네, 물론입니다. '식물 건강 신호등' 아이디어를 반영하여, 아두이노가 식물 상태를 판단하고 LED로 표시하며, 파이썬 Streamlit이 이 모든 정보를 받아와 웹 대시보드에 보여주는 통합 프로젝트의 회로도와 전체 코드를 작성해 드릴게요.

## 🔌 회로 연결 방법

온습도 센서 (DHT11): DATA → D2, VCC → 5V, GND → GND

조도센서: 신호선 → A0 (10kΩ 저항과 함께 연결)

토양 수분 센서: AOUT → A1, VCC → 5V, GND → GND

초록색 LED: 긴 다리(+) → 220Ω 저항 → D12

빨간색 LED: 긴 다리(+) → 220Ω 저항 → D13

모든 LED의 짧은 다리(-)는 GND에 연결합니다.

## 1단계: 아두이노 코드 📝

먼저 아두이노 IDE에서 DHT sensor library를 설치해야 합니다. (툴 > 라이브리 관리에서 DHT sensor library by Adafruit 검색 후 설치)

아래 코드는 센서 값을 읽고 식물 상태를 'Good' 또는 'Bad'로 판단한 뒤, LED를 제어하고 모든 데이터를 쉼표로 구분하여 컴퓨터로 전송합니다.

C++

#include <DHT.h>

// --- 핀 설정 ---

#define DHTPIN 2

#define DHTTYPE DHT11

const int LDR\_PIN = A0;

const int SOIL\_PIN = A1;

const int GREEN\_LED\_PIN = 12;

const int RED\_LED\_PIN = 13;

// --- 식물 상태 판단 기준값 (※환경에 맞게 조절하세요!) ---

const int SOIL\_THRESHOLD = 400; // 이 값보다 낮으면 '건조'

const float TEMP\_LOW = 18.0; // 이 값보다 낮으면 '추움'

const float TEMP\_HIGH = 30.0; // 이 값보다 높으면 '더움'

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

pinMode(GREEN\_LED\_PIN, OUTPUT);

pinMode(RED\_LED\_PIN, OUTPUT);

}

void loop() {

// 1. 센서 값 읽기

int soilMoisture = analogRead(SOIL\_PIN);

float humidity = dht.readHumidity();

float temperature = dht.readTemperature();

int lightValue = analogRead(LDR\_PIN);

if (isnan(humidity) || isnan(temperature)) {

Serial.println("DHT11 센서 읽기 실패!");

return;

}

// 2. 식물 상태 판단

String plantStatus = "Good"; // 기본 상태는 '좋음'

if (soilMoisture < SOIL\_THRESHOLD || temperature < TEMP\_LOW || temperature > TEMP\_HIGH) {

plantStatus = "Bad"; // 조건 중 하나라도 나쁘면 '나쁨'

}

// 3. LED 제어

if (plantStatus == "Good") {

digitalWrite(GREEN\_LED\_PIN, HIGH);

digitalWrite(RED\_LED\_PIN, LOW);

} else {

digitalWrite(GREEN\_LED\_PIN, LOW);

digitalWrite(RED\_LED\_PIN, HIGH); // 상태가 나쁘면 빨간불만 켜기

}

// 4. 파이썬으로 데이터 전송 (쉼표로 구분)

Serial.print(soilMoisture);

Serial.print(",");

Serial.print(humidity, 1);

Serial.print(",");

Serial.print(temperature, 1);

Serial.print(",");

Serial.print(lightValue);

Serial.print(",");

Serial.println(plantStatus); // 마지막에 상태 정보 추가

delay(1000);

}

## 2단계: 파이썬 Streamlit 앱 코드 💻

아래 코드는 아두이노가 보낸 데이터를 받아 실시간 대시보드를 만듭니다. pyserial 라이브러리가 필요합니다.

Bash

# 터미널에 아래 명령어 입력하여 라이브러리 설치

pip install streamlit pyserial pandas

Python

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

import streamlit as st

import serial

import time

import pandas as pd

st.set\_page\_config(page\_title="🌱 스마트 화분 대시보드", layout="wide")

st.title("🌱 스마트 화분 실시간 대시보드")

st.markdown("아두이노 센서 데이터를 실시간으로 모니터링합니다.")

