

'We are the makers - IoT' Learning Scenario: biofeedback with an IoT-health sensors

Author: Thomas Jörg, Johannes-Kepler-Gymnasium Weil der Stadt

The following paper was developed and tested in a school-environment with ca. 18 students of age 13-17 in the schoolyear 2018/2019. It reflects the experience with many meanders and some failures. Since the IoT-field is complex, teaching materials must be chosen carefully. This paper is supposed to be a recommendation, as a starting point.

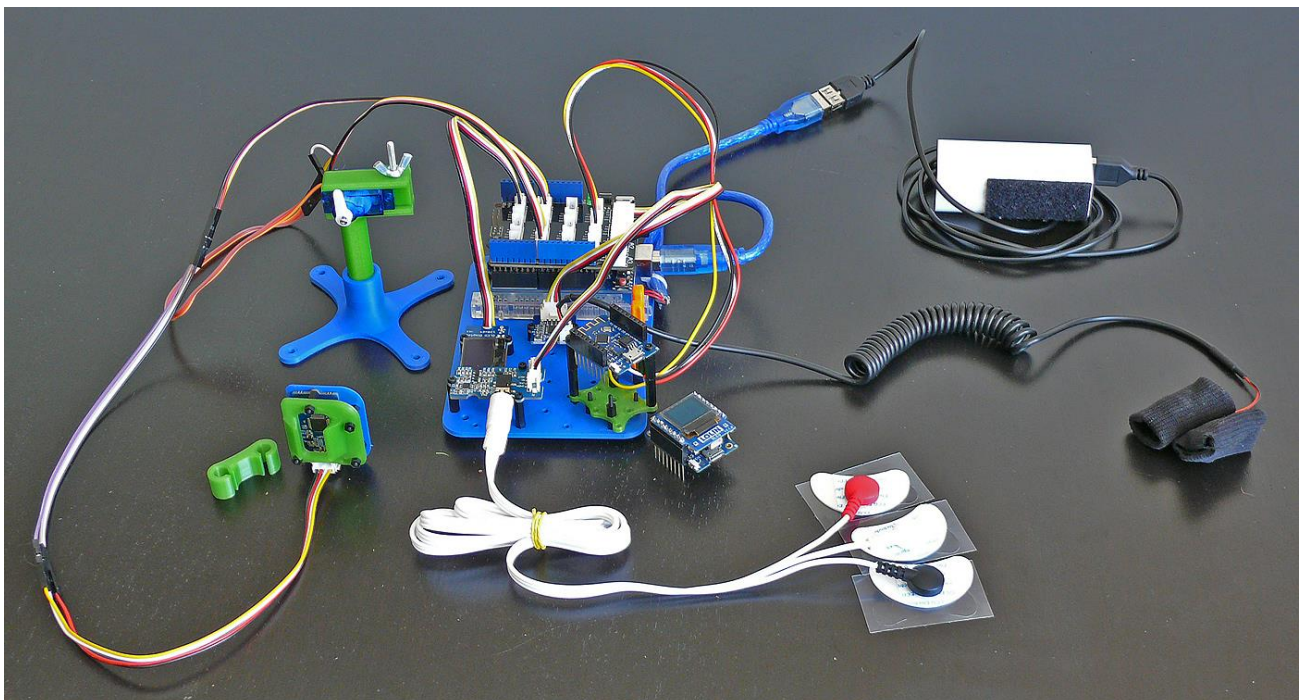
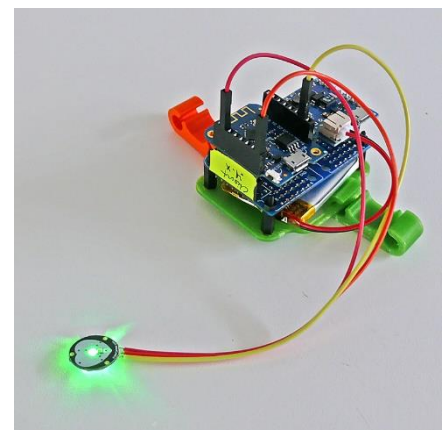


Figure 1: Prototype of a IoT biofeedback station

What about a self-made lie detector or a fitness tracker with self-programmed control software? Or a pulsometer which helps me to relax and keeps track of my strain?

What about a device which gives me feedback of my muscle tone and helps me to control myself in stress situations? Controlling the output of a machine with the power of my thoughts – possible or not?



And: what about sleep tracking devices which help to optimize my sleep and inform a physician if I'm not ok? Where are possibilities and threats of modern IoT biosensor technology?

1. Title of Scenario	Learn how to grow plants with the help of an IoT-plantrobot
2. Target group	14 - 17 years
3. Duration	At minimum 5 weeks of 2*45min-lessons per week: in sum about 6-8 hours.
4. Learning needs covered through the exercise	<ul style="list-style-type: none"> ▪ Interaction between electronic parts and human bodies ▪ Monitoring and affecting human biological parameters ▪ Communication chain of IoT-devices ▪ Principles of sensors and actors ▪ Different principles of measuring biosignals. ▪ EMG I: How does the muscle conduction system work? ▪ EMG II: Principles of Instrumentation Amplifiers ▪ Principles of wireless communication networks ▪ Construction and 3D-printing of helpers for measuring.
5. Expected learning outcomes	<ul style="list-style-type: none"> ▪ How does an IoT-system work? ▪ Where are possibilities and limitations of health-based IoT-systems? ▪ Which components – hard and software – are key to build an IoT-device? ▪ How to give biofeedback to help people?
6. Methodologies	In this scenario students will construct, build and program an interactive biosignal device from scratch by themselves. Students will also use the Arduino Serial Monitor and Serial plotter for visualizing and plotting biofeedback.
7. Place/ Environment	<ul style="list-style-type: none"> ▪ a laboratory with a set of electronic parts and components; ▪ each group of students need to have a computer or laptop with administrative privileges for installing different software packages ▪ A projector for teaching tutorials and presenting students works; ▪ each student has to keep a laboratory journal

8. Tools/ Materials/ Resources

3D-Printers

About 2-3 3D-printers are necessary since students will print their IoT-biofeedback-stations. Of course it's possible for the students to construct machine parts by themselves

3d printed components:

As a starting point, all necessary parts are provided in .stl-format and as Autodesk Fusion 360 Files.

