```
/*
  Web and switch control of "addressable" PL9823 controlled leds.
  Deid Reimer 2018-09
  deid@drsol.com
*/
#include <FastLED.h>

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

// About the leds
#define NUM_LEDS 6
#define LED_PIN 1
#define LED_TYPE PL9823

// Instantiate and tell the web server to listen on port 80
ESP8266WebServer server(80);

//  ID and Password for web control
const char* id = "pl9823";
const char* pw = "neopixelnot";

// Initialization values for the various led programs.

// Rainbow colours for the rainbow program - this vector is  shifted to move the rainbow
int rainbow[] = {0xff0000, 0xff4800, 0xffff00, 0x00ff00, 0x0000ff, 0x3000ff, 0x9400d3};

// RGB colours for the chase and random programs.  These are incremented
byte cred = 0;
byte cgreen = 0;
byte cblue = 0;

// Miscellaneous
int rbStart = 0;              // Rainbow start led
int chase = 0;                // chase first led
int chaser = NUM_LEDS-1;        // reverse chase first led
byte colour[] = {255, 0, 0};  // chasing primaries program
const int webSafe = 51;       // Number of web safe colours
bool dirty = false;           // Set if the lights need to be changed

// Variables for the push button and debouncing.
const int selectButton = 4;  // GPIO pin
int selectButtonState;
int lastSelectButtonState = LOW;
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output flickers

// Overal Program variables
int lightProgram = 0;       // Current program selected
int maxProgram = 8;         // Maximum program number

// Count millis for timing of random colour/time programs
unsigned long currentTime;
unsigned long loopTime[NUM_LEDS];

// Web page button variables
String onColour = "style=\"background-color: #f4a742\"";
String offColour = "style=\"background-color: #CCCCCC\"";
String s[] = {offColour, offColour, offColour, offColour, offColour, offColour, offColour, offColour,
offColour};

// Tell the library about the number of leds
CRGB led[NUM_LEDS];

// ******** Functions ********
// Get credentials
```

```
void checkCreds () {
  // If we don't have the credentials ask for them
  if (!server.authenticate(id, pw)) {
    return server.requestAuthentication();
    // } else {
    // Display the select program page
    //   displayPage()
  }
}

// Display the page.
void showPage() {
  server.send(200, "text/html", makePage());
}

// Function to build the select light program web page.
String makePage() {

  // Clear all the button colours to offColour and set the selected to onColour;
  for (int i = 0; i <= maxProgram ; i++) {
    s[i] = offColour;
  }
  s[lightProgram] = onColour;

  String webPage00 = "<html><head><meta charset=\"UTF-8\"><meta name=viewport content=\"width=device-
width, initial-scale=1.5\"><title>Leds</title>";
  String webPage000 = "<meta http-equiv=\"refresh\" content=\"10; url=http://" + WiFi.localIP().toString
() + "/\">";
  String webPage01 = "</head><body>";
  String webPage0 = "<h3>Addressable Leds V0.9</h3>";
  String webPage02 = "<table>";
  String webPage1 = "<tr><td>1 - Primary </td><td><a href=\"program?s=2\"><button " + s[0]  + ">ON</
button></a></td></tr>";
  String webPage2 = "<tr><td>2 - Random </td><td><a href=\"program?s=3\"><button " + s[1]  + ">ON</
button></a></td></tr>";
  String webPage3 = "<tr><td>3 - Web Safe </td><td><a href=\"program?s=4\"><button " + s[2]  + ">ON</
button></a></td></tr>";
  String webPage4 = "<tr><td>4 - Follow </td><td><a href=\"program?s=5\"><button " + s[3] + ">ON</
button></a></td></tr>";
  String webPage5 = "<tr><td>5 - Lead </td><td><a href=\"program?s=6\"><button " + s[4] + ">ON</button></
a></td></tr>";
  String webPage6 = "<tr><td>6 - Random On</td><td><a href=\"program?s=7\"><button " + s[5] + ">ON</
button></a></td></tr>";
  String webPage7 = "<tr><td>7 - All On   </td><td><a href=\"program?s=8\"><button " + s[6] + ">ON</
button></a></td></tr>";
  String webPage8 = "<tr><td>8 - All Off   </td><td><a href=\"program?s=9\"><button " + s[7] + ">ON</
button></a></td></tr>";
  String webPage9 = "<tr><td>9- Rainbow  </td><td><a href=\"program?s=10\"><button " + s[8] + ">ON</
button></a></td></tr></table>";
  String webPage99 = "</body></html>";

  return webPage00 + webPage000 + webPage01 + webPage0 + webPage02 + webPage1 + webPage2 + webPage3 +
webPage4 + webPage5 + webPage6 + webPage7 + webPage8 + webPage9 + webPage99;
}


void newPage() {
  checkCreds();
  makePage();
  showPage();
}
// Function to get and execute a query selecting a program
void execQuery() {
  // Check Credentials and Build the new web page.
  checkCreds();

  // Get the program  number. If there is an argument separate out the value.
