'We are the makers - IoT' Learning Scenario: smart farming with an IoT-plantrobot

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.

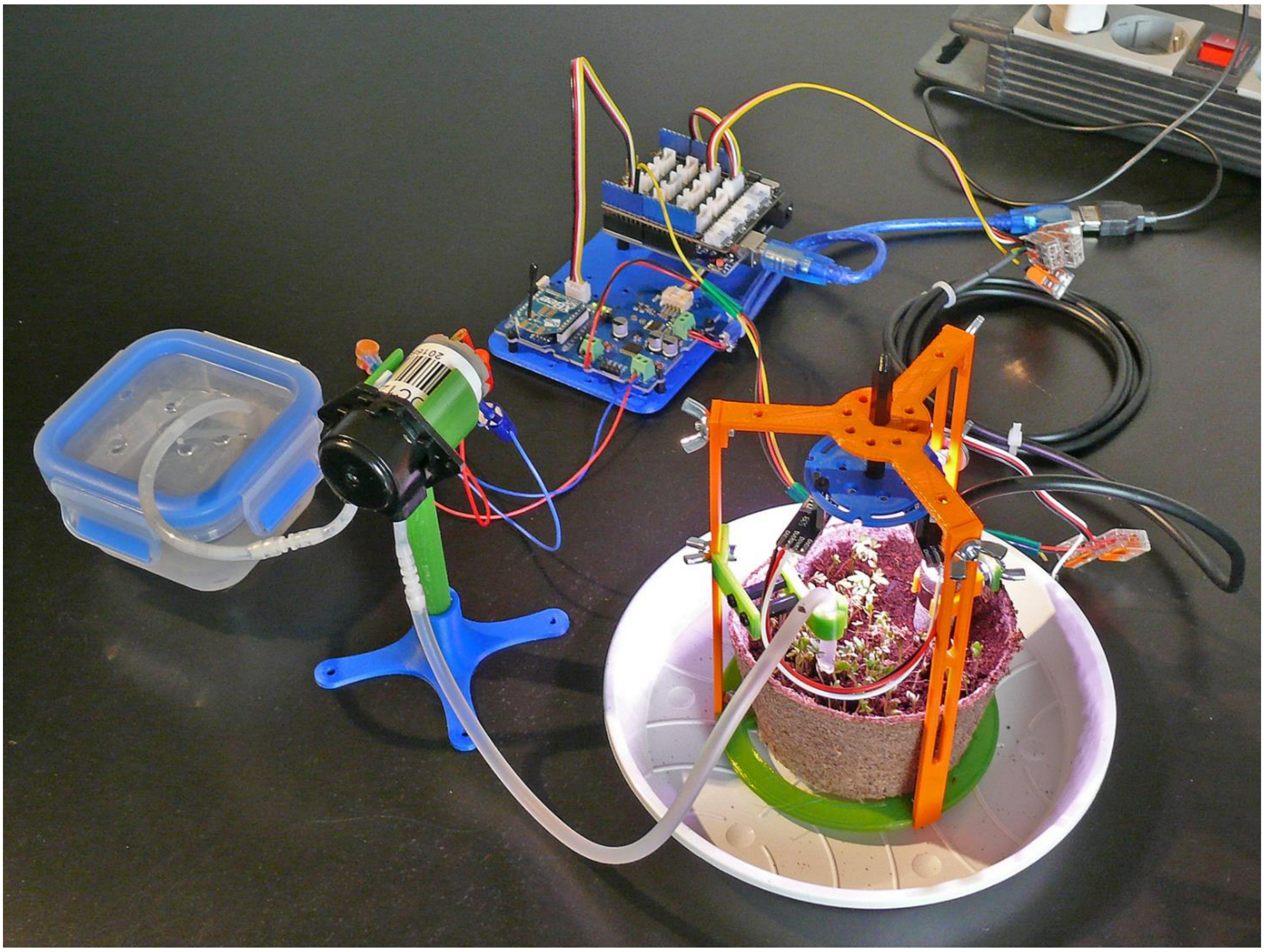
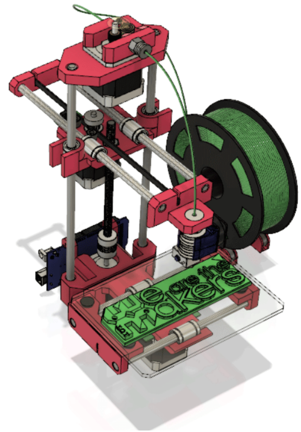
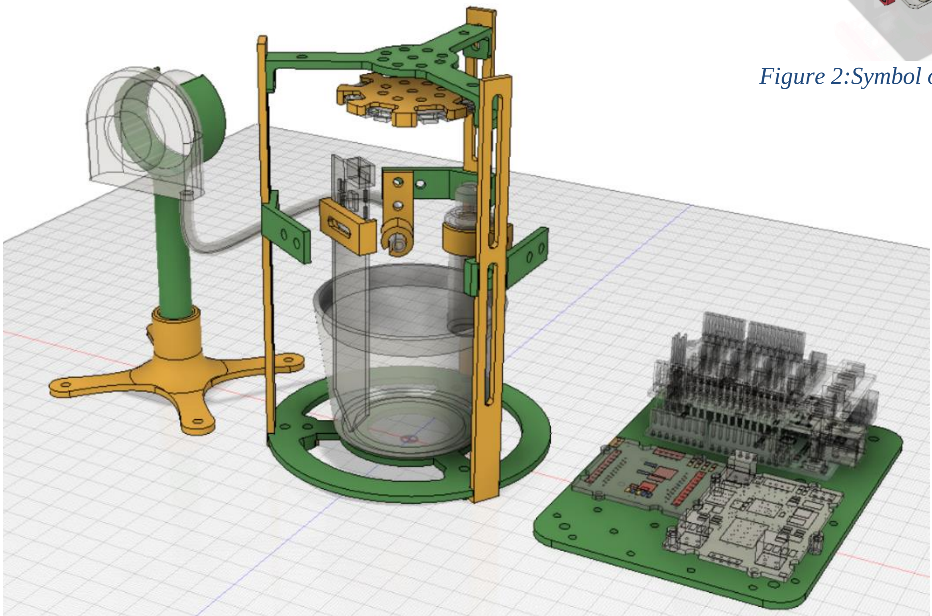
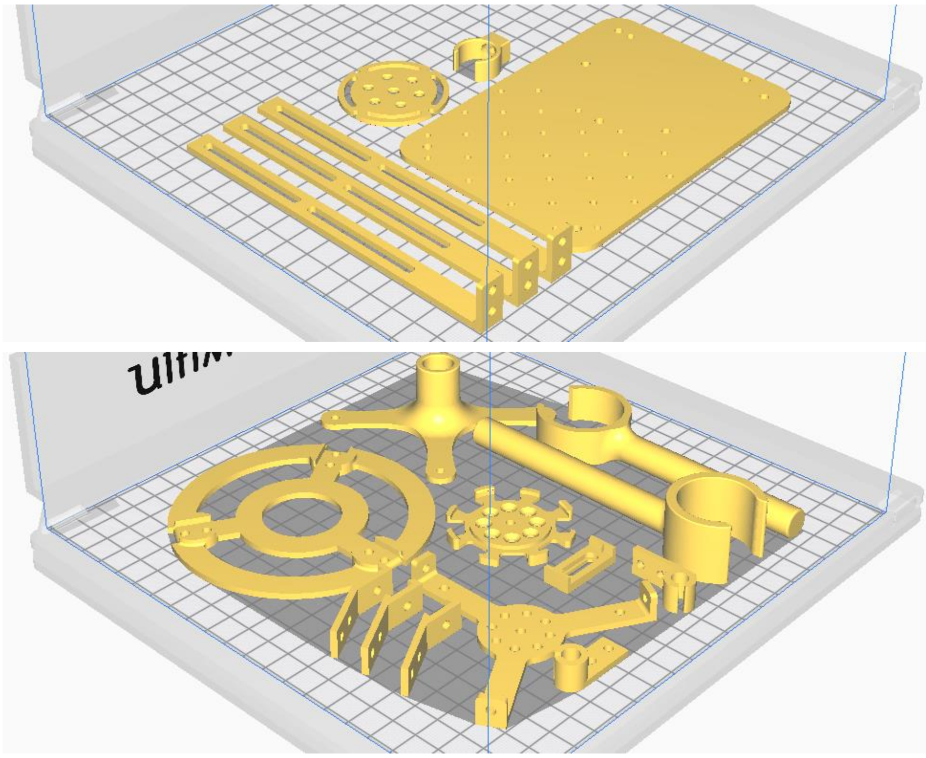


