**Camp Tecnològic de Drons**

**Juliol 2024**

****

|  |  |
| --- | --- |
|  | **Universitat Politècnica de Catalunya BarcelonaTech** |

**1. Display**

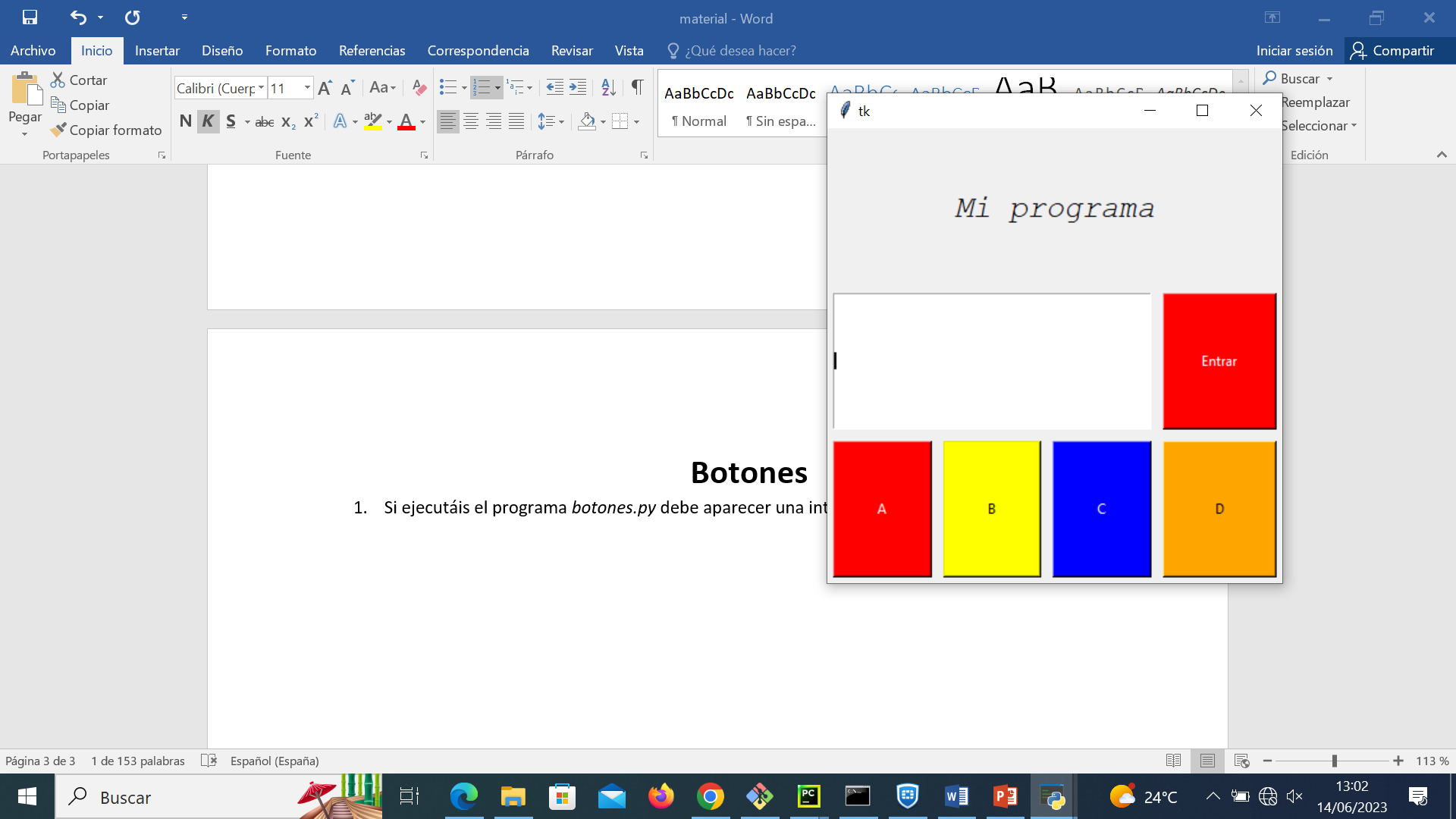
1. Probad el programa *display.py*

from djitellopy import Tello  
import time  
  
  
tello = Tello()  
tello.connect()  
print('BATERIA: ',tello.get\_battery())  
  
tello.send\_control\_command('EXT led 255 0 0')  
time.sleep(5)  
tello.send\_control\_command('EXT led 0 255 0')  
time.sleep(5)  
tello.send\_control\_command('EXT led 0 0 255')  
time.sleep(5)  
tello.send\_control\_command("EXT mled g 000000000rr00rr0rrrrrrrrrrrrrrrr0rrrrrr000rrrr00000rr00000000000")  
time.sleep(5)  
tello.send\_control\_command("EXT mled g 000bb00000bb00000bb00000bbbbbbbbbbbbbbbb0bb0000000bb0000000bb000")  
time.sleep(5)  
tello.send\_control\_command("EXT mled l r 2.5 HOLA")  
time.sleep(5)  
tello.send\_control\_command("EXT mled u g 2.5 0000b00bbb0b0b000b00b00000bb0000000b0000bbb00bbb000b0b0b0b00b0b0")  
time.sleep(5)  
tello.send\_control\_command('EXT mled s p I')  
time.sleep(5)

