



PHOTONICS WORKSHOP SMARTPHONE CONTROLLED RGB LAMP

DISCLAIMER:

By using this information you agree to be legally bound by these terms, which shall take effect immediately on your first use of the information.

PHABLABS 4.0 consortium and its member organizations give no warranty that the provided information is accurate, up-to-date or complete. You are responsible for independently verifying the information. VUB cannot be held liable for any loss or damage that may arise directly or indirectly from the use of or reliance on the information and/or products provided. PHABLABS 4.0 consortium and its member organizations disclaim all responsibility to the maximum extent possible under applicable laws:

- All express or implied warranties in relation to the information and your use of it are excluded.
- All liability, including for negligence, to you arising directly or indirectly in connection with the information or from your use of it is excluded.

This instruction is published under the Creative Commons licence CC-BY-NC.

PROPERTIES OF THIS WORKSHOP

SUMMARY:



In this workshop, we are going to build an RGB lamp, which can be remote controlled by your smartphone. The lamp is centered on the Arduino-like platform WeMos, which has WLAN capability and is set up as an Access Point (AP) for your smartphone. The WeMos also has a programmable microcontroller with several inputs and outputs, which control 3 red, green and blue high-power-LEDs. The provided sketch (program code) can be manipulated using a freely available programming environment for future projects.



TARGET AUDIENCE:

Students (15-18 years old)

SUGGESTED TIME PLANNING: (Total: 2,5h)



Timing in minutes	activity
0-20	Welcome group and explaining of the concept of 'colours' and 'basics of electronics'.
20-60	Controlling and implementing electronics.
60-90	Programming the Wemos.
90-110	Checking the lamp.
110-150	Assembling the box.



TOOLS:

Laser Cutter
Soldering Iron
Computer for programming Wemos



WEBLINK:

All needed files for lasercutting and Wemos can be found on:
<http://www.phablabs.eu/workshop/wi-fi-controlled-led-color-mixing>
or via the QR code.

Step 1: Parts list

Collect all materials for each participant.

Photonics Parts:



Red, green & blue 1W LED on starboard

1 piece of each

'LED' stands for Light Emitting diode and generates light of certain color. A diode is a one-way-street for electrical current. When you turn around the diode, the current is blocked.



Lenses

3 pieces

The lenses we use concentrate the light provided by each LED. They are simply attached to the LED.

Not included:
breadboard jumpers and wire

Electronic Parts:



Wemos

1 piece

A microcontroller, which can be programmed as easily as an Arduino, will control the LEDs according to the uploaded 'sketch' (a term for program coined in the Arduino community).



ULN2003A

1 piece

This integrated circuit (IC) is a 'driver' for the LEDs: the microcontroller does not provide enough current for the power-hungry LEDs, so we need an amplifier, which is controlled by a small current and provides a large current. That's what the ULN2003A does.



Breadboard 170 pins & breadboard 55 pins

1 piece of each



Resistor 3.3, 4.7 & 5.6 Ohm

Resistors limit the current in an electrical circuit. Here we need them for protecting the LEDs. Each LED-color needs a different resistor. A resistor has a set of colored rings on it which denote its electrical resistance; the larger its resistance, the more it limits the current.

A complete toolkit is available on the www.phablab.eu website.

Or via email: phablab4.0@gmail.com

Step 2: Colours

Light is an electromagnetic wave, which moves through space at a speed of about $c=300,000$ kilometers per second, which is a little more than $1,000,000,000$ km/hour. Like every other wave (sound, water, ...), it oscillates while it propagates, creating crests and troughs, see Figure 1. The distance between two successive crests is called the wavelength. Light oscillates very quickly, at a frequency of about $f=600,000,000,000,000$ times per second (600 THz) for green light, which results in a wavelength of $=c/f= 500\text{nm}$.

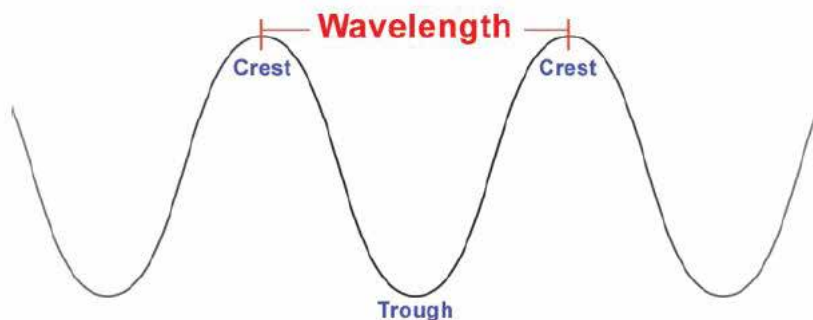


Figure 1: A wave of a given wavelength.

There is a wide range of electromagnetic waves, which differ only by their wavelengths. Figure 2 shows the most important ones, ranging from radio waves with a wavelength of meters, over microwaves, light, X-rays and Gamma rays, which are emitted in radioactive processes. The 'visible spectrum', which contains all colors we see, is only a tiny part of the existing electromagnetic spectrum. Its wavelengths range from 700nm (red) to 400nm (violet), with all rainbow colors in between.

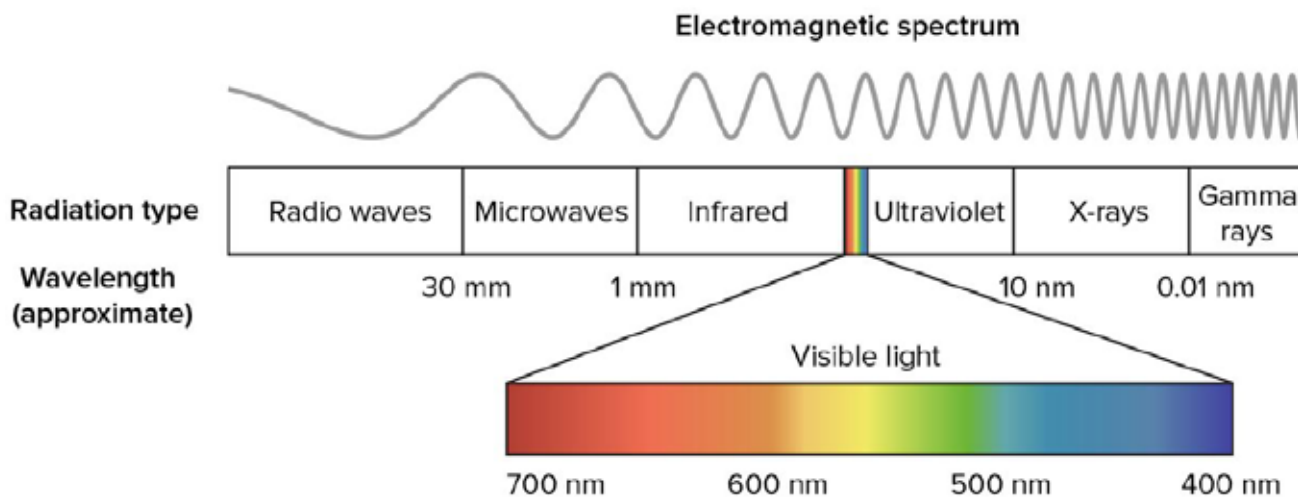


Figure 2: Electromagnetic spectrum

Our eyes have 3 types of receptor, for red, green and blue. All other colors are mixing products in the eye. For example, yellow light does not have a special receptor in the human eye. But, its wavelength is between red and green and excites the red and the green receptors about the same as equal amounts of red and green do, see Figure 2. It also means that you can create yellow light by mixing red and green light! The human eye cannot see any difference between "true" yellow light and a suitable combination of red and green (or other colored) light. There are some colors which are not present in the spectrum in Figure 2, because they do not have their own unique wavelength. They can only be created from mixing colors. Those are, for example, magenta, which is similar to pink (a mixture of red and blue), white (a mixture of red, blue and green), and so on. Black, on the other hand, is just the absence of light.

Figure 3 shows the color mixing of red, green and blue. Because our eyes have red, green and blue receptors, a computer screen or smartphone display also use red, green and blue light to create almost every possible color. You can see their individual color pixels when you look very closely.



