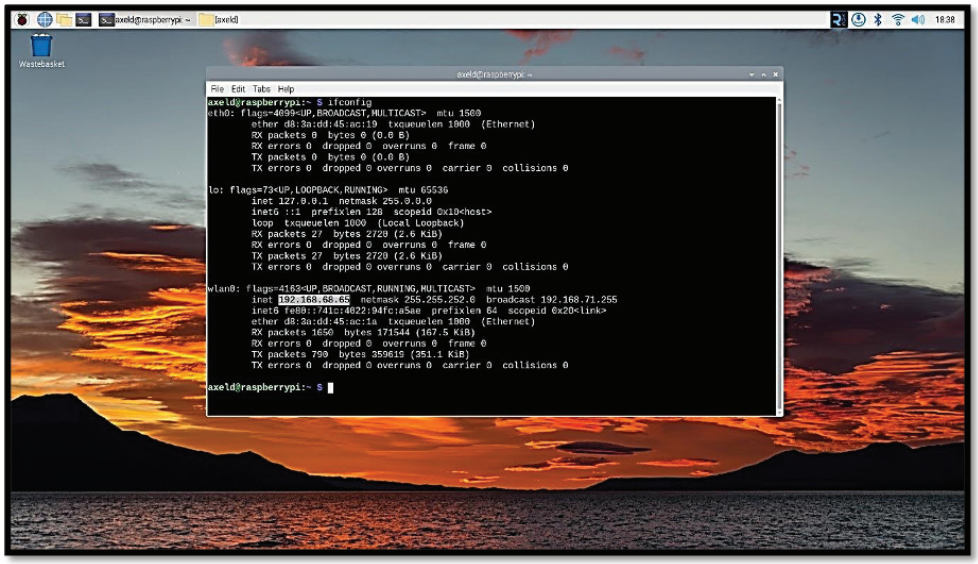

Figura 3.19. Obtener la dirección IP de la Raspberry si se encuentra conectada a internet de forma inalámbrica (wi-fi)

Si por alguna razón, el dispositivo no contara con una dirección IP, es decir, en la terminal no se mostrará el apartado *wlan0* y solo aparecieran los apartados *lo* y *eth0*, debes cerciorarte de que el dispositivo esté conectado correctamente a internet de forma inalámbrica.

Ahora bien, si la Raspberry se encuentra conectada mediante un cable ethernet y no vía inalámbrica a internet, entonces la dirección IP que se requiere es la que se despliega en el apartado *eth0*, como se muestra en la Figura 3.20, la cual, en este ejemplo sería la 192.168.68.90.

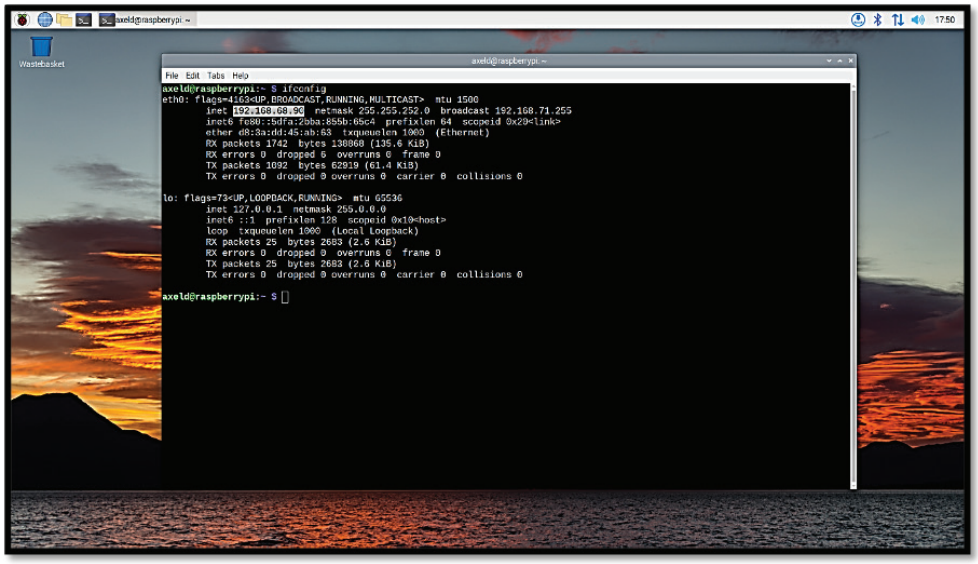

Figura 3.20. Obtener la dirección IP de la Raspberry si se encuentra conectada a internet mediante un cable ethernet

La dirección IP asignada a nuestra Raspberry se ingresa dentro del programa *VNC Viewer* de la computadora por la cual nos conectaremos a la Raspberry, dentro del campo RVNC CONNECT. Damos un enter y nos aparecerá una ventana como la siguiente, Figura 3.21.

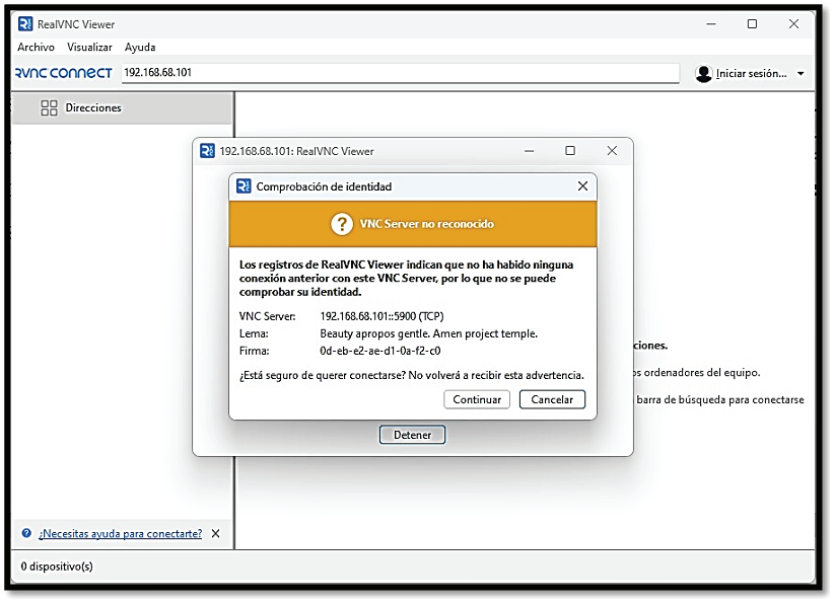

Figura 3.21. Conectarse remotamente a la Raspberry

Después ingresamos las credenciales de la Raspberry, es decir, el usuario y la contraseña que configuramos cuando instalamos el sistema operativo, Figura 3.22.

