PHABLABS 4.0

PHOTONICS WORKSHOP

REMOTE SENSOR WITH SMARTPHONE READOUT

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.

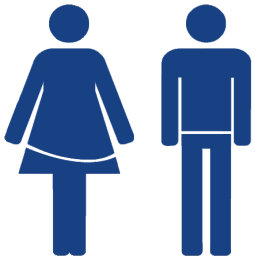
PHABLABS 4.0 consortium and its member organizations give no warranty that the provided information is accurate, up-to-date or complete. You are responsible for independently verifying the information. VUB cannot be held liable for any loss or damage that may arise directly or indirectly from the use 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.

PROPERTIES OF THIS WORKSHOP

SUMMARY:

Build your own remote sensor which can be read out by your smartphone. In addition, the measured values are shown on an OLED display or a RGB-Neopixel ring.



TARGET AUDIENCE:

Entrepreneurs (+18 years old)

SUGGESTED TIME PLANNING: (Total: 4h)

Timing in minutes	Activity

TOOLS:

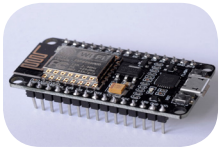
ESTIMATED COST:

€ 20

Step 1: Parts list

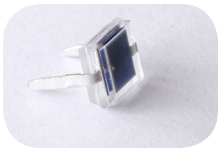
Collect all materials for each participant.

Electronic Parts:



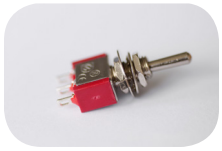
ESP 8266 or WeMos
A WeMos is a very inexpensive microcontroller, which can be programed as easily as an Arduino. This WeMos will control the LEDs according to the uploaded 'sketch' (a term for program coined in the Arduino community).

Neopixel ring
With 12 or 16 LED. An array of RGB-LEDs, with individual drivers and memory included.



Photodiode BPW34

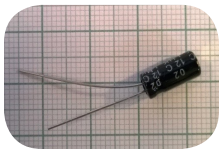
Circuit board



Toggle switch

Resistor 220 Ohm 0,25W
2 pieces/participant

Loudspeaker 45 Ohms



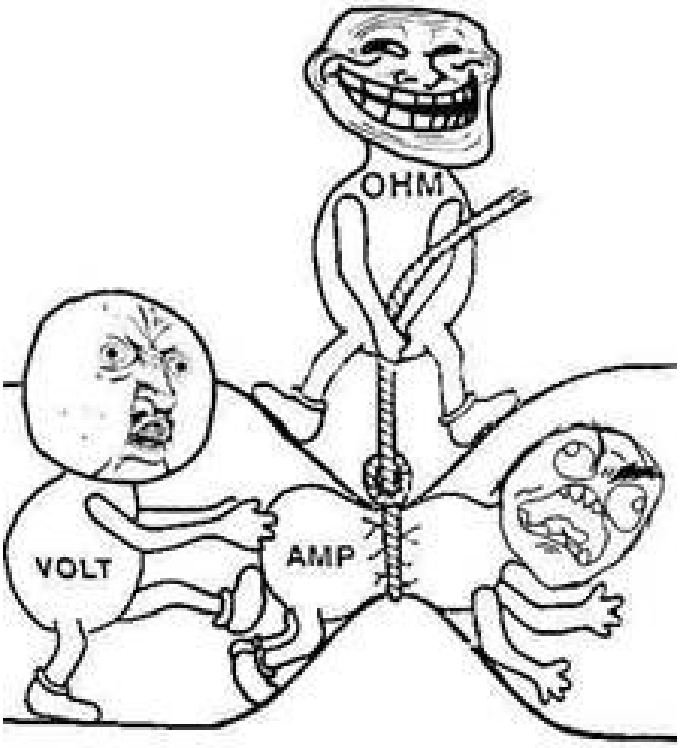
Electrolytic capacitor 4,7 microF

The photonics parts can be bought by EYESTvzw.
The electronic parts can be bought by Fablabfactory.

Step 2: Electronic basics

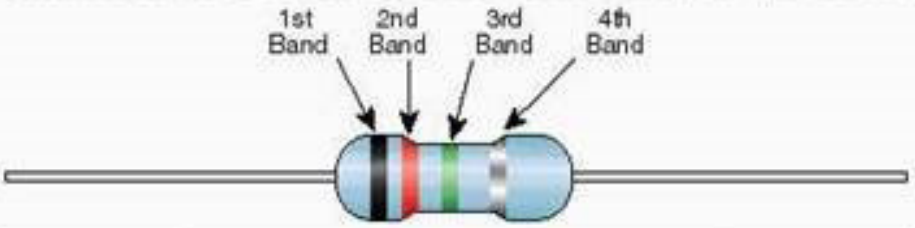
Voltage and Current

It is useful to compare an electrical circuit to something we have a better understanding of, like a water pipe system. Batteries or solar cells are like a water pumps; they generate water 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.



Another way to memorize the realtion between voltage (VOLT), current (AMP) and resistance (OHM).

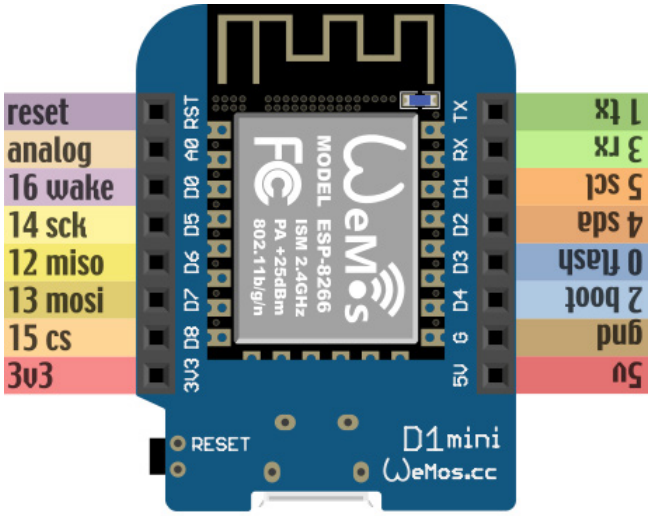
Standard EIA Color Code Table 4 Band: ±2%, ±5%, and ±10%

