Phase 3: Development part1

Real-Time Environmental Monitoring

ABSTRACT:

Purpose: The system provides real-time weather monitoring in parks to ensure visitor safety, enhance comfort, and aid park management.

Components: It comprises strategically placed sensor-equipped monitoring stations, data fusion, predictive analytics, and user-friendly interfaces.

Benefits: The system improves park operations, visitor satisfaction, and serves as an educational tool, making parks safer and more enjoyable.

AI BASED DATA SET:

Weather Data:

Meteorological Data: Collect historical and real-time weather data, including temperature, humidity, wind speed, wind direction, atmospheric pressure, and precipitation.

Solar Radiation Data: Gather data related to solar radiation, UV levels, and solar exposure.

Air Quality Data: Include measurements of pollutants such as PM2.5, PM10, CO2, NO2, and O3.

Rainfall Data: Data on the timing, intensity, and duration of rainfall or snowfall events.

Geospatial Data:

GIS Data: Geographic Information System (GIS) data, such as maps, land use data, and terrain information.

GPS Data: Collect data related to the location of monitoring stations or weather sensors

.

Sensor Data:

Weather Station Data: Data from weather stations that include temperature sensors, anemometers, barometers, and rain gauges.

Air Quality Sensor Data: Data from air quality monitoring sensors, which measure air pollutants and meteorological parameters.

Satellite Imagery: Utilize satellite images to gather data on cloud cover, surface temperature, and more.

Historical Data:

Past weather records, ideally spanning several years, to enable long-term trend analysis.

Crowdsourced Data:

Data collected from public sources, such as social media posts or citizen science initiatives.

User-Generated Data:

Data collected from mobile apps or devices used by the public for weather-related information sharing.

Annotations:

Annotate the data with labels such as weather conditions (e.g., clear, cloudy, rainy), air quality levels (e.g., good, moderate, unhealthy), and geospatial coordinates.

Images and Videos:

Images or videos capturing real-time weather conditions can be valuable for training AI models.

Natural Language Data:

Textual data from news articles, weather reports, or social media that describe weather events and conditions.

PYTHON SCRIPT:

```
import Adafruit_DHT
import requests
import time
# Sensor setup (DHT22)
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4  # GPIO pin on the Raspberry Pi

# Server URL for sending data (optional)
SERVER_URL = "https://yourserver.com/upload_data"
# Main loop
while True:
    try:
        # Read sensor data
        humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR,
DHT_PIN)
        if humidity is not None and temperature is not None:
            # Print data to the console
            print(f"Temperature: {temperature:.2f}°C, Humidity: {humidity:.2f}%")
            # Send data to a server (optional)
            if SERVER_URL:
                data = {"temperature": temperature, "humidity": humidity, "location":
"Park Name"}
                response = requests.post(SERVER_URL, json=data)
                if response.status_code == 200:
                    print("Data sent to the server successfully.")
                else:
                    print("Failed to send data to the server.")
        else:
            print("Failed to retrieve data from the sensor.")
        time.sleep(60)  # Wait for 60 seconds before reading data again
    except KeyboardInterrupt:
        print("Exiting the program.")
        break
```

ARDUINO BASED SENSOR CODE:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>
#include <Adafruit_DHT.h>
#include <LiquidCrystal.h>
#define DHTPIN 7
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);  // Initialize the LCD
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP085 sensor, check wiring!");
    while (1) {}
  }
}
void loop()
 {
  sensors_event_t event;
  bmp.getEvent(&event);
  float temperature;
  float pressure;
  float humidity = dht.readHumidity();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temperature: ");
  if (isnan(humidity))
 {
    lcd.print("Failed to read DHT");
  }
else
 {
    temperature = event.temperature;
    lcd.print(temperature, 1);
    lcd.print("C");
    lcd.setCursor(0, 1);
    lcd.print("Humidity: ");
    lcd.print(humidity);
    lcd.print("%");
```

```
}
delay(2000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Pressure: ");
lcd.print(event.pressure, 2);
lcd.print("hPa");
lcd.setCursor(0, 1);
lcd.print("Altitude: ");
lcd.print(bmp.readAltitude(1013.25));
lcd.print("m");
delay(2000);
```

IOT DEVICE:

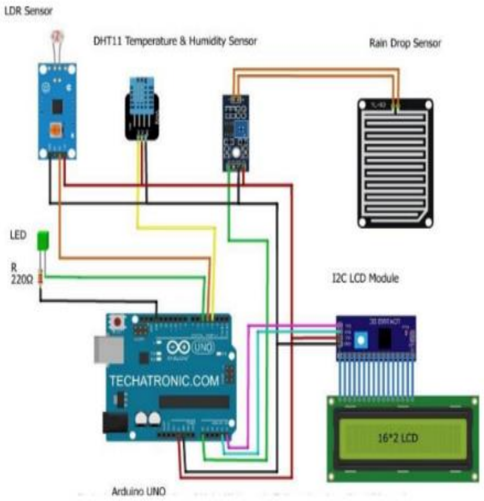
Arduino Specification:

- A0 – Analog output of the Sensor
- D0 – Digital output of the sensor
- GND – Ground
- VCC – 5V

Components Required:

- Arduino Uno
- USB Cable
- LCD Display screen
- Gas sensor – MQ-135
- Bread Board
- Jumper wires Pack

Circuit Diagram:


LDR Sensor
DHT11 Temperature & Humidity Sensor
Rain Drop Sensor
LED
R
220Ω
I2C LCD Module
16*2 LCD
TECHATRONIC.COM
UNO
Arduino UNO