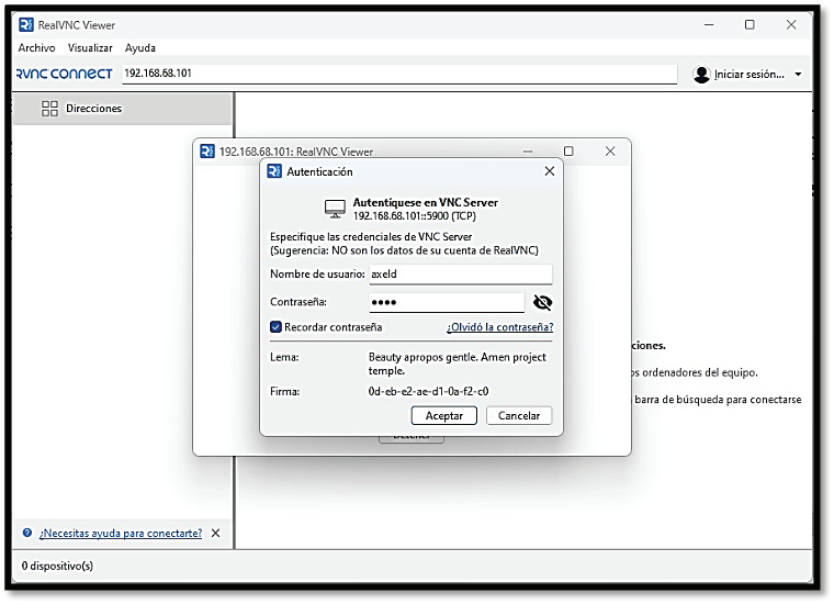

Figura 3.22. Ingresar credenciales de la Raspberry para realizar conexión remota

¡Listo! Ya podemos utilizar nuestra Raspberry desde nuestra computadora sin necesidad de tenerla conectada a un monitor y demás dispositivos periféricos.

Figura 3.23. Escritorio remoto de la Raspberry

3.3.3 Problemas de visualización

Si aparecieran problemas de visualización (es decir, la conexión fue exitosa pero el despliegue de la ventana no es la mejor), podemos ingresar a las configuraciones de la Raspberry nuevamente desde la misma terminal y adaptar la resolución de la pantalla.

```
$ sudo raspi-config
$ 2 Display Options
$ D5 VNC Resolution
```

Elegir la resolución más conveniente.

```
$ <Ok>
```

Finalmente presionamos la tecla escape (esc) y nos pedirá reiniciar la Raspberry.

Pero si definitivamente no pudimos conectarnos de forma remota a nuestra Raspberry, primero debemos cerciorarnos de si la Raspberry permite la conexión SSH, para ello, ingresaremos de nuevo a la configuración del dispositivo y seleccionar las siguientes opciones. Al finalizar es necesario reiniciar la Raspberry e intentar de nuevo la conexión remota.

```
$ 3 Interface Options
$ I2 SSH
$ <Yes>
```

Si la solución anterior no resolvió el problema, lo siguiente que tenemos que hacer es, revisar si el servidor VNC se instaló de forma correcta, para ello, lo mejor será quitarlo e instalarlo nuevamente del sistema, por lo que necesitaremos ingresar los siguientes comandos a la terminal de nuestra Raspberry.

```
$ sudo apt update
$ sudo apt purge realvnc-vnc-server
$ sudo apt purge realvnc-vnc-viewer
$ sudo apt install realvnc-vnc-server
$ sudo apt install realvnc-vnc-viewer
```

Después de ingresar estos comandos, tenemos que realizar de nuevo los pasos expuestos en los apartados anteriores (capítulo 3.3, 3.1.1 y 3.2.2).

Si después de implementar esto, aún no podemos conectarnos de forma remota con nuestra Raspberry, se prosigue con la regeneración de las claves SSH del sistema.

Primero tenemos que eliminar las claves antiguas, para ello, vamos a ingresar el siguiente comando en la terminal.

```
$ sudo rm /etc/ssh/ssh_host_*
```

Y para generar las nuevas claves, se tiene que ingresar el siguiente comando.

```
$ sudo dpkg-reconfigure openssh-server
```

3.3.4 Instalación de Visual Studio Code

Visual Studio Code, también conocido como VS Code, es un editor de código fuente desarrollado por Microsoft, disponible gratuitamente para Windows, macOS y Linux. Lo que lo hace destacar es su versatilidad a través de extensiones que permiten personalizar el entorno de desarrollo y añadir nuevas características.

Visual Studio Code, además de ofrecer herramientas que agilizan y facilitan el desarrollo de software, brinda soporte multiplataforma, incluyendo compatibilidad con el lenguaje de programación de Python. Este lenguaje es ampliamente utilizado para desarrollar programas destinados a ejecutarse en los equipos Raspberry Pi.

Se puede acceder a las descargas del entorno de desarrollo VS Code, así como a su documentación, desde el siguiente enlace.

https://code.visualstudio.com/.

Instalar el entorno de **Visual Studio Code** en la Raspberry es muy sencillo, únicamente tenemos que abrir una terminal y escribir los siguientes dos comandos.

```
$ sudo apt update
$ sudo apt install code
```

Para cerciorarnos de que efectivamente, el programa se haya instalado correctamente, podemos desplegar el menú principal de la Raspberry y seleccionar la opción *Programming*, allí debería aparecer *Visual Studio Code*, como se puede observar en la Figura 3.24.

