식물 생육 변화 사진 분석

조장 김준형
조원 오현준
조원 정도영

목차

- 프로젝트 개요 및 목표
- 시스템 아키텍처
- 주요 컴포넌트 소개
- 핵심 기능 시연
- 개발 과정 및 기술 스택
- 결론 및 향후 과제
- Q&A

프로젝트 개요 및 목표

프로젝트 개요:

* ESP32-CAM을 이용한
* Yolo 기반의 정확한 객체 탐지 및 식물 생육과정 구별 처리 시스템 구현
* Backend, Arduino, DB간의 효율적인 통신 구축

주요 목표:

* ESP32-CAM 으로 찍은 사진을 Spring Boot를 이용해 로컬 DB에 저장
* Flask를 이용한 Spring Boot와 욜로간의 통신(사진 파일 전송 및 결과 수신)
* 욜로가 해당 이미지를 통해 토마토의 레벨측정 및 정확도 판별

CameraWebServer.ino

```
#include "esp_camera.h"
#include "WiFi.h"
#include "HTTPClient.h"

// --- 🔧 사용자가 수정해야 할 부분 ---
const char* ssid = "makerland";
const char* password = "24132413";
const char* server_url = "http://192.168.0.116:8080/api/photos";
const char* deviceSerialNumber = "111-222-333";
// ------------------------------------

// ESP32-CAM 핀 설정 (AI-THINKER 모델 기준)
#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

```

```
// 카메라 초기화 함수
bool initCamera() {
  Serial.println("카메라 초기화 시작...");

  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 = 20;
  config.fb_count = 1;

```

```
  // 카메라 초기화
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("❌ 카메라 초기화 실패, 오류 코드: 0x%x\n", err);
    return false;
  }
  Serial.println("✅ 카메라 초기화 성공!");

  // ⭐⭐⭐ 180도 회전 설정 ⭐⭐⭐
  sensor_t * s = esp_camera_sensor_get();
  s->set_vflip(s, 1);   // 상하 반전
  s->set_hmirror(s, 1); // 좌우 반전
  Serial.println("✅ 카메라 180도 회전 설정 완료");

  return true;
}

// 📤 서버로 사진 업로드 함수
bool uploadPhoto() {
  Serial.println("📤 서버 업로드 시작...");

  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("❌ 사진 촬영 실패!");
    return false;
  }

  Serial.printf("📷 사진 촬영 완료: %u bytes\n", fb->len);

  HTTPClient http;
  http.begin(server_url);
  http.setTimeout(15000);

  String boundary = "----WebKitFormBoundary7MA4YWxkTrZu0gW";
  String formDataStart = "";

