프로젝트명 : AI와 열화상 데이터 분석을 활용한 IoT 기반 서버 쿨링 시스템

캡스톤 디자인II, 중간보고서

Version 1.0

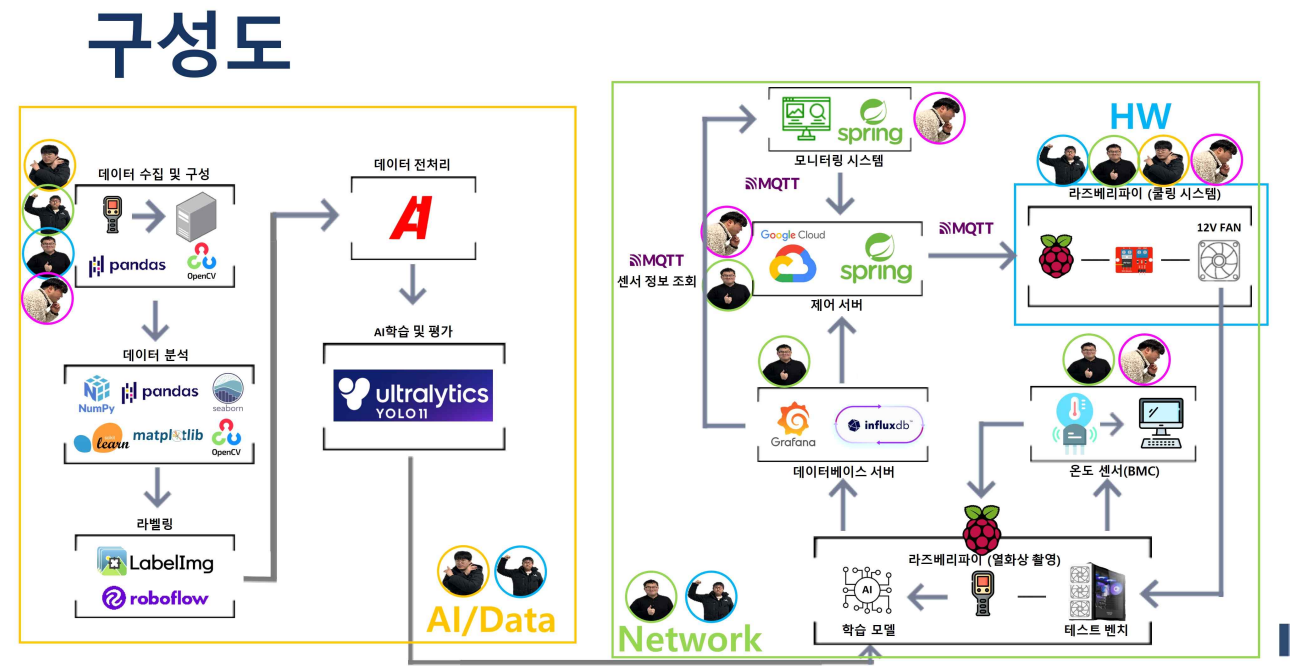
개발 팀원 명(팀리더):홍수민
오민석
길기훈
지원근

대표 연락처:010-2737-8034
e-mail: 20201793@edu.hanbat.ac.kr

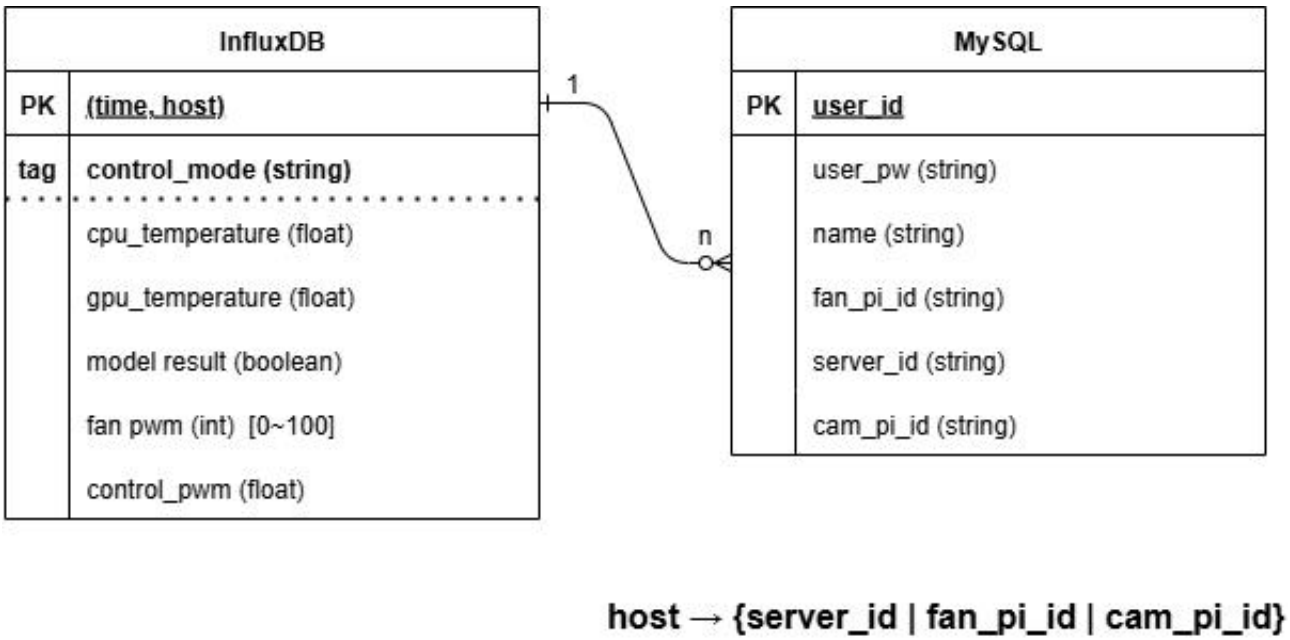
캡스톤 디자인 II 중간보고서 내용

1. 요구사항 정의서에 명세된 기능에 대하여 현재까지 진척된 결과 및 그 내용을 기술하시오.

