[스마트 팩토리(6조) – DeeFacto] 프로젝트 수행계획서

구분	내용
프로젝트 주제	**IoT** 센서 기반 공장 환경 모니터링 및 최적의 임계치 제안 시스템
프로젝트 기간	**2025.07.05 ~ 2025.09.11**
팀구성 방식	Team-Leader : 김수현 Front-end : 강명지, 김지은, 박찬규 Back-end : 박찬규, 전성민, 정모경, 황세현 Infra : 김수현, 허장원
프로젝트 개요	**[**배경**]** 환경 데이터 관리 및 분석 시스템이 여전히 온프레미스 환경에 머물러 있어, 관리 효율성과 확장성 측면에서 제약이 발생하고 있다. 더불어 품질 문제의 직접적인 요인은 내부 환경 조건(온도, 습도, 미세먼지 등)과 장비 이상 공정 과정에서 비롯되지만, 현재 많은 기업에서는 이러한 환경 데이터를 단순 수집하거나 수동 관리 시스템에 의존하고 있어 체계적인 분석과 활용에 한계가 존재한다. 이에 따라 본 프로젝트는 기존 온프레미스 기반 시스템을 클라우드 환경으로 전환하는 AM(Architecture Modernization) 프로젝트로 진행된다. 클라우드의 장점을 활용하여 최적의 아키텍처를 제안하고, 이를 통해 관리 및 운영 효율성을 동시에 확보하는 것을 목표로 한다. 또한 환경 데이터 기반의 자동 모니터링 시스템을 구축하여 이슈 분석 및 문제 해결을 지원하고, 나아가 AI를 활용해 MES 생산량 데이터와 센서 데이터 간의 상관관계를 분석함으로써 공정 환경의 최적화된 센서 임계치를 도출하고, 이를 통해 제품 품질을 향상시킬 수 있는 최적화 시스템을 마련하고자 한다. **[**목표**]** 본 프로젝트는 IoT 환경 센서에서 발생하는 대용량 데이터를 안정적으로 처리할 수 있는 클라우드 기반 데이터 파이프라인을 구축하고, 이를 활용하여 실시간 모니터링과 이상치 탐지 알람 서비스를 제공하는 것을 목표로 한다.

<table>
<tr><td></td><td colspan="3">또한 수집된 데이터를 기반으로 교대 근무자를 위한 인수 인계 요약 보고서를 자동 생성하고, AI 분석을 통해 최적의 센서 임계치를 제안함으로써 공장 환경 관리의 신속하고 자동화된 의사결정 및 생산 품질의 지속적 향상을 이끈다.

이를 통해 공정 환경의 안정성, 관리 효율성, 그리고 생산성을 종합적으로 향상시키고자 한다.

[기대효과 요약]
✔ “이상 감지 → 이상 알림 경고/경보 → 조치”의 실시간 알람 시스템 구축
✔ 공정 환경의 안정성과 관리 효율성을 확보하고, 이를 기반으로 생산성 향상 및 품질 안정화에 기여할 수 있는 인사이트를 축적
✔ 이슈 발생 이력 접근성 향상 및 운영 환경 분석 제공
✔ 보고서 생성 자동화로 업무관리 효율성 향상 및 의사결정 지원</td></tr>
<tr><td rowspan="4">습득
직무역량</td><td>직군</td><td>직무</td><td>직무별 핵심 기술역량</td></tr>
<tr><td>개발</td><td>Back</td><td>- Spring Boot 기반 REST API 서버 구성 및 데이터 응답 처리
- MQTT로 전송된 센서 데이터를 수신/가공하는 백엔드 로직 구현
- Kafka를 활용한 이벤트 기반 처리 로직 작성
- SSE 기반 실시간 사용자 알림 API 구현
- OpenSearch 및 S3와의 API 연동 처리</td></tr>
<tr><td>개발</td><td>Front</td><td>- React 기반 UI 개발
- SSE(Server-Sent Events)를 통한 실시간 데이터 표시
- Grafana 연동을 통한 시각화 대시보드 제공
- Three.js를 활용한 3D 도면 시각화
- Tailwind CSS 기반 반응형 웹 구현
- Redux를 활용한 전역 상태 관리
- React Router 기반 SPA 라우팅
- REST API 연동 및 비동기 데이터 처리</td></tr>
<tr><td>개발</td><td>Infra</td><td>- AWS IoT Core 및 Kinesis Data Streams, Opensearch, Elasticache 기반 센서 데이터 파이프라인 구성
- EKS 내 Kafka 클러스터 운영 및 메시지 중계 인프라 구성</td></tr>
</table>

