**1. 프로젝트 목표**

* **목적**: Arduino와 Raspberry Pi를 이용한 스마트팜 시스템에서 **웹 페이지를 통해 유저가 직접 온도와 습도 값을 설정**하고, **카메라 모듈을 통해 실시간 식물 상태를 모니터링**하며, **캘린더를 통해 특정 시간대에 사진을 저장**하는 기능을 제공하는 시스템을 구축.
* **핵심 기능**:
  1. **웹 페이지에서 온도 및 습도 설정**.
  2. 설정된 값에 따라 **Arduino에서 릴레이를 제어**해 분무기(수중 모터)와 환풍기를 작동.
  3. **실시간 온습도 모니터링** 및 **자동 환경 유지**.
  4. **Raspberry Pi 카메라 모듈을 통해 실시간 영상 스트리밍**으로 식물 상태 모니터링.
  5. **캘린더 기능**을 통해 **특정 시간대에 사진을 저장**하고, 저장된 사진을 웹 페이지에서 확인 가능.

**2. 프로젝트 개요 및 구성 요소**

1. **Arduino Mega**:
   * **센서 및 액츄에이터 제어**.
   * **DHT11 센서**로부터 온도와 습도 데이터를 수집.
   * **릴레이 모듈**을 통해 환풍기(온도 제어)와 분무기(습도 제어)를 작동.
   * **Raspberry Pi**로부터 **유저가 설정한 온습도 값을 수신**하여 제어 로직 수행.
2. **Raspberry Pi**:
   * **웹 서버 호스팅**: **Django**를 사용하여 유저가 접근할 수 있는 웹 페이지 제공.
   * **카메라 모듈 연결**을 통해 **실시간 영상 스트리밍**.
   * **시리얼 통신**을 통해 유저가 설정한 온습도 값을 **Arduino로 전송**.
   * **캘린더 기능**을 통해 유저가 선택한 시간대에 **자동으로 사진 촬영 및 저장**.
3. **DHT11 온습도 센서**:
   * 현재의 **온도와 습도 데이터를 Arduino로 제공**하여 실시간 환경 정보를 바탕으로 제어 로직 수행.
4. **수중 모터 및 환풍기**:
   * **릴레이 모듈을 통해 제어**되며, 설정된 온도 및 습도에 맞춰 동작.
   * **릴레이 모듈을 통한 펌프와 환풍기 제어 :** 
     + 펌프 제어:
     + \*\*펌프(12V)\*\*는 릴레이의 NO 단자에 연결되어 있으며, COM 단자는 12V 전원과 연결됩니다.
     + Arduino가 습도가 50% 미만일 때, IN 핀에 HIGH 신호를 보내서 릴레이를 활성화합니다.
     + 활성화되면 COM과 NO가 연결되어 펌프에 12V 전원이 공급되고, 물을 분사하게 됩니다.
     + 습도가 50% 이상이 되면, IN 핀에 LOW 신호를 보내서 릴레이가 비활성화되고 펌프가 꺼집니다.
     + 환풍기 제어:
     + 환풍기(12V) 역시 릴레이의 NO 단자에 연결되어 있으며, COM 단자는 12V 전원과 연결됩니다.
     + Arduino가 온도가 23°C 이상일 때, IN 핀에 HIGH 신호를 보내서 릴레이를 활성화합니다.
     + 활성화되면 COM과 NO가 연결되어 환풍기에 12V 전원이 공급되고, 환풍기가 작동하여 온도를 낮추게 됩니다.
     + 온도가 23°C 미만으로 떨어지면, IN 핀에 LOW 신호를 보내서 릴레이가 비활성화되고 환풍기가 꺼집니다.
5. **릴레이 모듈(SRD-05VDC-SL-C)**:
   * **Arduino**에서 제어 신호를 받아 **12V 수중 모터와 환풍기의 전원을 스위칭**하여 작동.
6. **Raspberry Pi 카메라 모듈**:
   * **실시간 영상 스트리밍**을 통해 웹 페이지에서 **식물 상태 모니터링** 가능.
   * **캘린더 기능**을 통해 **자동 사진 촬영** 기능 제공.

**3. 작동 방식**

1. **유저 인터페이스 (웹 페이지)**:
   * 유저가 웹 페이지를 통해 원하는 **온도**(예: 23°C)와 **습도**(예: 50%)를 설정.
   * 설정된 값이 **Django 웹 서버**로 전송됨.
   * 웹 페이지에서 **실시간 카메라 피드**를 통해 **식물 상태를 모니터링**.
   * 유저가 **캘린더를 통해 특정 날짜와 시간대**를 선택하면, 해당 시간에 사진이 **자동으로 촬영**되고 저장됨.
2. **Raspberry Pi**:
   * Django 서버가 **온도와 습도 값을 처리**하고, **시리얼 통신**을 통해 **Arduino로 설정값 전송**.
   * 카메라 모듈을 통해 실시간 영상을 **스트리밍 서버**를 통해 전송.
   * **캘린더를 통해 스케줄된 시간**에 맞춰 **사진 촬영 및 저장**.
3. **Arduino Mega**:
   * **Raspberry Pi로부터 수신한 온도와 습도 값을 적용**하여 릴레이를 제어.
   * **현재 온도와 습도가 설정 값보다 높거나 낮을 경우** 릴레이를 통해 **환풍기와 분무기**를 작동시켜 환경을 조절.
4. **자동 제어**:
   * **온도 23°C 이상**일 때 **환풍기 작동**.
   * **습도 50% 이하**일 때 **분무기 작동**(수중 모터를 사용하여 물 분사).
5. **식물 모니터링**:
   * Raspberry Pi에 연결된 **카메라 모듈**을 통해 **실시간으로 식물 상태**를 웹 페이지에서 스트리밍.
   * **특정 시간대에 자동으로 사진 촬영 및 저장**.

