📌 ICT 어워즈 시연 패키지 (대본·Q&A·학습설정·코드해설·운영체크리스트)

 # 0) 한눈 개요

- 우리가 만든 것은 "스마트팩토리 데모"용 델타로봇+비전 시스템.
- 입력: 웹/앱(STT·수동)으로 모델 전환 → 웹캠 실시간 → YOLO 추론 → 3초 투표(voting) → 1바이트 패킷 전송(UART) → OpenRB-150 동작(흡착/이동).
- 현장 조명 제약 때문에 **시연은 트럼프카드(객체탐지)** 중심. (분류모델은 조명 민감 → 별도 환경에서 데모 권장)

 # 1) 대본 — 개념(2~3분)

🎤 오프닝

"안녕하세요. 저희는 '스마트팩토리의 미니 버전'을 컨셉으로 한 델타로봇 데모를 준비했습니다. 카메라가 컨베이어 벨트 위의 물체를 보고 AI로 판별하고, 결과를 시리얼 1바이트 패킷으로 컨트롤러(OpenRB-150)에 보내 흡착·이송하는 흐름입니다."

🧠 핵심 구성

- 비전: YOLOv8 (객체탐지/분류)
- 로봇: Delta + AX-12 x3 + 릴레이(흡착 on/off)
- 통신: UART(TTL) 115200bps, 1바이트 심볼(S/C/H/D)
- UI: 웹(브라우저) / 앱(모델 전환) / STT로 "모델 교체"

🃏 왜 트럼프카드로 시연?

- **탐지(detector) 모델**은 상자(box)로 물체를 찾아내 조명 변화에 상대적으로 강인.
- **분류(classifier) 모델**은 프레임 전체를 보고 판단 → 조명·반사에 민감.
- 현장 조명(지하, 반사) 영향으로 오늘은 탐지 기반의 트럼프 카드 위주로 안정 데모.

🧲 흡착기와 물체 크기

- 흡착기의 한계상, 종이/카드류가 가장 안정적이라 카드로 데모.
- 분류 모델은 실물(캔/병/육류 등)을 써야 조명 민감성이 훨씬 줄어듦(후속 확장 포인트).

🔌 시리얼 패킷 1바이트

- 지연 최소화를 위해 결과를 '한 글자'로 압축: S(Spade)/C(Club)/H(Heart)/D(Diamond) 등.
- 컨트롤러는 BUSY/READY 핸드셰이크로 유실 방지.

 # 2) 대본 — 심화(5~7분)

🧩 아키텍처

1) **웹/앱**: 모델 전환 API 호출(/v1/load_model), STT 키워드→모델 매핑

2) **FastAPI 서버(main.py)**: YOLO 로딩·추론, 3초 voting, cooldown, 1바이트 패킷 송신

3) **OpenRB-150**: 패킷 수신 → 티칭 포즈로 부드럽게 이동 → 흡착 on/off → LCD 카운트 표기

🧪 추론 안정화 로직

- **3초 Voting**: 프레임별 점수 누적 → 최종 Top1만 1회 전송
- **Cooldown**: 다음 트리거까지 최소 대기(중복 전송 방지)
- **그린 게이팅(선택)**: 벨트가 초록색만 보일 땐 대기, '다른 색 영역이 0.5s 이상 지속' 시 Voting 시작
- **ROI(검은 마스킹)**: 양쪽 프레임 컷(노이즈 절감, 처리량 감소)

🖥️ 모델 타입별 특성

- **YOLO Detection(카드)**: 위치+클래스 동시 예측 → 배경·조명 변화에 상대적 강건
- **YOLO Classification(음료/육류/재활용)**: 이미지 전체 평균화 → 반사·광량에 취약.

  ⇒ 실물 대상 + 안정 조명 + 벨트 동일 세팅에서 성능 ↑

🛠️ 하드웨어 팁

- U2D2 + Dynamixel Wizard로 AX-12 초기화/토크값 정렬
- 오프시 낙하 속도 제어를 위해 흡착 호스 말단 밸브 조절
- 카메라 링조명 + 가림막 + 양측 마스킹(검정)으로 환경 변수 축소
- 델타·벨트·카메라 모든 상대 위치 고정 (티칭 좌표 보호)

 # 3) Q&A 핸드카드

Q. 왜 카드만 시연하나요?

A. 현장 조명·반사로 분류모델이 민감합니다. 카드는 탐지모델을 써서 견고하고, 흡착기도 카드 두께/무게에서 가장 안정적입니다.Q. YOLOv8을 고른 이유?

A. 학습·배포가 단순하고, 탐지/분류 모두 지원, s/m/l로 경량~고정밀 트레이드오프 선택 가능. 커뮤니티/도구가 풍부합니다.Q. Voting/Cooldown이 뭔가요?

A. 프레임마다 결과가 흔들릴 수 있어 **3초 누적투표**로 안정화 후 1회만 패킷 전송, **Cooldown**으로 중복 방지합니다.Q. UART(TTL)를 쓴 이유?

A. 가장 단순·가벼운 비동기 직렬. 1바이트 패킷만 필요해 대역·지연 모두 유리, 라이브러리/툴도 풍부(115200bps).Q. 향후 개선?

A. 실물 데이터 재학습, 조명 차폐, 그린-게이팅 강화, 모델별 동작 딜레이 보정(보정 테이블), Jetson급 엣지 이식.

---

4) 학습 설정 & 용어(쉬운 말 + 한 줄 심화)