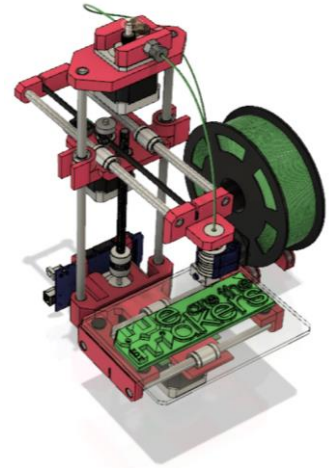


Figure 2: Symbol of 3D-Printer

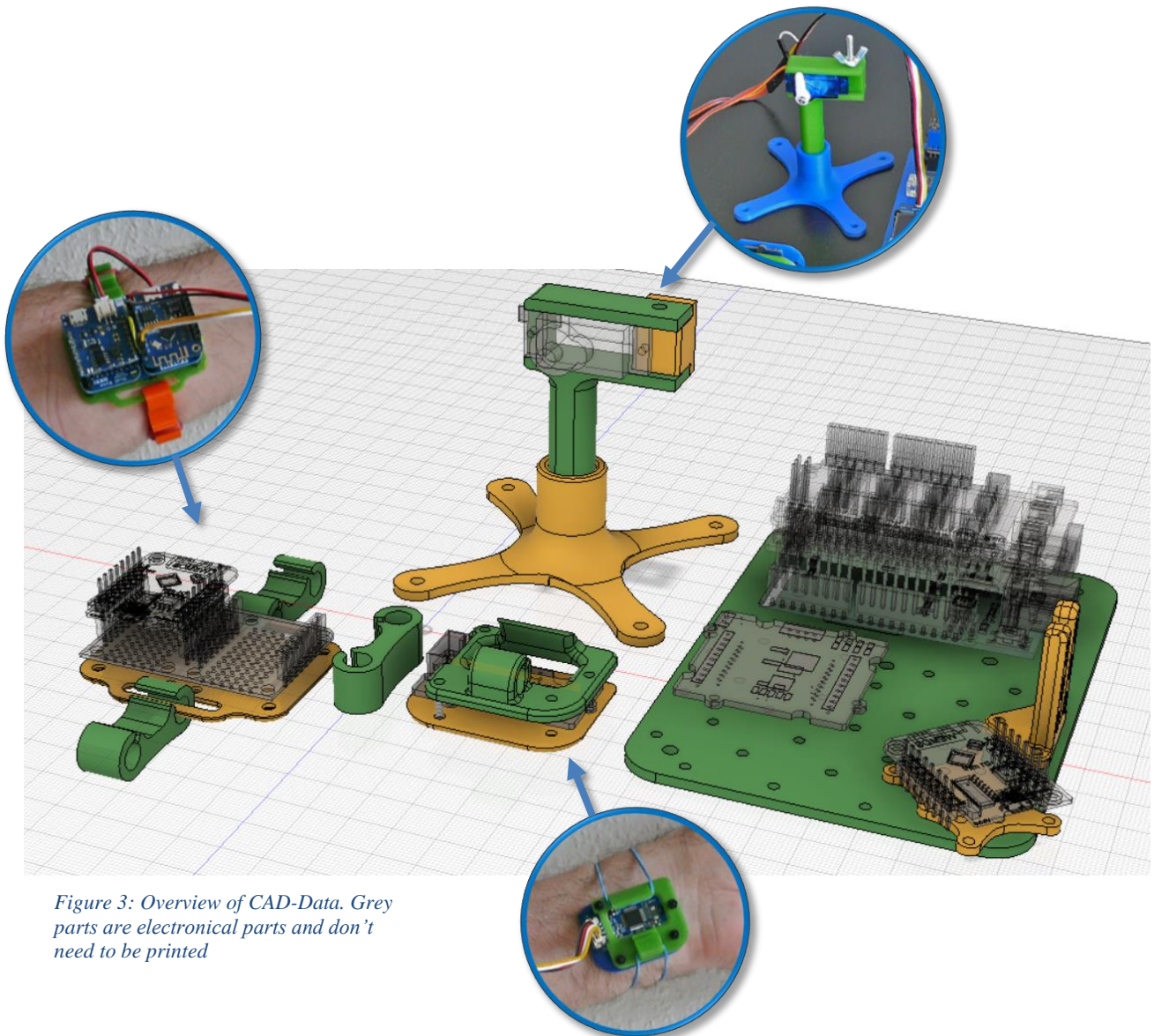


Figure 3: Overview of CAD-Data. Grey parts are electronical parts and don't need to be printed

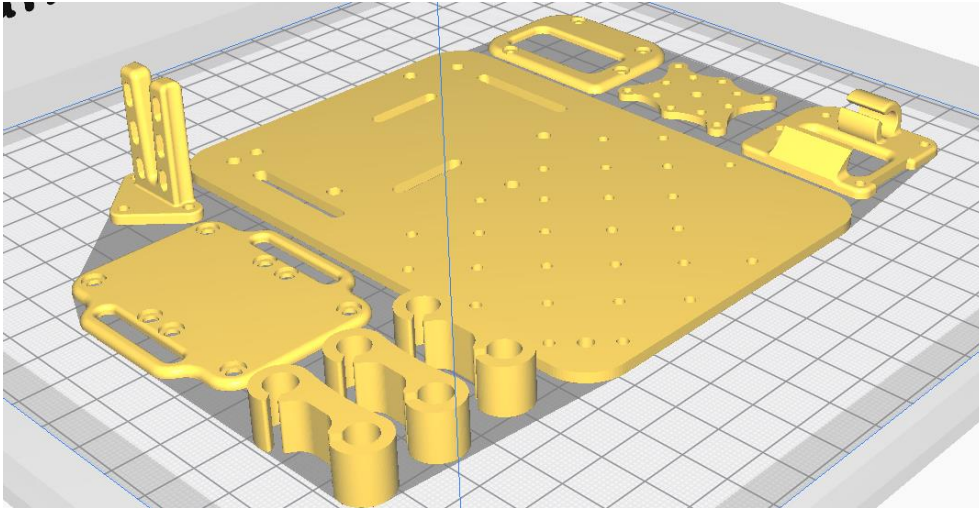
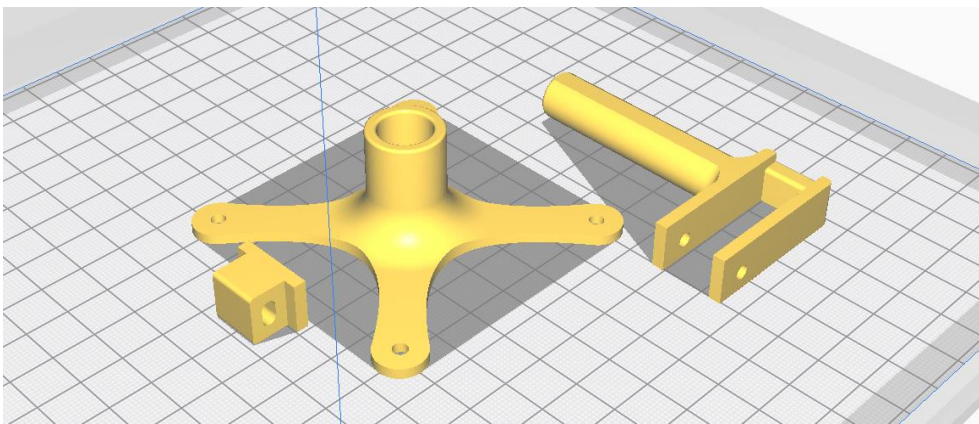


Figure 4: Set of .stl-files
ready for printing



Electronic components:

ATTENTION: Since we are making experiments with the human body, every precaution must be taken! Never connect a human body to the domestic power system. The human body must always be kept completely off the power grid !