<<전체 시스템 구성도>>



<<Database E-R Diagram>>



host → {server_id | fan_pi_id | cam_pi_id}

<<열화상 데이터셋 수집 코드>>

```python
import sys
import numpy as np
import cv2
from pylepton.Lepton3 import Lepton3
from datetime import datetime
import time
import math

MIN_TEMP = 10 + 273.15
MAX_TEMP = 60 + 273.15

def capture(flip_v = False, device = "/dev/spidev0.0"):
  with Lepton3(device) as l:
    #l.set_agc_enable(0)
    a,_ = l.capture()
    #np.right_shift(a, 8, scaled)
    min_adc = MIN_TEMP * 100
    max_adc = MAX_TEMP * 100
    temp_k = a / 100.0
    temp_c = temp_k - 273.15
    a[a<=min_adc] = min_adc + 1
    a[a>=max_adc] = max_adc - 1
    scaled = np.clip((a - min_adc) / (max_adc - min_adc), 0, 1)

  if flip_v:
    cv2.flip(a,0,a)
  cv2.normalize(scaled, None, alpha = 0, beta = 65535, norm_type = cv2.NORM_MINMAX)
  #np.right_shift(a, 8, a)
  scaled = scaled * 256



  print(f" 최소 온도: {np.min(temp_c):.2f} C / {np.min(temp_k): .2f} K")
  print(f" 최대 온도: {np.max(temp_c):.2f} C / {np.max(temp_k): .2f} K")

  return np.uint8(scaled), np.min(temp_c), np.max(temp_c)

```

```python
if __name__ == '__main__':
  from optparse import OptionParser

  a = 0

  usage = "usage: %prog [options] output_file[.format]"
  parser = OptionParser(usage=usage)

  parser.add_option("-f", "--flip-vertical",
                    action="store_true", dest="flip_v", default=False,
                    help="flip the output image vertically")

  parser.add_option("-d", "--device",
                    dest="device", default="/dev/spidev0.0",
                    help="specify the spi device node (might be /dev/spidev0.1 on a newer device)")

  (options, args) = parser.parse_args()

  if len(args) < 1:
    print("You must specify an output filename")
    sys.exit(1)

  while (a <= int(args[0])):
    image, min_temp, max_temp = capture(flip_v = options.flip_v, device = options.device)
    image = cv2.applyColorMap(image, cv2.COLORMAP_JET)
    img_name = datetime.now().strftime("%Y-%m-%d %H:%M:%S") + " min_" + str(round(min_temp, 2)) + " max_" + str(round(max_temp, 2)) + ".png"
    cv2.imwrite(img_name, image)
    a+=1
    time.sleep(int(args[1]))
```

<<오토 라벨링 코드>>

```python
import os
import re
import glob
import argparse


def extract_max_value(filename):
    """파일 이름에서 max_ 다음에 오는 숫자 값을 추출합니다."""
    match = re.search(r'max_(\d+\.\d+)', filename)
    if match:
        return float(match.group(1))
    return None

def create_yolo_label(output_path, class_id, bbox):
    """YOLO 형식의 라벨 파일을 생성합니다."""
    with open(output_path, 'w') as f:
        x_center, y_center, width, height = bbox
        f.write(f"{class_id} {x_center} {y_center} {width} {height}\n")

def auto_labeling(image_dir, output_dir, threshold, bbox, class_id=0, class_name='A'):
    """이미지 파일을 자동으로 라벨링합니다."""
    # 출력 디렉토리가 없으면 생성
    os.makedirs(output_dir, exist_ok=True)

    # 이미지 파일 검색 (jpg, png 확장자)
    image_extensions = ['*.jpg', '*.jpeg', '*.png']
    image_files = []
    for ext in image_extensions:
        image_files.extend(glob.glob(os.path.join(image_dir, ext)))

    abnormal_count = 0
    normal_count = 0

    for image_path in image_files:
        filename = os.path.basename(image_path)
        max_value = extract_max_value(filename)

        if max_value is not None:
            # 라벨 파일 경로 생성 (이미지와 같은 이름, 확장자만 .txt로)
            label_filename = os.path.splitext(filename)[0] + '.txt'
            label_path = os.path.join(output_dir, label_filename)

            # 임계값 이상이면 지정된 클래스로 라벨링
            if max_value >= threshold:
                create_yolo_label(label_path, 0, bbox)
                abnormal_count += 1

            else:
                create_yolo_label(label_path, 1, bbox)
                normal_count +=1

    print(f"라벨링 완료: 총 {len(image_files)}개 이미지 중 {abnormal_count}개가 '비정상' 클래스로, {normal_count}개가 '정상' 클래스로 라벨링되었습니다.")
    return len(image_files)


# 메인 실행 부분
if __name__ == "__main__":
    # 명령행 인자를 먼저 확인
    parser = argparse.ArgumentParser(description='YOLO 오토라벨링 도구')

    parser.add_argument('--image_dir', type=str, help='이미지가 있는 디렉토리 경로')
    parser.add_argument('--output_dir', type=str, help='라벨 파일을 저장할 디렉토리 경로')
    parser.add_argument('--threshold', type=float, help='max_ 값의 임계값')
    parser.add_argument('--class_id', type=int, default=0, help='클래스 ID (기본값: 0)')
    parser.add_argument('--class_name', type=str, default='A', help='클래스 이름 (기본값: A)')
    parser.add_argument('--x_center', type=float, default=0.5, help='바운딩 박스 중심 x 좌표 (0~1)')
    parser.add_argument('--y_center', type=float, default=0.5, help='바운딩 박스 중심 y 좌표 (0~1)')
    parser.add_argument('--width', type=float, default=0.8, help='바운딩 박스 너비 (0~1)')
    parser.add_argument('--height', type=float, default=0.8, help='바운딩 박스 높이 (0~1)')

    args = parser.parse_args()



    bbox = (args.x_center, args.y_center, args.width, args.height)
    auto_labeling(args.image_dir, args.output_dir, args.threshold, bbox, args.class_id, args.class_name)
```

