📄 MEMORIA TÉCNICA – RECONOCIMIENTO DE CARTAS CON VISIÓN ARTIFICIAL

1. Hardware

Dispositivos empleados

Componente	Detalle
Cámara	Smartphone mediante DroidCam – 720p/1080p
Ordenador	PC para procesamiento en Python
Iluminación	Luz blanca uniforme
Superficie	Tapete verde (alto contraste con las cartas)

Características relevantes

Requisito técnico	Justificación
Resolución ≥ 720p	Necesaria para capturar detalles de número y palo
Sensor con auto-enfoque	Permite detección nítida en tiempo real
Fondo homogéneo	Reduce ruido visual y mejora segmentación
Iluminación estable	Evita sombras y pérdidas de contorno

Justificación del hardware

- La cámara del móvil proporciona **alta resolución y enfoque automático**, mucho más fiable que cámaras USB baratas.
- El tapete verde contrasta con las cartas blancas → la segmentación se vuelve mucho más estable.
- El PC permite procesamiento en tiempo real con OpenCV (~30 FPS).

---

2. Software

Elemento	Detalle
Lenguaje	Python 3.11+
Librerías	OpenCV, NumPy
Gestor cámara	DroidCam
Sistema operativo	Windows 10/11

Justificación de uso

Software	Motivo
Python	Fácil prototipado y gran soporte IA/visión
OpenCV	Librería estándar para procesamiento de imagen
NumPy	Operaciones matemáticas y matrices optimizadas
DroidCam	Convierte el móvil en webcam de alta calidad sin hardware extra

---

3. Hoja de Ruta del Desarrollo

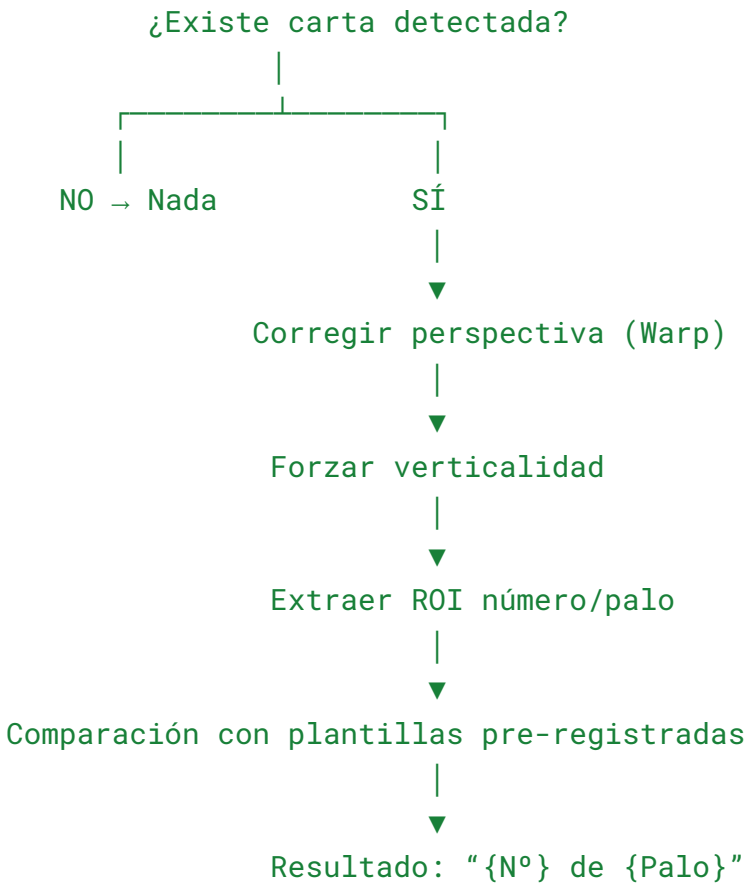
Fase	Actividad	Resultado
1	Captura de cámara y segmentación	Detección inicial de carta
2	Creación de plantillas	Recorte número/palo desde cartas reales
3	Warp + corrección perspectiva	Normalización independ. del ángulo
4	Extracción de ROI	Obtención de número y palo
5	Comparación por correlación	Reconoce carta en vivo

La base del proyecto es que el sistema pueda tomar **una carta en cualquier orientación** y reconstruirla verticalmente para leer sus símbolos.

---

4. Solución Final

🔽 *Diagrama de decisión de clasificación*



---

5. Secuencia de operaciones sobre la imagen

Etapa	Técnica	Parámetros	Justificación
1) Umbral adaptativo	`cv2.ADAPTIVE_THRESH_GAUSSIAN_C`	blockSize=21, C=7	Detecta carta incluso con sombras
2) Contornos	`cv2.findContours`	RETR_EXTERNAL	Nos quedamos solo con el objeto más grande
3) Aproximación poligonal	`approxPolyDP`	0.02 * perímetro	Buscar carta como cuadrilátero

4) Warp perspective	`getPerspectiveTransform + warpPerspective`	salida 200x280 px	Normaliza escala de todas las cartas
5) Auto-verticalización	`if width > height → rotate`	—	Soluciona inclinación >60º
6) Extracción ROI	Recortes 2–18% altura/ancho	calibrado visual	Zona donde SIEMPRE está número/palo
7) Matching plantillas	`cv2.matchTemplate` normalizada	método TM_CCOEFF_NORMED	Reconocimiento robusto por similitud

---