CARPETA DE CAMPO

ONIET-PROTOTIPOS II

Nombre del proyecto:

NEOCULI

Integrantes:

Abuin Leonel

Casella Camilo

Franco Esteban Neoren

Giacometi Mateo

Docente Responsable

Kuc Alejandro

NEOCULI

Identificación

Objetivos

Nombre del proyecto:
Neoculi

Integrantes:
Abuin *Leonel* – D.N.I. 44290906 – 7°A – E.E.S.T. N°6 San Nicolás de los Arroyos
Casella *Camilo* – D.N.I. 44380726 – 7°A – E.E.S.T. N°6 San Nicolás de los Arroyos
Franco *Esteban Neoren* – D.N.I. 44241147 – 7°A – E.E.S.T. N°6 San Nicolás de los Arroyos
Giacometti *Mateo* - D.N.I. 43983261 - 7°A – E.E.S.T. N°6 San Nicolás de los Arroyos

Foto del grupo:

Docentes responsables y/o tutores:
Kuc *Alejandro*

Fecha de inicio:
11/5/2021

Duración:
16 semanas

Esfuerzo en horas:
50 aproximadamente

Personas afectadas:
4 personas en un promedio de 3 Hs semanales.

Carpeta de Campo del proyecto

Objetivos:

● Generar un dispositivo que permita a las personas con ceguera completa o parcial, vincularse al mundo que los rodea, mediante las tecnologías de machine learning y computer vision.

● Diseñar un dispositivo capaz de reconocer los objetos que se muestran en un video en tiempo real.

● Programar un sistema que permita reproducir de manera auditiva el nombre de los objetos detectados.

Temáticas:

Inteligencia Artificial - Machine Learning - Computer Vision- Redes Neuronales - Deep Learning

Alcance (tanto social como geográfico):

Para todas las personas con discapacidad visual total o parcial (3,6% en Argentina y 0,7% mundial) en cualquier parte del mundo.

Segmento destino:

Para personas con ceguera visual tanto parcial como total y para adultos mayores.

Ámbito de incumbencia:

Diseño universal - Desplazamiento

Carpeta de campo del proyecto

Procedimiento

11/5/2021 Se conforma el grupo de trabajo.

Se piensa qué prototipo podríamos hacer. Comenzamos a idear el proyecto y se plantea de forma superficial cómo lo podríamos realizar.

Comenzamos a repartirnos tareas en base a los conocimientos y habilidades/fortalezas de cada uno de los integrantes de nuestro equipo:

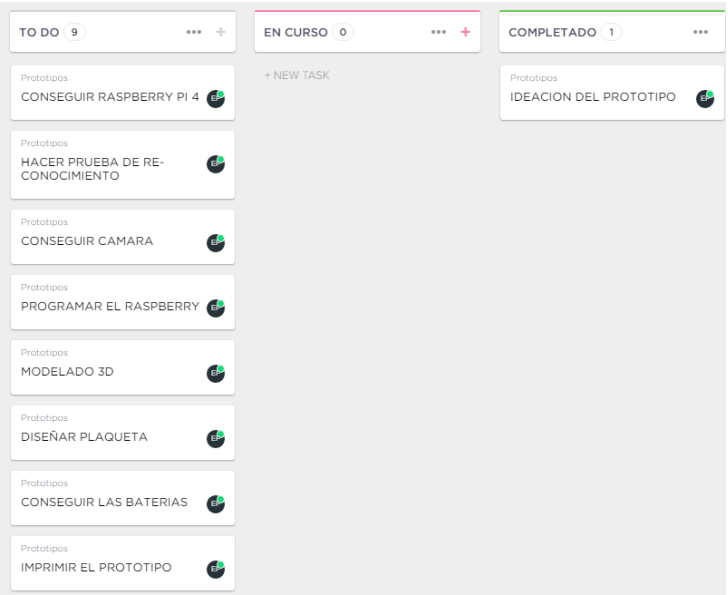
Abuin Leonel---- Programación en Python y conocimiento en base de datos.

Casela Camilo---Diseño en 3D y programación en Python

Franco Esteban---Conocimiento en programar tarjetas ESP-8266, Arduino IDE, Project Management y coordinador del proyecto.

Giacometi Mateo---Programación web y conocimiento en base de datos.

Para llevar a cabo el Project Management se utilizó una aplicación llamada ClickUp en la cual se escribieron las tareas a hacer y en qué estado estaba cada tarea. Esto es útil para administrar las tareas y asignarlas a cada integrante del grupo, reduciendo tiempo y haciendo más eficiente el trabajo.



18/5/2021 Análisis y búsqueda de información del uso de Machine Learning, redes neuronales y Computer Vision en Raspberry Pi, en las siguientes páginas web:

https://www.tensorflow.org/?hl=es-419

https://www.youtube.com/watch?v=szNPBn_RBfA&list=LL&index=2

https://www.youtube.com/watch?v=6_2hzRopPbQ&list=LL&index=6

https://www.youtube.com/watch?v=iKQC4oCvSXU&list=LL&index=23

https://www.youtube.com/watch?v=0m387MkOyWw&list=LL&index=24

Optamos por usar la librería de Google llamada Tensor Flow para llevar a cabo el reconocimiento de objetos por su versión de Machine Learning subida en 2020 llamada Tensor Flow Lite la cual nos beneficia por su ligereza a comparación de OpenCV y su versión normal.

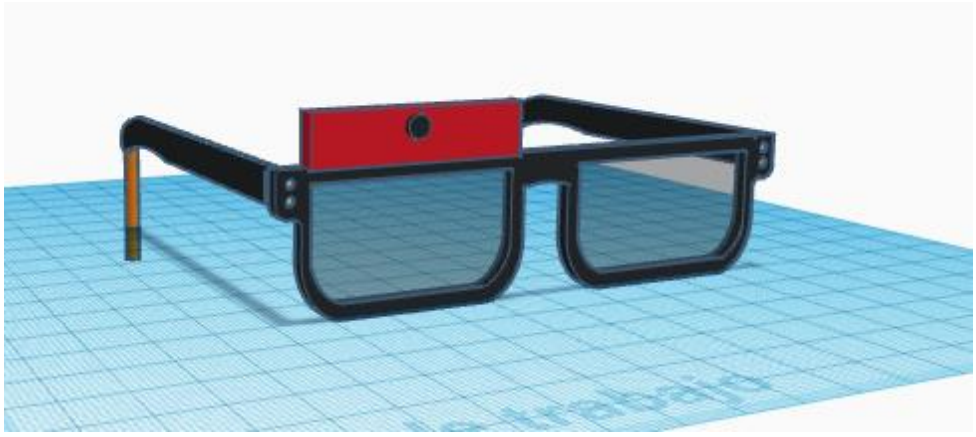
La escuela nos proporcionó un Raspberry Pi 3 (1GB RAM). Este se nos fue dado para no posponer el proyecto porque se anticipa unas semanas de cuarentena en nuestra ciudad.

