Sistema de Seguridad

con

Raspberry Pi & Sensor Pir

DESCRIPCIÓN BREVE:
EN ESTE PROYECTO SE IMPLEMENTARÁ UN SISTEMA DE SEGURIDAD CON SENSOR DE MOVIMIENTO, QUE CUANDO DETECTE MOVIMIENTO CAPTURE IMÁGENES Y VIDEOS Y LOS ENVÍE A TELEGRAM Y A GOOGLE DRIVE.

Autor: Xavier Espinosa

Sumario

MEMORIA

En este proyecto hemos realizado un sistema de seguridad, basado en la configuración de un RaspberryPi 3b+ con un sensor Pir HC-SR501, una Pi Camera y un buffer activo para la alarma

Cuando detecte movimiento el sensor Pir, se activara la Pi Camera y realizara una ráfaga de imágenes en un periodo de 1 mín, cada 2 segundos y las guardara en la memoria de la Raspberry , seguidamente con todas las imágenes capturadas, creara un video en formato mp4, que lo enviara tanto a un directorio de Google Drive , como a un bot de Telegram, previamente creado .

Para finalizar creamos un auto-ejecutable para que se ejecute al arrancar el sistema operativo Raspbian (Distribución Debian de Linux para Raspberry) de la Raspberry3b+ .

COMPONENTES

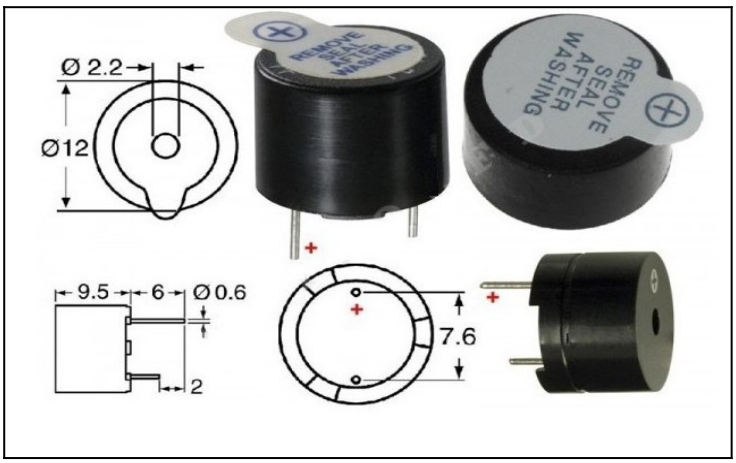
Para este proyecto hemos utilizado los siguientes componentes:

BUZZER ACTIVO

El Buzzer activo o zumbador electrónico es alimentado por CC y es muy utilizado para computadoras, impresoras,fotocopiadoras, alarmas, juguetes electrónicos, dispositivos electrónicos,automotores, teléfonos, alarmas y otros productos electrónicos para dispositivos de voz.

El buzzer activo tiene tiene una fuente oscilante integrada, por lo que va a generar un sonido cuando se electrifica.

El precio orientativo de este componente es de unos 8€.



SENSOR HC-SR501 PIR

El sensor Pir HC-SR501 es un detector de movimiento, que emplea IR, es decir, infrarrojos. En base a esta radiación funcionará detectando movimientos o proximidad. Todo gracias a los elementos de los que está integrado para lograr captar la presencia de personas o movimiento, también el de animales y otros objetos.

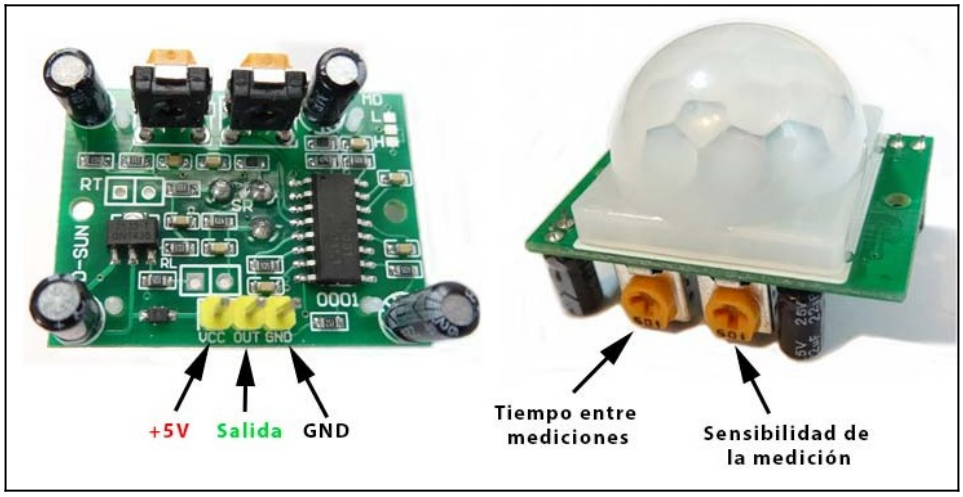
Todo basándose en la radiación baja IR que emiten los cuerpos de objetos, personas o mascotas. Y que serán captadas por este tipo de sensores con una gran precisión.

Este tipo de sensor se puede configurar para ajustar su sensibilidad, para que pueda tener mayor o menor alcance. Por ejemplo, algunos sensores pueden tener un alcance de detección de movimientos o presencia desde los 3 metros hasta los 7 metros o más, y con rangos de 90-110º. Eso permite detectar movimiento en un buen rango, permitiendo su instalación en cualquier muro, por ejemplo.

Por otro lado, el sensor PIR estará cubierto por un protector en forma de cúpula. Esto no es más que la llamada lente de Fresnel. Es decir, una lente nombrada así por el nombre del físico francés que la inventó, Augustin-Jean Fresnel, y gracias a la que se puede conseguir esa

gran apertura focal sin necesidad de otras lentes de mayor peso y volumen que se necesitaría usando una lente convencional.

El precio orientativo es de 2,70€.



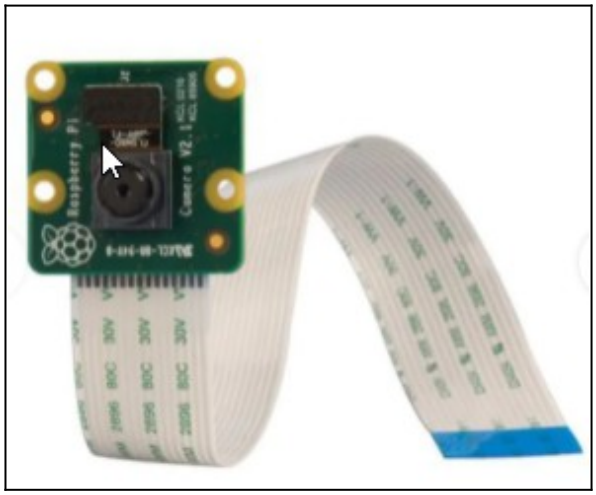
Pi CAMERA

El módulo de la Pi Camera es el dispositivo que conectado a la RaspberryPi3b+ nos realiza las fotografías e video y nos reproduce imágenes nítidas de alta resolución a través de una cámara de 5 mega-píxeles.

Gracias a los filtros infrarrojos (filtro IR) también es posible sacar fotografías excelentes con esta cámara de vídeo y fotográfica en la luz brillante.

La cámara trasmite los datos a la Raspberry Pi con un cable de datos.

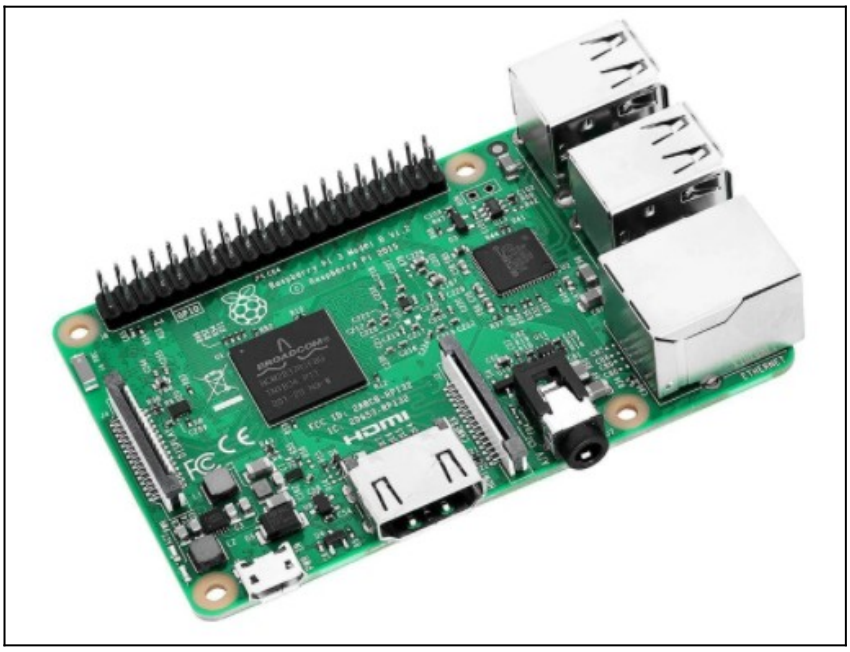
El precio orientativo de la Pi Camera es de 8€.



RASPBERRY PI3B+

El modelo que se ha utilizado en este proyecto es la Raspberry Pi 3 B+ apareció en marzo del 2018 para actualizar el modelo anterior la Raspberry Pi 3 Model B y entre sus mejoras cuenta con un nuevo procesador y mejor conectividad,así que pasa de tener 1.2Ghz a tener 1.4Ghz y en cuanto a la conectividad inalámbrica ahora incorpora doble banda a 2,4GHz y 5GHz, y su nuevo puerto Ethernet se triplica, pasa de 100 Mbits/s en el modelo anterior a 300 Mbits/s en el nuevo modelo, también cuenta con Bluetooth 4.2 (Low Energy).

El precio orientativo es de 40€.



