Smart Agriculture Project

MAHIB

May 31, 2025

Overview

This document describes the design and implementation of an Arduino-based automated plant care system. It uses temperature, humidity, and soil moisture sensors to control a fan and a water pump via relays.

Project Link

You can view and simulate this project online at: `https://wokwi.com/projects/432429538689503233`

Components Used

- Arduino Uno
- DHT22 Sensor (Temperature Sensor)
- Custom Soil Moisture sensor (mimics the YL-69 Sensor)
- Relay Modules to turn Fan and Pump ON/OFF
- Fan
- Pump
- (Optional) A serial monitor for Debugging and Logging
- (Optional) LED(s) for Reset and ON/OFF indications

Schematic

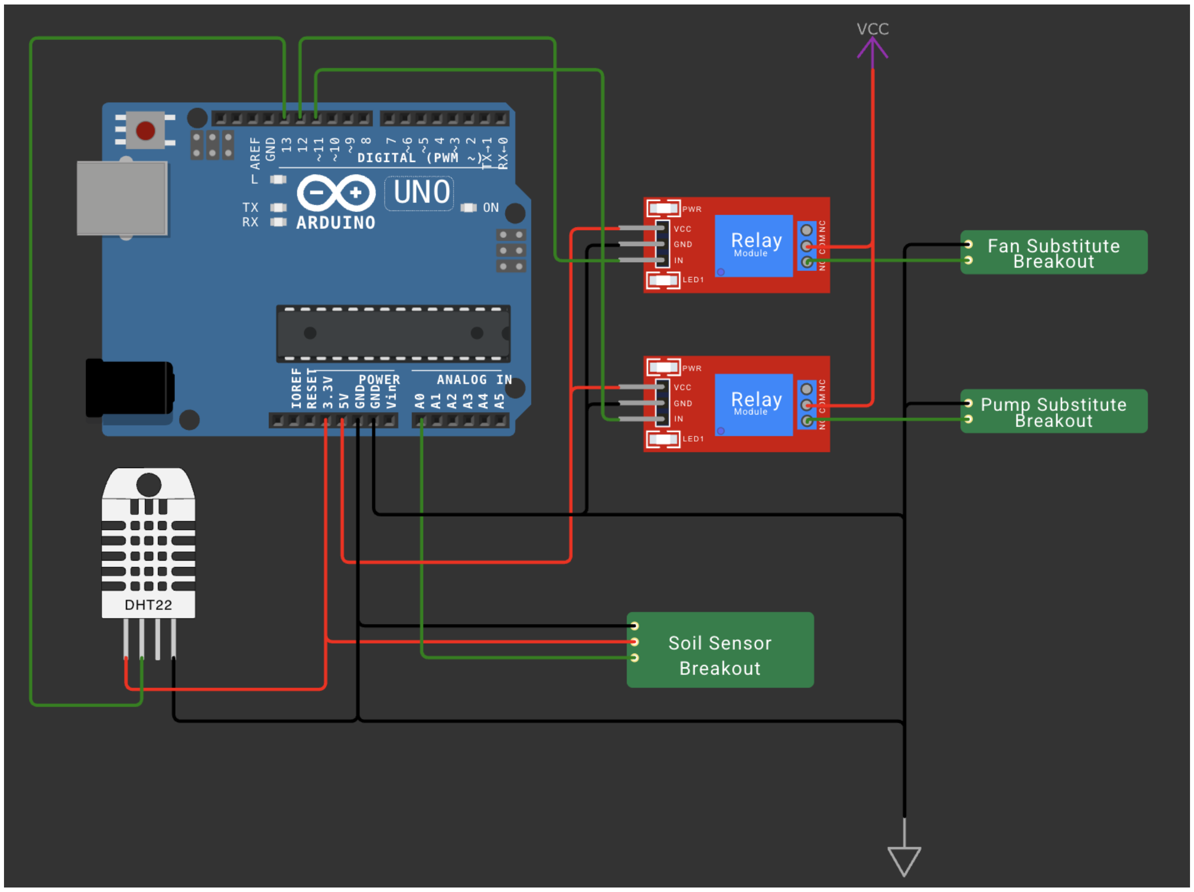

Figure 1: Wiring schematic of the system

Arduino Sketch

The main controller logic is written in Arduino C++. The sketch reads sensor values and activates relays as needed.

```
% Arduino sketch goes here
% Example:
#include <DHT.h>

// DHT22 CONFIG
#define DHTPIN 13
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// PINS
const int soilMoisturePin = A0;
const int fanRelayPin = 12;   // Relay 1
const int pumpRelayPin = 11;  // Relay 2

// THRESHOLDS
const float temperatureThreshold = 25.0; // °C
const int soilMoistureThreshold = 800;

// SETUP
void setup() {
  Serial.begin(115200);
  dht.begin();

  pinMode(fanRelayPin, OUTPUT);
  pinMode(pumpRelayPin, OUTPUT);

  // Initialize relays as OFF
  digitalWrite(fanRelayPin, LOW);
  digitalWrite(pumpRelayPin, LOW);
}

// MAIN LOOP
void loop() {
  // Read temperature from DHT22
  float temperature = dht.readTemperature();
  if (isnan(temperature)) {
    Serial.println("Failed to read temperature from DHT22!");
    return;
  }

  // Read soil moisture
  int soilValue = analogRead(soilMoisturePin);
  // LOGGING (For Debugging Purposes)
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Soil Moisture: ");
  Serial.println(soilValue);

  // CONTROL LOGIC
  // FAN CONTROL
  if (temperature > temperatureThreshold) {
    digitalWrite(fanRelayPin, HIGH);  // fan ON
  } else {
    digitalWrite(fanRelayPin, LOW);   // fan OFF
  }

  // PUMP CONTROL
  if (soilValue < soilMoistureThreshold) {
    digitalWrite(pumpRelayPin, HIGH);  // pump ON
  } else {
    digitalWrite(pumpRelayPin, LOW);   // pump OFF
  }

  delay(2000);  // Delay for stability
}
```

Custom Chip – C Source File

This is the simulated behavior logic for a custom component (e.g., soil sensor or fan) defined in a '.chip.c' file.
Given is the code for the Soil Moisture Sensor Chip. The other chips also have similar .chip.c files which can be viewed in the project via the link given in the **??** section.

```
#include "wokwi-api.h"
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  pin_t pin;
  float value;
} chip_data_t;

void chip_timer_callback(void *data) {
  chip_data_t *chip_data = (chip_data_t*)data;
  float analog = 5 * (chip_data->value / 4096.0);
  pin_dac_write(chip_data->pin, analog);
}
```

```
void chip_init() {
  chip_data_t *chip_data = (chip_data_t*)malloc(sizeof(chip_data_t));
  chip_data->value = attr_init("moisture", 2910.0); // Replace with relevant attribute
  chip_data->pin = pin_init("A0", ANALOG);

  const timer_config_t config = {
    .callback = chip_timer_callback,
    .user_data = chip_data,
  };

  timer_t timer_id = timer_init(&config);
  timer_start(timer_id, 1000, true);
}
```

Custom Chip – JSON Definition

This file tells Wokwi how to render and place the custom component in the simulator.

```
% Soil_Sensor.chip.json
{
  "name": "Soil Sensor",
  "author": "MAHIB",
  "pins": [
    "GND",
    "VCC",
    "A0",
    "",
    "",
    "",
    "",
    ""
  ],
  "controls": [
    {
      "id": "moisture",
      "label": "Soil Moisture",
      "type": "range",
      "min": 1680,
      "max": 3620,
      "step": 1
    }
  ]
}
```