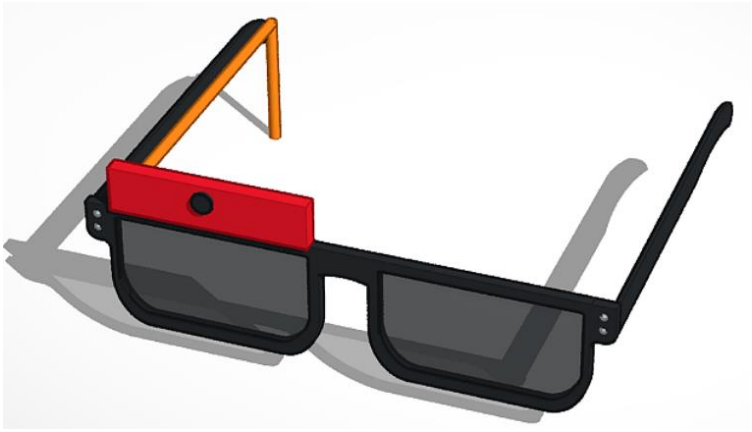
Elaboramos una lista de materiales los cuales usaremos para el montaje final.

Lista de materiales:

- Raspberry Pi
- Cámara USB (la cual reciclamos de una laptop)
- Filamento de PLA
- Cable USB
- Baterías 18650 con su módulo de carga
- Conector Micro-USB hembra (para una versión 2.0 del producto)

Pensamos e ideamos modelos en 3d por medio de la plataforma Tinkercad para visualizar nuestra idea del resultado final del prototipo.





Este modelo está adjuntado en Drive en la carpeta de Modelos 3D en forma de archivo .obj

22/5/2021 Averiguamos los precios de cada insumo material en la página de MercadoLibre

https://www.mercadolibre.com.ar/

Detalles en el Anexo de Costos de insumos materiales

25/5/2021 Gracias al Raspberry Pi proporcionado por la escuela, se descargó Raspbian en una tarjeta microSD y codificamos la primera prueba de reconocimiento de imágenes, en la cual logramos que el programa reconozca en un video la presencia de un pájaro.
Imagen captada de la imagen de la computadora:

El código utilizado y su explicación se encuentran en el Anexo de Códigos con el nombre de Prueba n°1

11/6/2021 Por un problema de un aumento de restricciones en nuestra ciudad, no pudimos reunirnos para continuar el prototipo, pero hacemos una reunión virtual en la que definimos en lo que se va a enfocar cada integrante del equipo.

Además, escribimos el código final del programa, el cual no pudimos probar ya que, por las cuestiones anteriormente mencionadas, los locales de informática están cerrados, lo que nos impide conseguir una cámara para realizar una prueba del proyecto.

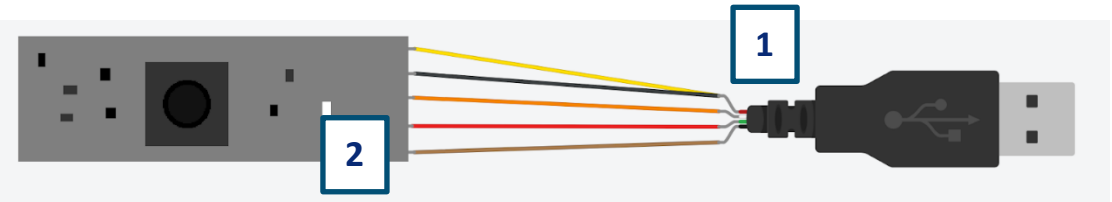
16/6/2021 Comenzamos a diseñar una página web del proyecto y definimos un nombre para el grupo “NEOCULI”

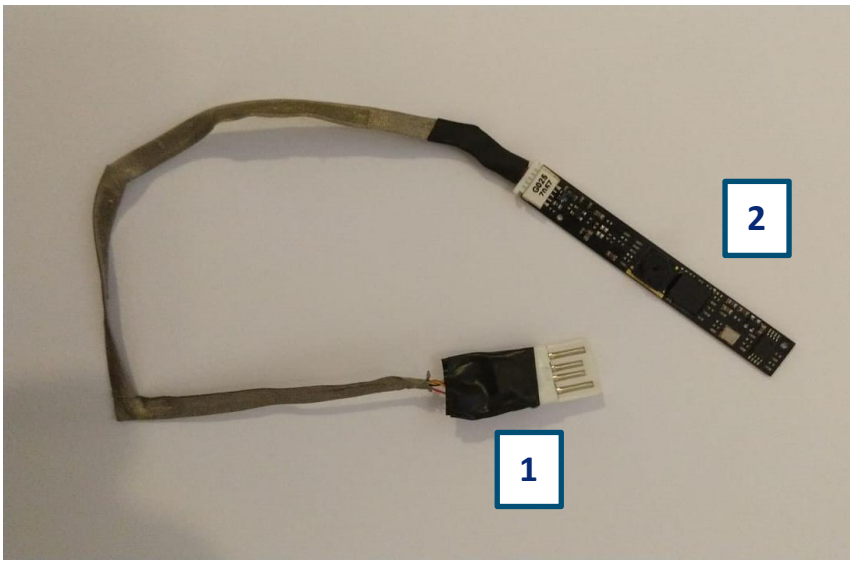
https://prototipoeestn6.wixsite.com/oculi-1

24/6/2021 Nos juntamos para testear el código y vinculamos actividades.

Logramos obtener una cámara web reciclada de una laptop, la cual adaptaremos para que funcione por USB.

02/7/2021 De vuelta en la escuela, utilizamos la cámara reciclada de una laptop (modelo G1) a la cual le soldamos a la salida de sus cables un cable USB con la siguiente conexión:

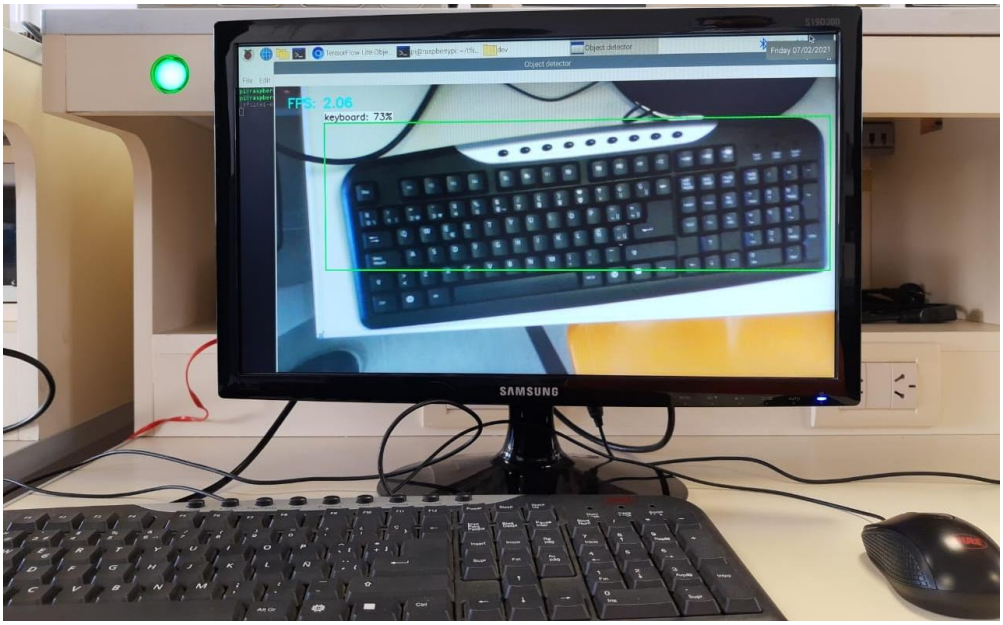




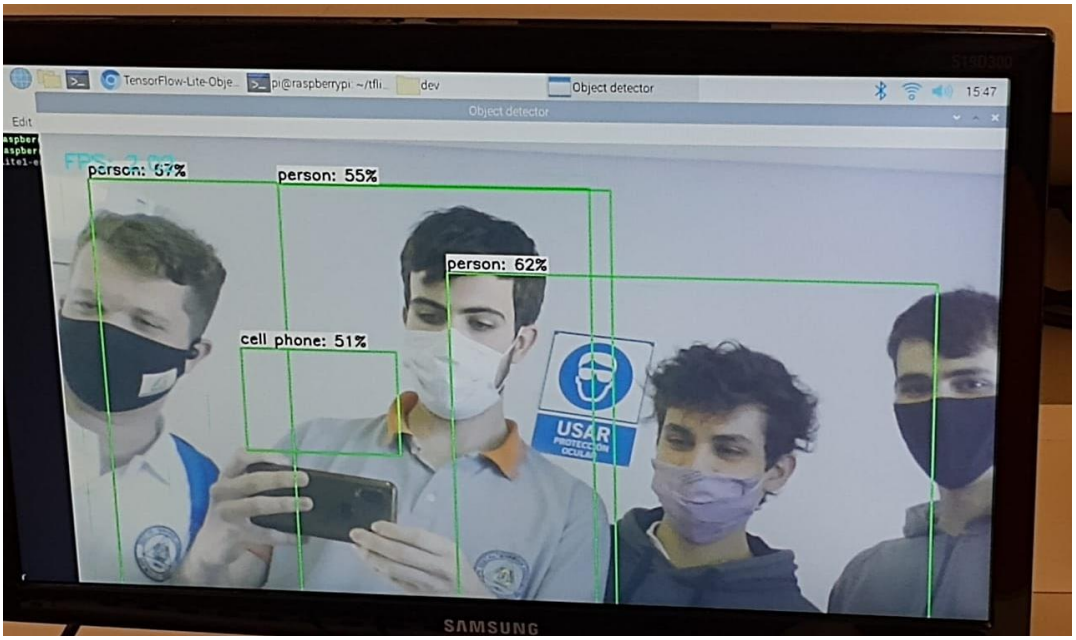
1- USB tipo A
2- Cámara Web

Hacemos una prueba de reconocimiento en tiempo real a partir de que la cámara nos funciona. Para esto ejecutamos el código de nombre prueba 2 que se puede encontrar en el Anexo de códigos.

Este puedo reconocer los objetos que eran capturados por la cámara USB anteriormente nombrada. Y reconoce tanto objetos como personas, como podemos ver en las siguientes imágenes que capturamos de la pantalla de nuestra computadora:



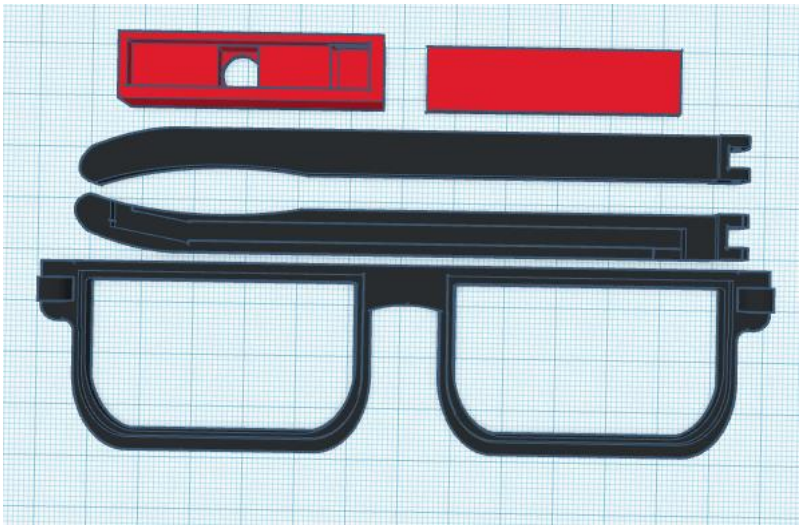
Reconocimiento de un Teclado



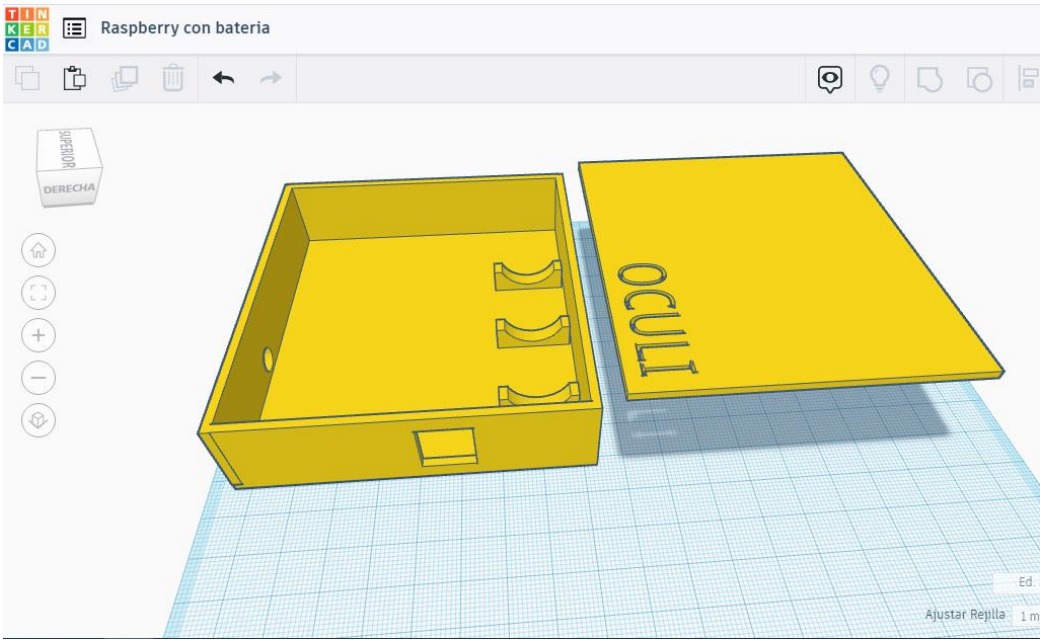
Reconocimiento de personas

6/7/2021 A la cámara que conseguimos le quitamos los plásticos que la cubría y la malla que cubría sus cables, para luego utilizar un termo contraíble para que sus cables ocupen el menor espacio posible, con el objetivo de que estos quepan bien en la “patita” de los lentes.

