Co-funded by the
Erasmus+ Programme
of the European Union

‘We are the makers - IoT’ Learning Scenario:
EDA-Cube: get an idea of how your partner feels’

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

The following paper was developed as a derivative of the biofeedback station (IO2). As the biofeedback station was built with general education in mind, the EDA-cube can be used for non-technical purposes as for example a tool for monitoring dialogues: It reveals emotions of its user as far as EDA-technology allows.

Figure 1: EDA-cubes in action

What if two people talk together and want to be polite to each other. A big problem always is you don’t know how your dialogue partner feels. Is he/she angry, frightened, nervous, happy, tired? What are the effects of your words? Are you – in your partner’s eyes – rude or thoughtful?
Is there an immediate answer to this, at least a ‘hint’?

And: What if you are wearing a smartwatch which is also capable of measuring your emotional reactions? Do you want your feelings monitored by a machine? Be aware of the capabilities of modern sensorics! Let’s build an intelligent device with an ‘emotion’-sensor which will interpret your bodies reactions!

1. Title of Scenario	**Learn how to grow plants with the help of an IoT-plant robot**
2. Target group	12- 17 years
3. Duration	At minimum 3 weeks of 2*45min-lessons per week: in sum about 6-8 hours.
4. Learning needs covered through the exercise	▪ Interaction between electronic parts and human bodies ▪ Monitoring and affecting human biological parameters ▪ Communication chain of IoT-devices ▪ Principles of sensors and actors ▪ What is EDA? ▪ Principles of wireless communication networks ▪ Construction and 3D-printing of helpers for measuring.
5. Expected learning outcomes	▪ How does an intelligent IoT-system work? ▪ Where are the possibilities and threats of EDA systems? ▪ Which components – hard and software – are key to build an IoT-device? ▪ Getting awareness of biosensorics and the ability to judge these tools.
6. Methodologies	In this scenario students will construct, build and program an interactive EDA device from scratch by themselves which interprets the measured values and visualizes them. Students can use Wi-Fi technology to pass the values to a computer for furthermore processing.
7. Place/ Environment	▪ 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;

8. Tools/ Materials/ Resources

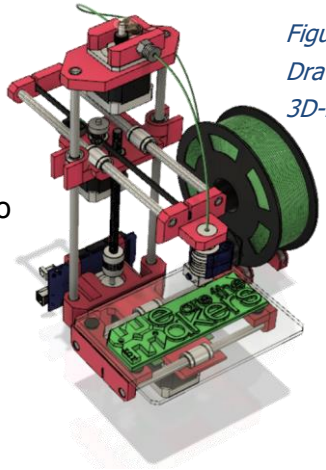


Figure 2: Drawing of a 3D-Printer

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.