```

```
  // 디바이스 시리얼 넘버 필드
  formDataStart += "--" + boundary + "\r\n";
  formDataStart += "Content-Disposition: form-data; name=\"serialNumber\"\r\n\r\n";
  formDataStart += deviceSerialNumber;
  formDataStart += "\r\n";

  // 이미지 파일 헤더
  formDataStart += "--" + boundary + "\r\n";
  formDataStart += "Content-Disposition: form-data; name=\"imageFile\"; filename=\"camera.jpg\"\r\n";
  formDataStart += "Content-Type: image/jpeg\r\n\r\n";

  String formDataEnd = "\r\n--" + boundary + "--\r\n";

  size_t totalSize = formDataStart.length() + fb->len + formDataEnd.length();

  http.addHeader("Content-Type", "multipart/form-data; boundary=" + boundary);

  uint8_t* postData = (uint8_t*)malloc(totalSize);
  if (!postData) {
    Serial.println("❌ 메모리 할당 실패!");
    esp_camera_fb_return(fb);
    return false;
  }

  size_t offset = 0;
  memcpy(postData + offset, formDataStart.c_str(), formDataStart.length());
  offset += formDataStart.length();
  memcpy(postData + offset, fb->buf, fb->len);
  offset += fb->len;
  memcpy(postData + offset, formDataEnd.c_str(), formDataEnd.length());

  Serial.println("📨 서버 전송 중...");
  int httpResponseCode = http.POST(postData, totalSize);

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.printf("✅ HTTP 응답 코드: %d\n", httpResponseCode);
    Serial.printf("📄 서버 응답: %s\n", response.c_str());
```

```
    if (httpResponseCode == 201) {
      Serial.println("🎉 업로드 완료!");
    }
  } else {
    Serial.printf("❌ 업로드 실패! 오류 코드: %d\n", httpResponseCode);
    Serial.printf("❌ 오류 상세: %s\n", http.errorToString(httpResponseCode).c_str());
  }

  free(postData);
  esp_camera_fb_return(fb);
  http.end();

  Serial.printf("✓ 메모리 정리 완료, 현재 힙: %d bytes\n", ESP.getFreeHeap());
  return (httpResponseCode == 201);
}

void setup() {
  Serial.begin(115200);
  delay(2000);

  Serial.println("=== ESP32-CAM 사진 업로드 시스템 (회전 적용) ===");

  if (!initCamera()) {
    Serial.println("❌ 카메라 초기화 실패! 재시작합니다...");
    delay(5000);
    ESP.restart();
  }

  Serial.println("\nWiFi 연결 시작...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\n✅ WiFi 연결 성공!");
  Serial.printf("IP 주소: %s\n", WiFi.localIP().toString().c_str());

  Serial.println("\n=== 초기화 완료 ===");
}
```

```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\n--- 📸 사진 촬영 및 업로드 ---");
    uploadPhoto();
  } else {
    Serial.println("❌ WiFi 연결 끊어짐, 재연결 시도...");
    WiFi.reconnect();
  }

  Serial.printf("💤 30초 대기 중... (현재 힙: %d bytes)\n", ESP.getFreeHeap());
  delay(30000);
}

```

PhotoController.java

```java
package com.metaverse.planti_be.photo.controller;

import com.metaverse.planti_be.photo.dto.PhotoRequestDto;
import com.metaverse.planti_be.photo.dto.PhotoResponseDto;
import com.metaverse.planti_be.photo.service.PhotoService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;

@RestController
@RequestMapping("/api")
@RequiredArgsConstructor // 생성자 주입을 위한 Lombok 어노테이션
public class PhotoController {

    private final PhotoService photoService;

    @PostMapping("/photos")
    public ResponseEntity<PhotoResponseDto> uploadPhoto(
            @ModelAttribute PhotoRequestDto requestDto) throws IOException {

        // 서비스 호출
        PhotoResponseDto responseDto = photoService.savePhoto(requestDto);

        // 생성 성공 시, 201 CREATED 상태 코드와 생성된 리소스의 DTO를 함께 반환
        return ResponseEntity.status(HttpStatus.CREATED).body(responseDto);
    }
}
```

Photo.java

```java
package com.metaverse.planti_be.photo.domain;

import com.metaverse.planti_be.common.TimeStamped;
import com.metaverse.planti_be.device.domain.Device;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;

@Entity
@Getter
@NoArgsConstructor
public class Photo extends TimeStamped {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "device_serial", referencedColumnName = "serial_number")
    private Device device;

    @Column(nullable = false, length = 512) // 파일 경로 길이를 고려해 넉넉하게 설정
    private String filePath;

    @Column(nullable = false)
    private String fileName;

    // 직접 선언
    public Photo(Device device, String filePath, String fileName) {
        this.device = device;
        this.filePath = filePath;
        this.fileName = fileName;
    }
}
```

PhotoRequestDto.java

```java
package com.metaverse.planti_be.photo.dto;

import lombok.Getter;
import lombok.Setter;
import org.springframework.web.multipart.MultipartFile;

@Getter
@Setter // @ModelAttribute는 필드에 값을 할당하기 위해 Setter가 필요합니다.
public class PhotoRequestDto {

    private MultipartFile imageFile;
    private String serialNumber;

}
```

