Combined Air Quality, Temperature, and Humidity Sensor

By: Rhushya KC, Rithvik Muthyalapati, Rohan S

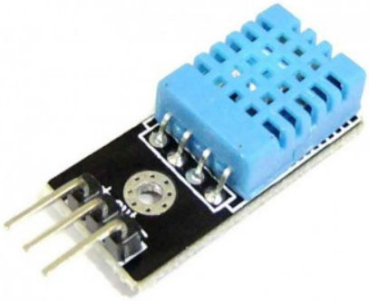
Overview

For our project, we have designed and created an arduino circuit that when placed in an environment, will display important parameters such as the quality of the air, the temperature, and the humidity. We utilized the DHT11 component which consists of the temperature and humidity sensor, the MQ 135 component which consists of the air quality sensor, the I2C OLED display to display environmental readings, the LDR sensor module which detects day or night based on the brightness of the environment, and the Arduino Nano V3 to connect the sensors and display together. The programming would be done with Arduino IDE. Our product is advantageous in regards to ease of operation of our device to detect whether an environment is safe. By using Arduino IDE we aim to offer user-friendly solutions to the users. The significance of our product lies in its simplicity and ease of operation, allowing users to swiftly ascertain the safety of their surroundings. Through our collaborative efforts, we seek to contribute to the advancement of accessible and affordable environmental monitoring solutions. By fostering innovation and leveraging open-source platforms like Arduino, we aspire to create a positive impact on society by promoting awareness and facilitating informed decision-making regarding environmental well-being.

Components

DHT11 Temperature and Humidity Sensor

1. **Type**: Combined temperature and humidity sensor
2. **Measurement**: Measures temperature and humidity
3. **Output**: Provides digital output (temperature and humidity values)
4. **Accuracy**:
   - Temperature: ±2°C
   - Humidity: ±5% RH
5. **Range**:
   - Temperature: 0°C to 50°C
   - Humidity: 20% to 90% RH
6. **Operating Voltage**: Typically operates at 3.3V or 5V
7. **Application**:
   - Commonly used in home automation systems for climate control
   - Widely employed in weather stations
   - Utilized in HVAC systems for monitoring temperature and humidity



MQ135 Air Quality Sensor

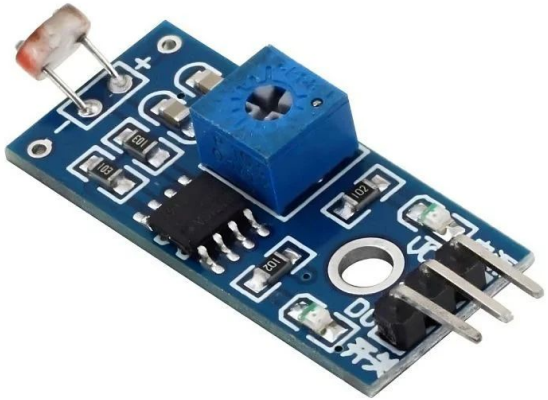
1. **Type**: Air quality sensor
2. **Measurement**: Measures multiple gases such as NH3, NOx, alcohol, CO2, etc.
3. **Output**: Provides analog output, which varies based on the detected gases
4. **Accuracy**: Accuracy varies depending on the gas being detected
5. **Range**: Varies based on the gas being detected
6. **Operating Voltage**: Typically operates at 5V
7. **Application**:
    - Used in air quality monitoring systems.
    - Employed in indoor air quality monitoring in buildings and homes.
    - Utilized in environmental monitoring systems.