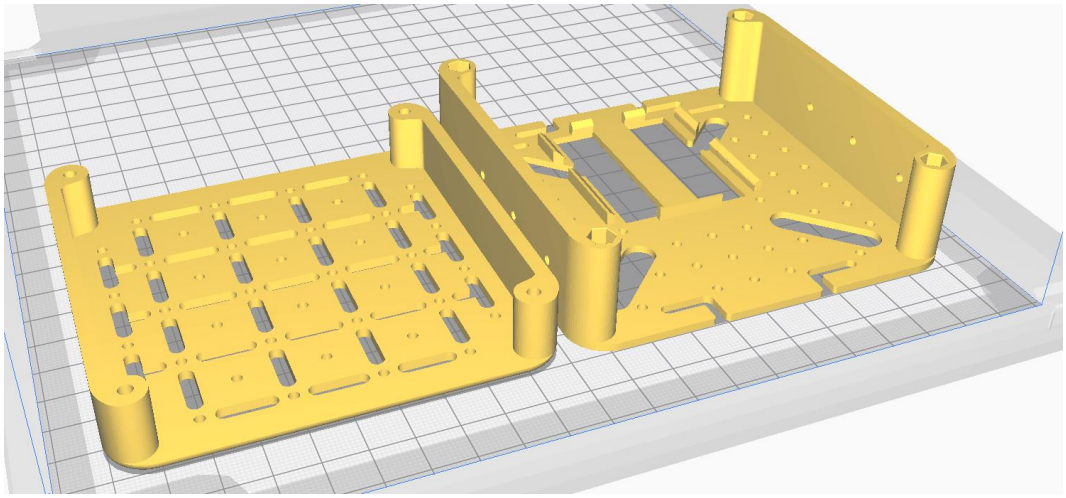


Figure 3: STL data of regular PLA parts

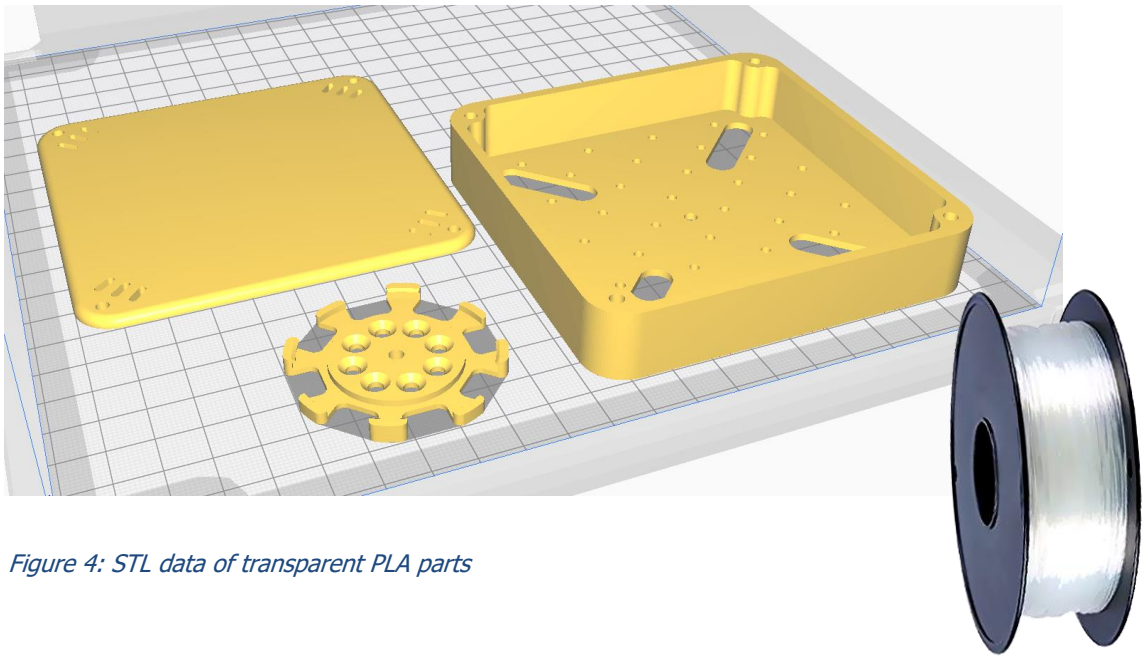


Figure 4: STL data of transparent PLA parts

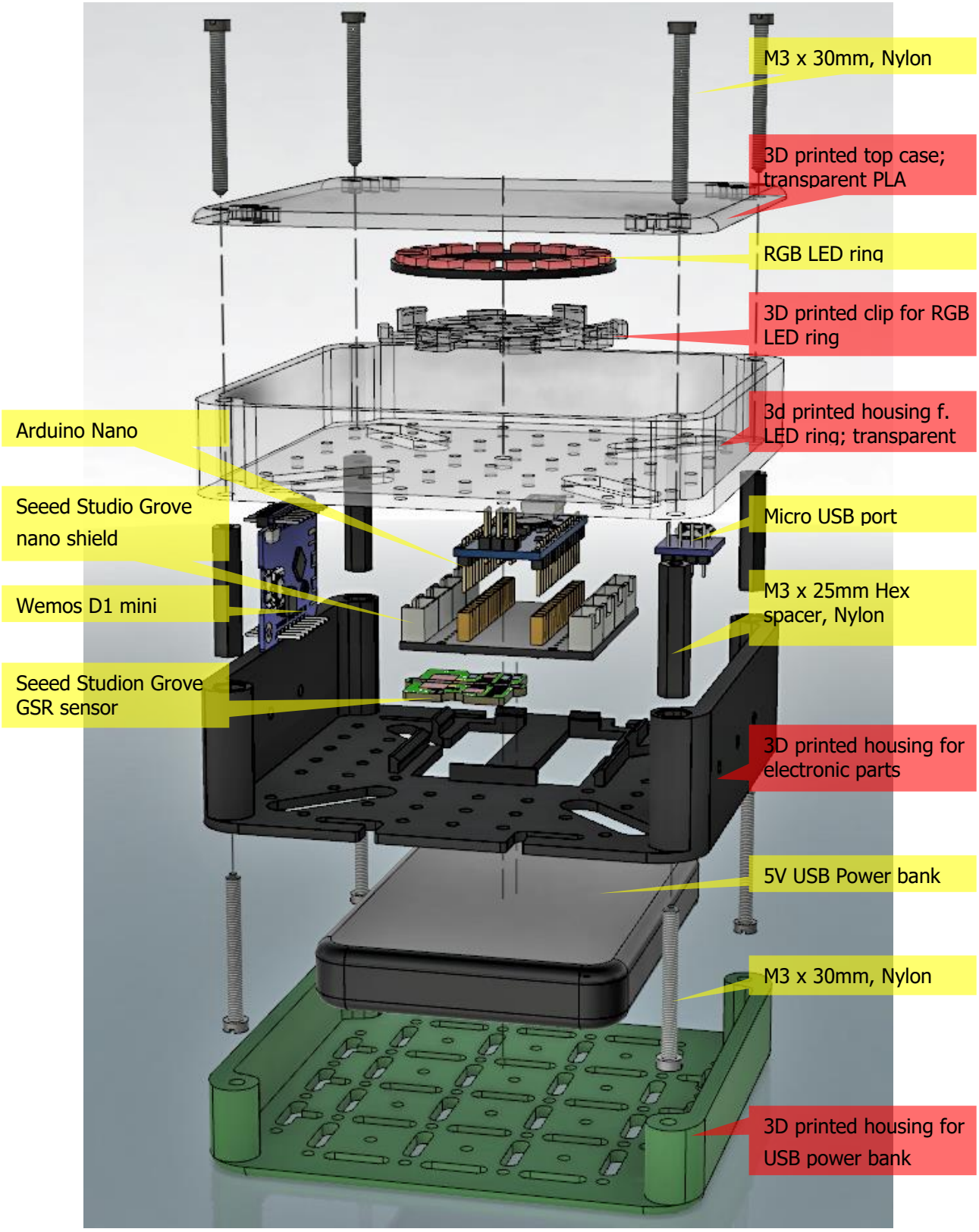


Figure 5: Exploded assembly drawing of the cube

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.

9. Setup components

In this work, we recommend the Seeed Grove system as a basis since its ease of use: (http://wiki.seeedstudio.com/Grove_System/):

Seeed Studio Components:

1x Grove Shield for Arduino Nano

https://www.seeedstudio.com/Grove-Shield-for-Arduino-Nano-p-4112.html

1x Grove GSR

http://wiki.seeedstudio.com/Grove-GSR_Sensor/

4 x Seeed Studio Cables:

1x Grove - 4 pin Male Jumper to Grove 4 pin Conversion Cable

https://www.seeedstudio.com/Grove-4-pin-Male-Jumper-to-Grove-4-pin-Conversion-Cable-5-PCs-per-Pack.html

2x Grove - Universal 4 Pin Buckled 5cm Cable

https://www.seeedstudio.com/Grove-Universal-4-Pin-Buckled-5cm-Cable-5-PCs-Pack.html

1x Grove - Universal 4 Pin 20cm Unbuckled Cable

https://www.seeedstudio.com/Grove-Universal-4-Pin-20cm-Unbuckled-Cable-5-PCs-Pack-p-749.html

Microcontrollers:

1x Arduino Nano (or equivalent)

https://store.arduino.cc/arduino-nano

1x Wemos LOLIN D1 mini (or equivalent)

https://wiki.wemos.cc/products:d1:d1_mini

Electronic parts:

1x Adafruit RGB-LED Ring

https://www.adafruit.com/product/1463

1x Micro USB to DIP Adapter

https://www.google.com/search?q=Micro+USB+to+DIP+Adapter&oq=Micro+USB+to+DIP+Adapter&aqs=chrome..69i57.3497j0j7&sourceid=chrome&ie=UTF-8

Miscellaneous parts:

- 4x M3 Nylon Standoffs 25mm (Hex spacer)
- 8x M3 Nylon Screws 30mm
- M2 Nylon Standoffs (Hex spacer) for Grove (has 2mm holes)
- Small USB Power bank, max. size 15mm x 100mm x 60mm
- Small micro USB cable for connecting Power bank with Nano
- A Soldering Iron to attach cables to electronic components

Figure 6: M2 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

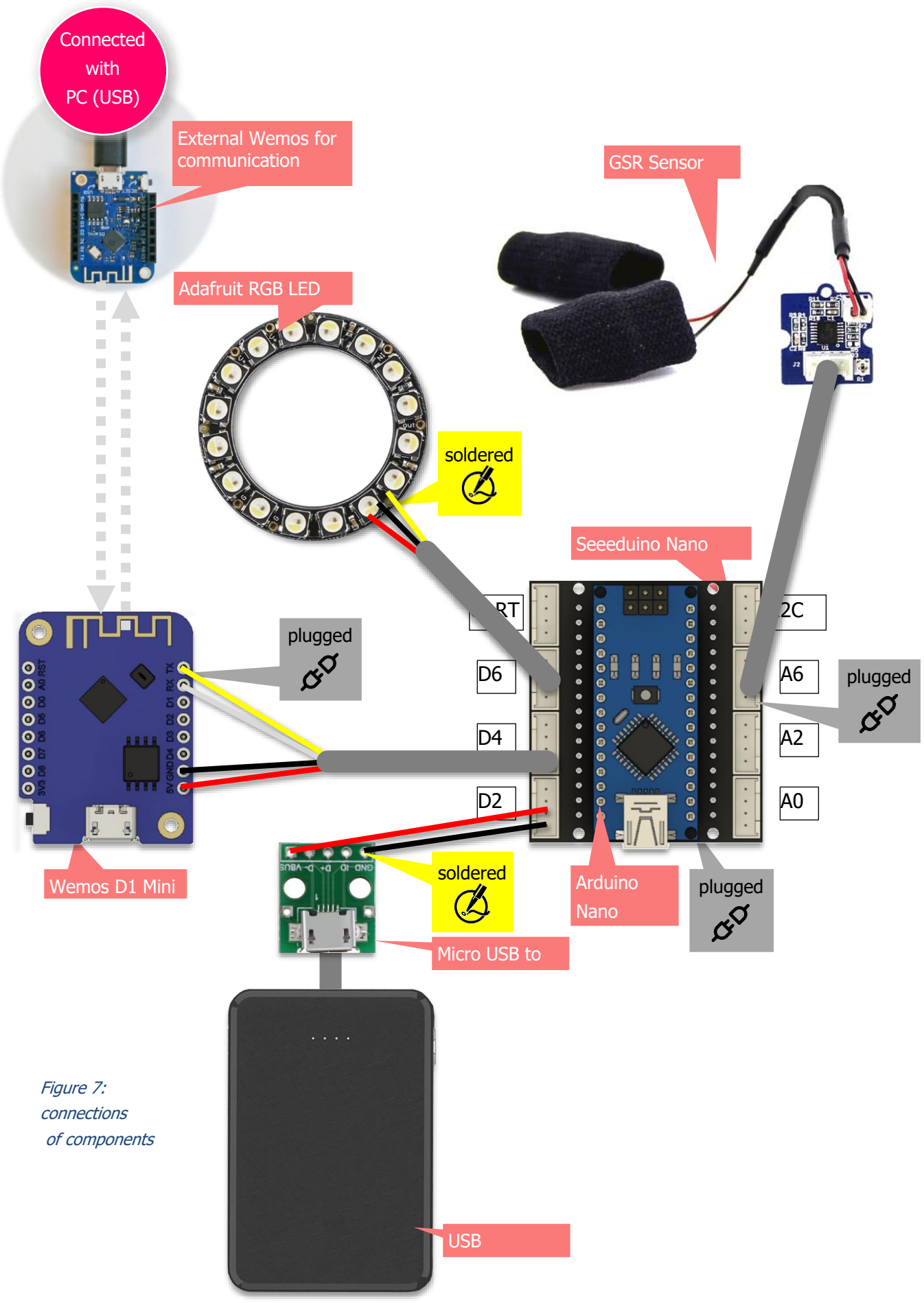


Figure 7: connections of components

Arduino libraries for components:

The Wemos D1 Mini needs libraries for the Arduino IDE to work properly. How to import a library is described here: https://www.arduino.cc/en/Guide/Libraries

Neopixel (Adafruit):
https://github.com/adafruit/Adafruit_NeoPixel/archive/master.zip

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.