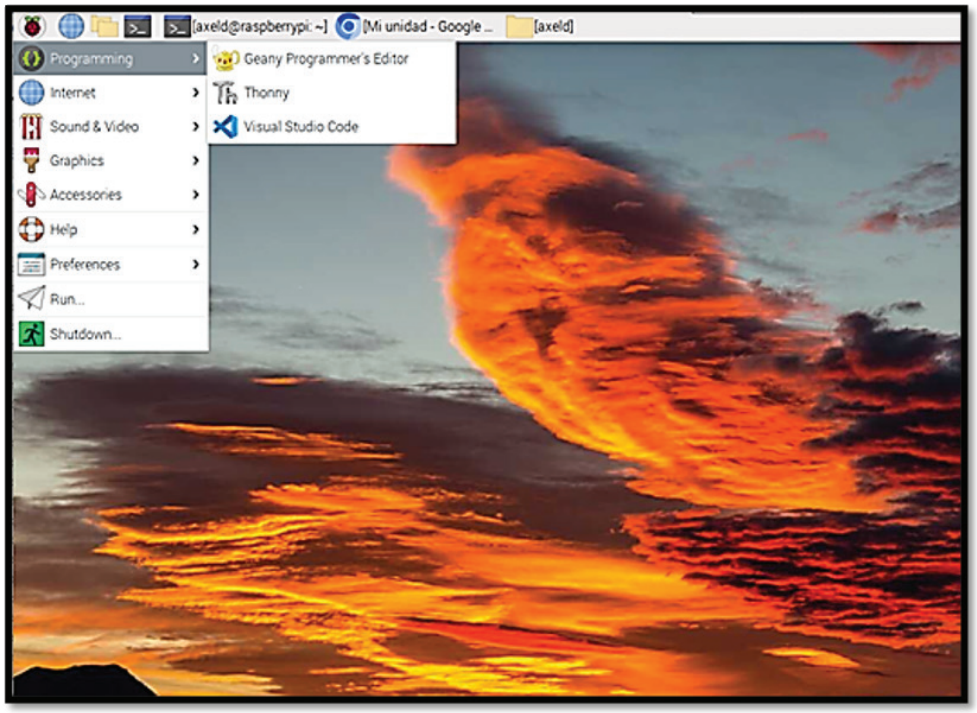

Figura 3.24. Comprobar la instalación del entorno VS Code

3.3.5 Instalación de librería Pyrealsense

Para poder conectar la *Cámara Realsense*, requerida por su función de visión de profundidad y su red neuronal convolucional, se requiere realizar los siguientes pasos, los cuales darán paso a la creación de la librería *PyRealSense.*

ⓘ NOTA

Las letras que se muestran al principio de los comandos en los siguientes pasos, son incisos aquí en el texto, por lo que estos no forman parte de los comandos.

Inicializar la actualización e instalación de las dependencias y herramientas necesarias, ingresando el siguiente comando en la terminal.

```
$ sudo apt-get install automake libtool vim cmake libusb-1.0-0-dev
libx11-dev xorg-dev libglu1-mesa-dev
```

ⓘ NOTA

Aquí también se pregunta si se desea continuar con la instalación, por lo que aceptamos, escribiendo la letra 'y', y dando un enter.

Aumentar el tamaño del intercambio a 2 GB realizando los siguientes pasos:

```
$ sudo nano /etc/dphys-swapfile
```

Buscar la línea `CONF_SWAPZIE=100 y cambiar el 100 por 2048`. Guardar el archivo con la siguiente combinación de comandos (Ctrl + O, y después un enter) y regresamos a la terminal con esta otra combinación de teclas (Ctrl + X). He ingresar el siguiente comando para ejecutar los cambios:

```
$ sudo /etc/init.d/dphys-swapfile restart swapon -s
```

Crear una nueva regla de udev, para ellos ingresamos los siguientes comandos en la terminal, uno por uno.

```
$ cd ~
$ git clone https://github.com/IntelRealSense/librealsense.git
$ cd librealsense
$ sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
```

Para aplicar la nueva regla, se requiere ejecutar los siguientes comandos, lo cual debe realizarse desde el usuario root.

```
$ sudo su
$ udevadm control --reload-rules && udevadm trigger
$ exit
```

Modificar la ruta de bashrc con nano (para que el programa sea capaz de encontrar las librerías) agregando las siguientes líneas al archivo .nano .bashrc **(esta línea crea en archivo llamado .bashrc y dentro de este archivo se pega la línea de abajo y se guarda).**

```
$ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

Aplicar cambios con el siguiente comando.

```
$ source ~/.bashrc
```

Es necesario realizar la instalación de la herramienta SDK de C++ para que los programas se puedan compilar de una manera correcta, para ello se siguen los siguientes pasos:

Clonar el repositorio de pico-SDK y los ejemplos para verificar el correcto funcionamiento, para ello vamos a ingresar los siguientes comandos en la terminal.

```
$ git clone https://github.com/raspberrypi/pico-sdk.git --branch master
$ cd pico-sdk
$ git submodule update --init
$ cd ..
$ git clone https://github.com/raspberrypi/pico-examples.git --branch
master
%Por último se corren estas líneas
$ sudo apt update
$ sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi
build-essential
```

Verificar que se hayan clonado las carpetas de los repositorios *pico-sdk* y *pico-examples* en el mismo directorio, como se muestra en la Figura 3.25.

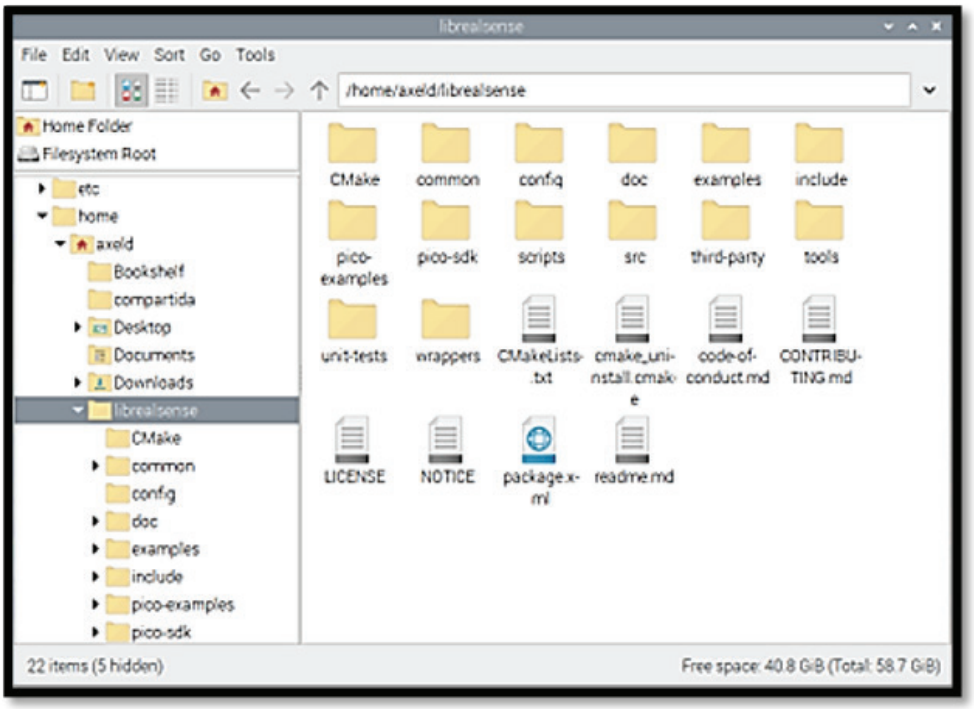

Figura 3.25. Directorios *pico* correctamente clonados

Configurar el PICO_SDK_PATH, ingresando en la terminal el siguiente comando:

