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

‘We are the makers - IoT’ Lern Scenario: smart farming mit einem IoT-Pflanzenroboter

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

Das folgende Papier wurde im Schuljahr 2018/2019 in einem Schulumfeld mit ca. 18 Schülern im Alter von 13-17 Jahren entwickelt und getestet. Es spiegelt die Erfahrung mit vielen Verzweigungen und einigen Misserfolgen wider. Da das IoT-Feld komplex ist, müssen Lehrmaterialien sorgfältig ausgewählt werden. Dieses Papier soll als Ausgangspunkt dienen.

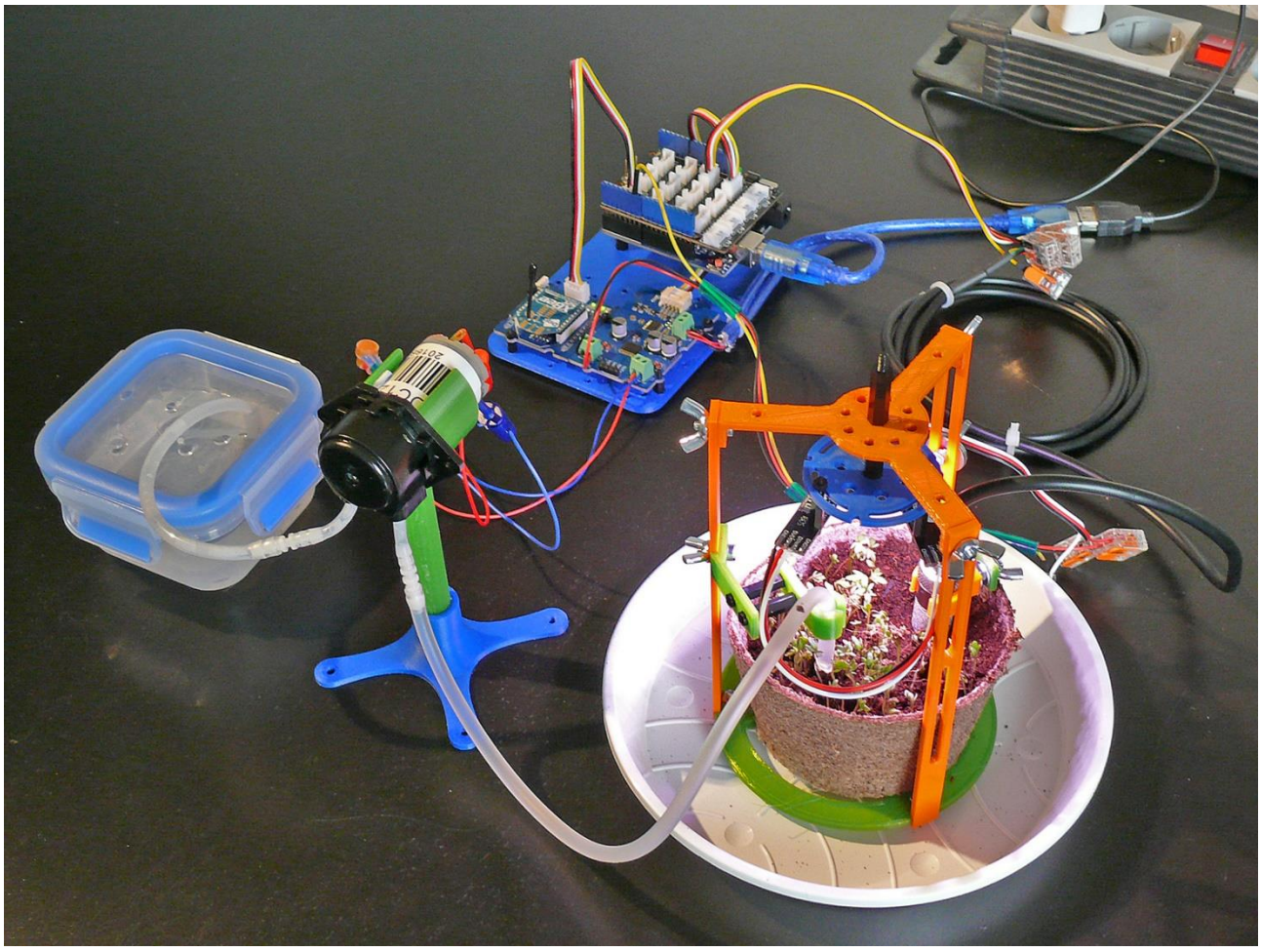


Abbildung 1Prototyp eines IoT-Pflanzenroboters

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

1. Titel des Scenarios	**Erfahren Sie, wie Sie Pflanzen mit Hilfe eines IoT-Pflanzenroboters züchten**
2. Zielgruppe	14 - 17 years
3. Dauer	At minimum 5 weeks of 2*45min-Lektionen per week: in sum about 6-8 hours.
4. Lernbedürfnisse, die durch die Übung abgedeckt werden	• Interaktion zwischen elektronischen Teilen und Lebewesen (hier: Pflanzen) • Überwachung und Beeinflussung biologischer Parameter • Kommunikationskette von IoT-Geräten • Grundsätze von Sensoren und Aktoren • Verschiedene Prinzipien der Feuchtigkeitsmessung im Boden. • Prinzipien der LED-Beleuchtung für Pflanzenwachstum • Feinjustierung von Maschinenparametern zur Optimierung des Wachstums • Grundsätze drahtloser Kommunikationsnetze • Konstruktion und 3D-Druck einer Roboterumgebung