<<BMC 코드>>

```python
import psutil
import cpuinfo
import GPUtil
import platform
import os

# CPU 정보
def get_cpu_info():
    os_type = platform.system()

    if os_type == 'Linux':
        try:
            temps = []
            base_dir = '/sys/class/thermal/'
            for zone in os.listdir(base_dir):
                if zone.startswith('thermal_zone'):
                    with open(f'{base_dir}{zone}/temp', 'r') as f:
                        temp = float(f.read().strip()) / 1000.0  # 밀리섭씨 ->
                        temps.append(temp)
            if temps:
                return round(sum(temps) / len(temps) ,2)  # 평균 온도 반환
            return "온도 정보를 찾을 수 없습니다."
        except:
            return "Linux 온도 정보 접근 실패"

    elif os_type == 'Windows':
        try:
            import wmi
            w = wmi.WMI(namespace="root\\wmi")
            temperature_info = w.Win32_TemperatureProbe()[0]
            return temperature_info.CurrentReading
        except Exception as e:
            return "Windows 온도 정보 접근 실패: " + str(e)

    elif os_type == 'Darwin':  # macOS
        try:
            import subprocess
            result = subprocess.run(['sudo', 'powermetrics', '--samplers', 'smc', '-n', '1'],
                                    capture_output=True, text=True)
            for line in result.stdout.split('\n'):
                if 'CPU die temperature' in line:
                    return float(line.split(':')[1].strip().rstrip(' C'))
            return "macOS 온도 정보를 찾을 수 없습니다."
        except:
            return "macOS 온도 정보 접근 실패"

    return "지원되지 않는 운영체제입니다."

# RAM 사용량
def get_ram_info():
    ram = psutil.virtual_memory()
    return {
        "총 메모리": f"{ram.total / (1024**3):.2f} GB",
        "사용 중인 메모리": f"{ram.used / (1024**3):.2f} GB",
        "사용 가능한 메모리": f"{ram.available / (1024**3):.2f} GB",
        "사용률": f"{ram.percent}%"
    }

# GPU 정보
def get_gpu_info():
    try:
        gpus = GPUtil.getGPUs()
        gpu_info = []

        for i, gpu in enumerate(gpus):
            gpu_info.append({
                "GPU ID": i,
                "이름": gpu.name,
                "부하율": f"{gpu.load * 100:.2f}%",
                "메모리 사용": f"{gpu.memoryUsed} MB / {gpu.memoryTotal} MB",
                "메모리 사용률": f"{gpu.memoryUtil * 100:.2f}%",
                "온도": f"{gpu.temperature}°C"
            })

        if not gpu_info:
            return "GPU를 찾을 수 없습니다."

        return gpu_info
    except:
        return "GPU 정보 접근 실패"

# 모든 정보 출력
def print_system_info():
    print("=" * 50)
    print("시스템 정보")
    print("=" * 50)

    # CPU 정보
    print("\n[CPU 정보]")
    print(f"CPU 모델: {cpuinfo.get_cpu_info()['brand_raw']}")
    print(f"코어 수: {psutil.cpu_count(logical=False)} (물리적), {psutil.cpu_count(logical=True)} (논리적)")
    print(f"CPU 사용률: {psutil.cpu_percent()}%")
    print(f"CPU 온도: {get_cpu_info()}°C")

    # RAM 정보
    print("\n[RAM 정보]")
    ram_info = get_ram_info()
    for key, value in ram_info.items():
        print(f"{key}: {value}")

    # GPU 정보
    print("\n[GPU 정보]")
    gpu_info = get_gpu_info()
    if isinstance(gpu_info, list):
        for i, gpu in enumerate(gpu_info):
            print(f"\nGPU {i+1}:")
            for key, value in gpu.items():
                if key != "GPU ID":
                    print(f"  {key}: {value}")
    else:
        print(gpu_info)

    print("\n" + "=" * 50)

if __name__ == "__main__":
    print_system_info()
```

