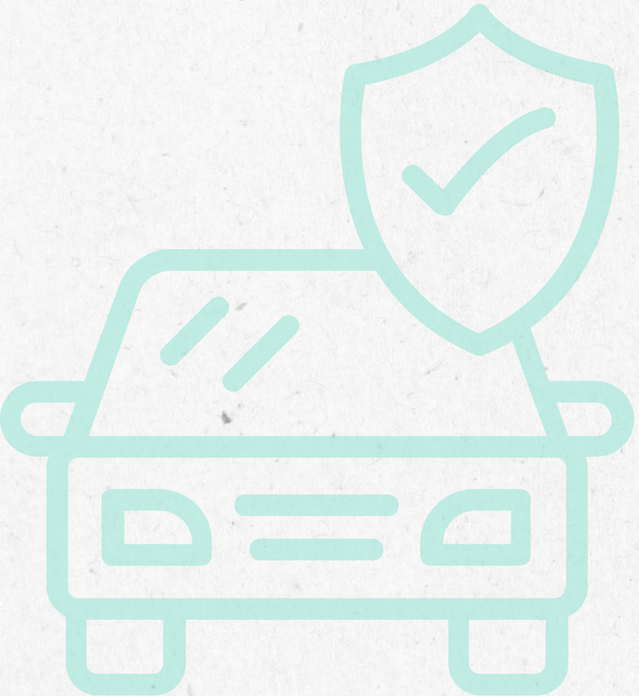
Estimación de la velocidad de un vehículo

