Cloud Computing Inferece

이 문서는 AIoT 메이커톤을 준비하는 참가자들이 반드시 알아야 할 기술 개념과, 실제 프로젝트에 사용된 코드들을 설명합니다.

개념 설명

1. IP / 포트

- **IP 주소**: 네트워크에서 각각의 기기를 식별하기 위한 고유 주소.
- **포트(Port)**: 하나의 IP 주소에서 여러 서비스에 접근할 수 있게 해주는 창구.
  예: 웹은 보통 80번, HTTPS는 443번, 본 프로젝트는 3001번 사용.

---

2. WebSocket

- **WebSocket**은 브라우저와 서버가 **쌍방향으로 실시간 통신**할 수 있는 기술.
- HTTP는 요청에 대해서만 응답하지만, WebSocket은 연결을 유지하며 계속 데이터 송수신 가능.

---

3. FastAPI

- Python 기반 초고속 웹 프레임워크.
- REST API와 WebSocket을 쉽게 만들 수 있으며 문서 자동 생성 기능도 있음.

---

4. YOLO (You Only Look Once)

- 실시간 객체 인식 알고리즘.
- 이미지를 한 번만 살펴보고 어떤 객체가 어디에 있는지 빠르게 예측.
- 최신 버전 YOLOv11은 경량화 + 정확도 개선이 특징.

---

5. Object Detection

- 이미지나 영상에서 **사람, 고양이, 자동차 등** 객체를 탐지하고 위치를 알아냄.
- YOLO, SSD, Faster R-CNN 등이 대표적인 알고리즘.

Python 서버 코드 - `main.py`

```
from websockets.server import serve
from websockets.exceptions import ConnectionClosedOK
import cv2
import numpy as np
from fastapi import FastAPI, WebSocket, WebSocketDisconnect
from fastapi.responses import StreamingResponse
import os
```

```
from ultralytics import YOLO

app = FastAPI()

PLACEHOLDER_PATH = "placeholder.jpg"
IMAGE_PATH = "image.jpg"

# Initialize YOLO model with optimized settings
model = YOLO('yolo11n.pt')  # Latest YOLO11 model
model.overrides['verbose'] = False

def is_valid_image(image_bytes):
    try:
        nparr = np.frombuffer(image_bytes, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
        return img is not None
    except Exception as e:
        print("image invalid:", e)
        return False

def mjpeg_generator():
    while True:
        try:
            with open(IMAGE_PATH, "rb") as f:
                image_bytes = f.read()
            nparr = np.frombuffer(image_bytes, np.uint8)
            img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
            results = model(img, conf=0.25, iou=0.45, verbose=False, imgsz=640)
            annotated_img = results[0].plot()
            _, img_encoded = cv2.imencode('.jpg', annotated_img)
            img_bytes = img_encoded.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + img_bytes + b'\r\n')
        except Exception as e:
            print("encountered an exception: ", e)
            if os.path.exists(PLACEHOLDER_PATH):
                try:
                    with open(PLACEHOLDER_PATH, "rb") as f:
                        image_bytes = f.read()
                    nparr = np.frombuffer(image_bytes, np.uint8)
                    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
                    img_bytes = img.tobytes()
                    yield (b'--frame\r\n'
                           b'Content-Type: image/jpeg\r\n\r\n' + img_bytes +
b'\r\n')
                except Exception as e2:
                    print("placeholder error:", e2)
            import time
            time.sleep(0.1)

@app.get("/")
def index():
    return StreamingResponse(mjpeg_generator(), media_type='multipart/x-mixed-
replace; boundary=frame')
```

```
async def ws_handler(websocket: WebSocket):
    await websocket.accept()
    try:
        while True:
            message = await websocket.receive()
            if message["type"] == "websocket.disconnect":
                break
            if "bytes" in message and message["bytes"] is not None:
                data = message["bytes"]
                print(len(data))
                if len(data) > 5000:
                    if is_valid_image(data):
                        with open(IMAGE_PATH, "wb") as f:
                            f.write(data)
            elif "text" in message and message["text"] is not None:
                print("Text message:", message["text"])
            print()
    except WebSocketDisconnect:
        print("WebSocket disconnected.")
    except ConnectionClosedOK:
        print("WebSocket connection closed cleanly.")
    except Exception as e:
        print(f"WebSocket error: {e}")

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await ws_handler(websocket)

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=3001)
```

ESP32-CAM 코드 - esp32Stream.ino

```
#include <Arduino.h>
#include "esp_camera.h"
#include <WiFi.h>
#include <ArduinoWebsockets.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "fb_gfx.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "driver/gpio.h"

#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
```

```
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

const char* ssid     = "cycnus";
const char* password = "14714714";

const char* websockets_server_host = "192.168.0.15";
const uint16_t websockets_server_port = 3001;

using namespace websockets;
WebsocketsClient client;

void onMessageCallback(WebsocketsMessage message) {
  Serial.print("Got Message: ");
  Serial.println(message.data());
}

esp_err_t init_camera() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 15;
  config.fb_count = 2;

  esp_err_t err = esp_camera_init(&config);
```

```
  if (err != ESP_OK) {
    Serial.printf("camera init FAIL: 0x%x", err);
    return err;
  }
  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_VGA);
  Serial.println("camera init OK");
  return ESP_OK;
};

esp_err_t init_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi OK");
  client.onMessage(onMessageCallback);
  bool connected = client.connect(websockets_server_host, websockets_server_port,
"/ws");
  if (!connected) {
    Serial.println("WS connect failed!");
    Serial.println(WiFi.localIP());
    return ESP_FAIL;
  }
  client.send("hello from ESP32 camera stream!");
  return ESP_OK;
};

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.begin(9600);
  Serial.setDebugOutput(true);
  init_camera();
  init_wifi();
}

void loop() {
  if (client.available()) {
    camera_fb_t *fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("img capture failed");
      esp_camera_fb_return(fb);
      ESP.restart();
    }
    client.sendBinary((const char*) fb->buf, fb->len);
    Serial.println("image sent");
    esp_camera_fb_return(fb);
    client.poll();
  }
}
```