			- ECR 기반 컨테이너 구성 및 이미지 관리 - ArgoCD, Jenkins를 통한 CI/CD 자동 배포 환경 구성 - 모니터링 및 로그 수집 시스템 설계 (CloudWatch 등) - Bedrock 근무 요약 보고서 및 최적의 임계치를 제안하는 Agent 개발 - S3, Lambda, SQS 등 유수의 AWS 리소스 구축 및 연동
	디자인	UI/UX	- Figma를 활용한 UI·UX 프로토타이핑 및 와이어프레임 설계 - 사용자 친화적 레이아웃 구성 및 반응형 웹 디자인 설계 - Tailwind CSS 기반 UI 컴포넌트 디자인 - 사용자 흐름(UX Flow) 설계 및 인터랙션 정의 - React 개발을 위한 디자인 시스템 정립 - 실시간 모니터링 대시보드의 정보 시각화 레이아웃 구성 - Figma를 활용한 UI·UX 프로토타이핑 및 와이어프레임 설계 - 사용자 친화적 레이아웃 및 인터랙션 설계 - UX Flow 및 사용자 여정(User Journey) 정의 - 디자인 시스템 및 컴포넌트 가이드라인 수립 - 실시간 모니터링 대시보드의 정보 구조(그래프)설계
	디자인	3D Modeling	- Rhino 및 Blender 활용한 3D 모델링 및 시각화 - IoT 센서 설치 구조 및 장비 배치에 대한 3D 설계 구현
프로젝트 요구사항	상세 내용	1. 공장 내 주요 환경 데이터(온도, 습도, 미세 먼지등)를 실시간으로 수집할 수 있어야 한다. 2. 수집된 센서 데이터는 MQTT 프로토콜을 통해 안정적으로 전송되어야 하며, 스트리밍 기반으로 처리되어야 한다. 3. 환경 기준 임계치를 초과하는 이상 징후 발생 시, 실시간 알림을 제공해야 한다. 4. 과거 센서 데이터를 시계열로 저장하고, 대시보드를 통해 환경 변화 추이를 시각적으로 분석할 수 있어야 한다. 5. 교대 근무 Shift별 정기 요약 보고서와 매달 공정 환경에 최적의 센서 임계치를 제안할 수 있어야 한다. 6. 시스템은 AWS 기반 클라우드 환경에서 구축되어야 하며, 안정성을 최우선으로		

		하고, 확장성과 보안성을 갖춰야 한다. 7. 사용자 인증 및 접근 제어가 가능해야 하며, 로그 이력 저장 기능도 포함되어야 한다.
	수행 기법	● 센서 데이터 수집 및 전송 - MQTT 프로토콜을 통해 IoT 센서 데이터를 AWS IoT Core로 실시간 전송 ● 실시간 데이터 처리 및 이상 탐지 - AWS Kinesis, Opensearch, Elasticache를 활용해 스트리밍 데이터를 수집 ● 이벤트 처리 및 알림 전파 - 임계치 초과 데이터는 알림 서비스로 전송 - SSE 등을 통한 알림 시스템 구현 ● 데이터 저장 및 분석 기반 구축 - AWS S3에 센서 데이터 저장 - OpenSearch에 적재된 데이터를 Grafana를 통해 시각화/실시간 모니터링 구현 ● 사용자 인터페이스 및 AI agent 연동 - React 기반의 사용자 실시간 모니터링 대시보드를 구현 - Bedrock agnent 기반 분석 보고서 자동 생성 기능 구현 ● 운영 및 배포 자동화 - ECR, ArgoCD 등 DevOps CI/CD 구성
	참고 자료	[IoT Core 무작정 따라하기 Step #1] IoT Core에 MQTT 데이터 보내기] (tistory, mini_world, 2021.10.12) https://1mini2.tistory.com/133 [[IoT Core 무작정 따라하기 Step#2] IoT Core 데이터 핸들링하기] (tistory, mini_world, 2021.10.18) https://1mini2.tistory.com/135 [AWS Lambda의 모든 것: 초보자를 위한 완벽한 가이드 2/2] (smileshark, Hyojung Yoon, 2023.12.20)

		https://www.smileshark.kr/post/all-about-aws-lambda-the-complete-beginners-guide-2 [AWS IoT Core 자습서 시작하기] (aws docs) https://docs.aws.amazon.com/ko_kr/iot/latest/developerguide/iot-gs.html#aws-iot-get-started [Amazon Kinesis Data Streams란?] (aws docs) https://docs.aws.amazon.com/ko_kr/streams/latest/dev/introduction.html [Managed Service for Apache Flink: 작동 방식] (aws docs) https://docs.aws.amazon.com/ko_kr/managed-flink/latest/java/how-it-works.html [Amazon S3란 무엇인가요?] (aws docs) https://docs.aws.amazon.com/ko_kr/AmazonS3/latest/userguide/Welcome.html [Amazon OpenSearch Service란 무엇입니까?] (aws docs) https://docs.aws.amazon.com/ko_kr/opensearch-service/latest/developerguide/what-is.html [Amazon Managed Grafana 리소스 생성 및 사용 방법 알아보기] (aws docs) https://docs.aws.amazon.com/ko_kr/grafana/latest/userguide/getting-started-with-AMG.html [React Three R3F를 활용한 3D 웹 개발 강의] React Three fiber(R3F)로 배우는 인터렉티브 3D 웹 개발 강의	코딩의세계 한태재 - 인프런
	활용 도구	• **Infra / Platform** - 클라우드 인프라 (Provider : AWS) IoT Core, Kinesis Data Stream, Kinesis Data Firehose, Lambda, S3, S3 Glacier, OpenSearch, Athena, Glue, Bedrock, Elasticache, RDS, EC2, CloudWatch, VPC, Route 53, CloudFront, AWS Certificate Manager, EKS, ArgoCD, ECR, ELB, SQS - CI/CD 및 컨테이너: Jenkins, Docker, ArgoCD, Terraform • **Back-end** 개발 도구 - 프레임워크 및 언어: Spring Boot, Java, WebFlux, Reactor • **Front-end** 개발 도구 - UI 프레임워크 및 디자인 도구: React, JavaScript, Tailwind CSS, Redux,

