Arduino temperature humidity code

Write an Arduino code to measure and display temperature and humidity using DHT11 module.

Here’s a simple and well-commented **Arduino code** to measure and display **temperature** and **humidity** using a **DHT11 sensor** and the **DHT library**:

🧠 Requirements

Before uploading the code:

1. Install the **DHT sensor library** by Adafruit:
   - Go to **Sketch → Include Library → Manage Libraries...**
   - Search for **“DHT sensor library”** by *Adafruit* and install it.
   - Also install **Adafruit Unified Sensor** if prompted.
2. Connect your **DHT11 module** properly:
   - **VCC → 5V**
   - **GND → GND**
   - **DATA → Digital Pin 2** (you can change in code)

✅ Arduino Code

```cpp
#include "DHT.h"

// Define DHT pin and type
#define DHTPIN 2       // Digital pin connected to DHT11
#define DHTTYPE DHT11   // DHT 11 Sensor

// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHT11 Temperature and Humidity Sensor");

  dht.begin();  // Start the DHT sensor
}

void loop() {
  // Wait a few seconds between measurements
  delay(2000);

  // Read humidity (in %)
  float humidity = dht.readHumidity();
  // Read temperature as Celsius
  float temperature = dht.readTemperature();
  // Read temperature as Fahrenheit
  float temperatureF = dht.readTemperature(true);

  // Check if readings failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Print the results to Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\t");

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C  |  ");
  Serial.print(temperatureF);
  Serial.println(" °F");
}
```

📟 Output Example

When you open the **Serial Monitor (9600 baud)**, you'll see something like:

```yaml
DHT11 Temperature and Humidity Sensor
Humidity: 56.00 %	Temperature: 28.00 °C  |  82.40 °F
Humidity: 57.00 %	Temperature: 28.10 °C  |  82.58 °F
```

Would you like me to modify this code so it displays the readings on an **LCD (16x2)** instead of Serial Monitor?

How can I display the data on 0.96 inch OLED display module?

Excellent question 👍

To display **temperature and humidity** from a **DHT11** sensor on a **0.96-inch OLED display (128×64, SSD1306 driver)**, you can use the **Adafruit SSD1306** and **Adafruit GFX** libraries.

🧠 What You’ll Need

🧩 Components:

- Arduino UNO (or compatible board)
- DHT11 temperature & humidity sensor
- 0.96" OLED display (I²C type — usually 4 pins: VCC, GND, SDA, SCL)
- Jumper wires

⚙️ Connections

OLED Pin	Connect To
VCC	5V (or 3.3V, depending on display)
GND	GND
SDA	A4 (on UNO)
SCL	A5 (on UNO)

DHT11 Pin	Connect To
VCC	5V
GND	GND

DHT11 Pin	Connect To
DATA	D2

📦 Library Installation

Install the following libraries from the Arduino Library Manager:

1. **Adafruit SSD1306**
2. **Adafruit GFX Library**
3. **DHT sensor library by Adafruit**
4. **Adafruit Unified Sensor**

To install:

`Sketch → Include Library → Manage Libraries → Search and Install`

✅ Arduino Code

```cpp
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"

// OLED display dimensions
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Create display object (I2C address 0x3C is common)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// DHT11 setup
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  // Initialize OLED display
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Check address (0x3C or 0x3D)
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setCursor(10, 10);
  display.println("DHT11 Sensor Ready");
  display.display();
  delay(2000);
}

void loop() {
  // Wait 2 seconds between readings
  delay(2000);

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature(); // Celsius

  // Check if any reads failed
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    display.clearDisplay();
    display.setCursor(10, 25);
    display.setTextSize(1);
    display.println("Sensor Error!");
    display.display();
    return;
  }

  // Print to Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %  |  ");
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" *C");
  // Display on OLED
  display.clearDisplay();

  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("DHT11 SENSOR");

  display.setTextSize(2);
  display.setCursor(0, 20);
  display.print(temperature);
  display.print((char)247); // Degree symbol
  display.println("C");

  display.setTextSize(2);
  display.setCursor(0, 45);
  display.print(humidity);
  display.println("%");

  display.display();
}
```

🖥️ Output on OLED Display

Example display layout:

```shell
DHT11 SENSOR
 27°C
 56%
```

Would you like me to modify the display layout to show **Celsius, Fahrenheit, and Humidity** together neatly (like a mini weather screen)?