Figure 3: Color mixing of red, green and blue

Step 2: Electronic basics

Voltage and Current

It is useful to compare an electrical circuit to something we have a better understanding of, like a water pipe system, see figure 4. A battery or solar cell like a water pump; they generate a pressure, which corresponds to the voltage. When you connect a pipe with one end to a water reservoir and with the other to a water pump, the pump generates pressure (voltage) and water current. The water current corresponds to an electrical current. The higher the pump pressure (voltage), the larger the resulting water flow (current). You can limit the water flow (current) by putting something in its way (a ball of hair in a clogged up shower drain) or make the tube thinner. Such a restriction corresponds to an electrical resistor.

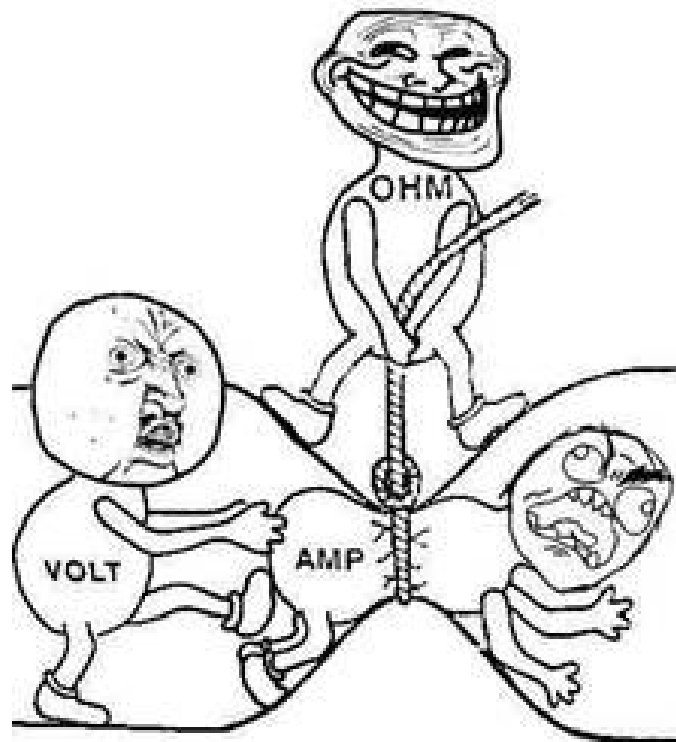
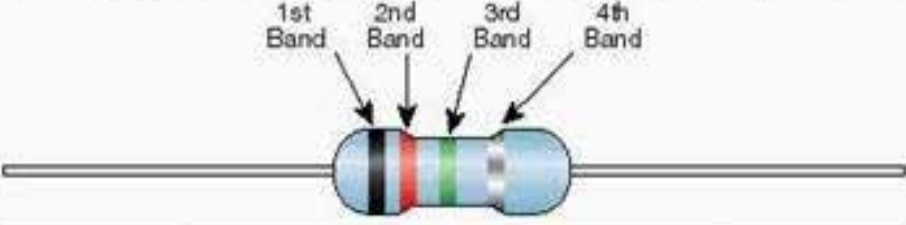


Figure 4: Another way to memorize the relation between voltage (VOLT), current (AMP) and resistance (OHM).

Protecting LEDs

LEDs get destroyed when the current becomes too high. When we operate them with a given voltage, each LED needs their individual resistor to limit their current. Red LEDs need the least voltage and therefore the largest protective resistor. Resistors usually have colored rings which denote their value. Here is the encryption key:

Standard EIA Color Code Table 4 Band: $\pm 2\%$, $\pm 5\%$, and $\pm 10\%$



Color	1st Band (1st figure)	2nd Band (2nd figure)	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	10^0	
Brown	1	1	10^1	
Red	2	2	10^2	$\pm 2\%$
Orange	3	3	10^3	
Yellow	4	4	10^4	
Green	5	5	10^5	
Blue	6	6	10^6	
Violet	7	7	10^7	
Gray	8	8	10^8	
White	9	9	10^9	
Gold			10^{-1}	$\pm 5\%$
Silver			10^{-2}	$\pm 10\%$

When we connect the LEDs to the driver ULN2003, we need to take into account the driver's inner resistor, which is like resistor already present in the IC. This means, that we need, in fact, smaller resistors than anticipated, which are best determined experimentally.

Here are the values we found:

LED	Resistor	Color ring code
Red	5,6 Ohms	green-blue-gold-gold
Green	3,3 Ohms	orange-orange-gold-gold
Blue	4,7 Ohms	yellow-violet-gold-gold

Depending on the LEDs you use, those values might differ. But don't worry; it's safe to operate your high-power-LEDs with the values given.

Step 3: The microcontroller

This project uses a WeMos to control 3 high-power LEDs, but a NodeMCU, an ESP8266 or similar will do as well. To keep the confusion at a reasonable level, we describe everything in terms of the WeMos. When using a different platform, the program environment needs to be adapted accordingly.

The WeMos is powered and programmed through a USB connection to a computer. Once it is programmed, it can be powered by phone charger.

The WeMos (Figure 6) is an Arduino based platform which can be directly programmed using a desktop or laptop and which can perform a large variety of tasks. Today, there are many variations of the Arduino platform coming in all sizes and peripherals. In our project, we choose the WeMos, which is a tiny, cheap and very powerful platform. There are many free Arduino tutorials available on the internet, which are perfect for any stage of advancement. In this tutorial, we therefore limit ourselves to those things necessary for building our WeMos controlled RGB-Lamp.

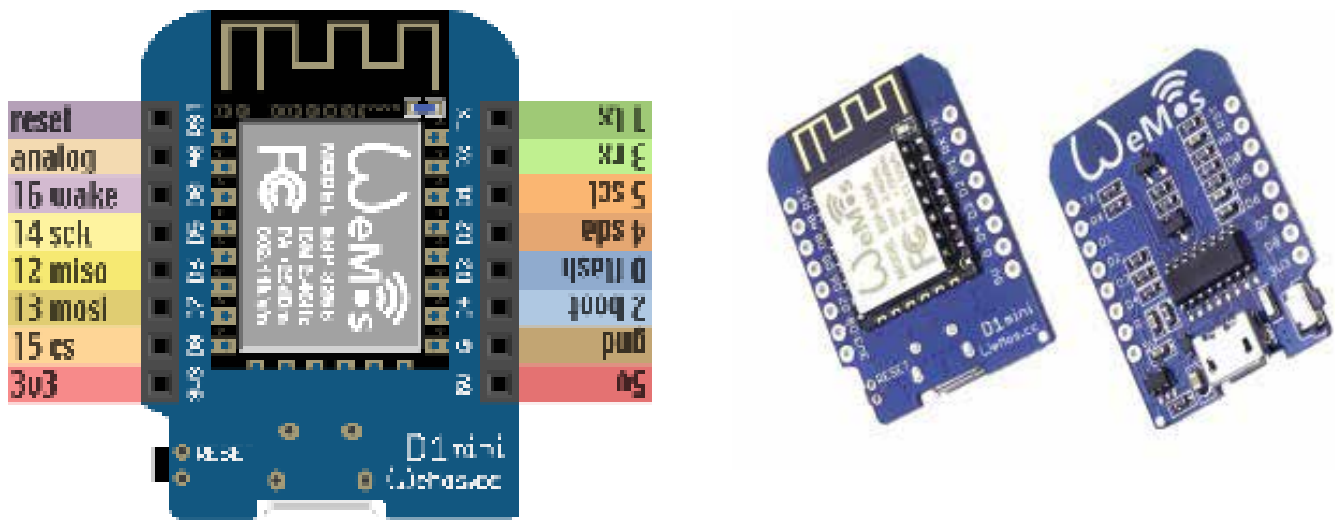
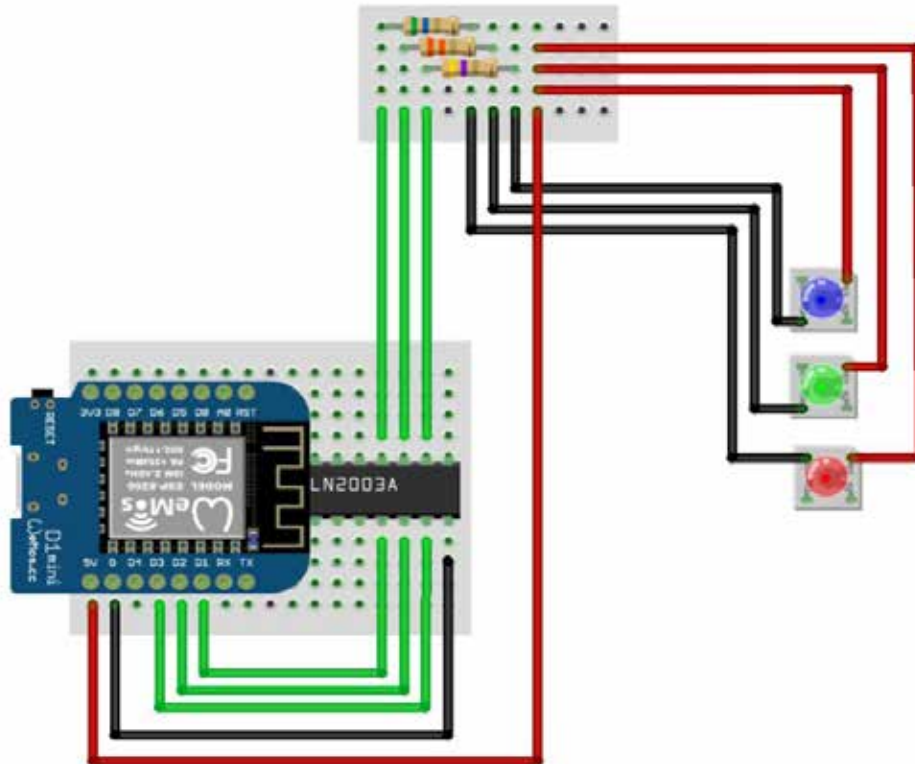


