

# Building an Arduino controlled Red-Green-Blue-(RGB)-LED-Lamp

Target audience: Students, 10-14yo

Time planning: 3-4 hours



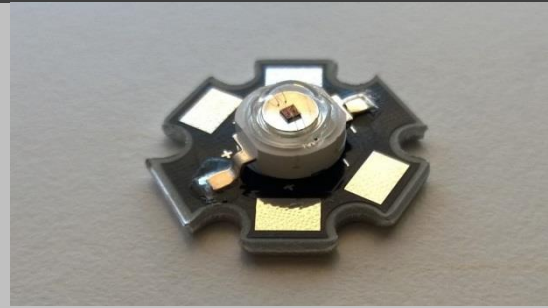
## Introduction

In this workshop, we are going to build an RGB Lamp providing 3 bright light cones in red, green and blue (or other colors) which turn on and off in a row. The speed of the light show can be manually controlled by a turning knob which connects to a potentiometer (adjustable resistor). At low speed, the light cones visibly light up one after the other; at maximum speed they appear to shine at the same time, which generates some interesting color mixing: When you put your hands between the lamp and a white wall, new colors and colorful shadows appear!

# 1. Part list



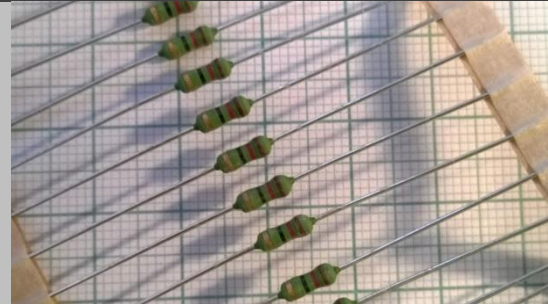
**1 WeMos or similar Arduino-like controller.** A WeMos is a very inexpensive microcontroller, which can be programmed as easily as an Arduino. This WeMos will control the LEDs according to the uploaded 'sketch' (a term for program coined in the Arduino community).



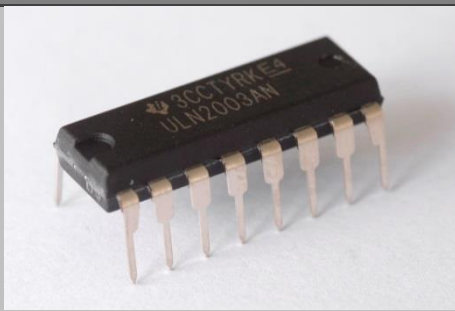
**3 high-power LEDs (one of each color red, green and blue).** 'LED' stands for Light Emitting Diode and generates light of certain color. A diode is a one-way-street for electrical current. When you turn around the diode, the current is blocked.



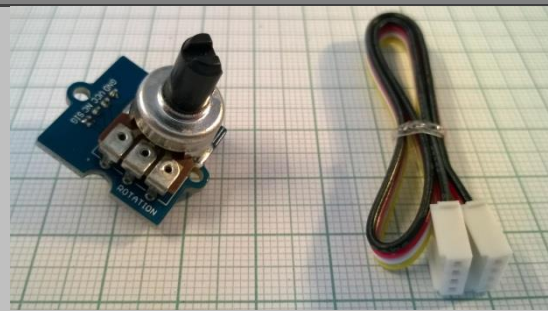
**3 lenses fitting the LEDs.** The lenses we use concentrate the light provided by each LED. They are simply attached to the LED.



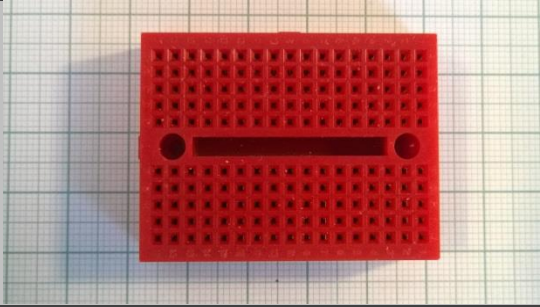

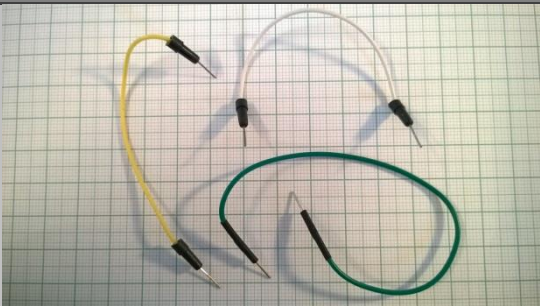
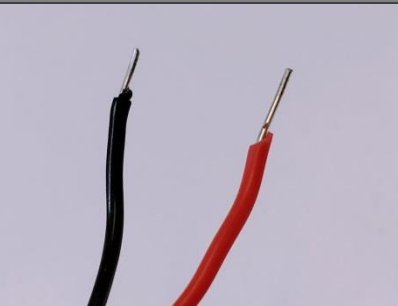

**3.3Ω, 4.7Ω and 5.6Ω resistors (0.5W).** **Resistors** limit the current in an electrical circuit. Here we need them for protecting the LEDs. Each LED-color needs a different resistor. A resistor has a set of colored rings on it which denote its electrical resistance; the larger its resistance, the more it limits the current



**1×ULN2003.** This integrated circuit (IC) is a 'driver' for the LEDs: The WeMos 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 ULN2003 does.



**1×Potentiometer** (some people call it 'rotary angle sensor' these days☺). A potentiometer is an adjustable resistor, with the resistance depending on the angle of rotation. It's widely used as a volume control in radios.

	
<p><b>1×Large breadboard (170 points).</b> This breadboard carries the WeMos and the ULN2003</p>	<p><b>1×small breadboard (55 points).</b> This breadboard carries the resistors for the LED</p>
	
<p><b>Breadboard jumpers, double male (6 green, 1 black, 1 red).</b> ‘Double male’ means they can be plugged into the board which each side.</p>	<p><b>Some insulated wire for connecting the LEDs (red and black, 50cm each, no braided wire!)</b></p>
	
<p><b>USB A to USB B connector for programming and powering the WeMos</b></p>	

## 2. Colors

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**  $\lambda$ . 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  $\lambda = c/f = 500\text{nm}$ .



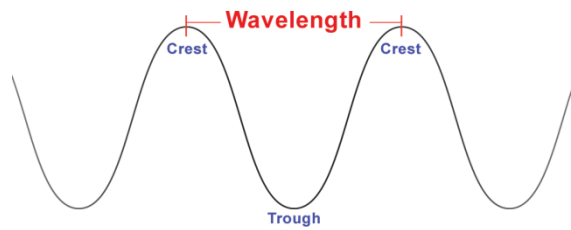


Figure 1: A wave of a given wavelength

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.

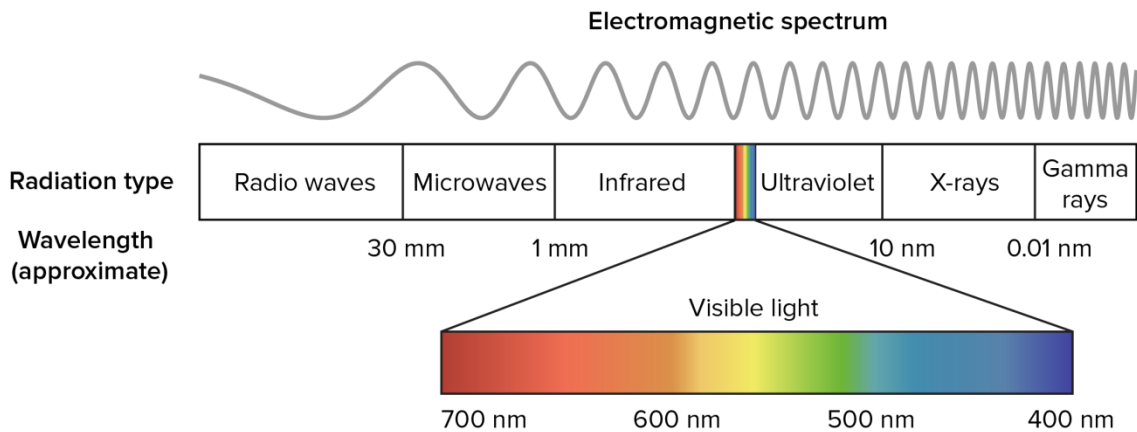


Figure 2: Electromagnetic spectrum

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

### 3. Electronics 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 is 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 (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.