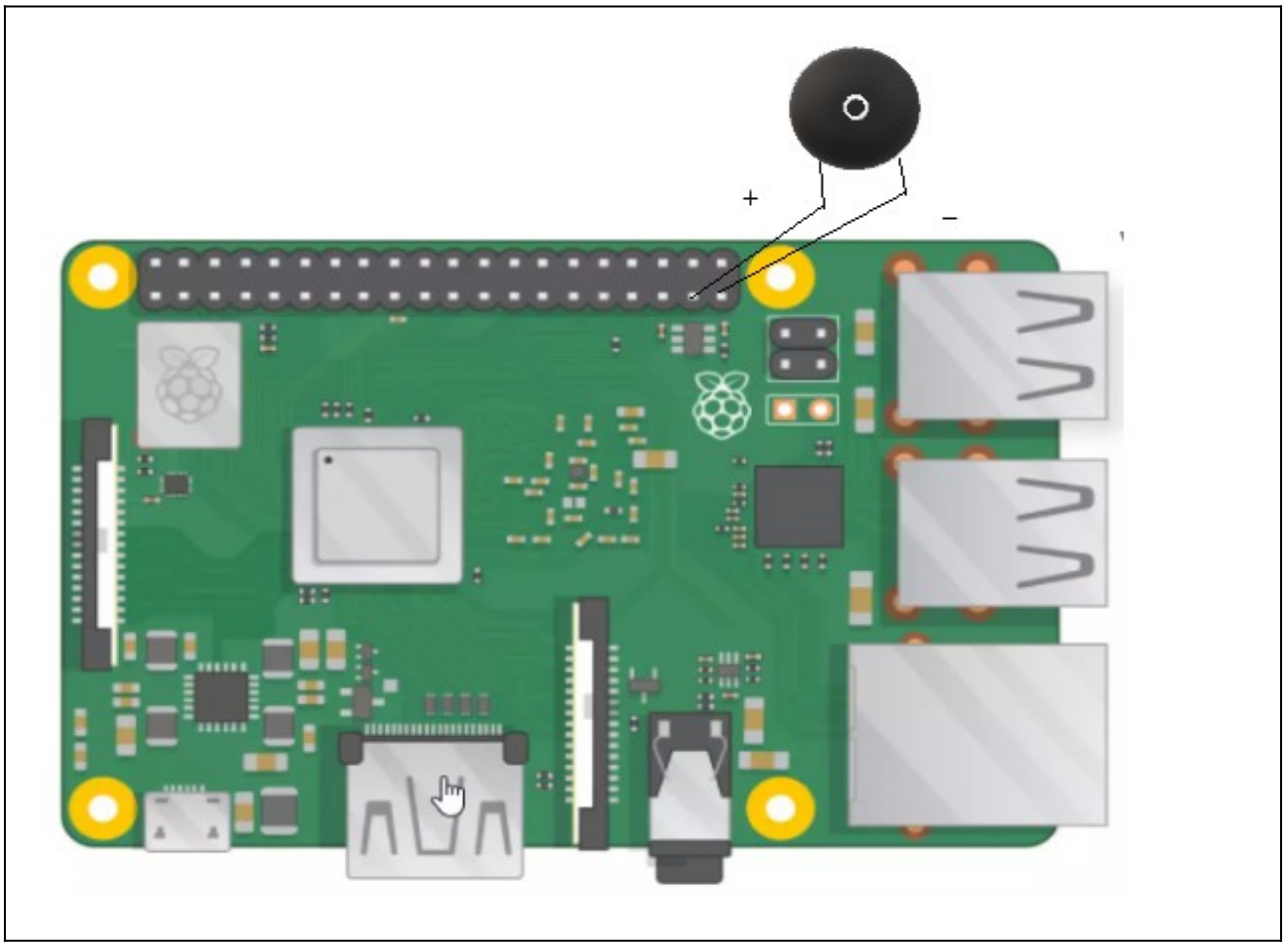
CABLES HEMBRA-HEMBRA PARA CONEXIONES ENTRE GPIOS DE LA RASPBERRY Y SENSOR PIR E BUZZER.

El precio orientativo es entre 7€ y 8€

INSTALACIÓN Y CONFIGURACIÓN DE BUZZER

Para instalar el buzzer a la raspberry Pi3b+ hemos seguido el siguiente esquema basándonos en el manual del Kit de Componentes Elegoo Uno

Se ha configurado el Buzzer en el pin GPIO26 de la Raspberry Pi3b+ que va a la patilla positiva del buzzer y la patilla negativa del buzzer va a un pin GND de la raspberry Pi3b+.



He pasado el código en C++ de Arduino a Python para que funcione en la Raspberry

El código de python ha sido el siguiente:

```
import RPi.GPIO as GPIO
import time

buzzer = 26

def delay(temps):
        time.sleep(temps/1000)


def peripheral_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(buzzer,GPIO.OUT)


def peripheral_loop():
    for step in range(80):
        GPIO.output(buzzer, 1)
        delay(1)
        GPIO.output(buzzer, 0)
        delay(1)

    for step in range(100):
        GPIO.output(buzzer,1)
        delay(2)
        GPIO.output(buzzer,0)
        delay(2)


peripheral_setup()
while 1:
    try:
        peripheral_loop()
    except KeyboardInterrupt:
        GPIO.output(buzzer, 0)
        break
```

En este código se han importado las librerias de Rpi.GPIO y time.

Se ha creado una variable llamada buzzer que hace referencia al GPIO26

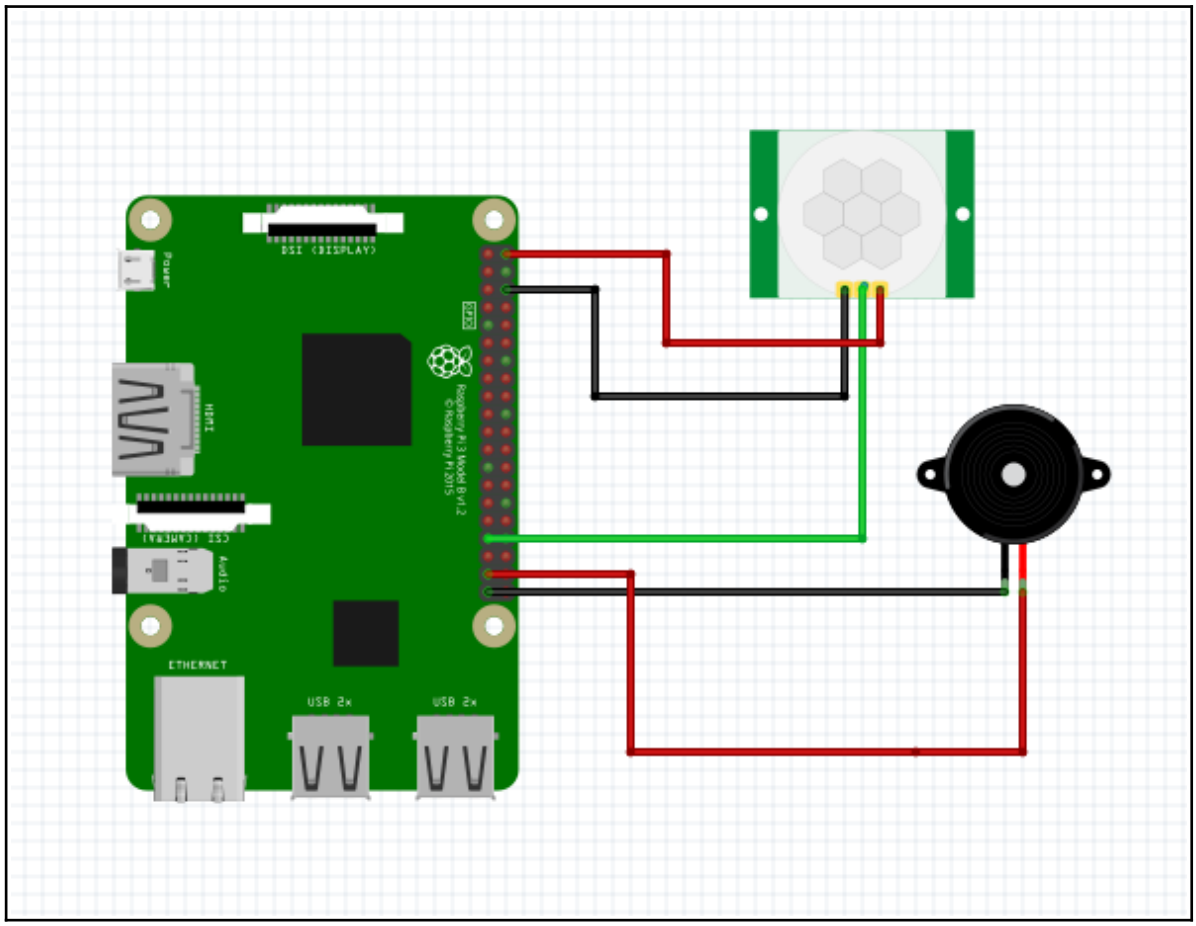
Se ha creado una función llamada delay para que nos transforme el tiempo de ms a s

En la función de la configuración_setup se han configurado los GPIO en modo BCM "BROADCOM CHANNEL"

En la función de loop se ha puesto dos frecuencias al buzzer de 80Hz y 100 Hz, se enciende y se apaga , con un tiempo de 1s en el primer for y en el segundo for un tiempo de 2s.

INSTALACIÓN Y CONFIGURACIÓN DE BUZZER CON PIR

Se ha configurado el Sensor Pir en el GPIO13 y el Buzzer en el GPIO 26 como se ve en el esquema siguiente:



Se ha soldado los cables que van al buzzer, se han conectado los cables del Sensor Pir, respetando los pines de conexión VCC, OUT y GND ,se han regulado los potenciómetros de sensibilidad e tiempo para la detención de movimiento del sensor

Se ha configurado con el siguiente código de Python:

```
import RPi.GPIO as GPIO
import time

zumbador = 26
pir = 13

def peripheral_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pir,GPIO.IN)
    GPIO.setwarnings(False)
    GPIO.setup(zumbador,GPIO.OUT)


def peripheral_loop():

    time.sleep(2) # estabilizar el sensor
    while 1:
        if GPIO.input(pir):
            GPIO.output(zumbador,1)
            time.sleep(1) #El zumbador de enciende durante 1s
            GPIO.output(zumbador,0)
            print("Movimiento detectado")
            time.sleep(5) ##para evitar la detección múltiple
        time.sleep(0.5) #retardo de bucle, debe ser menor que el retardo de
detección, menos de 1s


peripheral_setup()
while 1:
    try:
        peripheral_loop()
    except KeyboardInterrupt:
        GPIO.cleanup()
        break
```

Se ha configurado el GPIO del sensor Pir como una entrada (IN) y el GPIO del buzzer como una salida (OUT).

Se ha creado un bucle (loop) , para que suene el buzzer durante 1s, al final con un Keyboard Interrupt hacemos una excepción y a través del teclado con *Ctrl+C* salimos del programa y hacemos un reset de los pines de la raspberry con un GPIO.cleanup().

La función GPIO.cleanup () sirve para limpiar todos los puertos que se ha utilizado. Solo afecta a cualquier puerto que haya configurado en el programa actual. Restablece cualquier puerto que haya utilizado en este programa de nuevo al modo de entrada. Esto evita el daño de, por ejemplo, una situación en la que tiene un puerto configurado en ALTO como salida y accidentalmente lo conecta a GND (BAJO), lo que podría provocar un cortocircuito en el puerto y posiblemente freírlo. Las entradas pueden manejar 0V (BAJO) o 3.3V (ALTO), por lo que es más seguro dejar los puertos como entradas.

CONFIGURACIÓN DE WEBCAM CON LA RASPBERRY

Se conecta una webcam modelo Logitech de 5 Megapixels para realizar unas pruebas directamente a un puerto usb de la Raspberry.

Instalamos una aplicación en la distribución de Raspbian , llamada fswebcam para realizar unas capturas de imagen con la webcam.

Fswebcam es una programa optimizado para la toma de imágenes utilizando una webcam. Se utiliza mediante la línea de comandos con lo que es muy útil para programar scripts automáticos en bash que se ejecuten cada cierto tiempo, o incluso controlando los pines GPIO de nuestra Raspberry Pi.