Figure 6: A Wemos controller.

The metallic box is a shield for electromagnetic waves and contains the microcontroller and the WLAN transceiver, which allows the WeMos to wirelessly communicate. In the WeMos Lite, this shield is missing, which does not seem to pose a problem though. The curly printed wire outside this box on the left is its antenna. The WeMos is powered through the USB connector on the bottom, which runs on 5 volts. Since the WeMos can only take 3.3 volts, there are additional components on the bottom of the board which provide this voltage.

Step 4: Periphery



Wiring diagram of our RGB lamp

The WeMos switches its outputs D1, D2 and D3 on and off, executing the commands received from your smartphone. Those signals are fed to the driver IC ULN2003, where they are amplified. The ULN2003 powers the red, green and blue LEDs via protective resistors, without which the LEDs would burn up. All current circuits must be closed, so the ground (negative terminal) goes to the ULN2003, and the 5V positive terminal goes to each LED.

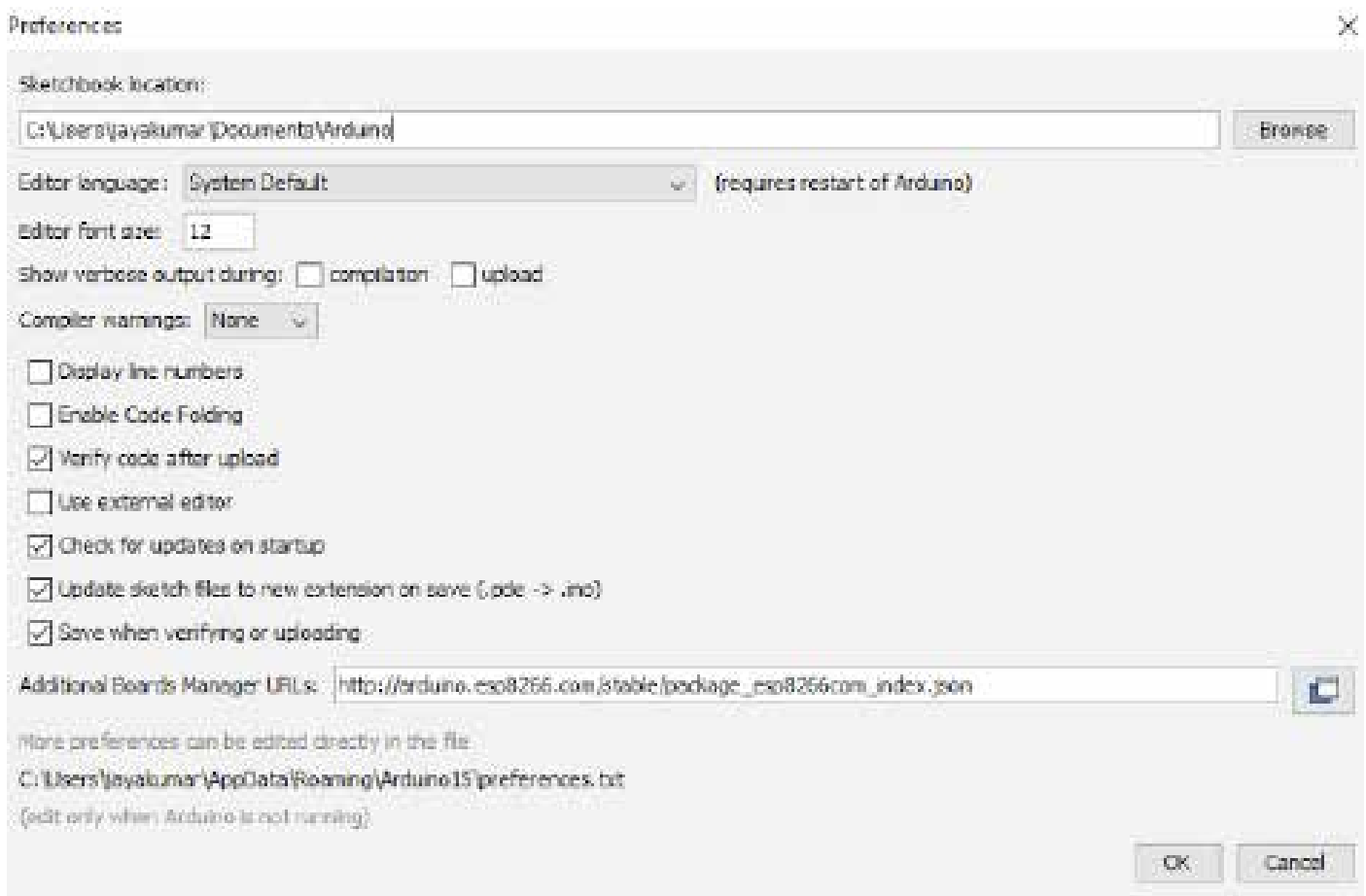
Step 5: Quick start guide for the WeMos

For programming and controlling the Wemos, we use the Arduino IDE (Integrated development environment), which can be downloaded from <https://www.arduino.cc/en/Main/Software>.

After we start it, we first need to tell this IDE that we use the WeMos (or whatever else we have) so that the appropriate additional software can be downloaded.

In order to program the WeMos, we connect it to the USB port of the computer which runs the Arduino programming environment. The USB-cable provides both the 5V voltage for running the WeMos as well as the programming connection.

