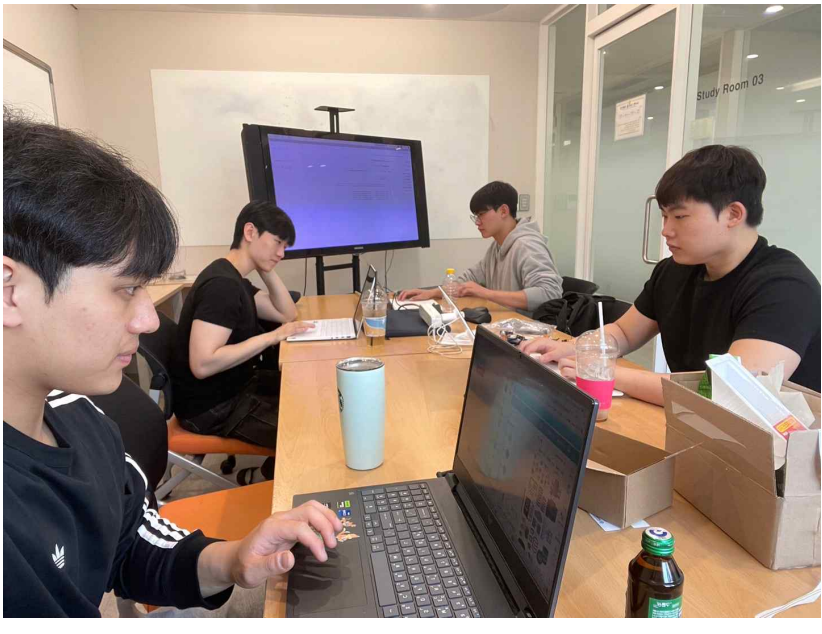


실증적SW개발프로젝트 주간보고 (6주차)

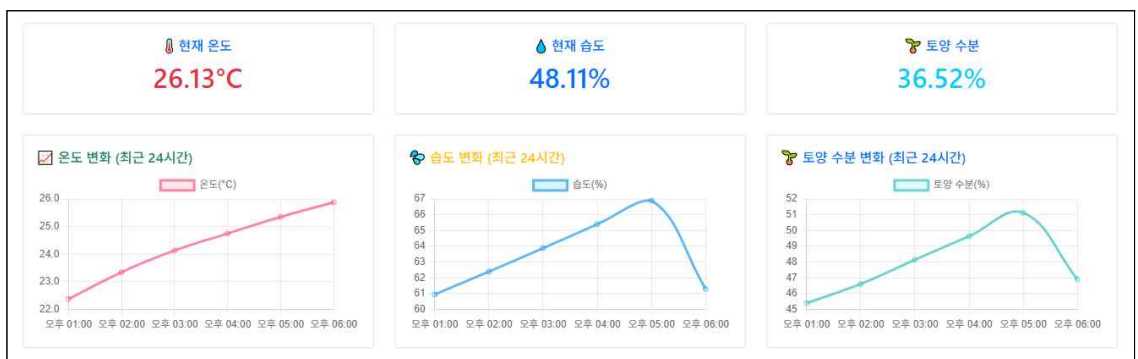
작성일: 2025/04/11 팀: ACG

팀 활동 보고	활 동 일 시	4월 2주차
	장 소	한림도서관 스터디룸
	참 석 자	김한재, 김현우, 김형진, 조재희, 최혁진
	특 이 사 항	
이번주 진행사항	<p>● 팀 공통사항</p> <p>- 이번 주는 개별적으로 구축한 기능들을 연동하는 작업을 중심으로 진행</p> <p>[구체사항]</p> <p>1. 웹앱과 TimescaleDB 기반 데이터베이스를 연동하여 실시간 센서 데이터를 조회 및 저장할 수 있는 구조 연동작업</p> <p>2. AWS 기반의 클라우드 서버를 중심으로 웹과 라즈베리파이, ESP32 간의 연결 구조를 구축하고, 센서 데이터 송수신과 제어 명령 전달이 이루어지도록 연동작업</p> <p>3. 라즈베리파이와 ESP32 간의 MQTT 및 HTTP 통신 구조를 통해 수집된 센서 데이터를 처리하고, 제어 명령을 전달하는 구조를 구현함으로써 시스템 전체의 데이터 흐름과 제어 연동작업</p> <p>● 팀원 사진</p>	
		

● 통신 진행 사진

		
ESP32	RaspberryPi	Cloud Server (AWS)

● 센서데이터 통신 인터페이스



● 문제 해결 내용

▶ 통신 프로토콜 문제점 및 해결방안

1. Raspberry Pi ↔ Cloud Server(학교 서버) 간 HTTP 통신의 한계

– Raspberry Pi에서 Cloud Server(학교 서버)로의 HTTP 요청은 정상적으로 이루어졌으나, 반대 방향인 Cloud Server에서 Raspberry Pi로의 HTTP 요청은 실패