PhotoResponseDto.java

```java
package com.metaverse.planti_be.photo.dto;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.metaverse.planti_be.photo.domain.Photo;
import lombok.Getter;

import java.time.LocalDateTime;

@Getter
public class PhotoResponseDto {

    private final Long id;
    private final String filePath;
    private final String fileName;
    private final String deviceSerialNumber;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd HH:mm:ss")
    private final LocalDateTime createdAt;

    // Photo 엔티티를 DTO로 변환하는 public 생성자
    public PhotoResponseDto(Photo photo) {
        this.id = photo.getId();
        this.filePath = photo.getFilePath();
        this.fileName = photo.getFileName();
        this.deviceSerialNumber = photo.getDevice().getId();
        this.createdAt = photo.getCreatedAt();
    }
}
```

PhotoRepository.java

```java
package com.metaverse.planti_be.photo.repository;

import com.metaverse.planti_be.photo.domain.Photo;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface PhotoRepository extends JpaRepository<Photo, Long> {
    // JpaRepository<관리할 엔티티, 엔티티의 ID 타입>
}
```

PhotoService.java

```java
package com.metaverse.planti_be.photo.service;

import com.metaverse.planti_be.device.domain.Device;
import com.metaverse.planti_be.device.repository.DeviceRepository;
import com.metaverse.planti_be.photo.domain.Photo;
import com.metaverse.planti_be.photo.dto.PhotoRequestDto;
import com.metaverse.planti_be.photo.dto.PhotoResponseDto;
import com.metaverse.planti_be.photo.repository.PhotoRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.nio.file.Paths;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class PhotoService {

    private final PhotoRepository photoRepository;
    private final DeviceRepository deviceRepository;

    @Value("${file.upload-dir}")
    private String uploadDir;

    @Transactional
    public PhotoResponseDto savePhoto(PhotoRequestDto requestDto) throws IOException {
        MultipartFile imageFile = requestDto.getImageFile();
        String serialNumber = requestDto.getSerialNumber();

        if (imageFile == null || imageFile.isEmpty()) {
            throw new IllegalArgumentException("이미지 파일이 필요합니다.");
        }
        Device device = deviceRepository.findById(serialNumber)
                .orElseThrow(() -> new IllegalArgumentException("등록되지 않은 기기입니다: " + serialNumber));

        File directory = new File(uploadDir);
        if (!directory.exists()) {
            directory.mkdirs();
        }

        String extension = getFileExtension(imageFile.getOriginalFilename());
        String fileName = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMdd_HHmmss"))
                + "_" + UUID.randomUUID().toString() + "." + extension;

        String filePath = Paths.get(uploadDir, fileName).toString();
        imageFile.transferTo(new File(filePath));

        // 제공해주신 Photo.java의 public 생성자를 사용하여 엔티티 생성
        Photo photo = new Photo(device, filePath, fileName);

        Photo savedPhoto = photoRepository.save(photo);

        // PhotoResponseDto의 생성자를 사용하여 DTO로 변환 후 반환
        return new PhotoResponseDto(savedPhoto);
    }

    private String getFileExtension(String fileName) {
        if (fileName == null || fileName.isEmpty()) {
            return "";
        }
        try {
            return fileName.substring(fileName.lastIndexOf(".") + 1);
        } catch (StringIndexOutOfBoundsException e) {
            return "";
        }
    }
}
```