1. Open the Arduino IDE, go to “Files” and click on “Preferences”.



Copy the following line into the “Additional Boards Manager URLs” text box:

```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

and press OK to close the Preferences tab.

2. Select “Tools” and “Board”, and click on “Boards Manager...” in the pop-up menu.

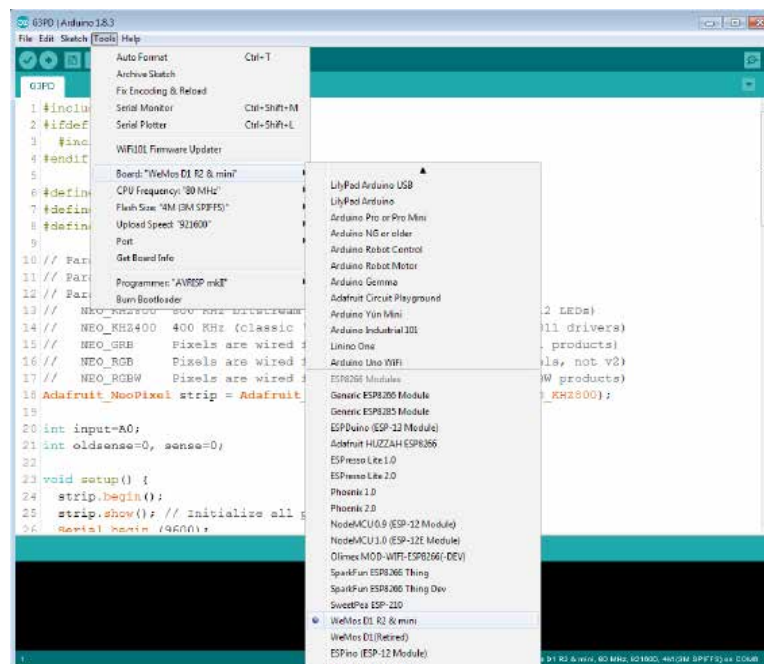
Navigate to “esp8266 by ESP8266 Community” and click into the field. This will install the programming environment we need, for the WeMos we use, or other boards included in this package as shown.



Also, include the library “Websockets” by Markus Sattler via the ‘library manager’:



Finally, click again in Tools Board and select WeMos D1 R2 & mini from the list:




Now we are ready to program our WeMos within our Arduino IDE (Integrated development environment).

Step 6: The sketch (program code)

We use the sketch with the code below, which enables the WeMos to be controlled by your smartphone. The program code is included with this workshop and can also be downloaded at <http://phablab.eu/workshop/wi-fi-controlled-led-color-mixing>

For visual clarity the code is split up in four parts, which the IDE sees as a single file as long as they are in the same directory named after the main file.



Name	Änderungsdatum	Typ	Größe
credentials.ino	03.12.2017 19:54	INO-Datei	1 KB
G2.ino	03.12.2017 19:54	INO-Datei	2 KB
handleHttp.ino	03.12.2017 19:54	INO-Datei	15 KB
tools.ino	03.12.2017 19:54	INO-Datei	1 KB

Here, the WeMos acts as an Access Point (AP), which can be directly connected to by your smartphone.

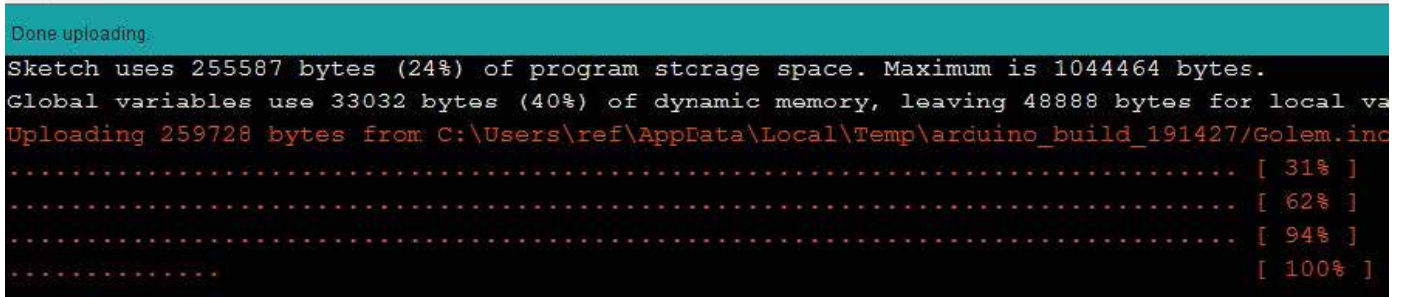
```
1 #include <ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <ESP8266WebServer.h>
4 #include <DNSServer.h>
5 #include <ESP8266mDNS.h>
6 #include <EEPROM.h>
7 #include <WebSocketsServer.h>
8 #include <Hash.h>
9
10 // Parts of this sketch are based on "ioglow" by fablab Roskilde
11 // Declaring function prototypes. Important, as preprocessor
12 // sometimes forgets to declare everything
13 void loadCredentials();
14 void saveCredentials();
15 void connectWifi();
16 void loopWifi();
17 void setupWifi();
18 void handleCSS();
19 void sendScript(String _server);
20 void sendTop(String _name);
21 void sendBottom();
22 void sendButton(String _name, String _cmd);
23 void sendSlider(String _name, int _id);
24 void handleRoot();
25 boolean captivePortal();
26 void handleWifi();
27 void handleWifiSave();
28 void handleNotFound();
29 bool handleFileRead(String path);
30 String getContentType(String filename);
31 String IPAddress2String(const IPAddress& ipAddress);
32 String getUniqueId();
33 void websocketEvent(uint8_t num, wstypet_t type, uint8_t * payload, sizet_t length);
```

```
34 boolean isIp(String str);
35 String toStringIp(IPAddress ip);
36 void usSetup();
37 float usRead();
38
39 WebSocketsServer webSocket = WebSocketsServer(81);
40
41 int sliderValues[5] ;
42 char ssid[32] = "";
43 char password[32] = "";
44 String name = "RGB-Server-";
45
46 void setup() {
47     pinMode(D1, OUTPUT);
48     pinMode(D2, OUTPUT);
49     pinMode(D3, OUTPUT);
50     delay(1000);
51     Serial.begin(115200);
52     Serial.println();
53     setupWifi();
54 }
55
56 void loop() {
57     loopWifi();
58     rgb();
59 }
60
61 void rgb() {
62     analogWrite(D1, sliderValues[0]);
63     analogWrite(D2, sliderValues[1]);
64     analogWrite(D3, sliderValues[2]);
65 }
```




Connect the WeMos to a free USB-port of your computer and find out the name of this port, using the Device Manager (COM/LPT). In the IDE, select the port with Tools Port. Make sure the right type of WeMos is selected (Tools Board ...). Compile and upload the file using the button displaying an arrow pointing right. If everything goes well, the IDE should look like this after the upload:

```
19 | digitalWrite(D3, HIGH);  
20 | delay(t);  
21 | }
```



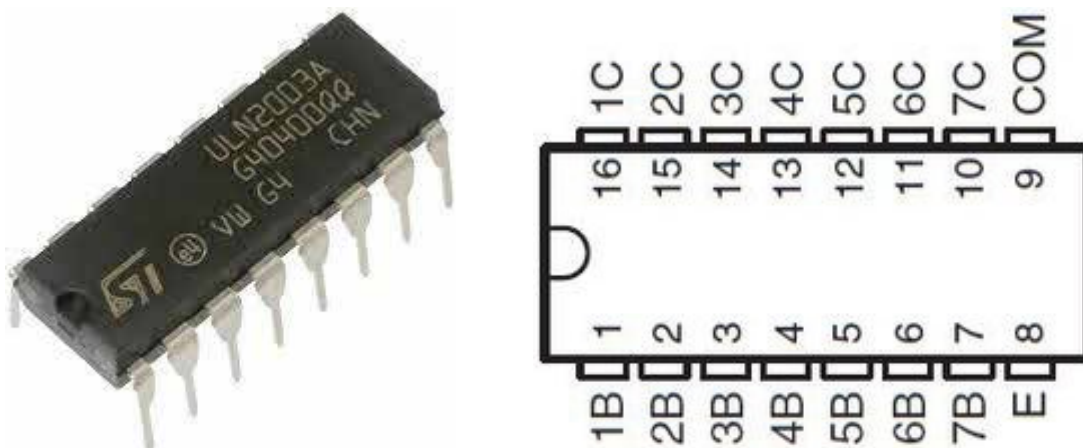
Step 7: Building the RGB-Lamp

The large breadboard for WeMos and ULN2003A



We will build the circuit on a breadboard. Breadboards come in different sizes and have one more common rail. The large breadboards we use have a total of $17 \times 10 = 170$ points. Each 10-point-column consists of 2 segments with 5 mutually connected points each.

The high-power LEDs we use can handle about 100mA without cooling. With appropriate cooling, which is not part of this workshop, they can support more current. The WeMos module cannot provide such large currents; its output terminals are limited to about 12mA. This is why we need a driver, which is basically an amplifier for the low-power outputs of the WeMos. Here, we use the universal driver IC ULN2003A, which costs only a few cents and has 7 individual drivers. It also has 16 pins. There is a notch on one of the short edges. When this notch is on the left as in Fig. 10, the lower row is defined as pins 1...8, the upper row 16...9. As seen from the diagram, the pin numbers are counted counterclockwise.



When the notch is on the left, the 7 inputs (1B...7B) are on the bottom, the outputs (1C...7C) on the top. On the right are 2 additional terminals; we only need "E", which is connected to ground.

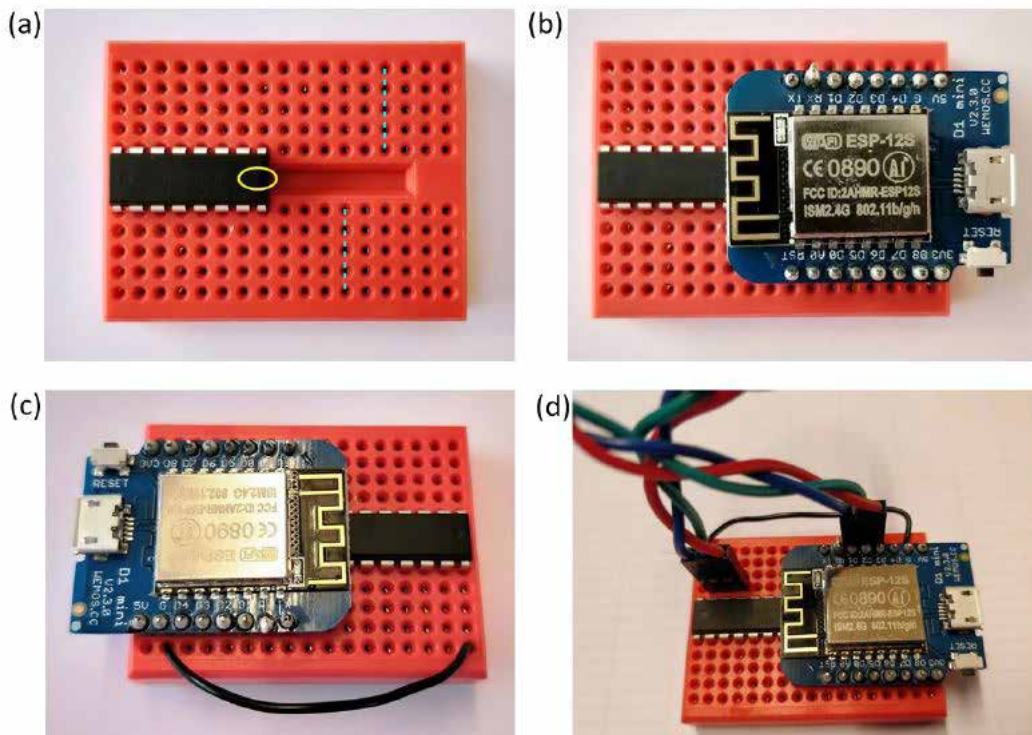
Assembling the electronics

First we solder about 12cm (4") long insulated wires to the terminals. Again, use black wires for the negative and red or white wires for the positive LED terminal.

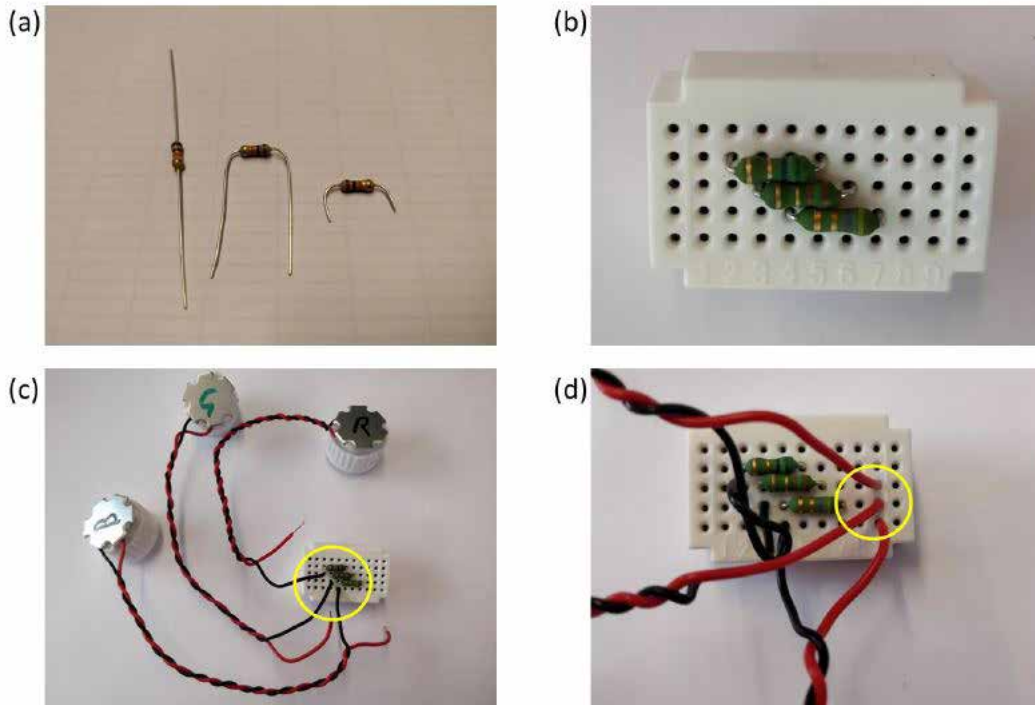


Now it's time to mount everything on those breadboards. The following figure shows the first four steps.

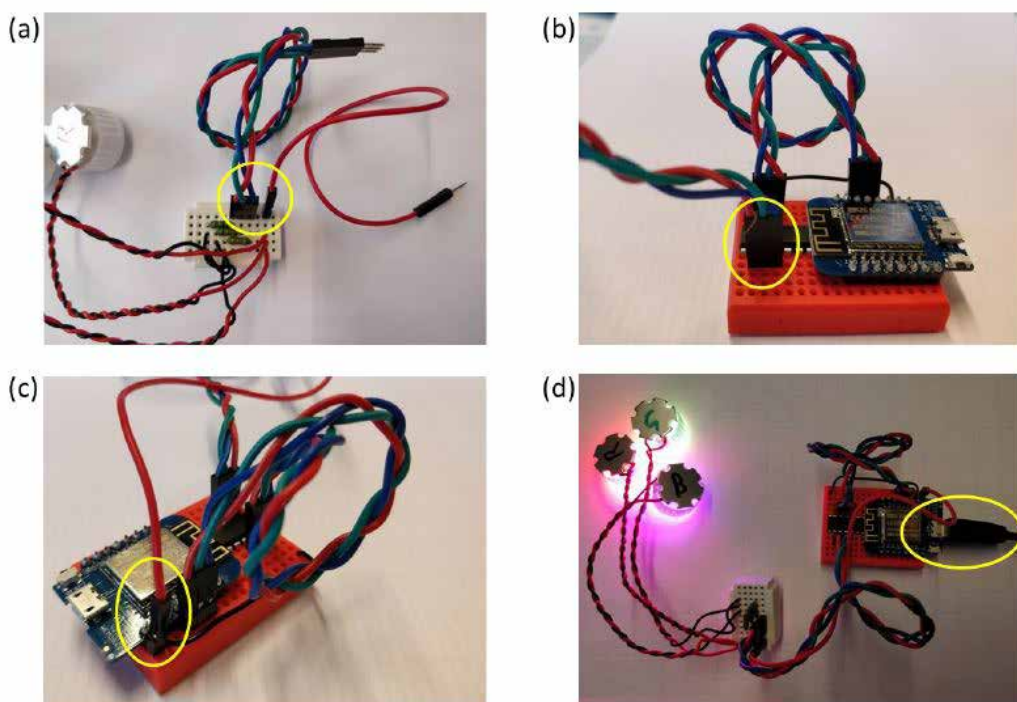
Always remember, **5 points in a row are connected**, as marked by a cyan line in (a). Insert the **ULN2003** across the bridge as shown. Make sure the U-shaped notch (marked here by a yellow ellipse) points towards the breadboard center. Then insert the **WeMos** controller symmetrically across the bridge, so that its USB port points outwards and its rightmost pins are inside the breadboard's rightmost points. (c) Connect the WeMos ground (labelled "G" or "GND") to the ULN2003 ground. (d) Connect WeMos outputs D1,D2,D3 with inputs 5,6,7 of the ULN2003.