This also includes AC adapters which are plugged into the wall socket. This kind of circuits must be avoided. Only use batteries and accumulators with low voltage of ca. 3-5V.

In this work, we recommend the Seeed Grove system as a basis since its ease of use:
(http://wiki.seeedstudio.com/Grove_System/) All core components except Wemos Chips, accumulators and heart-rate-sensors are belonging to the grove standard:

Seeed Studio Components:

- 1: Grove Base Shield for Arduino-Uno (http://wiki.seeedstudio.com/Base_Shield_V2/)
- 2: Grove OLED 128x64 (http://wiki.seeedstudio.com/Grove-OLED_Display_0.96inch/)
- 3: Grove EMG Detector (http://wiki.seeedstudio.com/Grove-EMG_Detector/)
- 4: Grove Finger Clip heart rate http://wiki.seeedstudio.com/Grove-Finger-clip_Heart_Rate_Sensor/
- 5: Grove GSR http://wiki.seeedstudio.com/Grove-GSR_Sensor/

Regular sensors and actors:

- 1: Arduino Uno (or equivalent)
- A: **2x** Wemos LOLIN D1 mini (or equivalent)
- B: Analog pulse finger sensor (www.pulsesensor.com)
- C: Wemos battery shield (https://wiki.wemos.cc/products:d1_mini_shields:battery_shield)
- D: Micro servo motor.

Miscellaneous parts:

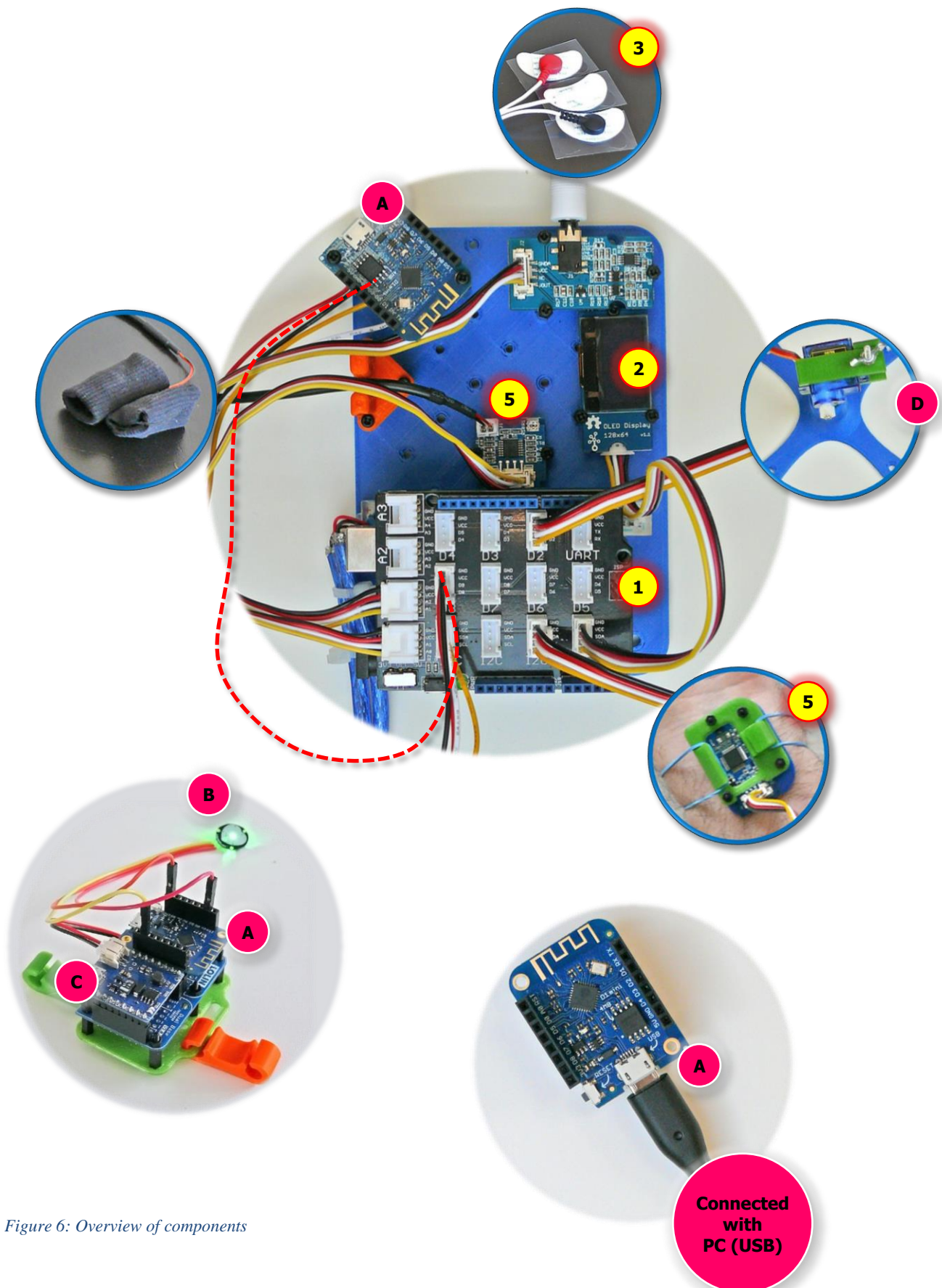
- Blister tape
- Self-adhesive copper band
- M3 Nylon Standoffs (Hex spacer)
- M2 Nylon Standoffs (Hex spacer) for Grove (has 2mm holes)
- Grove wires
- WAGO lever nuts
- Jumper wires
- USB power supply for charging the USB power pack
- USB power pack maximum current
- A Soldering Iron to attach cables to electronic components



Figure 5: Nylon Standoffs

computers with the following software preinstalled:

- Autodesk Fusion 360 (or any other 3D-modeling-Software, e.g. Wings3D)
- CURA slicing software,
- An internet connection for downloading libraries
- Arduino IDE
- Processing IDE