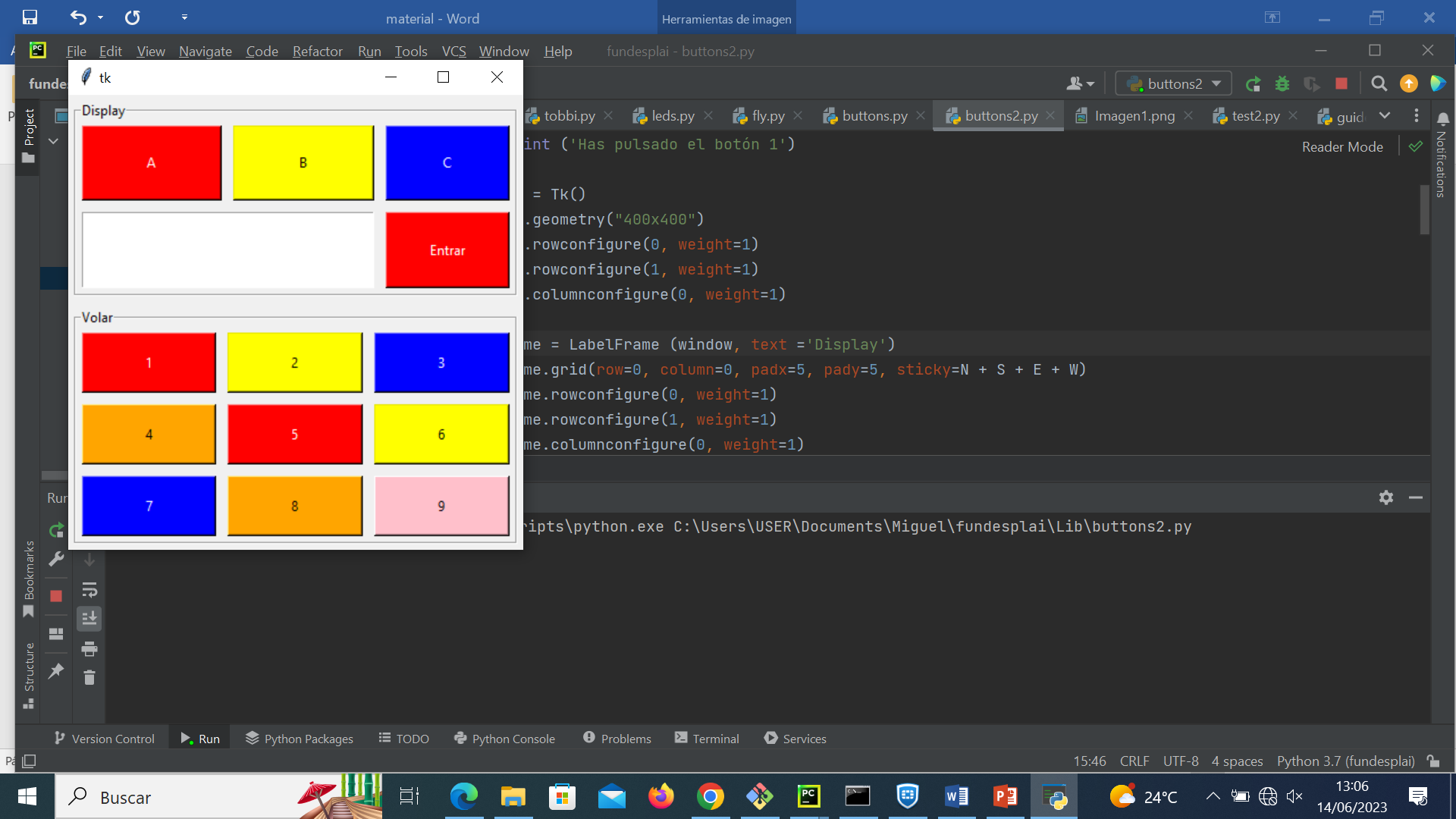
1. Diseñad un par de imágenes divertidas y elegid un nombre para el equipo. Luego, haced que se muestren en el display de vuestro dron.
2. **Volar**
3. Probad el programa *volar.py*

from djitellopy import Tello  
import time  
  
tello = Tello()  
tello.connect()  
print('BATERIA: ',tello.get\_battery())  
  
tello.takeoff()  
tello.move\_left(50)  
tello.move\_up(50)  
tello.rotate\_clockwise(90)  
tello.move\_forward(50)  
tello.land()