Además, diseñamos el modelo final del prototipo tanto los lentes como la “caja” donde se ubicará el Raspberry Pi.

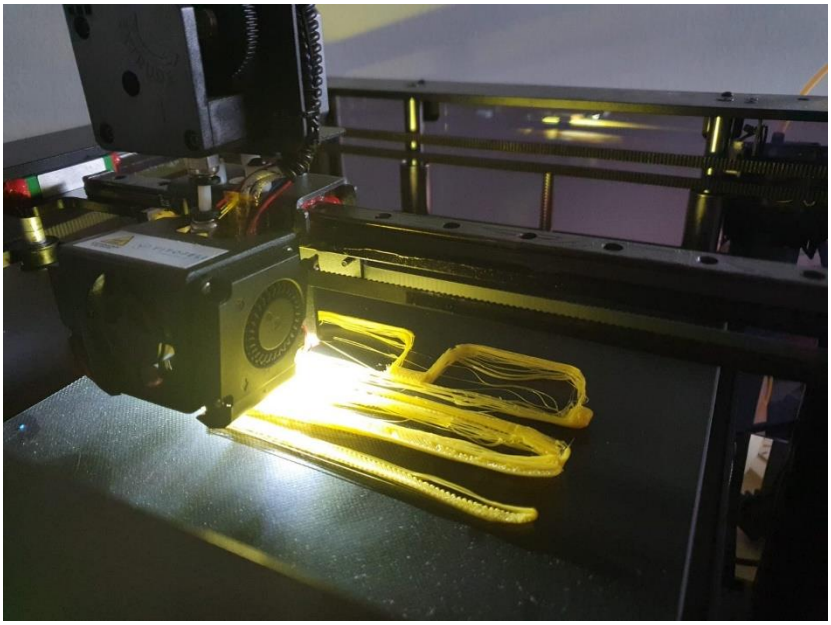


Este diseño se puede encontrar en el drive como Modelo final de los lentes.zip en la carpeta de Modelos 3D.



Este diseño de caja de contención de la placa se puede encontrar en el drive, como Modelo final caja.zip en la carpeta de Modelos 3D.

7/7/2021 Imprimimos el modelo “Modelo final de los lentes.zip”. Siendo la primera vez que imprimimos en 3D y del desconocimiento de ciertos parámetros como el Infill (relleno) y la falta del uso de un aerosol adhesivo, la impresión no resultó como lo esperábamos. Estos errores resultaron en un modelo de mala calidad, ya que el relleno estaba en un 20% formando un objeto en forma de panal de abeja en vez de un objeto sólido, y este se empezó a despegar de la base de impresión lo que provocó que el filamento no se coloque donde debía.

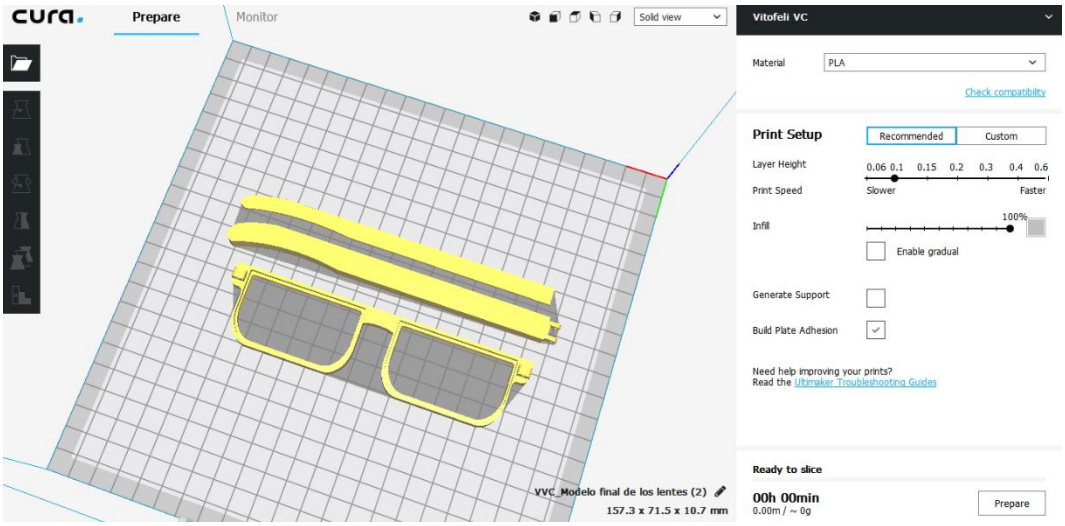


El video de la impresión lo podemos encontrar en la carpeta de Modelos 3D - Videos con el nombre de "Prueba impresión fallida.mp4"

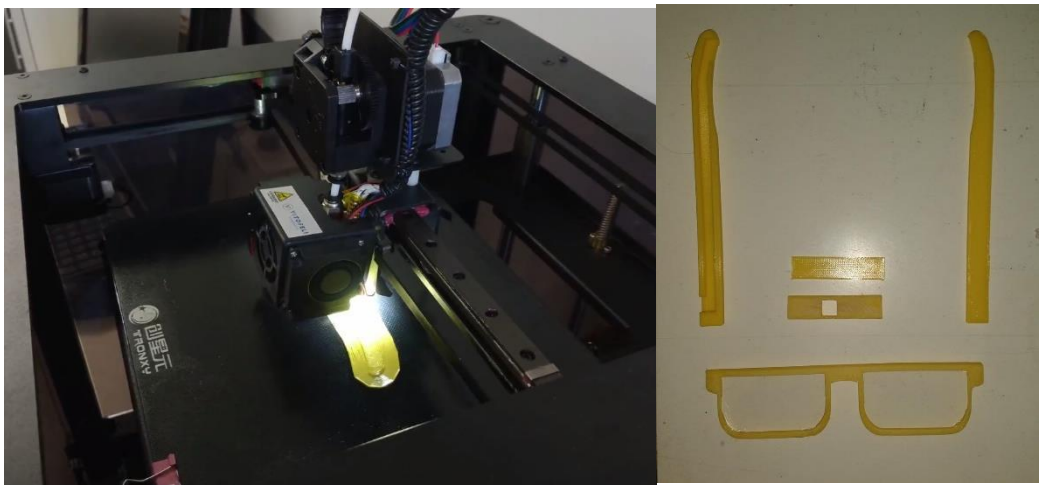
11/7/2021 Ingresamos en la página web de la empresa que fabrica las impresoras 3D que utilizamos llamada "Vitofeli" para descargar un software especial para imprimir en esas impresoras. En este encontramos una adaptación de Cura (un programa para configurar impresoras 3D) para las Impresoras de esta empresa.

https://www.vitofeli.com.ar/es/

Al descargarlo, configuramos los parámetros de la impresión para evitar los errores que cometimos anteriormente.