Arduino libraries for components:

Some components like the Wemos D1 Mini or some of the Grove boards need libraries for the Arduino IDE to work properly. How to import a library is described here:

<https://www.arduino.cc/en/Guide/Libraries>

OLED lib (Seeed):

https://github.com/Seeed-Studio/OLED_Display_128X64/archive/master.zip

Software I2C Master (Felix Fogg):

<https://github.com/felix-fogg/SoftI2CMaster>

Preference URL for WEMOS-Boards (ESP8266):

To install the wemos, the so-called "board-definition" needs to be installed. It is described here:

<http://arduino.esp8266.com/Arduino/versions/2.0.0/doc/installing.html>

1. Inside the Arduino IDE open Preferences window.
2. Enter the following URL into "Additional Board Manager" field:
http://arduino.esp8266.com/stable/package_esp8266com_index.json
3. Open Boards Manager from Tools > Board menu and find esp8266 platform.
4. Select the current version from a drop-down box and click the "install" button.
5. Select "(LOLIN)Wemos D1 R2 and Mini" from Tools > Board menu after installation.

Pulsesensor:

<https://pulsesensor.com/pages/installing-our-playground-for-pulsesensor-arduino>

The Grove GSR, Grove Finger Clip Sensor and the Grove EMG Detector don't need any libraries, since they can be controlled with simple Arduino commands.

Wemos D1 mini as wireless connection between electronic components

- **Wemos boards should be prepared by the teacher not by the students before the lesson starts!**
- Wemos-ESP8266-Wifi-Boards are intended as a less expensive alternative to the reliable but also costly Xbee technology.

Two Wemos are building a pair which is connected via Wifi ethernet port 23 (which is Telnet). The only purpose is to **replace the serial communication cable**. Usually an experimental electronic device is connected via USB cable with the PC. To achieve a completely autonomous design which is not connected to the domestic power system, a wireless connection has to be established.

Therefore, the usual Serial communication (UART) is translated to Wifi and sent by one Wemos, received by the other Wemos and re-translated to Serial communication again. For compatibility reasons, the baud rate is fixed to 9600 baud, since Software-Serial-Communication by an Arduino Uno is limited to 9600 baud.

A Wemos D1 mini pair consists of a Server and a client. The server should be connected to the PC. It should be started at first and is doing the following steps:

1. Scanning of all available wifi networks,
2. Determining, if there is an unused, free channel or a weak network in the background,
3. Establishing a Wifi Access point using the first free channel, also combined with DHCP
4. Waiting for ONE (only one!) Client which connects.
5. If Client disconnects, server will wait until client reconnects.
6. If Server is reset, just begin at 1. (scanning networks)

The client should be started as the second one and will automatically connect and reconnect.

How to configure the Wemos Server and Client, explained on "Better Server sourcecode":

Here are the relevant excerpts from server- and client-sourcecode which have to be adapted for configuring individual pairs of Wemos-boards:

```
#include <ESP8266WiFi.h>

const char *ssid = "Erasmus";
const char *password = "12345678";
IPAddress Ip(192, 168, 3, 1);
IPAddress NMask(255, 255, 255, 0);

WiFiServer server(23);
WiFiClient serverClient;
char inChar;
```

Figure 7: cutting of server sourcecode

```
#include <ESP8266WiFi.h>

const char* ssid = "Erasmus";
const char* password = "12345678";
IPAddress server(192, 168, 3, 1);

WiFiClient client;
char inChar;
```

Figure 8: cutting of client sourcecode

- Both underlined lines of code need to be exactly the same for one Wemos pair.
- Both underlined lines of code must be adapted for every single Wemos pair.

Change the **IP-adress** to

192.168.1.1 OR 192.168.2.1 OR 192.168.4.1 OR 192.168.5.1 ...etc.

Change the **ssid** to

"Erasmus1" OR "Erasmus2" OR "Erasmus4" OR "Erasmus5" ...etc.

... compile the scripts inside the Arduino IDE and upload them to the appropriate Wemos boards.

8b Some theory of Axone potentials and EMG Measuring

This text is intended as a short overview and can be considered as a collection of important keywords. It is not intended as a textbook!

The following article is based on the "EMG Fibel V1.1.pdf", "and the Wikipedia-entries to the keywords "Action Potential", "Design of an EMG Detector", "EMG" at the 'electronic compendium':

- <http://www.velamed.com/wp-content/uploads/EMG-FIBEL-V1.1.pdf>
- https://en.wikipedia.org/wiki/Action_potential
- <https://iem.kug.ac.at/sid/sonic-interaction-design/forschung/hardware-software-prototyping/design-and-evaluation-of-an-emg-based-recording-and-detection-system.html>
- <https://www.elektronik-kompodium.de/public/schaerer/emg1.htm>

Muscles are contracting since they are getting electrical signals from nerves: The so called "Action Potential" is an electric potential change of intramuscular nerve-endings. The change occurs between -80mV ("Resting Potential") to +30mV ("Depolarisation"), which makes in total a theoretically measurable potential change of 110mV. Since nerve-endings are inside the muscle fibres, (at school) we can only measure on top of the skin. As a consequence, much of the action potentials signal strength vanishes caused by the electrical resistance of skin and connective tissue. A typically measurable change in electrical potential is ca. 30mV.

Microcontrollers which are able to compute these signals are using Analog-Digital-Converters. Those ADCs usually have an input voltage range of 0-3.3V (Wemos-Type) or 0-5V (Arduino Uno-Type). The resolution of both ADCs is 10bit, which means the microcontrollers can divide 3.3V full measuring range in 1024 discrete steps: $3.3V / 2^{10} = 3300mV / 1024 = 3,2mV$. If we would measure changes in skin potential with a lone standing microcontroller we only could achieve a range between 0 and 10 of 1024 possible discrete values. This is far to less. In Addition, skin potential of the human body changes with the influence of external electrical fields, which leads to potential drifts 1000 times bigger than the signal strength.

Therefore, we need an upstream electronical component which a) amplifies our signal from 0.03V to 3,3V and b) is able to compensate the electrical field drift. The so-called instrumentation amplifier is a circuit with three inputs: One input for '+', one for '-' and one for reference. While bot '+' – and '-'-electrodes' purpose is to measure the 0.03V potential difference, the reference electrode will take care of the influence of external electrical fields to compensate drift. Inside this electronical precision instrument, the signal is amplified and that's simply all it does! Now we can measure and digitalize the output potential with the microcontroller.