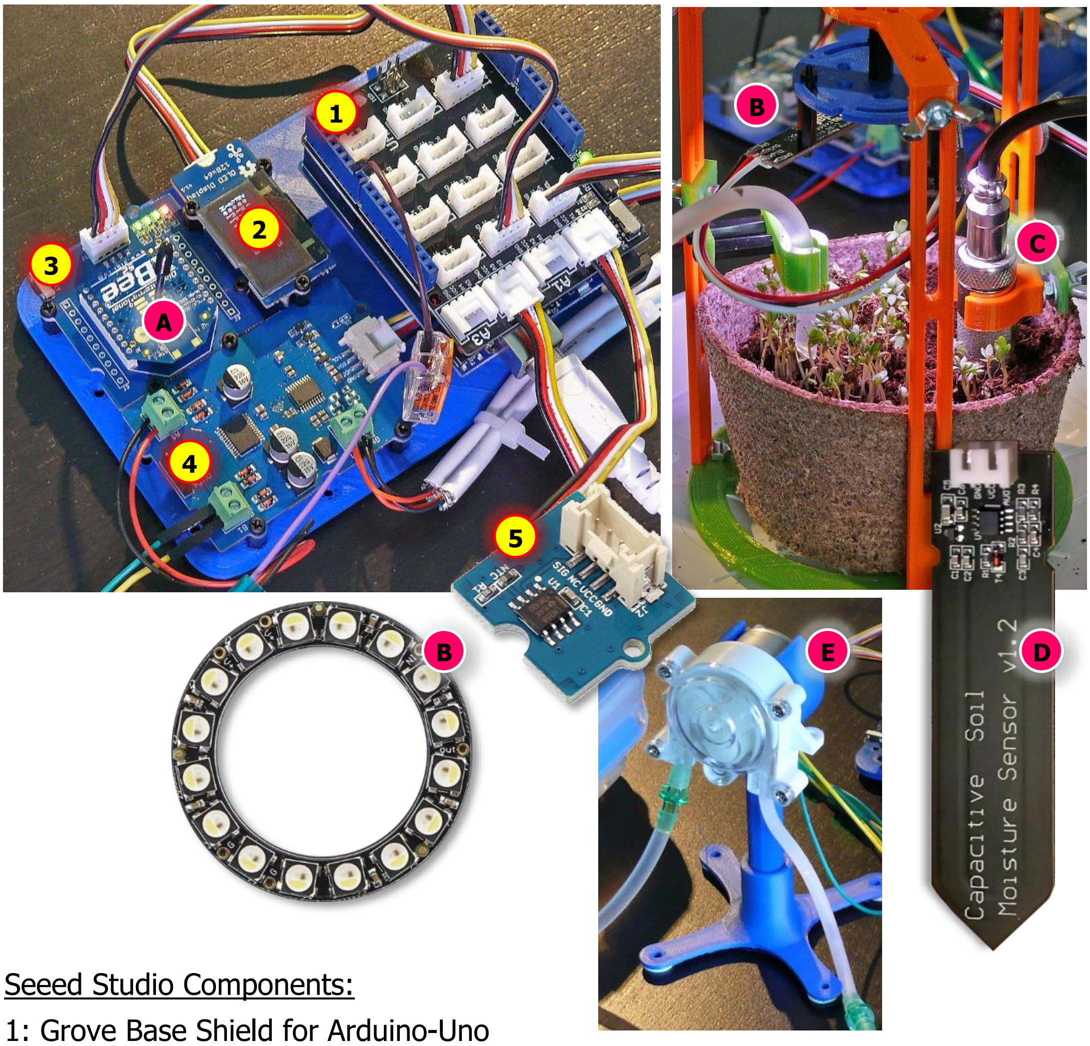
Figure 1: Prototype of a IoT plantgrowing robot