What if I use Arduino NANO

Excellent question 👏 — using an **Arduino Nano** instead of an **UNO** is **perfectly fine**!
The code remains **exactly the same**, but there are **two small wiring differences** because the **I^{2}C pins** are located differently on the Nano.

Let’s go over the details 👇

⚙️ Wiring for Arduino Nano + DHT11 + 0.96" OLED

🧩 Connections

🔹 DHT11 Sensor

DHT11 Pin	Connect To
VCC	5V
GND	GND
DATA	D2

🔹 0.96" OLED Display (I^{2}C Type)

OLED Pin	Connect To (Nano)
VCC	5V (or 3.3V if your OLED supports it)
GND	GND
SDA	**A4**
SCL	**A5**

> 💡 On the **Nano**, I^{2}C (Wire library) uses:
> - “SDA → A4”
> - “SCL → A5
>   These are the same as on the UNO, but physically placed differently on the board.”

🔌 Power Notes

- The Nano’s **5V pin** can power both the DHT11 and the OLED.

- If you use multiple sensors, consider powering the OLED with **3.3V** (if supported) to reduce load.

💻 Code

✅ The same code I shared above for the UNO works **without any change**.

Here's the key part that defines the pins:

cpp

```cpp
#define DHTPIN 2      // DHT11 data pin
#define DHTTYPE DHT11
```

If you connect the DHT11 data line to another digital pin (say D3), just update that line.

Everything else remains identical — including the libraries and display address (`0x3C`).

🧠 Tip for Nano Users

If your OLED doesn't display anything:

1. Double-check the I2C address using an **I2C Scanner** (you can upload a simple scanner sketch to find it — often it's `0x3C` or `0x3D`).
2. Ensure proper soldering on the OLED's I2C pins.
3. Use shorter wires for reliable signal.

Would you like me to include a **clean "weather-style" OLED layout** showing temperature in both °C and °F plus humidity (like a mini weather screen)?

Yes

Perfect 🌤️ — here's an upgraded **Arduino Nano + DHT11 + 0.96" OLED** code that displays **Temperature (°C + °F)** and **Humidity (%)** in a clean **weather-style layout**.

It looks great on the **128×64 OLED** (SSD1306 driver).

🧩 Connections

Component	Pin	Connect To (Arduino Nano)
DHT11	VCC	5V
DHT11	GND	GND
DHT11	DATA	D2
OLED	VCC	5V (or 3.3V if supported)
OLED	GND	GND
OLED	SDA	A4
OLED	SCL	A5

📚 Required Libraries

Install these via **Sketch → Include Library → Manage Libraries...**

1. **Adafruit SSD1306**
2. **Adafruit GFX Library**
3. **DHT sensor library** by Adafruit
4. **Adafruit Unified Sensor**