# 사이드바: 연결 설정

st.sidebar.header("연결 설정")

port = st.sidebar.text\_input("아두이노 포트", "COM3")

if 'ser' not in st.session\_state:

st.session\_state.ser = None

if 'is\_running' not in st.session\_state:

st.session\_state.is\_running = False

if st.sidebar.button("🔌 연결 시작"):

try:

st.session\_state.ser = serial.Serial(port, 9600, timeout=1)

st.session\_state.is\_running = True

st.sidebar.success(f"{port}에 연결되었습니다.")

time.sleep(2)

except serial.SerialException as e:

st.sidebar.error(f"연결 실패: {e}")

st.session\_state.is\_running = False

if st.sidebar.button("❌ 연결 중지"):

st.session\_state.is\_running = False

if st.session\_state.ser and st.session\_state.ser.is\_open:

st.session\_state.ser.close()

st.sidebar.info("연결이 중지되었습니다.")

# --- 데이터 디스플레이 ---

if st.session\_state.is\_running:

st.header("📊 실시간 센서 데이터")

col1, col2, col3, col4, col5 = st.columns(5)

placeholder1 = col1.empty()

placeholder2 = col2.empty()

placeholder3 = col3.empty()

placeholder4 = col4.empty()

status\_placeholder = col5.empty() # 상태 표시를 위한 공간

chart\_placeholder = st.empty()

df = pd.DataFrame(columns=['수분', '습도', '온도', '조도'])

while st.session\_state.is\_running:

try:

line = st.session\_state.ser.readline().decode('utf-8').strip()

if line:

data = line.split(',')

if len(data) == 5:

soil, hum, temp, light, status = data

# 메트릭 업데이트

placeholder1.metric("💧 토양 수분", soil)

placeholder2.metric("🌫️ 공중 습도", f"{hum} %")

placeholder3.metric("🌡️ 온도", f"{temp} °C")

placeholder4.metric("☀️ 조도", light)

# 상태 업데이트

if status == "Good":

status\_placeholder.success("🟢 행복함")

else:

status\_placeholder.error("🔴 도움 필요")

# 차트 데이터 추가 및 업데이트

new\_data = pd.DataFrame([[float(soil), float(hum), float(temp), float(light)]], columns=['수분', '습도', '온도', '조도'])

df = pd.concat([df, new\_data], ignore\_index=True)

chart\_placeholder.line\_chart(df.tail(30))

except Exception as e:

st.error(f"데이터 읽기 오류: {e}")

st.session\_state.is\_running = False

time.sleep(0.1)

else:

st.info("사이드바에서 아두이노 포트를 설정하고 '연결 시작' 버튼을 누르세요.")

## 실행 방법 🚀

1단계의 아두이노 코드를 보드에 업로드합니다.

아두이노 IDE의 시리얼 모니터를 반드시 닫아주세요.

터미널에서 2단계의 파이썬 코드가 저장된 폴더로 이동합니다.

터미널에 streamlit run your\_script\_name.py 명령어를 입력하여 Streamlit 앱을 실행합니다.

웹 브라우저가 열리면, 사이드바에 자신의 아두이노 포트를 입력하고 '연결 시작' 버튼을 누르면 실시간 데이터와 식물 상태가 표시됩니다.

양 수분, 온도, 습도, 조도 가 각각 원래 몇이야 적정수준이고 아니면 나쁨수준이야?

네, 식물 종류마다 최적의 환경이 다르지만, 일반적인 실내 관엽 식물을 기준으로 \*\*'적정 수준'\*\*과 \*\*'주의가 필요한 나쁨 수준'\*\*을 알려드릴게요.

## 💧 토양 수분 (Soil Moisture)

아두이노 토양 수분 센서는 보통 \*\*아날로그 값 (0~1023)\*\*으로 측정됩니다. 센서마다, 흙의 종류마다 값이 다르므로 반드시 직접 테스트해야 합니다.

적정 수준 (손가락으로 흙을 만졌을 때 촉촉함): 400 ~ 700

이 범위는 물을 준 직후가 아니라, 흙이 적당히 수분을 머금고 있는 상태입니다.

나쁨 수준 (물 부족): 0 ~ 300

값이 300 이하로 떨어지면 흙이 마르기 시작했다는 신호이며, 물을 줄 준비를 해야 합니다.

나쁨 수준 (과습): 750 이상

값이 너무 높게 유지되면 흙이 항상 젖어있다는 뜻으로, 뿌리가 썩을 위험이 있습니다.

