

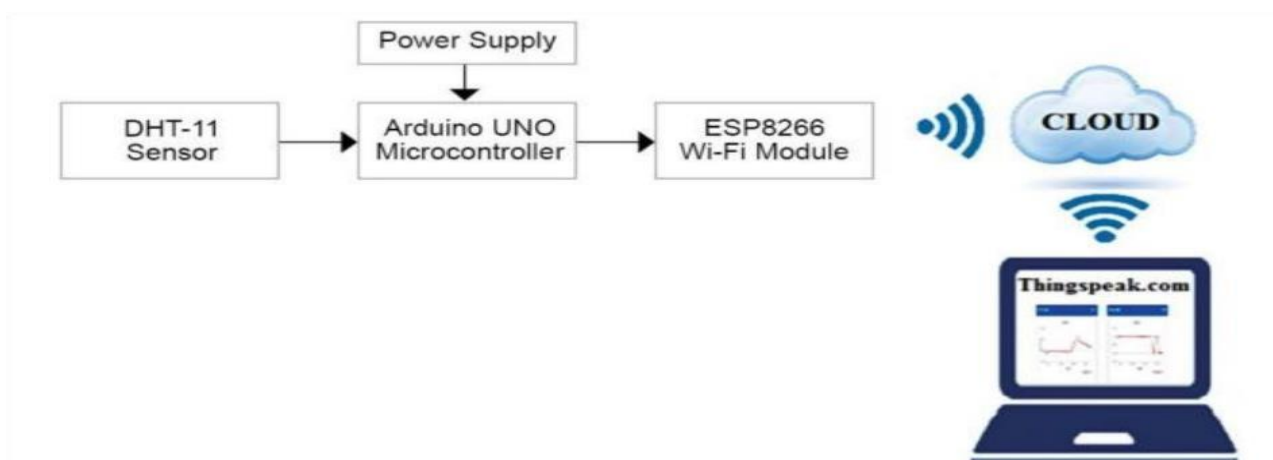
IOT BASED TEMPARATURE AND HUMIDITY MONITERING SYSTEM

INTRODUCTION

In this Arduino Project we will learn how to use the DHT11 or the DHT22 sensor for measuring temperature and humidity with the Arduino board. Traditional analog humidity sensor need signal circuit design, adjustment and calibration, for which, the precision can not be guaranteed, so does the linearity, repetition, interchanging and consistency. DHT11 and DHT22 is a new type temperature/humidity sensor made by Sensirion Company based on CMOSens technology, which combines CMOS chip with sensor. Temperature /humidity sensor, signal magnifier, A/D switch and 12C bus interface are intergared in one chip. The author designs a temperature/ humidity measurement and sends system with SHT1x sensor for temperature/humidity measurement in close circumstances. Live temperature or humidity value is sent to a forane receiver through wireless signal.