Utilizamos los siguientes comando para instalar el programa fswebcam:

Actualizamos los repositorios del sistema operativo Raspbian.

sudo apt update & apt upgrade

sudo apt install fswebcam

Si tenemos algún problema al captura con la webcam y no salen las capturas negras, debemos instalar los siguientes programas:

sudo apt install libjpeg8-dev

sudo apt install imagemagick

Para realizar una captura de imagen con el programa ejecutamos el siguiente

comando:

fswebcam -p YUYV -d /dev/video0 -r 640x480 foto.jpg

El programa fswebcam tiene varias opciones para realizar la captura de la imagen:

-d, --device [:]

Establece la fuente o dispositivo a usar. La fuente se selecciona automáticamente a menos que especificado en el prefijo. El valor predeterminado es / dev / video0.

-r, --resolución

Establece la resolución de imagen de la fuente o dispositivo. La resolución real utilizada puede difierir si la fuente o el dispositivo no pueden capturar a la resolución especificada. El valor predeterminado es "384x288".

--no-banner

Deshabilita el banner

Algunas opciones de salida:

--top-banner

Posiciona la imagen en la parte superior de la imagen

-bottom-banner

Posiciona el bannern en la parte inferior de la imagen. Es la posición por defecto.

--banner-colour <#AARRGGBB>

Especifica el color del banner. Utiliza formato hexadecimal para describir el color

--line-color <#AARRGGBB>

Establece el color de la línea divisoria. El valor predeterminado es "# ooFFoooo".

--text-color <#AARRGGBB>

Establece el color del texto. El valor predeterminado es "# ooFFFFFF".

--font <[nombre de archivo o fuente]: [tamaño de fuente]>

Establece la fuente utilizada en el banner. Si no se especifica ninguna ruta, la ruta en GDFONTPATH se busca la variable de entorno para la fuente. Los nombres de Fontconfig también se pueden usar si la biblioteca GD tiene soporte. El valor predeterminado es "sans: 10".

--no-shadow

Deshabilita la sombra de texto.

--shadow

Habilita la sombra de texto. Este es el comportamiento por defecto.

--title

Establece el texto principal, ubicado en la esquina superior izquierda del banner.

--no-title

No incluye el texto principal.

--subtitle

Establece el texto del subtítulo, ubicado en la parte inferior izquierda del banner.

--no-subtitle

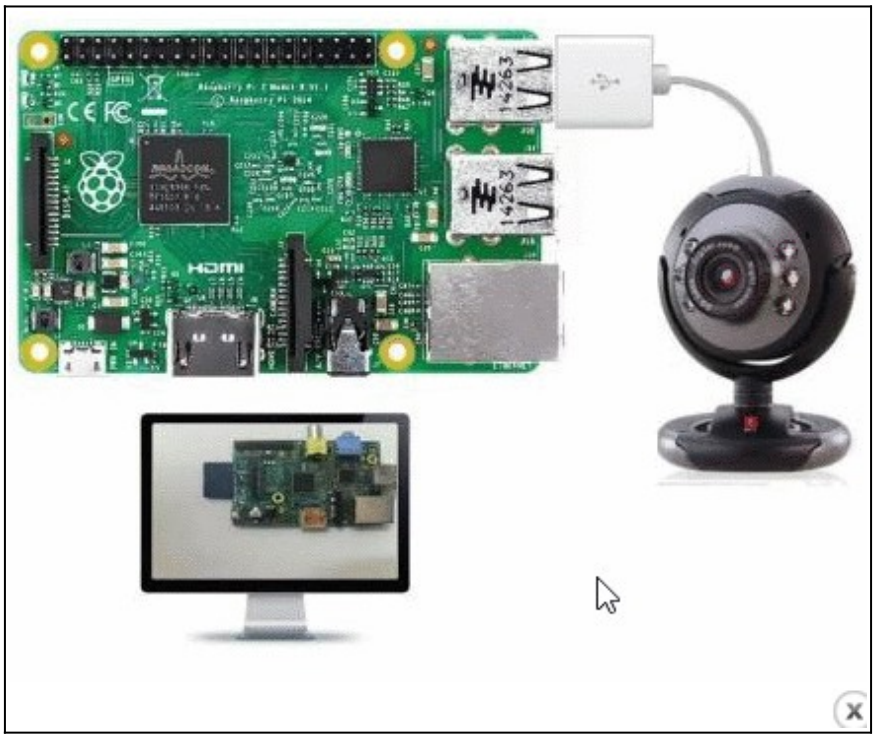
No incluye el texto del subtítulo.

-- timestamp

Establece el texto de la marca de tiempo, ubicado en la parte superior derecha del banner. Esta cadena es formateado por strftime. El valor predeterminado es "% Y-% m-% d% H:% M (% Z)".

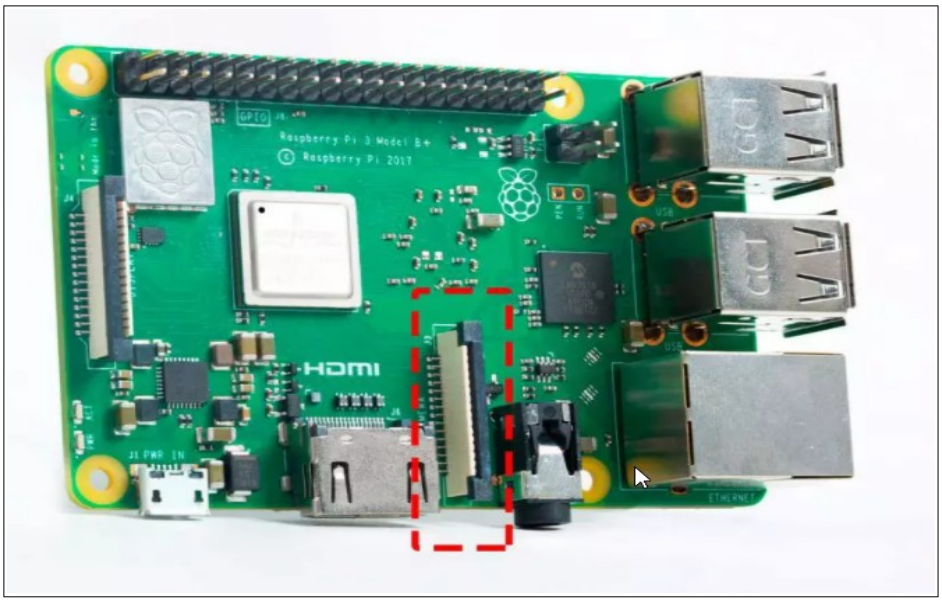
--no- timestamp

No incluye el texto de la marca de tiempo.

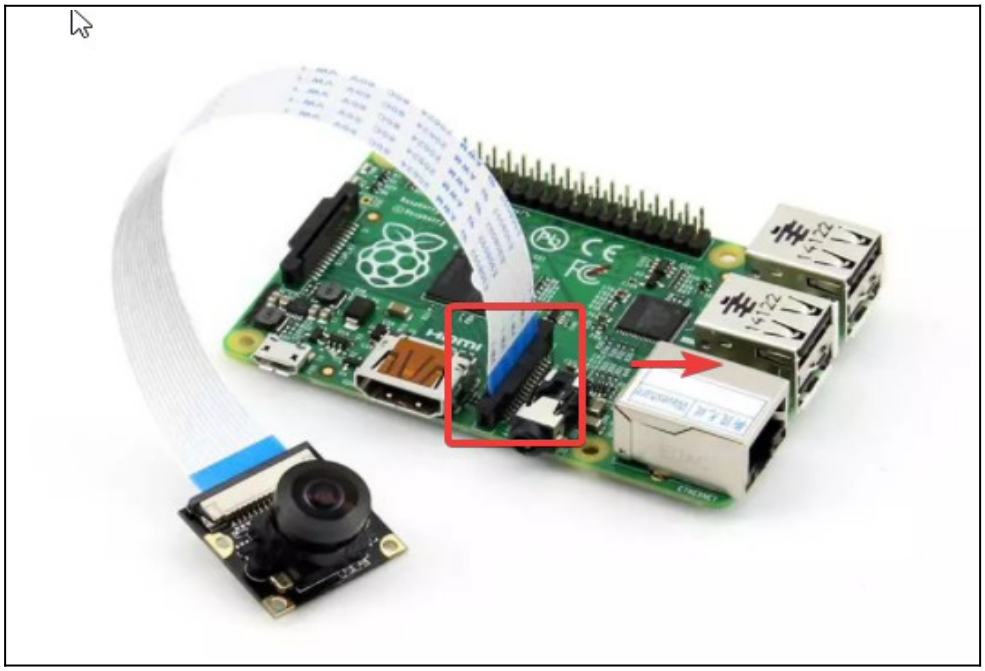


CONFIGURACIÓN Y INSTALACIÓN DE PiCAMERA

Para instalar la Pi Camera en la Raspberry Pi, debemos apagar la Raspebrry y conectarla en el puerto que esta entre el puerto HDMI y el puerto jack, es un conector que tiene escrito en la placa “CAMERA” como se ve en la imagen.



El cable cinta solo tiene una posición en el conector, la banda de la cinta con la lengüeta de plástico azul debe colocarse como se ve en la imagen , mirando hacia el conector jack y los usb, debemos asegurarnos de alinear ambos conectores en el mismo lado.



Antes de instalar la Pi Camera debemos tener actualizado el sistema operativo de la Raspberry , en este caso Raspbian.

Cuando tengamos puesta físicamente la cámara , encendemos la Raspberry Pi y nos dirigimos a la configuración de la Raspberry Pi para habilitar la cámara en el S.O de Raspbian, por defecto esta deshabilitada.

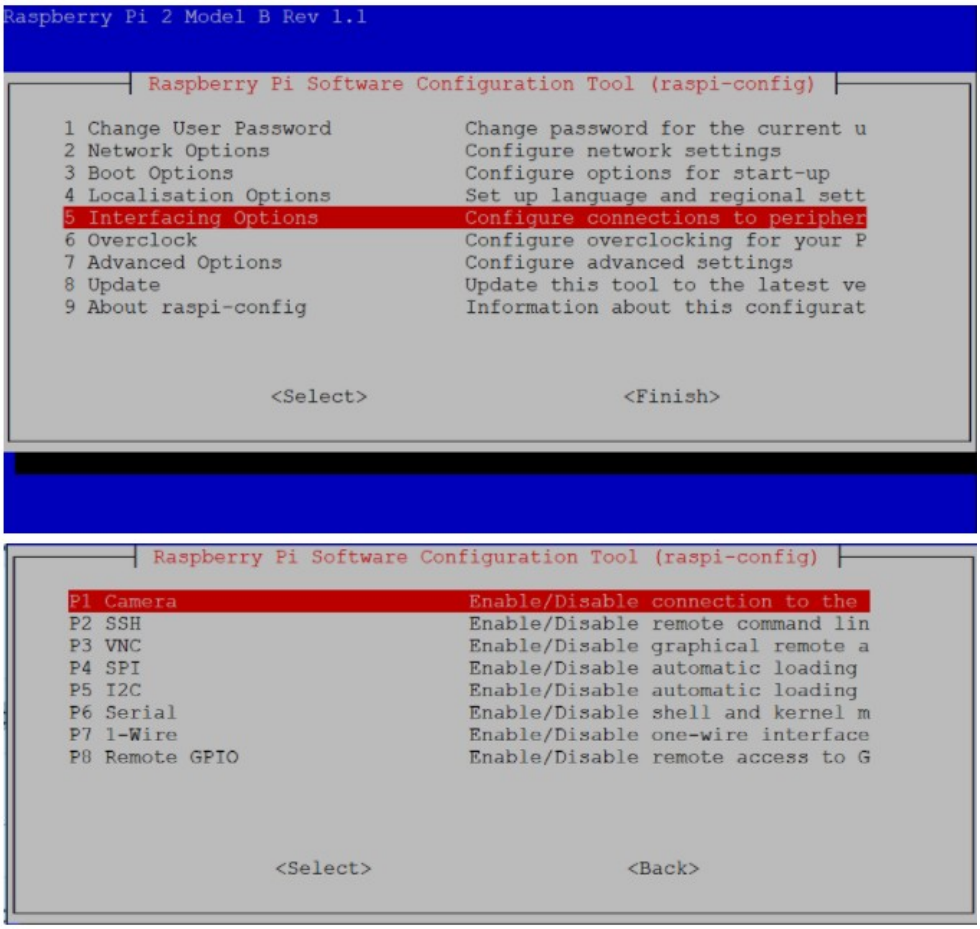
Podemos conectaros a la Raspberry por SSH o a través de la conexión HDMI a una pantalla.

