ABSTRACT

This project explores the development of a weather monitoring system for CubeSat platforms, aimed at providing comprehensive environmental data from space. The system integrates a suite of sensors to measure various atmospheric parameters, including temperature, humidity, pressure, and radiation. By utilizing components such as the DHT11 for temperature and humidity, the BMP180 for pressure, and a radiation sensor, the CubeSat will gather and transmit real-time meteorological data.

The collected data is processed and transmitted back to Earth, offering valuable insights into the space environment and contributing to the understanding of atmospheric conditions in low Earth orbit. This system will enhance our ability to monitor and predict space weather phenomena, improve satellite operation and longevity, and provide crucial information for scientific research and mission planning. The integration of these sensors into the compact CubeSat form factor demonstrates the feasibility of deploying advanced weather monitoring technology in space.

PROJECT DEVELOPMENT

PROGRAM:

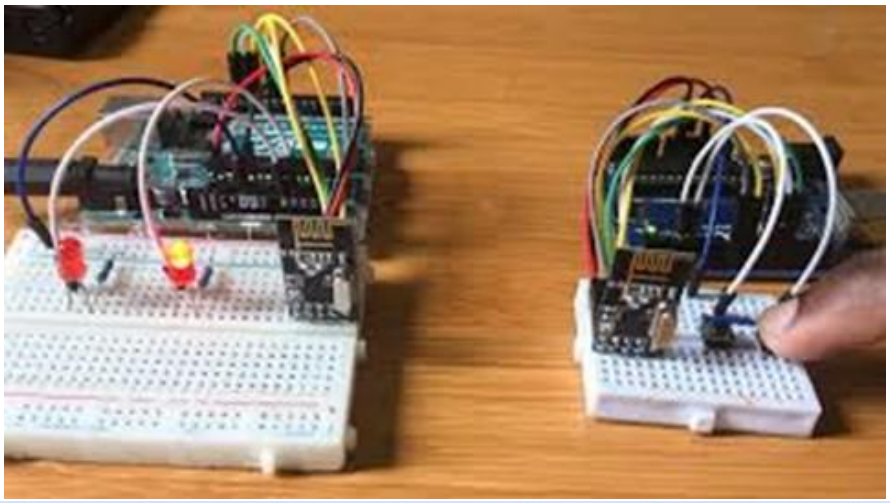
```
#include <dummy.h>
#include <MPU6050_tockn.h>
#include <Wire.h>
#include <SimpleDHT.h>
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
// Blynk settings
char auth[] = "krvSyTkdxJb8c0czxKFTUUS6UsF3IgD1"; // Replace with your Blynk Auth Token const char* templateId = "TMPL3f37Nmnma"; // Replace with your Blynk Template ID const char* templateName = "weather monitoring system"; // Replace with your Blynk Template Name
// WiFi credentials const char* ssid = "Kausik"; const char* password = "1234567890"; // Define pins for the ultrasonic sensor const int trigPin = 4;  //
```