✅ Arduino Code

cpp

```cpp
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHT.h"
```

```
// OLED display configuration
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// DHT11 sensor configuration
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();

  // Initialize OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);
  display.setTextSize(1);
  display.setCursor(20, 25);
  display.println("Weather Station");
  display.display();
  delay(2000);
}

void loop() {
  delay(2000); // DHT11 reads slowly

  float humidity = dht.readHumidity();
  float tempC = dht.readTemperature();
  float tempF = dht.readTemperature(true);

  if (isnan(humidity) || isnan(tempC)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    display.clearDisplay();
    display.setCursor(15, 25);
    display.setTextSize(1);
    display.println("Sensor Error!");
    display.display();
    return;
  }

  // Print to Serial
  Serial.print("Temp: ");
  Serial.print(tempC);
  Serial.print(" °C  ");
  Serial.print(tempF);
  Serial.print(" °F  ");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  // Display on OLED
  display.clearDisplay();

  // Header
  display.setTextSize(1);
  display.setCursor(30, 0);
  display.println("Weather Monitor");

  // Temperature in Celsius
  display.setTextSize(2);
  display.setCursor(0, 18);
  display.print(tempC, 1);
  display.print((char)247); // Degree symbol
  display.print("C");

  // Temperature in Fahrenheit
  display.setTextSize(1);
  display.setCursor(0, 40);
  display.print(tempF, 1);
  display.print((char)247);
  display.println("F");

  // Humidity
  display.setTextSize(2);
  display.setCursor(80, 25);
  display.print(humidity, 0);
  display.println("%");
  // Optional labels
  display.setTextSize(1);
  display.setCursor(85, 48);
  display.println("Humidity");

  display.display();
}
```

🖥️ Display Layout Example

```mathematica
    Weather Monitor
26.4°C      58%
79.5°F     Humidity
```

🌈 Optional Enhancements

You can:

- Add a **° symbol** more clearly using a custom bitmap icon
- Include a **temperature trend icon** (like ☀️ / 🌧️)
- Use a **bigger font** for temperature emphasis

Would you like me to enhance this version with **animated weather icons** (sun, cloud, rain) that change based on humidity and temperature values?

I am not understanding this syntex - DHT dht(DHTPIN, DHTTYPE);

Excellent question — and a very important one 👏

Let's break down the syntax:

```cpp
```

```
DHT dht(DHTPIN, DHTTYPE);
```

This line is part of **object-oriented programming (OOP)** in C++ — and it's how we **create an object** from the `DHT` **class** provided by the DHT library.

🧩 Let's understand it step by step

1️⃣ `DHT`
This is the **class name** — defined inside the DHT library.
It contains all the **functions and variables** that know how to talk to your **DHT sensor** (like `readTemperature()` and `readHumidity()`).

Think of it like a **blueprint** for handling a DHT sensor.

2️⃣ `dht`
This is the **object name** you're creating — you can name it anything you want, for example:

```cpp
DHT mySensor(DHTPIN, DHTTYPE);
```

or

```cpp
DHT dhtSensor(DHTPIN, DHTTYPE);
```

This object will represent **your actual sensor** in the code.
Later, you'll use it to **call methods**, like:

```cpp
dht.begin();
float temp = dht.readTemperature();
float humidity = dht.readHumidity();
```

3️⃣ `(DHTPIN, DHTTYPE)`

These are **parameters** passed into the constructor — i.e., when you "build" the object.

- `DHTPIN` → tells which **digital pin** your DHT sensor's **data line** is connected to (example: pin 2).
- `DHTTYPE` → tells what **type of DHT sensor** you are using (DHT11, DHT22, DHT21, etc.)

So this line:

```cpp
DHT dht(DHTPIN, DHTTYPE);
```

means:

> "Create a DHT sensor object named `dht` that uses the data pin defined by `DHTPIN` and is of type `DHTTYPE` (for example, DHT11)."

🧠 Analogy

Imagine you have a **remote control class** that can operate any TV:

```cpp
Remote remote(tvPin, tvBrand);
```

This means:
"You're creating a remote that controls the TV connected to this pin and of this brand."

Same with:

```cpp
DHT dht(DHTPIN, DHTTYPE);
```

You're telling Arduino:
"This is my DHT sensor object, connected to pin 2, and it's a DHT11 type sensor."

Would you like me to show you **how the DHT class internally works** (like what’s inside it and how it uses those parameters)? It’ll help you understand this concept deeply for all Arduino libraries.