```
/*
  Sensor Module Code
  Written by Varad Chaskar
*/

#include <WiFi.h>
#include <HTTPClient.h>
#include <WebServer.h>
#include <EEPROM.h>
#include "ThingSpeak.h"
#include <DHT.h>

// Variables
int i = 0;
int statusCode;
const char* ssid = "Default SSID";
const char* passphrase = "Default password";
String st;
String content;
String esid;
String epass = "";

// Function Declarations
bool testWifi(void);
void launchWeb(void);
void setupAP(void);

// Establishing Local server at port 80
WebServer server(80);

#define DHTPIN 2    // Define the pin to which DHT11 data pin is
connected
#define DHTTYPE DHT11    // DHT11 sensor type
DHT dht(DHTPIN, DHTTYPE);
const int potPin = 34;
int potValue = 0;
```

```
const int potPin2 = 35;
int potValue2 = 0;
WiFiClient client;

// ThingSpeak channel details
unsigned long myChannelNumber = 2384399;                //
Channel number taken from cloud
const char *myWriteAPIKey = "ELW2NF5Q83OGB39G";         // Write
API key of cloud
const char *myCounterReadAPIKey = "3D8NH4JCI0EDYMIU";   // Read
API key from cloud

void setup()
{
  Serial.begin(115200); // Initializing Serial Monitor
  Serial.println();
  Serial.println("Disconnecting current WiFi connection");
  WiFi.disconnect();
  EEPROM.begin(512); // Initializing EEPROM
  delay(10);
  pinMode(15, INPUT);
  Serial.println();
  Serial.println();
  Serial.println("Startup");

  // Read EEPROM for SSID and Password
  Serial.println("Reading EEPROM SSID");
  for (int i = 0; i < 32; ++i)
  {
    esid += char(EEPROM.read(i));
  }
  Serial.println();
  Serial.print("SSID: ");
  Serial.println(esid);
  Serial.println("Reading EEPROM Password");
  for (int i = 32; i < 96; ++i)
```

```
  {
    epass += char(EEPROM.read(i));
  }
  Serial.print("Password: ");
  Serial.println(epass);

  WiFi.begin(esid.c_str(), epass.c_str());
  dht.begin();

  ThingSpeak.begin(client);
  pinMode(22, OUTPUT);
}

void loop()
{
  if (WiFi.status() == WL_CONNECTED)
  {
    digitalWrite(22, HIGH);
    delay(2000);  // Wait for 2 seconds to avoid too frequent
readings

    // Read temperature and humidity from the sensor
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

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

    // Display the temperature and humidity on the Serial
Monitor
    Serial.print("Temperature: ");
    Serial.print(temperature);
```

```
    Serial.print(" °C, Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
    potValue = analogRead(potPin);
    Serial.println(potValue);
    potValue2 = analogRead(potPin2);
    Serial.println(potValue2);

    // Set ThingSpeak fields with sensor values
    ThingSpeak.setField(1, temperature);
    ThingSpeak.setField(2, humidity);
    ThingSpeak.setField(3, potValue);
    ThingSpeak.setField(4, potValue2);

    // Write to the ThingSpeak channel
    int x = ThingSpeak.writeFields(myChannelNumber,
myWriteAPIKey);
    if (x == 200)
    {
      Serial.println("Channel update successful.");
    }
  }
  else
  {
    digitalWrite(22, LOW);
  }

  if (testWifi() && (digitalRead(15) != 1))
  {
    Serial.println("Connection status positive");
    return;
  }
  else
  {
    digitalWrite(22, LOW);
    Serial.println("Connection Status Negative / D15 HIGH");
```

```
    Serial.println("Turning the HotSpot On");
    launchWeb();
    setupAP(); // Setup HotSpot
  }

  Serial.println();
  Serial.println("Waiting.");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(100);
    server.handleClient();
  }
  delay(1000);
}

// Functions used for WiFi credentials saving and connecting to
it (no need to change)
bool testWifi(void)
{
  digitalWrite(22, LOW);
  int c = 0;
  while (c < 20)
  {