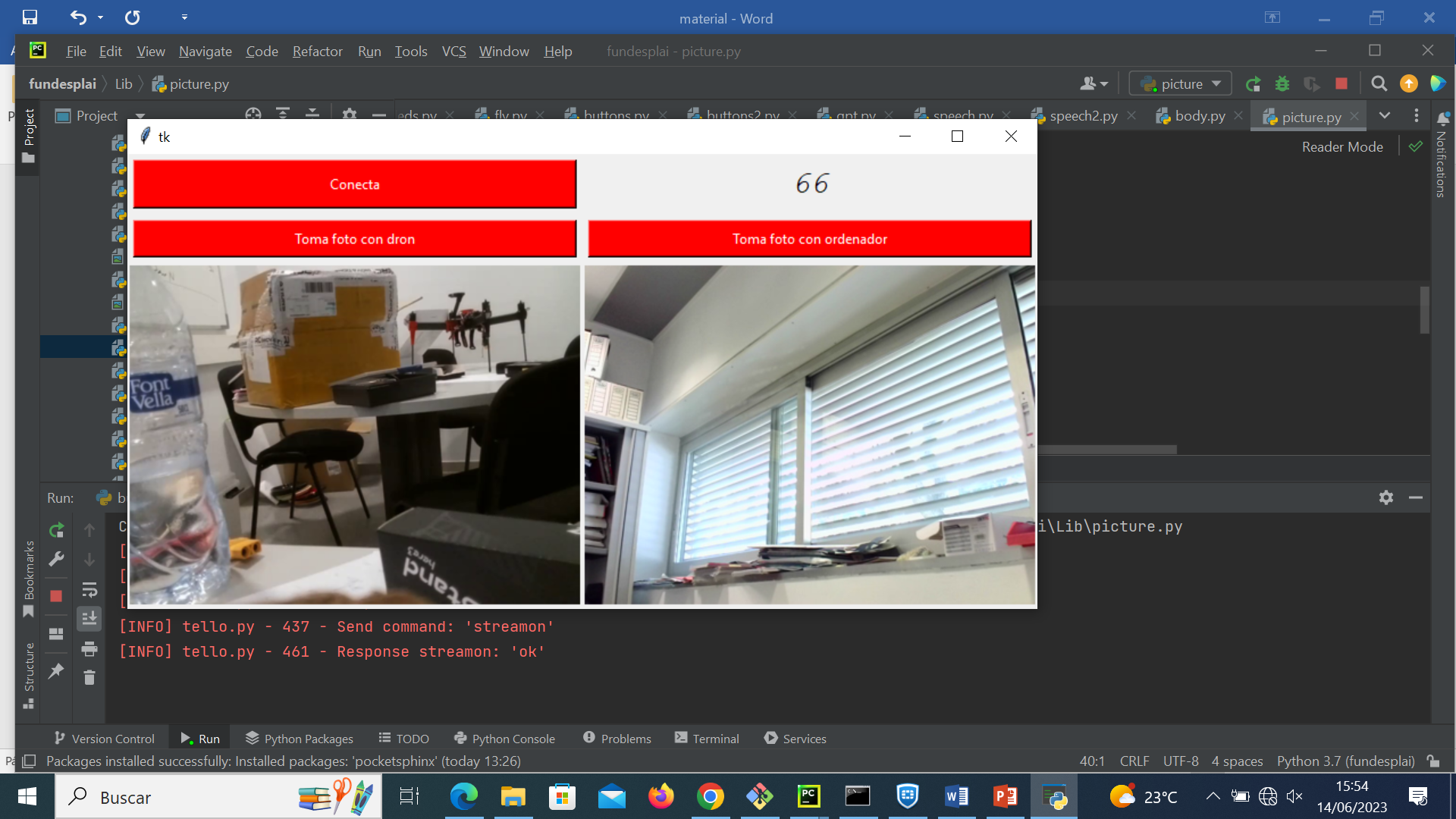
1. Investigad en internet qué otras funciones puede hacer la librería *djitellopy* y preparad un nuevo plan de vuelo a vuestro gusto.
2. **Botones**
3. Si ejecutáis el programa *botones.py* debe aparecer una interfaz gráfica como la de la figura.



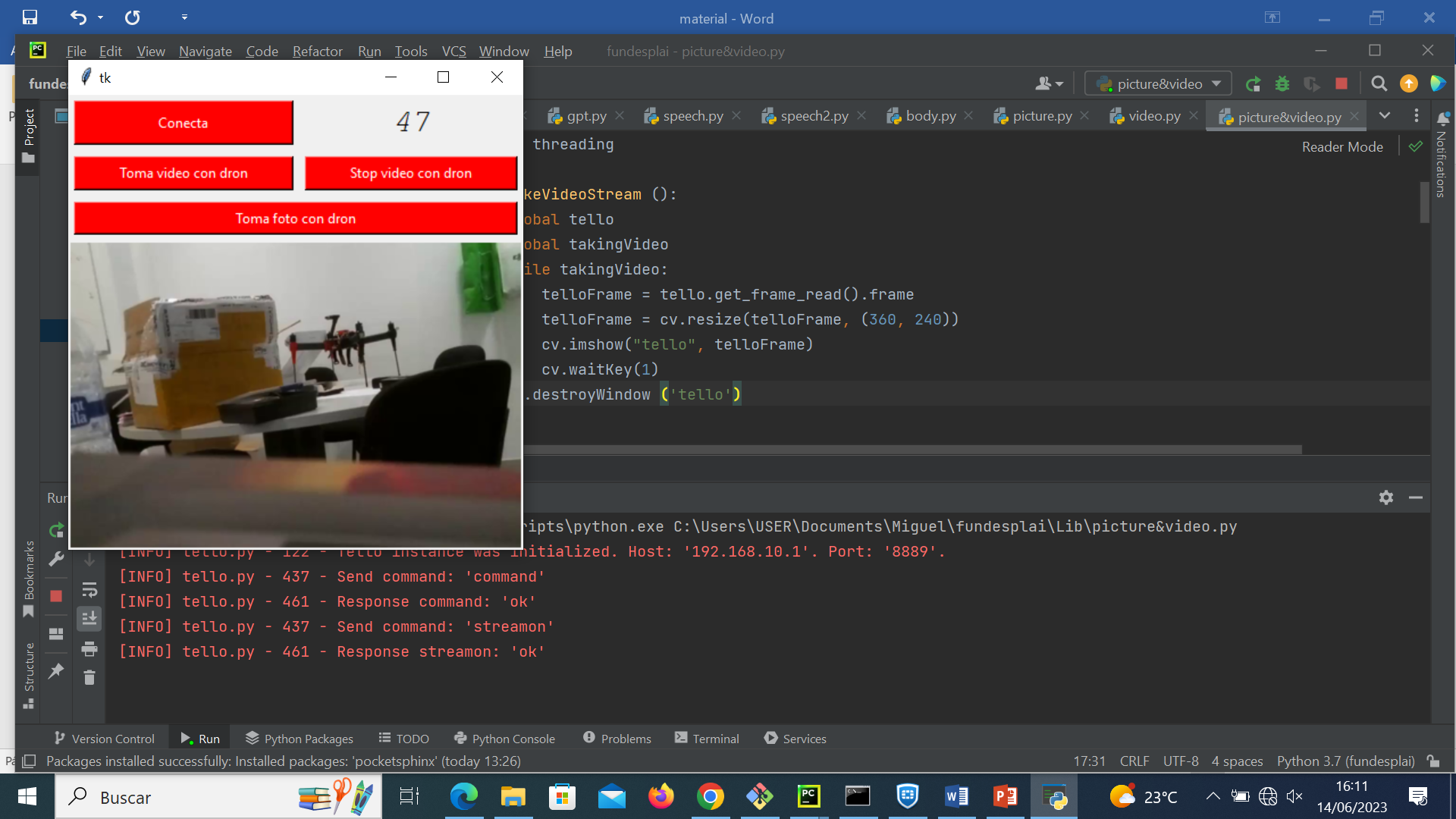
1. Tratad de comprender el código del programa y experimentad un poco con él (cambiad colores, poned y quitad botones, añadid funcionalidad a los botones).
2. El programa *botones2.py* os presenta la interfaz que se muestra en la figura de abajo. Introducid funcionalidad en los botones para mostrar cosas en el display del dron y para hacerlo volar usando comandos como los que habéis usado en los ejercicios anteriores.