0.96” LCD OLED Display

1. **Type**: OLED (Organic Light Emitting Diode) display
2. **Size**: 0.96 inches
3. **Communication Protocol**: I2C (Inter-Integrated Circuit)
4. **Resolution**: 128 x 64 pixels
5. **Color**: Monochrome (usually white or blue)
6. **Controller Chip**: SSD1306
7. **Viewing Angle**: Wide viewing angle (>160 degrees)
8. **Brightness**: Bright and clear display
9. **Operating Voltage**: Typically operates at 3.3V or 5V
10. **Interface**: 4-wire SPI (Serial Peripheral Interface) or I2C
11. **Driver Library**: Available for Arduino, Raspberry Pi, and other microcontrollers
12. **Application**:
    - Used for displaying information in various projects, such as IoT devices, wearables, and small gadgets
    - Suitable for displaying sensor data, text, and simple graphics

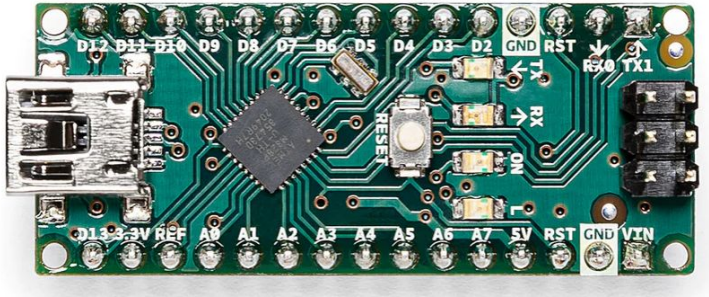
LDR Sensor

1. **Type**: Light-dependent resistor (LDR)
2. **Measurement**: Measures ambient light intensity
3. **Output**: Analog voltage output inversely proportional to light intensity
4. **Accuracy**: Depends on the sensor quality and calibration
5. **Range**: Typically 10k ohms to a few hundred ohms
6. **Operating Voltage**: Generally operates at 5V
7. **Application**:
    - Light sensing applications in automated lighting systems
    - Sunlight intensity monitoring for solar panel optimization
    - Darkness detection for security systems

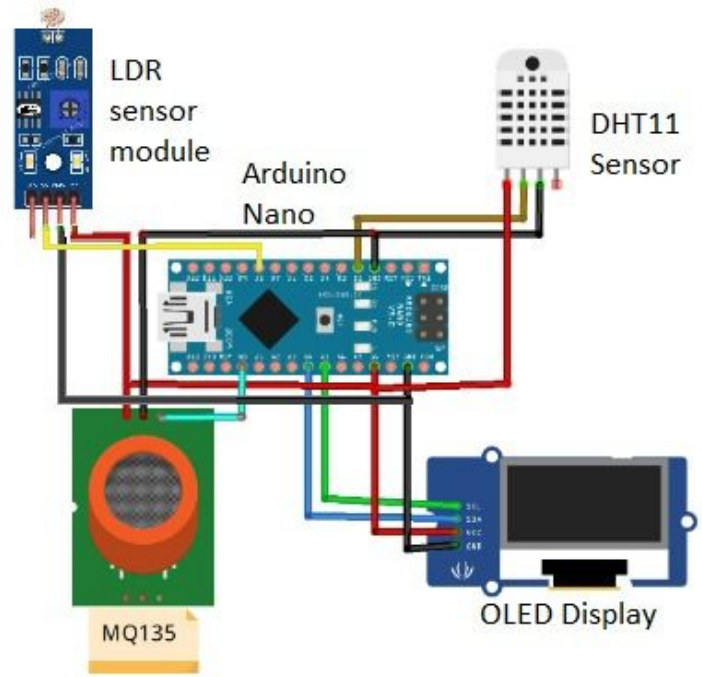


Arduino Nano V3

1. **Type**: Microcontroller board.
2. **Microcontroller**: Atmel ATmega328P
3. **Clock Speed**: 16 MHz
4. **Digital I/O Pins**: 14
5. **Analog Input Pins**: 8
6. **Operating Voltage**: 5V
7. **Input Voltage (recommended)**: 7-12V
8. **Input Voltage (limits)**: 6-20V
9. **Flash Memory**: 32 KB (of which 2 KB is used by bootloader)
10. **SRAM**: 2 KB
11. **EEPROM**: 1 KB
12. **Communication**: USB, UART, SPI, I2C
13. **Dimensions**: 18 x 45 mm
14. **Application**:
    - Used for prototyping, projects with space constraints, and applications where lower power consumption is preferred
    - Commonly used in robotics, automation, and IoT projects



