International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Weather Adaptive Light Intensity Controller

Niranjana R, Nithin Kumar K P, Paavana O M, Parvathi
Pavan K Y , Pooja Sunilkumara Sindhe , Priyanka N , Thanushree R, Mohammed Zubair
Students, Dept of Electronics and Communication Engineering ,
DRR Government Polytechnic, Davanagere , Karnataka.
Manju Haller R
Lecturer of Dept. of Electronics and Communication Engineering,
DRR Government Polytechnic, Davanagere , Karnataka.

…. ……***……

Abstract:

In many regions, outdoor lighting systems operate at fixed intensities, regardless of the current weather conditions, resulting in energy wastage and reduced lighting efficiency. During foggy, cloudy, or rainy weather, the fixed lighting may not provide adequate illumination, compromising visibility and safety. Conversely, during clear weather or dawn/dusk periods, the lighting may be unnecessarily bright, leading to excessive energy consumption.

There is a need for an intelligent, weather-adaptive light intensity controller that dynamically adjusts the brightness of outdoor lighting systems based on real-time weather and ambient light conditions. This solution should aim to enhance visibility and safety while minimizing energy usage

Keywords:

1. Arduino IDE program software.
2. Development board Arduino Uno.
3. Sensors.
4. Real time Data Moitoring

I. INTRODUCTION

The weather adaptive light intensity controller is an innovative system designed to automatically adjust the brightness of a light source based on environmental conditions such as temperature, humidity, and rainfall. Using an Arduino Uno board, the system integrates a DHT11 sensor to monitor temperature and humidity levels, alongside a rain sensor to detect rainfall. This data is processed by the Arduino to modify the intensity of the light accordingly.

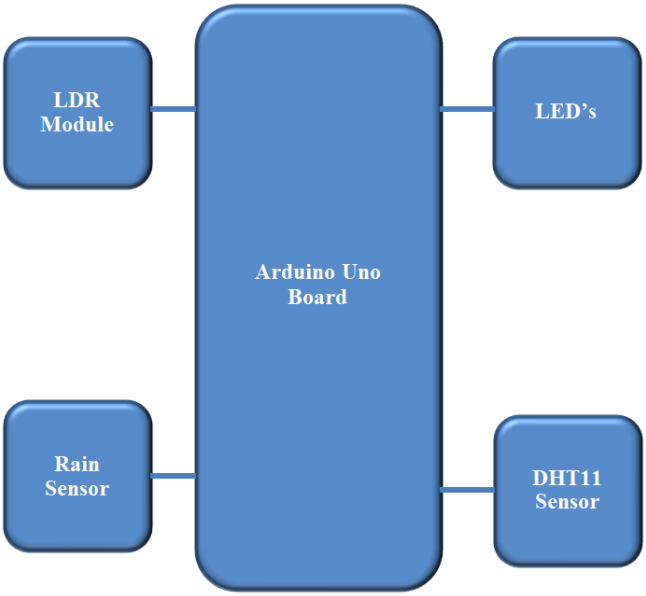
In rainy conditions, the light intensity is reduced, while in dry or sunny weather, it is increased. The system is further enhanced with the ESP8266 WiFi module, enabling remote control and monitoring via a mobile app or web interface.

METHODOLOGY

- Connect the DHT11 sensor, rain sensor, LDR module, and LEDs to the Arduino Uno using appropriate pins
- Continuously read data from DHT11, rain sensor, and LDR module.
- Use conditional statements to determine LED brightness based on sensor inputs.
- Implement PWM to adjust LED brightness dynamically according to environmental changes.

COMPONENTS OF WEATHER ADAPTIVE LIGHT INTENSITY CONTROLLER.