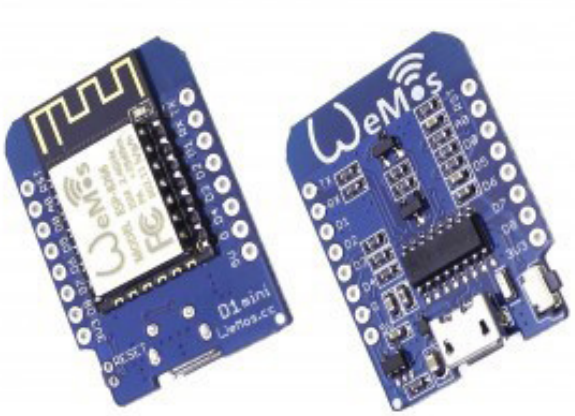


Color	1st Band (1st figure)	2nd Band (2nd figure)	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	10^0	
Brown	1	1	10^1	
Red	2	2	10^2	±2%
Orange	3	3	10^3	
Yellow	4	4	10^4	
Green	5	5	10^5	
Blue	6	6	10^6	
Violet	7	7	10^7	
Gray	8	8	10^8	
White	9	9	10^9	
Gold			10^{-1}	±5%
Silver			10^{-2}	±10%

Step 3: The microcontroller

This project uses a WeMos to control a NeoPixel ring, but a NodeMCU 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, which we will describe below. Also, the pinout will most likely be different.
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 is an Arduino based platform which can be directly programmed by any user through 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 made in China by a company other than Arduino, but which can basically perform the same tasks. There are many free Arduino tutorials available on the internet, which are perfect for any stage of advancement. In this tutorial, we therefore focus on everything necessary for building our WeMos controlled photometer.





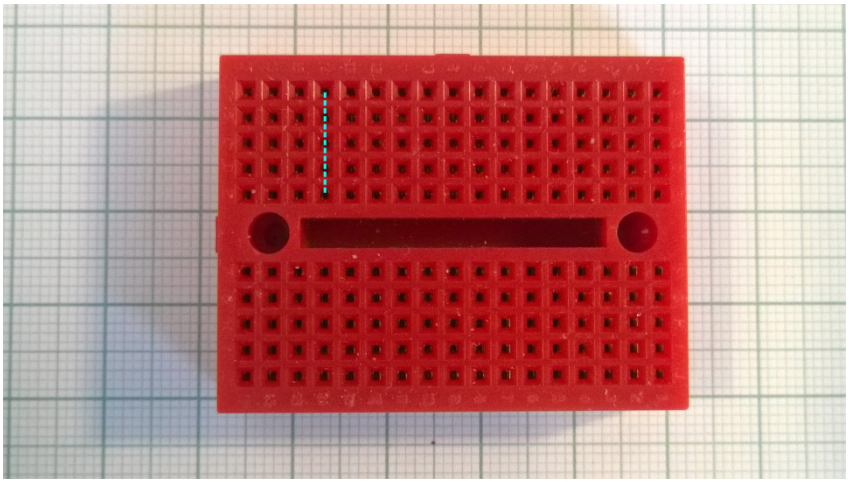
The metallic box is a shield for electromagnetic waves and contains the microcontroller and the WLAN transceiver, which allows the WeMos to wirelessly communicate. In the WeMos Lite, this shield is missing, which does not seem to pose a problem though. The curly printed wire outside this box on the left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

Step 4: Periphery

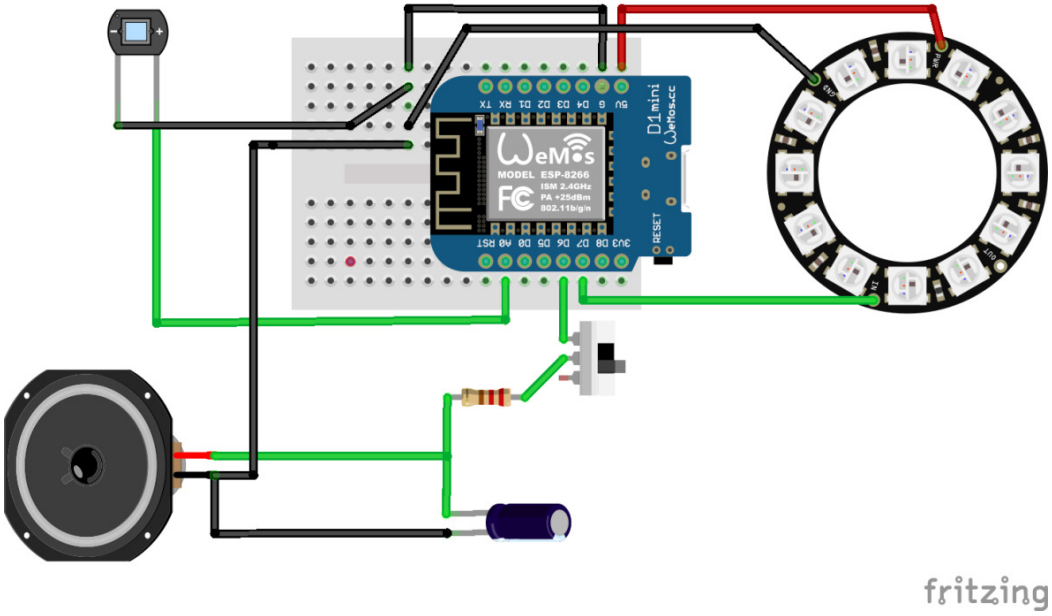
The NeoPixel ring

…is one of many funny devices offered by Adafruit. Once your project today is up and running, you can extend if with other RGB devices that you like.

We will build the circuit on a breadboard according to the wiring scheme.
Breadboards come in different sizes and have one more common rail. The large breadboard we use has a total of 17×10=170 points. Each 10-point-column consists of 2 segments with 5 mutually connected points each, as indicated by the cyan dashed line.



The NeoPixel ring has 3 terminals: plus(+) and ground (GND) for its supply voltage and Data Input (DI). It needs 5V, but also works with 3.3V. We connect its Plus-Terminal to the “5V” pin and its ground terminal next to the WeMos into a 5pin-column we call “ground column” of the WeMos. Connect this ground column to the “G” pin of the WeMos. “Data Input” of the ring will be connected to pin 7 of the WeMos. Here is the fritzing diagram of the photometer:



The WeMos reads the photo-voltage provided by the photodiode on its analog-digital-converting input A0. It feeds these data to the NeoPixel Ring via pin D7 and generates a sound signal at pin D6. The loudspeaker is connected to D6 through a resistor and with a capacitor in parallel, which keep the higher harmonics of the square wave at bay so the tone sounds more sinus-like.

Now take a 1×4cm² piece of printed circuit board and drill 2 4mm-holes into it, spaced at 30mm. Insert and solder the photodiode in the center. Solder 2×10cm long wires to it; a green wire to the anode (which is the side with the transverse metallic bar, see green arrow), and a black wire to the cathode.

Plug the green wire next to the A0-pin of the WeMos, and the black wire to ground column.
Look at the terminals of the loudspeaker; one of them is marked with a minus sign “-“. One of the terminals of the electrolytic capacitor is also marked by a minus sign. Solder the 4.7µF electrolytic capacitor to the loudspeaker terminals, so that alike terminals connect. Then, cut the wires of a 270 ohms resistor down to 1cm, and solder one end to the positive capacitor terminal. Also add the switch to the other end of the resistor using 2×10cm long wires. Connect everything according to the fritzing scheme.