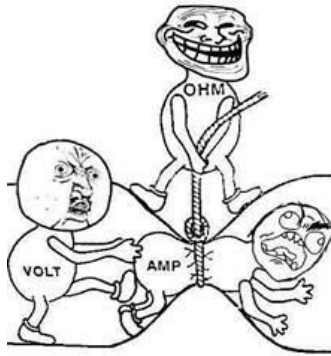


Figure 4: Another way to memorize the relation between voltage (VOLT), current (AMP) and resistance (OHM).

## 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			$10^{-2}$	$\pm 10\%$

Figure 5: Resistor code table

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

#### 4. The controller (WeMos)

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 peripheries. 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.

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.

## 5. Periphery

Figure 7 below shows the wiring of the WeMos to the periphery electronics.

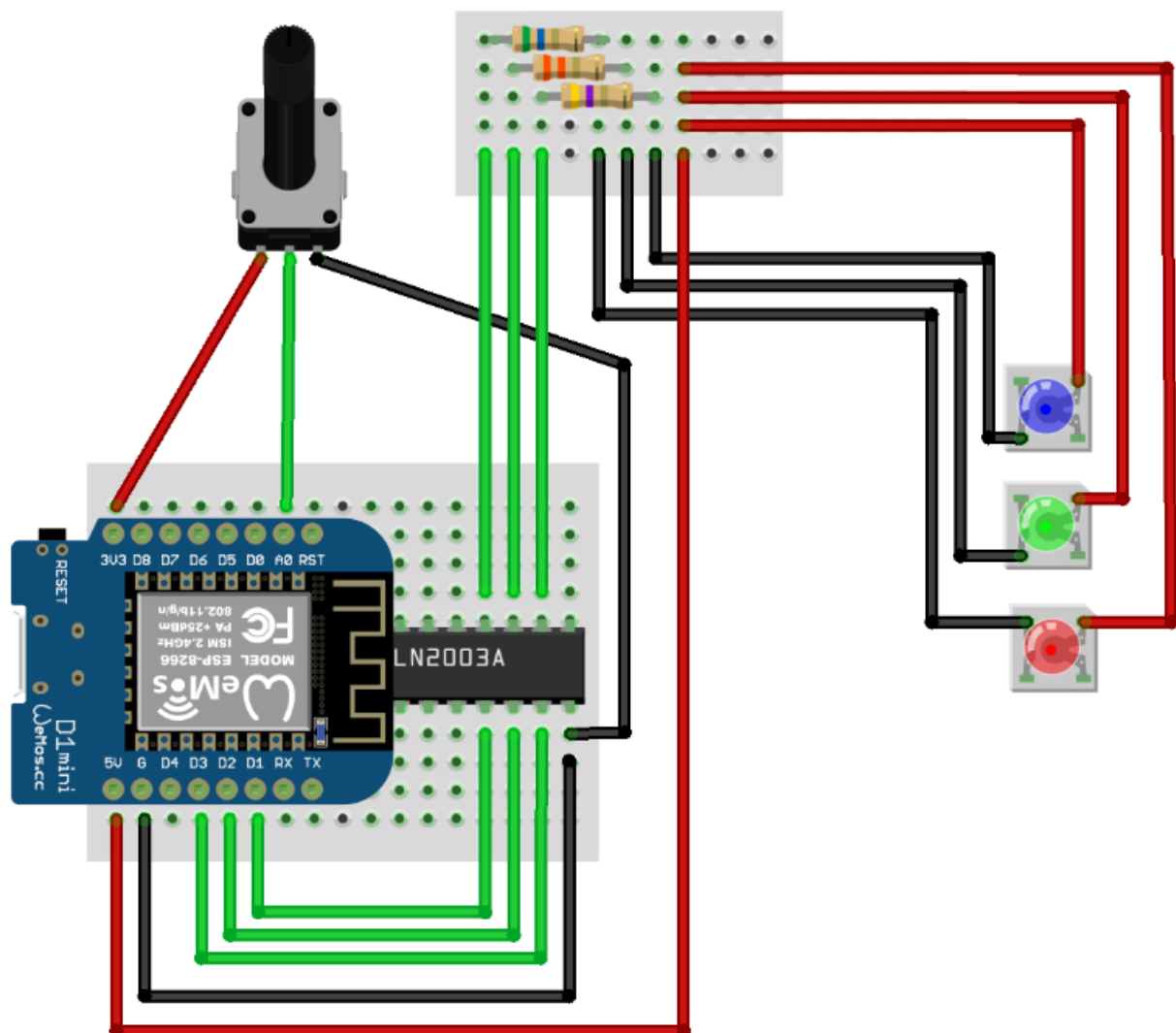


Figure 7: Wiring diagram of our RGB lamp



The WeMos switches its outputs D1, D2 and D3 on and off in a row. 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. The current circuit must be closed, so the ground (negative terminal) goes to the ULN2003, and the 5V positive terminal goes to each LED.

The potentiometer is connected to the **3,3V (!)** positive terminal and ground. Its center terminal then provides a voltage of 0...3,3V, depending on the angular position of the turning knob. This voltage is fed to the analog input A0 of the WeMos and determines the speed at which the LEDs blink.

## 6. 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>.

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

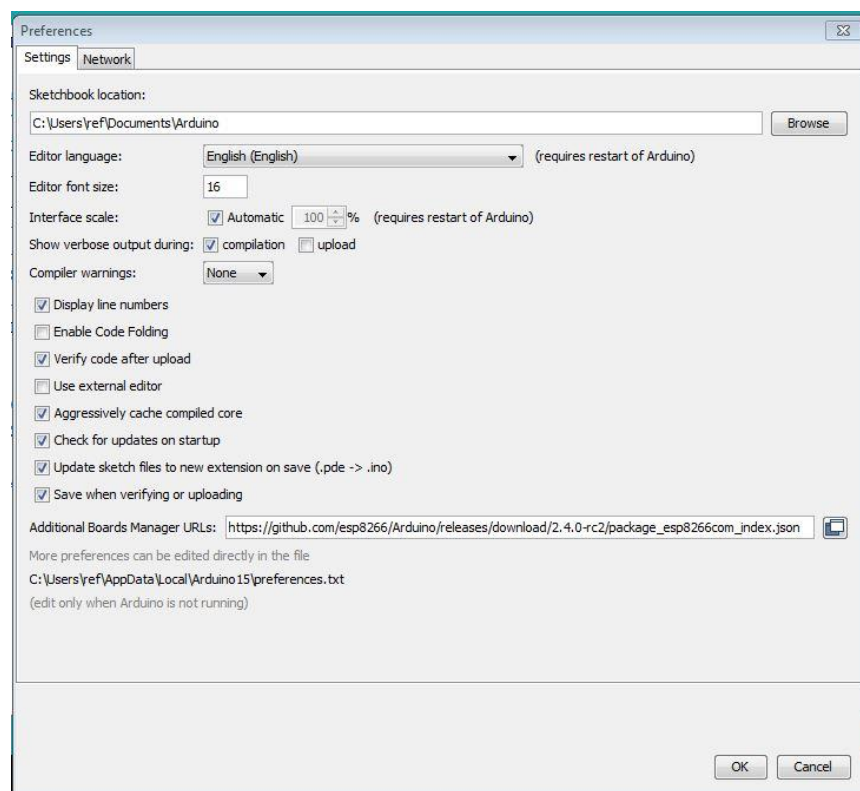


Figure 8: Adding the IDE for WeMos

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

[https://github.com/esp8266/Arduino/releases/download/2.4.0-rc2/package\\_esp8266com\\_index.json](https://github.com/esp8266/Arduino/releases/download/2.4.0-rc2/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.