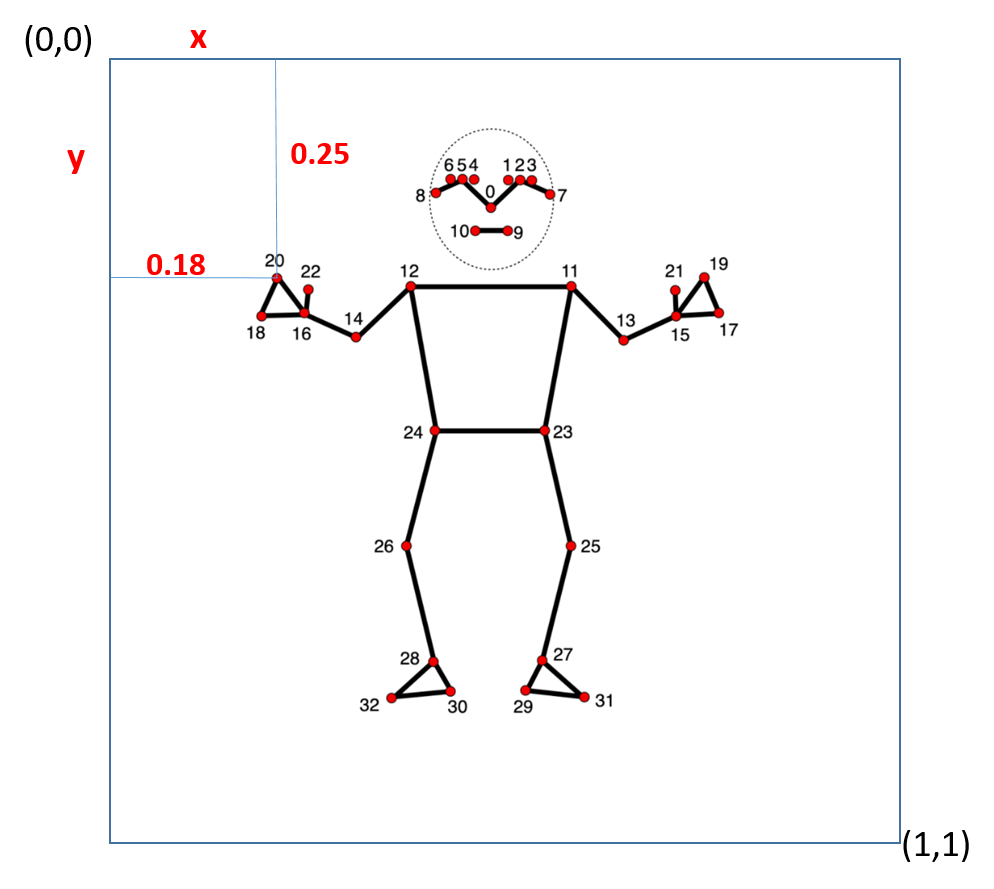
1. **Fotos y vídeos**
2. Conectad el dron y ejecutad el programa *fotos.py*, que tiene una interfaz gráfica como la de la figura. Con este programa os podéis conectar al dron (y ver cuánta batería le queda, en una escala de 0 a 100). Además, podéis pedir una foto al dron y una foto a la cámara del portátil.



1. Ejecutad ahora el programa *video.py*. Ahora vais a ver en una ventana el stream de video que os envía el dron y en otra el stream de video que os envía la cámara del portátil. Si observáis el código de ese programa veréis que para ver el vídeo lo único que hay que hacer es tomar fotos constantemente y mostrarlas. Recordad que el cine no es más que fotos que se muestran muy rápido una detrás de otra. Pero claro, si estamos tomando fotos todo el rato el programa no puedo hacer otra cosa, como, por ejemplo, atender al usuario cuando pulsa un botón para tomar una foto o para hacer volar el dron.
2. Ejecutad ahora el programa *fotos&video.py*, que tiene una interfaz como la de la figura. Con ese programa sí que podemos mostrar el video al mismo tiempo que atendemos a los botones, por ejemplo, para hacer una foto.