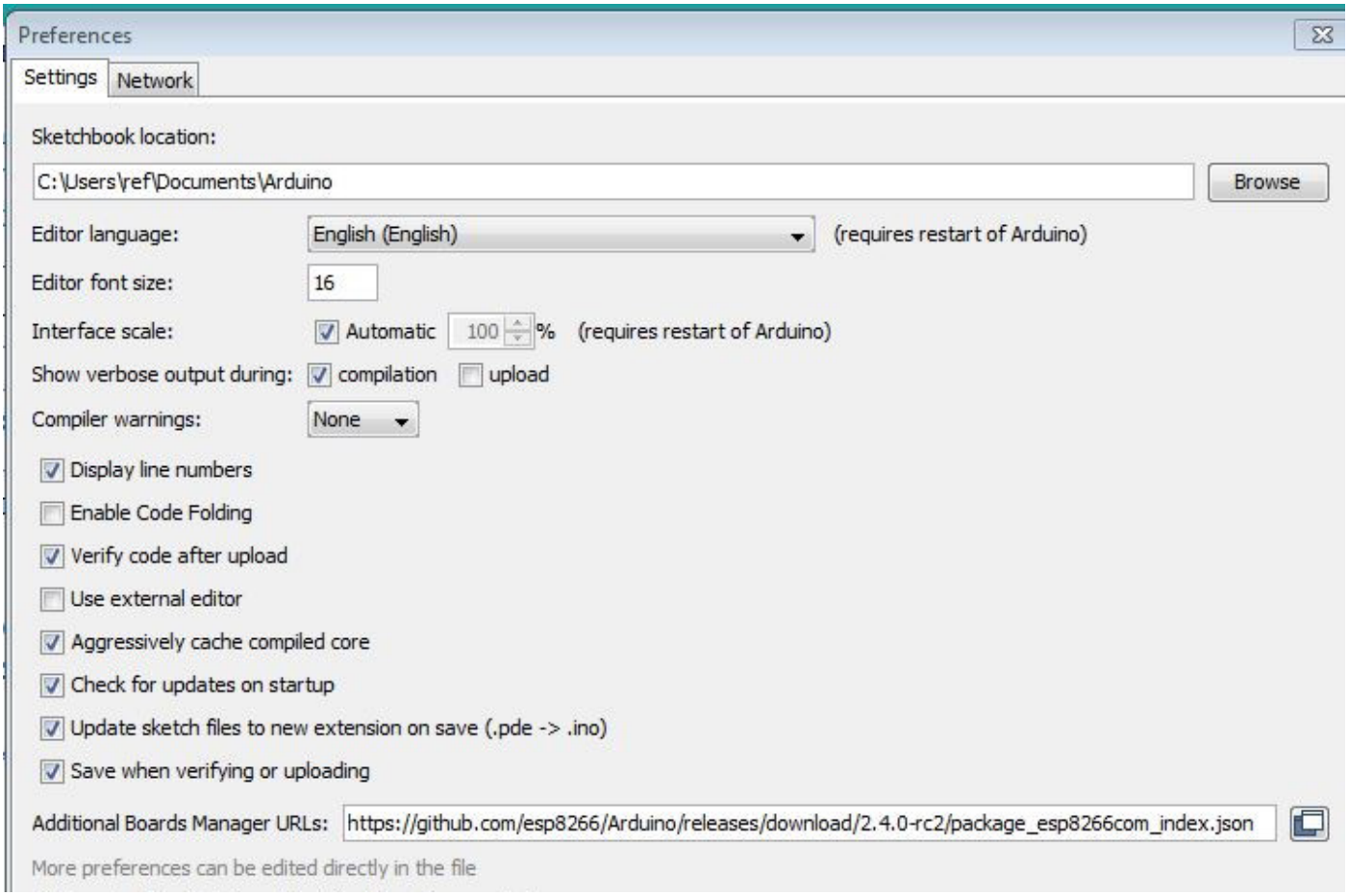
Step 5: Quick start guide for the WeMos

For programming and controlling the ESP, 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 we 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”.



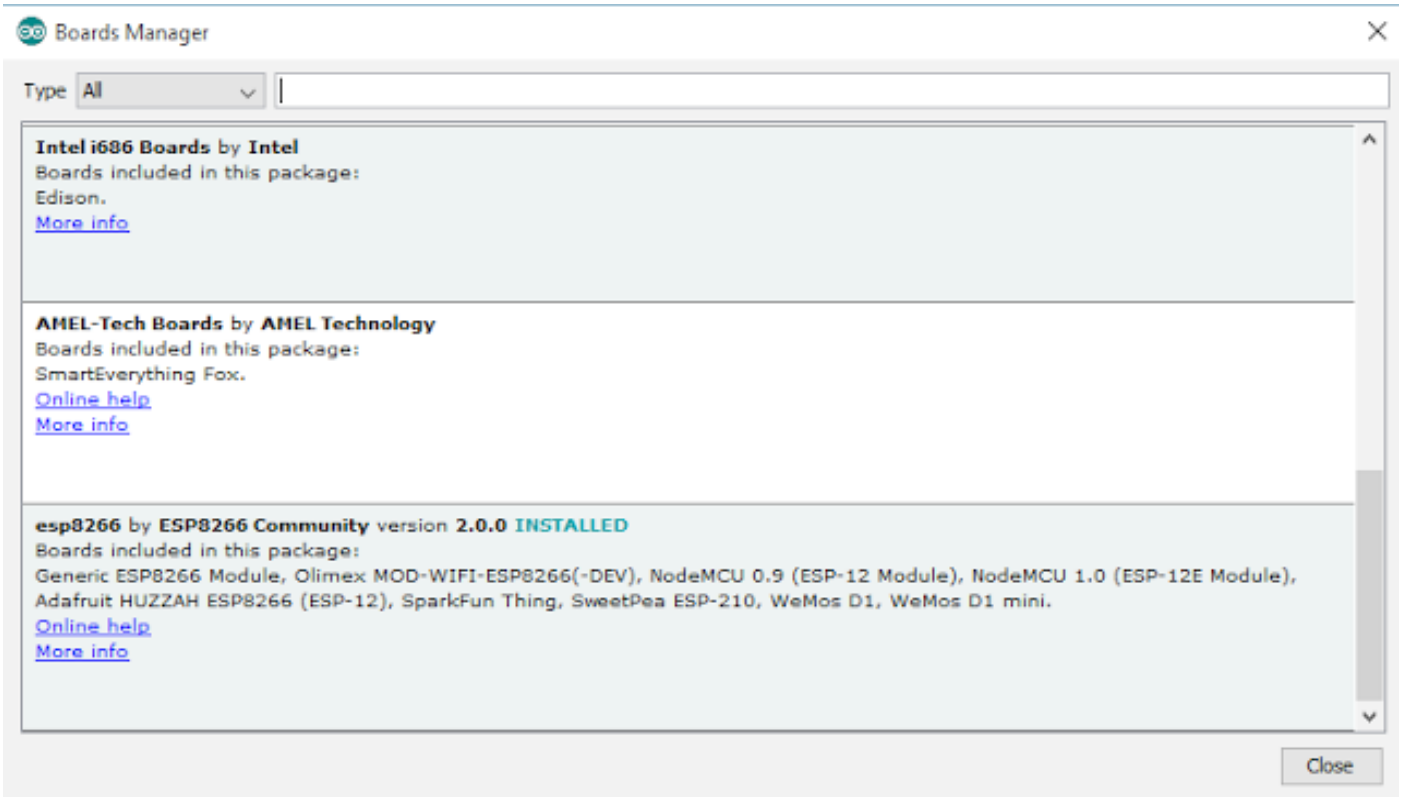
Copy the following line into the “Additional Boards Manager URLs” text box:

https://github.com/esp8266/Arduino/releases/download/2.4.0-rc2/package_esp8266com_index.json

and press OK to close the Preferences tab.

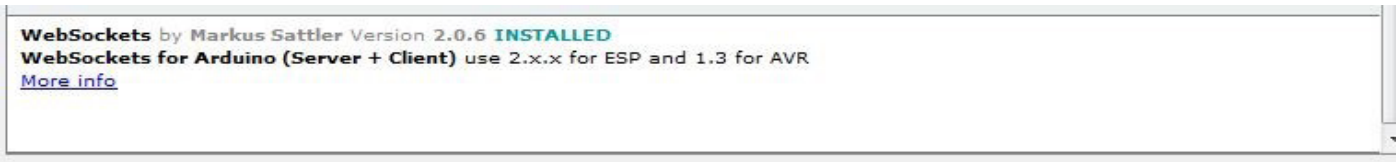
2. Select “Tools” and “Board”, and click on “Boards Manager…” in the pop-up menu.

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.



Additional libraries we need

For our programs, we need additional libraries. The library “WebSockets” by Markus Sattler provides easy WLAN communication and can be downloaded from the same menu as the esp8266 library:



We need another library that is not included in the Arduino IDE, so we get it from the internet. It is Timelib.h, which provides the time in hours, minutes and seconds relative to a predetermined starting point. This library is available at

https://github.com/PaulStoffregen/Time

On this site, press on the green “Clone or download” button and select “Download ZIP”. Do NOT unzip the zip-file; it can be directly included in the Arduino IDE:

Sketch Include Library Add .ZIP library

In this workshop, you can either use a NeoPixel ring or an OLED display to display the measured sensor values. The NeoPixel ring uses artful colored LEDs and is easy to read out from the distance. The small OLED-Display displays the actual measured number. In each case we need one additional library to control those output devices.

For the NeoPixel ring, we need library Adafruit_NeoPixel.h.
Go to

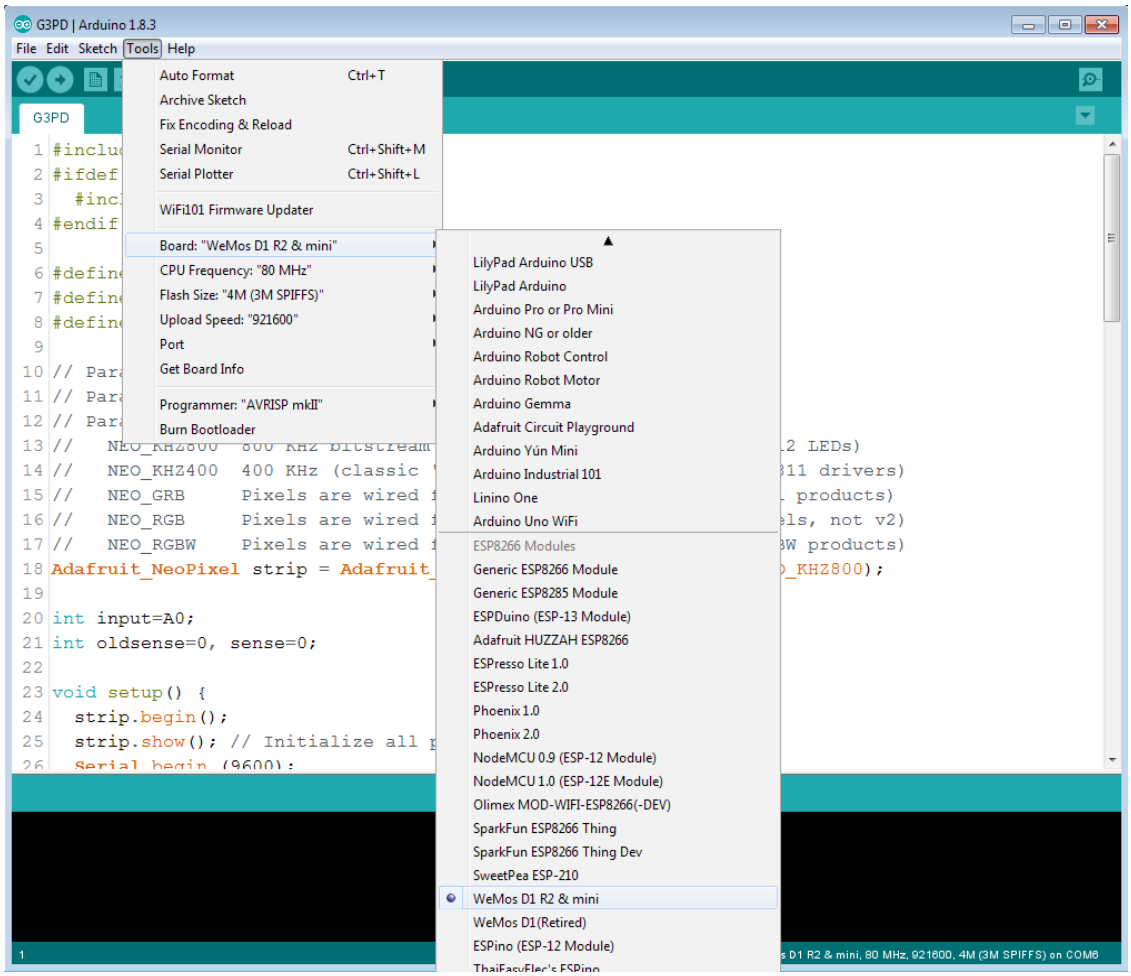
https://github.com/adafruit/Adafruit_NeoPixel

Proceed as for the Timelib.h library.