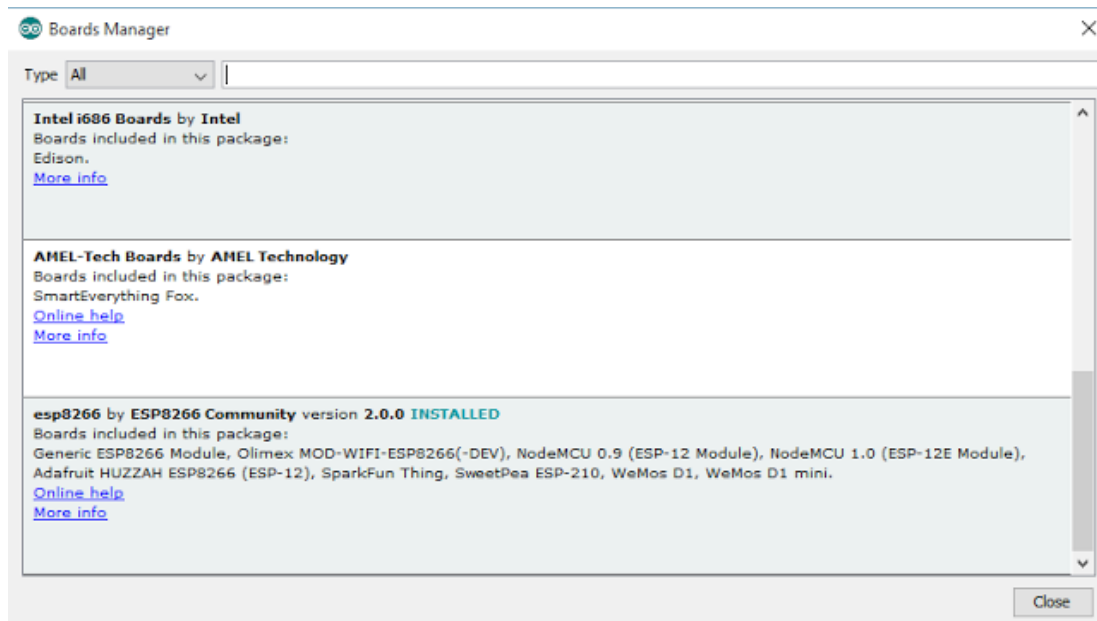


Figure 9: Installing the right package

Finally, click again in Tools→Board and select the appropriate WeMos (D1 R2 & mini, Lite, or whatever you use) from the list:

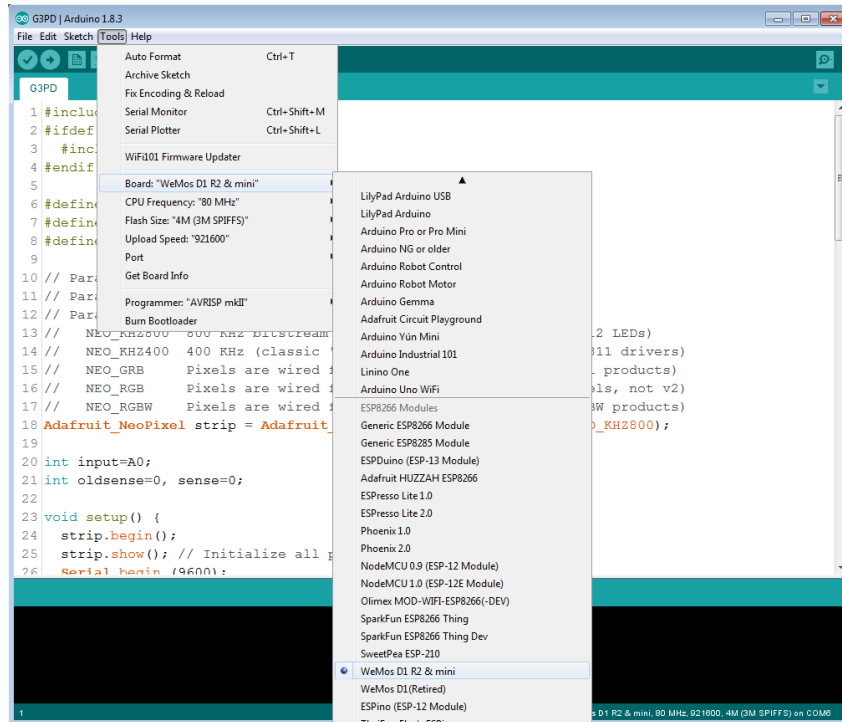


Figure 10: Selecting the WeMos from the drop-down menu

Now our WeMos is ready to be programmed.

## 7. The sketch (program code)

```

1 void setup() {
2   pinMode(D1, OUTPUT);
3   pinMode(D2, OUTPUT);
4   pinMode(D3, OUTPUT);
5   pinMode(A0, INPUT);
6 }
7
8 void loop() {
9   int t;
10  digitalWrite(D3, LOW);
11  digitalWrite(D1, HIGH);
12  t = analogRead(A0);
13  t /= 20;
14  delay(t);
15  digitalWrite(D1, LOW);
16  digitalWrite(D2, HIGH);
17  delay(t);
18  digitalWrite(D2, LOW);
19  digitalWrite(D3, HIGH);
20  delay(t);

```

Figure 11: Sketch for our RGB lamp

This is the sketch that controls the 3 LEDs. It is included with this workshop and will also be downloadable at <http://phablabs.eu>. It consists of 2 parts: the setup and an endless loop. In the setup function `void setup(){...}` the WeMos terminals D1, D2 and D3 are defined as outputs for the 3 LEDs and the terminal A0 is defined as input for the potentiometer. Further down, the function `void loop() {...}` switches off one LED and lights up the next, then waits for  $t$  milliseconds and moves on the next LED. The delay time  $t$  corresponds to the voltage at terminal A0, which is determined by the position of the potentiometer. There are many excellent basic and advanced tutorials on the web which explain everything in more detail.