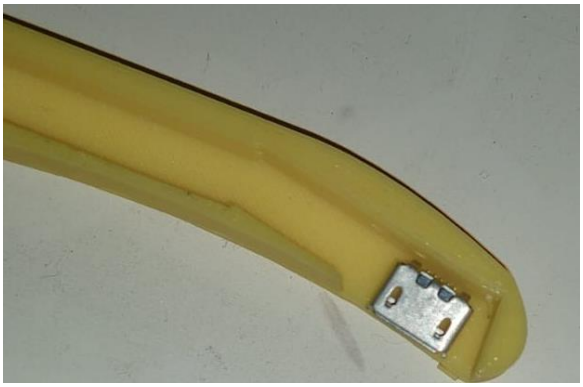


Además, al imprimir nos dimos cuenta de que los resultados mejoraron notablemente con el uso de un aerosol para cabello como pegamento en spray, el cual aplicamos antes de que se apoye el filamento en la camilla.



El video de la impresión lo podemos encontrar en la carpeta de Modelos 3D - Videos con el nombre de “Impresión exitosa de la patilla.mp4”

26/7/2021 Compramos un conector Micro-USB hembra para que se pueda usar un cable Micro-USB a USB tipo A (cable que se utiliza en celulares y demás dispositivos) para conectar la placa (Raspberry Pi) a los lentes. La siguiente imagen muestra la ubicación del conector Micro-USB de los lentes.



Para conectarlo utilizaremos la siguiente conexión, siendo el “B DEVICE” el conector USB tipo A y el “A DEVICE” el conector micro USB hembra:

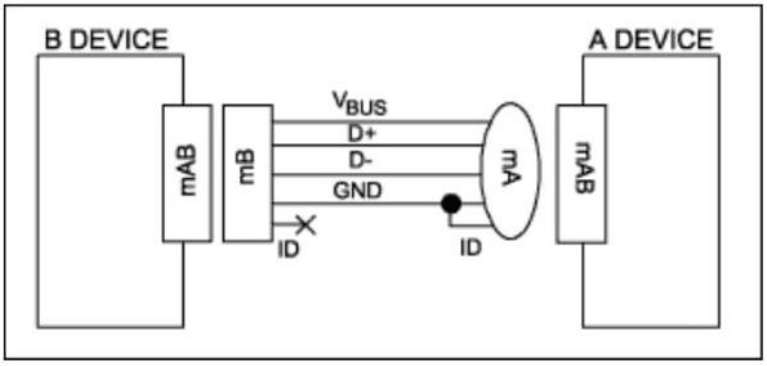


Imagen de conexiones entre Micro USB hembra y USB tipo A

19/8/2021 Pegamos con soldadura plástica cada pieza a excepción de la cámara y quitamos los excedentes por medio de una lija y organizamos para comprar un aerosol negro, con la idea de pintarlo próximamente.

24/8/2021 Empezamos con la impresión de la carcasa donde se ejecutará el procesamiento de imágenes.
Imprimimos la tapa de la carcasa la cual tiene un encastre que le permite ser colocada en el borde de un pantalón al igual que se hacía con los walkmans. Debido a que su impresión fue de dudosa calidad tendremos que volver a imprimir

25/8/2021 Compramos un aerosol negro y pintamos los lentes con el aerosol.

El video del prototipo siendo pintado lo podemos encontrar en la carpeta de Videos con el nombre de "Pintando la pieza.mp4"

26/8/2021 Una vez pintada la cámara y los lentes, pegamos otra vez los plásticos que simulan los vidrios de los anteojos y pegamos la cámara. En la parte superior de los anteojos:

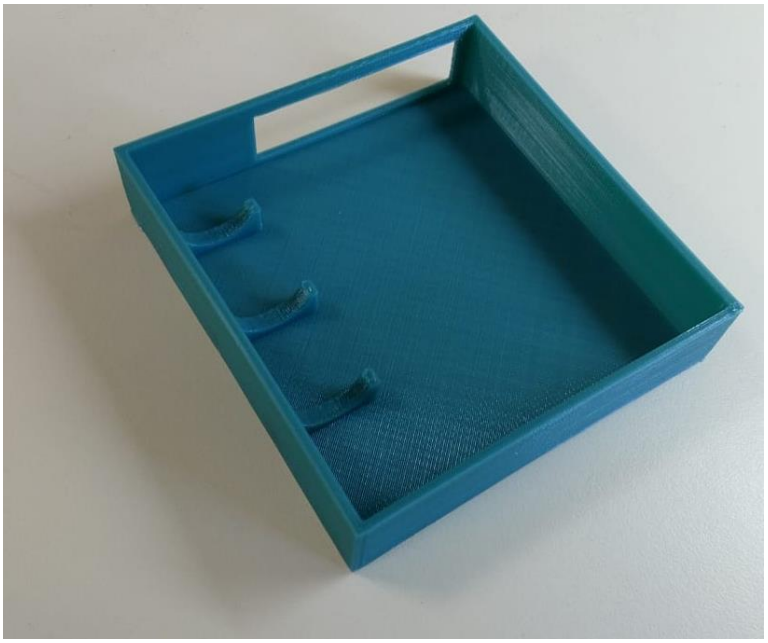
Además, con el diseño del circuito del conector Micro-USB hembra traspasamos este, a una placa de cobre, mediante el método de planchado.

Al final, la opción de poner un MicroUSB la pospusimos para un próximo modelo del prototipo debido al tiempo que se tarda en editar el modelo de los lentes para que entre la plaqueta correctamente.

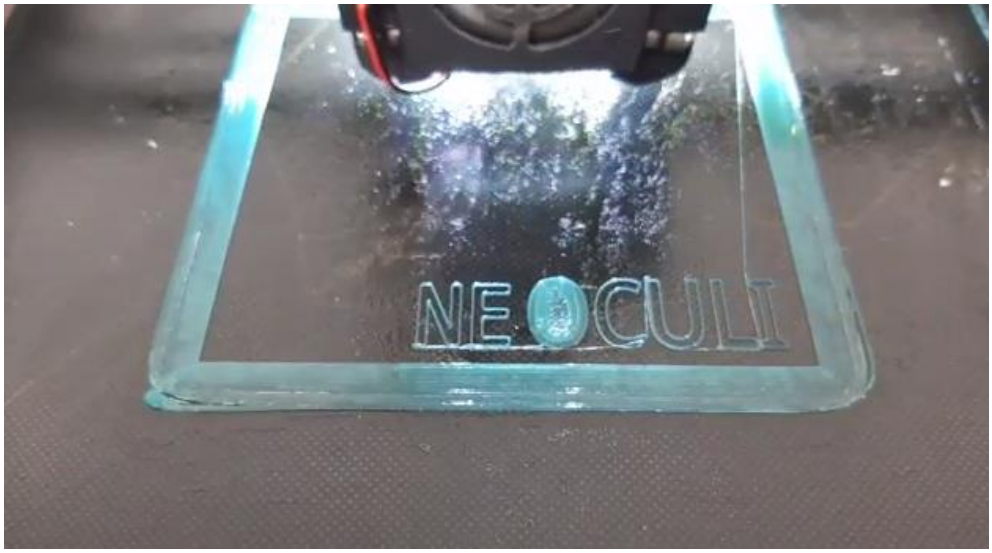
Imagen de placa para el conector MicroUSB hembra

27/8/2021 Pudimos probar el código final y este nos funcionó correctamente. Además, imprimimos la carcasa del prototipo.