```
$ export PICO_SDK_PATH=../../pico-sdk
```

Reiniciar la Raspberry.

En los siguientes pasos se utiliza la función make -j1. El número a la derecha de la letra j hace referencia al número de procesadores que se utilizarán al compilar, por lo que se decidió modificar a -j4 para aprovechar todos los procesadores de la Raspberry pi 4.

Instalar protobuf, formato de serialización de datos estructurados, desarrollado por Google, tiempo aproximado de 15 a 20 min. Reiniciar este la Raspberry después de correr estas líneas, Figura 3.26.

```
$ cd ~
$ git clone --depth=1 -b v3.10.0 https://github.com/google/protobuf.git
$ cd protobuf
$./autogen.sh
$./configure
```

```
$ make -j4
$ sudo make install
$ cd python
$ export LD_LIBRARY_PATH=../src/.libs
$ python3 setup.py build --cpp_implementation
$ python3 setup.py test --cpp_implementation
$ sudo python3 setup.py install --cpp_implementation
$ export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=cpp
$ export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION=3
$ sudo ldconfig
$ protoc –version
```

Instalar la biblioteca de paralelismo libtbb-dev para C++

```
$ cd ~
$ wget https://github.com/PINTO0309/TBBonARMv7/raw/master/libtbb-
dev_2018U2_armhf.deb
$ sudo dpkg -i ~/libtbb-dev_2018U2_armhf.deb
$ sudo ldconfig
$ rm libtbb-dev_2018U2_armhf.deb
```

Configuración del SDK de C++, se debe tener seleccionada la configuración 1 como en la Figura de abajo.

```
$ sudo apt-get install llvm
$ sudo apt-get install clang
%En la siguiente línea se abre el menú
$ sudo update-alternatives --config c++
```

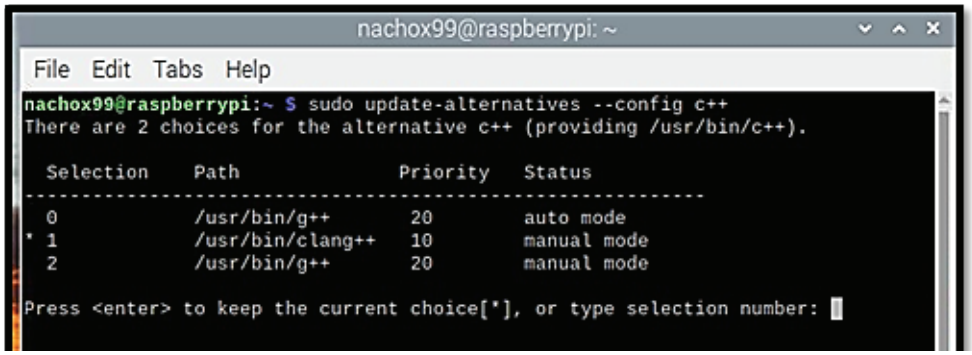

Figura 3.26. Para la configuración adecuada del SDK de C++, es importante asegurarse de que la 'Opción Uno' esté activada/marcada

```
%Reiniciar
$ sudo reboot
Instalar la librería Real Sense SDK librealsense, este proceso puede
durar hasta 3 horas.
$ wget https://github.com/IntelRealSense/librealsense/archive/master.
zip
$ unzip ./master.zip -d .
$ cd ./librealsense-master
$ echo Install udev-rules
$ sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
$ sudo cp config/99-realsense-d4xx-mipi-dfu.rules /etc/udev/rules.d/
$ sudo udevadm control --reload-rules && sudo udevadm trigger
$ mkdir build && cd build
$ sudo cmake ../ -DBUILD_EXAMPLES=true -DCMAKE_BUILD_TYPE=Release
-DFORCE_LIBUVC=true -DOTHER_LIBS="-latomic"
$ sudo make -j4
$ sudo make install
% Después de correr las líneas se corren en la misma terminal las
líneas de abajo
$ sudo cmake .. -DBUILD_PYTHON_BINDINGS=bool:true -DPYTHON_
EXECUTABLE=$(which python3)
$ sudo make -j4
$ sudo make install
% Reinciar la raspberry
```

Ejecutar *rs-enumerate-devices* desde la terminal para verificar la instalación.

Una vez terminado los pasos anteriores, es recomendable probar el comando a la cámara utilizando el comando *realsense-viewer.*

3.3.6 Instalación de librería OpenCV

Abrir una terminal para poder ingresar los siguientes comandos uno por uno:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install python3-opencv
```

Es importante mencionar que, al ingresar el segundo y el tercer comando, en algún punto de su ejecución, el instalador pregunta si se desea continuar con la instalación, por lo que debemos ingresar la letra ‘y’ para luego darle un enter, Figura 3.27.

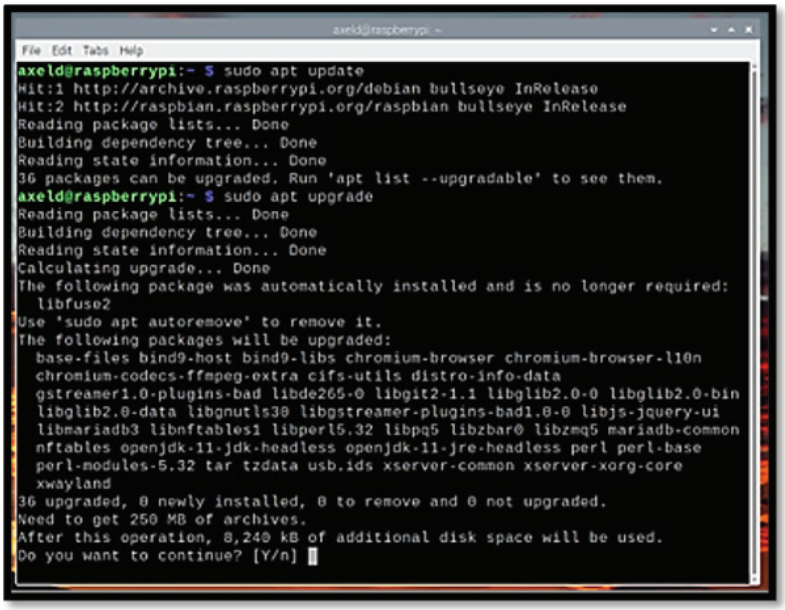

Figura 3.27. Permitir continuar con la instalación

También cabe indicar que, durante la ejecución del segundo comando, puede salir una pantalla como la siguiente, en la cual debemos presionar la letra ‘q’ para salir de ella y que el sistema continúe con la actualización de este, Figura 3.28.

