

## ARDUINO 코드

```
#include <Servo.h>

Servo myservo_LR; // 좌우
Servo myservo_UD; // 상하

int pin_servo_LR = 9;
int pin_servo_UD = 10;

void setup() {
    Serial.begin(9600);
    myservo_LR.attach(pin_servo_LR);
    myservo_LR.write(90); // 초기 위치
    myservo_UD.attach(pin_servo_UD);
    myservo_UD.write(90);
}

void loop() {
    if (Serial.available()) {
        String data = Serial.readStringUntil('\n');
        int comma = data.indexOf(',');
        if (comma > 0) {
            int angle_x = data.substring(0, comma).toInt(); // 좌우
            int angle_y = data.substring(comma + 1).toInt(); // 상하

            angle_x = constrain(angle_x, 0, 180);
            angle_y = constrain(angle_y, 0, 180);

            myservo_LR.write(angle_x);
            myservo_UD.write(angle_y);
        }
    }
}
```

## VISUAL STUDIO 코드

```
from ultralytics import YOLO
import cv2
import serial
import time
# 1. 모델 로드
model = YOLO("best.pt")
# 2. 아두이노 연결(COM 포트 확인 필요)
arduino = serial.Serial('COM14', 9600)
time.sleep(2)
# 3. 카메라 시작
cap = cv2.VideoCapture(0)
# 4. 카메라 해상도 플러오기
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
# 5. 아두이노로 데이터 전송 함수
def send_to_arduino(x, y):
    try:
        # 화면 좌표를 0~180 범위의 서로 각도로 변환
        angle_x = int((x / frame_width) * 180)
        angle_y = int((y / frame_height) * 180)
        data = f'{angle_x},{angle_y}\n'
        arduino.write(data.encode())
        print(f'전송 X: {angle_x}, Y: {angle_y}')
    except:
        print("아두이노 전송 오류")
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # 6. YOLO로 예측
    results = model(frame)
    for result in results:
        for box in result.boxes:
            cls = int(box.cls[0])
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            cx, cy = ((x1 + x2) // 2, (y1 + y2) // 2)
            # 경시면 클래스가 volleyBall일 때
            if model.names[cls] == "volleyball":
                cv2.circle(frame, (cx, cy), 5, (0, 255, 0), -1)

            # 6. 공의 중심 좌표를 아두이노로 전송
            send_to_arduino(cx, cy)
            # 7. 트래킹 시작화
            cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)
            cv2.putText(frame, f"volleyBall", (x1, y1 - 10),
                       cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
    # 8. 화면 표시
    cv2.imshow("Ball Tracking", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    # 9. 종료 처리
    cap.release()
    cv2.destroyAllWindows()
    arduino.close()
```

## 참고 문헌

**Ultralytics YOLO 공식 문서** – best.pt를 제작하기 위한 참고 문서

**OpenCV 문서** – visual studio 코드를 작성하기 위한 참고 문서

**Arudino 공식 문서** – 아두이노 코딩과 본체를 만들기 위한 참고 문서

**Roboflow Blog** – YOLO모델의 새로운 카테고리를 만들기 위한 블로그

**Google** – 다양한 잡지식과 심화탐구를 하기위한 웹

## 동기 및 목적

최근 인공지능 기술과 영상처리 기술이 발전면서, 실시간 물체 인식과 제어 기술이 로봇공학, 스마트카, 드론 등 다양한 분야에서 활용되고 있는 것을 보았다. 이에 착안하여, 카메라로 공을 인식하고 공의 위치에 따라 서보모터가 움직이도록 하는 시스템을 구현하고자 하였다. 이 프로젝트의 목표는 YOLO 객체 탐지 모델을 통해 공을 실시간으로 인식하고, 그 좌표를 Arduino 서보모터에 전달하여 자동으로 반응하는 것이다.

## 탐구 방법

YOLO 모델을 활용하기 위해 Ultralytics YOLO를 사용하여 배구공을 인식하도록 인공지능을 학습시킨다. 기본적으로 제시되어 있는 객체가 아닌 새로운 카테고리를 만들어야 한다. OpenCV를 활용해 웹캠 실시간 영상을 입력받고, 인식 결과를 시각적으로 출력한다. 시리얼 통신을 통해 Python과 Arduino를 통신하여 서로 정보를 주고 받으며 인식된 공의 중심 좌표에 맞게 두 개의 서보모터를 이용하여(X축, Y축) 제어한다.

## 탐구 결과

웹캠이 공을 인식할 때, 서보모터가 공의 위치에 따라 자동으로 회전하는 것을 확인할 수 있다. YOLO 모델의 신뢰도를 0.7 이상으로 설정할 때 인식 정확도가 높게 유지된다. 좌표 데이터 전송 및 서버 모터 반응 속도를 확인할 때 인식된 공의 중심 좌표를 시리얼 통신 (9600bps)으로 아두이노에 전달한다. 평균 데이터 전송 속도는 약 0.09초, 서보모터의 회전 반응 시간은 약 0.12초로 확인된다. 결과적으로 카메라 입력부터 모터 반응까지의 전체 지연 시간은 약 0.2초 이내로 측정되었다. 서보모터 움직임의 안정성으로 좌우 및 상하 서보모터가 공의 중심 위치에 따라 부드럽게 회전한다. 다만 공이 화면 가장자리로 이동할 경우, 좌표 변화량이 급격해 모터가 빠르게 진동하는 현상이 발생했다. 그리고 카메라가 아닌 웹캠을 사용하여서 화질, 프레임이 낮아진다. 이로 인해 공을 인식하지 못하거나 자연스럽게 알고리즘이 작동하지 않는 경우도 생기게 된다.