📦 분류(예: Beverage)

- 명령: `yolo classify train data=beverage_cls.yaml model=yolov8s-cls.pt imgsz=224 epochs=60 batch=32 device=0`
- 이유: s-클래스 가벼움, 224 입력으로 빠르게 수렴, 60epoch 전후에서 과적합 전 성능 안정.

🃏 탐지(예: Playing Cards)

- 권장: `yolov8s.pt`, imgsz 640, epochs 80~120, batch 16~32
- 팁: 조명 유사 환경에서 촬영해 라벨링, **mosaic/augment 완화**(반사에서 과적합 방지), val셋은 다른 조명.

📚 용어 요약

- **YOLO**: CNN 기반 단일 스테이지 탐지(End-to-End로 박스+클래스 동시 예측)
- **분류(Classification)**: 이미지 전체 → 라벨 하나
- **탐지(Detection)**: 박스 여러 개 + 각 라벨
- **Voting**: 프레임들의 점수 합산으로 최종 결정(지터 방지)
- **Cooldown**: 1회 송신 후 일정 시간 입력 무시(중복/오동작 방지)
- **UART 8N1**: Start(0) + 8비트 + Stop(1), 패리티 없음. 가장 보편적 직렬 규약.

---

5) 코드 해설 (두 파일 핵심만)

---

(1) main.py — FastAPI(추천 구조: 옵션 A)

- `/v1/load_model` : 앱/웹(STT)이 호출 → 모델 코드로 YOLO 가중치 로드
- `/v1/predict` : (웹 실시간 오버레이를 쓰는 경우) 프레임 업로드 → 즉시 추론
- 내부:
  - MODEL_PATHS / CLASSNAMES / label→packet 매핑
  - 3초 voting & cooldown 로직
  - Mac은 `device="mps"`로 Metal 가속
  - 시리얼 포트(예: `/dev/cu.usbserial-0001`, 115200bps) 열고 1byte 송신
  - READY/ BUSY 핸드셰이크(유실 방지) 지원(2) OpenRB-150(Arduino) — 델타 제어
- `Serial2`로 호스트 수신: 'S/C/H/D' 1바이트 → 티칭 포즈로 부드럽게 이동
- 이동 시퀀스:
  기본→픽업→기본→목표(스페이드/클럽/하트/다이아)→서브포즈→기본 복귀
- `smoothMoveToPose()` : cos 이징으로 진동 최소화
- LCD: 모델/카운트 갱신, 'READY' 전송로 다음 패킷 타이밍 제어

- 모델 전환('1~4') 입력 시 LCD·카운트 초기화

---

9) 우리가 실제로 한 학습(예시 값)

- Beverage(분류): yolov8s-cls, 224, **epochs=60**, batch=32
- Cards(탐지): yolov8s, 640, **epochs≈100**, batch 16~32 (라벨 품질/광량 맞춤)
- 데이터: 벨트 위 실물/인쇄물 촬영, 링조명 켠 상태로 수집(현장 유사 광량 맞춤)
- 증강: 과한 색상 변화는 반사 환경에서 오히려 역효과 → 약하게

---

10) 왜 UART(시리얼, TTL)?

- 하드웨어/펌웨어에서 가장 널리 쓰이고, 드라이버/툴(파이썬 pyserial, 아두이노) 생태계가 견고
- 클럭 없는 비동기, 프레이밍 간단(8N1), 배선 3가닥(TX/RX/GND)로 충분
- 이번 프로젝트처럼 "1바이트 트리거"에 최적: 저지연·저부하·디버깅 쉬움

🎯개요요약30초

우리는 스마트팩토리 데모용 델타 로봇 라인을 만들었다. 카메라가 컨베이어 위 물체를 보고 YOLOv8으로 판단 → UART(시리얼) 1바이트 패킷 전송 → OpenRB-150이 Dynamixel(AX-12) 3축과 흡착 릴레이를 구동해 분류/이동. 오늘은 조명 영향 때문에 **트럼프카드(탐지 모델)**만 시연한다. (음료/재활용/고기등급은 분류 모델 + 5초 투표 구조라 조명 변화에 약함)

🎙무대용 대본 — 개념 버전 (2분)