1. Añadid al programa un botón para hacer despegar al dron y otro para aterrizarlo, de manera que podáis tomar fotos y videos mientras está en el aire.
2. **Poses**
3. Ejecutad el programa *cuerpo.py* y observad lo que se ve en la pantalla. La librería de *Mediapipe* que usa el programa es capaz de detectar 33 puntos claves del cuerpo que está siendo enfocado por la cámara. Los puntos clave son los mostrados en la figura.

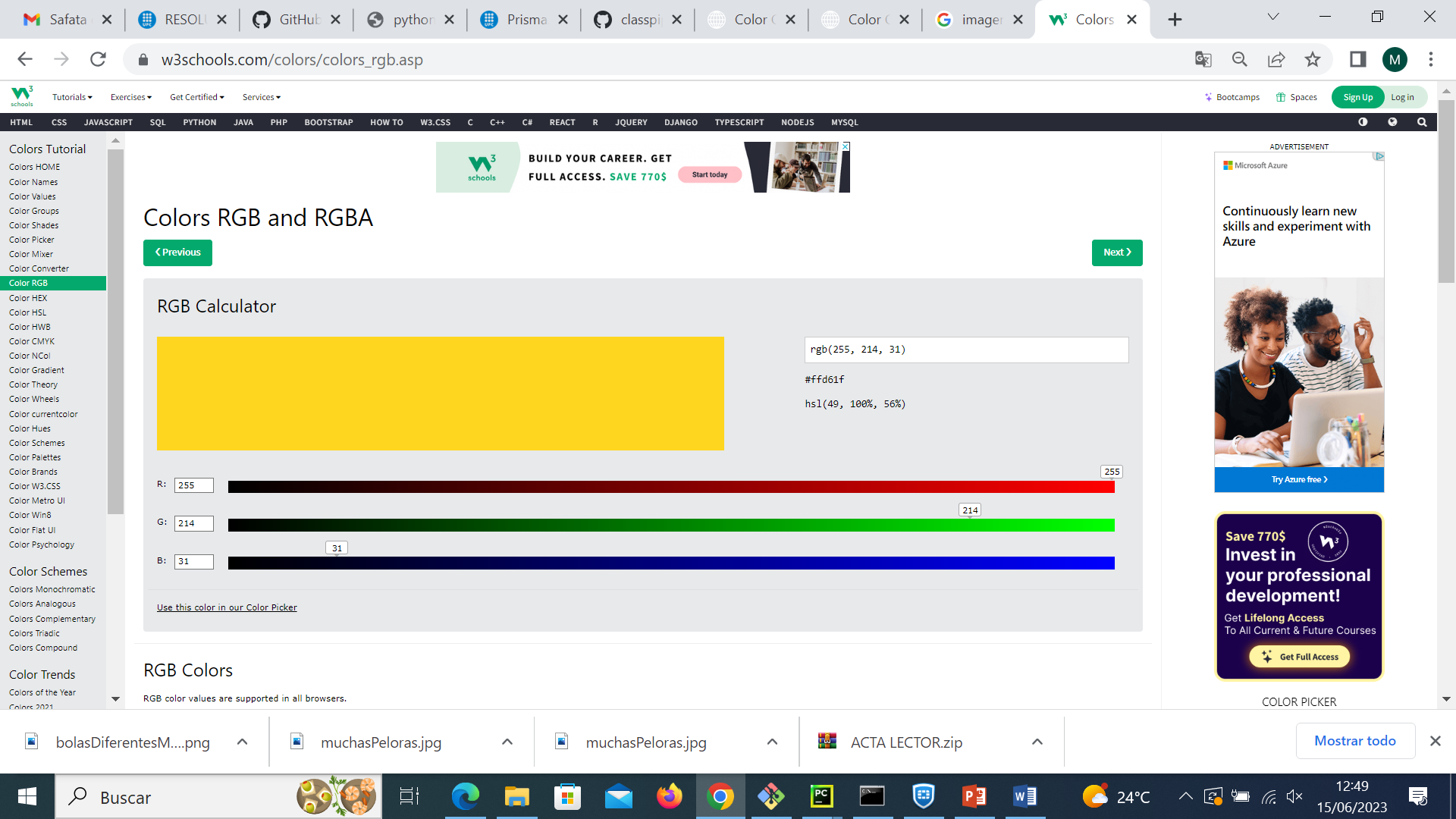


1. La función *prepareBody* nos devuelve una lista (que se llama *bodyLandmarks*) con los 33 puntos detectados en la imagen que le enviamos (la que acabamos de tomar con la cámara del portátil). Para cada uno de esos puntos en esa lista tenemos sus coordenadas cartesianas (x,y), en una escala entre 0 y 1. Por ejemplo, las coordenadas del punto 20 en la imagen serían (0.25, 0.18).
2. Añadid las siguientes instrucciones en el punto que os parezca apropiado para que el programa escriba en consola las coordenadas del punto 20 de la imagen.

print ('coordenada X del punto 20: ', bodyLandmarks[20][0])  
print ('coordenada y del punto 20: ', bodyLandmarks[20][1])