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	▪ Interaction between electronic parts and creatures (here: plants) ▪ Monitoring and affecting biological parameters ▪ Communication chain of IoT-devices ▪ Principles of sensors and actors ▪ Different principles humidity measuring in soil. ▪ Principles of LED-lighting for growing plants ▪ Fine adjustment of machine parameters for optimizing plant growing ▪ Principles of wireless communication networks ▪ Construction and 3D-printing of a robotic environment
5. Expected learning outcomes	▪ How does an IoT-system work? ▪ Where are possibilities and limitations of IoT-systems? ▪ Which components – hard and software – are key to build an IoT-device? ▪ How to build the rules for biomonitoring and influencing living creatures?
6. Methodo-logies	In this scenario students will construct, build and program a fully interactive plant growing device from scratch by themselves. Students will also build an app for remote controlling the IoT-plantrobot
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; ▪ each student has to keep a laboratory journal

8. Tools/ Materials/ Resources	3D-Printers About 3-4 3D-printers are necessary since students will print their IoT-plantrobots by themselves.  *Figure 2:Symbol of 3D-Printer* 3d printed components: As a starting point, all necessary parts are provided in .stl-format and as Autodesk Fusion 360 Files.  *Figure 3: Overview of CAD-Data*  *Figure 4: complete set of 3D-data on a 20cm x 20cm 3D-printer*

Electronic components:

In this work, we recommend the seeed grove system since it’s ease of use: (http://wiki.seeedstudio.com/Grove_System/) All core components except XBees, Humidity sensors and LED-lighting are belonging to the grove standard:



Seeed Studio Components:

1: Grove Base Shield for Arduino-Uno

2. Grove OLED 128x64

3. Grove Bee socket

4. Grove - I2C Motor Driver (TB6612FNG)

5. Grove Temperatur Sensor v1.2

Regular sensors and actors:

1: Arduino Uno (or equivalent)

A: XBee Series 2C or Series 2

B: Adafruit Neopixel-ring with 16 RGBW (NOT RGB!) at 4500 K (warm-white)

C: SHT20-Temperature and Humidity-Sensor in waterproof housing or

D: Capacitive analog Soil Moisture Sensor

E: Peristaltic pump with 6V DC motor

	Miscellaneous parts: ▪ 5-6 mm silicone tubing (for aquarium purpose) ▪ Adapter for connecting silicone tubes with peristaltic pump ▪ M3 screws and butterfly nuts ▪ M3 Nylon Standoffs (Hex spacer) ▪ M2 Nylon Standoffs (Hex spacer) for Grove (has 2mm holes) ▪ Grove wires ▪ WAGO lever nuts ▪ Jumper wires ▪ Nursery pots with 8cm diameter ▪ USB power supply with 2-2.5A maximum current ▪ XBee USB adapter (e.g. https://www.waveshare.com/xbee-usb-adapter.htm)  *Figure 5: Nylon Standoffs* Plants: Suitable for doing experiments at school are fast growing plants which are called "Microgreens" / "Microgreen Sprouts"; they are nontoxic and eatable: ▪ Garden cress ▪ Mung Beans ▪ Red stem radish ▪ Red clover sprouts ▪ Broccoli sprouts ▪ Valerianella locusta ("Vit" field salad)  *Figure 6: Garden cress* 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 ▪ XCTU Software for configuring XBees

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

Arduino libraries for components:

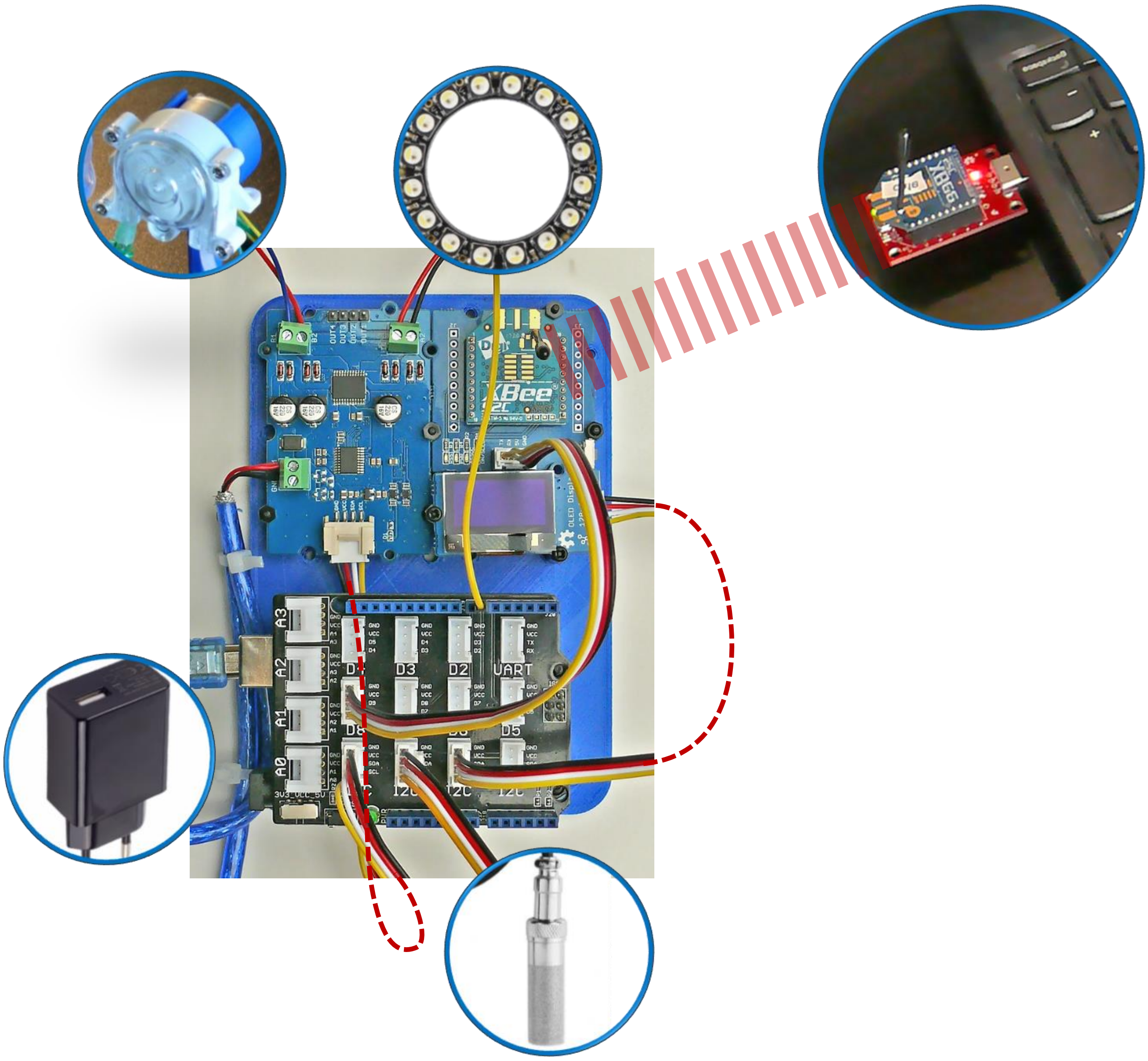
Some components like the SHT20 humidity sensor or the motor driver need libraries for Arduino IDE to work properly. How to import a library is described here: https://www.arduino.cc/en/Guide/Libraries

SHT20 lib (DF Robot):	https://codeload.github.com/DFRobot/DFRobot_SHT20/zip/master
OLED lib (Seeed):	https://github.com/Seeed-Studio/OLED_Display_128X64/archive/master.zip
Motor driver (Seeed):	https://github.com/Seeed-Studio/Grove_Motor_Driver_TB6612FNG
Neopixel (Adafruit):	https://github.com/adafruit/Adafruit_NeoPixel/archive/master.zip

Special code snippets for temperature sensor can be found here:

http://wiki.seeedstudio.com/Grove-Temperature_Sensor_V1.2/

Connections:



Number of project: 2017-1-DE03-KA201-035615

8b Some theory of LED growlights and soil moisture measuring

LED growlighting

Foundation of using LEDs for plantgrowing is the theory of PAR, "photosynthetically active radiation": Plants are using light photons for chemical reactions to build sugar from carbon dioxide; these chemical reactions occur using chlorophyll pigments inside the chloroplasts of each plant cell.

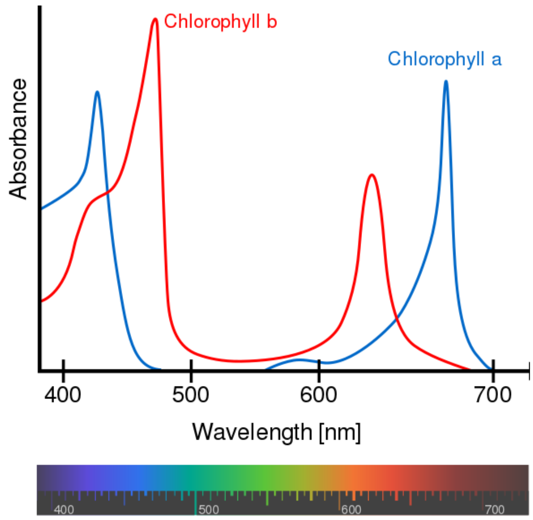


Figure 7: The absorption spectrum of both the chlorophyll a and the chlorophyll b pigments. https://en.wikipedia.org/wiki/Chlorophyll_b

Chlorophyll, when irradiated with sunlight, absorbs red and blue light. The green color parts are not absorbed directly for the photosynthetic process directly. Therefore plants are green.

LED light for plantgrowing purposes must mainly provide blue and red light from the chlorophyll absorption spectrum. That is why
we use the "R" and the "B" parts of neopixel RGBW-high power LEDs. The green part of the LED is not used.

Figure 8: Adafruit neopixel, RGBW-LED, https://www.adafruit.com/product/2758

But a plant also uses other parts of the continuous sunlight spectrum, the photosynthetic process is more complex and is a field of current research. In short: green light dives deeper into a plant and makes the photosynthetic process more efficient, since it affects the absorbance rate of chlorophyll. Therefore small amounts of continuous green- to yellow spectrum is necessary.
https://academic.oup.com/pcp/article/50/4/684/1908367

In addition, plant growing is affected by plant hormons, which also react to sunlight and mostly need a continuous sunlight spectrum. As an example, phytochromes react to infrared lighting.
https://en.wikipedia.org/wiki/Plant_hormone
https://en.wikipedia.org/wiki/Phytochrome

As a consequence, an optimal growlight must not be limited to red and blue parts of the spectrum but also needs a 'white' LED part which produces a warm-white continuous sprectrum to affect secondary photosynthetic systems and plant hormons. Therefore we use the 4500K-RGBW LED from Adafruit Industries.