Figure 12: Connecting the WeMos to a vacant computer USB slot

Connect the WeMos to a free USB-port of your computer (Figure 12) and find out the name of this port, using the Device Manager (German: Geräte-Manager→Anschlüsse (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 | }
```

```
Done uploading.  
Sketch uses 255587 bytes (24%) of program storage space. Maximum is 1044464 bytes.  
Global variables use 33032 bytes (40%) of dynamic memory, leaving 48888 bytes for local va  
Uploading 259728 bytes from C:\Users\ref\AppData\Local\Temp\arduino_build_191427\Golem.ino  
..... [ 31% ]  
..... [ 62% ]  
..... [ 94% ]  
..... [ 100% ]
```

Figure 13: Compiling and uploading to the WeMos

## 8. Building the electronics

### The large breadboard for WeMos and ULN2003



Figure 14: Large breadboards for carrying WeMos and driver IC ULN2003

We will build the circuit on a breadboard. Breadboards come in different sizes and have one more common rail. The large breadboards we use (Figure 14) 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 ULN2003, 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 Figure 15, 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.

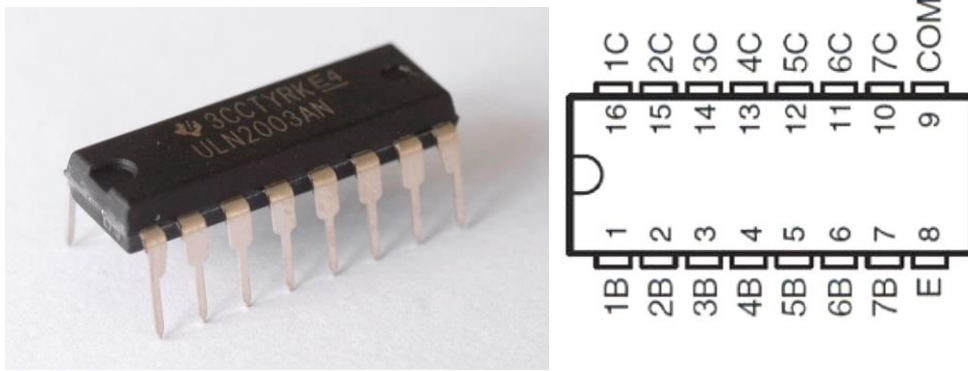


Figure 15: The driver IC “ULN2003”