– 이는 Raspberry Pi가 내부 네트워크에서 Node.js 서버를 구동하고 있었기 때문에, 외부에서 직접 접근이 불가능한 구조였기 때문

→ 해결 방안: HTTP 통신 대신 MQTT 프로토콜을 활용한 양방향 통신 구조로 전환을 고려함

2. Raspberry Pi ↔ Cloud Server 간 MQTT 통신의 제약

– MQTT를 통한 통신을 시도했으나, 학교 서버는 외부 IP로의 접근을 차단하고 있어 Raspberry Pi가 해당 서버에 MQTT로 접속할 수 없었음

– 또한, 본인이 부여받은 서버 권한으로는 해당 방화벽 설정을 변경하는 것이 불가능하다고 판단

→ 해결 방안: 학교 서버 대신 AWS를 중계 서버로 활용하여 Raspberry Pi와 사용자 간의 통신을 중개하도록 로직을 재설계함

▶ 통신 프로토콜 로직 재설계 및 구축 결과

1. 사용자 연동을 위한 HTTP 통신 구현

- AWS를 활용하여 사용자(User)로부터의 원격 제어 명령을 수신하고, 센서 데이터를 사용자에게 전송하는 HTTP 기반 통신 기능을 구현

2. AWS를 중계 서버로 활용한 Raspberry Pi 연동 구현 (MQTT)

- Raspberry Pi로부터 수집한 센서 데이터를 AWS를 통해 송신하고, 사용자로부터 수신한 제어 명령을 다시 Raspberry Pi로 전달하는 MQTT 통신 로직을 구현

3. ESP32 ↔ Raspberry Pi 간의 MQTT 통신 구축

- ESP32(센서 및 제어 역할)와 Raspberry Pi 간에 직접적인 MQTT 통신을 구축하여, 양방향 데이터 송수신이 가능한 구조로 시스템을 구성

→ 이러한 재설계를 통해 네트워크 구조의 제약을 극복하고, 사용자와 디바이스 간 안정적인 양방향 통신을 실현할 수 있었음

● 개별 진행사항

[김한재]

- 주간 리뷰를 통해 모든 팀원이 InfluxDB의 쿼리 언어에 대한 진입장벽을 느낀 점을 고려하여, 보다 익숙한 SQL 문법을 사용할 수 있는 PostgreSQL 기반의 시계열 데이터베이스인 TimescaleDB를 도입.

- TimescaleDB를 활용한 테이블 설계 및 데이터베이스 구축을 완료

[김현우]

- ESP32와 연동되는 센서 데이터 수집 코드 효율적으로 관리하고 확장 가능하도록 하기 위해 모듈화 작업을 병행

- 센서별 기능을 분리하고 공통된 구조를 통합하는 방식으로 코드를 설계 및 구축 이후 구조적 개선 진행

	<div data-bbox="344 96 1476 387"> <p>[김형진]</p> <ul style="list-style-type: none"> - 상추가 자라기 위한 최적의 환경을 정밀하게 조사하고 이를 기반으로 적절한 온도, 습도, 조도 등의 기준 값을 설정 - 상추 성장 조건 기반의 환경 설정 중심으로 환경 및 코드 구축 </div> <div data-bbox="344 387 1476 719"> <p>[조재희]</p> <ul style="list-style-type: none"> - Admin 페이지와 사용자 페이지를 분리하여 각 페이지의 역할에 맞는 기능을 구현 - Node.js 기반 서버 API 구축 및 TimescaleDB 연동 - JWT를 활용한 로그인 및 사용자 인증 기능 구현 </div> <div data-bbox="344 719 1476 1160"> <p>[최혁진]</p> <ul style="list-style-type: none"> - ESP32 <-> RaspberryPi MQTT 통신, RaspberryPi <-> CloudServer MQTT 통신, CloudServer <-> User HTTP 통신 구조를 통해 수집된 센서 데이터를 처리 - 제어 명령을 전달하는 구조를 구현함으로써 시스템 전체의 데이터 흐름과 제어 연동작업 - 통신 프로토콜 문제점 분석, 로직 재설계 및 구축 </div>
<p>다음주 계획</p>	<ul style="list-style-type: none"> ● 팀 공통사항 <ul style="list-style-type: none"> - 구비한 센서와 액추에이터를 연동하여, 실시간 센서 데이터를 기반으로 구축된 시스템 내에서 액추에이터를 자동 제어할 수 있도록 구현 ● 개별사항

[김한재]

- 센서값에 따른 액추에이터 동작 조건(임계값 위주) 정의 및 구현

[김현우]

- 각 센서(온습도, 조도, 토양, CO2 등)의 데이터를 정확히 읽어오는 코드 구현

[김형진]

- 수집된 센서 데이터를 기반으로 이상 패턴 감지 또는 통계 분석

[조재희]

- 클라우드에 저장된 데이터를 시각화하여 사용자에게 표시
- 액추에이터 수동 제어 UI 테스트

[최혁진]

- MQTT 토큰별로 RaspberryPi, AWS의 NodeJS 모듈화
- AWS에 설계된 TimescalDB를 도입하기 위한 AWS TimescalDB 구축 세팅 및 도입