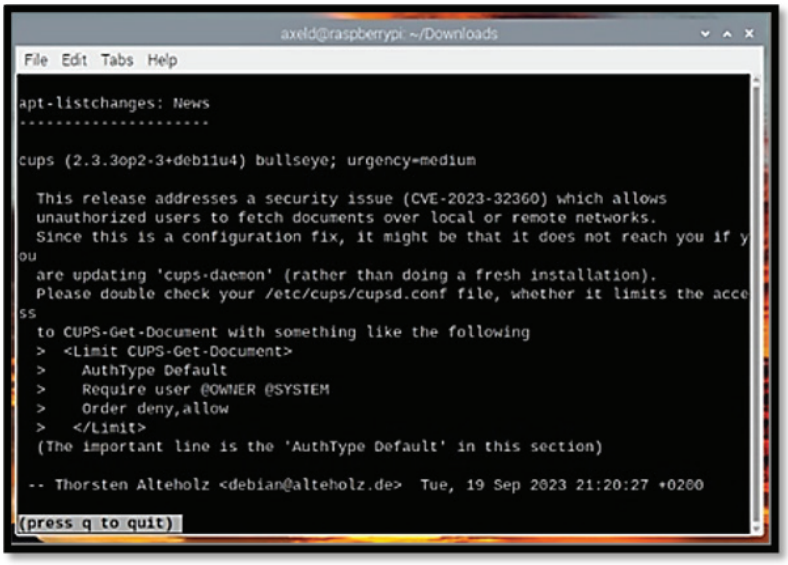

Figura 3.28. Salir de la ventana emergente en la instalación

3.4 IMPLEMENTACIÓN DE CÓDIGO

Vamos a transferir la carpeta comprimida **RealsenseCode** que se incluye en los recursos de este libro a la Raspberry, y la colocaremos en la carpeta de **Descargas**. Daremos clic derecho sobre el comprimido, para seleccionar *Extract Here*. Se creará la carpeta *RealsenseCode*. En esa carpeta se encuentran los códigos requeridos, Figura 3.29.

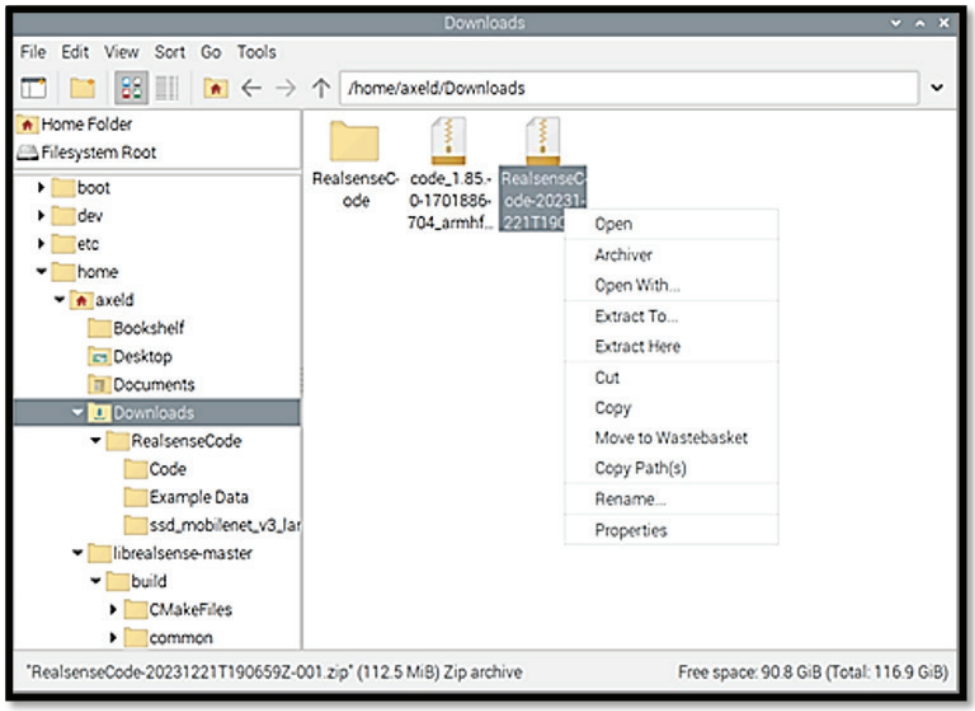

Figura 3.29. Extraer zip comprimido

A continuación, debemos abrir la carpeta de **RealsenseCode** con el entorno de Visual Studio Code, el cual preguntara si se confía en los autores, damos clic en sí, Figura 3.30.