Take the 3.3Ω (orange-orange-gold-gold), 4.7Ω (yellow-violet-gold-gold) and 5.6Ω (green-blue-gold-gold) resistors, bent over and cut their wires as shown in Figure a. Then take the small breadboard. Like for the large breadboard, 5 points in each row are mutually connected. Insert the 3 different resistors as shown. The uppermost (green-blue-gold-gold) is for the red LED, the one in the middle (orange-orange-gold-gold) for the green and the one at the bottom (yellow-violet-gold-gold) for the blue one. Connect the red, green and blue LEDs as shown in (c) and (d).



Connect the LED-board to the main board as shown in Figure 20(a,b,c). Before you proceed to (d), compare you complete setup up with Figure 7 to make sure all components are the right way around and everything is connect correctly. Then connect your WeMos to one of your computer's USB sockets (d).



Now compare your setup with the fritzing scheme one last time.

- Any short circuits or open circuits?
- Are the WeMos and the IC the right way around?
- Are the polarities of the LEDs correct?

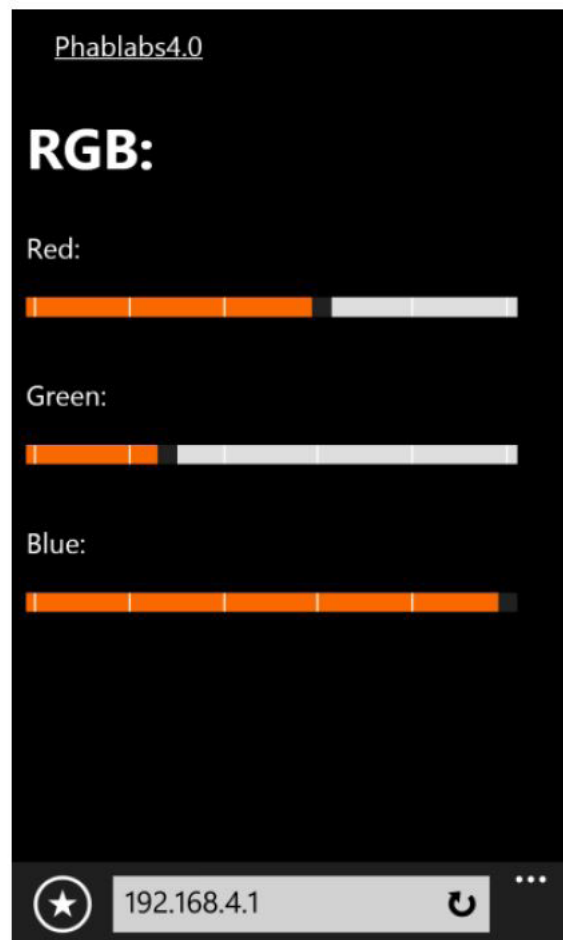
Only if everything looks fine, go ahead.

Step 8: Checking the lamp

Connect the lamp back to the USB cable and the computer. Check if any of the components gets very warm. If this is the case or you smell something funny, unplug the 5V immediately and search for the fault.

Open your smartphone (Android ok, Windows Phone ok, iPhone sometimes tricky) and scan for the WiFi APs (access points). There should be an AP named "RGB-Server-***" available.

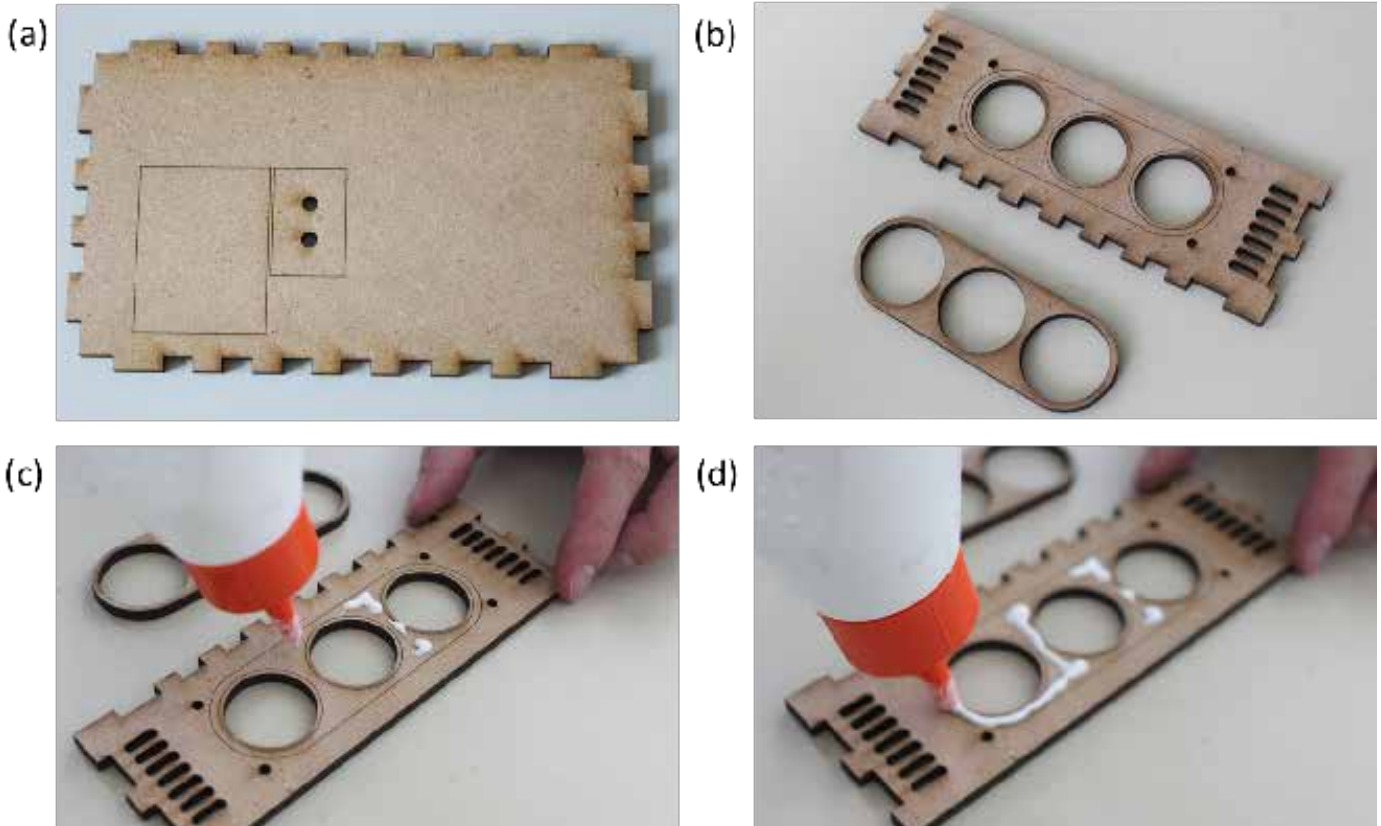
Connect to this server. A welcome screen will pop up and prompt you for input; just skip it. Open your smartphone browser and go to <http://192.168.4.1>.



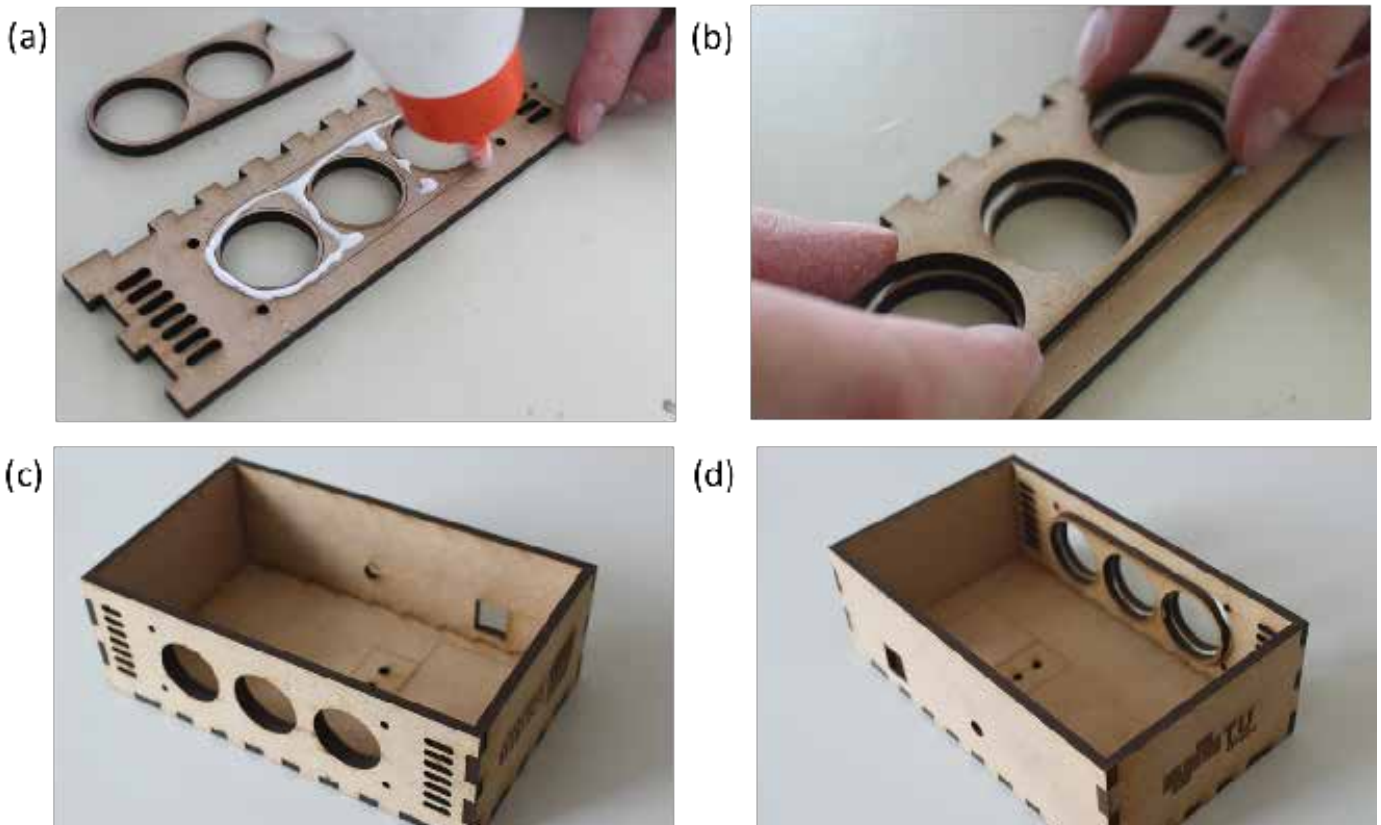
