GOVERNMENT COLLEGE OF ENGINEERING

ERODE

(FORMERLY KNOWN AS IRTT)

BE., ELECTRONICS AND COMMUNICATION ENGINEERING

TECHNOLOGY NAME **: INTERNET OF THINGS**

ENVIRONMENTAL MONITORING AUTOMATIC SOIL IRRIGATION SYSTEM

NAME OF THE STUDENTS	**REGISTER NUMBER**
TEAM LEADER:	
P.DINESHKUMAR	731121106010
TEAM MEMBERS:	
1. V.JANANI	731121106018
2. M.DEVAKI	731121106004
3. S.DHANALAKSHIMI	731121106005
4. P.JEEVITHA	731121106306

PHASE 4

Under the mentor of

DR.M.POONGOTHAI ME, Phd.,

DEPARTMENT OF INFORMATION TECHNOLOGY(IT)

Government College of Engineering, Erode

Near Vasavi College Post,

TamilNadu – 638316.

Affiliated to Anna University , Chennai.

AUTOMATIC SOIL IRRIGATION SYSTEM

PROGRAM:

```
#include <WiFi.h>
#include <DHT.h>
#include <FirebaseArduino.h>
#include <PubSubClient.h>

// Replace with your Wi-Fi credentials
const char* ssid = "Wokwi-Guest";
const char* password = "YourWiFiPassword";

// Your Firebase project credentials
const char* firebaseHost = "https://ibmgr1-default-rtdb.firebaseio.com/ibmgr1";
const char* firebaseAuth = "AIzaSyBy9qLB8-kIxS1Kftv3I6pY3aVWmOI83w8";

const int DHTPin = 4; // Pin to which the DHT sensor is connected
DHT dht(DHTPin, DHT22);

// MQTT Broker
const char* mqttServer = "ed342ccf55c1484eb534c8c92861048b.s2.eu.hivemq.cloud"; //
Replace with your MQTT broker address
const int mqttPort = 8883;
const char* mqttUser = "aiyengar";
const char* mqttPassword = "Mh12hn4226!!!";
const char* mqttTopic = "/ESP32";

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");

  // Initialize Firebase
  Firebase.begin(firebaseHost, firebaseAuth);

  // MQTT setup
  mqttClient.setServer(mqttServer, mqttPort);
  mqttClient.setCallback(mqttCallback);

  // Connect to MQTT broker
  if (mqttClient.connect("ESP32Client", mqttUser, mqttPassword)) {
    Serial.println("Connected to MQTT broker");
  }
}

void loop() {
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (!isnan(humidity) && !isnan(temperature)) {
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print(" °C, Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");

    // Firebase
    Firebase.pushFloat("/environment/temperature", temperature);
    Firebase.pushFloat("/environment/humidity", humidity);

    // MQTT publish
    String tempPayload = String(temperature);
    String humidityPayload = String(humidity);

    mqttClient.publish((String(mqttTopic) + "/temperature").c_str(), tempPayload.c_str());
    mqttClient.publish((String(mqttTopic) + "/humidity").c_str(), humidityPayload.c_str());
  } else {
    Serial.println("Failed to read from DHT sensor");
  }

  mqttClient.loop();
  delay(60000); // Send data every 60 seconds
}

void mqttCallback(char* topic, byte* payload, unsigned int length) {
  // Handle MQTT subscription messages if needed
}
```

PRORAM DISCRIPTION:

This program is written for an ESP32 microcontroller to read data from a DHT22 temperature and humidity sensor and then send this data to both Firebase and an MQTT broker. Here's a breakdown of the code:

Importing Libraries:

The program begins by including several libraries:

WiFi.h:

This library allows the ESP32 to connect to a Wi-Fi network.

DHT.h:

Used for working with DHT sensors, in this case, a DHT22.

FirebaseArduino.h:

This library enables communication with the Firebase Realtime Database.

PubSubClient.h:

Used for MQTT (Message Queuing Telemetry Transport) communication.