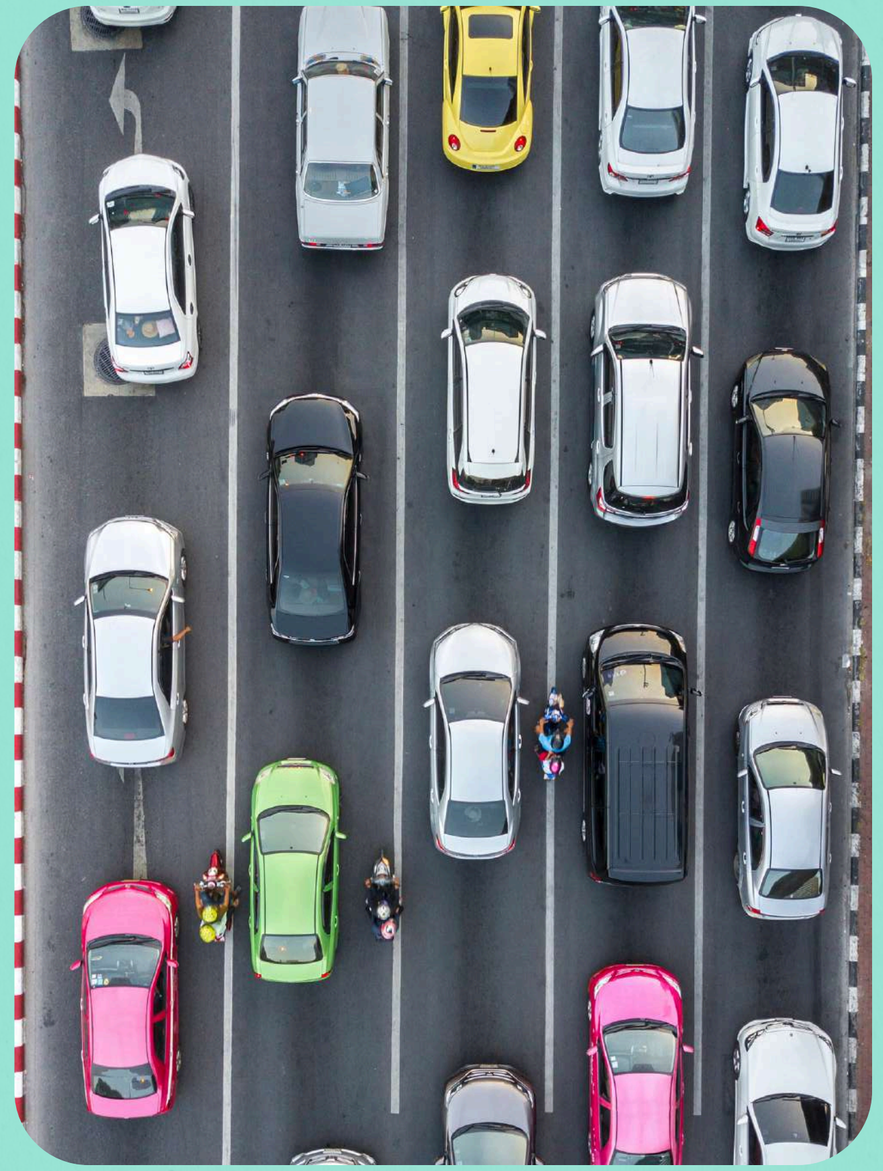


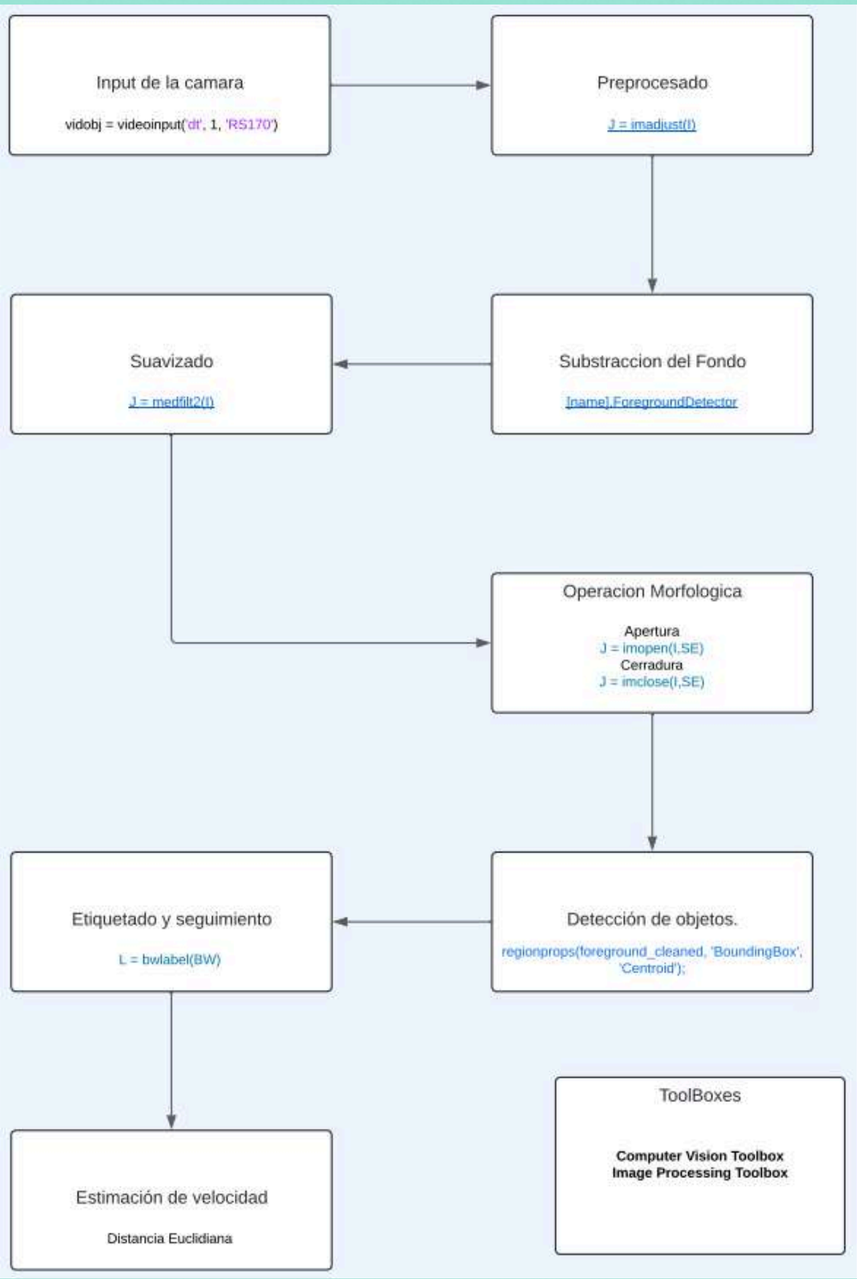
Diseño de sistemas embebidos avanzados

Oscar Ortiz Torres A01769292
Yonathan Romero Amador A01737244
Emmanuel Lechuga Arreola A01736241

Objetivo

Hacer uso de MATLAB, procesamiento de imágenes y Raspberry Pi para hacer un sistema embebido dedicado a calcular la velocidad de un vehículo dentro del campus, esto con el objetivo de regular las normas de tránsito y evitar que los conductores superen los límites indicados.





Speed Estimation On Moving Vehicle Based On Digital Image Processing

Autores: Danang Wicaksono & Budi Setiyono

```m
function rasp_final_e()
    %% Implementación del algoritmo de estimación de la velocidad de un vehículo

    %{
    Paper de referencia:
    "Speed Estimation On Moving Vehicle Based On Digital Image Processing"
    (https://www.researchgate.net/publication/317312246)
    %}

    % Autores: Oscar Ortiz Torres, Yonathan Romero Amador y Emmanuel Lechuga Arreola

    %% 0. Definición de constantes, objetos y variables
    r = raspi('192.168.33.204', 'YRA', 'yra');  % Conexión con la Raspberry Pi
    %{
    Tv = 67.6; % Ángulo
    H = 2.12; % Altura de la cámara en m
    v = 0.050; %% Dimensión vertical del formato de imagen
    f = 0.002; % Distancia focal de la cámara
    %}
    foregroundDetector = vision.ForegroundDetector('NumGaussians', 10, ...
                                                   'NumTrainingFrames', 35, ...
                                                   'LearningRate', 0.002, ...
                                                   'MinimumBackgroundRatio', 0.7);


    prev_centroide = [0 0];
    vel_vector=[];
    area_vector = [];
    flag = 0;

    cam = webcam(r); % Acceso a la cámara web
    fps = 29.73;
    %{
    % Calibración del campo de visión y factor de conversión
    Tc = 2 * atan(v / (2 * f)); % Campo de visión de la cámara
    T = Tv + Tc / 2; % Ángulo total de visión
    D = H * tan(T); % Distancia de la cámara al objeto
    I_height = cam.Resolution(2); % Altura del video en pixeles
    P = 2 * tan(Tc / 2) * sqrt(H^2 + D^2); % Proyección del campo de visión (en metros)
```

```
39      k = P / I_height; % Factor de conversión de pixeles a metros
40      %}
41      I_height = cam.Resolution(2);
42      k = 0.0317; %0194  - 5:15
43      %% Procesamiento en tiempo real
44      for f = 1:120
45          %% B. Preprocessing
46          img = snapshot(cam);    % Captura del frame de la cámara
47          img_pre = im2gray(img);
48          img_pre = imadjust(img_pre, [0.32 0.92], [0 1], 2);
49
50          %% C. Background Subs (restar el fondo del frame actual) & E. Shadow Removal
51          bg_sub = step(foregroundDetector, img_pre);
52
53          %% D. Smoothing
54          img_suav = medfilt2(bg_sub, [20 20]);
55
56          %% F. Operaciones morfológicas para eliminar ruido
57          img_morfo = imopen(img_suav, strel('square', 4));
58          img_morfo = imclose(img_morfo, strel('square', 15));
59
60          %% G. Object detection
61          img_morfo_uint8 = uint8(img_morfo) * 255;
62          [labels, num] = bwlabel(img_morfo_uint8);
63          stats = regionprops(labels, 'BoundingBox', 'Centroid', 'Area');
64
65          %% H. Labeling and tracking
66          %imshow(img)    % Se reemplaza por la línea 60, con la función displayImage, ya que imshow no es soportado por la Rasp
67          for i = 1:num
68              if stats(i).Area > 100 && flag == 0
69                  bbox = stats(i).BoundingBox;
70                  centroide = stats(i).Centroid;
71
72                  %%%
73                  % Coordenada inferior del objeto (bounding box)
74                  y_inferior = bbox(2) + bbox(4); % y_inicial + altura del bounding box
75
```

```
                % Condición para ignorar objetos que alcanzan el borde inferior del frame
                if y_inferior >= I_height - 10 % Margen de 10 píxeles
                    %continue; % Salta al siguiente objeto
                    flag = 1;
                end
                %%%

                %rectangle('Position', bbox, 'EdgeColor', 'y','LineWidth',%2); % Se reemplaza por la línea 57, con la función insertShape en lugar de rectangle
                img = insertShape(img, 'Rectangle', bbox, 'Color', [255, 255, 0], 'LineWidth', 2);
                %plot(centroide(1), centroide(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2); % Se reemplaza por la linea 59 con la función insertShape en lugar de plot
                img = insertShape(img, 'Circle', [centroide(1), centroide(2), 5], 'Color', [255, 0, 0], 'LineWidth', 3);
                %displayImage(r, img, 'Title', 'Fotomulta');

                %% I. Speed Estimation
                d_e = sqrt(sum((prev_centroide - centroide) .^ 2));
                vel = 3.6 * (k * (d_e / (1 / fps)));
                vel_vector = [vel_vector, vel];
                area_vector = [area_vector, vel];

                % Guardado de coordenadas para la siguiente iteración
                prev_centroide = centroide;
            end
        end
    displayImage(r, img, 'Title', 'Fotomulta');
    end
    vel_vector(1) = []; % Eliminación de la primera muestra

    % Impresión de todas las velocidades calculadas
    for v_idx = 1:length(vel_vector)
        fprintf('Velocidad en el frame %.0f: %.2f km/h\n', v_idx, vel_vector(v_idx));
    end
    for a_idx = 1:length(area_vector)
        fprintf('Area de objeto en el frame %.0f: %.2f km/h\n', a_idx, area_vector(a_idx));
    end

    % Impresión de la velocidad promedio del objeto
    fprintf('\nVelocidad promedio del objeto: %.2f km/h\n', mean(vel_vector));
end
```