**4. 기술 스택**

1. **Arduino Mega**:
   * DHT11 센서를 사용해 온도 및 습도 측정.
   * 시리얼 통신을 통해 Raspberry Pi와 데이터 교환.
2. **Raspberry Pi (웹 서버 및 카메라)**:
   * **Django** 기반의 웹 서버 구축.
   * **카메라 모듈을 통한 실시간 영상 스트리밍**.
   * 유저로부터 설정 값을 받아 **Arduino에 전달**.
   * **캘린더 기능**을 통해 사진 촬영 스케줄 설정 및 실행.
3. **릴레이 모듈**:
   * 설정된 온습도 값에 따라 **환풍기 및 분무기 제어**.

**5. 캘린더 및 사진 촬영 기능 구현**

1. **Django 모델 설정**: **models.py**:

from django.db import models

class PhotoSchedule(models.Model):

date = models.DateField() # 날짜

time = models.TimeField() # 시간

photo\_path = models.CharField(max\_length=255, null=True, blank=True) # 사진 파일 경로

def \_\_str\_\_(self):

return f"{self.date} at {self.time}"

1. **웹 페이지 캘린더 폼**: **templates/schedule\_photo.html**:

<!DOCTYPE html>

<html>

<head>

<title>Schedule Plant Photo</title>

</head>

<body>

<h2>Schedule Photo Capture</h2>

<form method="POST">

{% csrf\_token %}

<label for="date">Select Date:</label>

<input type="date" id="date" name="date" required><br><br>

<label for="time">Select Time:</label>

<input type="time" id="time" name="time" required><br><br>

<input type="submit" value="Schedule Photo">

</form>

</body>

</html>

1. **뷰 설정**: **views.py**:

from django.shortcuts import render, redirect

from .models import PhotoSchedule

from datetime import datetime

def schedule\_photo(request):

if request.method == 'POST':

date = request.POST['date']

time = request.POST['time']

# 선택된 날짜와 시간을 저장

schedule = PhotoSchedule.objects.create(date=date, time=time)

return redirect('photo\_list')

return render(request, 'schedule\_photo.html')

def photo\_list(request):

# 사진 스케줄 리스트를 출력

schedules = PhotoSchedule.objects.all()

return render(request, 'photo\_list.html', {'schedules': schedules})

1. **스케줄된 시간에 사진 촬영**: **스케줄러 설정 (스케줄러.py)**:

import os

from datetime import datetime

from apscheduler.schedulers.background import BackgroundScheduler

from .models import PhotoSchedule

scheduler = BackgroundScheduler()

def capture\_photo(schedule\_id):

# 해당 스케줄의 사진 촬영 및 저장

schedule = PhotoSchedule.objects.get(id=schedule\_id)

photo\_path = f'/path/to/photos/{schedule.date}\_{schedule.time}.jpg'

os.system(f'raspistill -o {photo\_path}') # 카메라로 사진 촬영

schedule.photo\_path = photo\_path

schedule.save()

def schedule\_photos():

# 데이터베이스에서 스케줄된 시간을 가져와 스케줄링

schedules = PhotoSchedule.objects.all()

for schedule in schedules:

schedule\_time = datetime.combine(schedule.date, schedule.time)

scheduler.add\_job(capture\_photo, 'date', run\_date=schedule\_time, args=[schedule.id])

scheduler.start()

**6. 보완 및 추가 기능**

* **보안 강화**: 웹 페이지 접근에 대한 인증 시스템 추가.
* **실시간 데이터 피드백**: 현재 환경 상태를 유저에게 제공하여 실시간 모니터링 가능.
* **원격 제어 가능성**: 외부 네트워크를 통해 원격으로도 온습도 값을 설정할 수 있도록 확장.

**7. 기대 효과**

* **온습도 자동 제어**와 **실시간 모니터링**을 통해 농작물의 최적 환경 유지 가능.
* **캘린더 기능**을 통해 특정 시간대에 식물 사진을 저장하여 **장기적인 식물 상태 모니터링** 가능.
* **웹 기반 설정**으로 사용자가 간편하게 환경을 제어할 수 있으며, 이를 통해 **유연하고 효율적인 스마트팜 시스템**을 구축할 수 있음.

**8. 다음 단계**

1. **캘린더 기능 및 사진 저장 기능 테스트**.
2. **카메라 모듈을 통한 실시간 스트리밍 안정성 테스트**.
3. **Django 서버와 시리얼 통신의 안정성 테스트**.
4. **실제 스마트팜 환경에서의 테스트 및 피드백 반영**.