Project Objectives:

Environmental Monitoring: Ensure that temperature and humidity levels in a particular environment (e.g., a greenhouse, data center, or museum) remain within a specified range to create ideal conditions for the intended purpose.

Energy Efficiency: Optimize heating, ventilation, and air conditioning (HVAC) systems to save energy by maintaining temperature and humidity levels at the most energy-efficient settings without sacrificing comfort or product quality.

Predictive Maintenance: Implement predictive maintenance strategies by monitoring temperature and humidity in industrial equipment to detect anomalies that may indicate the need for maintenance before a breakdown occurs.

Climate Research:Collect and analyze temperature and humidity data for scientific research, including climate modeling and studies on weather patterns.

Remote Monitoring: Implement remote monitoring systems to provide real-time data and alerts for temperature and humidity conditions, enabling quick response to deviations or issues.

Code Implementation:

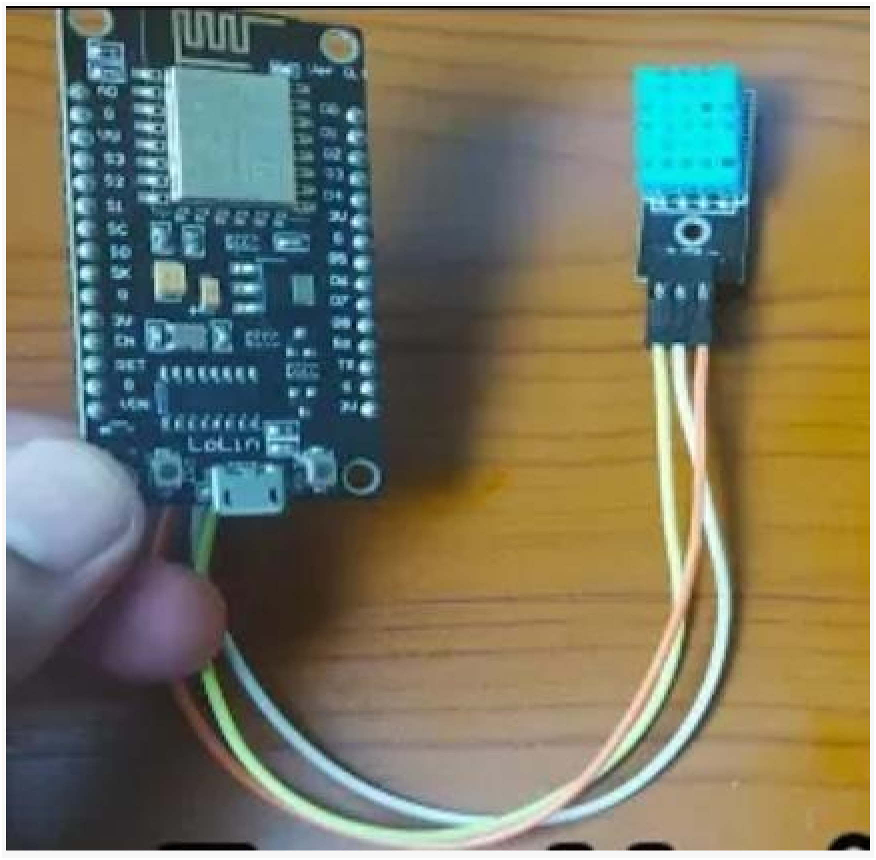
```
#define SECRET_SSID ""
#define SECRET_PASS ""
#define SECRET_DEVICE_KEY ""




#include "arduino_secrets.h"
/*
 Sketch generated by the Arduino IoT Cloud Thing "Temperature and Humidity
monitoring"
 https://create.arduino.cc/cloud/things/b03581ea-7134-40e3-92db-cffc33abfa0b

 Arduino IoT Cloud Variables description
```

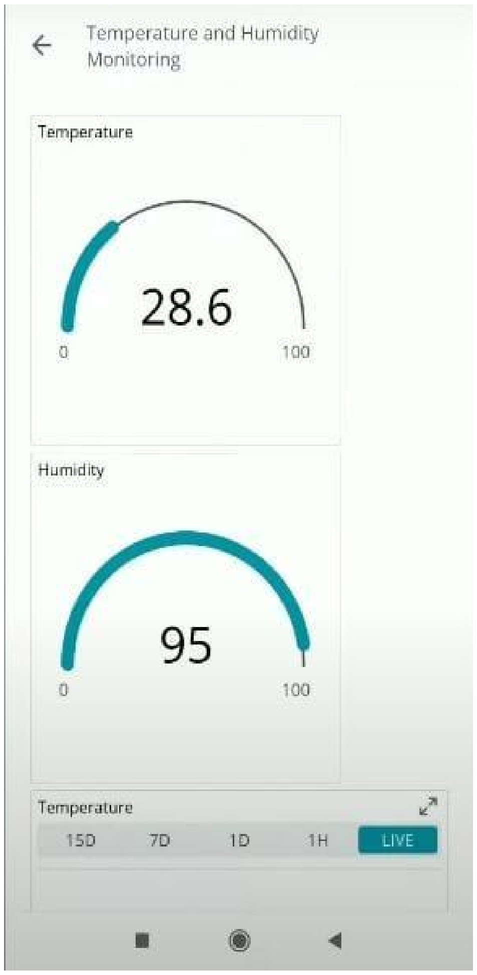
```
  The following variables are automatically generated and updated when changes are
made to the Thing

  CloudRelativeHumidity humidity;
  String msg;
  CloudTemperatureSensor temperature;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have
functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include "DHT.h"
#define DHTpin 2 // D4 on the nodemcu ESP8266
#define DHTTYPE DHT11
DHT dht(DHTpin,DHTTYPE);

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is
found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
```

```
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  // Your code here

   float hm= dht.readHumidity();
    Serial.print("Humidity ");
    Serial.println(hm);
    float temp=dht.readTemperature();
      Serial.print("Temperature ");
    Serial.println(temp);
    humidity=hm;
    temperature=temp;
    message="Temperature = " + String (temperature)+"  Humidity = " + String(humidity);

}

/*
  Since Humidity is READ_WRITE variable, onHumidityChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onHumidityChange()  {
  // Add your code here to act upon Humidity change
}
/*
  Since Msg is READ_WRITE variable, onMsgChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onMsgChange()  {
  // Add your code here to act upon Msg change
}

void dht_sensor_getdata()
  {
    float hm= dht.readHumidity();
    Serial.print("Humidity ");
    Serial.println(hm);
    float temp=dht.readTemperature();
      Serial.print("Temperature ");
    Serial.println(temp);
    humidity=hm;
    temperature=temp;
    message="Temperature = " + String (temperature)+"  Humidity = " + String(humidity);
  }
```