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 (Figure 16).



Figure 16: The three LEDs with their wires attached

Then we prepare the potentiometer. The braided wire connector that comes with the potentiometer is too soft for the breadboard, so we don't use it. Instead, we take 10cm long solid wires (0.5...0.6mm diameter) and solder them to the potentiometer terminals as described in Figure 17.

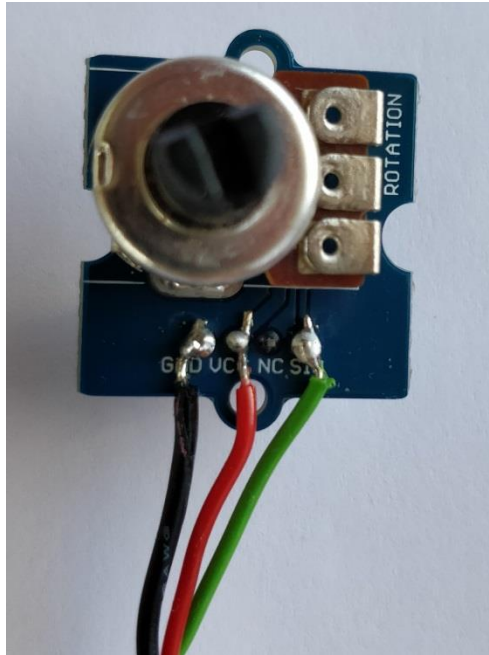


Figure 17: Connecting the potentiometer: Black goes to GND (ground), red to VCC (positive voltage) and green to SIG (signal). NC (not connected) will not be connected.

Now it's time to mount everything on those breadboards. Figure 18 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 **the potentiometer** terminals as shown: black to ground, red to 3.3 volts and green to the WeMos analogue input A0.

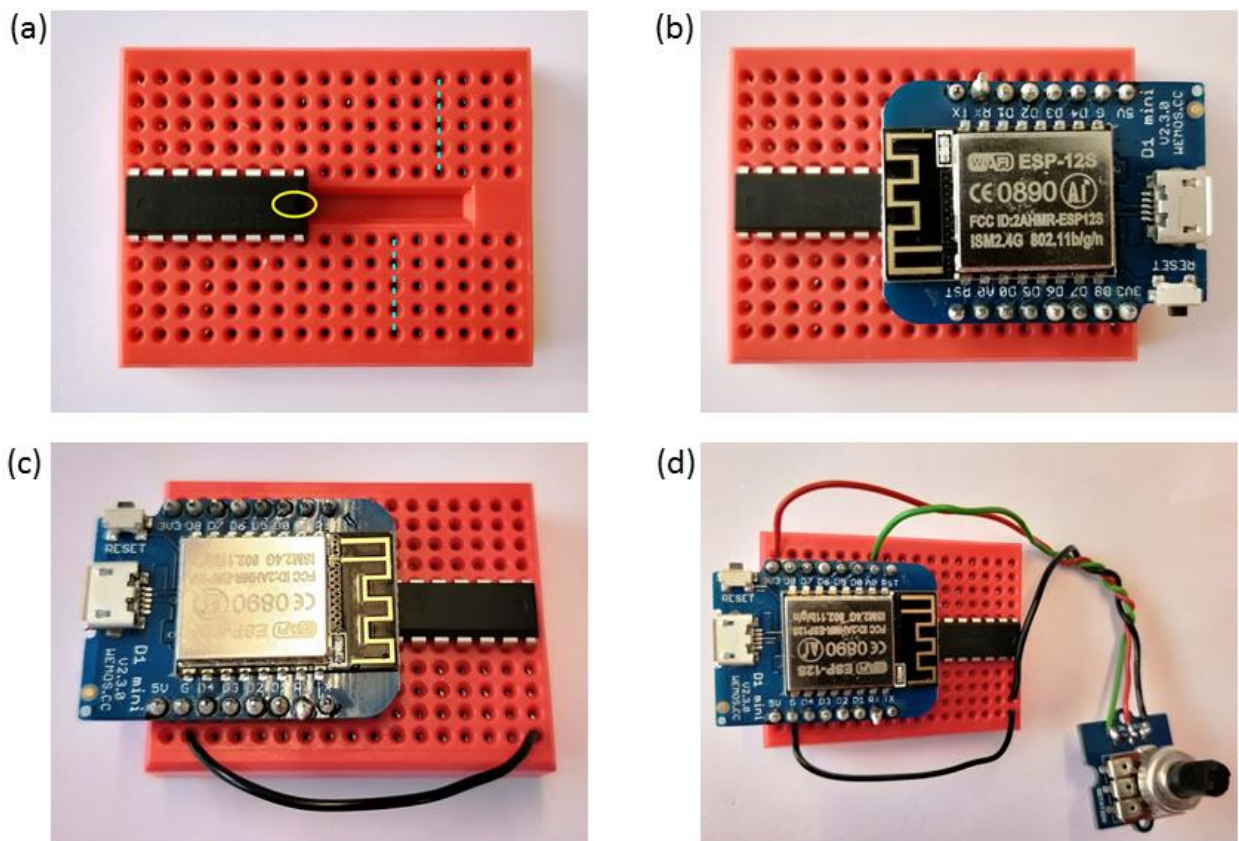


Figure 18: Steps 1-4 (see text for details)

Now, connect WeMos outputs D1,D2,D3 with inputs 5,6,7 of the ULN2003 as shown in Figure 19(a). Then we 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).



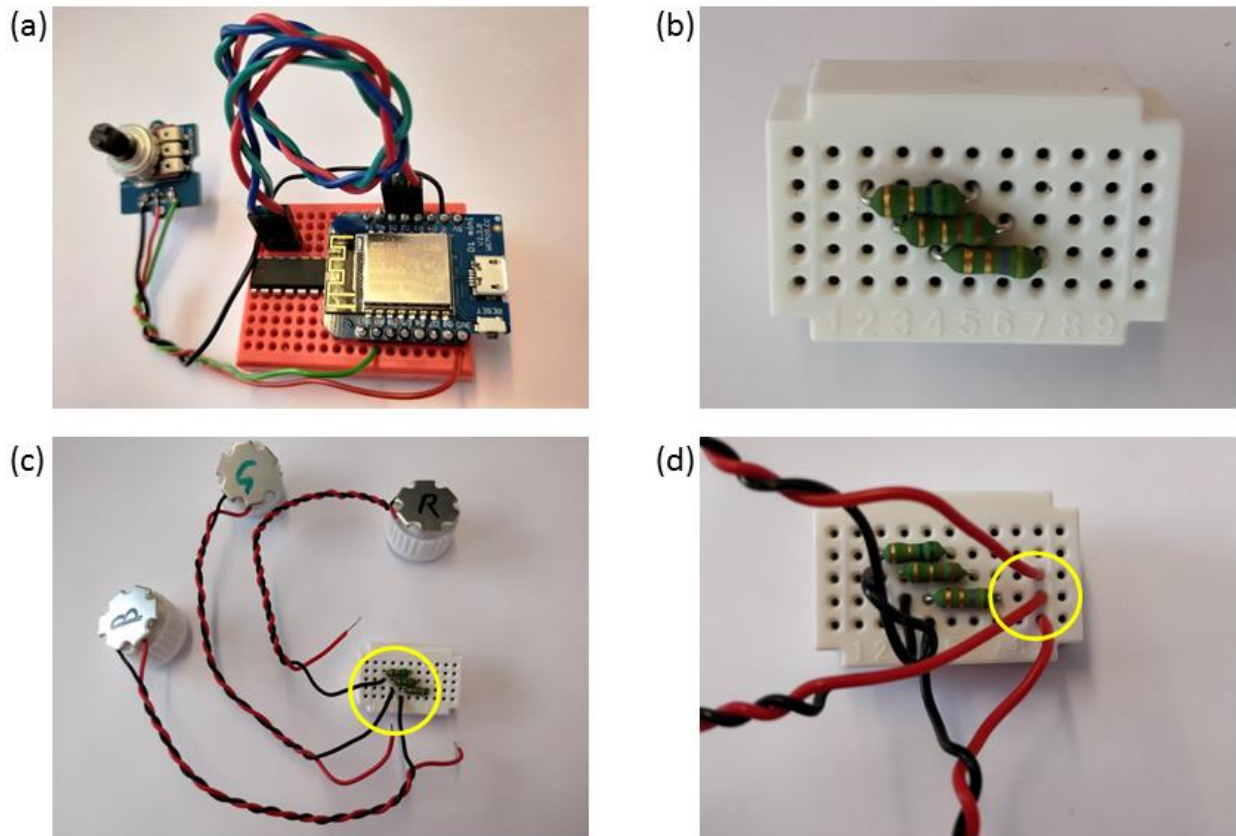


Figure 19: Steps 5-8 (see text for details)

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).