This is the website generated by the WeMos. When you tap on or move the sliders to the right, the LEDs should light up. As you will notice, you can select their light intensity with those sliders.

Step 9: Building the box

Now it's time to build a box around it. A laser cutter cuts out the parts from a sheet of wood according to the program files provided to you. You can easily push out the single element. We use wood glue on same parts as indicated. The first steps are shown in the following figures.

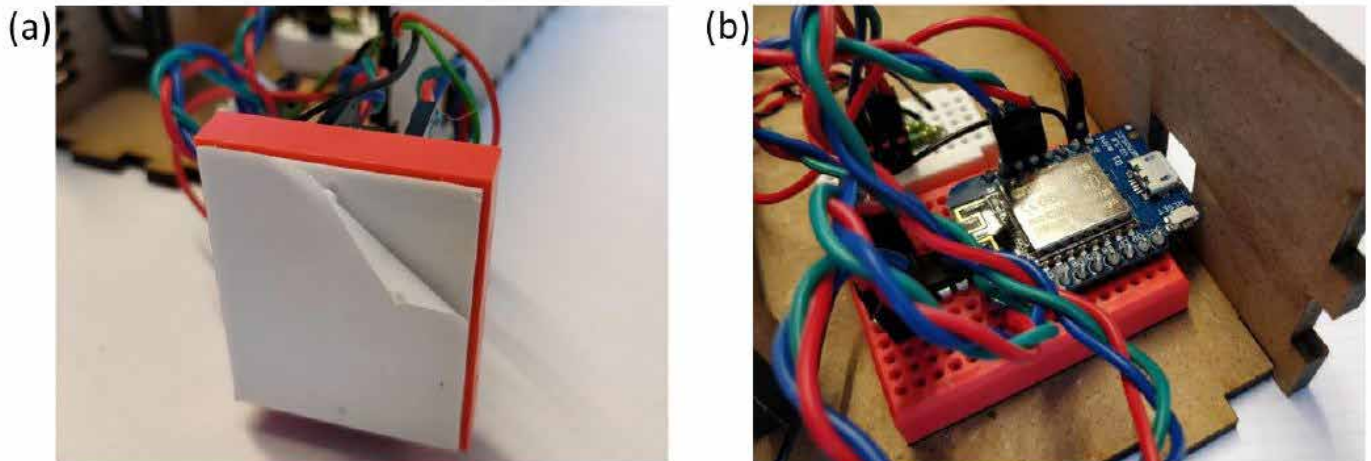


(a): Bottom plate, (b-d) Mounting the LED-plate



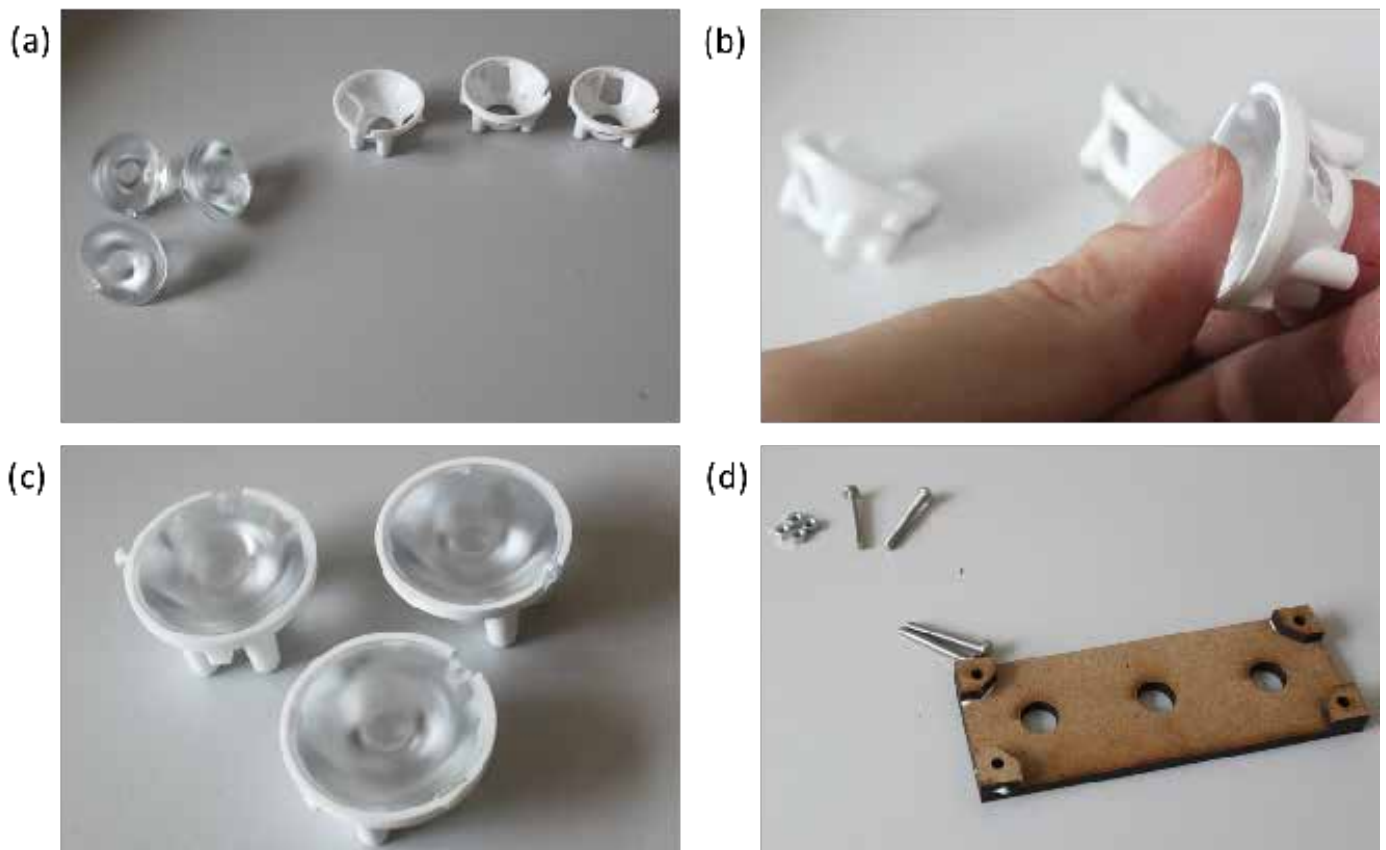
(a-b): Gluing the LED-wall, (c-d): Assembling the remaining parts (without lid yet!)

Now that the box is finished, we insert the electronics. The figure below shows the individual steps. The large breadboard has a sticky bottom, just remove the protective foil (Fig. a). Place the board into the box as (Fig. b), **make sure the WeMos USB-port aligns with the box opening** and press down until it sticks to the ground.



(a) The large board has a sticky bottom surface so it can be (b) glued to the bottom of the box, so that the USB port is accessible through the opening (check from the outside first!).

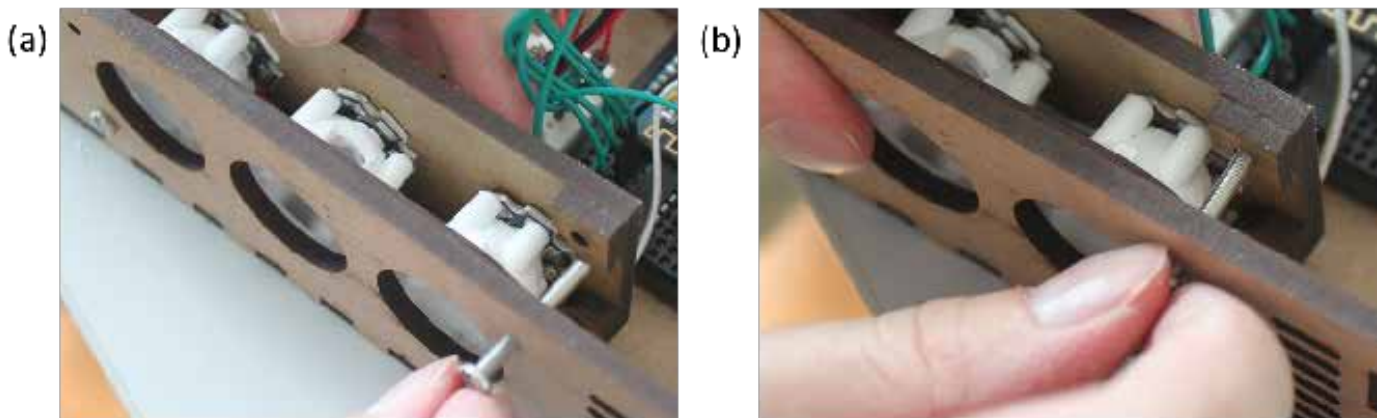
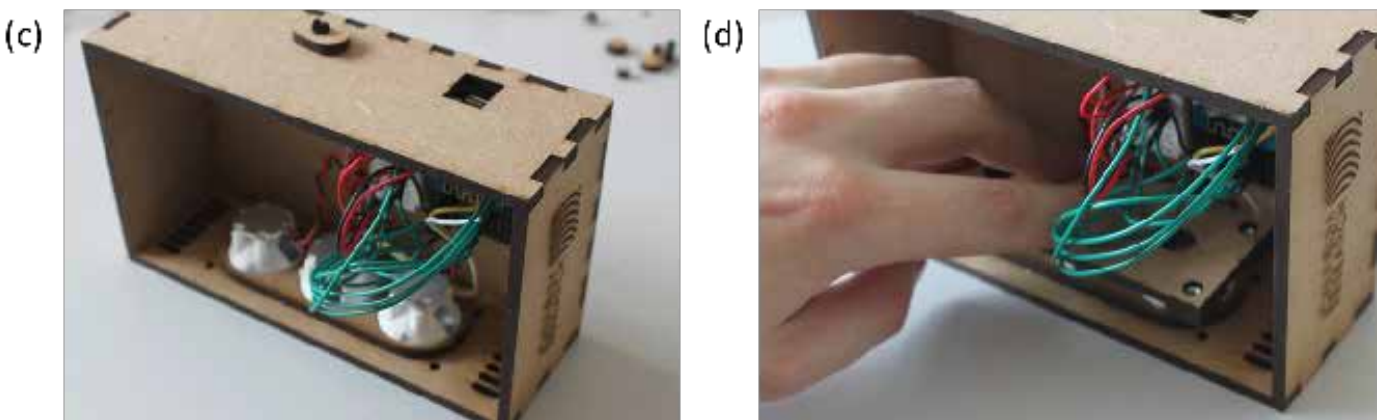
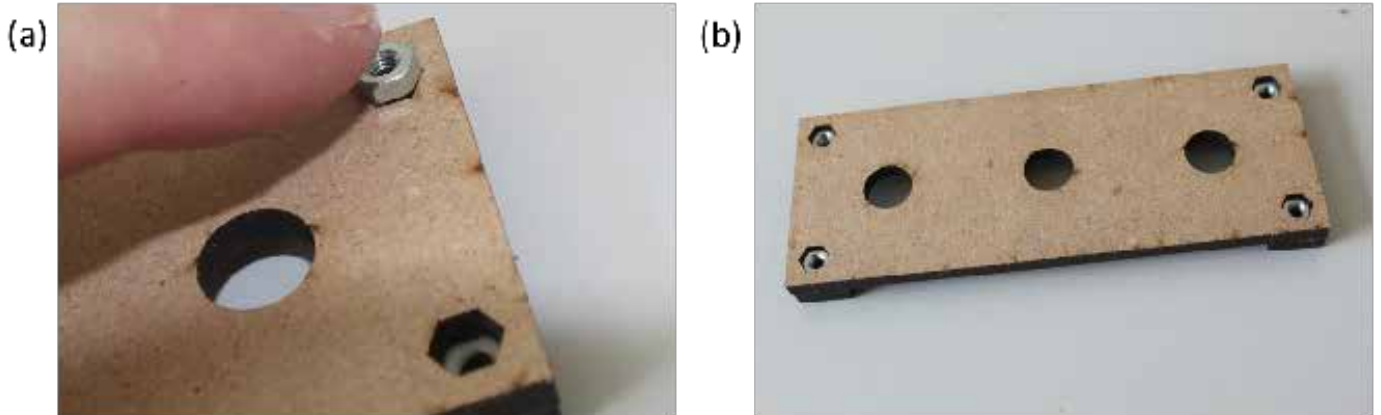
Now we install the LED and the lenses. Assemble the 3 lens kits as shown in the following figure (a-c). In order to keep those lenses safely attached to the LEDs, we prepare an auxiliary plate (Fig. d). Glue those corner pieces to the auxiliary plate as shown.



(d) Auxiliary plate for keeping lenses in place.

Insert the nuts in each corner hole as (Following fig's. a,b). Then put the boards with the screws on top of the LEDs (c,d), and carefully insert the screws. Turn those screws not too tight, just enough to keep everything in place.

Glue 4 more corner pieces to the cover lid (Fig c), and then, as soon as the glue is dry, close the box with the lid. Done! :)



Reconnect the RGB-device with the USB power supply. Use your phone app to switch on and off each LED. Is everything working? You can modify the WeMos program if you like. Make sure you save the working program first!