El video de la prueba del prototipo lo podemos encontrar en la carpeta de Videos con el nombre de "Prueba final exitosa.mp4" y el código ejecutado lo podemos encontrar en el Anexo de códigos como "Prueba 3"



Carcasa del Raspberry Pi



Video de la impresión disponible en el Drive del proyecto, en la carpeta "Videos". Con el nombre de "Impresión de Neoculi"

30/8/2021 Realizamos un video explicativo del proyecto y lo subimos a YouTube con el siguiente enlace:

NEOCULI (https://www.youtube.com/watch?v=bfHsYO-KhBU&t=3s)

5/9/2021 Comenzamos a realizar los documentos necesarios para la entrega del trabajo.

20/10/2021 Terminamos de realizar los documentos necesarios para la entrega del trabajo.

Anexo de Costos de insumos materiales

Raspberry Pi v3 $15820

https://articulo.mercadolibre.com.ar/MLA-720304557-raspberry-pi-3-b-plus-element14-originales-placa-board-sbc-_JM#position=8&search_layout=grid&type=item&tracking_id=8215c4e8-ad32-425b-9b4a-2cd9cdd7adac

Camara de laptop reciclada $300

https://articulo.mercadolibre.com.ar/MLA-920024799-notebook-dell-studio-1535-pp33l-para-repuestos-camara-_JM?searchVariation=84195697353#searchVariation=84195697353&position=45&search_layout=stack&type=item&tracking_id=4d021b56-0e94-4cf1-b5f4-f97c75155b92

Filamento de PLA $1200

https://articulo.mercadolibre.com.ar/MLA-829172933-filamento-pla-impresora-3d-gst-175mm-x1-kg-320mt-_JM?searchVariation=47565076315#searchVariation=47565076315&position=33&search_layout=stack&type=item&tracking_id=73d891ab-1be3-40ec-a256-2797203db28a

Baterias 18650 $500

https://articulo.mercadolibre.com.ar/MLA-621014283-pila-bateria-recargable-26650-37v-7200mah-mejor-que-18650-_JM#position=7&search_layout=stack&type=item&tracking_id=0b1e0792-0d58-4610-8d10-804532dfd7bc

Módulo Step-Up $300

https://articulo.mercadolibre.com.ar/MLA-630807199-fuente-step-up-xl6009-dc-dc-ajustable-5v-35v-3a-max-arduino-_JM#position=1&search_layout=grid&type=item&tracking_id=8e08e0da-e9b7-4c32-a794-05ec8272cec5

Cable USB $190

https://articulo.mercadolibre.com.ar/MLA-910791635-cable-micro-usb-tipo-b-carga-rapida-1m-android-cargador-dato-_JM?searchVariation=78054148356#searchVariation=78054148356&position=11&search_layout=stack&type=item&tracking_id=984b6ce0-1d29-4a25-b0fb-8b9b4cfda955

Módulo de carga de baterias 18650 $197

https://articulo.mercadolibre.com.ar/MLA-853545584-modulo-cargador-tp4056-micro-usb-5v-18650-con-proteccion-_JM#position=6&search_layout=stack&type=item&tracking_id=b2d4d96e-2ea0-45e6-b467-74ef49d70cd0

Anexo de códigos

Prueba N°1

En este código mediante el cmd de Raspbian el sistema operativo que utilizamos en la plaqueta Raspberry. Este programa nos permite descargar las librerías de python de Tensor Flow y la red neuronal que vamos a utilizar para nuestro prototipo. Además, mediante este código pudimos ejecutar un programa de reconocimiento de objetos, en un video de ejemplo.

Programa de cmd:

```
1. git clone https://github.com/EdjeElectronics/TensorFlow-Lite-Object-
   Detection-on-Android-and-Raspberry-Pi.git
2. mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1
3. cd tflite1
4. sudo pip3 install virtualenv
5. sudo pip3 install virtualenv
6. python3 -m venv tflite1-env
7. source tflite1-env/bin/activate
8. bash get_pi_requirements.sh
9. wget
   https://storage.googleapis.com/download.tensorflow.org/models/tflite/coco
   _ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
10.     unzip coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip -d
   Sample_TFLite_model
11.     python3 TFLite_detection_video.py --video=test.mp4
```