Variable Declarations:

Wi-Fi and Firebase Credentials:

To specify Wi-Fi SSID and password, as well as Firebase project credentials (Firebase Realtime Database URL and authentication key).

DHT Sensor Pin:

The GPIO pin to which the DHT22 sensor is connected.

DHT Instance:

The program creates an instance of the DHT sensor.

MQTT Broker: Configuration for connecting to the MQTT broker, including the broker's address, port, username, and password.

Setup Function:

It initializes the serial communication for debugging purposes.

Connects to the Wi-Fi network using the provided credentials.

Initializes the Firebase connection using the Firebase credentials.

Sets up MQTT client with broker information and connects to the broker.

Loop Function:

Reads temperature and humidity from the DHT22 sensor using the readHumidity and readTemperature functions.

If the readings are valid (not NaN), it does the following:

Prints the temperature and humidity values to the serial monitor.

Sends the temperature and humidity data to Firebase Realtime Database using the Firebase Arduino library.

Publishes temperature and humidity data to specific MQTT topics using the MQTT client.

If the DHT sensor readings are invalid, it prints an error message.

The mqttClient.loop() function is called to handle MQTT communication.

There's a delay of 60 seconds before taking the next set of readings and sending data again.

MQTT Callback Function:

This function is empty and can be used to handle incoming MQTT messages, although it's not utilized in this code.

In summary, this program allows an ESP32 to periodically read temperature and humidity data from a DHT22 sensor and transmit this data to both Firebase Realtime Database and an MQTT broker. The program will repeat this process with a one-minute delay between readings.

FUNCTIONING OF SYSTEM:

Library Inclusions:

The code begins by including necessary libraries, including WiFi.h, DHT.h, FirebaseArduino.h, and PubSubClient.h. These libraries are used for Wi-Fi connectivity, working with the DHT22 sensor, interfacing with Firebase, and MQTT communication, respectively.

Variable Declarations:

Wi-Fi and Firebase Credentials: Replace the ssid and password with Wi-Fi network credentials. Similarly, should set firebaseHost to your Firebase Realtime Database URL and firebaseAuth to Firebase authentication key.

DHT Sensor Pin: This variable specifies the GPIO pin to which the DHT22 sensor is connected.

DHT Instance: An instance of the DHT class is created, using the specified pin and DHT sensor type (DHT22).

MQTT Broker Configuration: The code sets up MQTT broker details, including server address (mqttServer), port (mqttPort), username (mqttUser), and password (mqttPassword). The mqttTopic variable defines the MQTT topic to which data will be published.

Setup Function:

The setup() function is responsible for initializing the hardware and establishing connections.

It begins serial communication for debugging, which allows to monitor the program's progress through the serial monitor.

It connects the ESP32 to the specified Wi-Fi network. The code waits until a successful connection is established.

Firebase is initialized using the provided Firebase credentials.

MQTT setup:

The MQTT client (mqttClient) is configured with the MQTT broker's information.

The code attempts to connect to the MQTT broker using the mqttClient.connect() method, providing a client ID, username, and password. A successful connection is printed to the serial monitor if it's established.

Loop Function:

The loop() function is where the main functionality of the program occurs. It runs repeatedly.

It reads the temperature and humidity data from the DHT22 sensor using the readHumidity() and readTemperature() functions.

If valid readings are obtained (not , which can happen when the sensor fails to provide data), the code does the following:

Prints the temperature and humidity data to the serial monitor.

Sends this data to Firebase by using the Firebase.pushFloat() function for both temperature and humidity.

Publishes the temperature and humidity data to specific MQTT topics using the mqttClient.publish() method.

If the sensor readings are invalid, it prints an error message.

The mqttClient.loop() is called to handle MQTT communication.

A delay of 60 seconds (delay(60000)) is included to control how often the data is sent. Data is sent every 60 seconds in this case.

MQTT Callback Function:

The mqttCallback() function is a placeholder for handling incoming MQTT subscription messages if required. In the provided code, it's empty and doesn't perform any specific actions.