BLOCK DIAGRAM AND WORKING

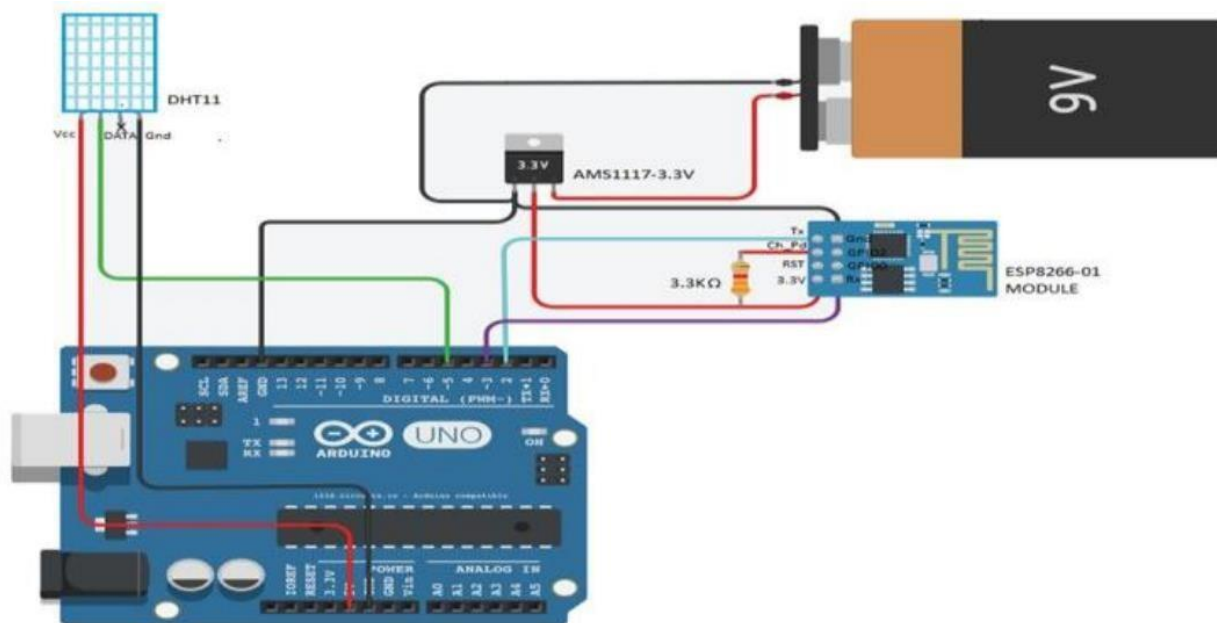
The DHT11 sensor senses humidity and temperature, and sends the information to digital pin 5 of Arduino MCU,. From Arduino MCU, humidity and temperature values are uploaded to the Cloud at regular intervals of time through ESP8266 Wi Fi module. From the Cloud, humidity and temperature values can be seen graphically on ThingSpeak platform from anywhere in the world. Once sketch uploading is done, it will upload humidity and temperature values on ThingSpeak platform and you will be able to see it graphically in Private View window. If you want to change channel or field name, you can change it from Channel Settings.



PROPOSED AIR POLLUTION MONITORING SYSTEM DESIGN

This project having four sections, firstly Humidity and Temperature Sensor DHT11 senses the Humidity and Temperature Data. Secondly Arduino Uno extracts the DHT11 sensor's data as suitable number in percentage and Celsius scale, and sends it to Wi-Fi Module. Thirdly Wi-Fi Module ESP8266 sends the data to ThingSpeak's Sever. And analyses the data and shows it in a Graph form. Optional LCD is also used to display the Temperature and Humidity.

Interface: The Temperature and Humidity Sensor Project will be controlled using an application named Blynk (Available for Android or IOS) using Arduino, an Ethernet Shield and its libraries. User can securely login over Blynk to control and monitor the room temperature and humidity. The code involves: The Arduino sketch. The Arduino sketch handles the communications by setting up the network. The sketch runs the program and communicates one line at a time over the server. Users can login remotely on this web server.

