WiFi with the ESP8266 and Arduino

Installation

See http://weigu.lu//microcontroller/tips_tricks/wemos_tips_tricks/index.html

WiFi scan

This little program scans for available networks and prints:
number of networks
SSID for each network.

```
#include <ESP8266WiFi.h>

void setup() {
   WiFi.mode(WIFI_STA);         //Station mode
   Serial.begin(115200);
}

void loop() {
  int n = WiFi.scanNetworks();  // n = number of networks
  Serial.println(n);

  // print available ssids
  for (int i=0; i<n; ++i){
    String ssid = WiFi.SSID(i);
    Serial.println(ssid);
  }

  Serial.println();
  delay(1000);
}
```

You can get some more information by using these functions:

```
String ssid = WiFi.SSID(i);         // ssid = identifier
long RSSI = WiFi.RSSI(i);           // signal strength in dB
byte encryption = WiFi.encryptionType(i);
```

More info on these functions can be found here:
https://www.arduino.cc/en/Reference/WiFi

An example is found here:
staff.ltam.lu/feljc/electronics/arduino/8266/WiFiScan_moreinfo.ino/WiFiScan_moreinfo.ino.ino

Connect to a WiFi network

This sketch connects to a known WiFi network and displays the WEMOS IP address:

```
#include <ESP8266WiFi.h>
const char* ssid = "*********";                 // your SSID
const char* password = "********";              // your password

void setup() {
   Serial.begin(115200);
   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED)  {
        delay(500);
        Serial.print(".");
        }

   Serial.println();
   Serial.println ("WiFi connected");
   Serial.println (WiFi.localIP());

}

void loop() {
}
```

Don't forget to change SSID and password to the correct settings.

To test the connection, you can ping the IP address that is displayed, and see the reply from the WEMOS IP address.

Create an access point

An access point can be reached from a mobile phone or a tablet, even if there is no WiFi or Internet connection.
This basic sketch is from Arduino ESP8266 examples.
It creates an access point that can be reached by typing the access point IP address into a browser.
Here the default address 192.168.4.1 is used.
Then, a simple text message is sent to the browser.

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

const char* ssid = "myAccesspoint";
const char* password = "myAccesspoint";         // choose a better pwd!

ESP8266WebServer server(80);                    // port 80

// react to browser call
void handleRoot() {
      server.send(200, "text/html", "<h1>You are connected</h1>");
}

void setup() {
```

```
      delay(1000);
      Serial.begin(115200);
      Serial.println();
      Serial.print("Configuring access point...");
      /* You can remove the password parameter if you want the AP to be open. */
      WiFi.softAP(ssid, password);

      IPAddress myIP = WiFi.softAPIP();
      Serial.print("AP IP address: ");
      Serial.println(myIP);
      server.on("/", handleRoot);
      server.begin();
      Serial.println("HTTP server started");
}

void loop() {
      server.handleClient();
}
```

Remarks:

- In this example, the network established by softAP has the default IP address of **192.168.4.1**. This address may be changed using `softAPConfig`

- The password can be omitted if you want an open access point.
  In this case, use **`WiFi.softAP(ssid);`**

- Further documentation is here:
  https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/soft-access-point-class.html

Switch a LED via WiFi, the very basics

The access point sketch can be extended so that an action can be taken via WiFi.
In this example a LED is switched on and off, and a message is sent back describing the action.
Therefore two functions are declared that switch the LED:

```
void ledon(){
  digitalWrite(led, 1);
  server.send(200, "text/html", "<h1>LED ON</h1>");
}

void ledoff(){
  digitalWrite(led, 0);
  server.send(200, "text/html", "<h1>LED OFF</h1>");
}
```

(Naturally, the LED pin has to be declared and set as output:

```
…
#define led 15        // D8
…
pinMode(led, OUTPUT);
...
```

)

In the setup function, the server is told what to do to call these functions:

```
…
server.on("/", handleRoot);
server.on("/ledon", ledon);
server.on("/ledoff", ledoff);
server.begin();
...
```

Now the LED can be switched on and off by typing

```
192.168.4.1/ledon       or
192.168.4.1/ledoff
```

in a browser connected to the accesspoint.

The pinout of the other pins is found here:
https://github.com/esp8266/Arduino/blob/master/variants/d1_mini/pins_arduino.h#L49-L61

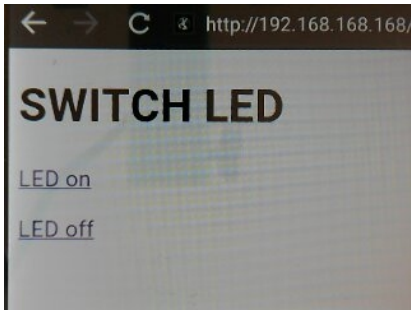
```
#define PIN_WIRE_SDA (4)
#define PIN_WIRE_SCL (5)

static const uint8_t SDA = PIN_WIRE_SDA;
static const uint8_t SCL = PIN_WIRE_SCL;