		three.js, Figma ● 협업 도구 - 협업 및 관리: Notion, Jira, GitHub, Discord
상세 프로젝트 수행내용		**1.** 요구사항 분석 및 시스템 설계 - 모니터링 대상 환경 항목(온도, 습도, 미세먼지, VOC 등) 선정 - 요구사항 분석, 기능 요구사항, 알림 조건, 데이터 흐름 정의 - 시스템 전체 아키텍처 설계 - 시스템 전체 인프라 설계(AWS) **2.** 센서 데이터 수집 및 전송 체계 구축 - MQTT 기반으로 IoT 센서 데이터 전송 구조 구현 - AWS IoT Core와 연동하여 실시간 연결 상태 및 보안 설정 구성 - 테스트 센서 데이터를 활용한 시뮬레이션 환경 구성 및 데이터 수신 검증 **3.** 실시간 데이터 처리 및 저장 환경 구축 - AWS Kinesis 활용한 스트리밍 데이터 수집 - AWS Firehose 통해 L0 데이터 S3 저장 - OpenSearch 데이터 동기화하여 실시간 검색 및 조회 가능하도록 구성 - 저장된 데이터를 기반으로 시계열 분석, 이력 조회 기능 제공 - RDS에 1분 단위 센서 summary 데이터를 저장 - Elasticache와 SQS를 활용한 실시간 이상 알람 발생 기능 제공 **4.** 대시보드 및 사용자 인터페이스 개발 - React 기반 웹 UI에서 센서 데이터를 실시간 모니터링 - three.js를 활용해 도면의 3D 모델 시각화 및 실시간 상태 표시 - Grafana를 활용해 OpenSearch 시계열 데이터 및 RDS summary 데이터 시각화 대시보드 구성

	- 사용자 편의를 고려한 UX/UI 설계 및 반응형 페이지 구현 **5.** **AI** 기반 리포트 자동 생성 기능 및 임계치 추천 기능 구현 - 기간별 리포트 자동 생성 및 다운로드 기능 구현 (PDF 출력) - 매월 1일에 zone별 센서 타입에 따라 최적의 임계치를 제안하는 기능 구현 **6.** 이상 발생**(**알림 기능**)** 구현 **-** 이상 상황 데이터를 Lambda로 탐지 및 Elasticache에 적재 - 이상 발생 시 Kafka를 통해 이벤트 메시지 생성 및 전달 - SSE 기반 실시간 알림 전파 기능 구현 - 알림 기록 저장 및 사용자 알림 이력 조회 기능 제공 **7.** 시스템 테스트 및 통합 - 각 모듈 간 데이터 연동 - 기능별 단위 테스트 및 통합 테스트 수행 **8.** 운영 및 배포 자동화 - 전체 시스템을 ECR 기반으로 컨테이너화 - ArgoCD, Jenkins를 활용한 CI/CD 파이프라인 구성 - 운영 효율성을 위한 자동 배포 및 롤백 환경 구성 **9.** 최종 점검**,** 사용자 교육 및 보고 - 운영자 대상 사용자 교육 및 매뉴얼 제공 - 결과보고서 작성 및 향후 유지관리 방안 정리
활용기술	● 개발 및 운영 방식 - 방법론: Agile 방식 - 아키텍처: Microservices Architecture (MSA), 이벤트 기반 구조, Serverless 구조 - 데이터 흐름 설계: 실시간 스트리밍 처리, ETL 파이프라인, 데이터 카탈로그 기반 쿼리 분석

	● 통신 및 처리 기술 - IoT 통신: MQTT, REST API, WebSocket - 데이터 처리 기술: 시계열 분석, 데이터 전처리/정제 - 보안 기술: HTTPS, 인증서 관리, IAM 정책 구성, 스프링 시큐리티 ● **AI** 관련 기술 - 자동화 기술: 보고서 자동 생성, 최적 임계치 제안 기능 (Claude Haiku 기반 Bedrock)
프로젝트 산출물	**[**프로젝트 산출물**]** - 사용자 이용 서비스 애플리케이션 (Dee-facto) - 개발 코드 (GitHub) - 시연 영상 **[**문서 산출물**]** - 수행계획서 - 요구사항 정의서 - 기능명세서 - API 명세서 - WBS - 인프라 시스템 아키텍처 - 테스트 보고서 - 회의록 - 중간/최종 발표자료 - 클라우드 사용 계획서 - ERD - 화면 설계서