The electrodes need to be placed carefully, since it should measure nerve potentials, and the shorter the distance to the nerve ends the better the signal: A nerve signal travels through the muscle with a speed of ca. 5m/s. If we measure in the midst of the biceps, both +/- - electrodes should be placed ca. 10cm away from each other, while the reference electrode should be placed at a point far away from there, e.g. on the hand. As an expected measurement there should be a signal transit time from electrode to electrode of ("depolarisation wave"):

$$\frac{\text{distance of electrodes in cm}}{\frac{500 \text{ cm}}{\text{second}}} = 20 \text{ milliseconds}$$

In an EMG skin measurement, we will see an overlay of many signals from many different muscle fibers and therefore a waveform of 20ms will be hardly recognizable.

To contract a muscle with variable strength the value of the action potential does not change.

There is only a change of occurrence of nervous voltage changes: The harder a muscle should contract the higher the 'fire rate' of Action Potentials of the nerves.

8c Some theory of Biofeedback

This text is intended as a short overview and can be considered as a collection of important keywords. It is not intended as a textbook!

<https://en.wikipedia.org/wiki/Biofeedback>

A human body's respond to stress or external influences happens most of the time automatically and unconsciously. As an example, if a human lies or is in fear his skin begins to sweat. This sweat can be measured as a change in electrical conductivity since sweat contains electrolytes. If the measuring computer visualizes this change, the human can correlate his emotional state with the measured signal and can try to influence his reaction and learn how to control his emotions. The preceding hidden emotions became now conscious to this persons mind.

There are many examples & experiments students can try by themselves:

- Influence heart rate with change of respiratory rate, monitored by pulse sensors
- Influence fear reactions with change in electrodermal activity, monitored by GSR sensors (A fear reaction could be: a picture of a spider, a youtube video of a roller coaster)
- A polygraph (lie detector) is – among other things – based on the change of electrodermal activity and can be measured with GSR sensors
- Coactivity of muscles: Computer typing under stress conditions leads to contraction of trapezius muscle in humans neck. This can be measured with EMG.

9. Lesson plan: Step by step description of the activity/ content

Lesson 1 & 2 (90min):

Students will be introduced IoT by examples: Vacuum robots with app remotes, internet-based weather stations, smart farming and last but not least health applications. Students should examine how those devices work and which components are needed: a microcontroller based system controls and coordinates attached sensors and actors. Furthermore it communicates and coordinates with other systems of similar type often via wireless communication networks. Parts needed: Sensors, Actors, Communication devices. Possibilities and threats need to be discussed and also limitations: where does IoT make sense and where not?

Lesson 3&4 (90 min)

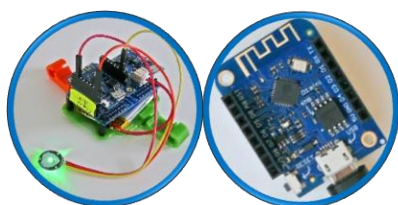


Figure 9: Necessary parts

Biosensor basics (pulse sensor): Students are presented the first sensor which doesn't need to be galvanically isolated: The pulse-sensor. It consists of two different parts: a simple green LED and a phototransistor plus directly attached amplifier circuit. It needs to be placed directly above a vein, e.g. the fingertips or ear tips.

If the vein is inflated since the heart is pumping, the blood will reflect the green light and the phototransistor will recognize a high value. If the vein contracts since the heart contracts, the missing blood will lead to greater absorption of light inside the connective tissue. The phototransistor will measure a smaller value.

Students should be introduced to Arduino analog programming and analog value computing: the analog-to-digital converter (ADC) has to be introduced and how it works. A very good introduction can be found here:

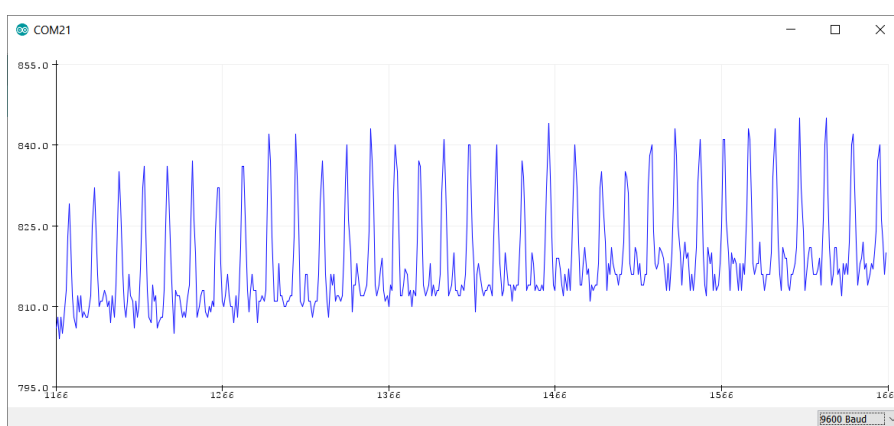
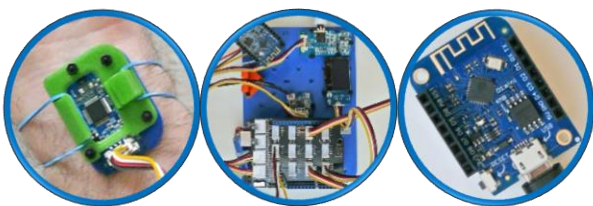


Figure 10: Sample data measured by pulse sensor

<https://www.generationrobots.com/media/DetecteurDePoulsAmplifie/PulseSensorAmpedGettingStartedGuide.pdf>

Lessons 5&6 (90 min)



Biofeedback of the heart rate: Students learn that many body reactions correspond to emotional processes unconsciously. But if we get access to these hidden body reactions we can start to control them: Biofeedback is a real time monitoring of

one's physical responses with the aim of achieving control of emotions

(<https://www.artofmanliness.com/articles/hack-your-mind-like-a-twenty-first-century-soldier-using-biofeedback-to-become-more-resilient/>).

If the pulse sensor software will work properly, students can begin to measure pulse changes and try to influence their heart rate while breathing: What is the influence of respiratory rate to the heart rate? What happens, if we breath faster or slower? What does it mean for stress situations? Can we get conscious about our heart rate?

Use this setup procedure for a biofeedback experiment:



Figure 11: Screenshot Processing App: Influence of brightness / contrast with heart rate

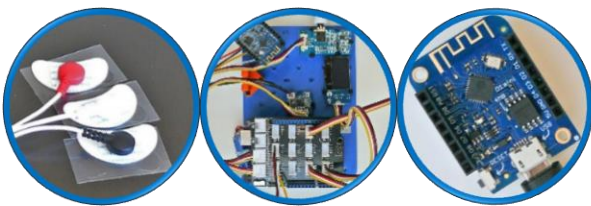
- a) I2C Grove heart rate finger clip attached to one finger of student's hand
- b) Connect the Arduino station with PC and start the processing source code. The faster the heart rate the darker the image: Influence brightness/contrast with mind control of heart's activity.

c) Do the following:

- Slowly inhale a deep breath for 4 seconds.
- Hold the breath in for 4 seconds.
- Slowly exhale the breath out for 4 seconds.
- Hold the empty breath for 4 seconds.
- Repeat until your breathing is under control.