#define LED_BUILTIN 2

static const uint8_t D0 = 16;
static const uint8_t D1 = 5;
static const uint8_t D2 = 4;
static const uint8_t D3 = 0;
static const uint8_t D4 = 2;
static const uint8_t D5 = 14;
static const uint8_t D6 = 12;
static const uint8_t D7 = 13;
static const uint8_t D8 = 15;
static const uint8_t RX = 3;
static const uint8_t TX = 1;
```

Switch a LED via WiFi, more comfortably

The basic sketch of the priveous chapter allows switching on and off a LED or anything, or doing more complicated things, but it lacks a decent user interface in the browser.

To do this is not very difficult. The ESP8266 has to send a webpage via WiFi, with clickable links to switch the LED on and off.

The simple webpage looks like this on my tablet:



For a better overview, the HTML code is put into a separate tab „html.h“ in the Arduino IDE. This means that it exists as a separate file „html.h“ in the project folder, that must be loaded at the beginning of the sketch with

```
#include "html.h"
```

Note the difference to the other include statements where '<' and '>' are used to include the file name (this means that the files are system files, whereas our include statement loads a local file in the same folder).

The file html.h contains the HTML code for our webpage, that is the user interface:

```
String html = R"***(
<!DOCTYPE html>

<html>
  <head>
    <title>SWITCH LED</title>
  </head>

  <body>
    <h1>SWITCH LED</h1>
    <p><a href=/ledon>LED on</a></p>
    <p><a href=/ledoff>LED off</a></p>

  </body>
</html>
)***";
```

Here a C++ raw string notation is used. A raw string has an attribute R before the " and is enclosed by parenthesis and some special characters that are unique.
So another possibility would be for example:

```
String html = R"====(
…
</html>
)====";
```

The text between the parenthesis must be entered as you would in any text editor.

The above webpage defines two links for Led on and LED off (and a heading).

The functions for handling root (which means the reaction when the IP address is called), ledon and ledoff are defined in slightly changed functions:

```
void handleRoot() {
      server.send(200, "text/html", html);
}

void ledon(){
  digitalWrite(led, 1);
  server.send(200, "text/html", html);
}

void ledoff(){
  digitalWrite(led, 0);
  server.send(200, "text/html", html);
}
```

The difference to the functions of the previous example is that as a reaction, they always send the same html code of the whole webpage.

Tip:
Eventually it is a good idea to put the handling functions into an extra tab, to have a better overview over the code. Don't forget the include statement for this!

You might encounter an error message saying that the variable server is not declared.
This happens if the include statement is set at the very beginning (as it might seem logical and fine). But our functions use the server variable that is only available after it is set up. So the include must be placed after this:

```
ESP8266WebServer server(80);
# include "functions.h"
```

The full code of this example is found here:

http://staff.ltam.lu/feljc/electronics/arduino/8266/WiFiAccessPoint_led_webpage_2.zip

Switch a LED with buttons on the webpage

A more beautiful webpage can be obtained by creating buttons via CSS:

This is easily obtained by just changing the HTML string:

```
String html = R"=====(
<!DOCTYPE html>

<html>
  <head>
    <title>creative-lab.lu</title>
  </head>

  <style>
    body      {font-family: Arial, Verdana, sans-serif; font-size: 12px; color:
red; background-color: green;}
    div       {margin: 0.2em auto; width: 100%; clear: both;}
    p         {margin: 0.2em;  padding: 0.1em;  border-radius: 1em; background-
color: yellow;
               border: 0.3em red solid; font-size: 4em; font-weight: bolder;
text-align: center;}
    p.center  {margin: 0.2em  auto; width: 27%; clear: both;}
    p.heading     {clear: both;}
    p.button   {background-color: orange; width: 27%; float: left;}


    a         {display: block; text-decoration: none;}
  </style>

  <body>
    <br><p class="heading">creative-lab.lu
          <br/>Lyc&eacute;e Technique des Arts et M&eacute;tiers</p><br>
    <div>
    <p class="button"><a href=/ledon>LED on</a></p>
    </div><div>
    <p class="button"><a href=/ledoff>LED off</a></p>
    </div>

  </body>
</html>
)=====";
```