<<Train // Test 분류 코드>>

```python
from glob import glob
import os
import shutil

dataset = "C:/Users/82102/OneDrive/사진/바탕 화면/zip/"

img_list_co = glob(dataset + 'dataset_idle/*.png')[:1500]
print(len(img_list_co))

from sklearn.model_selection import train_test_split

# resize 이미지를 test(20%)/train(80%) dataset으로 나누기
train_img_list_co, val_img_list_co = train_test_split(img_list_co, test_size=0.2, random_state=2000)

# 각각의 test(20%)/train(80%) dataset 이미지 수 출력
print(len(train_img_list_co), len(val_img_list_co))

# 기본 경로 설정
train_img_dir = os.path.join(dataset, 'train', 'image')
train_label_dir = os.path.join(dataset, 'train', 'labels')
val_img_dir = os.path.join(dataset, 'test', 'image')
val_label_dir = os.path.join(dataset, 'test', 'labels')

# 디렉토리 생성
os.makedirs(train_img_dir, exist_ok=True)
os.makedirs(train_label_dir, exist_ok=True)
os.makedirs(val_img_dir, exist_ok=True)
os.makedirs(val_label_dir, exist_ok=True)

# 이미지 이동 함수
def move_images(image_list, img_dst_dir, label_dst_dir):
    for img_path in image_list:
        # 이미지 파일 이름
        filename = os.path.basename(img_path)
        print(img_path)
        # 라벨 경로 추정 (예: .tif → .png)
        label_path = img_path.replace('dataset_idle', 'idleoutput').replace('.png', '.txt')

        # 이미지와 라벨 이동
        shutil.copy(img_path, os.path.join(img_dst_dir, filename))
        shutil.copy(label_path, os.path.join(label_dst_dir, os.path.basename(label_path)))


    # 이미지와 라벨 각각 이동
    move_images(train_img_list_co, train_img_dir, train_label_dir)
    move_images(val_img_list_co, val_img_dir, val_label_dir)

    print("데이터 분할 및 복사가 완료되었습니다.")
```

<<YOLO11 학습 코드>>

```
%pip install ultralytics
from ultralytics import YOLO
```

```
from google.colab import drive
import os

drive.mount('/content/drive')
```

```
print("현재 작업 경로: ", os.getcwd())
os.chdir("/content/drive/MyDrive")
print("변경된 작업 경로: ", os.getcwd())
```

```
model = YOLO("yolo11s.pt")
```

```
train_results = model.train(
    data = "/content/drive/MyDrive/yolo11_test/capstone_dataset/data.yaml",
    epochs=50,
    imgsz=160,
    batch=32,
    device=0,
)
```

```
metrics = model.val()
```

```
results = model("/content/drive/MyDrive/yolo11_test/capstone_dataset/valid/images/2025-04-28 11_38_43 min_20.31 max_37.39.png")
results[0].show()
```

```
path = model.export(format="onnx")
```

2. 프로젝트 수행을 위해 적용된 추진전략, 수행 방법의 결과를 작성하고, 만일 적용과정에서 문제점이 도출되었다면 그 문제를 분석하고 해결방안을 기술하시오.

애자일 방법론 적용 및 추진 전략

본 프로젝트는 애자일 방법론을 기반으로, 유연한 계획 수립과 반복적인 피드백을 통해 목표를 달성하는 전략을 적용하였습니다. 프로젝트 초기에는 팀원별 역할 분담, 실험 환경(장소, 테스트 벤치) 구축, 관련 논문과 자료확보, AI 학습(데이터 전처리 및 모델 학습) 등 기초 역량을 강화하는 데 집중하였습니다. 캡스톤 지원금 확보 전, 필요한 장비와 리소스를 사전에 준비하여 프로젝트의 원활한 시작을 도모하였습니다.

주요 수행 방법 및 결과

- 3월 말, 프로젝트의 핵심 재료인 열화상 카메라 모듈과 보드를 구입하고, 데이터 수집 및 실험을 본격적으로 시작하였습니다.
- 데이터 수집 및 전처리 과정에서 예상보다 많은 시간이 소요되어, 당초 1주 예정이었던 데이터 수집 기간을 2주로 연장하였습니다.
- 열화상 카메라의 AGC(Automatic Gain Control) 설정이 AI 학습에 적합하지 않아, 직접 Min-Max 값을 지정하고, 이상치(outlier)에 대한 별도 처리를 통해 데이터 품질을 최적화하였습니다.
- 전처리된 열화상 데이터를 기반으로 정상/비정상 상태를 라벨링하고, YOLOv11 모델에 학습시켜 테스트벤치의 CPU, GPU, RAM의 이상 여부를 효과적으로 판별할 수 있게 하였습니다.

문제점 도출 및 해결방안

1. 데이터 수집 및 전처리 과정의 문제점과 해결방안
   - 문제점: 열화상 카메라의 AGC 설정으로 인해, 환경 변화에 따라 데이터의 일관성이 떨어졌고, AI 학습에 적합하지 않은 데이터가 발생하였습니다.
   - 해결방안: 측정된 데이터를 기반으로 Min-Max 값을 직접 지정하고, Min-Max 범위 밖의 이상치는 각각 Min, Max값으로 치환하여 데이터의 일관성과 품질을 확보하였습니다. 반복적인 실험과 피드백을 통해 데이터 전처리 절차를 표준화하였습니다.

2. 팀원 역할 및 책임 수행 결과, 문제점과 해결방안
   - 결과: 각 팀원은 역할에 따라 실험 환경 구축, 데이터 수집, 전처리, 모델 학습 등 세부 업무를 분담하여 책임감을 가지고 수행하였습니다.
   - 문제점: 초기에는 역할 분담이 명확하지 않아 일부 업무가 중복되거나 누락되는 등 혼선이 있었습니다.
   - 해결방안: 정기적인 스크럼 미팅을 통해 업무 진행 상황을 공유하고, 역할과 책임을 구체적으로 재정의함으로써 효율적인 협업 구조를 마련하였습니다.

3. 프로젝트 일정 지연 문제점과 해결방안

- 문제점: 열화상 카메라의 구조적 한계(물리적으로 작은 크기와 연약한 배선 구조)와 데이터 전처리의 난이도로 인해 데이터 수집 및 처리 일정이 1주에서 2주로 지연되었습니다.
- 해결방안: 일정 지연의 원인을 분석하여 우선순위가 높은 핵심 업무에 리소스를 집중하였고, 병렬 작업이 가능한 부분은 동시 진행하여 전체 일정에 미치는 영향을 최소화하였습니다.