app.py

```python
from flask import Flask, request, jsonify
from ultralytics import YOLO
import os

# Flask 애플리케이션 생성
app = Flask(__name__)

# ! ! ! [수정 필요] 학습된 모델('best.pt')의 실제 경로를 지정해주세요.
# 아래 경로 예시는 testing.py 파일을 참고했습니다.
model_path = '/home/hyunjun/yoloTest/cherry tomato.v6i.yolov11/train_result/weights/best.pt'
model = YOLO(model_path)

# '/analyze' 주소로 POST 요청을 처리할 API 엔드포인트
@app.route('/analyze', methods=['POST'])
def analyze_image():
    # Spring Boot로부터 받은 JSON 데이터에서 'filePath' 추출
    data = request.get_json()
    if not data or 'filePath' not in data:
        return jsonify({'error': '"filePath"가 필요합니다.'}), 400

    image_path = data['filePath']

    # 파일이 존재하는지 확인
    if not os.path.exists(image_path):
        return jsonify({'error': f'파일을 찾을 수 없습니다: {image_path}'}), 404

    try:
        # YOLO 모델로 이미지 분석 수행
        results = model(image_path)

        # 분석 결과 중 가장 신뢰도 높은 것 하나만 선택
        best_result = {}
        highest_confidence = 0.0

        names = results[0].names
        for box in results[0].boxes:
            confidence = float(box.conf[0])
            if confidence > highest_confidence:
                highest_confidence = confidence
                class_id = int(box.cls[0])
                best_result = {
                    'objectName': names[class_id],
                    'confidence': round(highest_confidence, 4)
                }

        print(f"✅ 분석 완료: {image_path} -> {best_result}")

        # 가장 신뢰도 높은 결과를 JSON 형태로 Spring Boot에 반환
        return jsonify(best_result)

    except Exception as e:
        print(f"❌ 분석 중 오류 발생: {e}")
        return jsonify({'error': '이미지 분석 중 오류 발생', 'details': str(e)}), 500

if __name__ == '__main__':
    # 서버 실행 (IP는 모든 곳에서 접근 가능하도록 '0.0.0.0'으로 설정)
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Yolo 코드
testing.py과 yaml_1.py과 data.yaml

```python
# 확인된 객체의 총 개수와 각 레벨별 개수 코드 반영
import os
from ultralytics import YOLO
from collections import Counter

# 모델 경로
model_path = '/home/aa/yoloTest/Planti_videoProject/Yolo/result/weights/best.pt'
model = YOLO(model_path)

# 폴더 경로
input_folder = '/home/aa/yoloTest/Planti_videoProject/Yolo/sample_data'
output_folder = '/home/aa/yoloTest/Planti_videoProject/Yolo/predicted'
os.makedirs(output_folder, exist_ok=True)

# 카운터 초기화
class_counts = Counter()
total_detected = 0

# 이미지 처리
image_files = [f for f in os.listdir(input_folder) if f.lower().endswith(('.jpg', '.png', '.jpeg'))]

for image_name in image_files:
    image_path = os.path.join(input_folder, image_name)
    results = model(image_path)

    # 예측된 이미지 저장
    save_path = os.path.join(output_folder, f'pred_{image_name}')
    results[0].save(filename=save_path)

    print(f"\n{image_name} 예측 결과:")
    for box in results[0].boxes:
        cls_id = int(box.cls[0])
        conf = float(box.conf[0])
        class_name = model.names[cls_id]

        # 카운트 누적
        class_counts[class_name] += 1
        total_detected += 1

        print(f"  → Class: {class_name}, Confidence: {conf:.2f}")

# 결과 출력
print("\n최종 감지 결과 요약:")
print(f"총 감지된 객체 수: {total_detected}개\n")
for class_name in model.names.values():
    print(f"{class_name}: {class_counts[class_name]}개")
```

```python
import yaml
from ultralytics import YOLO

# 1. data.yaml 경로
yaml_path = '/home/aa/cherry tomato.v6i.yolov11/data.yaml'

# 2. data.yaml 생성
data = {
    'train': '/home/aa/yoloTest/Planti_videoProject/Yolo/train/images',
    'val': '/home/aa/yoloTest/Planti_videoProject/Yolo/valid/images',
    'test': '/home/aa/yoloTest/Planti_videoProject/Yolo/test/images',
    'names': ['bug', 'level 1', 'level 2', 'level 3', 'level 4', 'level 5', 'level 6'],
    'nc': 7
}

with open(yaml_path, 'w') as f:
    yaml.dump(data, f)

# 확인
with open(yaml_path, 'r') as f:
    print(yaml.safe_load(f))

# 3. 모델 로드
model = YOLO('yolo11n.pt')

# 4. 클래스 정보 출력
print("Before Training:")
print(type(model.names), len(model.names))
print(model.names)

# 5. 학습 시작
model.train(
    data=yaml_path,
    epochs=30,
    patience=5,
    imgsz=416,
    project='/home/aa/yoloTest/Planti_videoProject/Yolo',  # ⬅ 결과 저장 위치
    name='result'                                          # 폴더명: result
)