([https://en.wikipedia.org/wiki/Dave_Grossman_\(author\)](https://en.wikipedia.org/wiki/Dave_Grossman_(author)))

Lessons 7&8



EMG Basics: Teaching the keywords and relationships of Active potential of Nerves, muscle strain and relaxation and measuring principles. Students should understand how the sourcecode of muscle signal amplification works.

How to place the three electrodes at the arm: At first, the test person has to wash the arm with soap and water and afterwards those positions where the electrodes will be placed have to be cleaned with alcohol and cotton wool. Then all three electrodes can be placed.

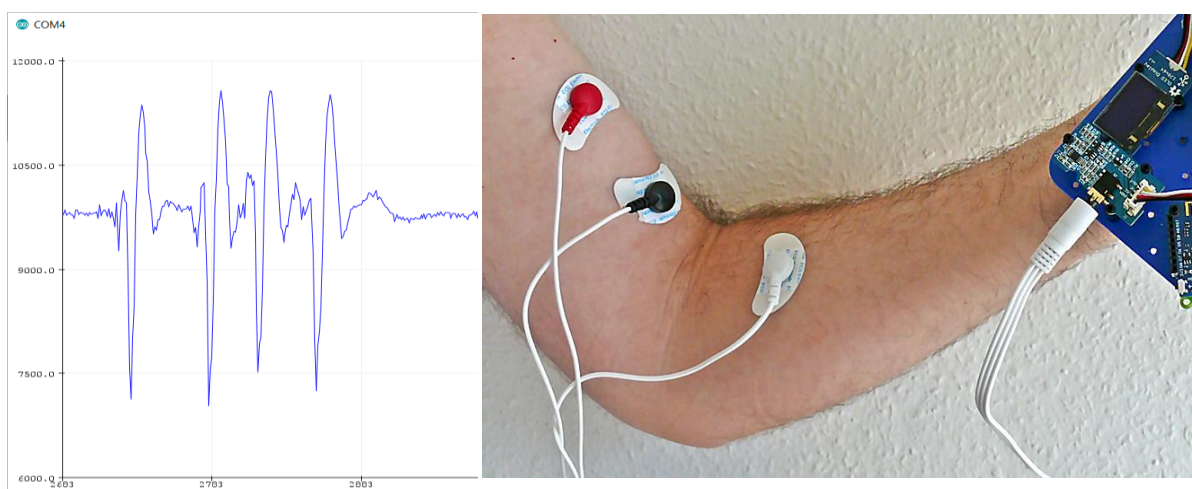
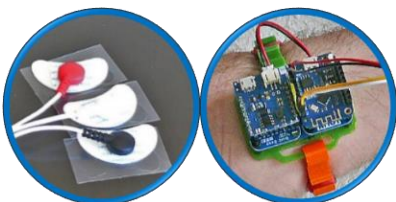


Figure 12: Biceps activity and corresponding electrode placement

Students should experiment with different speeds of arm movement, different lifting weights and the effect of relaxation. What happens if – with exactly this electrode placement – one will close and open his hand? What happens if one's removing the white electrode?

Lessons 9&10



EMG Biofeedback experiment:

The following experiment is based on the PhD thesis of Michael Schnoz:

<https://www.research-collection.ethz.ch/handle/20.500.11850/149225>

If a test person will type letters on the computer screen fast and highly concentrated it will probably lead to aching in the



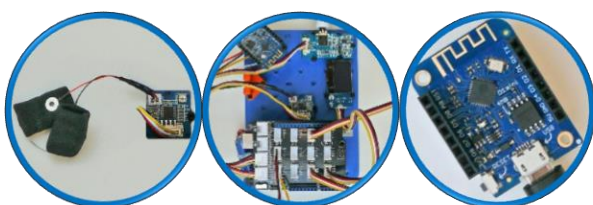
Figure 13: placement of electrodes in the neck

test persons neck which can be measured and afterwards influenced by the test person itself. To put the test person under stress conditions, he or she could use the following typewriting tutorial software:

https://portableapps.com/apps/education/tipp10_portable

The faster the person works the more tense the person – and probably his neck – will get. The test person can now try to influence this tense.

Lessons 11&12



Biofeedback GSR

<https://www.youtube.com/watch?v=ZultgAFrxuc>

This lesson is based on emotional reactions with viewing a "scary movie": The up- and downs of a roller coaster can have a huge effect on the test persons feelings. How to influence it?

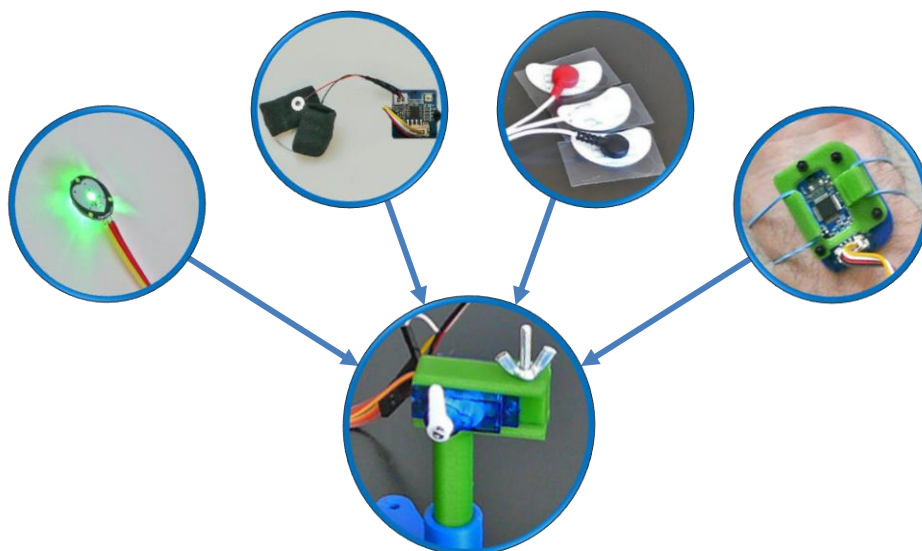
What about some pictures of spiders or snakes?

Or of something delightful / pleasing like music? What is the effect of disco music/classic music? Is there a special effect while hearing your favourite song?