For the OLED display we need the SSD1306Ascii.h library.
Go to
https://github.com/greiman/SSD1306Ascii

Proceed as for the Timelib.h library.

Finally, we need to tell the Arduino IDE who it should communicate with. Go to Tools Board and select the appropriate WeMos (D1 R2 & mini, Lite, or whatever you use) from the list:



Now our WeMos is ready to be programmed. Don't connect it just yet.

Step 6: The sketch (program code)

We use the sketch with the code below, which enables the WeMos to be controlled by your smartphone. Here, the WeMos acts as an Access Point. The program code is included with this workshop and can also be downloaded at http://phablabs.eu/workshop/remote-sensor-smartphone-readout

```
// Connect your cell phone with Wurstserver, open a browser,
// and go to 192.168.4.1.
// WeMos will send value to your phone, change the tone signal frequency
// and the number (and color) of NeoPixel ring LEDs.
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h>
#include <TimeLib.h>
#include <Adafruit_NeoPixel.h>

#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIN D7

ESP8266WebServer server(80);
MDNSResponder mdns;
String bewegung_html = "<table border = 1>";
int sensorValue;

String twoDigits(int digits){
        if(digits < 10) {
                String i = '0'+String(digits);
                return i;
        }
        else {
                return String(digits);
        }
}

Adafruit_NeoPixel strip = Adafruit_NeoPixel(16, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
        // WiFi
        WiFi.softAP("Wurstserver", "");
        Serial.begin(115200);
        if (mdns.begin("esp8266", WiFi.localIP())) {
                Serial.println("MDNS responder started");
}
sensorValue = analogRead(A0);
server.on("/", handleRoot);
server.on("/inline", [](){
```

```
server.send(200, “text/plain”, “Funktioniert”);
});
server.begin();

// NeoRing
strip.begin();
strip.show(); // Initialize all pixels to ‘off’
}

void loop() {
        delay(100);
        sensorValue = analogRead(A0);
        RingEinschalten(sensorValue/2);
        tone(D6, sensorValue*3);
        server.handleClient();
        mdns.update();
}

void handleRoot() {
        bewegung_html = bewegung_html + “<tr><td>” + twoDigits(hour()) + “:” + twoDigits(minute())
        + “:” + twoDigits(second()) + “</td> <td> Sensorwert “ + sensorValue
        + “</td> </tr>”;

        server.send(200, “text/html”, bewegung_html);
}

void RingEinschalten(uint8_t sensetemp)
{
        colorSet(strip.Color(0, 0, 0), 16);
        if(sensetemp>=32) {
                sensetemp = sensetemp % 32 +1;
                colorSet(strip.Color(255, 0, 0), sensetemp);
        }
        else if(sensetemp>=16) {
                sensetemp = sensetemp % 16 +1;
                colorSet(strip.Color(0, 255, 0), sensetemp);
        }
        else {
                colorSet(strip.Color(0, 0, 255), sensetemp);
        }
}

void colorSet(uint32_t c, uint8_t maxpixel) {
        if (maxpixel > strip.numPixels()){
                maxpixel = strip.numPixels();
        }
        for(uint16_t i=0; i<maxpixel; i++) {
                strip.setPixelColor(i, c);
        }
        strip.show();
}
```

Brief explanation of the program

The program starts by including all libraries needed and by defining all variables. In the function setup() our WiFi access point “Wurstserver” (choose an individual name if more than one WeMos are present in the room so you know which one you’re connecting to), the Serial Monitor (which is a window within our program environment which can be used to control variables) and the NeoPixel ring are started. All LEDs in the ring are switched off.

The function loop() runs endlessly. It reads the voltage provided by the photodiode at pin A0, which is the analog-digital-converting input of the WeMos. The obtained value sensorValue is passed on as input to the NeoPixel ring, the tone generator and the WiFi access point.

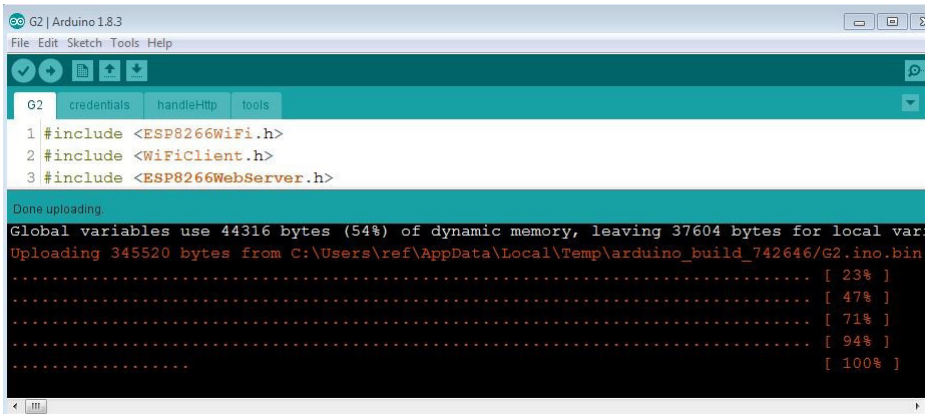
All other functions are defined after loop().

RingEinschalten() transforms the sensorValue to number and color of the LEDs switched on in the NeoPixel ring. For 1…16, the ring shines blue, for 17…32 green and above 32 red. The function colorSet is called in this function, with color and number of pixel as parameters. It is pretty much self-explanatory. There is a large variety of tutorials on NeoPixel devices in all languages and degrees of advancement on the internet.

Uploading the program

Now check your setup one last time by comparing it to the fritzing scheme in Fig. 6. Has everything been connected correctly? Are the solder points ok? Any short circuit or open circuit? Is the WeMos the right way around? Is the polarity of the electrolytic capacitor correct? Only if everything looks fine, go ahead.

Connect the WeMos to a free USB-port of your computer and find out the name of this port, using the Device Manager (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:

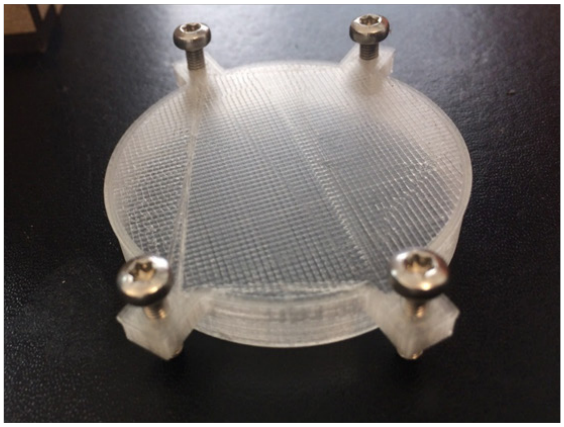


Now the photometer should already be up and running! The ring should light up and the tone generator should output a tone (when the switch is on!) with a frequency depending on the light intensity hitting the photo diode. If nothing happens or you smell something funny, immediately disconnect the WeMos and search for the error.