Soil moisture sensing

Usually moisture is measured as percentage of air humidity. Therefore a temperature and humidity sensor is necessary. Humidity itself can be measured in different ways and one of the most common methods is capacitive measurement. The sensor itself is a capacitor whose capacity is altered with the absorption / desorption of water.

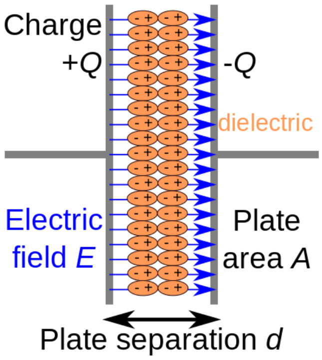


Figure 9: scheme of a capacitor. https://en.wikipedia.org/wiki/Capacitor

The capacity C of a capacitor depends on the plate-area A, the distance between the plates d and the dielectric medium between two metal plates with a given permittivity constant ε_R:

$$C = \varepsilon_R \frac{A}{d}$$

While distance and size of the plates cannot be altered with humidity, the permittivity constant does. Usually the permittivity of a given compound is compared with the permittivity of the perfect vacuum and therefore it's called the 'relative permittivity'. Here are some important values for the relative permittivity:

Medium	*Rel.permittivity*
Vacuum	*1*
Air	*1.0006*
Water	*80*
Dry soil mineral	*5*

As a consequence, the capacitance of soil is increasing the more water it contains. Watery, wet soil has a significantly higher permittivity than it's dry counterpart. Usually it is measured as so called "volumetric soil water content", SWC. It is defined as volumetric water content:

$$SWC = \frac{Volume\ of\ water}{Volume\ of\ soil}$$

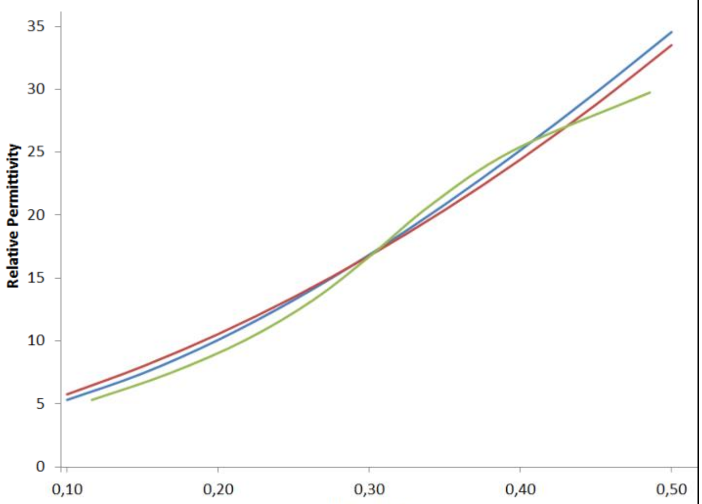


Figure 10: Porretta, Bianchi "Profiles of relative permittivity and electrical conductivity from unsaturated soil water content models", ANNALS OF GEOPHYSICS, 59, 3, 2016, G0320

An electronic circuit for measuring these changes in capacitance is build as an RC-circuit. Depending on its capacitance a RC-circuit has a characteristic time-constant which can be measured by a microcontroller. The lager the capacitance the longer the time constant. In Summary, humidity is measured this way:

$$increasing\ Humidity \xrightarrow{leads\ to} increasing\ capacitance \xrightarrow{leads\ to} increasing\ time\ constant$$

Comparison SHT20 with housing and Soil moisture sensor:

SHT 20	Soil moisture sensor
	
Measures Air humidity inside it's waterproof housing and simultaneously the temperature	Measures directly the relative permittivity of soil. An additional temperature measurement is necessary.
Communicates via I2C with the Arduino microcontroller	Produces an analog signal which must be digitalized with the ADC of the Arduino microcontroller
Must not be put completely inside the soil since airflow is necessary	Has to be kept as deep as possible inside the soil
Costs about 20 Euro	Costs about 5 Euro
ATTENTION: Since this sensor is producing relative air humidity as an output, the measurement values doesn't make any sense for the soil conditions. Most of the time the sensor produces values above 100% since the air inside the waterproof housing is saturated with soil humidity. You need to use the raw 16 bit values which are not converted to the unmeaningful air humidity values.	ATTENTION: Since this sensor is measuring the relative permittivity of the soil, it is absolutely essential, that the sensor is in perfect touch to the soil with no air layer between both surfaces. Furthermore the sensorvalues are drifting, since with watering the volume of the soil changes and therefore the covering surface of the sensor also does. In addition, growing plants are also changing the permittivity values.