The Grove GSR sensor does not need any libraries since it can be controlled with simple Arduino analog input 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 Wi-Fi 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 source code":

Here are the relevant excerpts from server- and client-source code 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 8: 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 9: 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.

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 lie 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 person's 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

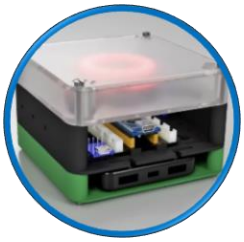
- Coactivity of muscles: Computer typing under stress conditions leads to contraction of trapezius muscle in humans' neck. This can be measured with EMG.

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

Lesson 1 & 2 (90min):

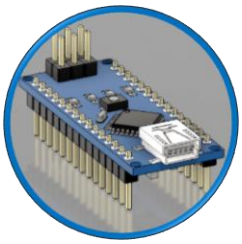
Students will be introduced IoT by examples: Vacuum robots with app remotes, internet-based weather stations, smart farming and finally health applications. Students should examine how those devices work and which components are needed: a microcontroller-based system controls and coordinates attached 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 and limitations need to be discussed: where does IoT make sense and where not?

Lesson 3 & 4 (90min):



3D-printing and assembling of the device: Optionally Students can 3D print the housing of the device, and after that they should connect all electronic parts by themselves. Students should develop a deeper understanding how parts fit together and hence build a complete device. What is the purpose of a LED-Ring, what is the Wi-Fi connection good for? How is a signal processed from its origin to the end – to the user? Starting from a bio signal being converted to an analog signal inside the device, converting it to a digital signal inside the ADC of the microcontroller, data processing using the software and communicating the results via LED light or transmitting information using Wi-Fi.

Lesson 5 & 6 (90min):



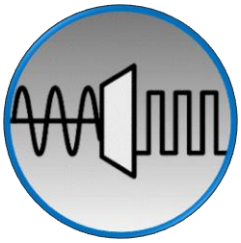
Introduction of Arduino Programming: Arduino IDE connection and communication setup of Arduino Nano with the IDE / the computer. The fundamental structure of the Arduino platform needs to be explained: what are general purpose input/output (GPIO) pins, what is digital logic and the what's the difference between digital input and output. Simple Scripts are written and modified using the basic examples that are shipped with the Arduino IDE:

"01.Basics → Blink",

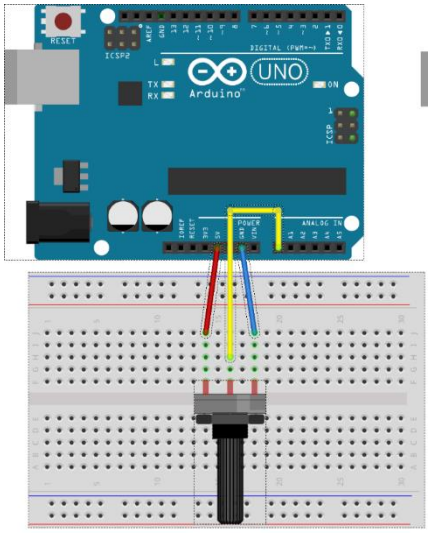
"01.Basics → DigitalReadSerial",

"04.Communications → SerialEvent",

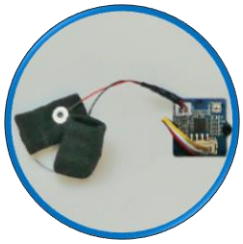
Lesson 7 & 8 (90min):



How is an analog signal processed by a digital machine? Theory of analog-to-digital-signal-conversion is introduced and can be tought using a simple voltage divider setup. Using the Arduino, any simple resistor-based sensor can be used to build up a simple circuit, e.g. an LDR, an thermoresistor or even a simple potentiometer.



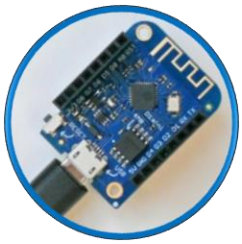
Lesson 9 & 10 (90min):



Introducing the GSR-Sensor: Teaching of signal amplification using operational amplifiers, measuring skin humidity with low voltages. How can the resistance of human skin be measured? What voltage is harmless to the user?

