```
// Code for Actuator Module
// Author: Varad Chaskar

#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WebServer.h>
#include <EEPROM.h>
#include "ThingSpeak.h"

// Variables
int i = 0;
int statusCode;
const char* ssid = "Default_SSID";
const char* passphrase = "Default_Password";
String st;
String content;

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

// Establishing Local server at port 80 whenever required
ESP8266WebServer server(80);
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

const int ledPin1 = 5;
const int ledPin2 = 4;
```

```
const int ledPin3 = 0;
const int ledPin4 = 2;

bool LED = false;
bool LEDValue = false;

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(LED_BUILTIN, OUTPUT);
  Serial.println();
  Serial.println();
  Serial.println("Startup");

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

  // Connect to WiFi
```

```
  WiFi.begin(esid.c_str(), epass.c_str());
  if (testWifi()) {
    Serial.println("Successfully Connected!!!");
    return;
  } else {
    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();
  }

  // Initialize ThingSpeak
  ThingSpeak.begin(client);

  pinMode(12, OUTPUT);
  pinMode(14, OUTPUT);
  pinMode(1, OUTPUT);
}

void loop() {
  if ((WiFi.status() == WL_CONNECTED)) {
    digitalWrite(1, HIGH);

    // Read and control additional fields from ThingSpeak
channel
    int A = ThingSpeak.readLongField(myChannelNumber, 5,
myCounterReadAPIKey);
    analogWrite(ledPin1, A);
    int B = ThingSpeak.readLongField(myChannelNumber, 6,
myCounterReadAPIKey);
```

```
    analogWrite(ledPin2, B);
    int C = ThingSpeak.readLongField(myChannelNumber, 7,
myCounterReadAPIKey);
    analogWrite(ledPin3, C);
    int D = ThingSpeak.readLongField(myChannelNumber, 8,
myCounterReadAPIKey);
    if (D == 100 && LED == false) {
      LED = true;
      digitalWrite(14, HIGH);
      delay(500);
      digitalWrite(14, LOW);
    }
    if (D == 0 && LED == true) {
      LED = false;
      digitalWrite(14, HIGH);
      delay(500);
      digitalWrite(14, LOW);
    }
    if (D == 200 && LEDValue == false) {
      LEDValue = true;
      digitalWrite(12, HIGH);
      delay(500);
      digitalWrite(12, LOW);
    }
    if (D == 300 && LEDValue == true) {
      LEDValue = false;
      digitalWrite(12, HIGH);
      delay(500);
      digitalWrite(12, LOW);
    }
    Serial.print("Reading Data");
  } else {
    digitalWrite(1, LOW);
  }
}
```

```
// Functions used for WiFi credentials saving and connecting
bool testWifi(void) {
  int c = 0;
  Serial.println("Waiting for Wifi to connect");
  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() {
  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(")");
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ?
" " : "*");
      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 += (WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " : "*";
    st += "</li>";
  }
  st += "</ol>";
  delay(100);
  WiFi.softAP("ElectronicsInnovation", "");
```

```
Serial.println("Initializing_softap_for_wifi_credentials_modific
ation");
  launchWeb();
  Serial.println("Over");
}

void createWebServer() {
  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", []() {
    // setupAP();
    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.reset();
    } else {
      content = "{\"Error\":\"404 not found\"}";
      statusCode = 404;
      Serial.println("Sending 404");
    }
    server.sendHeader("Access-Control-Allow-Origin", "*");
    server.send(statusCode, "application/json", content);
```

```
  });
}
```