Códigos de los sensores de calidad del agua

Sensor de temperatura

```
import os
import glob
import time

# Cargar módulos del kernel 1-Wire
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

# Ruta del sensor
base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw() :
```

```
    with open(device_file, 'r') as f:
        return f.readlines()


def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=
```

```
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c

try:
    while True:
        print("Temperatura: {:.2f} °C".format(read_temp()))
        time.sleep(2)
except KeyboardInterrupt:
    print("\nPrograma terminado.")
```

Código para leer pH y turbidez

```
import time
import board
import busio
from adafruit_ads1x15.ads1115 import ADS1115
from adafruit_ads1x15.analog_in import AnalogIn

# Inicializa I2C
i2c = busio.I2C(board.SCL, board.SDA)

# Crea objeto del ADC con dirección 0x48
ads = ADS1115(i2c, address=0x48
```

```
# Define los canales
canal_ph = AnalogIn(ads, 0)    # A0
canal_turbidez = AnalogIn(ads, 1)  # A1

# Calibraciones iniciales (ajustar luego)
# Asumimos pH = 7 a 2.5V, pH = 4 a 3.0V, pH = 10 a 2.0V
def calcular_ph(voltage):
    ph = 7 + ((2.5 - voltage) / 0.18)  # fórmula aproximada
    return ph

# Turbidez: más voltaje = más clara el agua
def calcular_turbidez(voltage):
    # fórmula de ejemplo, depende del sensor
    ntu = (4.2 - voltage) * 100  # ajustar experimentalmente
    if ntu < 0: ntu = 0
    return ntu

try:
    while True:
        volt_ph = canal_ph.voltage
        volt_turb = canal_turbidez.voltage

        ph_valor = calcular_ph(volt_ph)
        ntu_valor = calcular_turbidez(volt_turb)

        print(f"Voltaje pH: {volt_ph:.3f} V  → pH: {ph_valor:.2f}")
        print(f"Voltaje Turbidez: {volt_turb:.3f} V  → NTU aprox: {ntu_valor:.1f}"
```

```
        print("-"*50)

        time.sleep(2)

except KeyboardInterrupt:
    print("\nLectura finalizada.")
```

Código para usar la cámara USB con Python

```
import cv2

# Abre la cámara USB (por lo general /dev/video0)
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("No se puede abrir la cámara")
    exit()

while True:
    ret, frame = cap.read()
    if not ret:
        print("No se puede recibir frame (fin del stream?)")
        break

    cv2.imshow('Camara USB', frame)

    if cv2.waitKey(1) == ord('q'):
        break
```

```
cap.release()
cv2.destroyAllWindows()
```

Código para capturar las imágenes.

```
import cv2
import os

# Crear carpeta donde se guardarán las imágenes
carpeta = "imagenes_tilapia"
os.makedirs(carpeta, exist_ok=True)

# Inicializar cámara USB (0 o 1 según tu cámara)
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("❌ No se pudo abrir la cámara. Verifica la conexión USB.")
    exit()

contador = 0
print("✅ Cámara lista. Presiona 'c' para capturar, 'q' para salir.")

while True:
    ret, frame = cap.read()
```

```
    if not ret:
        print("❌ Error al leer la cámara.")
        break

    # Mostrar vista previa
    cv2.imshow("Captura de imágenes - Tilapia", frame)

    # Esperar tecla
    key = cv2.waitKey(1) & 0xFF

    # Si presionas 'c', captura y guarda imagen
    if key == ord('c'):
        nombre_archivo = f"tilapia_{contador:04d}.jpg"
        ruta = os.path.join(carpeta, nombre_archivo)
        cv2.imwrite(ruta, frame)
        print(f"💾 Imagen guardada: {ruta}")
        contador += 1

    # Si presionas 'q', salir del programa
    elif key == ord('q'):
        print("👋 Finalizando captura...")
        break

cap.release()
cv2.destroyAllWindows()
```

Codigo para entrenar modelo en Yolo v8

```
yolo train model=yolov8n.pt data=data.yaml epochs=100 imgsz=640
pip install ultralytics opencv-python
```