Last step: End results & conclusions

What have we learned?

In this workshop, we have learned how to build an inexpensive, wirelessly controllable RGB lamp with the microcontroller WeMos. We learned how to program such a device with freeware software available on the internet, and how to build the peripheral electronics to safely operate high-power-LEDs. We also got acquainted with making our own tailored wooden box around the device using laser-cutting.

The red, green and blue colors are most basic in the sense that the human eye has individual receptors for each of them and every other color we see can be reproduced as their superposition. This is why every screen, be it in a TV, smartphone or laptop, is comprised of red, green and blue pixels.

Concluding thoughts

Arduino-based systems are an inexpensive and easy way to solve all kinds of problems, that only a couple of years ago required extensive expert knowledge. Today, there is a huge Arduino community around the world who works on all kinds of open-source-projects and who are more than happy to share their knowledge and experience. You can use the introduction from this workshop to come up with and solve your own tasks.



PHABLABS 4.0

PHABLABS 4.0 is a European project where **two major trends** are combined into one powerful and ambitious innovation pathway for digitization of European industry:

On the one hand the growing awareness of **photonics** as an important innovation driver and a **key enabling technology** towards a better society, and on the other hand the **exploding network of vibrant Fab Labs** where next-generation **practical skills-based learning** using KETs is core but where photonics is currently lacking.

www.PHABLABS.eu

This workshop was set up by the *Joanneum Research* in close collaboration with *FabLab Graz*. For questions and comments, please email Frank Reil at frank.reil@joanneum.at or Lukas Kreilinger at lukas.kreilinger@tugraz.at.



FAB|Lab



PHOTONICS²¹

PHOTONICS PUBLIC PRIVATE PARTNERSHIP