Lessons 13 to finish (270 min):

Freestyle programming! And happy biofeedback! 😊

Try to combine the output of the different sensors with the servo motor. Which possibilities do you have to control the servos rotation angle?



10. Feedback	<p>At the end of the lesson, students should have a well-grounded knowledge of how IoT principles in medical devices work and how biofeedback can help with understanding our body's hidden features. During the lesson, important aspects of electronics, medical informatics and construction basics have been tutorised. Furthermore, biological aspects of muscle activities have been taught.</p>
11. Assessment & Evaluation	<p>Students keep their labor journal, which can be reviewed by the teacher. Students can also present the results of their experiments. In addition, a standard in-class-test has to be conducted at the end of the lessons.</p>

Wemos Client Sourcecode

```
#include <ESP8266WiFi.h>

const char* ssid      = "Erasmus";
const char* password = "12345678";
IPAddress server(192, 168, 3, 1);

WiFiClient client;
char inChar;

void setup() {
  Serial.begin(9600);
  WiFi.setSleepMode(WIFI_NONE_SLEEP);
  WiFi.mode(WIFI_STA);
  WiFi.setOutputPower(10); // 10: 10mW, 14: 25mW, 17: 50mW, 20: 100mW
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(5);
  }

  Serial.print("WiFi Channel: ");
  Serial.println(WiFi.channel());

  if (client.connect(server, 23)) {
    Serial.print("Local IP: ");
    Serial.println(WiFi.localIP());
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, LOW);
  }
}

void loop() {
  if (!client.connected()) {
    digitalWrite(LED_BUILTIN, HIGH);
    unsigned long startzeit = micros();
    client.connect(server, 23);
    Serial.println(micros() - startzeit);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }

  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  while (Serial.available() > 0) {
    inChar = Serial.read();
    if (client.connected()) {
      client.write(inChar);
      delay(1);
    }
  }
}
```

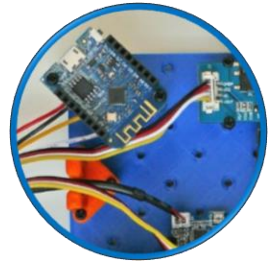
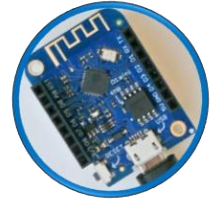


Figure 14: Source code for the Wemos client attached to the Arduino station

Wemos Server Sourcecode

Figure 15: This script should be compiled for the Wemos connected with PC



```
#include <ESP8266WiFi.h>

const char *ssid = "Erasmus";
const char *password = "12345678";
IPAddress Ip(192, 168, 3, 1);
IPAddress NMask(255, 255, 255, 0);

WiFiServer server(23);
WiFiClient sClient;
char inChar;

void setup() {
    Serial.begin(9600);
    unsigned int c_frei = SSID_scan();
    Serial.println("Configuring access point");
    WiFi.softAPConfig(Ip, Ip, NMask);
    WiFi.softAP(ssid, password, c_frei, false,
1);
    Serial.print("Channel: ");
    Serial.println(c_frei);

    Serial.println("Starting server");
    server.begin();
    server.setNoDelay(true);

    Serial.print("Server IP: ");
    Serial.println(WiFi.softAPIP());
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, HIGH);
}

void loop() {
    uint8_t i;
    if (server.hasClient()) {
        if (!Client || !sClient.connected()) {
            if (sClient) sClient.stop();
            sClient = server.available();
            digitalWrite(LED_BUILTIN, LOW);
        }
        else digitalWrite(LED_BUILTIN, HIGH);
        if (sClient.available()) {
            digitalWrite(LED_BUILTIN, LOW);
            while (sClient.available()) {
                inChar = sClient.read();
                Serial.write(inChar);
            }
        }
        else digitalWrite(LED_BUILTIN, HIGH);

        if (Serial.available()) {
            size_t len = Serial.available();
            uint8_t sbuf[len];
            Serial.readBytes(sbuf, len);
            if (sClient.connected()) {
                sClient.write(sbuf, len);
                Serial.write(sbuf, len);
            }
        }
    }
}

int SSID_scan() {
    int frei = 0;
    Serial.println("scan start");
    WiFi.disconnect();
    delay(100);
    int n = WiFi.scanNetworks();
    if (n == 0) {
        Serial.println("no networks found");
        frei = 1;
    } else {
        int belegt[n];
        int staerke[n];
        Serial.print(n);
        Serial.println(" networks found.");
        for (int i = 0; i < n; ++i) {
            belegt[i] = WiFi.channel(i);
            staerke[i] = WiFi.RSSI(i);
            delay(10);
        }
        for (int i = 0; i < 12; ++i) {
            int diff = belegt[i + 1] - belegt[i];
            if (diff > 1) {
                frei = belegt[i] + 1;
                break;
            }
        }
        if (frei != 0) {
            Serial.print("done. free channel: ");
            Serial.println(frei);
            return frei;
        } else {
            int maxnummer = 0;
            int maxstaerke = staerke[maxnummer];
            for (int j = 0; j < n; j++) {
                if (maxstaerke > staerke[j]) {
                    maxnummer = j;
                    maxstaerke = staerke[maxnummer];
                }
            }
            frei = belegt[maxnummer];
            Serial.print("done. weakest channel: ");
            Serial.println(frei);
            return frei;
        }
    }
}
```

EMG Sourcedcode example

```
#include <SeeedOLED.h>
#include <Wire.h>
#include <SoftwareSerial.h>
SoftwareSerial Serial_89(8, 9);

int max_analog_dta      = 300; // max analog data
int min_analog_dta      = 100; // min analog data
int static_analog_dta   = 0;   // static analog data
int level = 5;

int getAnalog(int pin) {
    long sum = 0;
    for (int i = 0; i < 32; i++){
        sum += analogRead(pin);
    }
    Serial.println(sum);
    Serial_89.println(sum);
    int dta = sum >> 5;
    max_analog_dta = dta > max_analog_dta ? dta : max_analog_dta;
    min_analog_dta = min_analog_dta > dta ? dta : min_analog_dta;
    return sum >> 5;
}

void setup() {
    Wire.begin();
    Serial.begin(9600);
    Serial_89.begin(9600);
    SeeedOled.init();
    SeeedOled.clearDisplay();
    SeeedOled.setNormalDisplay();
    SeeedOled.setPageMode();
    SeeedOled.setTextXY(1, 0);
    SeeedOled.putString("EMG prototype");
    long sum = 0;

    for (int i = 0; i <= 10; i++) {
        for (int j = 0; j < 100; j++){
            sum += getAnalog(A0);
            delay(1);
        }
    }
    sum = sum / 1100;
    static_analog_dta = sum;
    Serial.print("static_analog_dta = ");
    Serial.println(static_analog_dta);
}

void loop() {
    int val = getAnalog(A0);
    int level2;
    if (val > static_analog_dta) {
        level = 5 + map(val, static_analog_dta, max_analog_dta, 0, 10);
    } else {
        level = 5 - map(val, min_analog_dta, static_analog_dta, 0, 10);
    }
    for (int i = 0; i < 10; i++) {
        SeeedOled.setTextXY(1, i);
        SeeedOled.putChar(32);
    }
    for (int i = 0; i < level - 5; i++) {
        SeeedOled.setTextXY(1, i);
        SeeedOled.putChar(124);
    }
    delay(20);
}
```