# 6. 학습 후 클래스 정보 출력
print("After Training:")
print(type(model.names), len(model.names))
print(model.names)
```

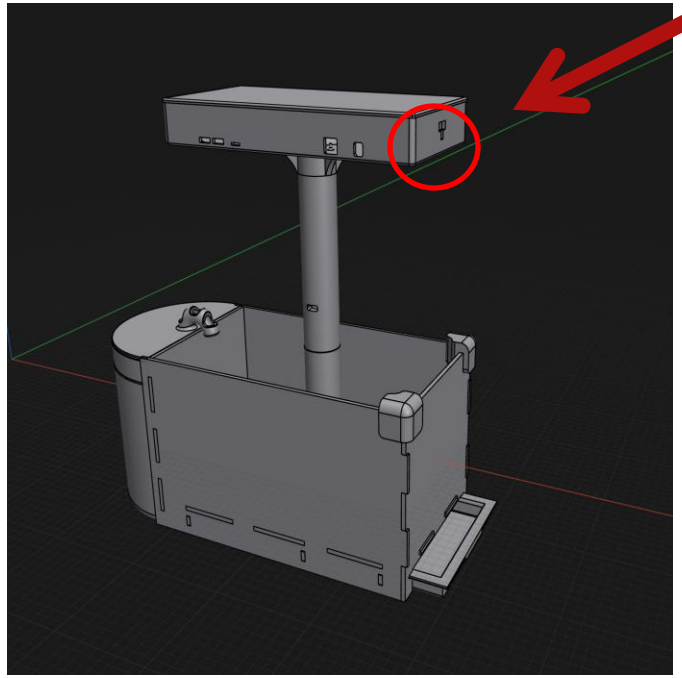
```yaml
names:
- bug
- level 1
- level 2
- level 3
- level 4
- level 5
- level 6
nc: 7
test: /home/aa/cherry tomato.v6i.yolov11/test/images
train: /home/aa/cherry tomato.v6i.yolov11/train/images
val: /home/aa/cherry tomato.v6i.yolov11/valid/images
```

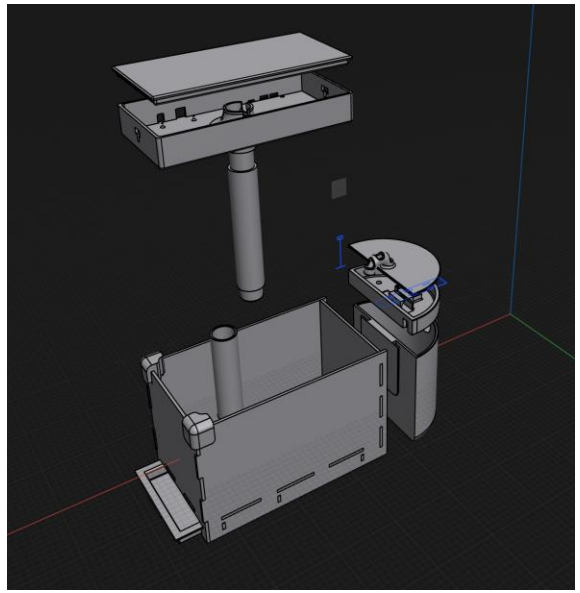
시연 연상

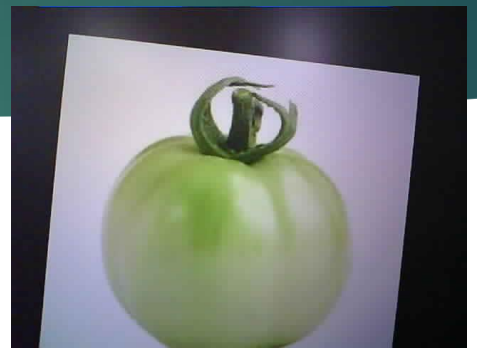

Output
Serial Monitor ×
Message (Enter to send message to 'AI Thinker ESP32-CAM' on 'COM5')
ESP32-Cam 작동
ELF file SHA256: 3a7efd5c5
Rebooting...