종합 평가 및 향후 계획

애자일 방법론을 적용한 덕분에 반복적인 피드백과 개선을 통해 문제를 신속히 파악하고 해결할 수 있었습니다. 앞으로도 정기적인 회고와 소통을 통해 프로젝트의 완성도를 높이고, 남은 과업(데이터 전처리 보완(업스케일링), 모델 성능 개선, 실시간 모니터링 시스템 구축 등)에 집중할 계획입니다.

프로젝트명 : AI와 열화상 데이터 분석을 활용한 IoT 기반 서버 쿨링 시스템

소프트웨어 요구사항 정의서

Version 1.0

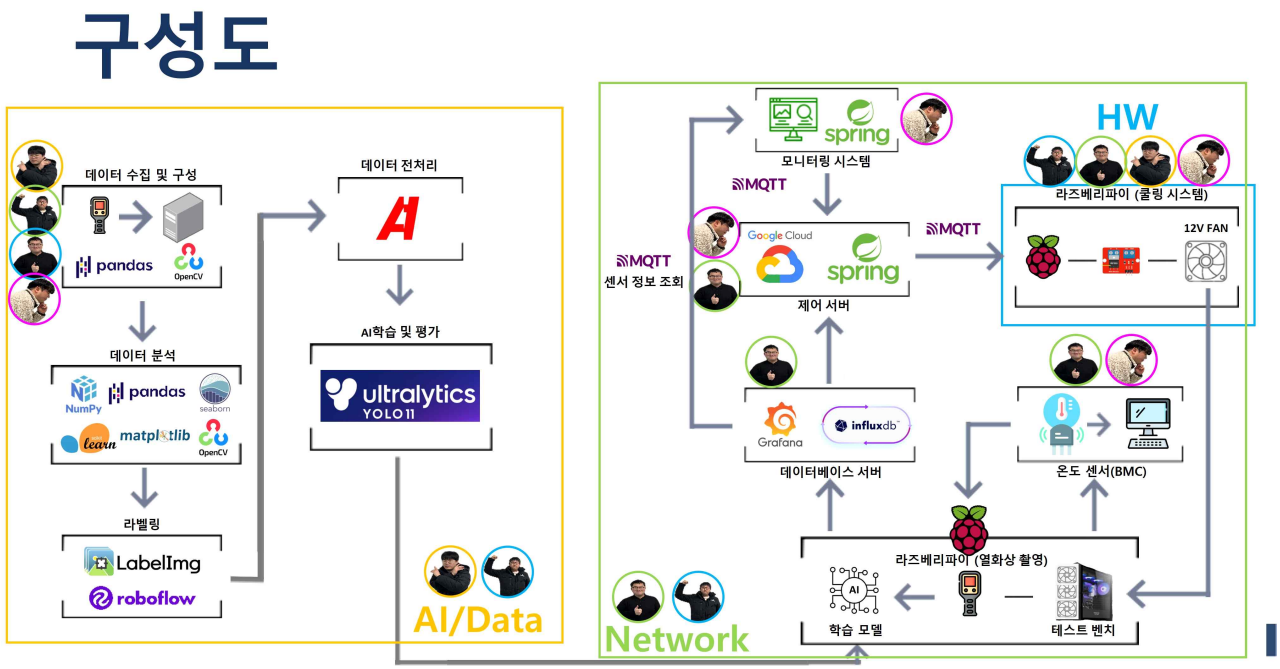
개발 팀원 명(팀리더):홍수민
오민석
길기훈
지원근

대표 연락처:010-2737-8034
e-mail: 20201793@edu.hanbat.ac.kr

목차

1. 시스템 개요

<<시스템 구성도>>



2. 시스템 장비 구성요구사항

요구사항 고유번호		ECR-001		
요구사항 명칭		FLIR Lepton 3.5 열화상 카메라 모듈 및 장착 보드 [수량: 각 1개]		
요구사항 분류		시스템 장비구성 요구사항	응락수준	필수
요구사항 상세설명	정의	장비 설치		
	세부 내용	• 대상 장비: 열화상 촬영에 도입되는 라즈베리파이 모듈형 열화상 카메라 및 카메라 장작 보드 • 장비 설치 요구사항: 본 장비는 수업 시간이 아닌 14:00~ 20:00 사이에 설치하며, 도입 이전에 테스트 환경에서 하드웨어 정비 및 소프트웨어 최적화가 필요함		

요구사항 고유번호		ECR-002		
요구사항 명칭		라즈베리파이 4 Model B, 4GB RAM, microSD 16GB [수량: 2개]		
요구사항 분류		시스템 장비구성 요구사항	응락수준	필수
요구사항 상세설명	정의	열화상 카메라 촬영 및 쿨링 시스템 제어		
	세부 내용	• 용도 : 열화상 카메라 및 센서 데이터 수집, 제어 소프트웨어 구동 • 장비 설치 요구사항: 네트워크 연결 필수, 5V/3A 전원 어댑터 사용		

요구사항 고유번호		ECR-003		
요구사항 명칭		MOSFET 기반 팬 제어 회로 [수량: MOSFET 1개, 12V FAN 3개]		
요구사항 분류		시스템 장비구성 요구사항	응락수준	필수
요구사항 상세설명	정의	온도 변화에 따른 서버 쿨링 팬을 자동으로 제어하기 위한 회로 구성		
	세부 내용	• 용도 : 컴퓨터 쿨링 시스템 • 장비 설치 요구사항: 장비에 안전하게 고정한뒤, 실제 적용전 회로 테스트 후, MOSFET을 이용하여 12V의 FAN 구동		