```

```
  if (server.args() == 1) {
    String m = server.arg("s");
    int n = m.toInt();

    // check the validity of the supplied number.
    if (n <2 or n > maxProgram + 2) {
      // Got an invalid number just redisplay the web page and return
      makePage();
      showPage();
      return;
    }

    // Set the number back to 0 base and set the new selected program.
    //     We started at 2 so that toInt returning 0 on fail can be caught.
    lightProgram = n - 2;
    makePage();
    showPage();

    // number of arguments not equal to 1 - just redisplay the page.
  } else {
    newPage();
    delay(100);
    return;
  }
}

// Get the current program number and display it.
void getNumber() {
  server.send(200, "text/plain", String(lightProgram));
}

// Invalid request just redisplay the main page.
void handleNotFound() {
  newPage();
}

// Function to request an IP address.
bool getIP(IPAddress ip, IPAddress gateway, IPAddress subnet, char* ssid, char* password, int colour) {

  // disconnect then reconnect to desired ssid
  WiFi.disconnect();
  WiFi.config(ip, gateway, subnet);
  WiFi.begin(ssid, password);

  // Setup Display wifi status on leds.
  int chaseWiFi = 0;
  int j = 0;
  // Loop until a connection or 20 tries
  while (WiFi.status() != WL_CONNECTED and j < 20) {
    delay(500);
    Serial.print(j);
    Serial.println(ssid);
    j++;

    // Sequence the leds to show connecting to WiFi.
    // Turn all leds off
    for ( int i = 0; i < NUM_LEDS; i++) {
      led[i] = 0;
    }
    // Set the next led to choosen colour
    led[chaseWiFi++] =  colour;
    if (chaseWiFi == NUM_LEDS) {
      chaseWiFi = 0;
    }
    FastLED.show();
  }
```

```
  // Display the connection information if connected and return status
  if (WiFi.status() == WL_CONNECTED) {
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.print("Connected to ");
    Serial.println(ssid);
    return true;
  } else {
    return false;
  }
}

// ************************************************
void setup() {
  Serial.begin(9600);
  delay(500);

  Serial.println();
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());

  WiFi.mode(WIFI_STA);
  //  Initialize the fast led library
  FastLED.addLeds<PL9823, LED_PIN>(led, NUM_LEDS);

  //  Connect to WiFi.   Try Steele then lalala and just give up if none.
  bool stat = getIP ({10, 10, 45, 10}, {10, 10, 45, 254}, {255, 255, 255, 0}, "Steele",
"xxxxxxxxxxxxx.", 0xff0000 );
  if (!stat) {
    stat = getIP ({192, 168, 0, 37}, {192, 168, 0, 1}, {255, 255, 255, 0}, "lalala", "yyyyyyyyyyyyyyyyy",
0x0000ff);
  }

  // Continue without an an internet connection.
  if (!stat) {
    Serial.println("Continuing without IP");
     // Set all the leds to green to show no connection
    for ( int i = 0; i < NUM_LEDS; i++) {
      led[i] = 0x00ff00;
    }
    FastLED.show();
    delay(2000);
  }

  // Set up the responders for the possible web requests.
  // If no command just serve the program select web page.  checkCreds will call the build web page
  server.on("/", newPage);

  // is a command get the query and set light program to display
  server.on("/program", execQuery);

  // Get and display the current light program number.
  server.on("/getNumber", getNumber);

  // 404
  server.onNotFound(handleNotFound);

  // Set the pushbutton GPI to input
  pinMode(selectButton, INPUT);

  // Clear the random times to 0
  for (int i; i < NUM_LEDS; i++) {
    loopTime[i] = 0;
  }

  // Set all the leds to off for two seconds.
  for (int i = 0; i < NUM_LEDS; i++) {
```

```
    led[i] = CRGB(0, 0, 0);
  }
  FastLED.show();
  delay(2000);

  // Start the web server
  server.begin();
}

// ***************************************************
void loop() {
  // Look for and handle a web request.
  server.handleClient();

  // Find the light program to execute.