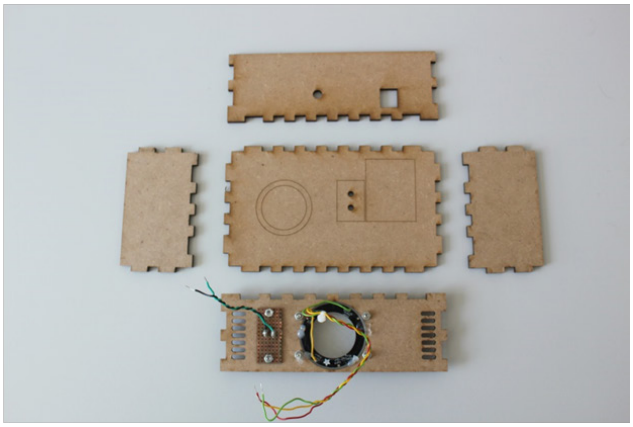
Step 7: Building the box fo the electronics

Disconnect the device from the USB plug. The following figure shows the mounting of the NeoPixel Ring and our 3D-printed scattering cap.

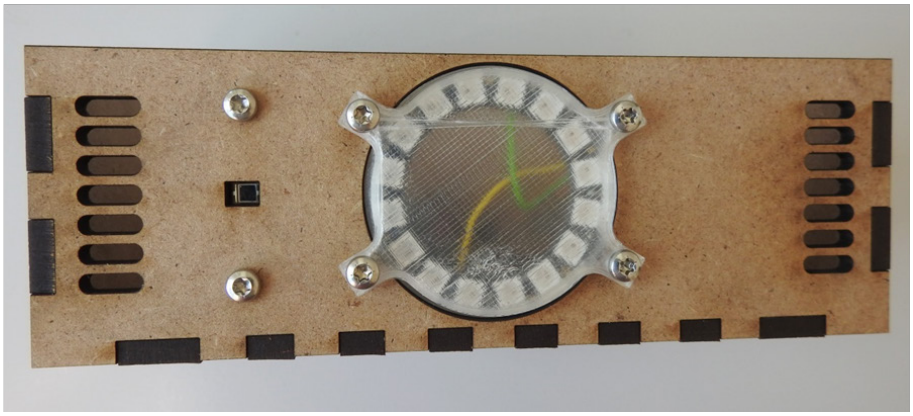
(a)



(b)



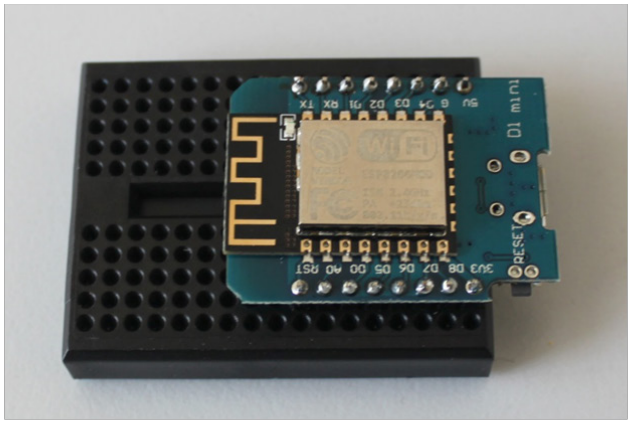
(c)



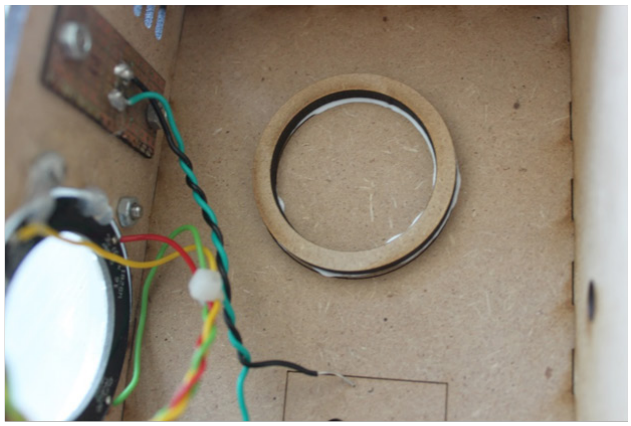
(a) Front cover for the NeoPixel ring, (b) box components with plugged in photodiode and NeoPixel ring, (c) front panel of photometer with photodiode (left) and NeoPixel ring.

Screw the small board with the photo diode to the front panel. You can also glue it (after soldering the cables to it!). The ring which holds the loudspeaker is glued to the bottom of the box. The bread-board with the WeMos has a sticky surface on its flip side, just peel of the protective layer. To keep the loudspeaker firmly in place you can either glue it into the ring or use a double-sided sticky tape.

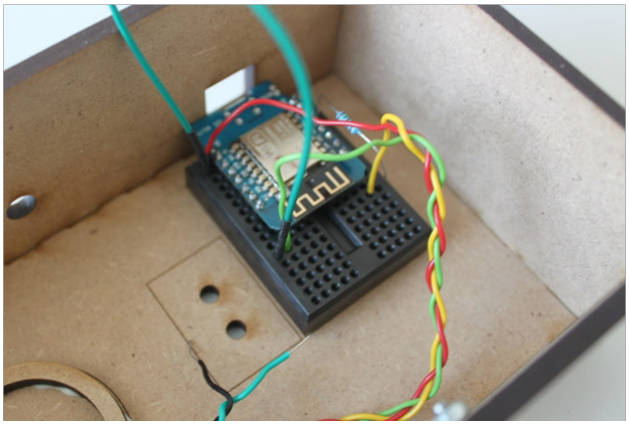
(a)



(b)