ets Jul 29 2019 12:21:46
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4980
load:0x40078000,len:16612
load:0x40080400,len:3480
entry 0x400805b4
=== ESP32-CAM 사진 업로드 시스템 (회전 적용) ===
카메라 초기화 시작...
카메라 초기화 성공!
카메라 180도 회전 설정 완료
WiFi 연결 시작...
.....

스마트팜 + Yolo11

카메라 설치 구역



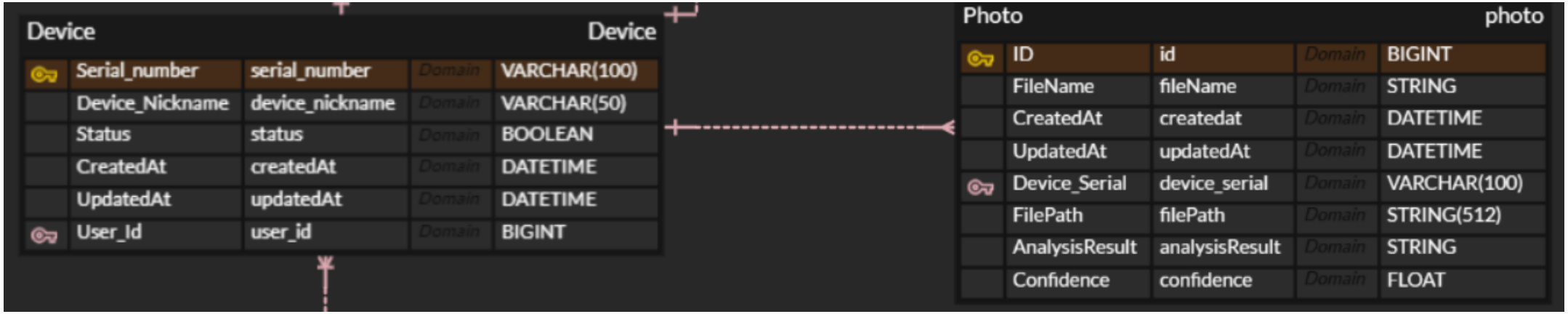




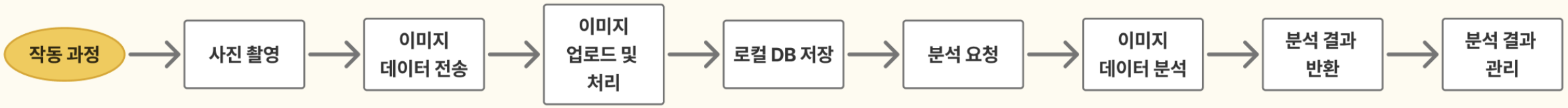
8	8	2025-09-23 17:00:44.975935	2025-09-23 17:00:45.060683	level 4	0.5

13	13	2025-09-23 17:05:04.205096	2025-09-23 17:05:04.267453	level 5	0.84

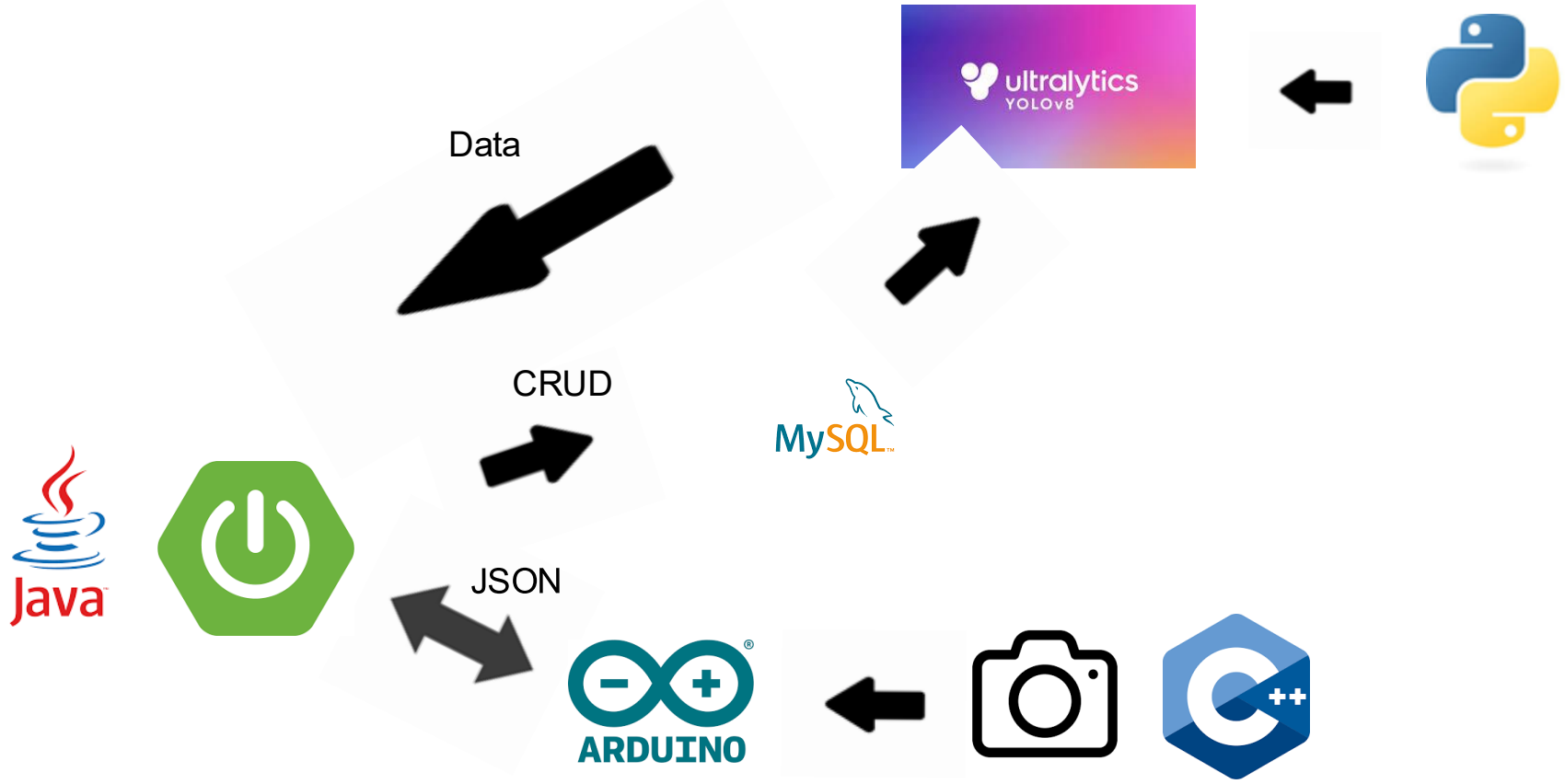
ERD


Device
Device
Serial_number
serial_number
Domain
VARCHAR(100)
Device_Nickname
device_nickname
Domain
VARCHAR(50)
Status
status
Domain
BOOLEAN
CreatedAt
createdAt
Domain
DATETIME
UpdatedAt
updatedAt
Domain
DATETIME
User_Id
user_id
Domain
BIGINT
Photo
photo
ID
id
Domain
BIGINT
FileName
fileName
Domain
STRING
CreatedAt
createdat
Domain
DATETIME
UpdatedAt
updatedAt
Domain
DATETIME
Device_Serial
device_serial
Domain
VARCHAR(100)
FilePath
filePath
Domain
STRING(512)
AnalysisResult
analysisResult
Domain
STRING
Confidence
confidence
Domain
FLOAT

플로우차트


작동 과정
사진 촬영
이미지 데이터 전송
이미지 업로드 및 처리
로컬 DB 저장
분석 요청
이미지 데이터 분석
분석 결과 반환
분석 결과 관리

시스템 아키텍처



Q & A

https://github.com/patrick005/Projects/blob/observer/reference/database.md