Metodología de trabajo

01

Código de procesamiento en MATLAB en base a un video

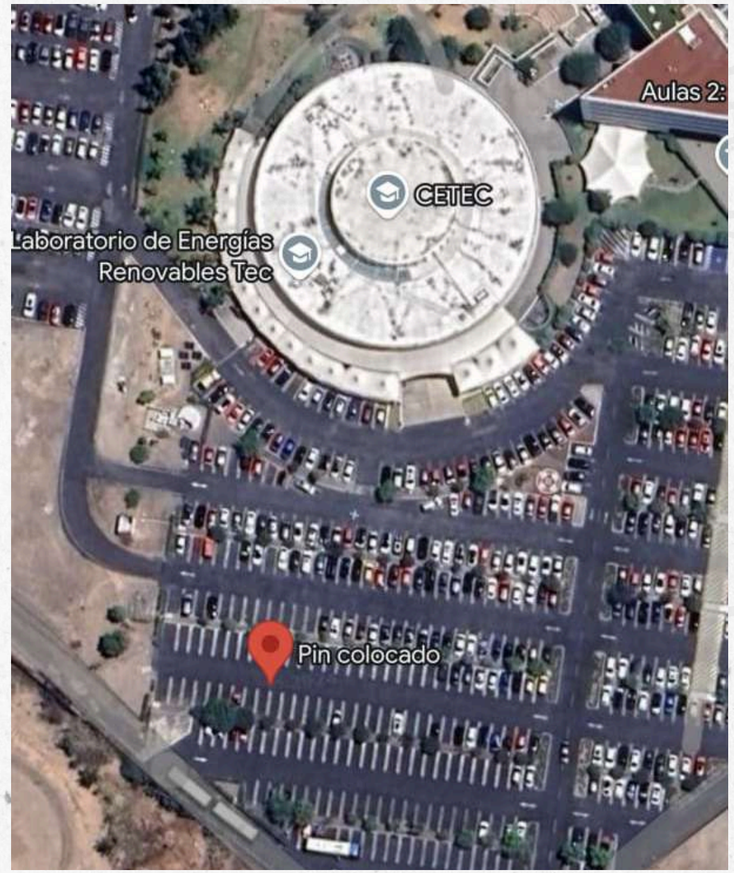
02

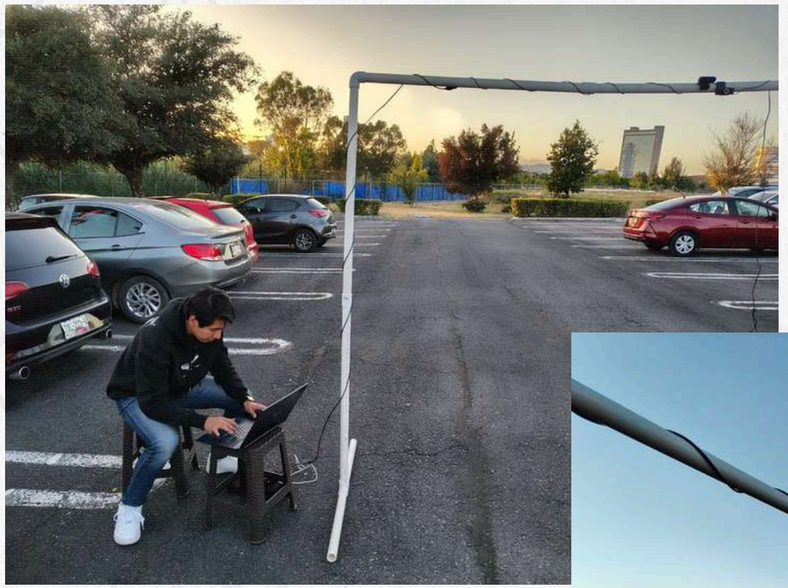
Implementación del código en Raspberry Pi usando una webcam