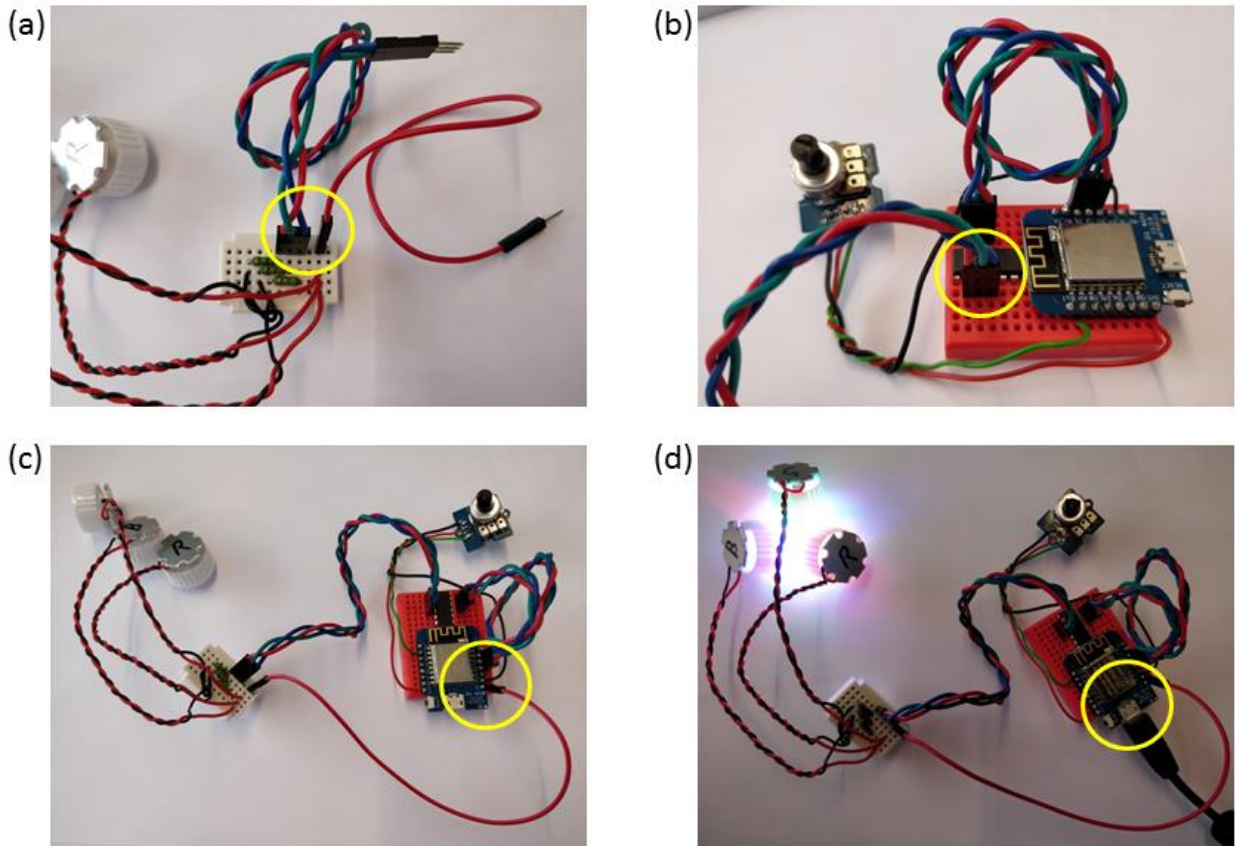


Figure 20: Steps 9-12 (see text for details)

## 9. Checking the lamp

Connect the lamp to a USB socket. The 3 LEDs should light up. If they don't, immediately disconnect the lamp from the USB socket and search for the error. When you turn the potentiometer the speed at which the LEDs blink should change.

## 10. 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 Figure 21 and Figure 22.

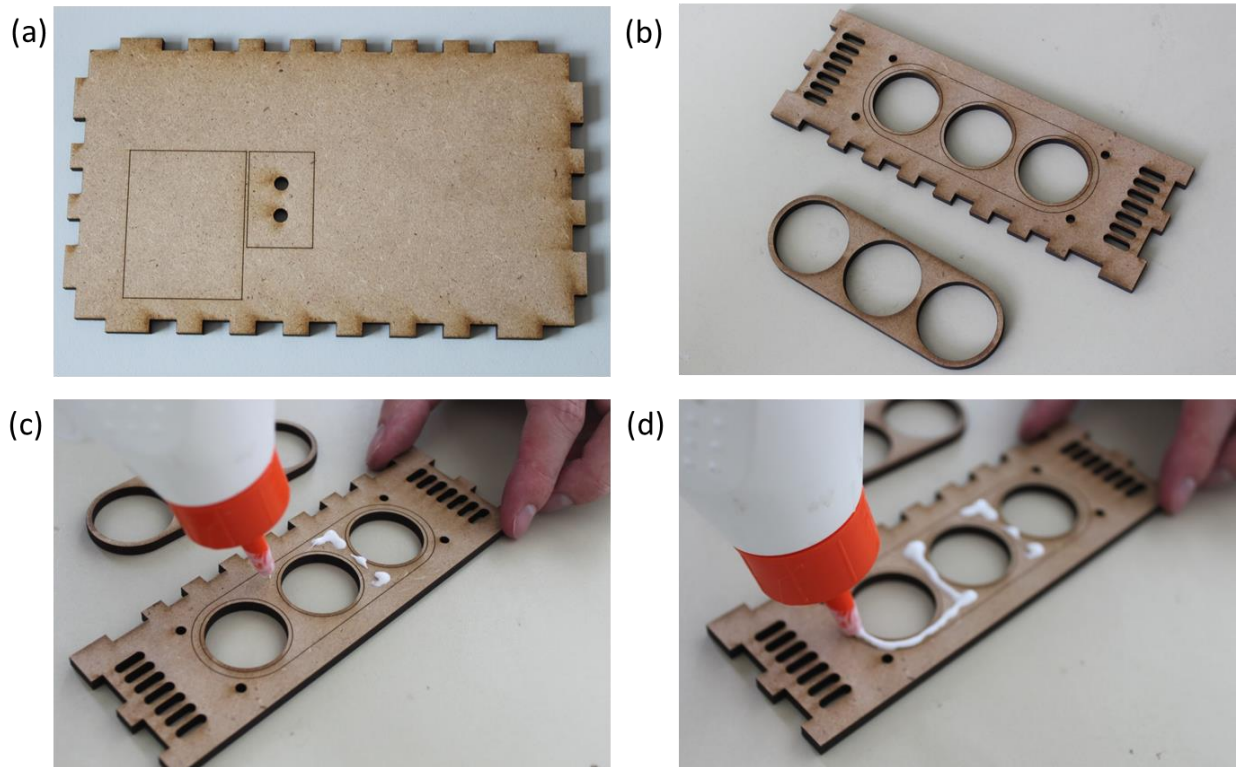


Figure 21: (a) Bottom plate, (b-d) Mounting the LED-plate

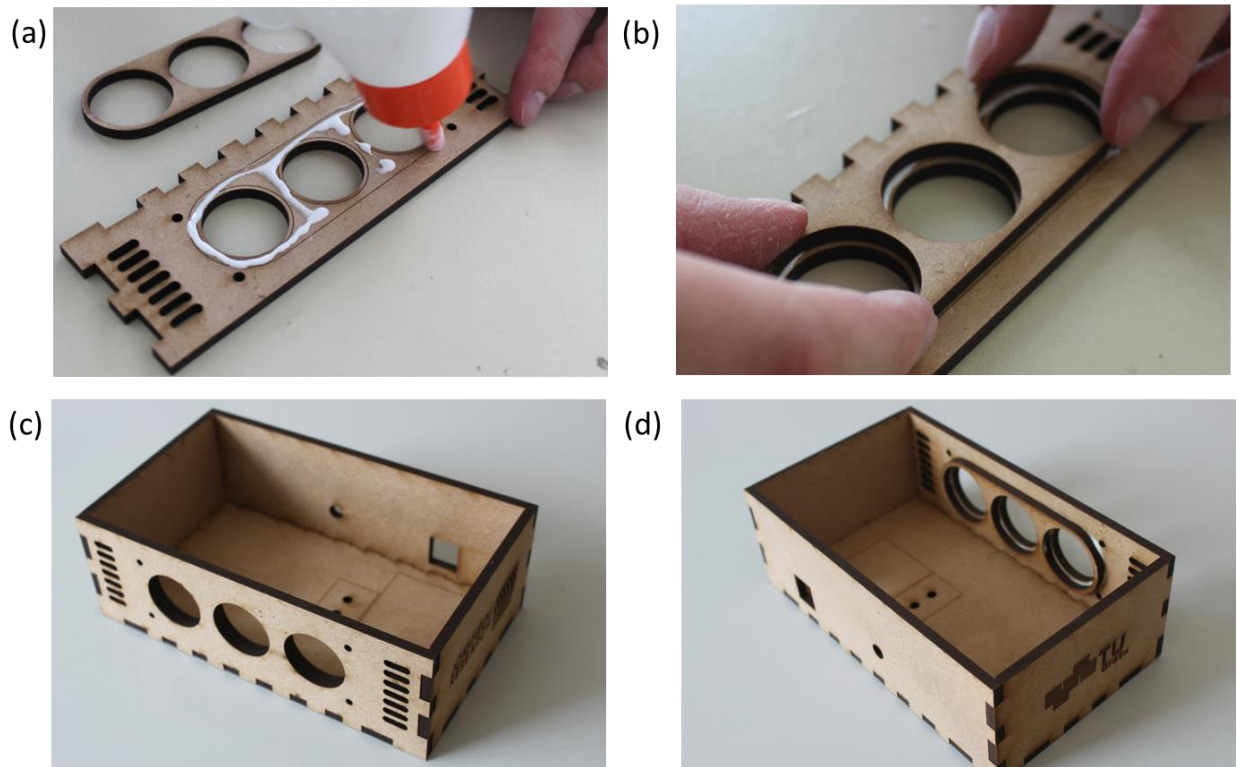


Figure 22: (a,b) Gluing the LED-wall, (c,d): Assembling the remaining parts (without lid yet!)

Now that the box is finished, we insert the electronics. Figure 23 shows the steps. The large breadboard has a sticky bottom, just remove the protective foil (Figure 23a). Place the board into the box (b), **make sure the WeMos USB-port aligns with the box opening** and press down until it sticks to the ground. Insert the potentiometer (c) and plug its turning knob to its axis (d).



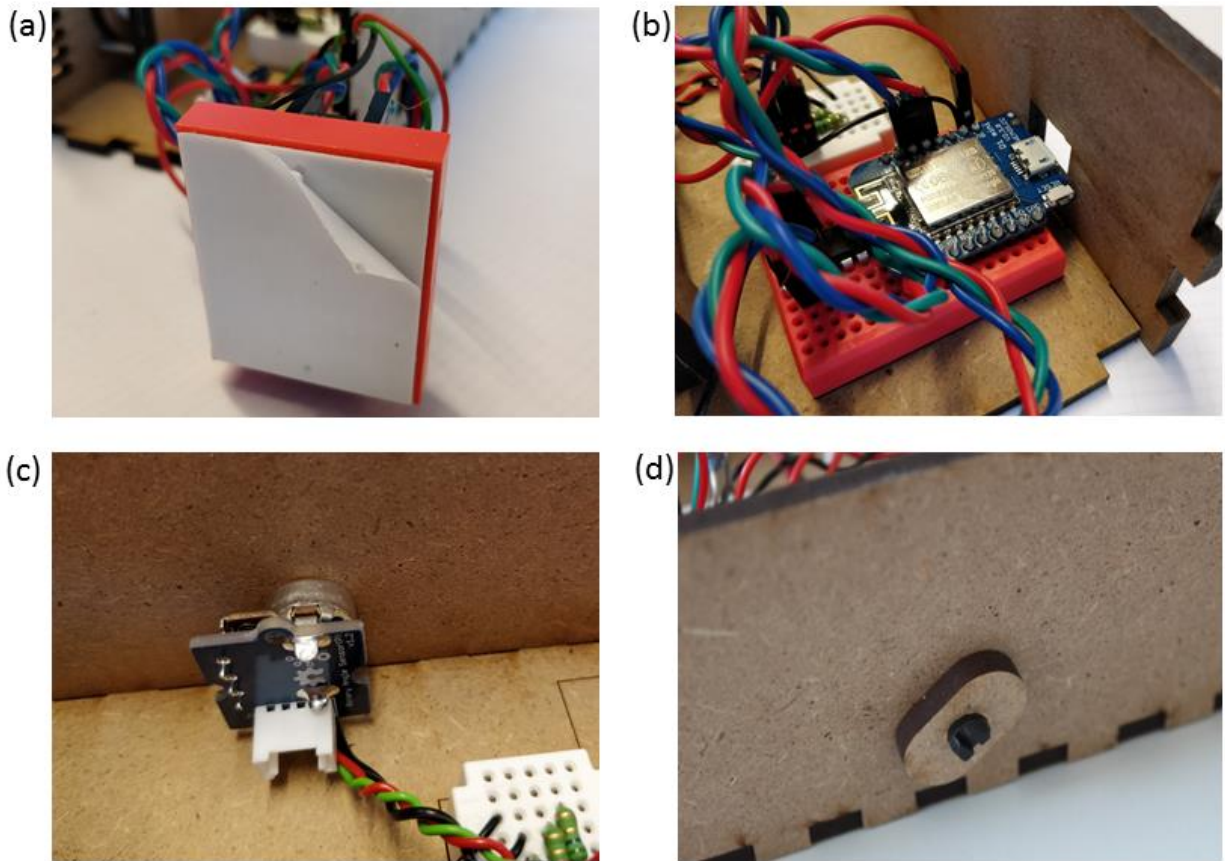


Figure 23: (a) The large board has a stick bottom surface so it can be (b) glued to the bottom of the box, so that the USB port is accessible through the opening (check from the outside first!). Insert the potentiometer (c) and add that funny little knob from the outside (d)