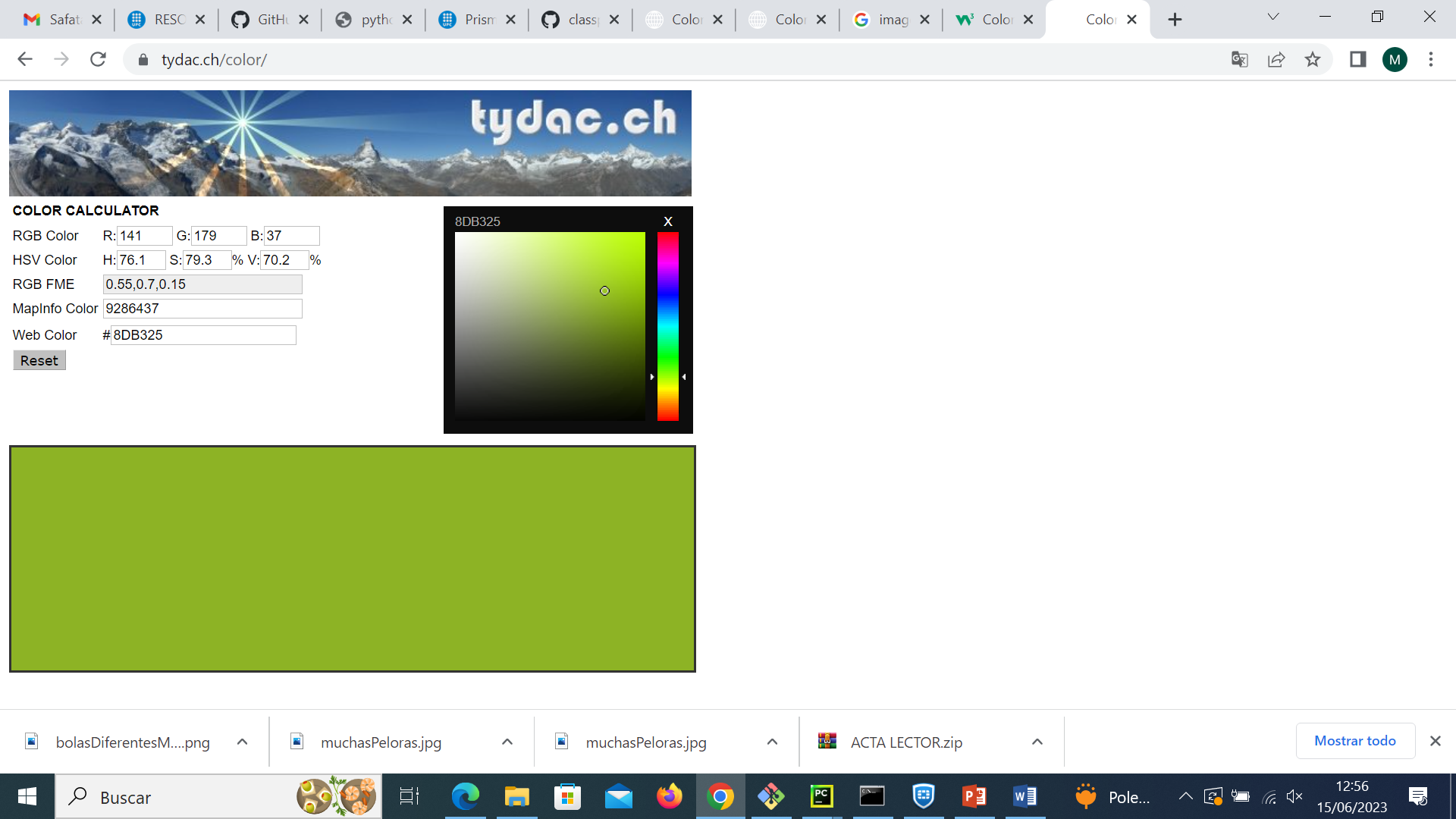
Aquí pasa algo raro ¿no?

1. Añadid ahora el código necesario para detectar si el brazo derecho está levantado (es decir, si el punto 13 está por encima del punto 11).
2. **Guiar con las poses**
3. Ejecutad el programa *tresPoses.py*. Este programa reconoce las tres poses de la imagen. Para ello, después de tomar una imagen la envía a una función que hace las consultas necesarias sobre las posiciones de los puntos para determinar cuál de las 3 poses tiene el cuerpo (solo se tienen en cuenta los puntos que se muestran en la figura).
4. Elegid tres poses más y completar la función para que detecte las tres poses nuevas además de las tres anteriores.
5. Añadir el código necesario para enviar al dron las ordenes que deseéis según la pose que se detecte.
6. **Colores**
7. Experimenta en esta página web cómo crear colores mezclando rojo, verde y azul (código RGB).



<https://www.w3schools.com/colors/colors_rgb.asp>

1. Usa el programa *RGB.py* para detectar el código RGB de los colores que aparecen en la foto que quieras (por ejemplo, *pelotas.png*). Comprueba que el programa no te da el código RGB sino el BGR.
2. Experimenta en esta página web cómo crear colores mediante el código HSV.