BLOCK DIAGRAM:

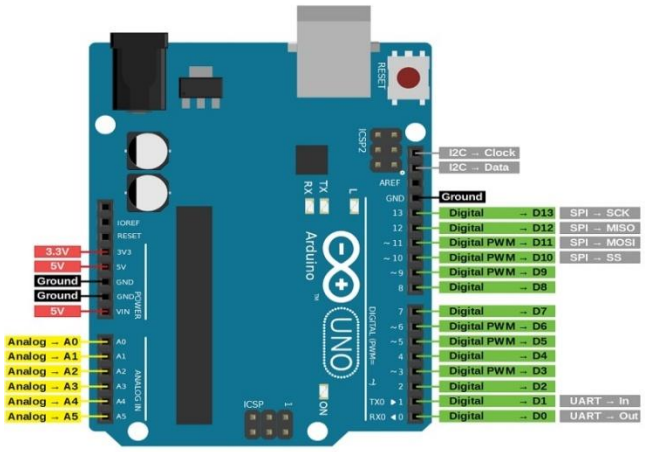


The weather adaptive light intensity controller functions by continuously monitoring real-time environmental conditions using DHT11 (temperature & humidity) and a rain sensor. The system is powered by an Arduino Uno, which processes sensor data and adjusts the bulb's brightness accordingly. The ESP8266 WiFi module enables wireless monitoring and control via an IoT platform.

The DHT11 sensor measures temperature and humidity, while the rain sensor detects rainfall intensity. These readings are continuously sent to the Arduino Uno.

The Arduino analyzes the sensor data and determines the appropriate brightness level based on predefined conditions: If rain or high humidity is detected, brightness increases for better visibility. If the weather is clear and dry, brightness decreases to save energy.

1. **Development Arduino Uno Board:**



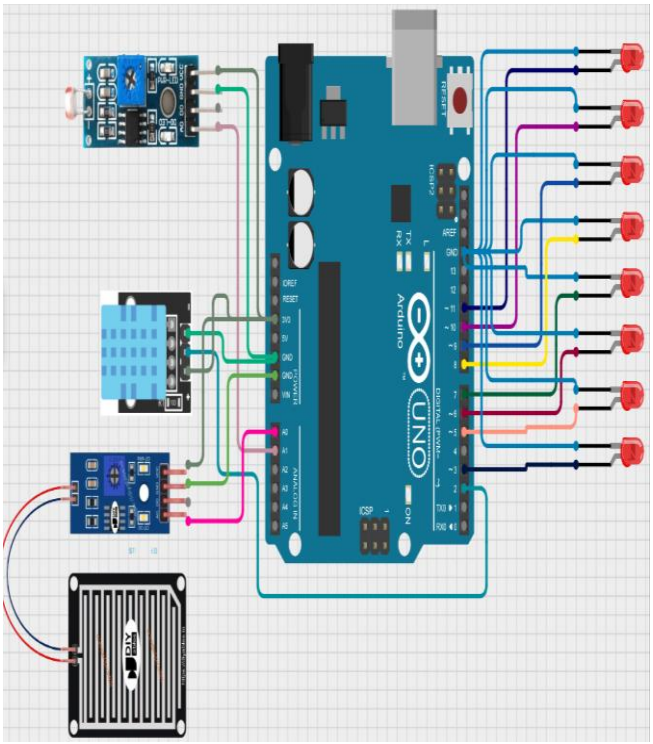
Sensors Input:

- DHT11 Sensor: Measures temperature and humidity.
- Rain Sensor: Detects rain or moisture.
- LDR (Light-Dependent Resistor): Measures ambient light intensity.

2. Arduino Processing:

- Reads data from the sensors.
- Uses conditions to decide LED brightness or on/off state based on weather.
- Example: If it's dark, turn on LEDs; if it's raining, reduce brightness.

Circuit Diagram:



Program:

```
#include <DHT.h>

// Pin Definitions for 8 Bulbs/LEDs
#define BULB_1 3   // PWM Pin
#define BULB_2 5   // PWM Pin
#define BULB_3 6   // PWM Pin
#define BULB_4 9   // PWM Pin
#define BULB_5 10  // PWM Pin
#define BULB_6 11  // PWM Pin
#define BULB_7 7   // Digital Pin (ON/OFF)
#define BULB_8 8   // Digital Pin (ON/OFF)

#define RAIN_SENSOR_PIN A0
#define DHT_PIN 2
#define LDR_PIN A1
#define DHT_TYPE DHT11

DHT dht(DHT_PIN, DHT_TYPE);
void setup() {
  // Set pins as output
  pinMode(BULB_1, OUTPUT); pinMode(BULB_2, OUTPUT);
  pinMode(BULB_3, OUTPUT); pinMode(BULB_4, OUTPUT);
  pinMode(BULB_5, OUTPUT); pinMode(BULB_6, OUTPUT);
  pinMode(BULB_7, OUTPUT); pinMode(BULB_8,
```

```
OUTPUT);

  pinMode(RAIN_SENSOR_PIN, INPUT);
  pinMode(LDR_PIN,INPUT);

  Serial.begin(9600);
  dht.begin();
}

void loop() {
  int rainValue =
analogRead(RAIN_SENSOR_PIN);
  int ldrValue = analogRead(LDR_PIN);
  float temp = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(temp) || isnan(humidity)) {
    Serial.println("DHT Sensor Fault!");
    turnOffAllBulbs();
    return;
  }
  int brightness = 0;
  if (rainValue < 500) {
    brightness = map(ldrValue, 0, 1023, 128, 255);
  } else {
    brightness = map(ldrValue, 0, 1023, 0, 255);
  }

  if (temp > 30) {
    brightness = brightness / 2;
  }

  controlBulbs(brightness);

  Serial.print("Rain: "); Serial.println(rainValue);
  Serial.print("LDR: "); Serial.println(ldrValue);
  Serial.print("Temp: "); Serial.println(temp);
  Serial.print("Humidity: "); Serial.println(humidity);
  Serial.print("Brightness:"); Serial.println(brightness);

  delay(1000);
}

void controlBulbs(int brightness) {
  analogWrite(BULB_1, brightness);
  analogWrite(BULB_2, brightness);
  analogWrite(BULB_3, brightness);
  analogWrite(BULB_4, brightness);
  analogWrite(BULB_5, brightness);
  analogWrite(BULB_6, brightness);
  if (brightness > 120) {
    digitalWrite(BULB_7, HIGH);
    digitalWrite(BULB_8, HIGH);
  } else {
    digitalWrite(BULB_7, LOW);
    digitalWrite(BULB_8, LOW);
  }
}

void turnOffAllBulbs() {
  analogWrite(BULB_1, 0); analogWrite(BULB_2, 0);
  analogWrite(BULB_3,0); analogWrite(BULB_4, 0);
  analogWrite(BULB_5,0); analogWrite(BULB_6, 0);
  digitalWrite(BULB_7, LOW);
  digitalWrite(BULB_8, LOW);
}
```

Advantages:

- **Efficient Water Usage:** The system optimizes irrigation based on real-time soil moisture levels, reducing water wastage.
- **Automated Monitoring:**Sensors continuously track environmental conditions, minimizing manual intervention.
- **Increased Crop Yield:** By providing precise water and nutrient levels, the system enhances plant health and productivity.
- **Remote Control & Monitoring:** IoT integration allows users to monitor and control the system from anywhere via the internet.
- **Energy Efficient:** Smart scheduling reduces unnecessary pump operation, conserving power.

Disadvantages:

- **Initial Setup Cost** – The installation of IoT-based sensors, controllers, and actuators can be expensive.
- **Internet Dependency** – Requires a stable internet connection for remote monitoring and control.
- **Maintenance Challenges** – Sensors and components require periodic maintenance and calibration for accuracy.

Conclusion:

This project successfully implements a weather-adaptive light intensity control system using an Arduino Uno, ESP8266 Wi-Fi module, rain sensor, and DHT11 sensor. By automatically adjusting bulb brightness based on real-time weather conditions, the system enhances energy efficiency, visibility, and user comfort. The integration of IoT enables remote monitoring and control, making it suitable for smart lighting applications. This automation reduces manual intervention, optimizes power consumption, and can be further improved with additional sensors and predictive algorithms for broader scalability in smart cities and industrial environment

Reference:

- Sharma, A., & Mehra, R. (2018). "Smart Street Light Using Arduino." International Journal of Scientific & Engineering Research, 9(4), 341-345.
- Rana, A., & Patel, V. (2019). "Weather Adaptive Street Lighting System Using IoT." International Journal of Electronics and Communication Engineering,6(7),445-450.