## 🌡️ 온도 (Temperature)

대부분의 실내 식물은 사람이 쾌적하다고 느끼는 온도에서 잘 자랍니다.

적정 수준: 18 ~ 26°C

이 범위는 식물의 성장과 광합성이 가장 활발한 온도입니다.

나쁨 수준 (저온/고온): 15°C 미만 또는 30°C 초과

온도가 너무 낮으면 냉해를 입거나 성장이 멈출 수 있고, 너무 높으면 잎이 타거나 시들 수 있습니다.

## 🌫️ 습도 (Humidity)

실내 식물은 생각보다 높은 습도를 좋아합니다. 특히 겨울철 건조한 환경에 주의해야 합니다.

적정 수준: 40 ~ 60%

이 정도 습도는 식물의 잎이 마르는 것을 방지하고 건강하게 유지해 줍니다.

나쁨 수준 (건조): 40% 미만

습도가 너무 낮으면 잎 끝이 갈색으로 마르거나, 병충해가 생기기 쉽습니다. 가습기를 사용하거나 잎에 분무를 해주는 것이 좋습니다.

## ☀️ 조도 (Light)

조도 센서는 \*\*아날로그 값 (0~1023)\*\*으로 빛의 상대적인 세기를 측정합니다.

적정 수준 (밝은 간접광): 400 ~ 800

창문을 통해 들어오는 밝은 거실 정도의 빛입니다. 대부분의 실내 식물이 이 정도의 빛을 좋아합니다.

나쁨 수준 (빛 부족): 300 미만

빛이 부족하면 식물이 웃자라거나 잎의 색이 옅어질 수 있습니다.

나쁨 수준 (과도한 직사광선): 850 이상

한여름의 직사광선은 오히려 잎을 태울 수 있으므로, 일부 식물에게는 좋지 않습니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

기존 스마트 화분 시스템에 \*\*조도 센서(LDR)\*\*와 \*\*온습도 센서(DHT11/DHT22)\*\*를 추가하여, 총 세 가지 환경 요인(습도, 온도, 빛)을 모니터링하고 스트림릿 앱에서 실시간으로 보여주는 회로도와 코드를 업데이트해 드리겠습니다.

1. 회로도 (Wiring Diagram) 업데이트

부품 핀 이름 아두이노 연결 핀 기능

토양 수분 센서 AOUT A0 토양 수분 레벨 (0~1023) 측정

DHT11/DHT22 온습도 센서 DATA D2 (디지털 핀 2) 온도 및 습도 데이터 전송

조도 센서 (CDS/LDR) Signal/Output A1 주변 빛의 밝기 (0~1023) 측정

아두이노 보드 5V, GND 5V, GND 전원 및 접지

직렬 통신 USB PC/노트북 데이터 전송 및 전원 공급

Sheets로 내보내기

회로 연결 순서:

토양 수분 센서: AOUT → A0, VCC → 5V, GND → GND

DHT11/DHT22 센서: Data → D2, VCC → 5V, GND → GND (센서에 따라 저항 필요)

조도 센서 (LDR 모듈): Signal → A1, VCC → 5V, GND → GND (LDR 단품 사용 시 풀다운/풀업 저항 필요)

2. 아두이노 코드 (Arduino Code) 업데이트

DHT 센서 라이브러리가 필요합니다. 아두이노 IDE의 스케치 → 라이브러리 포함하기 → 라이브러리 관리에서 \*\*"DHT sensor library"\*\*를 검색하여 설치해 주세요.

C++

#include "DHT.h"

// --- 센서 핀 및 타입 정의 ---

#define DHTPIN 2 // DHT11 온습도 센서가 연결된 디지털 2번 핀

#define DHTTYPE DHT11 // 사용하는 DHT 센서 타입

const int SOIL\_PIN = A0; // 토양 수분 센서가 연결된 아날로그 A0번 핀

// DHT 센서 객체 생성

DHT dht(DHTPIN, DHTTYPE);

void setup() {

// 시리얼 통신을 9600 속도로 시작

Serial.begin(9600);

// DHT 센서 초기화

dht.begin();

}

void loop() {

// 1. 각 센서로부터 데이터 읽기

float humidity = dht.readHumidity(); // 습도 값 읽기

float temperature = dht.readTemperature(); // 온도 값 읽기

int soilMoisture = analogRead(SOIL\_PIN); // 토양 수분 값 읽기 (0~1023)

// 2. DHT 센서 읽기 실패 시 오류 처리

// isnan() 함수는 값이 숫자가 아닐 때 true를 반환합니다.

if (isnan(humidity) || isnan(temperature)) {

// 오류가 발생하면 이번 반복은 건너뛰어 다음 측정을 시도합니다.