Now we install the LED and the lenses. Assemble the 3 lens kits as shown in Figure 24(a-c). In order to keep those lenses safely attached to the LEDs, we prepare an auxiliary plate; see Figure 24(d). Glue those corner pieces to the auxiliary plate as shown.

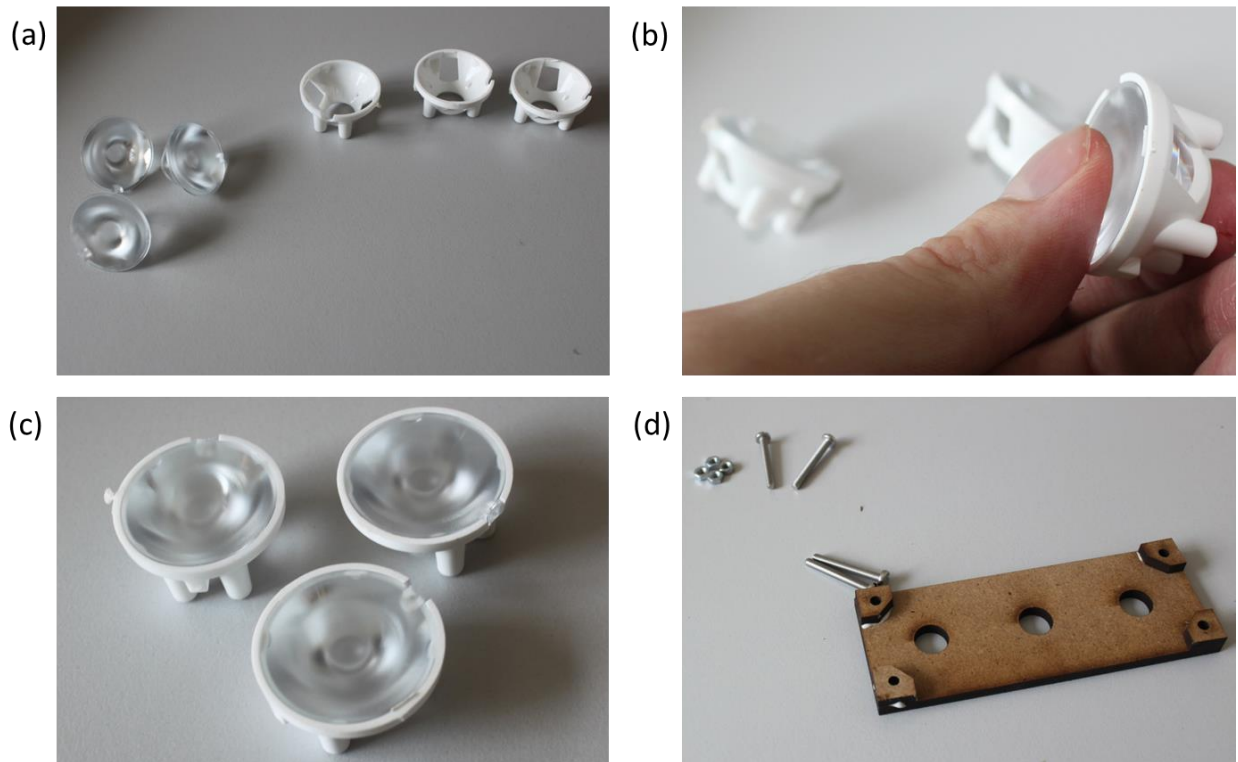


Figure 24(a-c): How to assemble those 3 lens kits. (d) Auxiliary plate for keeping lenses in place.

Insert the nuts in each corner hole (Figure 25a,b). Then put the boards with the screws on top of the LEDs (Figure 25c,d), and carefully insert the screws (Figure 26a). Turn those screws not too tight, just enough to keep everything in place (Figure 26b).

Glue 4 more corner pieces to the cover lid (Figure 26c), and then, as soon as the glue is dry, carefully (!) close the box with the lid (Figure 26d). Done! 😊

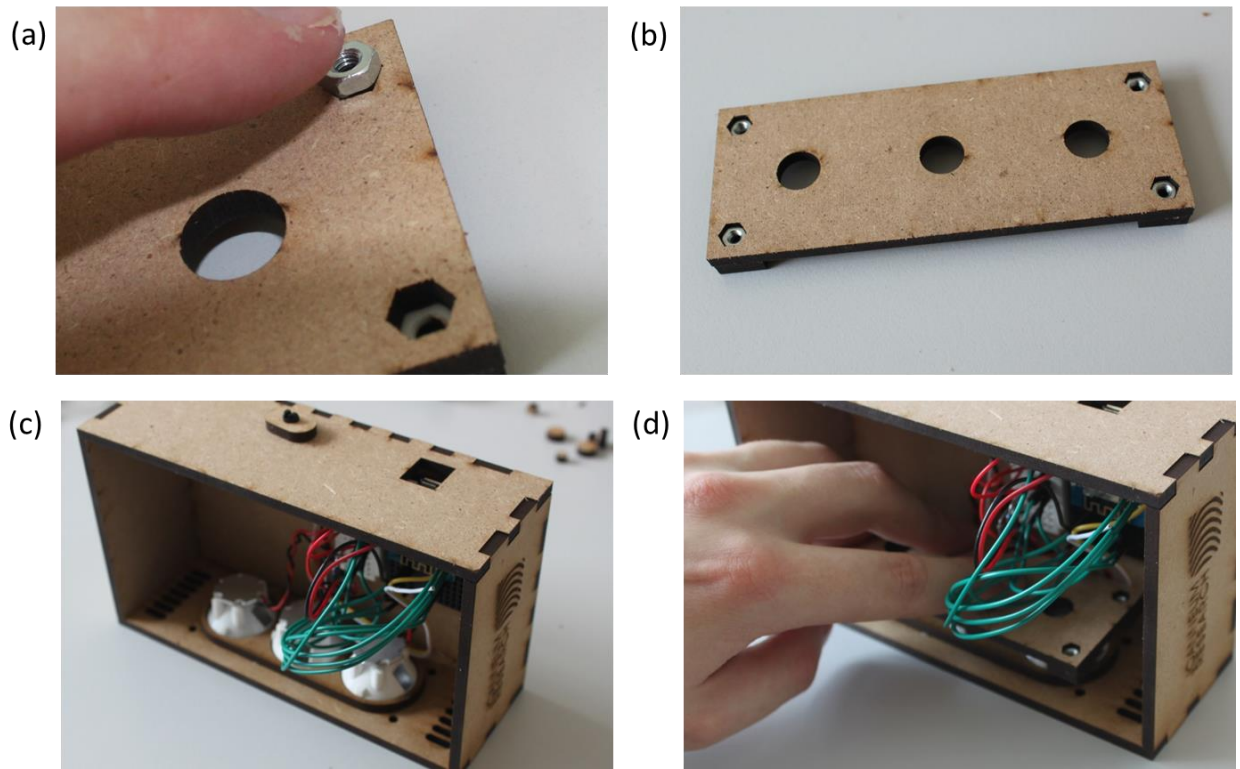


Figure 25: Preparing the auxiliary board

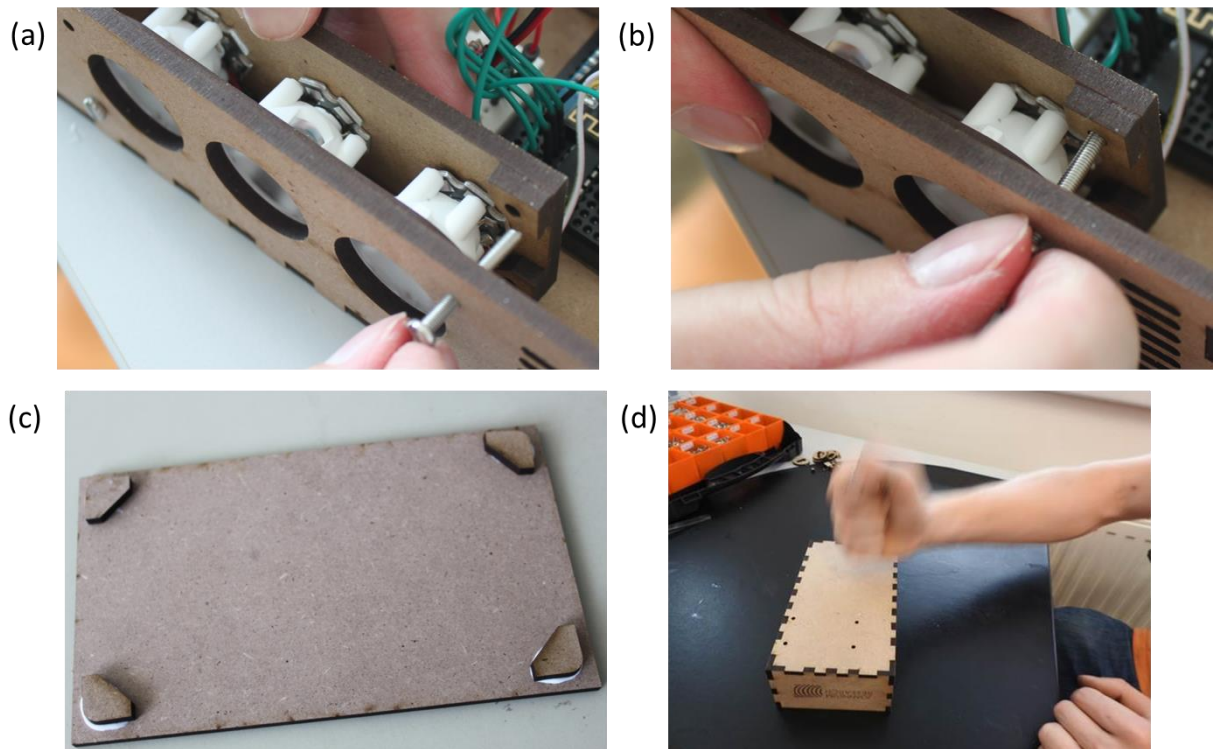


Figure 26: Attaching the auxiliary board and making the lid cover



## 11. Operation of the lamp

Reconnect the lamp to the power supply. Is everything still working? If not, unplug immediately and search for the fault.