8c Arduino source code as an example

```
#include <Adafruit_NeoPixel.h>
#include "Grove_Motor_Driver_TB6612FNG.h"
#include "DFRobot_SHT20.h"
#include <Wire.h>
#include <SoftwareSerial.h>

MotorDriver Energie;
DFRobot_SHT20     sht20;
Adafruit_NeoPixel GrowLED(16,6,NEO_RGBW);
SoftwareSerial Serial_89(8, 9);

unsigned long aktMillis = millis();
unsigned long readMillis = aktMillis;
unsigned long ledMillis = aktMillis;
unsigned long serialMillis = aktMillis;
unsigned long pumpMillis = aktMillis;
unsigned long Feuchtigkeit = 0;
unsigned long FeuchtSoll = 55300;
float Temperatur = 0;
int Giessdauer = 0;
int Helligkeit = 0;

void setup() {
  Wire.begin();
  Serial.begin(9600);
  Serial_89.begin(9600);
  Energie.init();
  GrowLED.begin();
  LEDsetzen();
  sht20.initSHT20();
  delay(100);
  sht20.checkSHT20();
  Energie.dcMotorRun(MOTOR_CHA, 255);
  Energie.dcMotorBrake(MOTOR_CHB);
}

void loop() {
  aktMillis = millis();

  if (aktMillis - serialMillis >= 1000) {
    while (Serial_89.available() > 0) {
unsigned long test = Serial_89.parseInt();
      if (test > 0 && test < 1000) {
        wasserpumpen(10);
        serialMillis = aktMillis;}
      if (test > 1000) {
        FeuchtSoll = test;}
    }
  }

  if (aktMillis - ledMillis >= 60000) {
    LEDsetzen();
    ledMillis = aktMillis;
  }

  if (aktMillis - readMillis >= 5000) {
   Feuchtigkeit = sht20.readHumidityRaw();
   Temperatur = sht20.readTemperature();
   Serial_89komm();
   readMillis = aktMillis;
  }

  if (aktMillis - pumpMillis >= 300000) {
    if (Feuchtigkeit < FeuchtSoll) {
      wasserpumpen(5);
      pumpMillis = aktMillis;
   }
  }
}
```

```
void wasserpumpen(int Sekunden) {
  Giessdauer = Giessdauer + Sekunden;
  Serial_89komm();
  Energie.init();
  delay(10);
  Energie.dcMotorBrake(MOTOR_CHA);
  delay(10);
  Energie.dcMotorRun(MOTOR_CHB, -255);
  delay(100);
  Energie.dcMotorRun(MOTOR_CHB, 255);
  delay(Sekunden * 1000);
  Energie.dcMotorBrake(MOTOR_CHB);
  delay(10);
  Energie.dcMotorRun(MOTOR_CHA, 255);
  delay(10);
  LEDsetzen();
}

void LEDsetzen() {
unsigned long Minuten=(aktMillis%8640)/6;
  int r = 0;
  int g = 0;
  int b = 0;
  int w = 0;
  if (Minuten < 840) {
    if (Minuten < 64) {
      r = Minuten * 4;
      g = 0;
      b = Minuten * 2;
      w = Minuten * 4;
    }
    if (Minuten >= 64 && Minuten <= 776) {
      r = 255 - (Minuten - 64) / 6;
      g = 0;
      b = 128 + (Minuten - 64) / 6;
      w = 255;
    }
    if (Minuten > 776) {
      r = 128 - (Minuten - 776) * 2;
      g = 0;
      b = 255 - (Minuten - 776) * 4;
      w = 255 - (Minuten - 776) * 4;
    }
  }
  Helligkeit = (int)(r + b + w) / 3;
  for (int i = 0; i < 16; i++) {
    GrowLED.setPixelColor(i,
GrowLED.Color(g, r, b, w));
  }
  GrowLED.show();
}

void Serial_89komm() {
  Serial_89.print("Zeit: ");
  Serial_89.print(aktMillis / 1000);
  Serial_89.print(", fIst: ");
  Serial_89.print(Feuchtigkeit);
  Serial_89.print(", Soll: ");
  Serial_89.print(FeuchtSoll);
  Serial_89.print(", Temp: ");
  Serial_89.print(Temperatur);
  Serial_89.print(", Wass: ");
  Serial_89.print(GesamtGiessdauer);
  Serial_89.print(", Hell: ");
  Serial_89.println(Helligkeit);
}
```

8d: Processing App with sourcecode

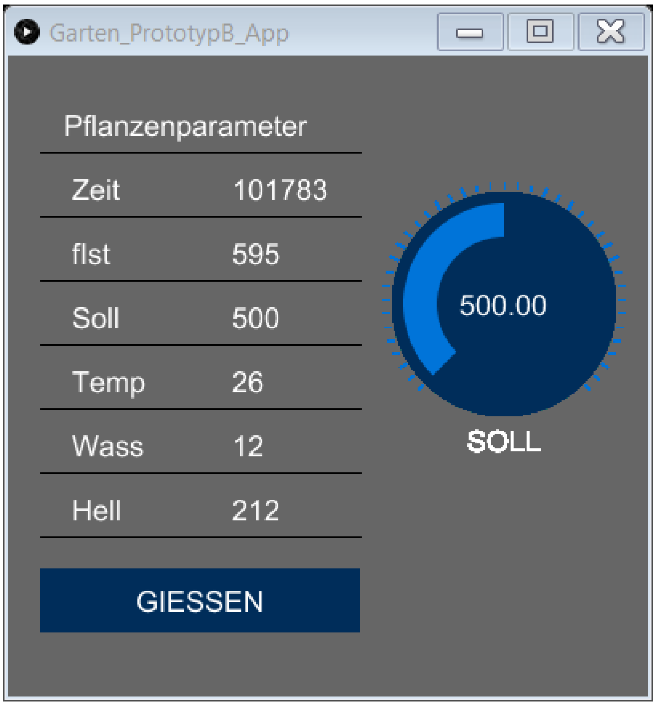


Figure 11: Screenshot App interface

A typical IoT device typically has an App interface which allows the user to monitor and remote control his connected device.

One (relatively) easiy way to achieve this with students is to use processing with it's capabilities to read serial data from the XBee and draws it's values to a graphical user interface.

In addition, the Arduino IDE and the Processing IDE are tightly related to each other since the Arduino IDE is a 'Fork' of processing.

For the button "GIESSEN" and the circular knob "SOLL" (which means optimal value for soil humidity") the "ControlP5"-library is used which can be easily installed from within the Processing IDE.

