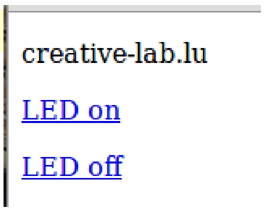
Snyder2 – WEMOS firmware

First test: switch a LED

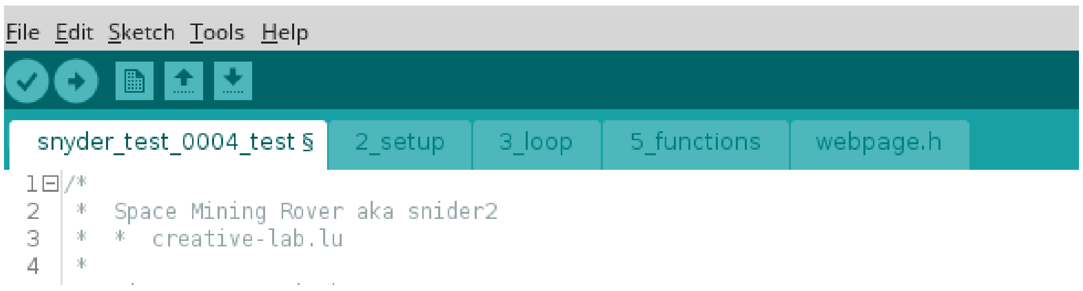

creative-lab.lu

LED on

LED off

The program just switches a LED on and off, as a first communication test.

The Arduino program has different tabs to make it easyer to overview:



```
/*
 *  Space Mining Rover aka snider2
 *  *  creative-lab.lu
 *
```

Main tab (snyder_xxxx):

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


const char *ssid = "snyder2";
const char *password = "snyder2";

IPAddress IP_AP(192, 168, 168, 168);
IPAddress MASK_AP(255, 255, 255, 0);

ESP8266WebServer server(80);      // create "server" object with port 80
#include "webpage.h"
```

The main tab creates the web server object that creates a local WiFi hotspot with

- IP address 192.168.168.168
- ssid / password = „snyder2 / snyder2“
- communication port 80 (this is the standard port)

The include file **webpage.h** defines the robot webpage as global string html:

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

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

  <body>
```

```
          <p>creative-lab.lu</p>
          <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.

Setup

```
void setup() {

  pinMode(LED_BUILTIN, OUTPUT);    // blue LED Pin (D4) as output
  digitalWrite(LED_BUILTIN, LOW); // on (negative logic)

  WiFi.softAP(ssid, password);
  WiFi.softAPConfig(IP_AP, IP_AP, MASK_AP);
  IPAddress myIP = WiFi.softAPIP();

  server.on("/", handleRoot);      // don't forget the "/"
  server.on("/ledon", ledon);
  server.on("/ledoff", ledoff);

  server.begin();
  server.setContentLength(html.length());
```

The setupfunction sets up the LED output and initializes the WiFi server.
There are three reactions to the incoming requests that are programmed.
These are handled by the functions handleRoot(), ledon(), ledoff() defined in the functions tab..

Loop

The loop function does nothing but wait for requests and handle them:

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

There are three functions defined in the functions tab:

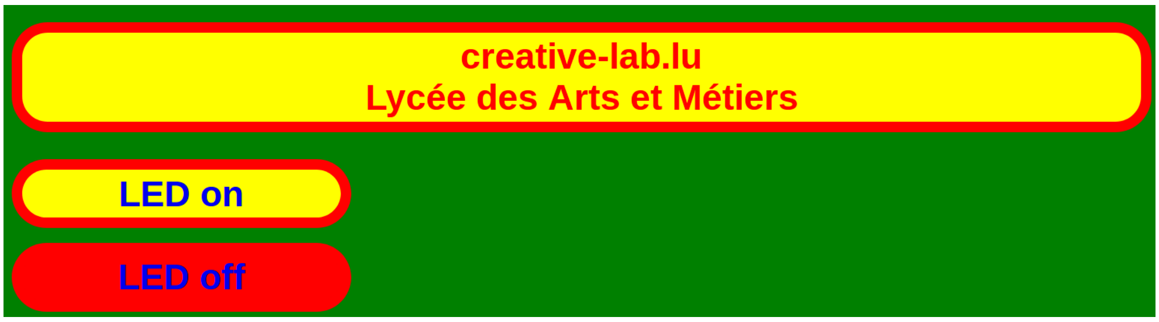
```
void handleRoot() {
  server.send(200,"text/html",html);
  }

void ledon() {
  digitalWrite(LED_BUILTIN, 0);
  server.send(200, "text/html",html);
  }

void ledoff() {
  digitalWrite(LED_BUILTIN, 1);
  server.send(200,"text/html",html);
  }
```

These correspond to the responses when you type
192.168.168.168/
192.168.168.168/ledon
192.168.168.168/ledoff
in a browser

or if you call 192.168.168.168/ and press the ledon and ledoff links defined on the simple webpage.

(the 192.168.168.168/ IP address is an example address defined at the beginning of the program)

Beautify the webpage with CSS



It is easy to switch between the simple and a more beautiful web page when a new tab is defined containing this:

```
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.lam      {clear: both;}
    p.main     {background-color: orange;
                width: 27%;
                float: left;}
    p.side     {width: 27%;
                float: left;}
    p.stop     {background-color: red;
                width: 27%;
                float: left;
                margin-bottom: 0.4em;}
    a          {display: block;
                text-decoration: none;}
  </style>

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

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

The webpage file should have the extension .h like the other one. To switch between both, only the include line in the main tab must be changed.