Script de detección en Raspberry

```
from ultralytics import YOLO
import cv2
import time

# Cargar el modelo entrenado
model = YOLO("/home/pi/tilapia_yolo/best.pt")

# Iniciar cámara
cap = cv2.VideoCapture(0)

# Ajustar resolución (opcional)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# Mostrar detección
while True:
    ret, frame = cap.read()
    if not ret:
        print("No se puede acceder a la cámara.")
        break

    # Detección
    start = time.time()
    results = model(frame)
```

```
    end = time.time()

    # Mostrar FPS aproximado
    fps = 1 / (end - start)
    print(f"FPS: {fps:.2f}")

    annotated_frame = results[0].plot()
    cv2.imshow("Tilapia Roja - Raspberry Pi", annotated_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Código completo (Raspberry Pi) — detección, tamaño, conteo y dispensar

```
# alimentador_auto.py
from ultralytics import YOLO
import cv2
import time
import math
import RPi.GPIO as GPIO
from collections import OrderedDict
import numpy as np

# ---------- CONFIG ----------
MODEL_PATH = "/home/pi/tilapia_yolo/best.pt"   # ruta a best.pt en RPi
CAM_INDEX = 0
```

```
REF_LENGTH_CM = 30.0   # longitud real de la regla que usarás para calibrar (cm)
GPIO_PIN_RELAY = 17    # pin BCM para relay que activa el motor/dispensador
DISPENSER_FLOW_G_PER_SEC = 2.0  # gramos por segundo que da tu dispensador (medir y ajustar)
FRACTION_PER_FEED = 0.01  # 1% del peso total en cada ración (ejemplo) o usa tu regla
# Allometric initial (solo como inicio; calibrar luego con datos reales)
a_init = 0.012
b_init = 3.0
# ------------------------------------


# Simple centroid tracker para evitar doble contar
class CentroidTracker:
    def __init__(self, maxDisappeared=10):
        self.nextObjectID = 0
        self.objects = OrderedDict()
        self.disappeared = OrderedDict()
        self.maxDisappeared = maxDisappeared

    def register(self, centroid):
        self.objects[self.nextObjectID] = centroid
        self.disappeared[self.nextObjectID] = 0
        self.nextObjectID += 1

    def deregister(self, objectID):
        del self.objects[objectID]
        del self.disappeared[objectID]
```

```
    def update(self, rects):
        # rects: list of bounding boxes [[x1,y1,x2,y2], ...]
        if len(rects) == 0:
            for oid in list(self.disappeared.keys()):
                self.disappeared[oid] += 1
                if self.disappeared[oid] > self.maxDisappeared:
                    self.deregister(oid)
            return self.objects

        inputCentroids = np.zeros((len(rects), 2), dtype="int")
        for (i, (x1,y1,x2,y2)) in enumerate(rects):
            cX = int((x1 + x2) / 2.0)
            cY = int((y1 + y2) / 2.0)
            inputCentroids[i] = (cX, cY)

        if len(self.objects) == 0:
            for i in range(0, len(inputCentroids)):
                self.register(inputCentroids[i])
        else:
            objectIDs = list(self.objects.keys())
            objectCentroids = list(self.objects.values())
            # distancia entre objetos y entradas
            D = np.linalg.norm(np.array(objectCentroids)[:,None] - inputCentroids[None,:],
axis=2)
            rows = D.min(axis=1).argsort()
            cols = D.argmin(axis=1)[rows]
```

```
            usedRows, usedCols = set(), set()
            for (row, col) in zip(rows, cols):
                if row in usedRows or col in usedCols:
                    continue
                objectID = objectIDs[row]
                self.objects[objectID] = inputCentroids[col]
                self.disappeared[objectID] = 0
                usedRows.add(row); usedCols.add(col)
            # check unused
            unusedRows = set(range(0, D.shape[0])) - usedRows
            unusedCols = set(range(0, D.shape[1])) - usedCols
            if D.shape[0] >= D.shape[1]:
                for row in unusedRows:
                    objectID = objectIDs[row]
                    self.disappeared[objectID] += 1
                    if self.disappeared[objectID] > self.maxDisappeared:
                        self.deregister(objectID)
            else:
                for col in unusedCols:
                    self.register(inputCentroids[col])
        return self.objects