    if (WiFi.status() == WL_CONNECTED)
    {
      return true;
    }
    delay(500);
    Serial.print("*");
    c++;
  }
  Serial.println("");
  Serial.println("Connect timed out, opening AP");
  return false;
}
```

```
void launchWeb()
{
  digitalWrite(22, LOW);
  Serial.println("");
  if (WiFi.status() == WL_CONNECTED)
    Serial.println("WiFi connected");
  Serial.print("Local IP: ");
  Serial.println(WiFi.localIP());
  Serial.print("SoftAP IP: ");
  Serial.println(WiFi.softAPIP());
  createWebServer();
  // Start the server
  server.begin();
  Serial.println("Server started");
}

void setupAP(void)
{
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
  int n = WiFi.scanNetworks();
  Serial.println("Scan done");
  if (n == 0)
    Serial.println("No networks found");
  else
  {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i)
    {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
```

```
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
      delay(10);
    }
  }

  Serial.println("");
  st = "<ol>";
  for (int i = 0; i < n; ++i)
  {
    // Print SSID and RSSI for each network found
    st += "<li>";
    st += WiFi.SSID(i);
    st += " (";
    st += WiFi.RSSI(i);
    st += ")";
    st += "</li>";
  }
  st += "</ol>";
  delay(100);
  WiFi.softAP("ElectronicsInnovation", "");
  Serial.println("Initializing SoftAP for WiFi credentials
modification");
  launchWeb();
  Serial.println("Over");
}

void createWebServer()
{
  digitalWrite(22, LOW);
  server.on("/", []() {
    IPAddress ip = WiFi.softAPIP();
    String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' +
String(ip[2]) + '.' + String(ip[3]);
```

```
    content = "<!DOCTYPE HTML>\r\n<html>Welcome to WiFi
Credentials Update page";
    content += "<form action=\"/scan\" method=\"POST\"><input
type=\"submit\" value=\"scan\"></form>";
    content += ipStr;
    content += "<p>";
    content += st;
    content += "</p><form method='get'
action='setting'><label>SSID: </label><input name='ssid'
length=32><input name='pass' length=64><input
type='submit'></form>";
    content += "</html>";
    server.send(200, "text/html", content);
  });

  server.on("/scan", []() {
    IPAddress ip = WiFi.softAPIP();
    String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' +
String(ip[2]) + '.' + String(ip[3]);
    content = "<!DOCTYPE HTML>\r\n<html>Go back";
    server.send(200, "text/html", content);
  });

  server.on("/setting", []() {
    String qsid = server.arg("ssid");
    String qpass = server.arg("pass");
    if (qsid.length() > 0 && qpass.length() > 0)
    {
      Serial.println("Clearing EEPROM");
      for (int i = 0; i < 96; ++i)
      {
        EEPROM.write(i, 0);
      }
      Serial.println(qsid);
      Serial.println("");
      Serial.println(qpass);
```

```
      Serial.println("");
      Serial.println("Writing EEPROM SSID:");
      for (int i = 0; i < qsid.length(); ++i)
      {
        EEPROM.write(i, qsid[i]);
        Serial.print("Wrote: ");
        Serial.println(qsid[i]);
      }
      Serial.println("Writing EEPROM Password:");
      for (int i = 0; i < qpass.length(); ++i)
      {
        EEPROM.write(32 + i, qpass[i]);
        Serial.print("Wrote: ");
        Serial.println(qpass[i]);
      }
      EEPROM.commit();
      content = "{\"Success\":\"Saved to EEPROM... reset to boot
into new WiFi\"}";
      statusCode = 200;
      ESP.restart();
    }
    else
    {
      content = "{\"Error\":\"404 not found\"}";
      statusCode = 404;
      Serial.println("Sending 404");
    }
    server.sendHeader("Access-Control-Allow-Origin", "*");
    server.send(statusCode, "application/json", content);
  });
}
```