5. Erwartete Lernergebnisse	• Wie funktioniert ein IoT-System? • Wo liegen Möglichkeiten und Grenzen von IoT-Systemen? • Welche Komponenten sind der Schlüssel zum Bau eines IoT-Geräts? • Wie können die Regeln für die Beeinflussung von Lebewesen aufgebaut werden?
6. Methoden	In diesem Szenario erstellen, bauen und programmieren die Kursteilnehmer ein vollständig interaktives Pflanzenwachstumsgerät von Grund auf selbst. Die Studierenden erstellen auch eine App zur Fernsteuerung des IoT-Pflanzenroboters.
7. Ort/ Umgebung	• ein Labor mit einer Reihe von elektronischen Teilen und Komponenten; • Jede Gruppe von Kursteilnehmern benötigt einen Computer oder Laptop mit Administratorrechten für die Installation verschiedener Softwarepakete • Ein Projektor für den Unterricht und die Präsentation von Studentenarbeiten • jeder Schüler muss ein Labortagebuch führen

8. Werkzeuge/ Materialien/ Ressourcen

3D-Drucker

Etwa 3-4 3D-Drucker sind notwendig, da die Schüler die IoT-Pflanzenroboter selbst drucken werden.

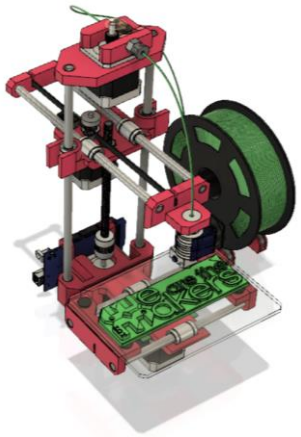


Figure 2:Symbol eines 3D-Druckers

3D-gedruckte Bauteile:

Als Ausgangspunkt werden alle notwendigen Teile im .stl-Format und als Autodesk Fusion 360-Dateien bereitgestellt.