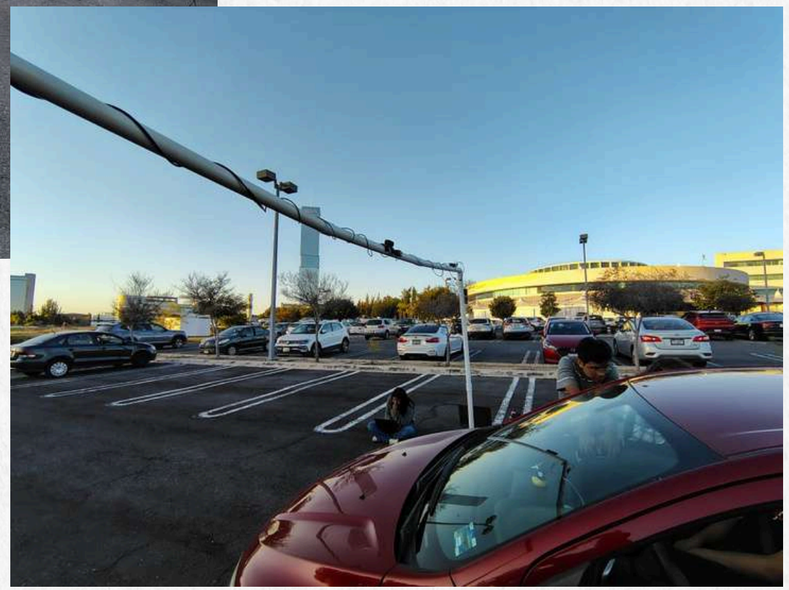
03

Verificación del sistema en tiempo real, usando vehículos y velocidades distintas

