Perfecto, aquí tienes una explicación **completa y profunda** directamente aquí en el chat sobre el funcionamiento del **modo autónomo** y su activación desde la interfaz:

🔁 Funciones automáticas

1. activar_modo_automatico()

- **Propósito:** Activa el modo automático del robot.
- **Cómo lo hace:**
    - Cambia la variable global modo_automatico a True.
    - Lanza la función modo_autonomo() en un nuevo hilo (con Thread(..., daemon=True)), para que se ejecute en segundo plano sin bloquear la interfaz.
- **Desde dónde se llama:** Desde un botón en la interfaz gráfica con texto "Auto ON".

2. desactivar_modo_automatico()

- **Propósito:** Desactiva el modo automático.
- **Cómo lo hace:**
    - Cambia modo_automatico a False, lo cual hará que el bucle while modo_automatico: en modo_autonomo() se detenga.
- **Desde dónde se llama:** Desde el botón "Auto OFF".

🤖 modo_autonomo() — El corazón del modo automático

Esta función ejecuta un bucle continuo mientras modo_automatico sea True. Dentro de este bucle hace lo siguiente:

Paso 1: Obtener la imagen actual

- Usa un mutex para acceder de forma segura a la variable compartida frame_actual.
- Si aún no hay frame, espera 0.5 segundos y sigue al siguiente ciclo.

Paso 2: Detección del cono amarillo

- Convierte el frame a espacio de color **HSV**.

- Crea una máscara que aísla el color amarillo usando cv2.inRange(hsv, lower_yellow, upper_yellow).
- Limpia la máscara con morfología (apertura y cierre) para eliminar ruido.
- Obtiene todos los contornos de objetos amarillos en la imagen.

Paso 3: Si no hay conos

- Si la lista de contornos está vacía, significa que no ve conos.
- Entonces, llama a mover("giro", 350) para hacer que el robot gire sobre su eje y busque conos.
- Espera un tiempo (tiempo_espera_movimiento) antes de continuar el bucle.

Paso 4: Si hay conos

- Toma el **contorno con mayor área** (el más grande) como el cono más relevante.
- Obtiene su rectángulo delimitador (x, y, w, h) y calcula el centro horizontal (cx).
- Calcula la distancia al cono en centímetros:

distancia_cm=ALTURA_CONO_REAL_CM×DISTANCIA_FOCALh×1.10\text{distancia_cm} = \frac{ALTURA_CONO_REAL_CM \times DISTANCIA_FOCAL}{h \times 1.10}

    - Donde ALTURA_CONO_REAL_CM = 9 y DISTANCIA_FOCAL = 550 (calibrados previamente).

Paso 5: Decisión de movimiento

Basado en la posición del cono (usando cx) y su distancia:

- **Centrado y cerca (18–19 cm):**
    - Considera que el robot ya alcanzó el cono.
    - Muestra un log: "¡Cono alcanzado! Quitar cono".
    - Desactiva modo_automatico y **termina** la función.
- **A la izquierda (cx < 220):**
    - Llama a mover("izquierda", 150) para corregir dirección.
- **A la derecha (cx > 420):**
    - Llama a mover("derecha", 150).

- **Centrado:**
    - Si el cono está **lejos** (más de 25 cm): avanza más tiempo → mover("adelante", 500).
    - Si está **más cerca**: avanza menos → mover("adelante", 300).
- Luego de cualquier movimiento, espera el tiempo definido en tiempo_espera_movimiento para permitir que el robot lo complete.

🧠 **Resumen de la lógica autónoma**

1. **Detecta conos** por su color.
2. **Calcula su posición y distancia.**
3. **Decide moverse** para centrarlo o avanzar.
4. **Se detiene** si el cono está lo suficientemente cerca.

🖼️ **Cómo se activa desde la interfaz (iniciar_interfaz)**

- La interfaz gráfica está hecha en **Tkinter**.
- Tiene:
    - video_frame: muestra el video en tiempo real.
    - log_console: muestra eventos y logs.
    - Controles manuales: botones para movimiento.
    - Botones especiales:
        - "Auto ON" llama a activar_modo_automatico().
        - "Auto OFF" llama a desactivar_modo_automatico().

Además, lanza otro hilo para mostrar_video_udp, que actualiza frame_actual constantemente.

¿Quieres que también profundice en la función mover() para ver cómo se ejecutan esos movimientos en físico o por red?