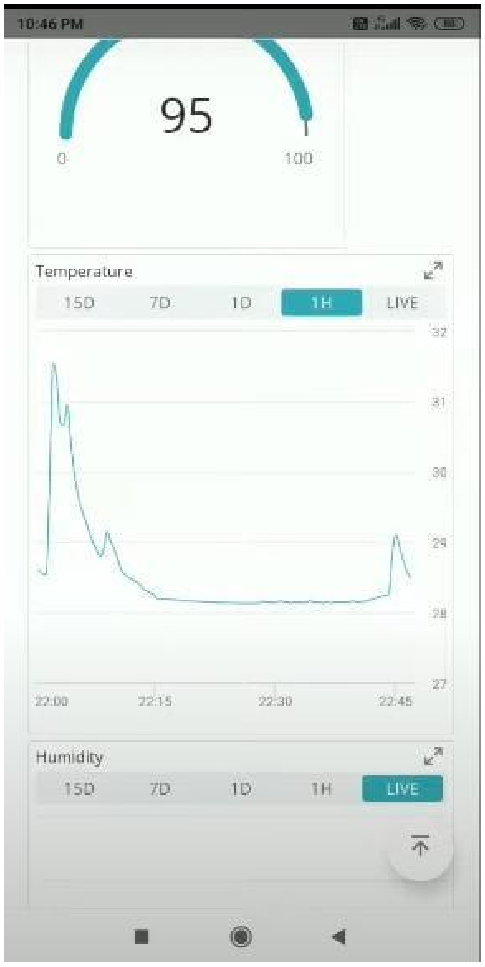
```
// Code generated by Arduino IoT Cloud

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>


const char THING_ID[]           = "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx";
const char DEVICE_LOGIN_NAME[]  = "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxx";

const char SSID[]               = SECRET_SSID;    // Network SSID (name)
const char PASS[]               = SECRET_PASS;    // Network password (use for WPA, or use
as key for WEP)
const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password

void onHumidityChange();
void onMsgChange();

CloudRelativeHumidity humidity;
String msg;
CloudTemperatureSensor temperature;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.setThingId(THING_ID);
  ArduinoCloud.addProperty(humidity, READWRITE, ON_CHANGE, onHumidityChange);
  ArduinoCloud.addProperty(msg, READWRITE, ON_CHANGE, onMsgChange);
  ArduinoCloud.addProperty(temperature, READ, ON_CHANGE, NULL);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

Output:

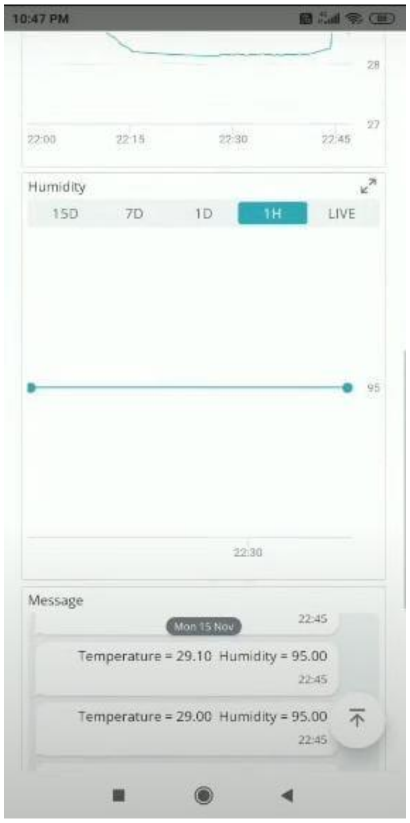




Temperature and Humidity Monitoring
Temperature
28.6
0
100
Humidity
95
0
100
Temperature
15D
7D
1D
1H
LIVE



10:46 PM
95
0
100
Temperature
15D
7D
1D
1H
LIVE
32
31
30
29
28
27
22:00
22:15
22:30
22:45
Humidity
15D
7D
1D
1H
LIVE



22:45

Temperature = 29.10 Humidity = 95.00
22:45

Temperature = 29.00 Humidity = 95.00
22:45

Conclusion:

In conclusion, the Temperature and Humidity Monitor project implemented with Arduino IoT Cloud and ESP8266 demonstrates the power of IoT technology in providing real-time environmental data monitoring and remote control capabilities. This project has shown how an accessible and cost-effective combination of hardware and cloud services can be harnessed to achieve various practical objectives, from environmental control to energy efficiency and data-driven decision-making.