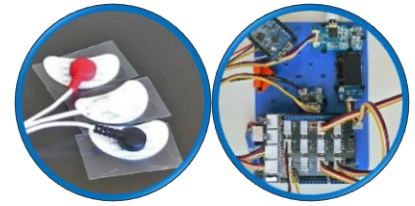


Figure 16: Script for lessons 7&8 at the Arduino Station

Pulse-Sensor on Wemos Client using AnalogRead

```
#include <ESP8266WiFi.h>

const char* ssid      = "Erasmus";
const char* password = "12345678";
IPAddress server(192, 168, 3, 1);

WiFiClient client;
char inChar;

void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT);
  WiFi.setSleepMode(WIFI_NONE_SLEEP);
  WiFi.mode(WIFI_STA);
  WiFi.setOutputPower(10); // 10: 10mW, 14: 25mW, 17: 50mW, 20: 100mW....
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(5);
  }

  Serial.print("WiFi Channel: "); Serial.println(WiFi.channel());

  if (client.connect(server, 23)) {
    Serial.print("Local IP: ");
    Serial.println(WiFi.localIP());
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, LOW);
  }
}

void loop() {
  if (!client.connected()) {
    digitalWrite(LED_BUILTIN, HIGH);
    unsigned long startzeit = micros();
    client.connect(server, 23);
    Serial.println(micros() - startzeit);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }

  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
  delay(50);
  if (client.connected()) {
    unsigned int value = analogRead(A0);
    Serial.println(value);
    String einsnachdemandern = String(value);
    for (int i = 0; i < einsnachdemandern.length(); i++) {
      client.write(einsnachdemandern[i]);
    }
    client.write(10);
  }
}
```

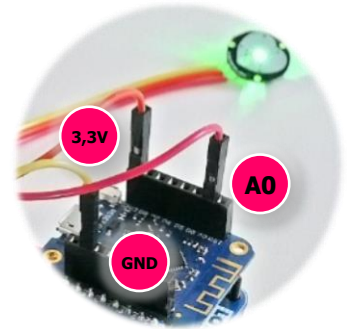


Figure 17: Wiring Wemos - Pulse Sensor

GSR Source code on Station client

```
#include <SeeedOLED.h>
#include <Wire.h>
#include <SoftwareSerial.h>
SoftwareSerial Serial_89(8, 9);

const int GSR = A1;
int sensorValue = 0;
int gsr_average = 0;

void setup() {
  Wire.begin();
  Serial.begin(9600);
  Serial_89.begin(9600);
  SeeedOled.init();
  SeeedOled.clearDisplay();
  SeeedOled.setNormalDisplay();
  SeeedOled.setPageMode();
  SeeedOled.setTextXY(1, 0);
  SeeedOled.putString("GSR prototype");
  delay(2000);
}

void loop() {
  long sum = 0;
  for (int i = 0; i < 20; i++)
  {
    sensorValue = analogRead(GSR);
    sum += sensorValue;
    delay(5);
  }
  gsr_average = sum / 10;
  Serial.println(gsr_average);
  Serial_89.println(gsr_average);
  SeeedOled.clearDisplay();
  SeeedOled.setTextXY(1, 2);
  SeeedOled.putNumber(gsr_average);
}
```

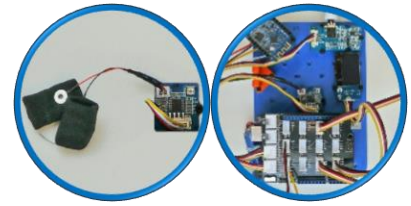


Figure 18: Source code for Arduino station lessons 11&12

Lesson 5&6 source code with processing app

Arduino sourcecode for Grove Finger clip sensor	Processing sourcecode for Grove Finger clip Sensor:
	
<pre> #define SDA_PORT PORTD #define SDA_PIN 3 #define SCL_PORT PORTD #define SCL_PIN 2 #include <SoftI2CMaster.h> #include <SoftWire.h> #include <SseedOLED.h> #include <Wire.h> #include <SoftwareSerial.h> SoftwareSerial Serial_89(8, 9); SoftWire SWire = SoftWire(); void setup() { Wire.begin(); SWire.begin(); Serial.begin(9600); Serial_89.begin(9600); SseedOled.init(); SseedOled.clearDisplay(); SseedOled.setNormalDisplay(); SseedOled.setPageMode(); SseedOled.setTextXY(1, 0); SseedOled.putString("heartrateprototype"); delay(2000); } void loop() { SWire.requestFrom(0xA0 >> 1, 1); while (SWire.available()) { unsigned char c = SWire.read(); Serial.println(c, DEC); Serial_89.println(c, DEC); SseedOled.clearDisplay(); SseedOled.setTextXY(1, 2); SseedOled.putString(c); } delay(500); } </pre>	<pre> PImage img; int heartrate; import processing.serial.*; Serial ardCom; String payload; String[] liste; void setup() { String portName = Serial.list()[0]; ardCom = new Serial(this, portName, 9600); size(1000, 500); frameRate(20); img = loadImage("biofeedback.png"); image(img, 0, 0); loadPixels(); } void draw() { if (ardCom.available() > 0) { payload = ardCom.readStringUntil('\n'); if (payload != null) { heartrate = parseInt(payload.trim()); println(heartrate); } } float percentage = -0.7/40*heartrate + 2.15; for (int x = 0; x < img.width; x++) { for (int y = 0; y < img.height; y++) { int loc = x + y*img.width; float r, g, b; r = red (img.pixels[loc]); g = green (img.pixels[loc]); b = blue (img.pixels[loc]); float bright = 64 - 64*percentage; float cont = percentage; r = r*cont - bright; g = g*cont - bright; b = b*cont - bright; r = constrain(r, 0, 255); g = constrain(g, 0, 255); b = constrain(b, 0, 255); color c = color(r, g, b); pixels[y*width + x] = c; } } updatePixels(); } </pre>