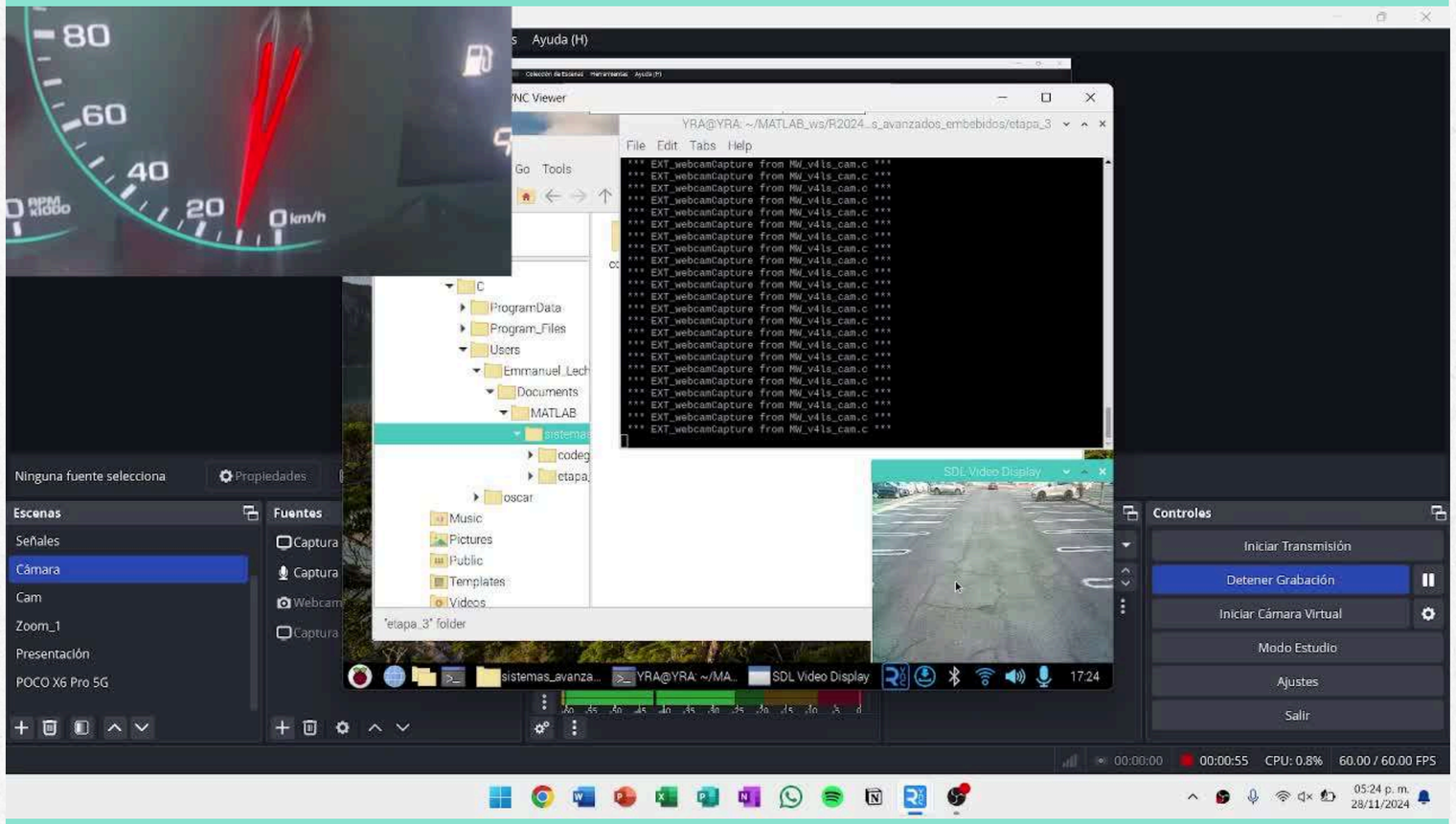
Experimento







Vehículo Rojo | 10 km/h

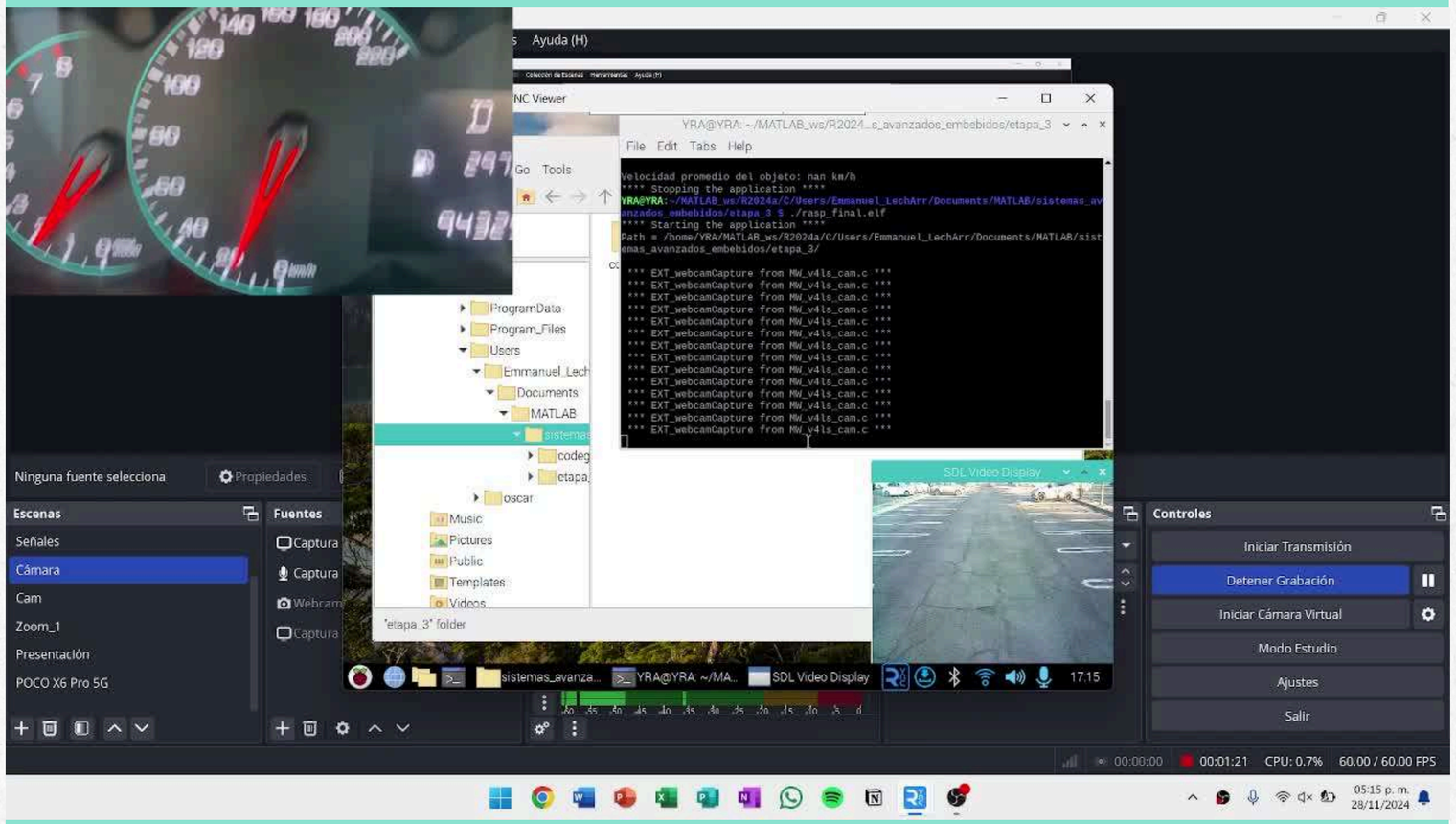


Pruebas

Vel. Real: 11.5 Km/h

Vel. Estimada: 13.10 Km/h

Vehículo Rojo | 20 km/h

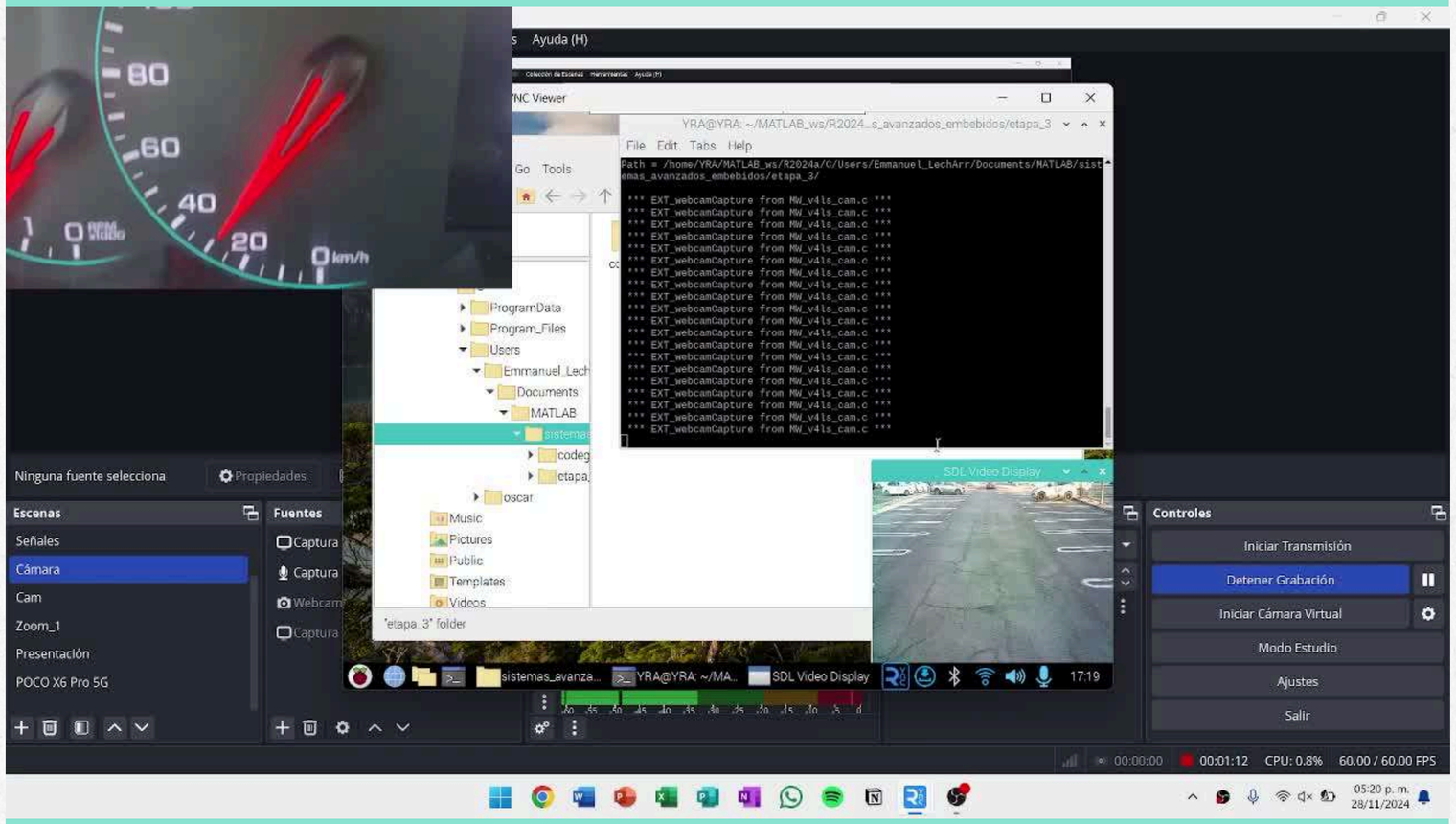


Pruebas

Vel. Real: 19 Km/h

Vel. Estimada: 20.01 Km/h