Motor driver functions

The motor driver needs an ENABLE and a DIRECTION signal for both motors.
The ENABLE signal may be a PWM signal, so that different voltages and so different velocities can be set for the motors. The PWM signal is generated by the analogWrite command.
The direction signal switches between left and right turn of the motor.

As example, here is the function to turn right:

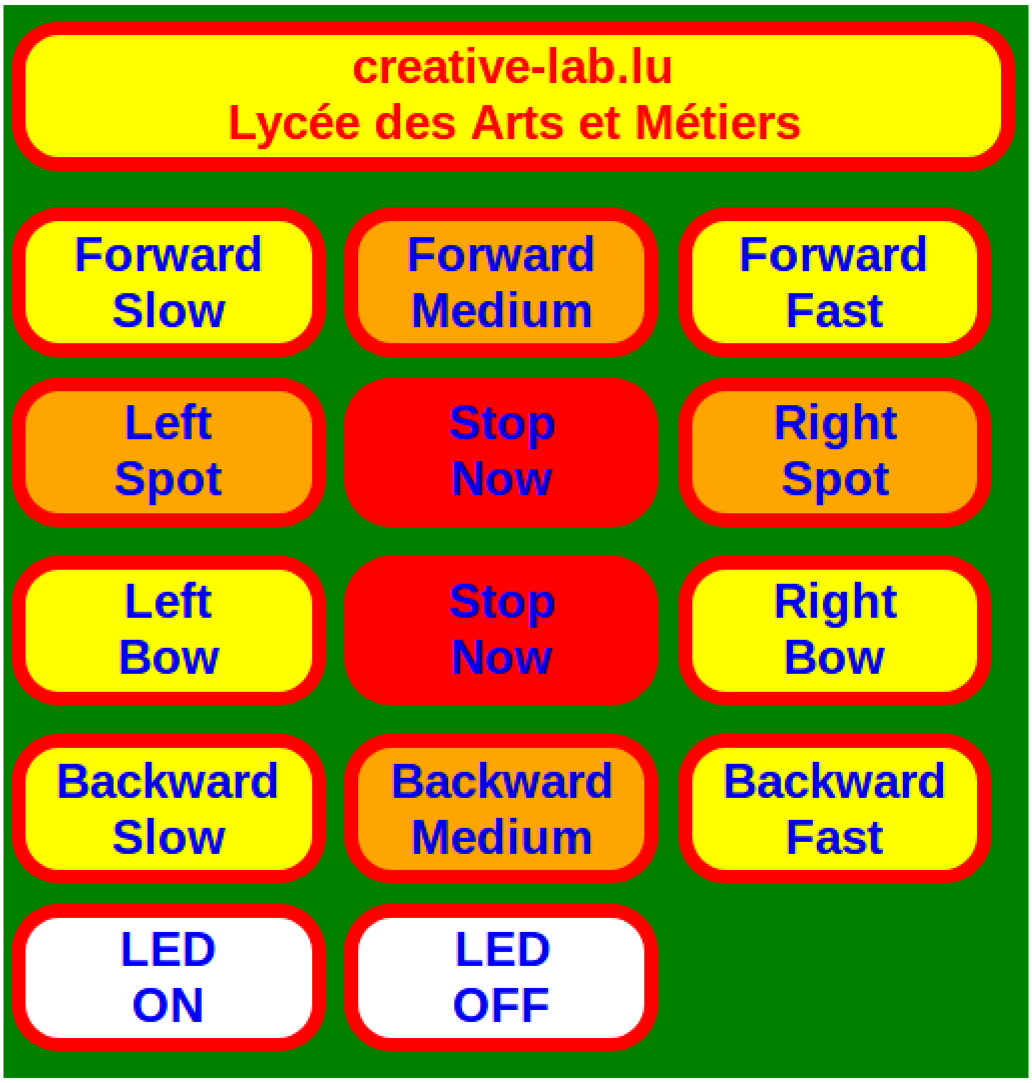
```
void right(int Motor_R_Speed, int Motor_L_Speed){
         analogWrite(Motor_R_ENA, Motor_R_Speed);
         digitalWrite(Motor_R_DIR, 0);
         analogWrite(Motor_L_ENA, Motor_L_Speed);
         digitalWrite(Motor_L_DIR, 1);
         }
```

This function is connected to the HTML tag by this line in the setup funtion:

```
    server.on("/rights", rights);
```

The other functions are defined in the same way.
The HTML string is extended to define buttons for the other functions.

The webpage looks like this now:



A test in the garden showed that Snyder2 could be commanded by tablet.
https://www.facebook.com/creativelab.lu/videos/1508062155938617/