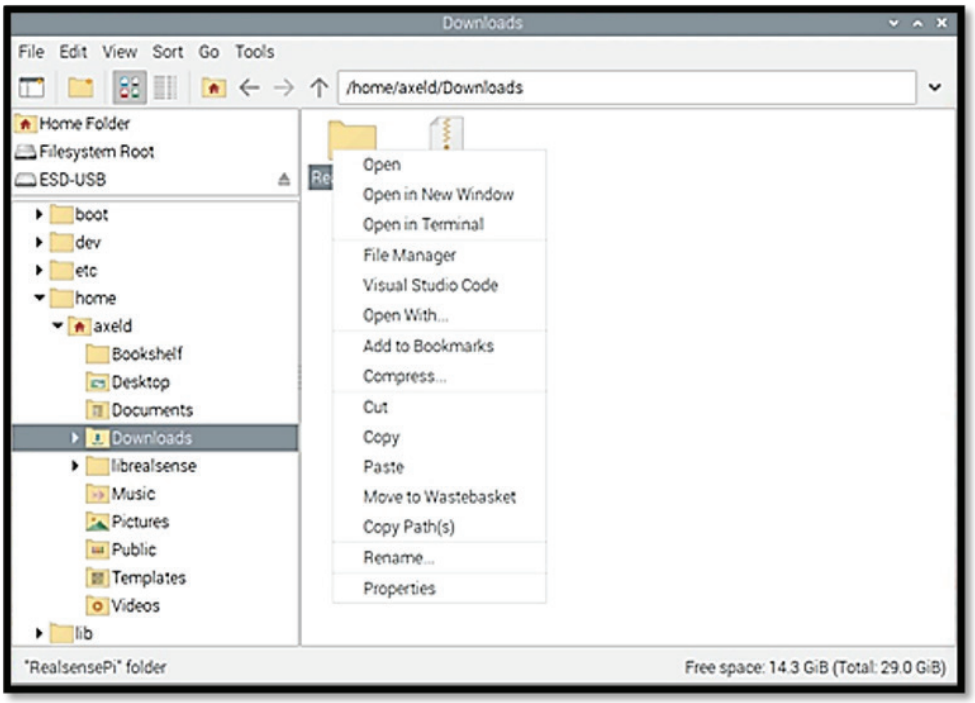

Figura 3.30. Abrir la carpeta como un proyecto de Visual Studio Code

Dentro del entorno, abriremos el código llamado **Video_analizer.py**, esto produce que Visual Studio nos sugiera, de manera automática, instalar la extensión de Python, la cual es requerida para el presente proyecto, como se muestra en la Figura 3.31.

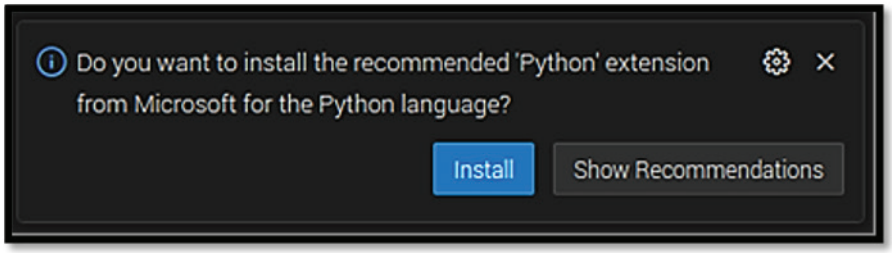

Figura 3.31. Instalación de la extensión de python en visual studio code

Previo a la ejecución del código, es necesario modificar el contenido de la variable *user* con el nombre del usuario que se le dio a la Raspberry cuando se configuró el sistema operativo. Posterior a esto, ejecutamos el algoritmo donde dice *Run Python File,* Figura 3.32.

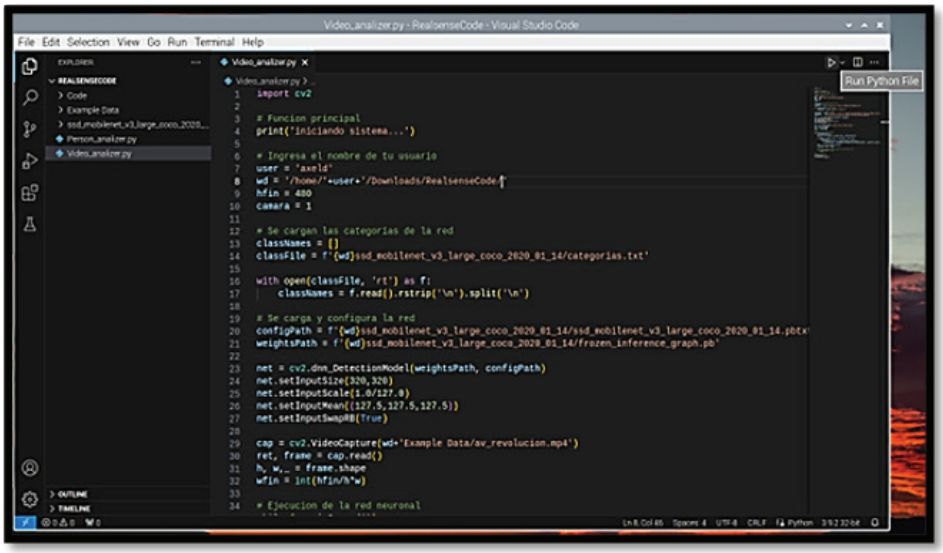

Figura 3.32. Instalación de la extensión de python en visual studio code

Debe aparecer una ventana como la siguiente, Figura 3.33.

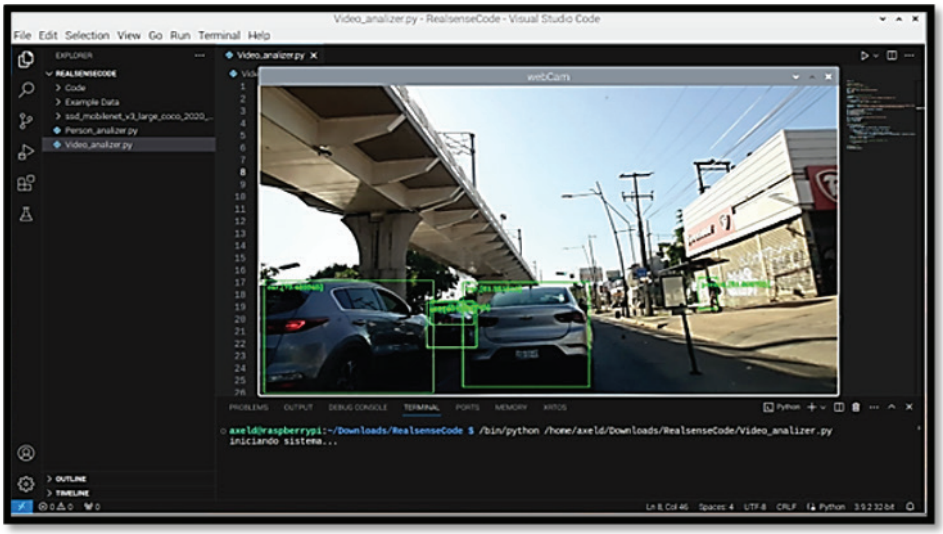

Figura 3.33. Código Video_analizer ejecutándose correctamente

Para detener la ejecución del algoritmo presionamos directamente la tecla escape, o podemos regresar a la terminal de Visual Studio y presionar la combinación de teclas CTRL + C, o hacer clic sobre el botón *Kill terminal*, el cual tiene forma de bote de basura.