Vehículo Rojo | 30 km/h

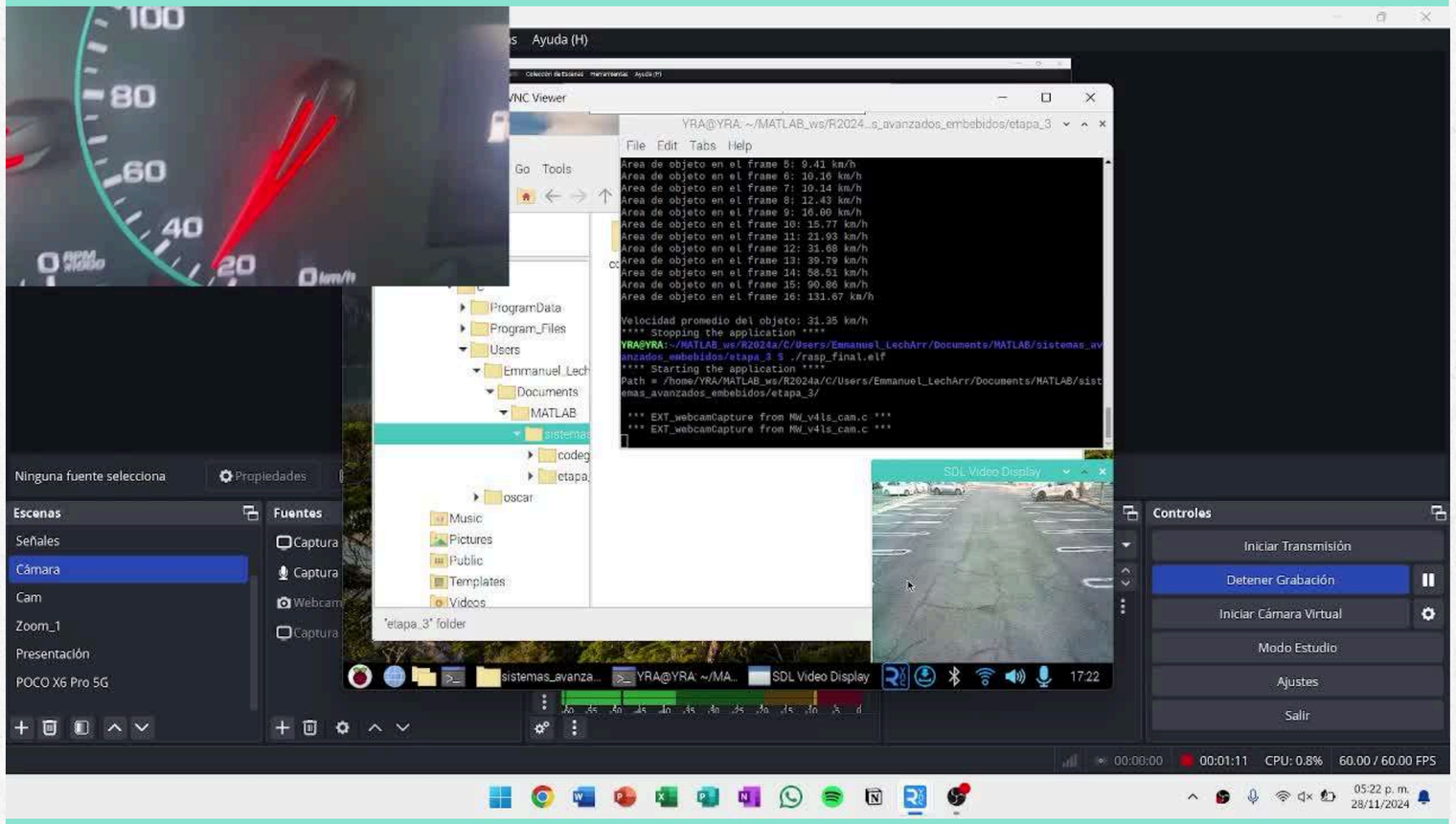


Pruebas

Vel. Real: 29 Km/h

Vel. Estimada: 31.35 Km/h

Vehículo Rojo | 40 km/h

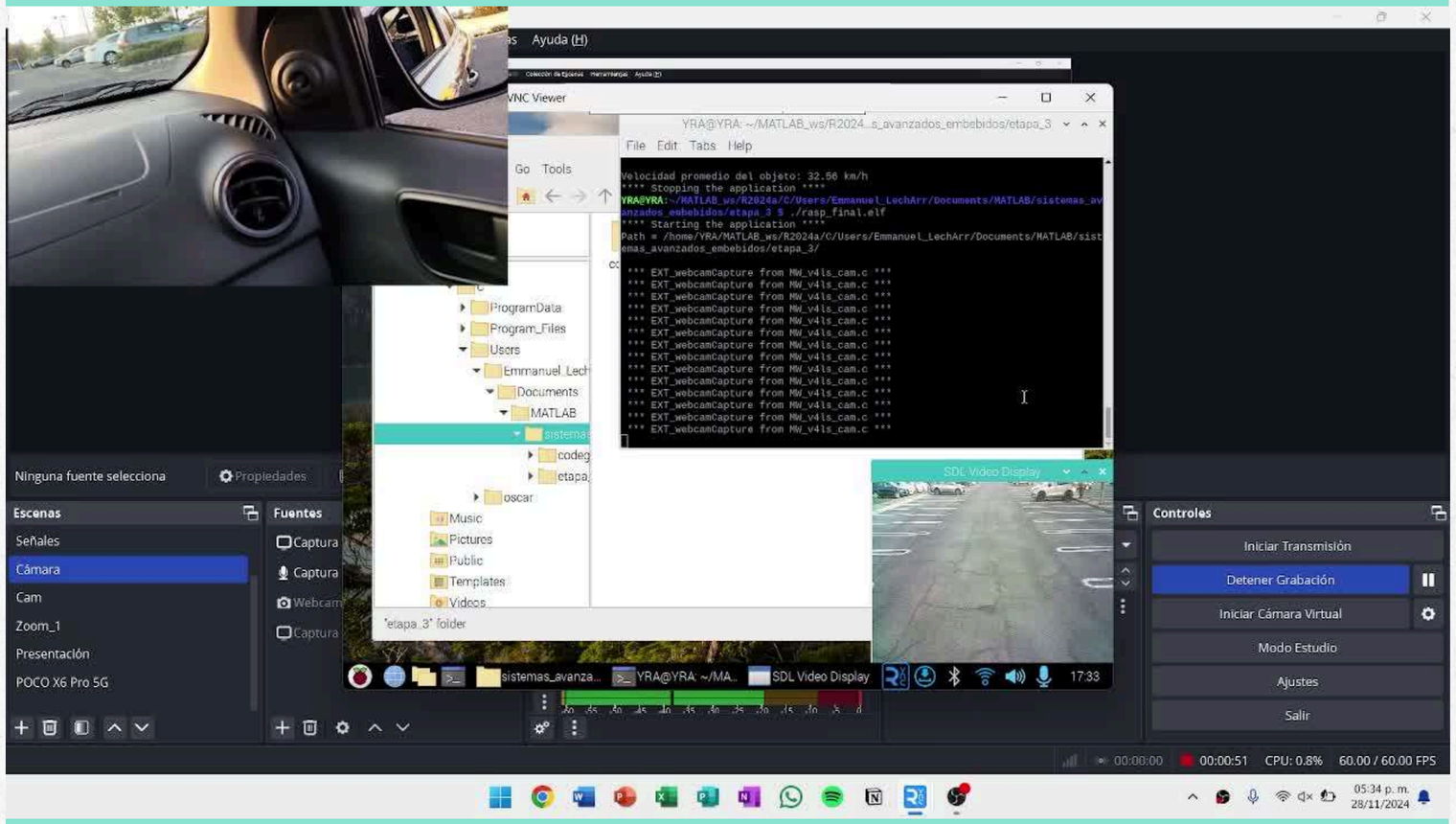


Pruebas

Vel. Real: 39.5 Km/h

Vel. Estimada: 40.77 Km/h

Vehículo Gris | 10 km/h

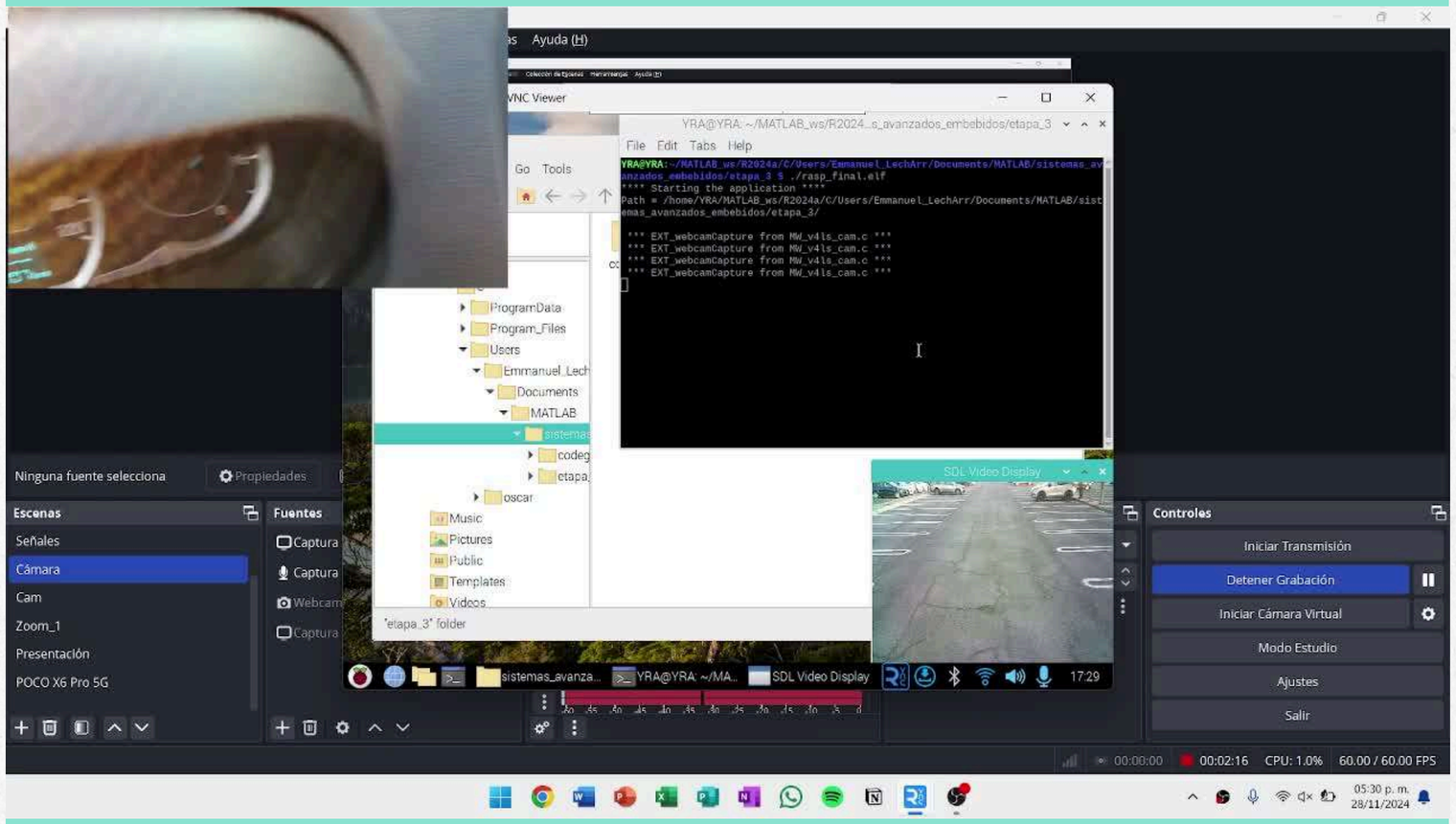