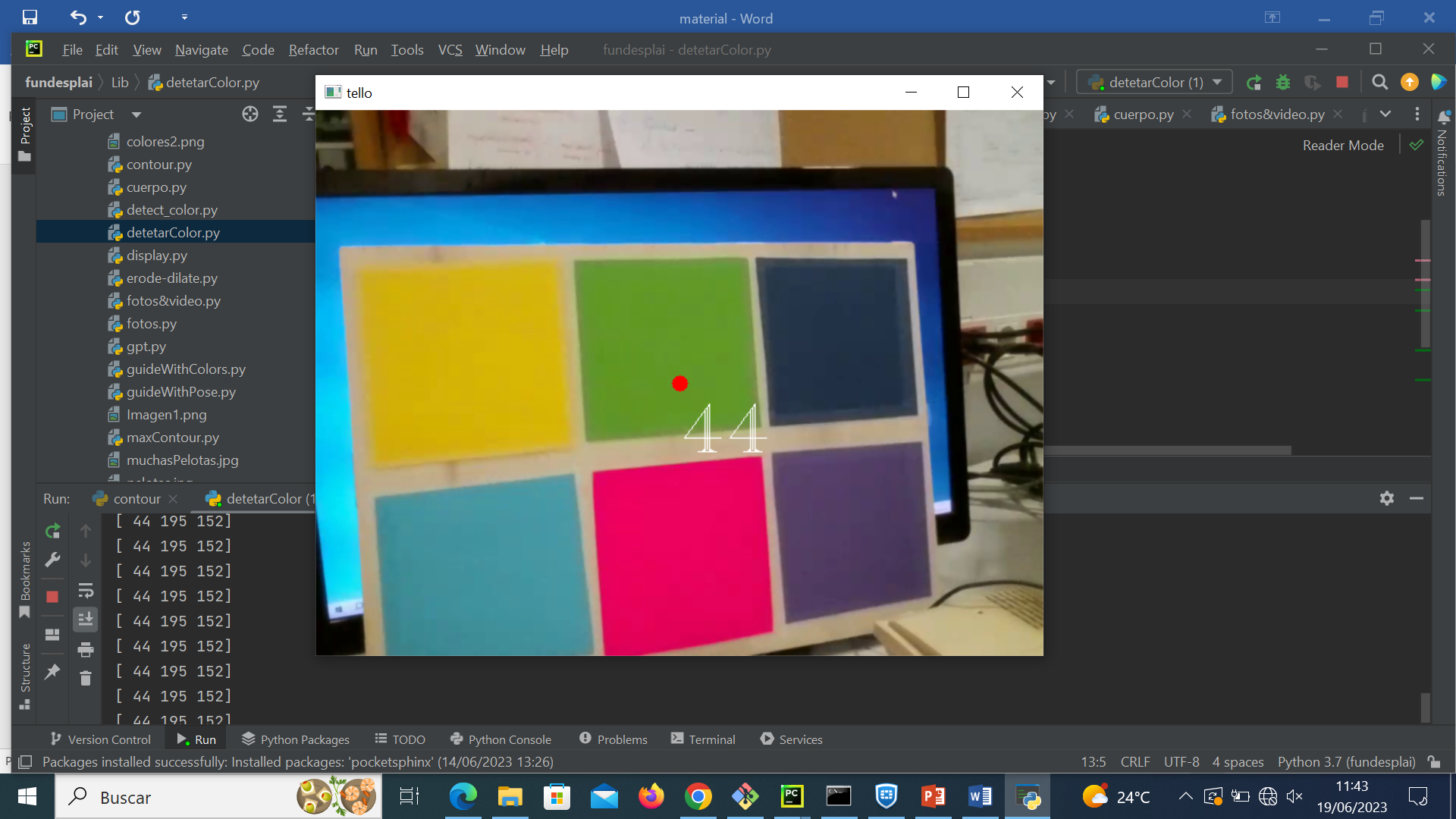


<https://www.tydac.ch/color/>

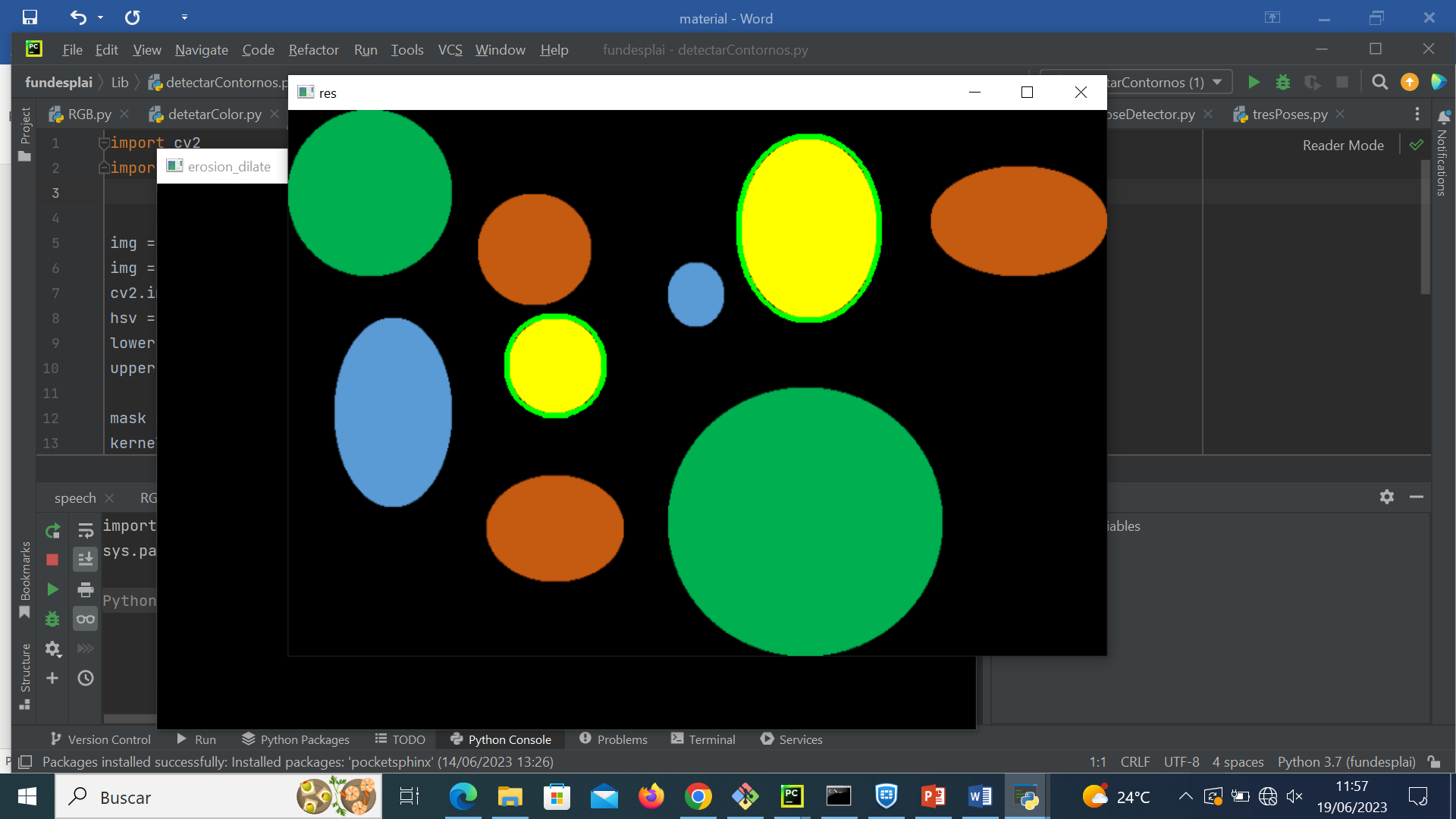
1. Añade al programa *RGB.py* las dos líneas de código siguientes donde te parezca oportuno para mostrar también el código HSV del color sobre el que se pulsa.