Cuando estemos en el terminal de la Raspberry Pi en linea de comandos para entrar en la configuración de la Raspberry ejecutamos el siguiente comando :

sudo raspi-config

Vamos a la opción Opciones de Interfaz→ Cámara y habilitamos la Cámara dando a la opción Si . Salimos de la configuración y reiniciamos la Raspberry con el siguiente comando:

En estas dos capturas se ven las opciones que debemos seleccionar dentro de la configuración de la Raspberry para habilitar la Pi Camera



Para probar la Pi Camera podemos utilizar los siguientes comandos:

Para probar una captura de imagen de la Pi Camera

raspistill -o imagen.jpg

Para enviar varias capturas de imágenes en un lapso de 30s, cada 2s hará una captura

raspistill -t 30000 -tl 2000 -o IMAGES/image%04d.jpg

Para pasar varias capturas de imágenes a un video

ffmpeg -r 10 -f image2 -pattern_type glob -i 'image*.jpg' -s 1280x720 -vcodec libx264 video.mp4

Para probar una captura de video de la Pi Camera

raspivid -o video.h264

El siguiente paso es instalar una librería de python para que podamos utilizar la Pi Camera de manera nativa en Python.

Hacemos una actualización de los repositorios de nuestra Raspberry Pi , con el comando siguiente:

sudo apt update & sudo apt upgrade

Este comando lo ejecutamos sin ser root, si estuviéramos como root lo ejecutaríamos sin el sudo.

apt update & apt upgrade

Instalamos la librería de la Pi Camera

sudo apt install python-picamera

Para visualizar una imagen en tiempo real y ver una vista previa ejecutamos el siguiente código :

```
import time
import picamera

with picamera.PiCamera() as picam:
    picam.start_preview()
    time.sleep(10)
    picam.stop_preview()
    picam.close()
```

Si queremos tomar una foto y guardarla en la memoria de la Raspberry , podemos utilizar este código , en este caso la captura la hacemos en formato **jpg**, pero podemos hacerla en **png, gif, bmp , yuv, rgb, raw.**

```
import time
import picamera

with picamera.PiCamera() as picam:
    picam.start_preview()
    time.sleep(5)
    picam.capture('nombre.jpg')
    picam.stop_preview()
    picam.close()
```

Si queremos hacer un video usamos el siguiente código , grabando en formato H264,es importante agregar el wait_recording() ya que aparte de servir como el time.sleep(), hace un reconocimiento al mismo tiempo que va gravando y si por ejemplo ya no hay espacio en disco para la grabación.

```
import time
import picamera

with picamera.PiCamera() as picam:
    picam.start_preview()
    picam.start_recording('video.h264')
        picam.wait_recording(20)
        picam.stop_recording()
    picam.stop_preview()
    picam.close()
```

También podemos indicar la resolución de la fotografía en el código

```
import time
import picamera

with picamera.PiCamera() as picam:
    picam.resolution = (2592, 1944)
    picam.start_preview()
    time.sleep(3)
    picam.capture('foto.jpg')
    picam.stop_preview()
    picam.close()
```

También debido a que entre mas resolución la imagen es mas pesada, así que también nos brindan un modo de re-escalar la imagen, es decir que abarque todo lo que la cámara ve pero que se guarde en una resolución menor .

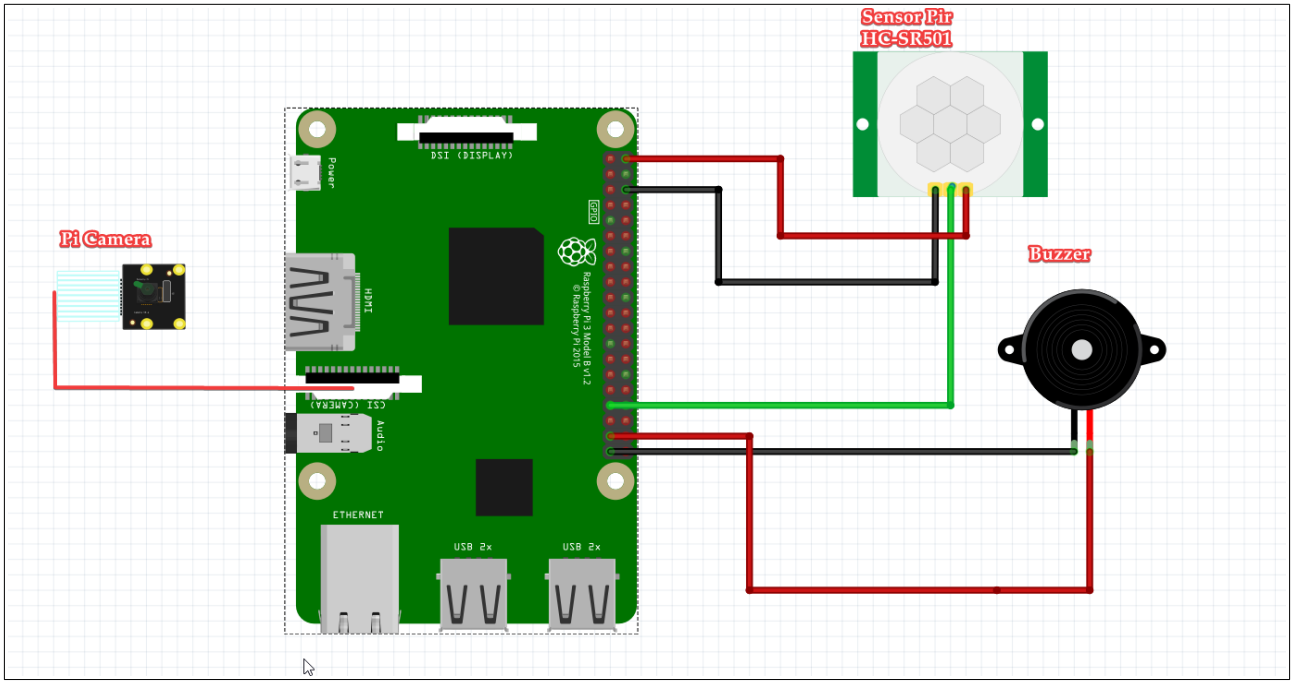
```
import time
import picamera

with picamera.PiCamera() as picam:
    picam.resolution = (2592, 1944)
    picam.start_preview()
    time.sleep(3)
    picam.capture('foto.jpg',resize=(1024,768))
    picam.stop_preview()
    picam.close()
```

También tenemos una opción sobre todo si usamos la cámara para un sistema de seguridad y no queremos que se encienda el led , tenemos la opción de desactivarlo con el siguiente código

```
import time
import picamera

with picamera.PiCamera() as picam:
    picam.led= False
    picam.start_preview()
    time.sleep(3)
    picam.stop_preview()
    picam.close()
```

INSTALACIÓN Y CONFIGURACIÓN DE RASPBERRY CON SENSOR PIR, PiCAMERA Y BUZZER

Para instalar el sistema de seguridad hemos utilizado el siguiente esquema el Sensor Pir que va al GPIO13 de la Raspberry y el Buzzer al GPIO26 , la configuración que se ha usado ha sido GPIO.BCM (BROADCAST CHANEL)



Se han utilizado 3 diferentes códigos para hacer funcionar el sistema de seguridad con el sensor Pir, la Pi Camera y el Buzzer.

El primer código llamado Pir_Proyecto_Securityv1.py, hemos utilizado las siguientes librerías:

```
import RPi.GPIO as GPIO
import time
import picamera
from datetime import datetime
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
import os
```

En este código están incluidas las librerías que se han utilizado para subir imágenes a Google Drive , que explicaremos más adelante.

En este primer código hemos utilizado una función que nos varia la frecuencia del

zumbador o buzzer inicia a 10hz y cambia a 1Khz configurando el ciclo de trabajo al 10%.

```
def zumbador_frecuencia():
    buzzer = GPIO.PWM(zumbador, 1000)  # Set frequency to 1 Khz
    buzzer.start(10)  # Set dutycycle to 10
    buzzer.ChangeDutyCycle(10)
    buzzer.ChangeFrequency(1000)
    time.sleep(5)
```

Podemos variar el volumen del sonido modificando el ciclo de trabajo. 0% no producirá ningún sonido, 50% será el volumen máximo. Entre 50% y 100% es lo mismo que entre 0% y 50%.

En siguiente código ponemos una función que nos indicara el formato de la imagen capturada con la fecha y la hora:

```
def recupera_datetime():
    ahora = datetime.now()
    dia = ahora.day
    mes = ahora.month
    ano = ahora.year
    hora = ahora.hour
    minuto = ahora.minute
    segundo = ahora.second
    return dia, mes, ano, hora, minuto, segundo
```

La siguiente función es la que nos guarda la fotografía o captura en en la memoria :

```
def disparo_camara():
    dia, mes, ano, hora, minuto, segundo = recupera_datetime()
    nom_archivo = "IMAGES/ Imagen {:02d}-{:02d}-{}_{:02d}{:02d}{:02d}.jpg".format(dia, mes, ano, hora, minuto, segundo)
    with picamera.PiCamera() as mi_imagen:
        mi_imagen.start_preview()
        time.sleep(0.1)
        mi_imagen.capture(nom_archivo)
        mi_imagen.stop_preview()
        mi_imagen.close()
    subir_googledrive(nom_archivo)
```

Y para finalizar utilizaremos la siguiente función para realizar una ráfaga de imágenes cuando el sensor Pir detecte movimiento, la Pi Camera hará diferentes capturas hasta que el contador llegue a 10 capturas y la alarma del buzzer se encenderá. Cuando no detecte movimiento se apagara el buzzer e imprimirá por pantalla que no detecta movimiento:

```
def comprobar_movimiento():
    if GPIO.input(pir):
        print("Movimiento detectado")
        #crear_archivo_texto('Alarma.txt', 'Se ha detectado movimiento del sensor', '1w1DdcuYVRdpbeTacb1C2hYMxMjX_nWg_')
        GPIO.output(zumbador,1)
        zumbador_frecuencia()
        cont_imagen = 0
        while cont_imagen < 10:
            disparo_camara()
            cont_imagen += 1
            time.sleep(0.7)
        GPIO.output(zumbador,0)
        os.system("rm -rf IMAGES/*.jpg")
    else:
        print("No se detecta movimiento")
```

El segundo código llamado Pir_Proyecto_Securityv2.py, una mejora del primer código, hemos utilizado las siguientes librerías:

```
import RPi.GPIO as GPIO
import time
import os
import requests
import json
import telebot
from datetime import datetime
from pydrive2.auth import GoogleAuth
from pydrive2.drive import GoogleDrive
```

En este segundo código se han incluido las librerías de sistema os , requests, json y telebot para importar el video generado por varias capturas de la Pi Camera a un bot de Telegram y la librería pydrive2 para importar el video a una carpeta de Google Drive especifica llamada Videos .