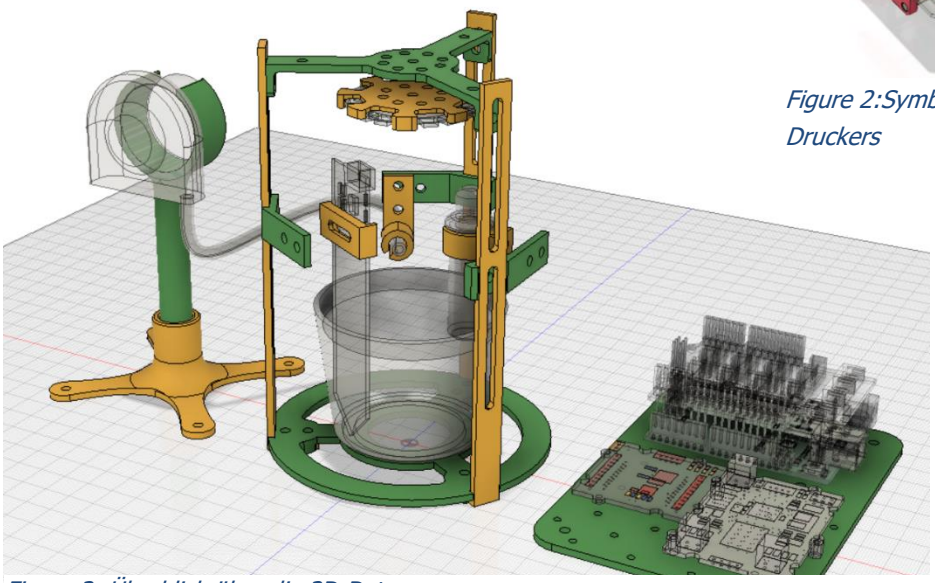


Figure 3: Überblick über die 3D-Daten

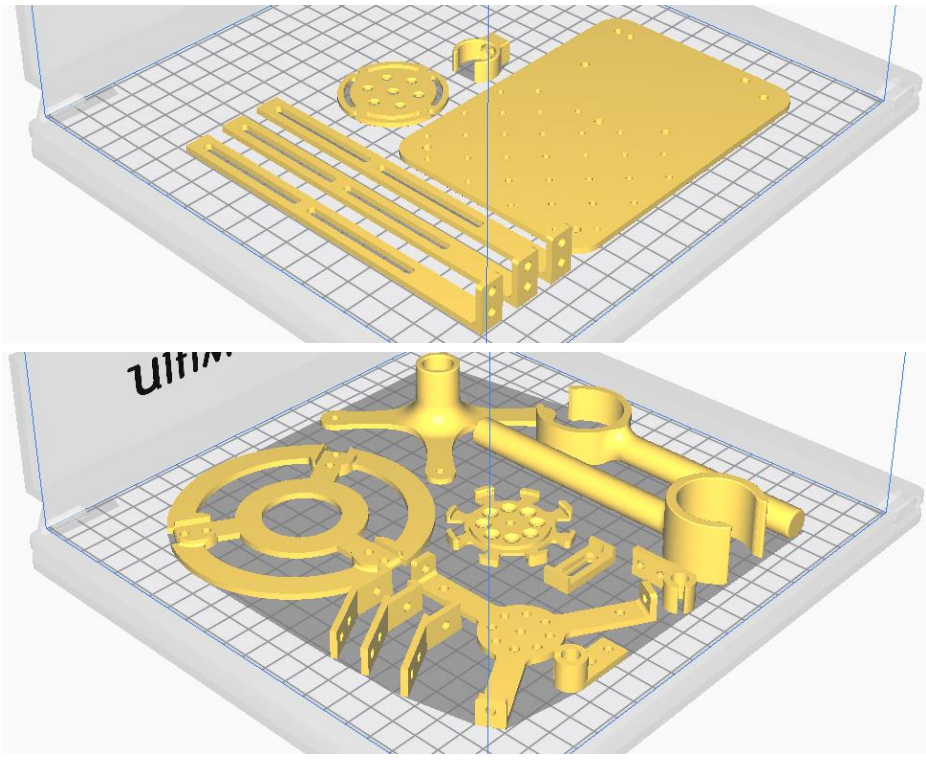
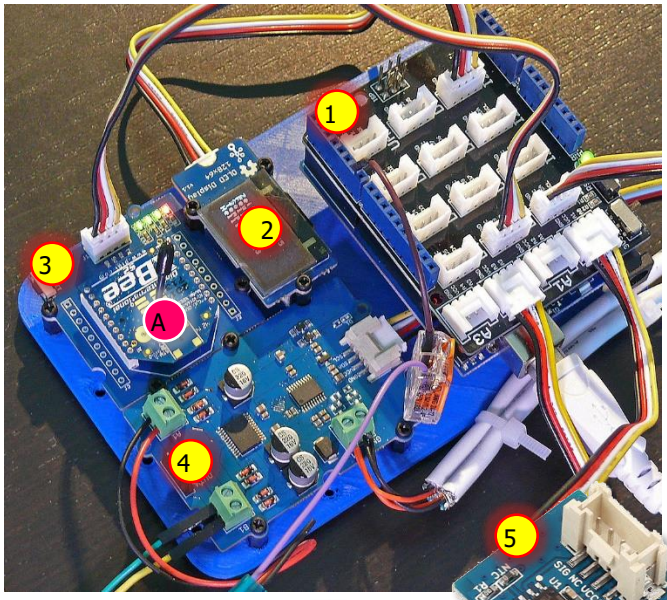
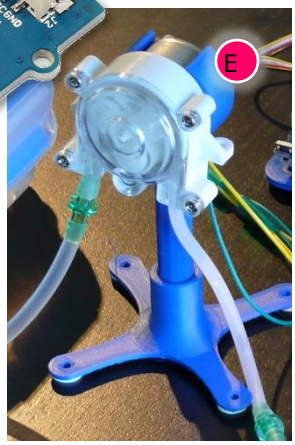


