UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE INGENIERÍA CIVIL, SISTEMAS Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



**ROBÓTICA**

**Tarea 13 - Construcción física**

**Integrantes**

Fernández Delgado Cleiver Norberto

Mayanga Muñoz Jonatan Arturo

**Docente**

Ing. Juan Villegas Cubas

Lambayeque, marzo del 2024

**Código de Ejecución del Prototipo**

**El código realiza las siguientes tareas:**

* Importa las bibliotecas necesarias.
* Configura los pines GPIO para controlar los motores , un sensor ultrasónico y un servo motor.
* Inicializa la cámara y carga un modelo de TensorFlow/Keras.
* Define funciones para calcular la distancia con el sensor ultrasónico, capturar una imagen y analizar la imagen.
* En la función main(), se ejecuta un bucle infinito que:
* Calcula la distancia usando el sensor ultrasónico.
* Enciende dos Motores.
* Si la distancia es menor a 10 cm:
* Captura una imagen de la cámara.
* Analiza la imagen utilizando el modelo de TensorFlow/Keras y obtiene el resultado ("mala" o "buena").
* Si el resultado es "mala", mueve el servo motor a 90 grados y espera 5 segundos.
* Si el resultado es "buena", mueve el servo motor a 0 grados.
* Espera 1 segundo.
* Si la distancia es mayor o igual a 10 cm, mueve el servo motor a 0 grados.
* El programa se puede detener con Ctrl+C.
* Al finalizar, se liberan los recursos utilizados (cámara y pines GPIO).

**import cv2**

**import RPi.GPIO as GPIO**

**from gpiozero import Servo**

**import time**

**import numpy as np**

**from PIL import Image**

**from tensorflow.keras.models import load\_model**

**from tensorflow.keras.preprocessing import image**

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(20, GPIO.OUT)

GPIO.setup(26, GPIO.OUT)

GPIO.setup(19, GPIO.OUT)

GPIO.setup(16, GPIO.OUT)

TRIG\_PIN = 23 # Pin GPIO para el pin TRIG del sensor

ECHO\_PIN = 24 # Pin GPIO para el pin ECHO del sensor

SERVO\_PIN = 18 # Pin GPIO para el servo

GPIO.setup(TRIG\_PIN, GPIO.OUT)

GPIO.setup(ECHO\_PIN, GPIO.IN)

servo = Servo(SERVO\_PIN)

**cap = cv2.VideoCapture(0) # Inicializar la cámara**

**modelo = load\_model("Modelos/modelo\_entrenado\_p21.h5")**

**def distance():**

GPIO.output(TRIG\_PIN, True)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, False)

pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 0:

pulse\_start = time.time()

pulse\_end = time.time()

while GPIO.input(ECHO\_PIN) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance\_cm = pulse\_duration \* 17150

distance\_cm = round(distance\_cm, 2)

return distance\_cm

**def capture\_image():**

ret, frame = cap.read()

frame = cv2.resize(frame, (299, 299)) # Redimensionar la imagen a 300x300

image\_name = "imagen\_palta.jpg"

cv2.imwrite(image\_name, frame)

print(f"Imagen capturada y guardada como {image\_name}")

**def analizar\_imagen(nombre\_archivo):**

img = image.load\_img(nombre\_archivo, target\_size=(299,299))

img = image.img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = img / 255.0

print("PREDICIENDO IMAGEN ...")

predicciones = modelo.predict(img)

clase\_predicha = np.argmax(predicciones)

if clase\_predicha == 0:

return "mala"

elif clase\_predicha == 1:

return "buena"

**def main():**

**try:**

while True:

dist = distance()

GPIO.output(20, GPIO.HIGH)

GPIO.output(16, GPIO.HIGH)

print("Distancia:", dist, "cm")

if dist < 10:

print("predicir")

capture\_image()

resultado = analizar\_imagen("imagen\_palta.jpg")

#image = Image.open("palta1.jpg")

#image.show()

print("predicion",resultado)

if resultado == "mala":

servo.value = 0.5 # 90 grados

time.sleep(5)

else:

servo.value = 0

time.sleep(1)

else:

servo.value = 0

except KeyboardInterrupt:

print("Programa detenido por el usuario")

**finally:**

cap.release()

cv2.destroyAllWindows()

GPIO.cleanup()

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

from gpiozero import PiGPIOFactory

ImportError: cannot import name 'PiGPIOFactory' from 'gpiozero' (/usr/lib/python3/dist-packages/gpiozero File "/home/xiomara/Documents/hola.py", line 26, in <module>

servo = pi.servo(SERVO\_PIN)

AttributeError: 'pi' object has no attribute 'servo/\_\_init\_\_.py) T T T T T T T

)

import os

import cv2

import RPi.GPIO as GPIO

import numpy as np

from PIL import Image

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

from gpiozero import Servo

import time

# Suprimir las advertencias de TensorFlow

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

# Suprimir la advertencia de OpenCV

cv2.setUseOptimized(False)

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(20, GPIO.OUT)

GPIO.setup(26, GPIO.OUT)

GPIO.setup(19, GPIO.OUT)

GPIO.setup(16, GPIO.OUT)

GPIO.setup(18, GPIO.OUT)

TRIG\_PIN = 23 # Pin GPIO para el pin TRIG del sensor

ECHO\_PIN = 24 # Pin GPIO para el pin ECHO del sensor

SERVO\_PIN = 18 # Pin GPIO para el servo

GPIO.setup(TRIG\_PIN, GPIO.OUT)

GPIO.setup(ECHO\_PIN, GPIO.IN)

servo = Servo(SERVO\_PIN)

cap = cv2.VideoCapture(0) # Inicializar la cámara

modelo = load\_model("Modelos/modelo\_entrenado\_p21.h5")

def distance():

GPIO.output(TRIG\_PIN, True)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN, False)

pulse\_start = time.time()

while GPIO.input(ECHO\_PIN) == 0:

pulse\_start = time.time()

pulse\_end = time.time()

while GPIO.input(ECHO\_PIN) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance\_cm = pulse\_duration \* 17150

distance\_cm = round(distance\_cm, 2)

return distance\_cm

def capture\_image():

ret, frame = cap.read()

frame = cv2.resize(frame, (299, 299)) # Redimensionar la imagen a 300x300

image\_name = "imagen\_palta.jpg"

cv2.imwrite(image\_name, frame)

def analizar\_imagen(nombre\_archivo):

img = image.load\_img(nombre\_archivo, target\_size=(299,299))

img = image.img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = img / 255.0

predicciones = modelo.predict(img)

clase\_predicha = np.argmax(predicciones)

if clase\_predicha == 0:

return "mala"

elif clase\_predicha == 1:

return "buena"

def main():

try:

while True:

dist = distance()

GPIO.output(20, GPIO.HIGH)

GPIO.output(16, GPIO.HIGH)

servo.value = 1

print("Distancia:", dist, "cm")

if dist < 10:

print("predicir")

capture\_image()

resultado = analizar\_imagen("palta1.jpg")

print("predicion", resultado)

if resultado == "mala":

servo.value = 0

time.sleep(2)

else:

servo.value = 0.5

time.sleep(3)

else:

servo.value = 1

except KeyboardInterrupt:

print("Programa detenido por el usuario")

finally:

cap.release()

cv2.destroyAllWindows()

GPIO.cleanup()

if \_\_name\_\_ == "\_\_main\_\_":

main()