// 이렇게 하면 파이썬으로 비정상적인 텍스트가 전송되는 것을 막을 수 있습니다.

return;

}

// 3. 파이썬이 처리하기 좋은 형태로 데이터 출력 (가장 중요한 부분)

// 형식: 온도,습도,토양수분

Serial.print(temperature);

Serial.print(",");

Serial.print(humidity);

Serial.print(",");

Serial.println(soilMoisture); // 마지막 값은 println으로 출력하여 줄바꿈

// 2초(2000ms) 대기 후 다음 측정 시작

delay(2000);

}

3. 파이썬 (Streamlit) 코드 업데이트

import streamlit as st

import serial

import serial.tools.list\_ports

import time

import pandas as pd

# --- 1. 페이지 기본 설정 및 초기화 ---

st.set\_page\_config(

page\_title="🌿 스마트 화분 실시간 대시보드",

layout="wide",

initial\_sidebar\_state="expanded"

)

# --- 식물별 기본 프로필 데이터 ---

PLANT\_PROFILES = {

"🌿 관엽식물 (일반)": {

"soil\_threshold": 600,

"temp\_range": (18, 28),

"humid\_range": (40, 70)

},

"🌵 다육식물": {

"soil\_threshold": 750, # 건조하게 관리

"temp\_range": (15, 30),

"humid\_range": (20, 50)

},

"💧 열대식물 (습함)": {

"soil\_threshold": 450,

"temp\_range": (20, 32),

"humid\_range": (60, 85)

}

}

# --- 2. 함수 정의 ---

@st.cache\_data

def find\_serial\_ports():

"""사용 가능한 시리얼 포트 목록을 찾아서 반환"""

return [port.device for port in serial.tools.list\_ports.comports()]

def initialize\_session\_state():

"""앱 세션 상태를 초기화"""

if 'connected' not in st.session\_state:

st.session\_state.connected = False

if 'data' not in st.session\_state:

st.session\_state.data = pd.DataFrame(columns=['시간', '온도', '습도', '토양수분'])

if 'plant\_profile\_name' not in st.session\_state:

st.session\_state.plant\_profile\_name = "🌿 관엽식물 (일반)"

def data\_to\_csv(df):

"""데이터프레임을 CSV로 변환"""

return df.to\_csv(index=False).encode('utf-8-sig') # 엑셀에서 한글 깨짐 방지

# --- 앱 상태 초기화 실행 ---

initialize\_session\_state()

# --- 3. 사이드바 UI ---

with st.sidebar:

st.title("🪴 스마트 화분 제어판")

st.divider()

# --- 연결 설정 ---

if not st.session\_state.connected:

st.header("1. 연결 설정")

available\_ports = find\_serial\_ports()

if not available\_ports:

st.warning("아두이노를 연결해주세요.")

else:

selected\_port = st.selectbox("아두이노 포트를 선택하세요", available\_ports)

if st.button("연결하기", type="primary"):

try:

ser = serial.Serial(selected\_port, 9600, timeout=2)

time.sleep(2) # 연결 안정화를 위한 대기

st.session\_state.serial\_connection = ser

st.session\_state.connected = True

st.success(f"{selected\_port}에 연결되었습니다.")

st.rerun()

except serial.SerialException as e:

st.error(f"연결 실패: {e}")

else:

st.success(f"{st.session\_state.serial\_connection.port}에 연결됨")

if st.button("연결 끊기"):

if 'serial\_connection' in st.session\_state and st.session\_state.serial\_connection:

st.session\_state.serial\_connection.close()

# 모든 세션 상태 초기화 후 재실행

for key in list(st.session\_state.keys()):

del st.session\_state[key]

st.rerun()

st.divider()

# --- 목표 환경 설정 (연결된 경우에만 표시) ---

if st.session\_state.connected:

st.header("2. 목표 환경 설정")