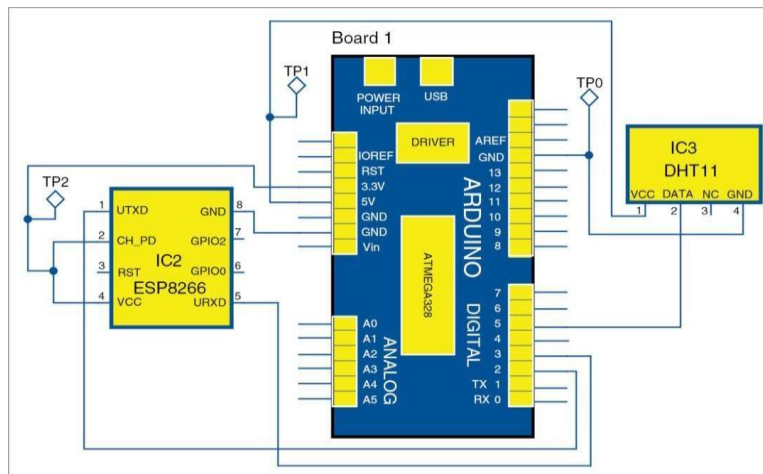


Use of DHT22 Sensor: DHT22 capacitive humidity sensing digital temperature and humidity module is one that contains the compound that has been calibrated digitally to signal output of the temperature and humidity sensors. The sensor includes a capacitive sensor, wet components and a high precision temperature measurement devices, and connected with a high-performance 8-bit microcontroller. The product has excellent quality, fast response, strong anti-jamming capability, and high cost. It is definitely long-lasting and has great endurance. ThingSpeak is an open source platform to store and retrieve a data for Internet of Things application. To use this, you need to register in ThingSpeak cloud and then login to your account. After create a new channel with temperature in one field and humidity in another field as shown in Fig: 1.2. Once you created a new channel, it will generate a two API keys, they are READ API keys and WRITE API keys. First, copy the WRITE API keys from ThingsSpeak and paste it into the line the program. Next, replace the Host_Name and Password with your WiFi name and WiFi password in the two lines given below in the program.

(String Host_Name = "Pantech" and String Password = "pantech123") The Arduino program Uses DHT library, if it is not presented in your arduino IDE, select SketchàInclude libraryàManage librariesàInstall DHT Sensor library. Then compile the program and upload to a Arduino Uno through Arduino IDE. Ensure that WiFi modem and internet connection in your Smartphone or PC are working properly. After uploaded a program, the Temperature and Humidity data is uploaded on ThingSpeak platform. You can see it graphically in the private view window of your channel as shown And you can able to see the uploaded data from serial port of Arduino IDE.

CIRCUIT DIAGRAM AND EXPLANATION

Circuit diagram for monitoring humidity and temperature is shown. It is built around Arduino MCU, DHT11 sensor and ESP8266 Wi-Fi module. In this project, we will build a small circuit to interface Arduino with DHT11 Temperature and Humidity Sensor. One of the main applications of connecting DHT11 sensor with Arduino is weather monitoring. All the DHT11 Sensors are accurately calibrated in the laboratory and the results are stored in the memory. A single wire communication can be established between any microcontroller like Arduino and the DHT11 Sensor. Also, the length of the cable can be as long as 20 meters. The data from the sensor consists of integral and decimal parts for both Relative Humidity (RH) and temperature. The data from the DHT11 sensor consists of 40 – bits.



The DHTxx sensors have four pins, VCC, GND, data pin and a not connected pin which has no usage. A pull-up resistor from 5K to 10K Ohms is required to keep the data line high and in order to enable the communication between the sensor and the Arduino Board. There are some versions of these sensors that come with a breakout boards with built-in pull-up resistor and they have just 3 pins.

Python Program

One popular IoT platform is the Raspberry Pi with a DHT11 or DHT22 sensor for temperature and humidity readings. Here's an example Python program to read temperature and humidity data using a Raspberry Pi and a DHT22 sensor. Before running this code, make sure you have the required libraries installed. You can install them using pip:

```
pip install Adafruit_DHT
```

Coding

```
import Adafruit_DHT import
```

```
time
```

```
# Set the sensor type (DHT11 or DHT22)
```

```
SENSOR = Adafruit_DHT.DHT22
```

```
# Set the GPIO pin where the sensor is connected PIN = 4
```

```
while True: try:
```

```
    # Attempt to read the temperature and humidity from the sensor
```

```
    humidity, temperature = Adafruit_DHT.read_retry(SENSOR, PIN) # Check
```

```
    if data was successfully read
```

```
    if humidity is not None and temperature is not None:
```

```
        # Print the temperature and humidity values
```

```
        print(f'Temperature: {temperature:.2f}°C')
```

```
        print(f'Humidity: {humidity:.2f}%')
```

```
    else:
```

```
        # Failed to retrieve data from the sensor
```

```
        print('Failed to retrieve data from the sensor. Please check the connection.') # Wait
```

```
        for a few seconds before reading the sensor again
```

```
        time.sleep(2)
```

```
except KeyboardInterrupt:
```

```
    # Exit the program if the user presses Ctrl+C
```

```
    print('Program terminated by user.')
```

```
    break
```

```
except Exception as e:
```

```
    # Handle other exceptions
```

```
    print(f'Error: {e}')
```

