#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <ESP8266WebServer.h>

#include <EEPROM.h>

#include <FirebaseArduino.h> //FireBase Arduino library

#define LED D2

#define outputPin D0

#define FIREBASE\_HOST "my-smart-lock-72383.firebaseio.com"

#define FIREBASE\_AUTH "8REtTVyDSp5YPSFXOrmbplx2CvaI2bN5QvGhB25y"

//Variables

int i = 0;

int p=0;

int statusCode;

const char\* ssid = "Default\_SSID";

const char\* passphrase = "Default\_Password";

String st;

String content;

//Function Decalration

bool testWifi(void);

void launchWeb(void);

void setupAP(void);

//Establishing Local server at port 80 whenever required

ESP8266WebServer server(80);

void setup()

{

Serial.begin(9600); //Initialising if(DEBUG)Serial Monitor

pinMode(LED,OUTPUT);

pinMode(outputPin,OUTPUT);

digitalWrite(outputPin,LOW);

Serial.println();

Serial.println("Disconnecting current wifi connection");

WiFi.disconnect();

EEPROM.begin(512); //Initialasing EEPROM

delay(10);

pinMode(LED\_BUILTIN, OUTPUT);

Serial.println();

Serial.println();

Serial.println("Startup");

//---------------------------------------- Read eeprom for ssid and pass

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);

WiFi.begin(esid.c\_str(), epass.c\_str());

if (testWifi())

{

Serial.println("Succesfully Connected!!!");

return;

}

else

{

Serial.println("Turning the HotSpot On");

launchWeb();

setupAP();// Setup HotSpot

}

Serial.println();

Serial.println("Waiting.");

while ((WiFi.status() != WL\_CONNECTED) && p<=60)

{

delay(1000);

server.handleClient();

p++;

}

if (WiFi.status() == WL\_CONNECTED)

{

Serial.println(WiFi.localIP());

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

}

}

void loop() {

if ((WiFi.status() == WL\_CONNECTED))

{

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

while(WiFi.status() == WL\_CONNECTED ){

if (Firebase.failed()) {

Serial.print("setting /number failed:");

Serial.println(Firebase.error());

return;

}

// get value

Serial.print("Status ");

Serial.println(Firebase.getString("Locks/75bbbrk5VmPM8V7WQoFvmJbS95F2/Lock1/Status"));

String status=Firebase.getString("Locks/75bbbrk5VmPM8V7WQoFvmJbS95F2/Lock1/Status");

delay(1000);

if(status.equals("ON")){

digitalWrite(outputPin,LOW);

}else{

digitalWrite(outputPin,HIGH);

}

digitalWrite(LED,HIGH);

delay(500);

digitalWrite(LED,LOW);

delay(200);

}

}

else

{

digitalWrite(outputPin,LOW);

p=0;

Serial.println();

Serial.println();

Serial.println("Startup");

//---------------------------------------- Read eeprom for ssid and pass

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);

WiFi.begin(esid.c\_str(), epass.c\_str());

if (testWifi())

{

Serial.println("Succesfully Connected!!!");

return;

}

else

{

Serial.println("Turning the HotSpot On");

launchWeb();

setupAP();// Setup HotSpot

}

Serial.println();

Serial.println("Waiting.");

while ((WiFi.status() != WL\_CONNECTED) && p<=60)

{

delay(1000);

server.handleClient();

p++;

}

if (WiFi.status() == WL\_CONNECTED)

{

Serial.println(WiFi.localIP());

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

}

}

}

//----------------------------------------------- Fuctions used for WiFi credentials saving and connecting to it which you do not need to change

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("Nodemcu", "");

Serial.println("Initializing\_softap\_for\_wifi credentials\_modification");

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 pass:");

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);

});

}

}