Circuit Diagram



Code

```
#include <DHT.h>
#include <DHT_U.h>

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include  <Adafruit_SSD1306.h>
#include <Fonts/FreeSans9pt7b.h>
#include <Fonts/FreeMonoOblique9pt7b.h>
#include <DHT.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define  SCREEN_HEIGHT 64 // OLED display height, in pixels

#define OLED_RESET     4  // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH,  SCREEN_HEIGHT, &Wire, OLED_RESET);

#define sensor    A0
#define DHTPIN  2         // Digital pin 2
#define DHTTYPE DHT11     // DHT 11

#define DO_PIN 8

int gasLevel  = 0;        //int variable for gas level
String quality ="";
DHT dht(DHTPIN,  DHTTYPE);

const unsigned char epd_bitmap_moon [] PROGMEM = {
  0x00, 0x00, 0x07, 0xc0, 0x1f, 0xf0, 0x3f, 0xf8, 0x01, 0xf8, 0x00, 0xfc, 0x00, 0x7c, 0x00, 0x7c,
  0x00, 0x7c, 0x00, 0x7c, 0x00, 0xfc, 0x01, 0xf8, 0x3f, 0xf8, 0x1f, 0xf0, 0x07, 0xc0, 0x00, 0x00
};

const unsigned char epd_bitmap_sun [] PROGMEM = {
  0x01, 0x00, 0x41, 0x04, 0x21, 0x08, 0x13, 0x90, 0x0c, 0x60, 0x0b, 0xa0, 0x17, 0xd0, 0xf7, 0xde,
  0x17, 0xd0, 0x0b, 0xa0, 0x0c, 0x60, 0x13, 0x90, 0x21, 0x08, 0x41, 0x04, 0x01, 0x00, 0x00, 0x00
};
```

```
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
  Serial.println("Failed  to read from DHT sensor!");
    return;
  }
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setFont();
  display.setCursor(0, 43);
  display.println("Temp  :");
  display.setCursor(80, 43);
  display.println(t);
  display.setCursor(114, 43);
  display.println("C");
  display.setCursor(0,  56);
  display.println("RH    :");
  display.setCursor(80, 56);
  display.println(h);
  display.setCursor(114, 56);
  display.println("%");
}
```

```
void air_sensor()
{
  gasLevel = analogRead(sensor);

  if(gasLevel<181){
    quality = "  GOOD!";
  } else if (gasLevel >181 && gasLevel<225){
    quality =  "  Poor!";
  } else if (gasLevel >225 && gasLevel<300){
    quality  = "Very bad!";
  } else if (gasLevel >300 && gasLevel<350){
    quality  = "Polluted!";
  } else{
    quality = " Toxic";
}

  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(1,5);
  display.setFont();
  display.println("Air Quality:");
  display.setTextSize(1);
  display.setCursor(20,23);
  display.setFont(&FreeMonoOblique9pt7b);
  display.println(quality);
}
```

```
void lightsense(){
  int light_state = digitalRead(DO_PIN);

  if (light_state == HIGH){
    Serial.println("The light is NOT present");
    display.drawBitmap(105, 15, epd_bitmap_moon, 16, 16, WHITE);
  } else{
    Serial.println("The light is present");
    display.drawBitmap(105, 15, epd_bitmap_sun, 16, 16, WHITE);
  }
}

void setup() {
  Serial.begin(9600);
  pinMode(sensor,INPUT);
  pinMode(DO_PIN,INPUT);
  dht.begin();
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c)) { // Address  0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
}
```

```
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(2);
  display.setCursor(50, 0);
  display.println("Air");
  display.setTextSize(1);
  display.setCursor(23, 20);
  display.println("Quality monitor");
  display.display();
  delay(1200);
  display.clearDisplay();

  display.setTextSize(1);
  display.setCursor(50, 0);
  display.println("By:");

  display.setCursor(10, 15);
  display.println("Rhushya K C");

  display.setCursor(10, 30);
  display.println("Rithvik M");

  display.setCursor(10, 45);
  display.println("Rohan S");
  display.display();
  delay(1000);
  display.clearDisplay();
}
```

```
void loop() {
display.clearDisplay();
air_sensor();
sendSensor();
lightsense();
display.display();
}
```