Biological background of electrodermal activity and its meaning/interpretation
https://en.wikipedia.org/wiki/Electrodermal_activity

Programming of the gsr sensor can be accomplished by using and adapting the pre-built Arduino-scripts from the preceding lessons.



Lesson 10 & 11 (90min):

Introducing the Wemos D1 Mini: This microcontroller board can be used as a wireless transmission device which completes the IoT-character of the GSR-Cube. Two Wemos are necessary for connecting two different nodes – one node is the Cube itself and the other node is for example a student's computer where all the processed signals are transferred to.

Since Wemos' networking Wi-Fi technology is very complicated and therefore needs a separate teaching unit, all Wemos devices must be preinstalled and configured carefully before the lesson. This must be done by the teacher.

It is important to emphasize the possibility of abuse: There are many aspects of privacy protection which can be easily illustrated with this setup. What are the consequences of bodies' own signal-data being transferred somewhere leading to a loss of control of ones private sphere?

Lessons 13 & finish (open):

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?

And now: Freestyle programming! And happy biofeedback! 😊

Try to make some experiments with your classmates. Make a discussion round and keep track of the output of your EDA-cube. Can you recognize how your colleague feels?

10. Feedback

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 tought.

11. Assessment & Evaluation

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.

Wemos Client Sourcecode

Figure 10: Source code for the Wemos client attached to the EDA-Cube

```
#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()) {           //wenn WIFI verfügbar ist,
    char c = client.read();           //lesen was reinkommt
    Serial.print(c);                  //und auf UART schreiben.
  }
  while (Serial.available() > 0) {    //solange auf UART Daten,...
    inChar = Serial.read();           //Daten von UART einlesen
    if (client.connected()) {         //und wenn WIFI läuft, ...
      client.write(inChar);           //auf WIFI schreiben
      delay(1);
    }
  }
}
```

Wemos Server Sourcecode

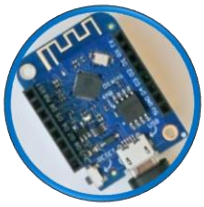


Figure 11: This script should be compiled for the Wemos connected with PC

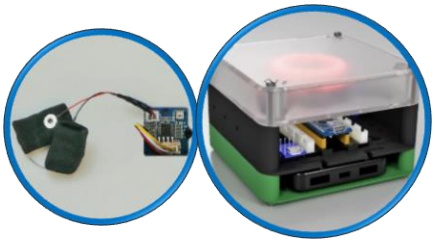
```
#include <ESP8266WiFi.h>
const char *ssid = "Erasmus";
const char *password = "12345678";
IPAddress Ip(192, 168, 3, 1);
IPAddress NMask(255, 255, 255, 0);
WiFiServer server(23);
WiFiClient sClient;
char inChar;

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 (!sClient || !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;
}}}
```

EDA-Cube Arduino Nano code

Figure 12: a working example source code for the EDA-Cube during all lessons



```
#include <Adafruit_NeoPixel.h>
#define NEOPIXELPIN        6
#include <SoftwareSerial.h>
SoftwareSerial Serial_45(4, 5);

Adafruit_NeoPixel pixels(16, NEOPIXELPIN, NEO_RGBW + NEO_KHZ800);
const int GSR = A6;
long sum = 0;
int gsr_average, sensorValue, r, g, gsr_alt, delta = 0;

void setup() {
  Serial.begin(9600);
  Serial_45.begin(9600);
  pixels.begin();
  pixels.clear();
}

void loop() {
  pixels.clear();
  sum = 0;
  for (int i = 0; i < 20; i++)
  {
    sensorValue = analogRead(GSR);
    sum += sensorValue;
    delay(5);
  }
  gsr_average = sum / 10;
  delta = abs(gsr_average - gsr_alt);
  delta = constrain(delta, 0, 255);
  gsr_alt = gsr_average;
  Serial.println(gsr_average);
  Serial_45.println(gsr_average);
  r = 255 - (int) ((gsr_average - 600) / 3.125);
  g = (int) ((gsr_average - 600) / 3.125);
  for (int i = 0; i < 16; i++) {
    pixels.setPixelColor(i, pixels.Color(g, r, 0, delta));
  }
  pixels.show();
  delay(10);
}
```