Función para enviar el video a Telegram, escribiendo un mensaje de Alarma al bot de Telegram :

```
def subir_telegram_boot(ruta_archivo):
        #ruta_archivo = ruta_archivo.split('/')[-1]

        r = requests.post('https://api.telegram.org/bot1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM/sendMessage',
                                        data={'chat_id': '6728832', 'text': 'Alerta video de seguridad'})

        data = json.loads(r.text)
        print(data['ok'])

        video = open(ruta_archivo,'rb')
        tb.send_video(6728832, video)
```

Función para enviar el video generado a Google Drive :

```
#Función para subira archivos a Google Drive a una ruta especifica y se autentifique es necesario que contenga los siguientes archivos:
# client_secrets.json -> datos de id_cliente y secreto_cliente de OAuth 2.0
# settings.yaml -> registramos los datos de cliente OAuth
# quickstart.py -> genera archivo de credenciales (credentials.json)

def subir_googledrive(ruta_archivo):
        id_carpetaDrive='1df1GAUOozcuHGBtFfNhaX7VghN7j8Esp'
        autenticacion = GoogleAuth()
        autenticacion.LocalWebserverAuth()
        drive = GoogleDrive(autenticacion)
        archivoDrive = drive.CreateFile({'parents':[{'kind':'drive#fileLink','id': id_carpetaDrive}]})
        archivoDrive['title'] = ruta_archivo.split('/')[-1]
        archivoDrive.SetContentFile(ruta_archivo)
        archivoDrive.Upload()
```

Función para realizar varias capturas con la PiCamera y convertir a un video a través de la librería del sistema **os.system** y el comando **ffmepg**, también utilizaremos el comando **raspistill** para que un 1 min nos haga una ráfaga de imágenes, una cada 2 segundos.

Enviara el video al bot de Telegram y a Google Drive, y eliminara las capturas y el video de las carpetas /IMAGES y /VIDEOS de la memoria de la Raspberry donde se han guardado:

```
def peripheral_loop():

        time.sleep(1)

        dia, mes, ano, hora, minuto, segundo = recupera_datetime()

        nom_archivo = "{:02d}-{:02d}-{}_{:02d}{:02d}{:02d}.mp4".format(dia, mes, ano, hora, minuto, segundo)

        if current_state != previous_state: # El operador != evalua si los valores son distintos.
                new_state = "HIGH" if current_state else "LOW"
                print("GPIO pin %s is %s" % (pir, new_state))
                if new_state == "HIGH":
                        print("Alerta Movimiento")
                        #os.system("raspivid -t 10000 -w 1920 -h 1080 -fps 30 -b 1200000 -p 0,0,800,600 -o video.h264")
                        #os.system("MP4Box -add video.h264  VIDEOS/{}".format(nom_archivo))
                        os.system("raspistill -t 60000 -tl 2000 -w 1920 -h 1080 -o IMAGES/image%04d.jpg")# durante un periodo de 60segundos(60000ms) haz una captura cada
2s(2000ms)
                        os.system("ffmpeg -f image2 -pattern_type glob -framerate 30 -i 'IMAGES/image*.jpg' -s 1920x1080 -vcodec libx264  VIDEOS/{}".format(nom_archivo))
                        #os.system("fswebcam -p YUYV -d /dev/video0 -r 800x600 --top-banner  PHOTOS/{}.jpg .format(nombre_imagen))
                        subir_googledrive('VIDEOS/{}'.format(nom_archivo)) # sube el archivo de video a googledrive
                        subir_telegram_boot('VIDEOS/{}'.format(nom_archivo))# suabe el archivo de video a telegram
                        os.system("rm -rf IMAGES/*.jpg")
                        os.system("rm -rf VIDEOS/*.mp4")
                        GPIO.output(zumbador,1)
                        zumbador_frecuencia()
                else:
                        GPIO.output(zumbador,0)
                        print("No hay movimiento")
                        time.sleep(2)
```

En el tercer código llamado Pir_Proyecto_Securityv3.py hemos hecho una variante del segundo código en el cual nos envía una ráfaga de imágenes tanto a Google Drive como a Telegram utilizando la librería de la Picamera, se han utilizado las siguientes librerias:

```
import RPi.GPIO as GPIO
import time
import picamera
import os
from datetime import datetime
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
import telebot
```

He utilizado un función, gracia a la colaboración compañero Sergio Santín que inicializa la cámara, captura la imagen , la detiene y la cierra enviando los archivos a Google Drive y al Bot de Telegram, utilizando la librería de la Pi Camera, realiza un contador hasta 15 capturas.

```
#Función que inicializa la cámara, captura la imagen, la detiene y la cierra.
Finalmente,
#el archivo generado lo publicacrá en la cuenta de Google Drive
def disparo_camara():
    nombre_archivo = generar_carpeta_archivo()
    print("··· Capturando imágen en -> '{}.".format(nombre_archivo))
    with picamera.PiCamera() as mi_imagen:
        mi_imagen.start_preview()
        cont_imagen = 0
        while cont_imagen < 15:
            nombre_archivo = generar_carpeta_archivo()
            print("··· Capturando imágen en -> '{}.".format(nombre_archivo))
            mi_imagen.capture(nombre_archivo)
            cont_imagen += 1

            try:
                subir_telegram_boot(nombre_archivo)
                subir_googleDrive(nombre_archivo)
                print("··· Archivo subido a Google Drive.")

            except Exception as e:
                print("··· No se ha completado la subida del archivo.")
                print("··· ERROR: {}".format(e))
        mi_imagen.stop_preview()
        mi_imagen.close()


#Función que recupera actividad del sensor y dispara la cámara
def comprobar_movimiento():
    if GPIO.input(pir):
        print("··· Sensor HC-SR501: Movimiento detectado.")
        GPIO.output(zumbador,1)
        zumbador_frecuencia()
        disparo_camara()
        GPIO.output(zumbador,0)
    else:
        print("··· Sensor HC-SR501: No se detecta movimiento.")
```

También se ha utilizado otra función para crear una carpeta que contendrá las imágenes capturadas durante las 24h de un día especifico en la memoria de la Raspberry y otra función para eliminar las carpetas cada mes , incluyendo los meses que contienen 31 días gracias al compañero Sergio aplicadas en su proyecto:

```
#Función que crea la carpeta que contendrá las imagenes capturadas durante las 24h
de
#un día específico.
def generar_carpeta_archivo():
    d, me, a, h, mi, s = recupera_datetime()
    n_c = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(d, me, a)#nombre carpeta
    r_a = "{}/PVMImg{:02d}{:02d}{:02d}{:02d}{:02d}{:02d}.jpg".format(n_c, d, me, a,
h, mi, s)#ruta(carpeta) y archivo de imagen jpg que se crea en el sistema
    genera_carpeta = "mkdir {}".format(n_c) #comando OS para crear carpeta
    if os.path.isdir(n_c): #verifica si existe la carpeta
        pass
    else:
        os.system(genera_carpeta)#método que permite ejecutar el comando OS
        print("··· La carpeta '{}' se ha creado satisfactoriamente.".format(n_c))

    return r_a
```

```
def eliminar_carpeta():
    d, me, a, h, mi, s = recupera_datetime()
    #permite eliminar las carpetas que se generaron en diciembre (mes 12) del año
anterior.
    if me == 1:
        m_anterior = 12
        a_anterior = a - 1
        b_carpeta = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(d, m_anterior,
a_anterior)
        borrar_carpeta(b_carpeta)
    #permite  eliminar, el dia 2 de marzo, las carpetas generadas los dias 29, 30,
31 del mes de enero
    #puesto que febrero no dispone de estos días, excepto el 29 de año bisiesto.
    #también eliminará la carpeta del 2 de febrero.
    elif(me == 3 and d == 2):
        d_enero = [29, 30, 31]
        for d_e in d_enero:
            b_carpeta_e = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(d_e, 1, a)
            borrar_carpeta(b_carpeta_e)
        b_carpeta_f = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(d, 2, a)
        borrar_carpeta(b_carpeta_f)
    #permite eliminar las carpetas de los días 31 de los pertinentes meses (marzo,
mayo, agosto, octubre)
    #junto a las carpetas del día 30
    elif ((me == 4 or me == 6 or me == 9 or me == 11) and (d == 30)):
        m_anterior = me - 1
        b_carpeta1 = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(d, m_anterior, a)
        b_carpeta2 = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(31, m_anterior, a)
        print("··· Se borrarán las carpetas del día 30 y 31 del mes anterior.")
        borrar_carpeta(b_carpeta1)
        borrar_carpeta(b_carpeta2)
    #permite eliminar todas las carpetas que no dispongan de 'atributo' especial.
    #tanto el dia 31 de julio como el 31 de diciembre, entran en este último
supuesto
    #dado que sus meses siguientes respectivamente también disponen de 31 días
(agosto y enero)
    else:
        m_anterior = me - 1
        b_carpeta = "PVM_IMAGES{:02d}-{:02d}-{:02d}".format(d, m_anterior, a)
        borrar_carpeta(b_carpeta)
```

ENVIAR CAPTURAS Y VIDEO A GOOGLE DRIVE

Para enviar cualquier tipo de dato de la Raspberry a Google drive , en este caso imágenes y videos hemos utilizado dos librerias de la API de Google PyDrive y Py Drive2.