  // --------------------------
  // Chasing primaries
  if (lightProgram == 0) {
    for (int i = 0; i < NUM_LEDS; i++) {
      led[i] = CRGB(colour[i % 3], colour[(i + 1) % 3], colour[(i + 2) % 3]);
    }
    dirty = true;

    byte s = colour[0];
    for (int i = 0 ; i < 3; i++) {
      colour[i] = colour[i + 1];
    }
    colour[2] = s;
    delay(200);

    // --------------------------
    // Random colours and times
  } else if (lightProgram == 1) {
    currentTime = millis();
    for (int i = 0; i < NUM_LEDS; i++) {
      if (currentTime >= loopTime[i]) {
        loopTime[i] = currentTime + random(900, 2000);
        byte red = random(0, 256);
        byte green = random(0, 256);
        byte blue = random(0, 256);
        led[i] = CRGB(red, green, blue);
        dirty = true;
      }
    }

    // --------------------------
    // Web safe
  } else if (lightProgram == 2) {
    currentTime = millis();
    for (int i = 0; i < NUM_LEDS; i++) {
      if (currentTime >= loopTime[i]) {
        loopTime[i] = currentTime + random(900, 2000);
        byte red = random(0, 5) * webSafe;
        byte green = random(0, 5) * webSafe;
        byte blue = random(0, 5) * webSafe;
        led[i] = CRGB(red, green, blue);
        dirty = true;
      }
    }

    // --------------------------
    // Chasing increasing colour
  }  else if (lightProgram == 3) {
    for ( int i = 0; i < NUM_LEDS; i++) {
      led[i] = CRGB(0, 0, 0);
    }
    cred += 3;
```

```
    cgreen += 5;
    cblue += 7;

    led[chase++] = CRGB(cred, cgreen, cblue);
    dirty = true;
    //Serial.println(j);
    if (chase == NUM_LEDS) {
      chase = 0;
    }
    delay(200);

    // --------------------------
    // Chasing  reverse increasing colour
  }  else if (lightProgram == 4) {

    for ( int i = NUM_LEDS - 1 ; i >= 0; i--) {
      led[i] = CRGB(0, 0, 0);
    }
    cred += 3;
    cgreen += 5;
    cblue += 7;

    led[chaser--] = CRGB(cred, cgreen, cblue);
    dirty = true;
    if (chaser < 0) {
      chaser = NUM_LEDS - 1;
    }
    delay(200);

    // --------------------------
    //  Twinkle
  } else if (lightProgram == 5 ) {
    for ( int i = 0; i < NUM_LEDS; i++) {
      led[i] = CRGB(0, 0, 0);
    }
    led[random(0, NUM_LEDS)] = CRGB(random(0, 255), random(0, 255), random(0, 255));
    dirty = true;
    delay(300);

    // --------------------------
    // All on
  } else if (lightProgram == 6) {
    for ( int i = 0; i < NUM_LEDS; i++) {
      led[i] = CRGB(255, 255, 255);
    }
    dirty = true;

    // --------------------------
    // All off
  } else if (lightProgram == 7) {
    for ( int i = 0; i < NUM_LEDS; i++) {
      //led[i] = CRGB(0, 0, 0);
      led[i] = 0x0000;
    }
    dirty = true;

    // --------------------------
    // Rainbow
  } else if (lightProgram == 8) {
    int j = NUM_LEDS-1;
    for (int i = rbStart; i < NUM_LEDS; i++)  {
      //Serial.println(i);
      led[j--] = rainbow[i + rbStart];
    }
    int t = rainbow[6];
    for (int i = 6; i > 0; i--) {
      //Serial.println(i);
```

```
      rainbow[i] = rainbow[i - 1];
    }
    rainbow[0] = t;
    dirty = true;
    delay(500);
  }

  // Refresh the leds if anything has changed
  if (dirty) {
    FastLED.show();
    dirty = false;
  }

  // Read button and debounce
  int reading = digitalRead(selectButton);
  if (reading != lastSelectButtonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != selectButtonState) {
      selectButtonState = reading;
      if (selectButtonState == HIGH) {
        //Serial.println("High");
        lightProgram++;
        if (lightProgram > maxProgram) {
          lightProgram = 0;
        }
      }
    }
  }
  lastSelectButtonState = reading;
}
```