For controlP5 refer to: https://code.google.com/archive/p/controlp5/downloads

8c. Processing source code as an example

```
import controlP5.*;
import processing.serial.*;

Serial arduinoKommunikation;
String payload;
String[] liste;
ControlP5 cp5;
Knob sollFeuchte;
int sollFeuchteWert = 500;
int arduinoSollWert = 0;

void setup(){
  size(400, 400);
  background(102);
  smooth();
  String portName = Serial.list()[0];
  arduinoKommunikation = new Serial(this, portName, 9600);

  cp5 = new ControlP5(this);
  PFont font = createFont("arial", 18);
  textFont(font);
  cp5.setFont(font);

  sollFeuchte =cp5.addKnob("Soll")
    .setRange(300, 700)
    .setValue(sollFeuchteWert)
    .setPosition(240, 85)
    .setRadius(70)
    .setDragDirection(Knob.VERTICAL)
    .setNumberOfTickMarks(40)
    .setTickMarkLength(4)
    .snapToTickMarks(true)
    .onRelease(new CallbackListener() {
    public void controlEvent(CallbackEvent theEvent) {
      sollFeuchteWert= int(theEvent.getController().getValue());
```

```
    }});

  cp5.addButton("giessen")
    .setValue(0)
    .setPosition(20, 320)
    .setSize(200, 40)
    ;
}

void draw(){
  if ( arduinoKommunikation.available() > 0) {
    payload = arduinoKommunikation.readStringUntil('\n');
    if (payload != null) {
      background(102);
      text("Pflanzenparameter", 35, 50);
      line(20, 60, 220, 60);
      liste = split(payload, ",");
      for (int i = 0; i<liste.length; i++) {
        liste[i] = trim(liste[i]);
        String[] Groesse = split(liste[i].trim(), ":");
        text(Groesse[0], 40, 90+40*i);
        int val= parseInt(Groesse[1].trim());
        if (i==2) {
          arduinoSollWert = val;
        }
        text(int(val), 140, 90+40*i);
        line(20, 100+40*i, 220, 100+40*i);
      }
      if (arduinoSollWert != sollFeuchteWert) {
        arduinoKommunikation.write(str(sollFeuchteWert));
      }
    }
  }
}

public void giessen() {
  if (millis()>5000) {
    arduinoKommunikation.write(str(2));
  }
}
```

<table>
<tr>
<td>9.
Lesson plan
Step by step description of the activity/ content</td>
<td>

Lesson 1 & 2 (90min):

Students will be introduced IoT by examples: Vacuum robots with app remotes, internet based weather stations, activity trackers with app communication and last but not least smart agricultural farms. 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 does it not?

Lesson 3&4 (90 min)

Students should plan the components for monitoring and optimizing plant growing. Afterwards they can start to build such a machine from scratch by using the provided parts. When finished, basic programming techniques can be taught for enabling students to do their own experiments.

Lessons 5&6 (90 min)

Theory of soil moisture measurement and plant lighting have to be taught. Students can make measurements of SWC with their individual sensors to make calibration curves. Students should compare their experimental results to realize that every student group has it's own measurement values which differ significantly.

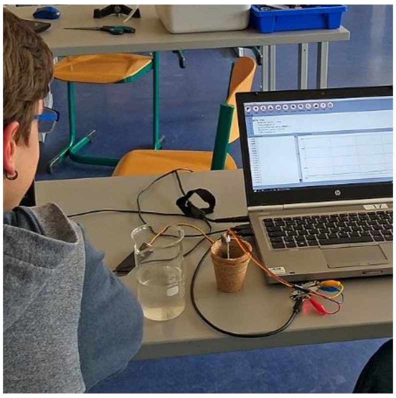


Figure 12: experimenting with soil sensors

Lessons 7&8 (90 min):

Students should start programming and controlling the peristaltic pumps with the motor driver. Theory of H-Bridges have to be taught, and in addition the basics of I2C-communication between electronic device components should be explained. Students should measure I2C communication with Oscilloscopes. The concept of PWM (Pulse width modulation) has to be introduced. Students can furthermore measure the communication between the Arduino and the Neopixel LEDs via Oscilloscope.

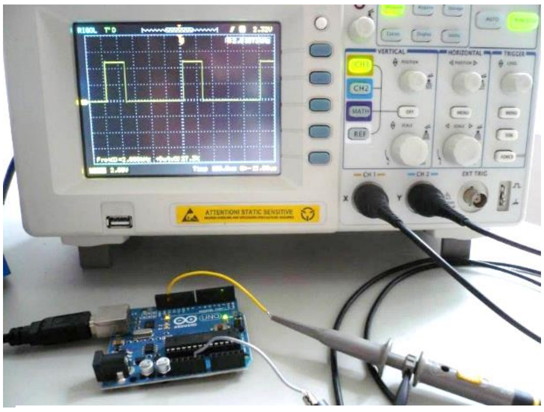
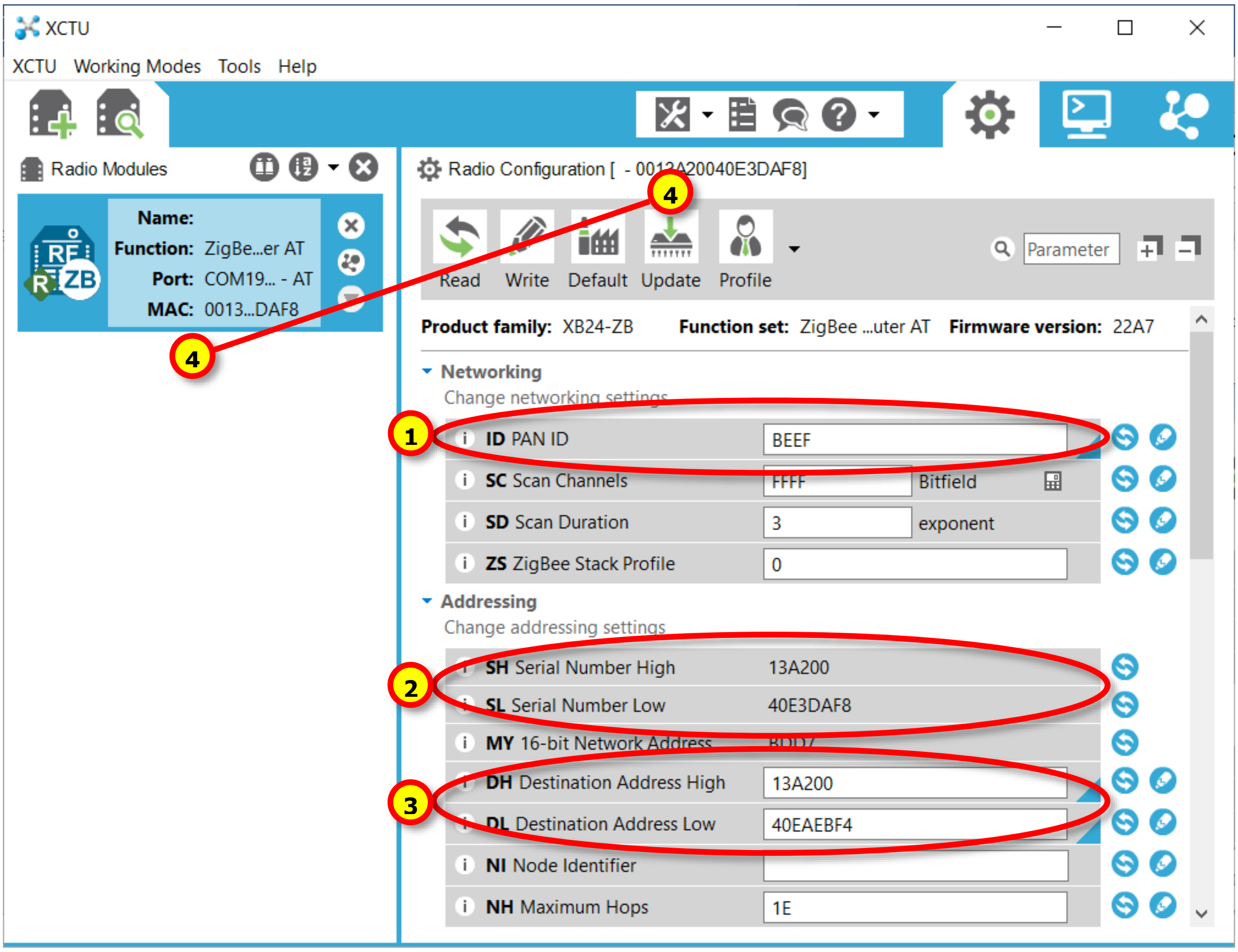