# 식물 프로필 선택

selected\_plant = st.selectbox(

"식물 프로필 선택",

list(PLANT\_PROFILES.keys()),

index=list(PLANT\_PROFILES.keys()).index(st.session\_state.plant\_profile\_name)

)

# 프로필이 변경되면 세션 상태 업데이트 후 재실행

if selected\_plant != st.session\_state.plant\_profile\_name:

st.session\_state.plant\_profile\_name = selected\_plant

st.rerun()

profile = PLANT\_PROFILES[st.session\_state.plant\_profile\_name]

# 목표 값 수동 조절 슬라이더

st.subheader("🎯 목표 값 수동 조절")

st.session\_state.temp\_target = st.slider(

"목표 온도 범위 (°C)", 0, 50, profile['temp\_range']

)

st.session\_state.humid\_target = st.slider(

"목표 습도 범위 (%)", 0, 100, profile['humid\_range']

)

st.session\_state.soil\_target = st.slider(

"물주기 기준선", 0, 1023, profile['soil\_threshold'],

help="값이 이 기준선보다 높아지면(더 건조해지면) '물 부족'으로 진단합니다."

)

# --- 4. 메인 대시보드 UI ---

if not st.session\_state.connected:

st.info("👈 사이드바에서 아두이노 포트를 선택하고 '연결하기'를 눌러주세요.")

else:

# --- 데이터 수신 및 처리 ---

if st.session\_state.serial\_connection and st.session\_state.serial\_connection.in\_waiting > 0:

try:

line = st.session\_state.serial\_connection.readline().decode('utf-8').rstrip()

# 아두이노에서 "온도,습도,토양수분" 순서로 보낸 데이터를 파싱

temp, humid, soil = map(float, line.split(','))

# 새 데이터를 데이터프레임에 추가

now = pd.Timestamp.now(tz='Asia/Seoul')

new\_data = pd.DataFrame([{'시간': now, '온도': temp, '습도': humid, '토양수분': soil}])

st.session\_state.data = pd.concat([st.session\_state.data, new\_data], ignore\_index=True).tail(100)

except Exception as e:

# 데이터 수신 중 오류 발생 시 화면에 표시

st.error(f"🚨 데이터 처리 중 오류: {e}")

st.warning("아두이노에서 보내는 데이터 형식이 '숫자,숫자,숫자' 형태인지 확인해주세요.")

st.header(f"'{st.session\_state.plant\_profile\_name}' 관리 대시보드")

# --- 데이터가 있을 때만 대시보드 표시 ---

if not st.session\_state.data.empty:

latest\_data = st.session\_state.data.iloc[-1]

# 목표 값 가져오기

temp\_min, temp\_max = st.session\_state.temp\_target

humid\_min, humid\_max = st.session\_state.humid\_target

soil\_threshold = st.session\_state.soil\_target

# --- 현재 상태 진단 ---

st.subheader("🚨 현재 상태 진단")

col1, col2, col3 = st.columns(3)

with col1: # 토양 수분 진단

if latest\_data['토양수분'] > soil\_threshold:

st.error(f"💧 물 부족! (기준 초과)")

else:

st.success("✅ 토양 수분 적정")

with col2: # 온도 진단

if latest\_data['온도'] > temp\_max:

st.error(f"🔥 고온 주의! (기준 초과)")

elif latest\_data['온도'] < temp\_min:

st.warning(f"❄️ 저온 주의! (기준 미달)")

else:

st.success("✅ 적정 온도")

with col3: # 습도 진단

if latest\_data['습도'] > humid\_max:

st.warning(f"💧 과습 주의! (기준 초과)")

elif latest\_data['습도'] < humid\_min:

st.warning(f"🌵 건조 주의! (기준 미달)")

else:

st.success("✅ 적정 습도")

st.divider()

# --- 실시간 수치, 차트, 데이터 테이블 ---

st.subheader("📈 실시간 수치 및 변화 추이")

m\_col1, m\_col2, m\_col3 = st.columns(3)

m\_col1.metric("🌡️ 현재 온도", f"{latest\_data['온도']:.1f} °C")

m\_col2.metric("💧 현재 습도", f"{latest\_data['습도']:.1f} %")

m\_col3.metric("흙 토양 수분", f"{latest\_data['토양수분']:.0f}", help="값이 높을수록 건조합니다.")