"저희는 스마트팩토리 '축소 실증 라인'을 만들었습니다. 컨베이어 위 물체를 카메라로 보고, AI가 판별한 뒤, 로봇이 자동으로 픽업→이송→배출까지 합니다. 모델 선택은 앱/웹(STT)으로 바꿀 수 있고, 서버가 모델을 로딩한 뒤 결과를 시리얼(UART)로 로봇에 전달합니다. 오늘은 트럼프카드만 시연합니다. 이 모델은 객체탐지(Detection) 기반이라 조명이 조금 바뀌어도 비교적 안정적입니다. 반면 음료/재활용/고기등급은 분류(Classification) 기반이라 5초간 투표(Voting) 후 1바이트를 보내는데, 조명(특히 반사)에 민감해 현장 환경에선 정확도가 떨어졌습니다. 하드웨어적으로도 흡착기의 한계 때문에 얇고 가벼운 카드 사이즈가 가장 안정적입니다. 분류 모델을 제대로 보여주려면 인쇄물이 아닌 '실물'로 하면 조명 영향이 줄어듭니다. 초록색 벨트만 보도록 ROI(마스킹)를 했고, 카드가 들어오면 Voting + Cooldown으로 중복 명령을 막습니다. 로봇은 티칭 좌표 고정, 벨트 속도/프로파일 고정, 흡착 낙하 속도는 밸브로 조절, 웹캠에는 원형 조명을 달아 흔들림을 최소화했습니다. 그럼 시연 바로 보시겠습니다."

🔬 무대용 대본 — 심화 버전 (3~4분)

연결 구조: 카메라 → 맥북 YOLOv8 추론 → 결과를 **1바이트(UART 115200, 8N1)**로 OpenRB-150 전송 → MCU는 BUSY/READY, 쿨다운, 중복 필터로 단발 동작 보장 → Dynamixel 3축을 코사인 이징으로 부드럽게 이동 + 릴레이로 흡착/해제. 탐지 vs 분류: - 카드 = 탐지(bbox) → 배경/조명 변화에 상대적으로 강인. - 음료/재활용/고기 = 분류(확률 벡터) → 5초 누적 투표 후 1위만 송신. 반사/광량에 민감. ⇒ 오늘은 카드(탐지)만 안정 시연. Voting & Cooldown: - 물체 들어오면 N초(예: 5초) 동안 프레임 점수 누적 → 최다 득표만 1바이트(s/c/h/d) 전송 - 이후 쿨다운으로 추가 전송 금지(중복·채터링 억제) ROI/Color Gate(옵션): - 초록 벨트만 보도록 좌우 마스킹 + 초록 비율이 무너지면(비초록 0.5s 지속) 그때만 투표 시작

💬 Q&A 핸드카드

Q. 왜 카드만 시연하나요?

A. 카드 모델은 '탐지'라 조명 변화에 강하고, 흡착기에 카드가 가장 안정적입니다. 분류 모델은 5초 투표 구조인데 현장 조명(지하/반사)에서 민감했습니다.

Q. YOLOv8을 왜 택했나요? CNN 기반인가요?

A. 네, CNN 백본 + 태스크별 헤드(탐지/분류/세그). 속도/정확도/툴링이 좋아 실시간 데모와 이식성이 뛰어납니다.

Q. UART를 고른 이유는?

A. 단순·가벼움·저지연. 1바이트만 보내면 수 ms 내 반응, 8N1/115200으로 충분합니다.

Q. Voting은 무엇인가요?

A. 프레임별 예측을 시간창에서 누적해 최다 득표 클래스를 1회 전송하는 안정화 기법입니다.

Q. 좌표는 어떻게 안정화했나요?

A. 티칭 좌표 고정, 델타/벨트 상대 위치 고정, 속도·프로파일 고정, 코사인 이징으로 부드럽게 이동, 흡착 낙하 속도는 밸브로 조절했습니다.

학습(Training) 설정 & 용어 풀이 (예시 값)

[카드 탐지 — YOLOv8s]

model: yolov8s.pt (COCO 프리트레인) - imgsz: 640 / epochs: 80~100 / batch: 16 - 증강: 밝기/대비·HSV 약하게, 좌우 뒤집기, mosaic 약화(반사 환경에서 과증강 금지)

-데이터: 벨트 위에서 실제 조명 ON 상태로 촬영/라벨링 명령

예: yolo detect train data=cards.yaml model=yolov8s.pt imgsz=640 epochs=100 batch=16 [음료/재활용/고기 분류 — YOLOv8s-cls

- model: yolov8s-cls.pt - imgsz: 224 / epochs: 50~80(예: 60) / batch: 32 명령

예: yolo classify train data=beverage_cls.yaml model=yolov8s-cls.pt imgsz=224 epochs=60 batch=32

[용어 미니사전]

Epoch: 데이터셋 전체 1회 학습

Weights(.pt): 학습된 파라미터

Backbone/Head: 특징 추출부/태스크 출력부

ROI: 관심 영역(벨트 중앙만 보기)

Voting/Cooldown: 시간 누적 다수결 / 재전송 금지 구간

UART 8N1: Start(0) + 8비트 + Stop(1), 패리티 없음