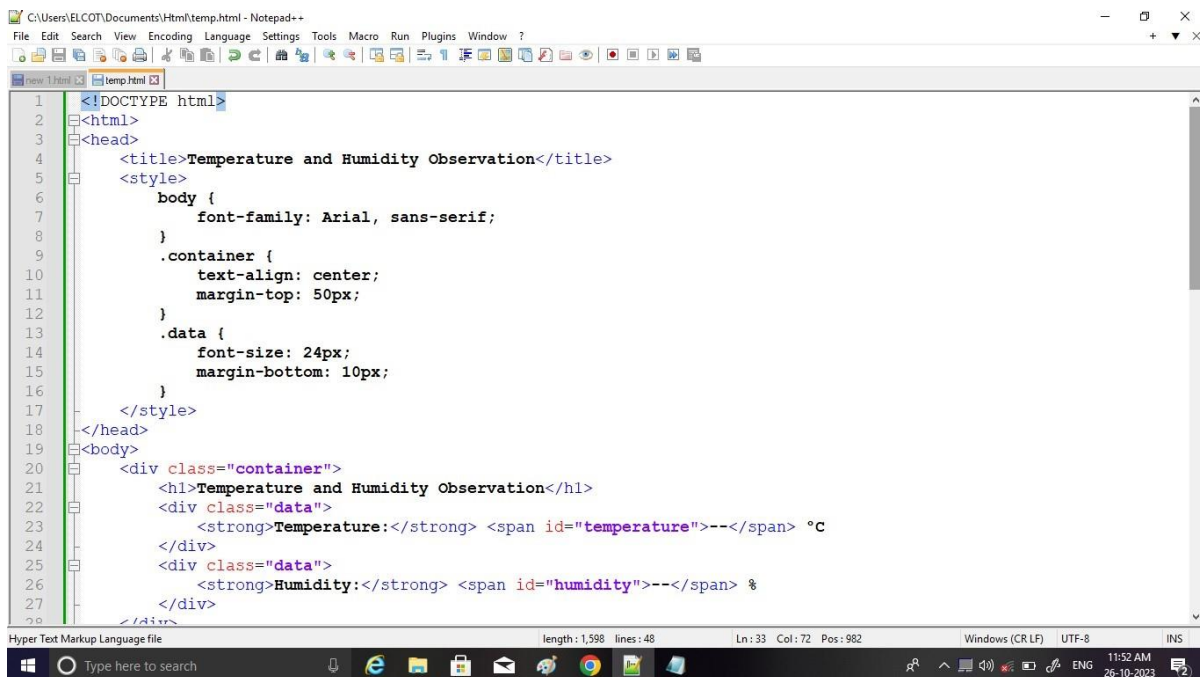
TECHNOLOGIES USED:

The following technologies used in this project are web based platforms OF
HTML , CSS

WEB PLATFORM:

Because HTML and CSS are primarily used for structuring and styling web pages, respectively, it is not possible to create a complete temperature and monitoring system using just these two languages. Additional technologies, such as JavaScript for functionality and data communication, as well as a backend server to handle data from sensors, are required to create a temperature and humidity monitoring system. However, I can provide you with a simple HTML and CSS template for a water monitoring system's user interface. Here's a demo of the following project with source code.

Source code:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Temperature and Humidity Observation</title>
5 <style>
6   body {
7     font-family: Arial, sans-serif;
8   }
9   .container {
10    text-align: center;
11    margin-top: 50px;
12  }
13  .data {
14    font-size: 24px;
15    margin-bottom: 10px;
16  }
17 </style>
18 </head>
19 <body>
20 <div class="container">
21 <h1>Temperature and Humidity Observation</h1>
22 <div class="data">
23 <strong>Temperature:</strong> <span id="temperature">---</span> °C
24 </div>
25 <div class="data">
26 <strong>Humidity:</strong> <span id="humidity">---</span> %
27 </div>
28 </div>
```