Figure 13: PWM measurement with oscilloscope

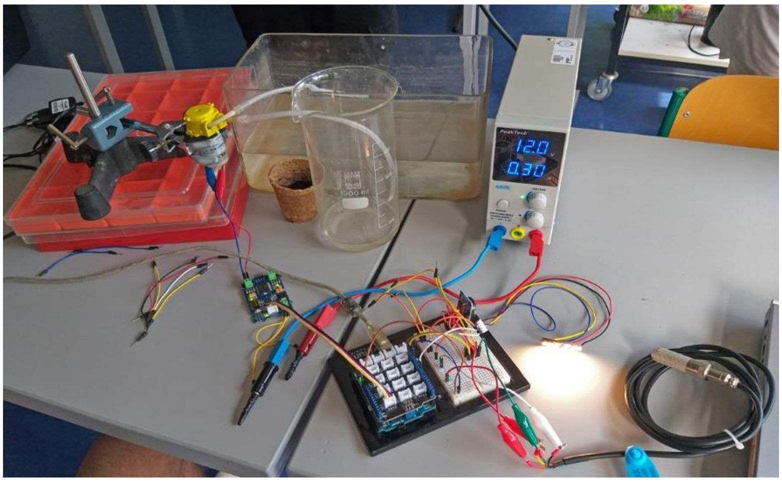
</td>
</tr>
</table>

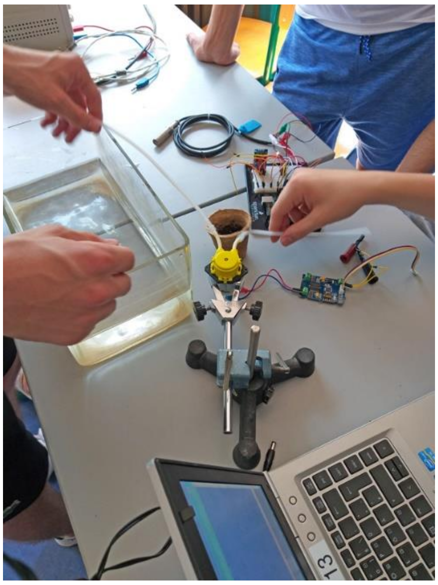
	Lessons 9&10 (90 min): Building a communication network: XBee modules and serial communication (UART). Students should use XCTU Software as a starting point for their communication experiments in wireless networks. ATTENTION: XBees should be preconfigured in pairs by the teacher, since otherwise much time will be lost by learning how to handle the many different configurations. Basically, XBee-Pairs are defined using three different user-based values which are framed here in red:  *Figure 14: Screenshot XCTU User Interface* 1: PAN ID has to be the same for both XBees. Use hexadecimal letters, e.g. so called "Hexspeak": "BEEF", "CAFE", "F00D", "AFFE", etc.. 2. The Serial Number High has to be copied from one XBee …. 3. … to the second one. 4. One XBee has to be configured as 'Coordinator' and the other as 'Router'. Attention: In some texts you can read that the second one should be configured as 'Endpoint'. That can probably produce some issues, since endpoint-XBees go to sleep for energy saving reasons.

After that, XBees can be put onto the UART communication port of the arduino.

Lessons 11 to finish (270 min):

Freestyle programming! And happy harvesting ☺





10. Feedback	At the end of the lesson, students should have a well-grounded knowledge of how IoT principles work and how machines connected to the internet are communicating. They have experienced by themselves chances and limitations of current technology. During the lesson, important aspects of electronics, informatics and construction basics have been tutorised. Furthermore, biological aspects of plant growing have been teached.
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.