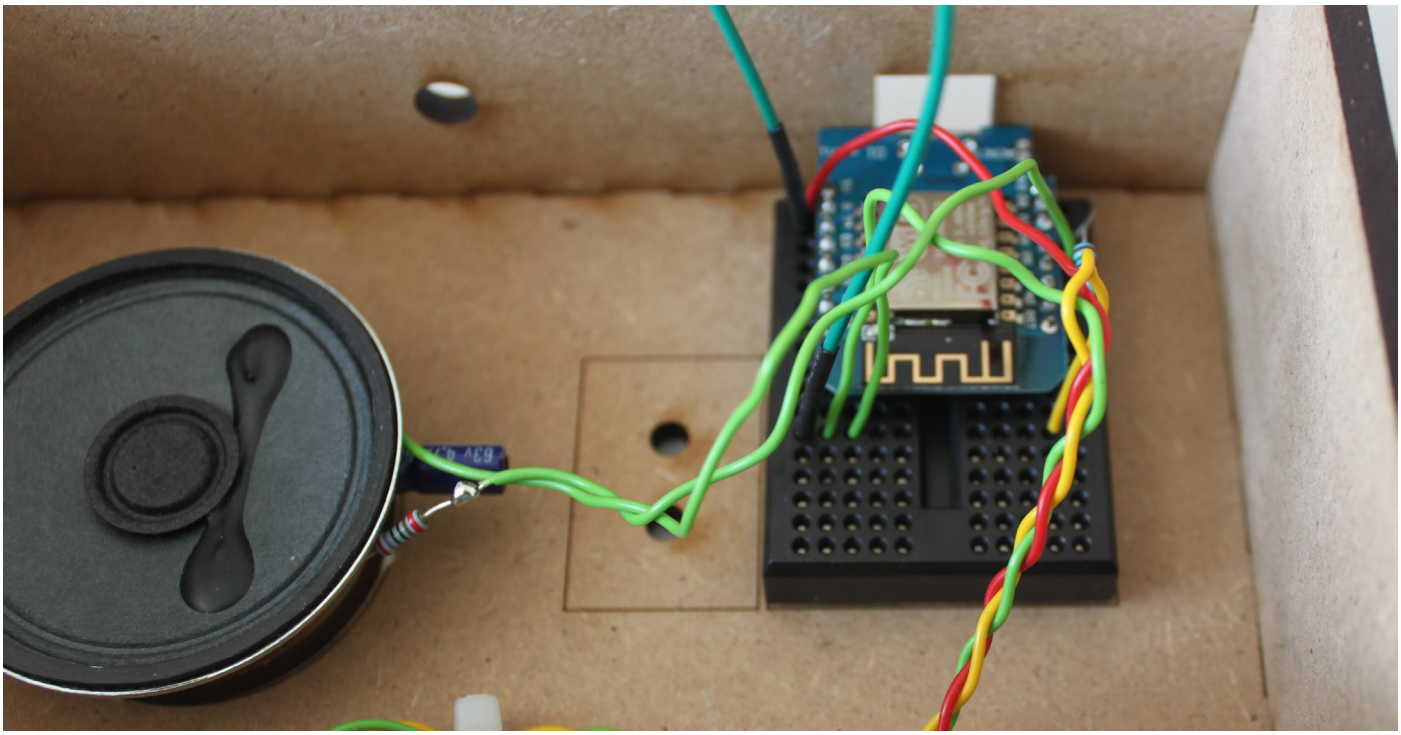
(c)



(d)

a) breadboard with WeMos, (b) interior of box with NeoPixel ring (left), photodiode (middle) and loudspeaker holder (ring glued to the bottom). (c) The breadboard has a sticky flipside (remove protective foil) and should be glued to the bottom of the box. (d) Loudspeaker with a resistor in series and a capacitor in parallel, to suppress the higher harmonics of the tone square wave.

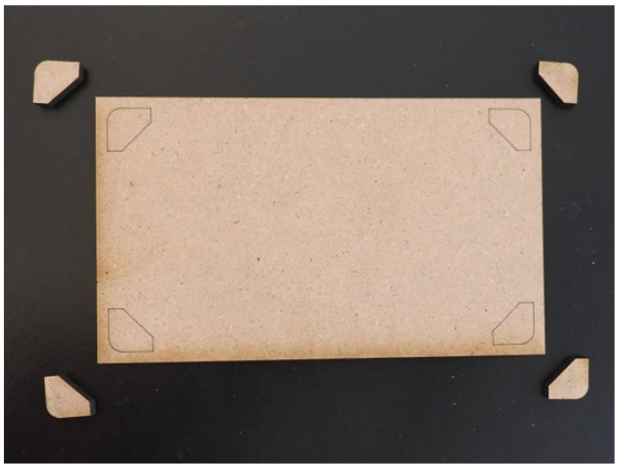
The figure below shows the completed electronics inside its box, consisting of the WeMos on a breadboard, the photodiode (green-blue wires), NeoPixel ring (green-yellow-red braided wires) and loudspeaker with its low-pass filter.



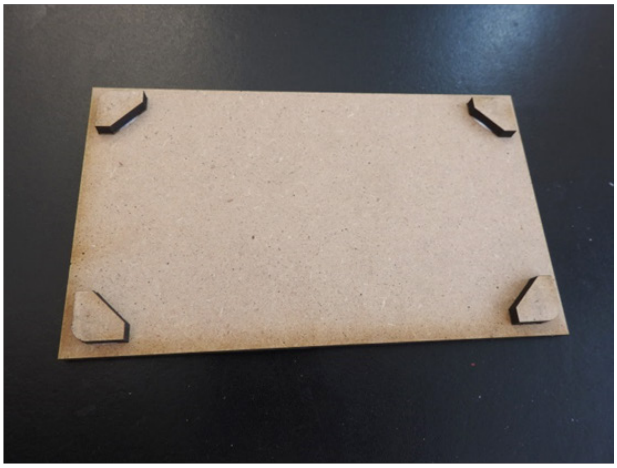
Completed electronics with WeMos on a breadboard, photodiode (green-blue wires), NeoPixel ring (green-yellow-red braided wires) and loudspeaker

We finish with the top of the box. Disconnect your photometer again. Glue the corner parts onto the lid as indicated by the gravure as shown in the figure below. Carefully close the lid.

(a)



(b)



(c)