```
C:\Users\ELCOT\Documents\Html\temp.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new.html temp.html
24 </div>
25 <div class="data">
26 <strong>Humidity:</strong> <span id="humidity">--</span> %
27 </div>
28 </div>
29 <script>
30 // You would need JavaScript to update the values in real-time.
31 // Here's a simplified example using setInterval:
32 function updateData() {
33 // Replace these values with your IoT data retrieval logic.
34 const temperatureValue = 25.5; // Change this with the actual temperature value.
35 const humidityValue = 50.2; // Change this with the actual humidity value.
36
37 document.getElementById("temperature").textContent = temperatureValue.toFixed(2);
38 document.getElementById("humidity").textContent = humidityValue.toFixed(2);
39
40
41 // Update the data every 5 seconds (for example).
42 setInterval(updateData, 5000);
43
44 // Call updateData() once immediately to show initial values.
45 updateData();
46 </script>
47 </body>
48 </html>
Hyper Text Markup Language file length: 1,598 lines: 48 Ln: 33 Col: 72 Pos: 982 Windows (CR LF) UTF-8 INS
Type here to search 11:52 AM 26-10-2023
```

Output:



RESULTS AND DISCUSSIONS

Temperature Response shows a linear correlation between the actual temperature (measured by LM35) and the temperature calculated from the Q factor of the sensor's response. As shown, there is a small offset in the linear line. This is due to the different temperature response times between the LCR sensor (the response time for the thermistor was 0.9 s, but the actual time for each measurement was 50 s) and the commercial LM35 temperature sensor (response time was about 10 ms), coupled with the fact that the measurements were recorded while the temperature was changing. Therefore, as temperature varied, the LM35, which had a slightly faster response time, would have already registered a change while the LCR sensor was still recording temperature from the previous moment.

The LCR sensor shows a linear response with the actual temperature, measured with a commercial temperature sensor LM35 (Texas Instruments, Dallas, TX). The resolution of the measured temperature sensor was 0.1 °C, and the standard deviation of the measurements compared to the ideal line was 0.1 °C. The measured and actual temperatures when the test chamber was repeatedly set at 37 °C and 40 °C. These two set points represent the temperature for a normal and severely-infected human. plots the sensor measurements at multiple cycles, showing no observable drift. Errors in Figs. 8 and 9 are mostly due to the uncertainties carried from the Q factor calculations.

However, we do not believe this will constitute a concern for the intended application because infection diagnosis does not require a very fast sampling rate. Typically, it would be sufficient to measure the internal wound temperature for a few times a day to catch the early onset of infections. In practice, the temperature measurements can be sampled at 5–10 Hz and then averaged to produce a single data point. This will increase the accuracy while still limit the measurement time to within a few seconds. Furthermore, as determined by Romano et al. [20], a differential temperature measurement of 0.9 °C on the skin surfaces is sensitive enough to identify an infection at the site of a surgery. With averaging, the LCR sensor should have sufficient accuracy for detecting infection. It was also observed that there was a small resonance frequency shift corresponding to the temperature change. The temperature dependency of the resonance frequency was small (~ 1 kHz out of 27 MHz per °C). This shift was due to the expansion of the inductive windings or the core.

CONCLUSION

It is evident from this project work that Temperature and Humidity Sensor Project can be cheaply made from low-cost locally available components and be used to monitor and control the temperature and humidity at the data center. And better still, the components required are so small and few that they can be packaged into a small container. The designed project was tested a number of times and certified to achieve the aim of the project. This Temperature and Humidity Sensor Project can also be done using the esp8266 or various other sensors. Hence, this system is scalable and flexible. The design and application of the LCR sensor for monitoring temperature at interference screws are presented. The LCR sensor was demonstrated to be stable with temperature, producing linear and consistent temperature with no observable measurement drift. Although the position and orientation of the sensor with respect to the detection coil affect its performance, this study has shown that the sensor can still produce accurate results as long as it is within 20 mm from the coil, and the

misalignment between their angles is 40° or less. Experimental results also show the signal from the sensor can further improve with increasing number of turns and a thicker wire in the sensor's inductor.S