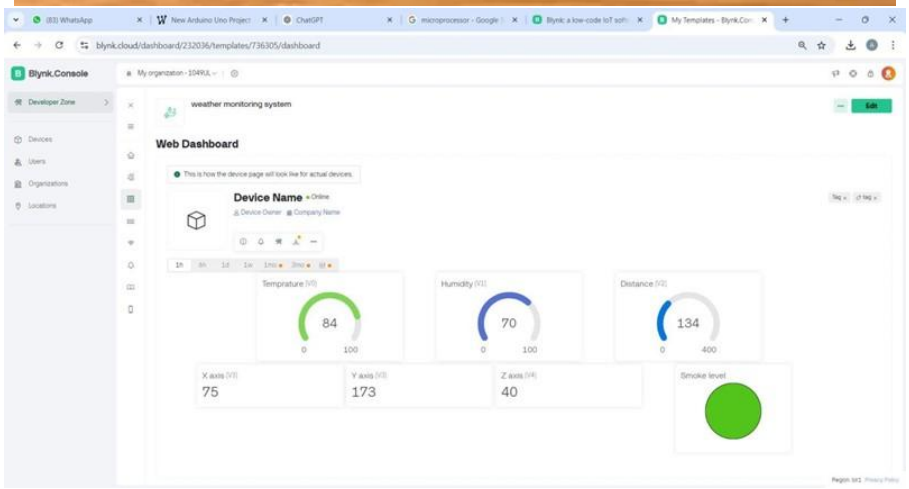
```
Trig pin connected to pin 4 const int echoPin = 15;  // Echo pin connected to pin
15
// Define variable for storing the duration of the pulse and the calculated
distance long duration; int distance;
int thresholdDistance = 10;  // Distance threshold in centimeters
// Initialize MPU6050 and DHT11 MPU6050 mpu6050(Wire); int pinDHT11 =
2;
SimpleDHT11 dht11(pinDHT11); // Gas Sensor and RGB LED pins const int
gasSensorPin = 5; int gasSensorValue = 0; const int redPin = 9; const int
greenPin = 10; const int bluePin = 11; // PWM channels const int redChannel =
0; const int greenChannel = 1; const int blueChannel = 2;
const int pwmFreq = 5000; // Frequency of PWM const int pwmResolution = 8;
// Resolution of PWM (8-bit)
// Timer long timer = 0; // WiFi client WiFiClient client; void setup() {
Serial.begin(9600);  // Initialize serial communication at 9600 bps
Wire.begin();   // Initialize MPU6050   mpu6050.begin();
mpu6050.calcGyroOffsets(true);
// Set RGB LED pins as output and configure PWM   ledcSetup(redChannel,
pwmFreq, pwmResolution);   ledcSetup(greenChannel, pwmFreq,
pwmResolution);   ledcSetup(blueChannel, pwmFreq, pwmResolution);
ledcAttachPin(redPin, redChannel);   ledcAttachPin(greenPin, greenChannel);
ledcAttachPin(bluePin, blueChannel);
// Set the trigPin as an output and the echoPin as an input for the ultrasonic
sensor   pinMode(trigPin, OUTPUT);   pinMode(echoPin, INPUT);   // Connect
to WiFi   WiFi.begin(ssid, password);   while (WiFi.status() !=
WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nWiFi connected");
// Initialize Blynk
Blynk.begin(auth, ssid, password, templateId, templateName);
}
void loop() {
Blynk.run(); // Run Blynk   mpu6050.update();
// Print MPU6050 data every second   if (millis() - timer > 1000) {
Serial.println("==========================================
=========");
// MPU6050 Readings     float mpuTemp = mpu6050.getTemp();     float accX =
mpu6050.getAccX();     float accY = mpu6050.getAccY();     float accZ =
```

```
mpu6050.getAccZ();     float gyroX = mpu6050.getGyroX();     float gyroY =
mpu6050.getGyroY();     float gyroZ = mpu6050.getGyroZ();
Serial.print("MPU Temp : "); Serial.println(mpuTemp);
Serial.print("accX : "); Serial.print(accX);
Serial.print("\taccY : "); Serial.print(accY);
Serial.print("\taccZ : "); Serial.println(accZ);
Serial.print("gyroX : "); Serial.print(gyroX);
Serial.print("\tgyroY : "); Serial.print(gyroY);
Serial.print("\tgyroZ : "); Serial.println(gyroZ);
// DHT11 Readings
Serial.println("==========================================
=========");
  Serial.println("Sample DHT11...");         byte temperature = 0;
byte humidity = 0;     int err = SimpleDHTErrSuccess;
if ((err = dht11.read(&temperature, &humidity, NULL)) !=
SimpleDHTErrSuccess) {
Serial.print("Read DHT11 failed, err=");
Serial.print(SimpleDHTErrCode(err));
Serial.print(",");
Serial.println(SimpleDHTErrDuration(err));
} else {
Serial.print("DHT11 Temp : ");
Serial.print((int)temperature); Serial.print(" *C, ");
Serial.print((int)humidity); Serial.println(" %H");
}
// Gas Sensor Readings and RGB LED Control
gasSensorValue = analogRead(gasSensorPin);  // Read the value from the gas
sensor
Serial.print("Gas Sensor Value: ");
Serial.println(gasSensorValue);  // Print the sensor value to the serial monitor
// Check gas sensor value and set RGB LED color accordingly     if
(gasSensorValue > 350) {
Serial.println("Gas detected high!");
setColor(0, 255, 0);  // Set RGB LED to green (high gas concentration)
} else if (gasSensorValue > 300) {       Serial.println("Gas detected!");
setColor(0, 0, 255);  // Set RGB LED to blue (moderate gas concentration)
} else {
Serial.println("Gas concentration normal.");
setColor(255, 0, 0);  // Set RGB LED to red (normal)
}
```

```
// Ultrasonic Sensor Readings     digitalWrite(trigPin, LOW);
delayMicroseconds(2);     digitalWrite(trigPin, HIGH);
delayMicroseconds(10);     digitalWrite(trigPin, LOW);     duration =
pulseIn(echoPin, HIGH);     distance = duration * 0.0343 / 2;
Serial.print("Ultrasonic Distance: ");     Serial.print(distance);
Serial.println(" cm");
Serial.println("==========================================
=========\n");
// Send data to Blynk
Blynk.virtualWrite(V0, temperature);
Blynk.virtualWrite(V1, humidity);     Blynk.virtualWrite(V2, distance);
timer = millis();  // Reset the timer
}
delay(100);  // Small delay for sensor updates
}
// Function to set RGB LED color
void setColor(int redValue, int greenValue, int blueValue) {
ledcWrite(redChannel, redValue);   ledcWrite(greenChannel, greenValue);
ledcWrite(blueChannel, blueValue);
}
```

OUTPUT:




Blynk.Console
Developer Zone
Devices
Users
Organizations
Locations
weather monitoring system
Edit
Web Dashboard
This is how the device page will look like for actual devices.
Device Name
Online
Device Owner
Company Name
Temprature
84
Humidity
70
Distance
134
X axis
75
Y axis
173
Z axis
40
Smoke level