With the potentiometer you can select the speed of the light change. Point the lamp towards a white wall and look what the shadows look like when you put your fingers in between. Where do those extra colors come from? If you don't remember, look in Chapter 2 (Colors).

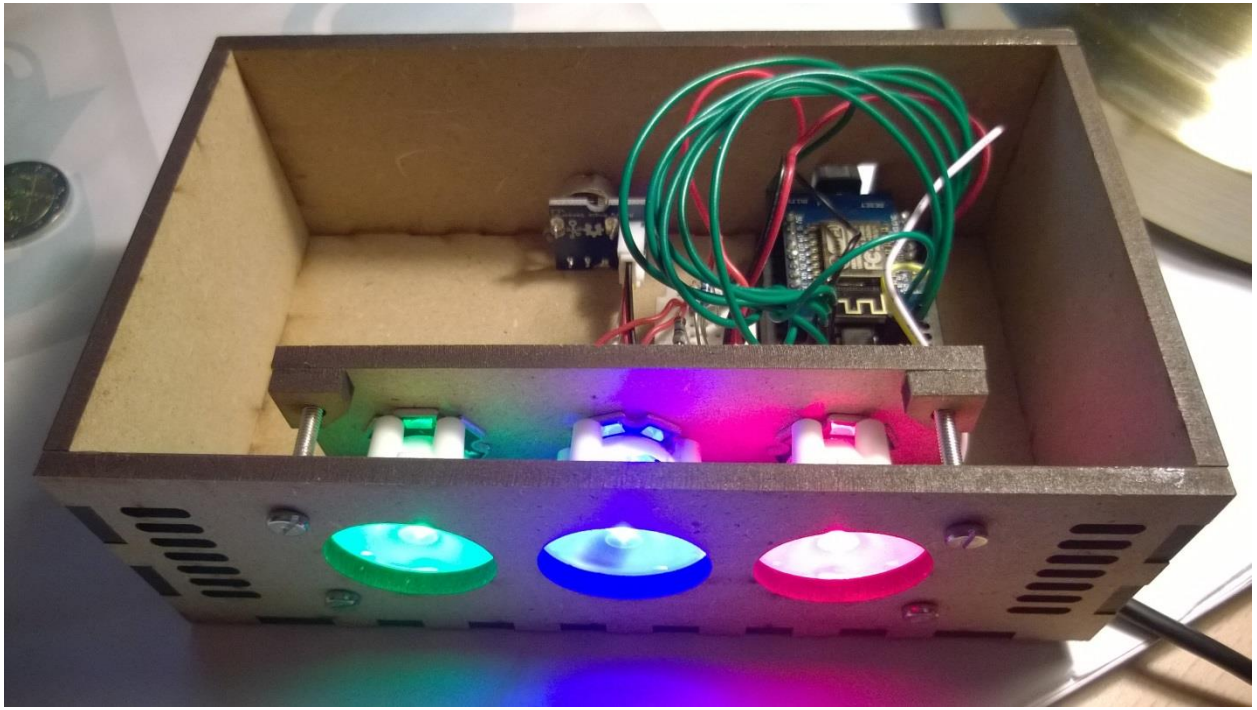


Figure 27: Finished device (without lid cover)





Figure 28: Colors and shadows on the wall

### **What have we learned?**

In this workshop, we have learned how to build an inexpensive 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.

### **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,

The logo for PHABLABS 4.0 features a stylized blue globe with orange sunburst rays emanating from it. To the right of the globe, the text "PHABLABS 4.0" is written in a bold, sans-serif font, with "PHABLABS" in blue and "4.0" in orange.

# 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](http://www.PHABLABS.eu)

This workshop was set up by the JOANNEUM RESEARCH in close collaboration with FAB|Lab Graz. For questions and comments, please email Frank Reil at [frank.reil@joanneum.at](mailto:frank.reil@joanneum.at) or Lukas Kreilinger at [lukas.kreilinger@tugraz.at](mailto:lukas.kreilinger@tugraz.at).



PHOTONICS PUBLIC PRIVATE PARTNERSHIP