Para instalar la librería debemos ejecutar el siguiente código:

sudo pip3 install PyDrive o sudo pip3 install PyDrive2

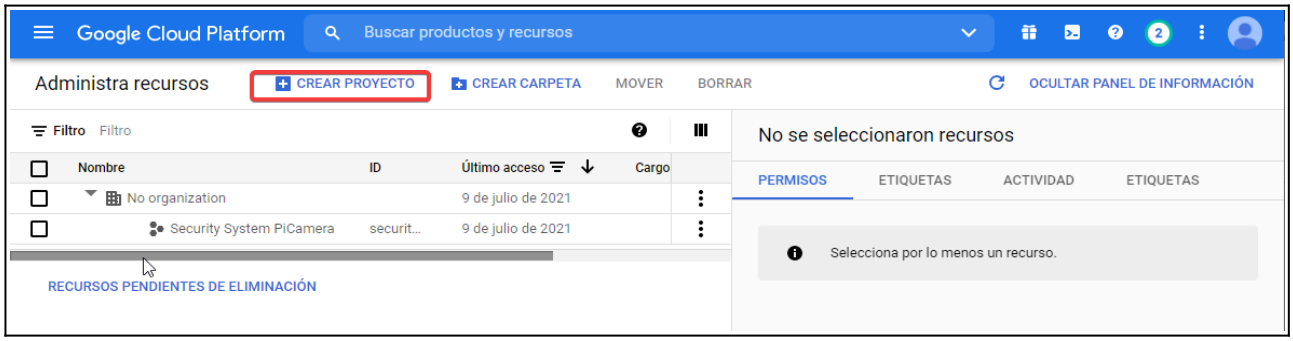
Después debemos seguir los siguientes pasos para habilitar la API de Google Drive para una cuenta especifica,para obtener las credenciales de autenticación.

De hecho, para configurar una copia de seguridad en línea en Google Drive, es necesario primero activar esta opción específica en su cuenta de Google .

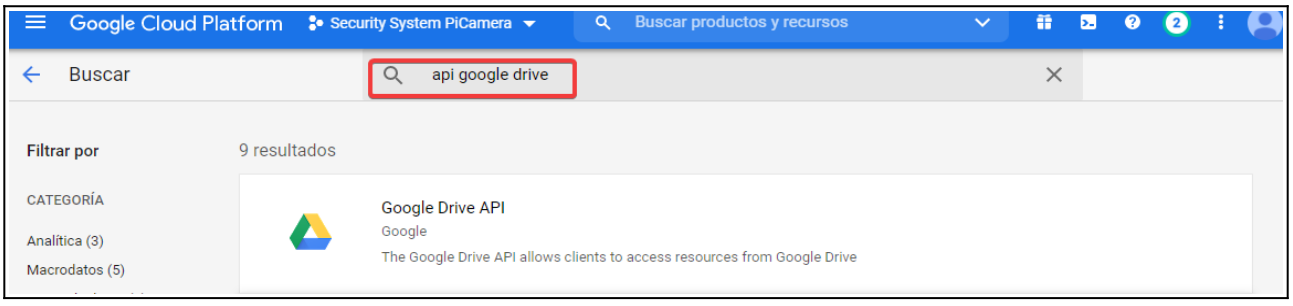
Este proceso nos genera un client id y un client secret.

Para generar el client secret.json desde el portal de Google, seguimos estos pasos:

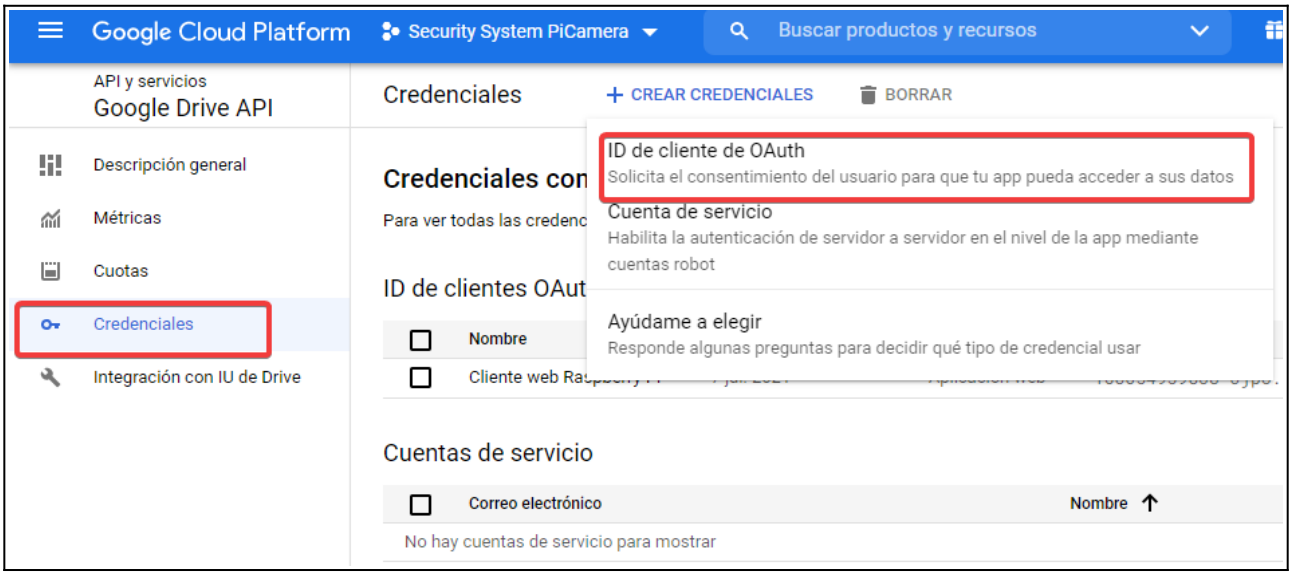
1 Vaya a la Consola de API y cree su propio proyecto.



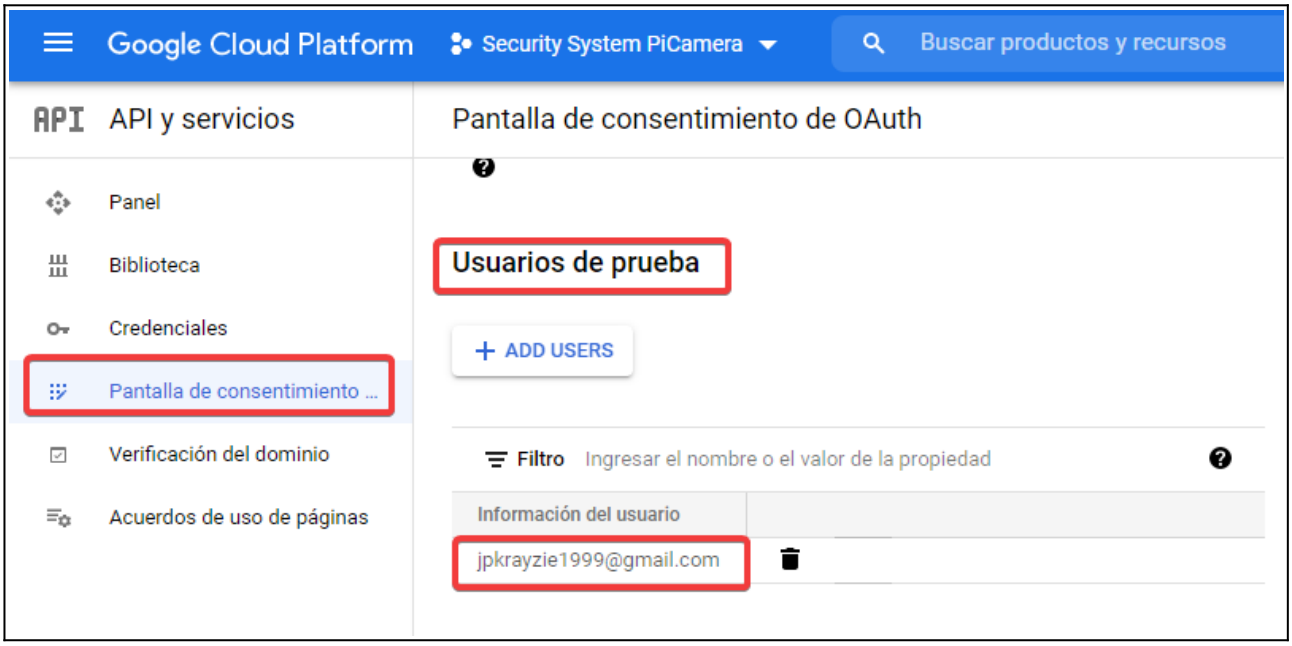
2 Busque "API de Google Drive", seleccione la entrada y haga clic en "Habilitar".



3 Seleccione 'Credenciales' en el menú de la izquierda, haga clic en 'Crear credenciales', seleccione 'ID de cliente Oauth'.



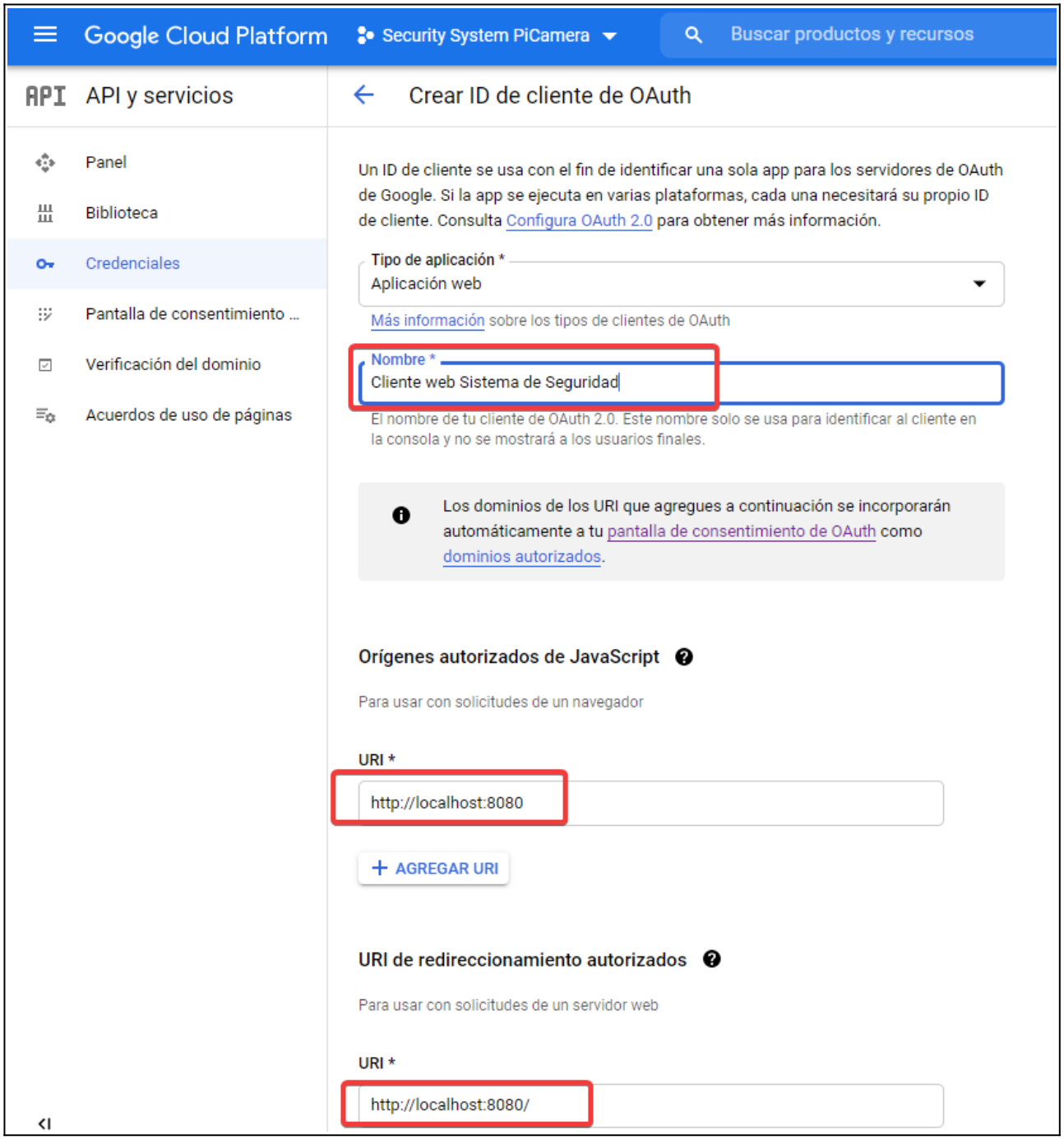
4 Ahora, debe configurar el nombre del producto y la pantalla de consentimiento -> haga clic en 'Configurar pantalla de consentimiento' y siga las instrucciones e indicar un usuario de prueba con nuestra dirección de correo.