Pruebas

Vel. Real: 13 Km/h

Vel. Estimada: 11.14 Km/h

Vehículo Gris | 20 km/h

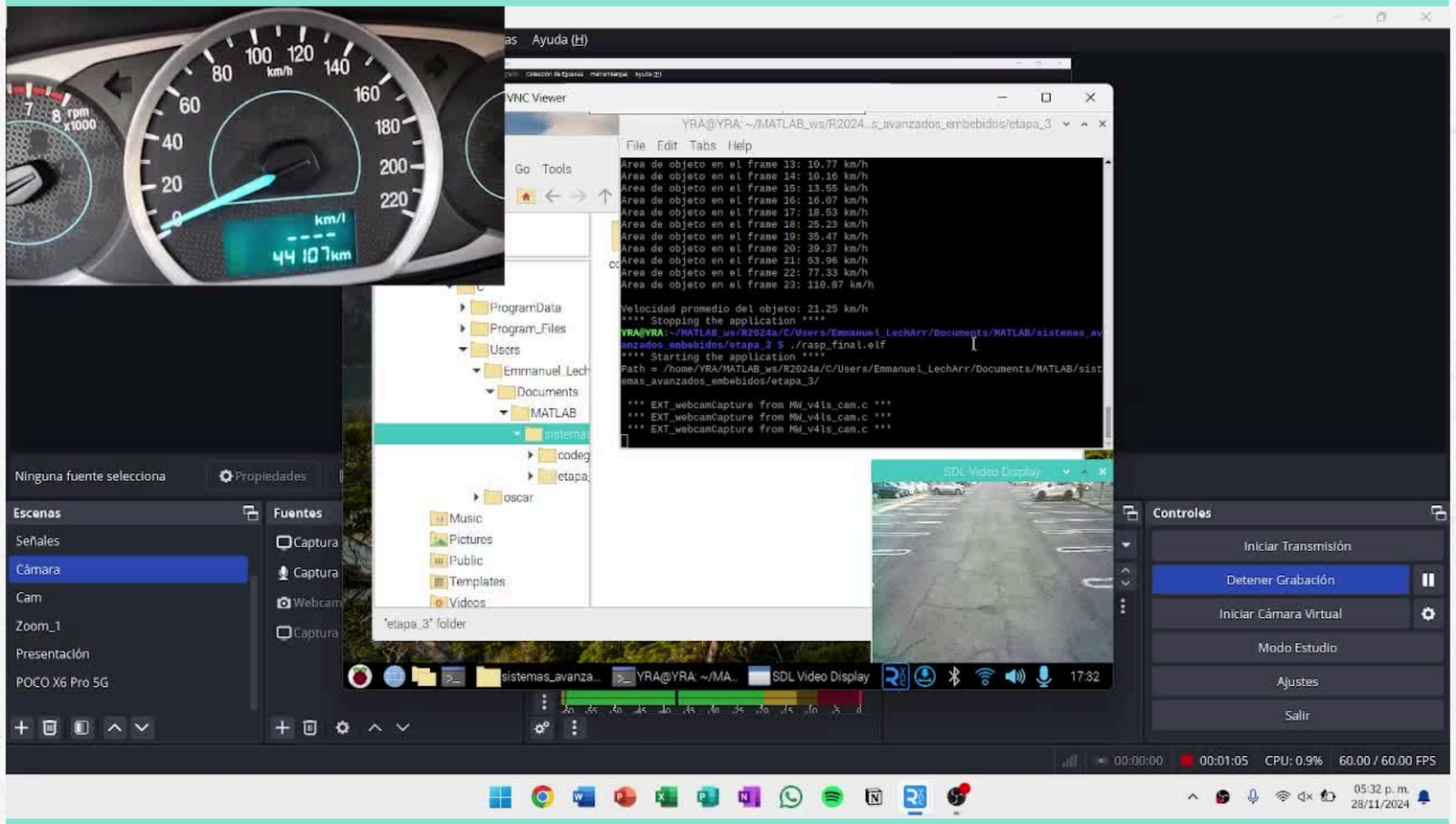


Pruebas

Vel. Real: 21.5 Km/h

Vel. Estimada: 21.25 Km/h

Vehículo Gris | 30 km/h



Pruebas

Vel. Real: 32 Km/h

Vel. Estimada: 32.56 Km/h

Prueba	Km Coche	Km detectados	Error	E. Porcentual
Rojo10	11.5	13.1	1.6	13.9130435
Rojo20	19	20.01	1.01	5.31578947
Rojo30	29	31.35	2.35	8.10344828
Rojo40	39.5	40.77	1.27	3.21518987
Gris10	13	11.14	1.86	14.3076923
Gris20	21.5	21.25	0.25	1.1627907
Gris30	32	32.56	0.56	1.75
			Promedio	6.82399344

6.82%

De error

¡Muchas gracias!