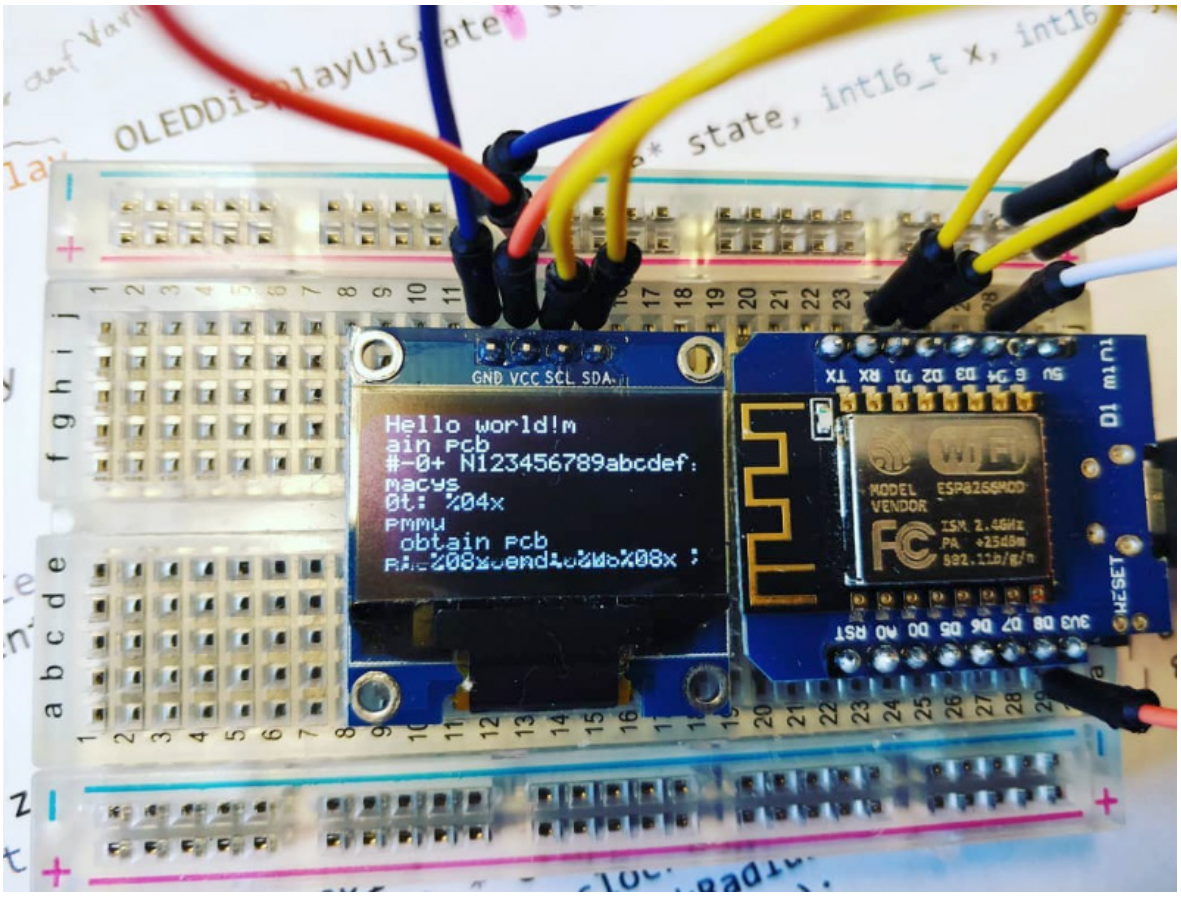
Step 8: Operation of the photometer

If all is fine, plug the 5V-USB connector into the socket on the WeMos board. Now the NeoPixel Ring should again light up and you should hear a tone from the loudspeaker.

When you change the illumination around the photodiode, the tone frequency should vary and the number and color of the Ring LEDs should change.

Now take a smartphone and look for the access point of your Wurstserver. You can connect to it without a password. Start your browser and go to URL 192.168.4.1. Wurstserver will provide you with the current light intensity in terms of a natural number. Every time you refresh the website, a request is sent and you will get an updated value.
You can replace the photodiode by other sensors, too. There is a large range of sensors available for Arduino-like systems.

Step 9: Using an OLED-display instead of a Neopixel ring



If you prefer numerical values over fancy colors, you can also use an (OLED) display instead of a NeoPixel ring. Here we use OLED display RIT253 (Fig. 14). This display has 128×64 pixels, and is controlled by the popular SSD1306 driver IC. It has 4 terminals and is controlled via I2C-bus: Vcc (+3.3V), GND (ground) for the power supply, and SCL and SDA for the data connection (CLock and DAta). Connect Vcc and GND to the respective terminals of the WeMos, SCL to D1 and SDA to D2. The program below “G3_AP_OLEDDisplay” can be downloaded from http://phablabs.eu/workshop/remote-sensor-smartphone-readout

```
// Connect your cell phone with Wurstserver, open a browser,
// and go to 192.168.4.1.
// WeMos will send value to your phone, change the tone signal frequency
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
#include <ESP8266mDNS.h>
#include <ESP8266WebServer.h>
#include <TimeLib.h>
#include <Wire.h>
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"
#define I2C_ADDRESS 0x3C
ESP8266WebServer server(80);
MDNSResponder mdns;
String bewegung_html = "<table border = 1>";
int sensorValue;
SSD1306AsciiWire oled;
String timenow;
char ssid[20] = "Wurstserver";

String twoDigits(int digits){
        if(digits < 10) {
                String i = '0'+String(digits);
                return i;
        }
        else {
                return String(digits);
        }
}

void setup() {
        strcat(ssid, "_Frank");
        WiFi.softAP(ssid, "");
        Serial.begin(115200);
        if (mdns.begin("esp8266", WiFi.localIP())) {
                Serial.println("\nMDNS responder started");
        }
        Wire.begin();
        Wire.setClock(400000L);
        oled.begin(&Adafruit128x64, I2C_ADDRESS);
        oled.setFont(ZevvPeep8x16);
        oled.clear();
        randomSeed(analogRead(A0));
        oled.setScroll(true); // ohne dies wird nur die letzte Zeile überschrieben //
        sensorValue = analogRead(A0);
        server.on("/", handleRoot);
        server.on("/inline", [](){
        server.send(200, "text/plain", "Funktioniert");
        });
        server.begin();
}
```

```
void loop() {
        delay(1000);
        sensorValue = analogRead(A0);
        tone(D6, sensorValue*3);
        server.handleClient();
        mdns.update();
        timenow = String(hour())+":"+twoDigits(minute())+":"+twoDigits(second());
        oled.print("\n"+timenow+" Wert"+String(sensorValue));
}

void handleRoot() {
        bewegung_html = bewegung_html + "<tr><td>" + timenow + " Wert" + "</td> <td>" +
        String(sensorValue) + "</td> </tr>";
        server.send(200, "text/html", bewegung_html);
}
```

Last step: End results & conlusions

What have we learned?

We have learned in this workshop how to

- Program a WeMos microcontroller, writing some simple sketches (programs)
- Measure an analog entity (here the local light intensity), display it on an LED ring array, convert it to sound and send it to a smartphone
- build some basic electronics, and connect some components by soldering
- construct a casing for the device using a laser cutter and wood

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 *Institute of Photonics Sciences, ICFO* in close collaboration with *Fablab Barcelona and Tinkerers Lab.*

JOANNEUM RESEARCH

FABLAB GRAZ

PHOTONICS21

PHOTONICS PUBLIC PRIVATE PARTNERSHIP

VUB B-PHOT BRUSSELS PHOTONICS

ICFO The Institute of Photonic Sciences

STEINBEIS 2i

FAB FACTORY

UNIVERSITY OF Southampton

Technische Hochschule Wildau Technical University of Applied Sciences

Leibniz Ferdinand Braun Institut

CENTER FOR PHYSICAL SCIENCES AND TECHNOLOGY

NUI Galway OÉ Gaillimh

JOANNEUM RESEARCH

CNR IFN Istituto di Fotonica e Nanotecnologie

TU Delft Delft University of Technology

UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Eyest vzw

ECOP European Centres for Outreach in Photonics