5 Seleccione 'Tipo de aplicación' para que sea una aplicación web .Ingrese un nombre apropiado.

6 Ingrese http: // localhost: 8080 para 'Orígenes autorizados de JavaScript'.

7 Ingrese http: // localhost: 8080 / para 'URI de redireccionamiento autorizado'.Clic en Guardar'



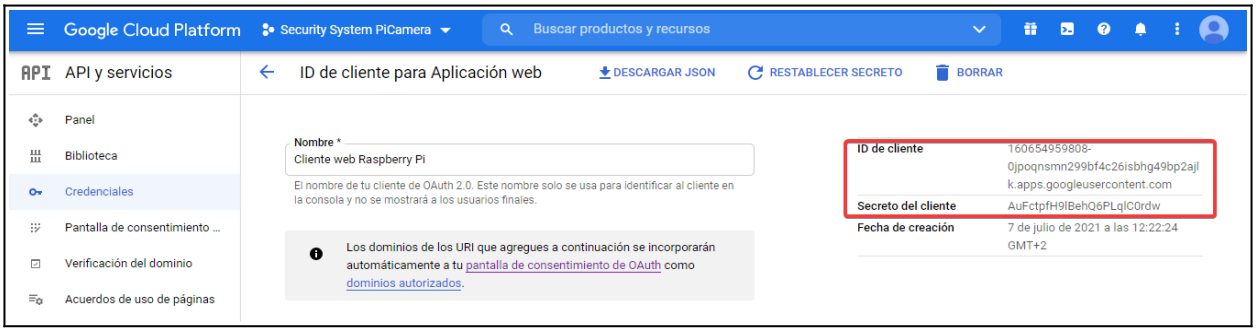
8 Haga clic en 'Descargar JSON' en el lado derecho del ID de cliente para descargar client_secret_ <ID realmente largo> .json .

9 El archivo descargado tiene toda la información de autenticación de su aplicación. Cambie el nombre del archivo a "client_secrets.json" y colóquelo en su directorio de trabajo.

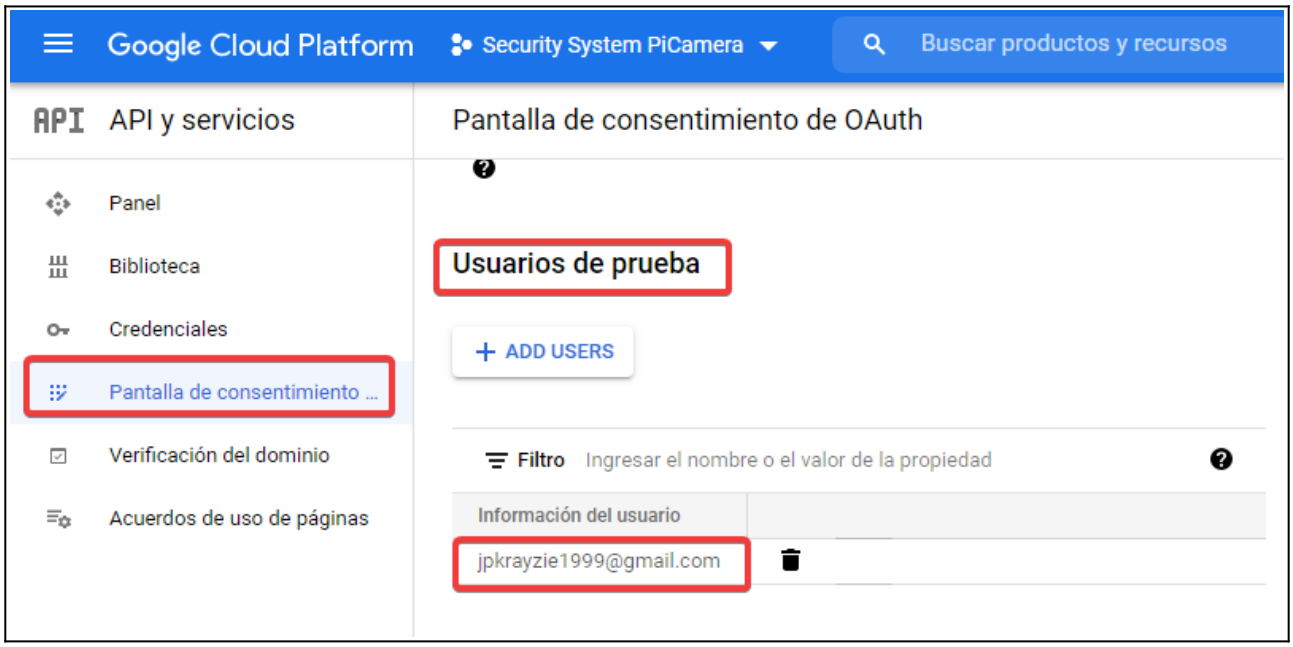
Ahora debemos crear el archivo **settings.yaml** y modificar el parámetro **client_id** y **client_secret** que encontramos en nuestras credenciales de Google Cloud como se ve en la captura:

```
client_config_backend: settings
client_config:
  client_id: 629742331936-solk8l4qu39r00sq9v0obolsa2f2fltn.apps.googleusercontent.com
  client_secret: uZ32T4stexXDpDEDWYj8ClPb

save_credentials: True
save_credentials_backend: file
save_credentials_file: credentials.json

get_refresh_token: True

oauth_scope:
  - https://www.googleapis.com/auth/drive
```

Estos datos los encontramos editando el cliente para la aplicación web:



Si nos da un error 403 de validación a Google Drive cuando nos abre el navegador es porque debemos crear un usuario de prueba y poner nuestra dirección de e-mail en el portal de Google Cloud:



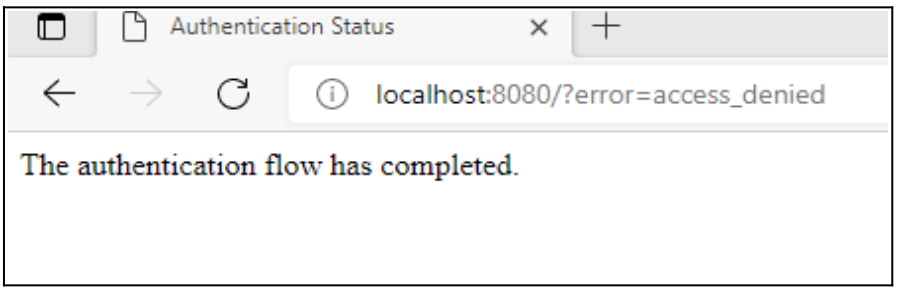
Seguidamente ejecutamos el código de python **quickstar.py** para que nos valide Google Drive desde la powershell:

Código **quickstar.py**:

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
gauth = GoogleAuth()
gauth.LocalWebserverAuth()
gauth.LoadCredentialsFile(gauth)
```

```
PS C:\Users\USUARI-A\Desktop\Api Google Drive> ls


    Directorio: C:\Users\USUARI-A\Desktop\Api Google Drive


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        07/07/2021     10:25                addicionals
-a----        07/07/2021     12:22            449 client_secrets.json
-a----        06/07/2021     16:48            160 quickstart.py
-a----        06/07/2021     16:46            362 settings.yaml


PS C:\Users\USUARI-A\Desktop\Api Google Drive> python .\quickstart.py
```

Si todo ha funcionado correctamente nos habrá creado un archivo llamado **credentials.json** y nos habrá validado y autorizado en Google Drive y nos saldrá la siguiente ventana en el navegador:



Ya tendremos el archivo credentials.json que podemos utilizarlo para subir cualquier tipo de archivo a Google Drive dentro de cualquier código de python.

En nuestro código de python deberemos poner la siguiente función para que nos suba los archivos a Google Drive ya sean imágenes, archivos o videos.

```
def google_drive(archivo):
    gauth = GoogleAuth()
    gauth.LocalWebserverAuth()
    drive = GoogleDrive(gauth)
    file_drive = drive.CreateFile({'title': archivo, 'mimeType':'image/jpeg'})
    file_drive.SetContentFile(archivo)
    file_drive.Upload()
```

Esta es la única función que utilizo para subir archivos a Google Drive , en la carpeta del Proyecto debe incluir el **setting.yaml, credentials.json, client_secrets.json** y internamente llama a estos archivos las librerías Pydrive y Pydrive2.

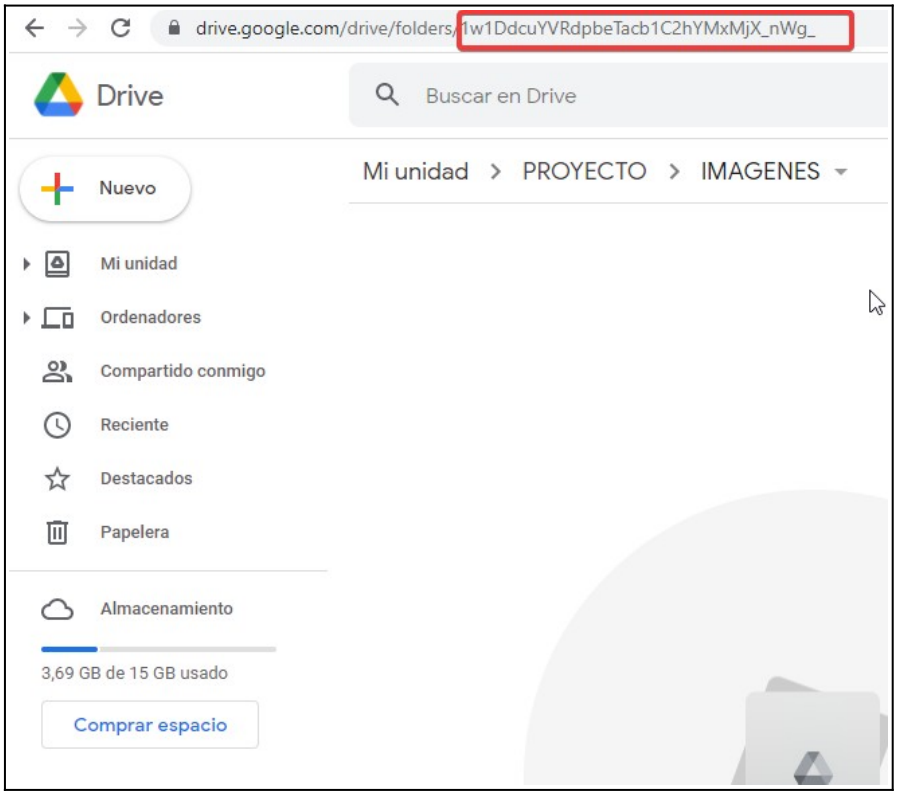
En los tres códigos de Python del proyecto de Sistema de Seguridad hemos utilizado la siguiente función para que envíe los datos a Google Drive:

```
#Función que permitirá subir las imagenes generadas al Google Drive
#Autenticación:
#client_secrets.json -> datos de id_cliente y secreto_cliente de OAuth 2.0
#settings.yaml -> registramos los datos de cliente OAuth
#quickstart.py -> genera archivo de credenciales (credencial.json)
def subir_googleDrive(ruta_archivo):
    id_carpetaDrive = "1w1DdcuYVRdpbeTacb1C2hYMxMjX_nWg_"
    autenticacion = GoogleAuth()
    autenticacion.LocalWebserverAuth()
    drive = GoogleDrive(autenticacion)
    archivoDrive = drive.CreateFile({'parents': [{'kind': 'drive#fileLink', 'id': id_carpetaDrive}]})
    archivoDrive['title'] = ruta_archivo.split('/')[-1]
    archivoDrive.SetContentFile(ruta_archivo)
    archivoDrive.Upload()
```

Para subir los archivos a una carpeta o otra de Google Drive debemos indicar el id de la carpeta de Google Drive en el código como observamos en la captura:

```
def subir_googleDrive(ruta_archivo):
    id_carpetaDrive = "1w1DdcuYVRdpbeTacb1C2hYMxMjX_nWg_"
    autenticacion = GoogleAuth()
    autenticacion.LocalWebserverAuth()
    drive = GoogleDrive(autenticacion)
    archivoDrive = drive.CreateFile({'parents': [{'kind': 'drive#fileLink', 'id': id_carpetaDrive}]})
    archivoDrive['title'] = ruta_archivo.split('/')[-1]
    archivoDrive.SetContentFile(ruta_archivo)
    archivoDrive.Upload()
```

Este dato lo vemos en Google Drive en la ruta de la carpeta como se ve en la imagen:



ENVIAR CAPTURAS Y VIDEOS A UN BOOT DE TELEGRAM

Para enviar capturas, videos o otro tipo de datos a Telegram, debemos seguir las siguientes pautas:

1. Debemos crear un boot donde podamos visualizar estos datos, para crear nuestro bot para Telegram escribimos en nuestro navegador de internet la dirección siguiente https://telegram.me/botfather. Otra opción es buscar directamente en Telegram botfather.
2. Seleccionamos o escribimos /newbot para crear nuestro propio boot de Telegram, le indicamos un nombre
3. El Bot nos responde ahora: ¿Que nick queremos usar para este bot?, recuerda que el nick debe de acabar en bot o _bot. ejem: @nick_bot, @nickbot, en nuestro proyecto el nick se llama @Krayziebot
4. Ya tendremos el bot de Telegram para enviar los datos en este caso video y imágenes de la Raspberry a Telegram, para poder enviar los datos necesitamos el Token de nuestro boot de Telegram y el chat id, para poder indicarlo en las dos API de python que utilizaremos
5. Para averiguar el Token, entramos en nuestro boot en este caso @Krayziebot y nos vamos al apartado API Token y copiamos el código que nos facilitan, para poder indicarlo en la función de nuestro código de python.
6. Para averiguar el chat id utilizamos el siguiente enlace donde una parte es el token que nos ha facilitado nuestro boot como se ve en la captura amarcado en rojo, enviamos un mensaje en nuestro boot y ejecutamos el enlace que se ve en la captura:

```
# Como saber el chatid
# https://api.telegram.org/bot1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM/getUpdates
```

Nos responderá indicanos una respuesta json, en el texto json busque su random message y obtenga la identificación de chat en ese objeto.


```
{"ok":true,"result":[{"update_id":273635669,
"message":{"message_id":165,"from":{"id":6728832,"is_bot":false,"first_name":"Xavier","language_code":"es"},"chat":
{"id":6728832,"first_name":"Xavier","type":"private"},"date":1626383995,"text":"Hola"}}]}
```


En este caso el chat id es 67228832

Con el Token y el chat id ya podemos utilizar estos parámetros para enviar datos de la Raspberry a Telegram.

He utilizado dos librerias para importar el video o capturas de imágenes del sistema de seguridad a un bot de Telegram.

En el primer caso he utilizado un API para python que se llama **Telebot**.

Para instalar la librería debemos ejecutar el siguiente código:

sudo pip install pyTelegramBotAPI

En la siguiente captura podemos ver un código de Python con esta librería que envía tanto un archivo como un video de un ruta especifica de la Raspberry:

```
import telebot # Importamos las libreria
TOKEN = '1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM' # Ponemos nuestro Token generado con el @BotFather
tb = telebot.TeleBot(TOKEN)

tb.send_message('6728832', 'Este el primer mensaje enviado al boot de Looki de Telegram')
photo = open('/home/pi/IMAGES/image0006.jpg', 'rb')
tb.send_photo(6728832, photo)

video = open('/home/pi/VIDEOS/video.mp4', 'rb')
tb.send_video(6728832, video)
```

En el segundo caso hemos utilizado una API para python que se llama **Requests.**

Para instalar la librería debemos ejecutar el siguiente código:

sudo apt install python3-requests

En la siguiente captura podemos ver un código de Python con esta librería que envía tanto un mensaje al bot de Telegram como una imagen:

```
import requests
import json

r = requests.post('https://api.telegram.org/bot1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM/sendMessage',
             data={'chat_id': '6728832', 'text': 'Mensaje de la Raspberry a Telegram '})

data = json.loads(r.text)
print(data['ok'])


requests.post('https://api.telegram.org/botbot1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM/sendMessage/sendPhoto',
             files={'photo': (IMAGES/image0006.jpg, open(IMAGES/image0006.jpg, 'rb'))},
             data={'chat_id':'6728832','caption':'Imagen PiCamera'})
```

En las 2 ultimas versiones del sistema de seguridad hemos utilizado las dos librerías , para ello hemos importado las librerias requests y telebot , y hemos creado una variable llamada TOKEN, con el código del Api Token de nuestro bot de Telegram. También la librería json es para saber si la respuesta del archivo json es True o False como se ve en la 2 imagen en la función subir a Telegram.

En la 3 imagen , subimos varias imágenes al boot de Telegram.

```
import RPi.GPIO as GPIO
import time
import os
import requests
import json
import telebot
from datetime import datetime
from pydrive2.auth import GoogleAuth
from pydrive2.drive import GoogleDrive

pir = 13
zumbador = 26

TOKEN = '1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM'
tb = telebot.TeleBot(TOKEN)
```

```
def subir_telegram_boot(ruta_archivo):
        #ruta_archivo = ruta_archivo.split('/')[-1]

        r = requests.post('https://api.telegram.org/bot1887343548:AAEuhLzE08ibMjyvzQ0c5tr22KdIvuYonPM/sendMessage',
                                data={'chat_id': '6728832', 'text': 'Alerta video de seguridad'})

        data = json.loads(r.text)
        print(data['ok'])

        video = open(ruta_archivo,'rb')
        tb.send_video(6728832, video)
```

```
def subir_telegram_boot(ruta_archivo):
    # ruta_archivo = ruta_archivo.split('/')[-1]

    tb.send_message('6728832', 'Alerta de Seguridad')
    photo = open(ruta_archivo, 'rb')
    tb.send_photo(6728832, photo)
```

REALIZAR UN ARCHIVO AUTOEJECUTABLE CON EL CÓDIGO DE PYTHON

Para crear un archivo autoejecutable con todas las librerias sin necesidad de ejecutar python necesitamos instalar la librería Pyinstaller con el siguiente comando:

pip install pyinstaller

Ejecutamos el código de python para que nos realice un ejecutable

pyinstaller Pir_Proyecto_Securityv2.py

Nos creara una carpeta llamada dist, en esta debemos poner los archivos *.json y el yaml, también los directorios donde van las imágenes y los videos

Al archivo autoejecutable le debemos darle permisos de ejecución, lectura y escritura tanto al propietario, al grupo como a todos con el siguiente comando

chmod 777 Pir_Proyecto_Securityv2

Si queremos que este autoejecutable funcione al arrancar la Raspberry , deberemos indicarlo en el siguiente archivo **etc/rc.local** antes del comando **exit** indicando la ruta del archivo ejecutable dentro de nuestra Raspberry

Por ejemplo, si quiero llamar a un script de Python 3 llamado example.py y ubicado en la carpeta /home/pi en el inicio, reemplazaré la línea de salida o con:

/usr/bin/python3 /home/pi/example.py

Cabe señalar algunos puntos importantes.

Primer punto, el programa será ejecutado por el usuario root y por tanto tendrá todos los derechos. ¡Ten cuidado con lo que estás haciendo!

Segundo punto, siempre debe usar rutas absolutas y no relativas, el comportamiento de las rutas relativas es impredecible.

En tercer y último punto, su programa debe devolver el control al script o la Raspberry Pi

nunca podrá terminar de arrancar. Si su programa realiza un bucle infinito, debe ejecutarlo en segundo plano agregando un & después de ordenar. En nuestro seria:

/usr/bin/python3 /home/pi/example.py &

BIBLIOGRAFÍA

Para realizar el siguiente proyecto se ha documentado en las siguientes fuentes de información :

TELEGRAM

https://geekytheory.com/telegram-programando-un-bot-en-python

https://qastack.mx/programming/32423837/telegram-bot-how-to-get-a-group-chat-id

https://atareao.es/tutorial/crea-tu-propio-bot-para-telegram/bot-en-python-para-telegram/

GOOGLE API DRIVE

https://drive.google.com/file/d/11u1l5UvAW7JbmbrBl3ibXB-j400B715R/view?usp=sharing

Welcome to PyDrive's documentation! — PyDrive 1.2.1 documentation (pythonhosted.org)

Quickstart — PyDrive 1.2.1 documentation (pythonhosted.org)

https://drive.google.com/file/d/1dbG0IplDf6LloHXE62GbKG9y6N5va7s6/view?usp=sharing

PI CAMERA & WEBCAM

https://geekytheory.com/tutorial-raspberry-pi-uso-de-picamera-con-python

raspivid - Raspberry Pi Documentation

raspistill - Raspberry Pi Documentation

https://www.raspberrypi.org/documentation/usage/camera/raspicam/timelapse.md

Cámara web en Raspberry Pi con FSWEBCAM y Python

Using a standard USB webcam

Tutorial Raspberry Pi - Uso de PiCamera con Python - Geeky Theory

https://www.mankier.com/1/fswebcamle

SENSOR PIR

https://github.com/raspberrypilearning/guides/blob/master/physical-computing/test-pir-python.md