요구사항 고유번호		ECR-004		
요구사항 명칭		서버 내부 온도 데이터 실시간 수집 BMC 구성		
요구사항 분류		시스템 장비구성 요구사항	응락수준	필수
요구사항 상세설명	정의	서버 내부 온도 데이터를 실시간으로 수집, 전송하기 위한 BMC 시스템 구축		
	세부 내용	• 용도 : 서버 주요 부품 온도 모니터링 및 데이터 수집 • 장비 설치 요구사항: 실시간 데이터 송수신을 해야하며, 네트워크 연동을 통해 DB와 연동되야 함		

요구사항 고유번호		ECR-005		
요구사항 명칭		데이터저장용 InfluxDB 서버 구축		
요구사항 분류		시스템 장비구성 요구사항	응락수준	필수
요구사항 상세설명	정의	수집된 데이터를 저장 관리하기 위한 DB서버 구축		
	세부 내용	• 용도 : 실시간 데이터 저장, 조회, 분석 원지 • 장비 설치 요구사항: InfluxDB 설치, 네트워크 연결, 데이터 백업 및 보안 설정		

3. 기능 요구사항

요구사항 고유번호		SFR-FA-001		
요구사항 명칭		서버 상태 분류		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	• 서버 상태 데이터를 바탕으로 서버 온도가 정상인지 비정상인지를 이진 분류하는 기능		
	세부 내용	① 서버 상태는 열화상 이미지로 분석되며 이를 통해 서버가 정상 상태인지 비정상 상태인지 구분 ② 모델은 YOLO기반의 딥러닝 모델을 사용하며 열화상 이미지에서 서버의 온도 패턴을 인식하고 분석한다 ③ 이진 분류는 정상과 비정상으로 구분되며 비정상 상태는 과열 상태 등을 포함해야 함		

요구사항 고유번호		SFR-FA-002		
요구사항 명칭		쿨링 세기 조절		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	• BMC 측정 데이터를 바탕으로 서버의 온도에 맞춰 쿨링 세기를 조절하는 기능		
	세부 내용	① BMC 측정 데이터를 통해 실시간으로 서버의 온도를 모니터링 한다. ② 온도가 낮으면 쿨링 세기는 약하게 설정되고 온도가 높으면 쿨링 세기는 강하게 설정됨 ③ 쿨링 세기는 서버의 온도에 비례하여 자동으로 조정되며 온도에 따라 적절한 쿨링 효과를 제공함 ④ 쿨링 조절은 서버의 온도 변화에 빠르게 반응하여 서버가 과열 상태가 되지 않도록 함		

요구사항 고유번호		SFR-FA-003		
요구사항 명칭		서버 상태 모니터링 대시보드		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	• 서버 상태 및 쿨링 시스템을 실시간으로 모니터링할 수 있는 대시보드 기능		
	세부 내용	① 대시보드에서 실시간으로 서버의 상태(정상/비정상), 온도, 쿨링 세기 등을 모니터링할 수 있어야 함 ② 서버 상태가 비정상인 경우 대시보드에 경고 메시지나 알림이 표시됨		

4. 성능 요구사항

요구사항 고유번호		PER-001		
요구사항 명칭		데이터 처리 속도 및 반응시간		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	- 처리속도 및 시간		
	세부 내용	• 열화상 데이터 수집은 최소 2초의 1번의 속도로 처리해야 함 • AI 모델의 이상 온도 탐지는 1초 이내에 처리되어야 함 • 냉각 제어 시스템은 이상 온도 탐지 후 2초 이내에 작동해야 함		

요구사항 고유번호		PER-002		
요구사항 명칭		시스템 처리량 및 온도 관리		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	- 처리량		
	세부 내용	• 데이터베이스의 데이터들은 실시간으로 Grafana로 표현 가능해야 함 • 테스트벤치 부품의 온도는 CPU 30~45℃, GPU 35~50℃ 범위 내로 유지되어야 함		

요구사항 고유번호		PER-003		
요구사항 명칭		자원 효율성 및 전력 관리		
요구사항 분류		성능	응락수준	필수
요구사항 상세설명	정의	- 자원 사용량		
	세부 내용	• 전력 소비는 기존 냉각 시스템 대비 10% 이상 절감되도록 함 • 시스템의 CPU 사용률은 최대 80% 이내로 유지되어야 함 • 메모리 사용량은 전체 가용 메모리의 80%를 초과하지 않아야 함		

5. 인터페이스 요구사항

1) 사용자 인터페이스 요구사항 분석 및 도출

요구사항 고유번호		SIR-001		
요구사항 명칭		서버 상태 대시보드		
요구사항 분류		사용자 인터페이스	응락수준	필수
요구사항 상세설명	정의	서버 상태를 실시간으로 확인할 수 있는 웹 기반 대시보드 제공		
	세부 내용	• 대시보드는 웹 브라우저에서 접근 가능함 • InfluxDB와 데이터 분석 시스템 간 RESTful API 연동이 가능해야 함 • 서버 상태(정상/비정상), 현재 온도, 과거 온도 기록, 쿨링 세기 등 주요 지표가 실시간으로 표시됨 • 서버 상태가 비정상인 경우 대시보드에 경고 메시지나 알림이 표시됨		

2) 시스템 인터페이스 요구사항 분석 및 도출

요구사항 고유번호		SIR-004		
요구사항 명칭		BMC 데이터 수집 및 저장 인터페이스		
요구사항 분류		시스템 인터페이스	응락수준	필수
요구사항 상세설명	정의	파이썬 기반 BMC 시스템과 데이터베이스 서버 간의 데이터 전송 인터페이스		
	세부 내용	• 파이썬으로 구현된 BMC는 CPU, RAM, GPU 등 주요 부품의 온도 데이터를 수집해야 함 • 수집된 데이터는 InfluxDB 프로토콜을 통해 시계열 데이터베이스에 저장해야 함 • 데이터 전송은 HTTP/HTTPS RESTful API를 통해 이루어져야 함		