# 데이터 변화 추이 차트

chart\_data = st.session\_state.data.set\_index('시간')

st.line\_chart(chart\_data)

# 데이터 기록 및 다운로드

with st.expander("📂 데이터 기록 보기 및 다운로드"):

st.dataframe(st.session\_state.data.sort\_values(by='시간', ascending=False))

st.download\_button(

label="💾 데이터 기록 다운로드 (CSV)",

data=data\_to\_csv(st.session\_state.data),

file\_name=f"plant\_log\_{time.strftime('%Y%m%d\_%H%M%S')}.csv",

mime="text/csv"

)

else:

st.info("데이터를 수신하는 중입니다. 잠시만 기다려주세요...")

# 1초마다 자동 새로고침

time.sleep(1)

st.rerun()ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

 **💧 과습 상태 (0 ~ 300):** 흙이 물에 잠겨 있거나 축축한 상태입니다. 대부분의 식물은 뿌리가 썩을 수 있어 위험합니다.

 **🌿 최적 습도 (300 ~ 600):** 손으로 만졌을 때 촉촉함이 느껴지는 이상적인 상태입니다. 대부분의 관엽식물, 채소 등이 이 범위에서 잘 자랍니다.

 **🍂 건조 상태 (600 이상):** 흙에 수분이 부족한 상태로, 물을 줘야 할 시점입니다.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

토양수분센서와 온습도센서를 활용하면, 단순히 데이터를 보여주는 것을 넘어 훨씬 유용하고 실용적인 \*\*'스마트 화분/농장 관리 대시보드'\*\*를 만들 수 있습니다.

기존에 만들었던 앱에서 한 단계 더 나아가, 식물의 상태를 진단하고 맞춤 관리하는 기능을 추가하는 아이디어입니다.

## 🪴 스마트 화분 관리 도우미 앱

이 앱의 핵심 목표는 센서 데이터를 바탕으로 식물의 현재 상태를 진단하고, 사용자에게 필요한 조치(물주기 등)를 알려주는 것입니다.

## 🌟 주요 기능

1. 상태 진단 및 알림 시스템

현재 값만 보여주는 대신, 미리 설정된 \*\*'적정 범위'\*\*를 기준으로 식물의 상태를 '좋음', '주의', '나쁨' 등으로 진단해 시각적으로 보여줍니다.

(예시)

토양 수분이 너무 낮으면: kırmızı"매우 건조! 물을 주세요! 💧"

온도가 너무 높으면: kırmızı"너무 더워요! 환기가 필요해요. 🔥"

모든 조건이 좋으면: yeşil"최적의 상태를 유지하고 있어요! 👍"

2. 식물별 맞춤 설정 기능

모든 식물이 동일한 환경을 필요로 하지는 않습니다. 식물 종류(다육식물, 관엽식물, 허브 등)를 선택하면, 그에 맞는 최적의 온/습도 및 토양 수분 기준값이 자동으로 설정되도록 합니다.

(예시)

다육식물 선택 시: 건조 기준을 높게 설정 (물이 적게 필요)

열대식물 선택 시: 습도와 토양 수분 기준을 높게 설정

3. 데이터 기록 및 다운로드

측정된 데이터를 앱이 실행되는 동안 CSV 파일로 자동 저장합니다. 사용자는 이 데이터를 다운로드하여 장기간의 환경 변화 추이를 엑셀 등으로 분석할 수 있습니다.

## 💻 화면 구성 예시

Streamlit의 st.columns와 st.sidebar를 활용하여 다음과 같이 구성할 수 있습니다.

사이드바 (제어판)

연결 설정: COM 포트 선택 및 연결 버튼

🌱 식물 선택: 드롭다운 메뉴로 관리할 식물 프로필 선택 (예: 다육이, 몬스테라, 바질)

메인 화면

🚨 현재 상태 진단: 가장 상단에 현재 식물 상태를 큰 아이콘과 텍스트로 표시 (예: "💧 물이 부족해요!")

실시간 수치: 온도, 습도, 토양 수분 값을 게이지나 메트릭으로 표시

📈 데이터 추이: 시간대별 센서 값 변화를 보여주는 실시간 라인 차트

💾 데이터 관리: "데이터 기록 다운로드" 버튼

## 👨‍💻 구현을 위한 핵심 코드 예시

기존 코드에 아래와 같은 로직을 추가하여 구현할 수 있습니다.