# Setup GPIO relay
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_PIN_RELAY, GPIO.OUT)
GPIO.output(GPIO_PIN_RELAY, GPIO.LOW)
```

```
def dispense_by_grams(grams):
    """
    Activa el dispensador el tiempo necesario para entregar 'grams' gramos.
    Ajusta DISPENSER_FLOW_G_PER_SEC con pruebas reales.
    """
    if grams <= 0:
        return
    seconds = grams / DISPENSER_FLOW_G_PER_SEC
    print(f"Dispensing {grams:.1f} g -> {seconds:.2f} s")
    GPIO.output(GPIO_PIN_RELAY, GPIO.HIGH)
    time.sleep(seconds)
    GPIO.output(GPIO_PIN_RELAY, GPIO.LOW)

# Cargar modelo
model = YOLO(MODEL_PATH)
cap = cv2.VideoCapture(CAM_INDEX)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

pixels_per_cm = None  # calibrar con 'r'
tracker = CentroidTracker(maxDisappeared=15)

print("Presiona 'r' para calibrar con regla, 'f' para dispensar ración ahora, 'q' para salir")

while True:
    ret, frame = cap.read()
    if not ret:
```

```
    break
display = frame.copy()
results = model(frame)  # inferencia
dets = results[0].boxes  # ultralytics boxes
rects = []
sizes_cm = []
for box in dets:
    x1,y1,x2,y2 = map(int, box.xyxy[0].tolist())
    rects.append((x1,y1,x2,y2))
objects = tracker.update(rects)

total_weight_g = 0.0
# iterate current rects to estimate length/weight
for i, (x1,y1,x2,y2) in enumerate(rects):
    bbox_h_pix = y2 - y1
    if pixels_per_cm:
        length_cm = bbox_h_pix / pixels_per_cm
        weight_g = a_init * (length_cm ** b_init)   # usar parámetros calibrados
        total_weight_g += weight_g
        cv2.putText(display, f"{length_cm:.1f}cm {weight_g:.0f}g", (x1, y1-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0), 2)
    else:
        cv2.putText(display, "Calibrar regla (r)", (x1, y1-10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,0,255), 2)
    cv2.rectangle(display, (x1,y1), (x2,y2), (0,255,0), 2)

# mostrar conteo de objetos únicos (tracked)
```

```
    cv2.putText(display, f"Count (tracked): {len(objects)}", (10,30),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,255,0), 2)
    cv2.putText(display, f"Peso estim total: {total_weight_g:.0f} g", (10,60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,0), 2)


    cv2.imshow("Alimentador Auto", display)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
    if key == ord('r'):
        # Calibración manual: click dos puntos sobre la regla
        print("Click dos puntos sobre la regla en la ventana")
        pts = []
        def click_event(event, x, y, flags, param):
            if event == cv2.EVENT_LBUTTONDOWN:
                pts.append((x,y))
                cv2.circle(display, (x,y), 5, (0,0,255), -1)
                cv2.imshow("Alimentador Auto", display)
        cv2.setMouseCallback("Alimentador Auto", click_event)
        while len(pts) < 2:
            cv2.waitKey(1)
        cv2.setMouseCallback("Alimentador Auto", lambda *args: None)
        (x1r,y1r),(x2r,y2r) = pts
        pixels_ref = math.hypot(x2r-x1r, y2r-y1r)
        pixels_per_cm = pixels_ref / REF_LENGTH_CM
        print(f"Calibrado: {pixels_per_cm:.3f} pixels/cm")
    if key == ord('f'):
```

```
        # calcular racion a dispensar y activar
        if pixels_per_cm is None:
            print("Primero calibra con 'r'")
            continue
        # ejemplo: dar FRACTION_PER_FEED del peso estimado
        grams_to_dispense = total_weight_g * FRACTION_PER_FEED
        if grams_to_dispense <= 0:
            print("No se detectaron peces")
        else:
            dispense_by_grams(grams_to_dispense)

cap.release()
cv2.destroyAllWindows()
GPIO.cleanup()
```