Programa de phyton:

```
1. import os
2. import argparse
3. import cv2
4. import numpy as np
5. import sys
6. import importlib.util
7. parser = argparse.ArgumentParser()
8. parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
9. required=True)
10.     parser.add_argument('--graph', help='Name of the .tflite file, if
   different than detect.tflite',
11.     default='detect.tflite')
12.     parser.add_argument('--labels', help='Name of the labelmap file, if
   different than labelmap.txt',
13.     default='labelmap.txt')
14.     parser.add_argument('--threshold', help='Minimum confidence threshold for
   displaying detected objects',
15.     default=0.5)
16.     parser.add_argument('--video', help='Name of the video file',
17.     default='test.mp4')
```

```
18.      parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to
   speed up detection',
19.      action='store_true')
20.      args = parser.parse_args()
21.      MODEL_NAME = args.modeldir
22.      GRAPH_NAME = args.graph
23.      LABELMAP_NAME = args.labels
24.      VIDEO_NAME = args.video
25.      min_conf_threshold = float(args.threshold)
26.      use_TPU = args.edgetpu
27.      if pkg:
28.          from tflite_runtime.interpreter import Interpreter
29.          if use_TPU:
30.              from tflite_runtime.interpreter import load_delegate
31.      else:
32.          from tensorflow.lite.python.interpreter import Interpreter
33.      if (GRAPH_NAME == 'detect.tflite'):
34.      GRAPH_NAME = 'edgetpu.tflite'
35.      CWD_PATH = os.getcwd()
36.      VIDEO_PATH = os.path.join(CWD_PATH,VIDEO_NAME)
37.      PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)
38.      PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)
39.      with open(PATH_TO_LABELS, 'r') as f:
40.      labels = [line.strip() for line in f.readlines()]
41.      if labels[0] == '???':
42.       del(labels[0])
43.      interpreter = Interpreter(model_path=PATH_TO_CKPT,
44.      experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
45.          print(PATH_TO_CKPT)
46.      else:
47.          interpreter = Interpreter(model_path=PATH_TO_CKPT)
48.      interpreter.allocate_tensors()
49.      input_details = interpreter.get_input_details()
50.      output_details = interpreter.get_output_details()
51.      height = input_details[0]['shape'][1]
52.      width = input_details[0]['shape'][2]
53.      floating_model = (input_details[0]['dtype'] == np.float32)
54.      input_mean = 127.5
55.      input_std = 127.5
56.      video = cv2.VideoCapture(VIDEO_PATH)
57.      imW = video.get(cv2.CAP_PROP_FRAME_WIDTH)
58.      imH = video.get(cv2.CAP_PROP_FRAME_HEIGHT)
59.      while(video.isOpened()):
60.          ret, frame = video.read()
61.          if not ret:
62.            print('Reached the end of the video!')
63.            break
64.          frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
65.          frame_resized = cv2.resize(frame_rgb, (width, height))
66.          input_data = np.expand_dims(frame_resized, axis=0)
67.
68.          if floating_model:
69.              input_data = (np.float32(input_data) - input_mean) / input_std
70.         interpreter.set_tensor(input_details[0]['index'],input_data)
71.          interpreter.invoke()
72.          boxes = interpreter.get_tensor(output_details[0]['index'])[0] #
   Bounding box coordinates of detected objects
```


```
73.          classes = interpreter.get_tensor(output_details[1]['index'])[0] #
   Class index of detected objects
74.          scores = interpreter.get_tensor(output_details[2]['index'])[0] #
   Confidence of detected objects
75.          for i in range(len(scores)):
76.              if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):
77.                  ymin = int(max(1,(boxes[i][0] * imH)))
78.                  xmin = int(max(1,(boxes[i][1] * imW)))
79.                  ymax = int(min(imH,(boxes[i][2] * imH)))
80.                  xmax = int(min(imW,(boxes[i][3] * imW)))
81.
82.                  cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0),
   4)
83.                  object_name = labels[int(classes[i])] # Look up object name
   from "labels" array using class index
84.                  label = '%s: %d%%' % (object_name, int(scores[i]*100)) #
   Example: 'person: 72%'
85.                  labelSize, baseLine = cv2.getTextSize(label,
   cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
86.                  label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to
   draw label too close to top of window
87.                  cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10),
   (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw
   white box to put label text in
88.                  cv2.putText(frame, label, (xmin, label_ymin-7),
   cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text
89.          # All the results have been drawn on the frame, so it's time to
   display it.
90.          cv2.imshow('Object detector', frame)
91.
92.          if cv2.waitKey(1) == ord('q'):
93.              break
94.
95.      video.release()
96.      cv2.destroyAllWindows()
```

Prueba N°2

Este código se basa en la captura de un video en tiempo real por medio de una webcam y el reconocimiento de los objetos capturados

Programa de cmd:

```
1. cd tflite1
2. source tflite1-env/bin/activate
3. python3 TFLite_detection_webcam.py --modeldir=Sample_TFLite_model
```

Programa de phyton:

```
4. import os
5. import argparse
6. import cv2
7. import numpy as np
```


```
8. import sys
9. import time
10.      from threading import Thread
11.      import importlib.util
12.      class VideoStream:
13.         """Camera object that controls video streaming from the Picamera"""
14.         def __init__(self,resolution=(640,480),framerate=30):
15.             self.stream = cv2.VideoCapture(0)
16.             ret = self.stream.set(cv2.CAP_PROP_FOURCC,
    cv2.VideoWriter_fourcc(*'MJPG'))
17.             ret = self.stream.set(3,resolution[0])
18.             ret = self.stream.set(4,resolution[1])
19.             (self.grabbed, self.frame) = self.stream.read()
20.             self.stopped = False
21.         def start(self):
22.             Thread(target=self.update,args=()).start()
23.             return self
24.         def update(self):
25.             while True:
26.                 if self.stopped:
27.                     self.stream.release()
28.                     Return
29.                 (self.grabbed, self.frame) = self.stream.read()
30.         def read(self):
31.             return self.frame
32.         def stop(self):
33.             self.stopped = True
34.      parser = argparse.ArgumentParser()
35.      parser.add_argument('--modeldir', help='Folder the .tflite file is
    located in',
36.                          required=True)
37.      parser.add_argument('--graph', help='Name of the .tflite file, if
    different than detect.tflite',
38.                          default='detect.tflite')
39.      parser.add_argument('--labels', help='Name of the labelmap file, if
    different than labelmap.txt',
40.                          default='labelmap.txt')
41.      parser.add_argument('--threshold', help='Minimum confidence threshold
    for displaying detected objects',
42.                          default=0.5)
43.      parser.add_argument('--resolution', help='Desired webcam resolution in
    WxH. If the webcam does not support the resolution entered, errors may
    occur.',
44.                          default='1280x720')
45.      parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator
    to speed up detection',
46.                          action='store_true')
47.      args = parser.parse_args()
48.      MODEL_NAME = args.modeldir
```


```
49.      GRAPH_NAME = args.graph
50.      LABELMAP_NAME = args.labels
51.      min_conf_threshold = float(args.threshold)
52.      resW, resH = args.resolution.split('x')
53.      imW, imH = int(resW), int(resH)
54.      use_TPU = args.edgetpu
55.      pkg = importlib.util.find_spec('tflite_runtime')
56.      if pkg:
57.         from tflite_runtime.interpreter import Interpreter
58.         if use_TPU:
59.             from tflite_runtime.interpreter import load_delegate
60.      else:
61.         from tensorflow.lite.python.interpreter import Interpreter
62.         if use_TPU:
63.             from tensorflow.lite.python.interpreter import load_delegate
64.      if use_TPU:
65.         if (GRAPH_NAME == 'detect.tflite'):
66.             GRAPH_NAME = 'edgetpu.tflite'
67.      CWD_PATH = os.getcwd()
68.      PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)
69.      PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)
70.      with open(PATH_TO_LABELS, 'r') as f:
71.         labels = [line.strip() for line in f.readlines()]
72.      if labels[0] == '???':
73.         del(labels[0])
74.      if use_TPU:
75.         interpreter = Interpreter(model_path=PATH_TO_CKPT,
76.                                   experimental_delegates=[load_delegate('lib
   edgetpu.so.1.0')])
77.      print(PATH_TO_CKPT)
78.      else:
79.      interpreter = Interpreter(model_path=PATH_TO_CKPT)
80.      interpreter.allocate_tensors()
81.      input_details = interpreter.get_input_details()
82.      output_details = interpreter.get_output_details()
83.      height = input_details[0]['shape'][1]
84.      width = input_details[0]['shape'][2]
85.      floating_model = (input_details[0]['dtype'] == np.float32)
86.      input_mean = 127.5
87.      input_std = 127.5
88.      frame_rate_calc = 1
89.      freq = cv2.getTickFrequency()
90.      videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
91.      time.sleep(1)
92.      while True:
93.      t1 = cv2.getTickCount()
94.      frame1 = videostream.read()
```


```
95.      frame = frame1.copy()
96.      frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
97.      frame_resized = cv2.resize(frame_rgb, (width, height))
98.      input_data = np.expand_dims(frame_resized, axis=0)
99.      if floating_model:
100.     input_data = (np.float32(input_data) - input_mean) / input_std
101.     interpreter.set_tensor(input_details[0]['index'],input_data)
102.     interpreter.invoke()
103.     boxes = interpreter.get_tensor(output_details[0]['index'])[0] #
   Bounding box coordinates of detected objects
104.     classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class
   index of detected objects
105.     scores = interpreter.get_tensor(output_details[2]['index'])[0] #
   Confidence of detected objects
106.     for i in range(len(scores)):
107.     if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):
108.     ymin = int(max(1,(boxes[i][0] * imH)))
109.     xmin = int(max(1,(boxes[i][1] * imW)))
110.     ymax = int(min(imH,(boxes[i][2] * imH)))
111.     xmax = int(min(imW,(boxes[i][3] * imW)))
112.     cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)
113.     object_name = labels[int(classes[i])] # Look up object name from
   "labels" array using class index
114.     label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example:
   'person: 72%'
115.     labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX,
   0.7, 2) # Get font size
116.     label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label
   too close to top of window
117.     cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10),
   (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) #
   Draw white box to put label text in
118.     cv2.putText(frame, label, (xmin, label_ymin-7),
   cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text
119.     cv2.putText(frame,'FPS:
   {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,
   0),2,cv2.LINE_AA)
120.     cv2.imshow('Object detector', frame)
121.     t2 = cv2.getTickCount()
122.     time1 = (t2-t1)/freq
123.     frame_rate_calc= 1/time1
124.     if cv2.waitKey(1) == ord('q'):
125.     break
126.     cv2.destroyAllWindows()
127.     videostream.stop()
```