hsv = cv2.cvtColor (img, cv2.COLOR\_BGR2HSV)  
print ("(HSV)", hsv[y,x])

1. Usa la imagen *colores2.png* para ver las diferencias entre los dos códigos y determinar cuál es la codificación más adecuada para determinar si el color seleccionado es rojo, o verde o amarillo.
2. Usad el programa *detectarColor.py* para detectar el código HSV del color al que apunta la cámara del dron Tello, tal y como muestra la figura.



1. **Contornos**
2. Ejecutad el programa *detectarContornos.py* usando la figura *bolasDiferentesMedidas.png*. Se mostrará algo parecido a esta imagen. El programa es capaz de detectar las zonas de color amarillo y señalar su contorno. Modificar el programa para que haga lo mismo, pero con las zonas de cada uno de los colores restantes.



1. Haced el mismo ejercicio que en el punto 1, pero ahora con una imagen más realista como la que hay en *muchasPelotas.py*. Modificad el programa para detectar las pelotas de cada uno de los colores. Fijaos que el programa muestra 5 imágenes: img (la original), mask, erosion, erosion\_dilate y res (el resultado final). Los valores que se usan para la erosión y la dilatación son importantes para poder separar bolas que están muy juntas o para eliminar zonas que puedan llevar a confusión. Experimentad con esos parámetros según las instrucciones del profesor, para comprenter la importancia de cada una de esas operaciones con el fin de detectar correctamente los contornos. ¿Cómo haríamos ahora para que el programa cuente el número de bolas que hay de cada color?
2. El programa *bolaMasGrande.py* detecta y marca el contorno de la bola amarilla más grande de la imagen *bolasDiferentesMedidas.png.* Ejecutadlo y comprobadlo. Modificad ahora el programa para que os enmarque la bola más grande de todas (de entre las de los 4 colores que hay). Cread vosotros otras imágenes con otros colores usando PowerPoint y probad el correcto funcionamiento de vuestro programa.
3. **Guiar el dron con colores**
4. Se trata ahora de detectar colores en el stream de video que nos envía el dron mientras vuela. Para ello, basta aplicar el mismo procedimiento que hemos aplicado a una foto, pero ahora a cada frame del video que recibimos en streaming. El programa que necesitamos para eso usará buena parte del código que hay en *detectarColor.py* (que detecta el código del color apuntado por la cámara del dron) y una buena parte del código *bolaMasGrande.py*, que enmarca la bola de mayor tamaño. Tratad de construir ese programa que debe permitirnos detectar la bola más grande en el video stream que recibimos del dron. Observad que para poner a punto este programa no es necesario hacer volar el dron. Basta con conectarlo, poner en marcha la recepción de video, coger el dron con la mano y apuntar la cámara a donde queráis.
5. Probad vuestro programa, pero instalando ahora en el dron el espejo reflector. Ese espejo permitirá a la cámara captar la imagen del suelo y es ahí, en el suelo, donde el dron detectará las bolas de colores.
6. Finamente, incluir en vuestro programa las instrucciones para guiar al dron según el color que detecte. Por ejemplo, que se vaya hacia delante si pasa por encima de un objeto de color amarillo, o aterrice si detecta un objeto de color verde. Para esta última parte os ayudará entender la diferencia entre estas dos operaciones:

tello.move\_forward(50)

Esta operación mueve el dron hacia delante 50 cm. El programa se bloquea hasta que el dron está en el destino.

tello.send\_rc\_control(0, 50, 0, 0)

Esta operación pone al dron a volar a una velocidad de 50cm/segundo hacia delante, pero no bloquea el programa. El dron sigue volando a esa velocidad y en esa dirección hasta que le enviemos otro comando. Otros comandos similares son:

tello.send\_rc\_control(0, -50, 0, 0) #hacia atras

tello.send\_rc\_control(0, 0, 50, 0) #hacia arriba

tello.send\_rc\_control(-50, 0, 0, 0) #hacia a derecha