3.5 DESCRIPCIÓN DEL PRIMER CÓDIGO A IMPLEMENTAR

El primer código con el que vamos a trabajar es el llamado **Video_analizer.py**. Este código tiene por objetivo demostrar el correcto funcionamiento de la red neuronal que nos permitirá detectar obstáculos, calcular a qué distancia se encuentran, así como obtener el ángulo al que se encuentran, sin utilizar aún la cámara de profundidad Intel Realsense, puesto que el algoritmo se ejecuta sobre un vídeo en demanda. Cada parte de este código se describe a continuación.

Configuración inicial. Se importa la biblioteca cv2 (es decir, la biblioteca necesaria de OpenCV) y se imprime un mensaje de bienvenida. Establece el nombre de usuario (user), el directorio de trabajo (wd), la altura final a la que se redimensionarán los fotogramas para su visualización, que, en este caso, se ha establecido en 480 píxeles (hfin) y la cantidad de cámaras a utilizar (cámara), en este caso una nada más.

```
import cv2

print('iniciando sistema...')
user = 'axeld'
wd = '/home/'+user+'/Downloads/RealsenseCode/'
hfin = 480
camara = 1
```

Carga de categorías de la red. Lee las categorías de los objetos detectables para la red neuronal desde un archivo de texto llamado “categorias.txt” y las almacena en la lista classNames.

```
classNames = []
classFile = f'{wd}ssd_mobilenet_v3_large_coco_2020_01_14/categorias.
txt'
with open(classFile, 'rt') as f:
classNames = f.read().rstrip('\n').split('\n')
```

Configuración de la red neuronal. Se establecen los archivos de configuración y pesos de la red, además se configuran parámetros como el tamaño de entrada,

la escala, la media y el intercambio de canales (RGB[6] a BGR, estos conceptos se detallan más adelante).

```
configPath = f'{wd}ssd_mobilenet_v3_large_coco_2020_01_14/ssd_mobilenet_
v3_large_coco_2020_01_14.pbtxt'
weightsPath = f'{wd}ssd_mobilenet_v3_large_coco_2020_01_14/frozen_
inference_graph.pb'
net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0 / 127.0)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

- **configPath:** especifica la ruta al archivo de configuración del modelo, que en este caso es "ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt". Este archivo contiene la configuración de la arquitectura de la red neuronal, como la disposición de las capas y los hiperparámetros.
- **weightsPath:** especifica la ruta al archivo de los pesos pre entrenados del modelo, llamado "frozen_inference_graph.pb". Estos pesos son los resultados del entrenamiento de la red neuronal en un conjunto de datos específico.
- **cv2.dnn_DetectionModel:** crea una instancia de la clase *cv2.dnn_DetectionModel* utilizando los archivos de configuración y pesos proporcionados. Esta clase en OpenCV se utiliza para cargar modelos de detección de objetos preentrenados.
- **net.setInputSize(320, 320):** establece el tamaño de entrada de la red neuronal a 320x320 píxeles. Esto significa que los fotogramas de entrada serán redimensionados a este tamaño antes de pasar por la red.
- **net.setInputScale(1.0 / 127.0):** normaliza los valores de los píxeles en los fotogramas dividiéndolos por 127.0.

La normalización de datos es una práctica común en el aprendizaje profundo, incluida la detección de objetos con redes neuronales convolucionales (CNN). En este contexto, la línea de código anterior se encarga de normalizar los valores de los píxeles de la imagen antes de pasarla a través de la red neuronal. Aquí hay una explicación más detallada de lo que implica normalizar los valores:

Rango de valores de píxeles: ¿Qué es la normalización de datos? Las imágenes suelen tener píxeles representados como valores enteros en el rango de 0 a 255 para cada canal de color (rojo, verde, azul en el caso de una imagen en color) puesto que

se trabaja con el sistema de colores RGB. Normalizar los valores implica escalarlos para que estén en un rango más pequeño y centrado alrededor de cero.

Motivación para la normalización: las redes neuronales a menudo se benefician de datos de entrada que tienen una media cercana a cero y una varianza moderada, por lo tanto, la normalización puede ayudar a que la red neuronal converja más rápido durante el entrenamiento y puede mejorar la estabilidad numérica en los cálculos internos de la red.

- **net.setInputScale(1.0 / 127.0)**: net.setInputScale establece la escala de entrada para la red neuronal. En este caso, la escala es 1.0 / 127.0.

  La escala 1.0 / 127.0 realiza una normalización dividiendo cada valor de píxel por 127.0. Esto coloca los valores de píxeles en un rango aproximado de -1 a 1.

  ¿Por qué 127.0? La elección de 127.0 se debe a que es la mitad de 255 (la intensidad máxima de un canal en una imagen de 8 bits por canal). Dividir por este valor lleva los píxeles al rango [-1, 1].

- **net.setInputMean((127.5, 127.5, 127.5)):** especifica los valores medios de los canales de color R, G y B en el rango de 0 a 255. Estos valores se restan de cada canal en cada píxel de la imagen antes de pasarla a la red neuronal. Esto también es parte del proceso de normalización.

- **net.setInputSwapRB(True):** indica que se deben intercambiar los canales de color R y B en los fotogramas antes de enviarlos a la red. Este intercambio puede ser necesario dependiendo de cómo se haya entrenado la red (en OpenCV, las imágenes se leen en formato BGR por defecto).

Configuración del vídeo. Abre un archivo de vídeo (av_revolucion.mp4) y obtiene el primer fotograma para determinar sus dimensiones. Calcula la anchura final (wfin) para mantener la proporción de aspecto original.