Prueba N°3 (código final)

Programa de cmd:

```
cd tflite1
source tflite1-env/bin/activate
python3 TFLite_detection_webcam.py --modeldir=Sample_TFLite_model
```

Programa de phyton:

```
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util
import webbrowser
class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(640,480),framerate=30):
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC,
   cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])

           (self.grabbed, self.frame) = self.stream.read()


        self.stopped = False

    def start(self):

        Thread(target=self.update,args=()).start()
        return self

    def update(self):
        while True:

            if self.stopped:
                self.stream.release()
                return
            (self.grabbed, self.frame) = self.stream.read()
    def read(self):
        return self.frame
    def stop(self):
        self.stopped = True

parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located
   in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if
   different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if
   different than labelmap.txt',
                    default='labelmap.txt')
```

```
177.     parser.add_argument('--threshold', help='Minimum confidence threshold for
   displaying detected objects',
178.                         default=0.5)
179.     parser.add_argument('--resolution', help='Desired webcam resolution in
   WxH. If the webcam does not support the resolution entered, errors may occur.',
180.                         default='1280x720')
181.     parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to
   speed up detection',
182.                         action='store_true')
183.
184.     args = parser.parse_args()
185.
186.     MODEL_NAME = args.modeldir
187.     GRAPH_NAME = args.graph
188.     LABELMAP_NAME = args.labels
189.     min_conf_threshold = float(args.threshold)
190.     resW, resH = args.resolution.split('x')
191.     imW, imH = int(resW), int(resH)
192.     use_TPU = args.edgetpu
193.
194.     pkg = importlib.util.find_spec('tflite_runtime')
195.     if pkg:
196.         from tflite_runtime.interpreter import Interpreter
197.         if use_TPU:
198.             from tflite_runtime.interpreter import load_delegate
199.     else:
200.         from tensorflow.lite.python.interpreter import Interpreter
201.         if use_TPU:
202.             from tensorflow.lite.python.interpreter import load_delegate
203.
204.
205.     if use_TPU:
206.         if (GRAPH_NAME == 'detect.tflite'):
207.             GRAPH_NAME = 'edgetpu.tflite'
208.     CWD_PATH = os.getcwd()
209.
210.     PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)
211.     PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)
212.
213.     with open(PATH_TO_LABELS, 'r') as f:
214.         labels = [line.strip() for line in f.readlines()]
215.     if labels[0] == '???':
216.         del(labels[0])
217.     if use_TPU:
218.         interpreter = Interpreter(model_path=PATH_TO_CKPT,
219.
   experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
220.         print(PATH_TO_CKPT)
221.     else:
222.         interpreter = Interpreter(model_path=PATH_TO_CKPT)
223.
224.     interpreter.allocate_tensors()
225.     input_details = interpreter.get_input_details()
226.     output_details = interpreter.get_output_details()
227.     height = input_details[0]['shape'][1]
228.     width = input_details[0]['shape'][2]
229.     floating_model = (input_details[0]['dtype'] == np.float32)
230.     input_mean = 127.5
```


```
    input_std = 127.5
    frame_rate_calc = 1
    freq = cv2.getTickFrequency()
    videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
    time.sleep(1)
    while True:
        t1 = cv2.getTickCount()
        frame1 = videostream.read()
        frame = frame1.copy()
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_resized = cv2.resize(frame_rgb, (width, height))
        input_data = np.expand_dims(frame_resized, axis=0)
        if floating_model:
            input_data = (np.float32(input_data) - input_mean) / input_std
        interpreter.set_tensor(input_details[0]['index'],input_data)
        interpreter.invoke()
        boxes = interpreter.get_tensor(output_details[0]['index'])[0]
        classes = interpreter.get_tensor(output_details[1]['index'])[0]
        scores = interpreter.get_tensor(output_details[2]['index'])[0]
        #num = interpreter.get_tensor(output_details[3]['index'])[0]
        for i in range(len(scores)):
            if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):
                ymin = int(max(1,(boxes[i][0] * imH)))
                xmin = int(max(1,(boxes[i][1] * imW)))
                ymax = int(min(imH,(boxes[i][2] * imH)))
                xmax = int(min(imW,(boxes[i][3] * imW)))
                cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0),
   2)
                object_name = labels[int(classes[i])] # Look up object name
   from "labels" array using class index
                label = '%s: %d%%' % (object_name, int(scores[i]*100)) #
   Example: 'person: 72%'
                labelSize, baseLine = cv2.getTextSize(label,
   cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
                label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to
   draw label too close to top of window
                cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10),
   (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw
   white box to put label text in
                cv2.putText(frame, label, (xmin, label_ymin-7),
   cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

                if (object_name=="tv"):

   webbrowser.open("http://www.python.org",new=2,autoraise=True)
                    time.sleep (60)
        cv2.putText(frame,'FPS:
   {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),
   2,cv2.LINE_AA)
        cv2.imshow('Object detector', frame)
        t2 = cv2.getTickCount()
        time1 = (t2-t1)/freq
        frame_rate_calc= 1/time1
        if cv2.waitKey(1) == ord('q'):
            break
    cv2.destroyAllWindows()
    videostream.stop()
```