요구사항 고유번호		SIR-005		
요구사항 명칭		AI 분석 및 제어 시스템 연동 인터페이스		
요구사항 분류		시스템 인터페이스	응락수준	필수
요구사항 상세설명	정의	AI 서버와 제어 서버 간 이상 발연 감지 및 제어 신호 전달을 위한 인터페이스		
	세부 내용	• YOLO11 기반 열화상 이미지에서 시스템의 정상/비정상을 구분하고, BMC 기반으로 해당 부분의 위치를 알아내어 열처리를 함 • 이상 발열 감지 시 RESTful API를 통해 제어 서버에 JSON형식으로 이벤트를 전송해야 함 • 제어 서버는 이상 발열 푸붐의 위치, 온도 값 등의 메타데아터를 수신할 수 있어야 함		

요구사항 고유번호		SIR-006		
요구사항 명칭		라즈베리파이 팬 제어		
요구사항 분류		시스템 인터페이스	응락수준	필수
요구사항 상세설명	정의	제어 서버와 라즈베리파이 간의 냉각 팬 제어를 위한 인터페이스		
	세부 내용	• GPIO 18핀을 사용하여 팬 속도를 제어해야 함 • MOSFET을 이용한 PWM 제어 회로를 구현하여 팬의 속도를 0~100% 범위에서 조절할 수 있어야 함 • 제어 서버는 RESTful API를 통해 라즈베리파이에 팬 속도 제어 명령을 전송해야 함		

요구사항 고유번호		SIR-007		
요구사항 명칭		열화상 카메라 스트리밍 인터페이스		
요구사항 분류		시스템 인터페이스	응락수준	필수
요구사항 상세설명	정의	열화상 카메라와 AI 서버 간 실시간 영상 전송을 위한 인터페이스		
	세부 내용	• GStreamer 기반의 RTP/UDP 프로토콜을 사용하여 열화상 카메라 영상을 실시간으로 스트리밍해야 함 • 스트리밍 데이터는 초당 최소 5프레임 이상의 속도로 전송되어야 함 • AI서버 측에서 받는 데이터를 최소 2초당 1번은 받아내야 함		

6. 데이터 요구사항

요구사항 고유번호	DAR-001		
요구사항 명칭	분석에 사용되는 열화상 이미지 데이터 품질 보장 저장		
요구사항 분류	데이터	응락수준	선택
요구사항 상세설명	• 분석에 사용되는 열화상 이미지 데이터는 분석 정확도를 유지할 수 있는 업스케일링 된 해상도와 품질로 원본형태로 저장되어야 함 • 데이터 손실, 압축, 변환없이 저장하여 AI 분석 및 통계 처리 시 신뢰성 확보		

요구사항 고유번호	DAR-002		
요구사항 명칭	CPU, RAM, GPU 온도 데이터 시간별 기록		
요구사항 분류	데이터	응락수준	필수
요구사항 상세설명	• CPU, RAM, GPU의 온도 데이터는 실시간(2초이내)으로 기록 • 시간별 데이터는 InfluxDB에 저장 • 온도 변화 추이 분석 및 이상 감지에 활용		

요구사항 고유번호	DAR-003		
요구사항 명칭	팬 속도 제어 이벤트 로그 기록		
요구사항 분류	데이터	응락수준	필수
요구사항 상세설명	• 팬 속도 변경, 제어 명령 발생 시마다 이벤트 로그 자동 저장 • 로그에는 발생 시각, 제어명령, 이전/ 변경속도 등 포함 • 로그를 이용하여 장애 분석 및 유지보수 지원		

요구사항 고유번호	DAR-004		
요구사항 명칭	InfluxDB 시계열 데이터베이스 저장		
요구사항 분류	데이터	응락수준	필수
요구사항 상세설명	• 온도, 팬 속도, 이벤트 등 모든 데이터는 InfluxDB에 저장 • 시계열 데이터 구조로 저장하여 빠른 조회 및 통계 분석 가능 • 데이터 백업 및 보안 설정 적용		

요구사항 고유번호	DAR-005		
요구사항 명칭	데이터 일일 단위 백업		
요구사항 분류	데이터	응락수준	선택
요구사항 상세설명	• 매일 정해진 시간에 전체 데이터 자동 백업 • 백업 데이터는 별도의 저장소에 보관 • 백업 성공 모니터링 및 장애 시 알림		

요구사항 고유번호	DAR-006		
요구사항 명칭	통계 분석용 구조화 데이터 저장		
요구사항 분류	데이터	응락수준	필수
요구사항 상세설명	• 온도 데이터는 분석에 용이하도록 필드별로 구조화하여 저장 • 통계, 시각화 AI 분석 등에 활용 가능 • 명확한 시스템 운영에 필요한, 데이터 정합성 및 일관성 유지 필요		

7. 테스트 요구사항

요구사항 고유번호	TER-001		
요구사항 명칭	AI 모델 성능 및 정확도 테스트		
요구사항 분류	테스트	응락수준	필수
요구사항 세부내용	• AI 모델의 이상 발열 탐지 정확도는 90%이상이어야 함 • 다양한 부하 상황에서 AI 모델의 탐지 정확도를 검증해야 함 • 오탐지(Fasle Positive)와 미탐지(False Negative) 비율을 측정하고 분석해야함		

요구사항 고유번호	TER-002		
요구사항 명칭	냉각 시스템 성능 테스트		
요구사항 분류	테스트	응락수준	필수
요구사항 세부내용	• 냉각 시스템의 반응 시간은 이상 발열 탐지 후 2초 이내여야 함 • 인공적 발열 상황에서의 냉각 시스템 검증을 테스트해야 함 • 제어된 환경에서 팬 속도 조절의 정확성을 검증해야 함 • 냉각 시스템이 온도 변화에 따라 적절히 대응하는지 검증해야 함		