```
cap = cv2.VideoCapture(wd+'Example Data/av_revolucion.mp4')
ret, frame = cap.read()
h, w, _ = frame.shape
wfin = int(hfin/h * w)
```

- **cv2.VideoCapture(wd+'Example Data/av_revolucion.mp4'):** cv2.VideoCapture es una función de OpenCV que crea un objeto para capturar vídeo desde una fuente, que puede ser un archivo de vídeo o un dispositivo de captura en tiempo real como una cámara. En este caso, se está abriendo un archivo de vídeo llamado 'av_revolucion.mp4'. La ruta completa del archivo

se forma concatenando wd (el directorio principal) con el resto de la ruta del archivo.

- **ret, frame = cap.read():** cap.read() lee un fotograma del vídeo. El valor de retorno ret indica si la operación de lectura fue exitosa (True si se leyó correctamente, False si no). frame contiene el fotograma leído.

- **h, w, _ = frame.shape:** frame.shape devuelve las dimensiones del fotograma en forma de una tupla (altura, anchura, canales). Se desempaqueta esta tupla en las variables h, w, y _ (el cual representa el número de canales, pero no se utiliza en este caso ya que se asume que es un fotograma en color).

- **wfin = int(hfin/h*w):** hfin es la altura final deseada para el vídeo. La expresión int(hfin/h*w) calcula la nueva anchura (wfin) proporcional al cambio en la altura. Esto se hace para mantener la relación de aspecto original del vídeo. Se convierte a int para asegurarse de que sea un número entero.

Ejecución de la red neuronal. Realiza la detección de objetos en cada fotograma del vídeo. Dibuja un rectángulo alrededor de los objetos detectados y muestra el nombre de la clase y la confianza en el fotograma. La ejecución se detiene si se presiona la tecla "Esc".

```
while (cap.isOpened()):
    ret, frame = cap.read()
    classIds, confs, bbox = net.detect(frame, confThreshold=0.5)
    for element, confidence, box in zip(classIds, confs, bbox):
        if element < 10:
            cv2.rectangle(frame, box, color=(0, 255, 0), thickness=2)
            cv2.putText(frame, f'{classNames[int(element-
1)]},{confidence*100}', (box[0]+10, box[1]+30), cv2.FONT_HERSHEY_
COMPLEX, 0.7, (0, 255, 0), 2)
    frame = cv2.resize(frame, (wfin, hfin))
    cv2.imshow('webCam', frame)
    if (cv2.waitKey(1) == 27):
        break
```

- **while (cap.isOpened()):** inicia un bucle que se ejecutará mientras la captura de vídeo esté abierta. La condición cap.isOpened() verifica si la captura de vídeo se abrió correctamente.

- **ret, frame = cap.read():** lee el siguiente fotograma del vídeo y lo almacena en la variable frame. La variable ret indica si la operación de lectura fue exitosa.

- **classIds, confs, bbox = net.detect(frame, confThreshold=0.5):** utiliza la red neuronal (net) para realizar la detección de objetos en el fotograma actual (frame).
  - classIds contiene las clases de los objetos detectados.
  - confs contiene las confianzas asociadas a cada detección.
  - bbox contiene las coordenadas de las cajas delimitadoras de los objetos.
- **for element, confidence, box in zip(classIds, confs, bbox):** itera sobre las detecciones y realiza las siguientes acciones para cada objeto. Si el índice de la clase (element) es menor que 10 (esto es específico para este código), se considera válido.

Dibuja un rectángulo alrededor del objeto en el fotograma usando cv2.rectangle. Agrega un texto que muestra la clase y la confianza sobre el objeto utilizando cv2.putText.

- **frame = cv2.resize(frame, (wfin, hfin)):** redimensiona el fotograma (frame) al tamaño deseado especificado por wfin y hfin. Esto asegura que la **visualización se adapte a la ventana correctamente.**
- **cv2.imshow('webCam', frame):** muestra el fotograma resultante en una ventana llamada 'webCam' mediante cv2.imshow.
- **if (cv2.waitKey(1) == 27):** espera la pulsación de una tecla. Si la tecla presionada es la tecla "esc" (cuyo código es 27), se rompe el bucle y la ejecución termina.

Cierra de la ventana y liberación de recursos. Se libera la captura de vídeo y se cierran todas las ventanas al finalizar la ejecución.

```
cap.release()
cv2.destroyAllWindows()
```

3.6 CORRER CÓDIGOS EN PYTHON

Si por alguna razón, la opción *Run file* de Visual Studio Code no funcionara, es decir, no realizara ninguna acción, existe otro método para poder correr los códigos fuera de este entorno.

Lo que debemos hacer es abrir una terminal dentro de la carpeta de cualquier método (RealsenseCode o RealsenseComp), como ya explicó anteriormente (Tools / Open Current Folder in Terminal) y escribir el siguiente comando:

```
$ python3 nombre_del_archivo
```

Por ejemplo, si se quiere ejecutar el código **Video_analizer.py** se tendría que escribir el comando mostrado en la terminal de la siguiente imagen, en la cual también se muestra el resultado, Figura 3.34.

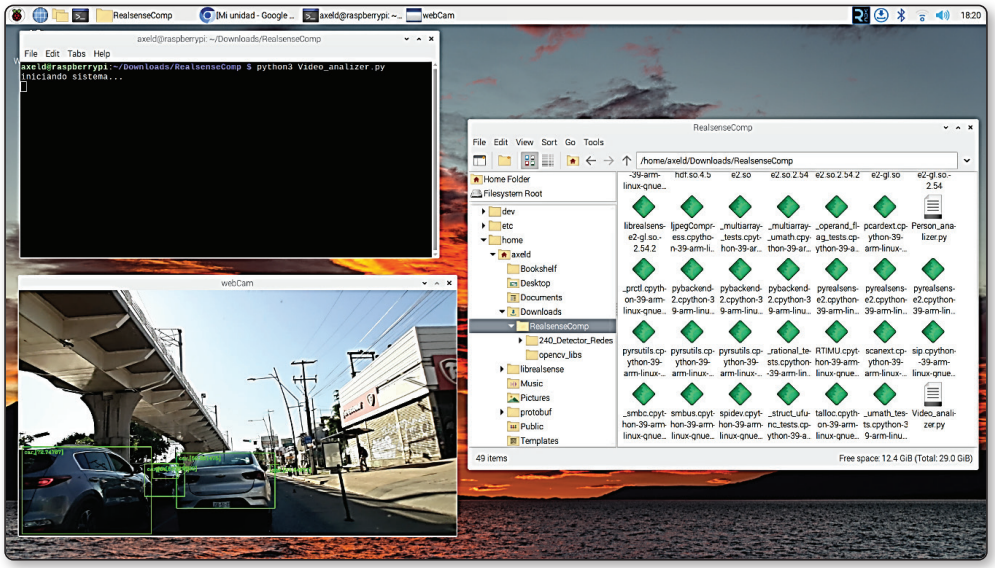

Figura 3.34. Método alternativo para correr código en python

Para detener la ejecución del código, simplemente presionamos la tecla escape o podemos regresar a la terminal y presionar la combinación de teclas CTRL + C.