요구사항 고유번호	TER-003		
요구사항 명칭	시스템 안정성 및 효율성 테스트		
요구사항 분류	테스트	응락수준	필수
요구사항 세부내용	• 다양한 서버 부하 상황에서 시스템의 동작을 테스트해야 함 • 전력 소비 효율성 테스트를 진행해야 함 • 시스템은 12시간 연속 작동 테스트를 통과해야 함 • 장기간 운영 시 시스템 안정성 및 데이터 무결성을 검증해야 함		

8. 보안 요구사항

요구사항 고유번호	SER-001		
요구사항 명칭	데이터 보안		
요구사항 분류	보안	응락수준	필수
요구사항 상세설명	• 촬영된 이미지에는 민감한 정보가 포함될 수 있으므로, 저장 시 암호화하거나 접근을 제한 • 학습 데이터가 변조되지 않도록 정기적인 검증이 필요		

요구사항 고유번호	SER-002		
요구사항 명칭	모델/결과 보안		
요구사항 분류	보안	응락수준	필수
요구사항 상세설명	• 학습된 모델(가중치 파일 등)을 외부에 유출되지 않도록 관리 • 비정상 감지 결과를 악의적으로 조작할 수 없도록 결과 저장 및 전송 과정에서 서명이나 암호화를 적용		

요구사항 고유번호	SER-003		
요구사항 명칭	시스템 보안		
요구사항 분류	보안	응락수준	필수
요구사항 상세설명	• 관리자 권한이 있는 사람만 서버 상태를 확인하고 조치할 수 있도록 설계 • 누가 시스템에 접근했는지 어떤 결과를 확인했는지 로그를 기록하고 이상 행위가 감지되면 알림을 받을 수 있도록 설계		

9. 품질 요구사항

요구사항 고유번호		QUR-001		
요구사항 명칭		정확한 서버 상태 분류		
요구사항 분류		품질	응락수준	필수
요구사항 상세설명	정의	품질관리(정확성 및 신뢰성 관점)		
	세부 내용	• 열화상 이미지를 기반으로 서버의 정상/비정상 상태를 90% 이상의 정확도로 분류할 수 있어야 함 • 동일 입력에 대하여 예측 결과가 95% 이상 일관되어야 함		

요구사항 고유번호		QUR-002		
요구사항 명칭		빠른 응답시간 확보		
요구사항 분류		품질	응락수준	필수
요구사항 상세설명	정의	품질관리(효율성 관점)		
	세부 내용	• 하나의 열화상 이미지를 입력받아 2초 이내에 서버 상태 분류 결과를 출력해야 함 • 모델 경량화 및 최적화를 통해 실시간 모니터링이 가능해야 함		

요구사항 고유번호		QUR-003		
요구사항 명칭		사용성과 유지보수성		
요구사항 분류		품질	응락수준	필수
요구사항 상세설명	정의	품질관리(사용성 및 유지보수성 관점)		
	세부 내용	• 서버 상태 분류 결과를 사용자가 직관적으로 이해할 수 있도록 간단한 형태로 제공해야 함 • 코드와 모델 구조는 모듈화하여 추후 수정 및 개선이 용이하도록 작성되어야 함		

10. 제약 사항

요구사항 고유번호	CON-001		
요구사항 명칭	열화상 카메라 거리 제한		
요구사항 분류	제약사항	응락수준	필수
요구사항 상세설명	• FLIR Lepton 3.5 카메라 특성상 서버와 카메라 간 거리는 1.5m 이내로 설치		

요구사항 고유번호	CON-002		
요구사항 명칭	데이터 수집 및 활용제약		
요구사항 분류	제약사항	응락수준	필수
요구사항 상세설명	• 프로젝트에서는 직접 수집한 열화상 데이터만 사용		

요구사항 고유번호	CON-003		
요구사항 명칭	모델 사용 제약		
요구사항 분류	제약사항	응락수준	필수
요구사항 상세설명	• 서버 상태 분류 모델은 YOLO 모델을 사용		

11. 프로젝트 관리 요구사항

요구사항 고유번호	PMR-001		
요구사항 명칭	일정관리		
요구사항 분류	프로젝트 관리	응락수준	필수
요구사항 상세설명	• 프로젝트 수행 기간 동안 주요 마일스톤(데이터 수집 완료, 모델학습 완료, 테스트 완료 등)을 설정하고 일정에 따라 진행 상황을 주기적으로 점검 • 일정 지연 발생 시 즉시 원인 분석 및 대응 방안을 수립		

요구사항 고유번호	PMR-002		
요구사항 명칭	산출물 관리		
요구사항 분류	프로젝트 관리	응락수준	필수
요구사항 상세설명	• 데이터셋, 모델 학습 결과, 코드, 보고서 등 모든 산출물은 팀 공유 저장소를 통해 관리함 • 파일 버전 관리를 실시하여 작업 이력을 명확히 남겨야 함		

요구사항 고유번호	PMR-003		
요구사항 명칭	품질 관리		
요구사항 분류	프로젝트 관리	응락수준	필수
요구사항 상세설명	• 데이터셋 변경, 모델 아키텍처 변경 등 주요 변경 사항 발생 시 사전 팀원 합의를 통해 변경 내용을 기록하고 관리해야 함 • 품질 요구사항에 따라 모델 정확도, 응답 속도 등을 체크리스트로 관리하고 기준에 미달하는 경우 보완 작업을 진행함		