Figure 4: Kompletter Satz von 3D-Daten auf einem Standard 20cm x 20cm Drucker

Elektronische Komponenten:

In dieser Arbeit empfehlen wir das SeeedStudio Grove -System, da es einfach zu bedienen ist: (http://wiki.seeedstudio.com/Grove_System/) Alle Kernkomponenten außer XBees, Feuchtigkeitssensoren und LED-Beleuchtung gehören zum Standard:





Seeed Studio Komponenten:

1: Grove Base Shield für Arduino-Uno

2. Grove OLED 128x64

3. Grove Bee socket

4. Grove - I2C Motor Treiber (TB6612FNG)

5. Grove Temperatur Sensor v1.2

Übliche Sensoren und Aktoren:

1: Arduino Uno (or gleichwertig)

A: XBee Series 2C or Series 3

B: Adafruit Neopixel-Ring mit 16 RGBW (NICHT RGB!) bei 4500 K (warm-weiß)

C: SHT20-Temperatur and Feuchtigkeits-Sensor in wasserdichter Hülle or

D: Kapazitiver analoger Bodenfeuchte-Sensor

E: Peristaltische Pumpe mit 6V DC motor

Sonstige Teile:

Figure 5: Nylon Standoffs

- 5-6 mm Siliconschläuche (Aquariumbedarf)
- Adapter, um Schläuche und Peristalitische P. zu verbinden
- M3 Schrauben and Flügelmuttern
- M3 Nylon Standoffs (Hex spacer)
- M2 Nylon Standoffs (Hex spacer) für Grove (2mm)
- Grove Kabel
- WAGO Klemmen
- Jumper wires
- Pflanzenkübel 8cm Durchmesser
- USB Ladegerät mit 2-2.5A Maximalstromstärke
- XBee USB adapter (e.g. https://www.waveshare.com/xbee-usb-adapter.htm)

Pflanzen:

Geeignet für Experimente in der Schule sind schnell wachsende Pflanzen, die "Microgreens" / "Microgreen Sprouts" genannt werden; sie sind ungiftig und essbar:

Figure 6: Kresse

- Kresse
- Mung Bohnen
- Roter Stielrettich
- Rotkleesprossen
- Broccoli Sprossen
- Valerianella locusta (“Vit” field salad)

Computer mit der folgenden Software vorinstalliert:

- Autodesk Fusion 360 (oder jede andere 3D-Modellierungssoftware, z.B. Wings3D)
- CURA slicing software,
- Eine Internetverbindung zum Herunterladen von Bibliotheken
- Arduino IDE
- Processing IDE
- XCTU Software für XBees

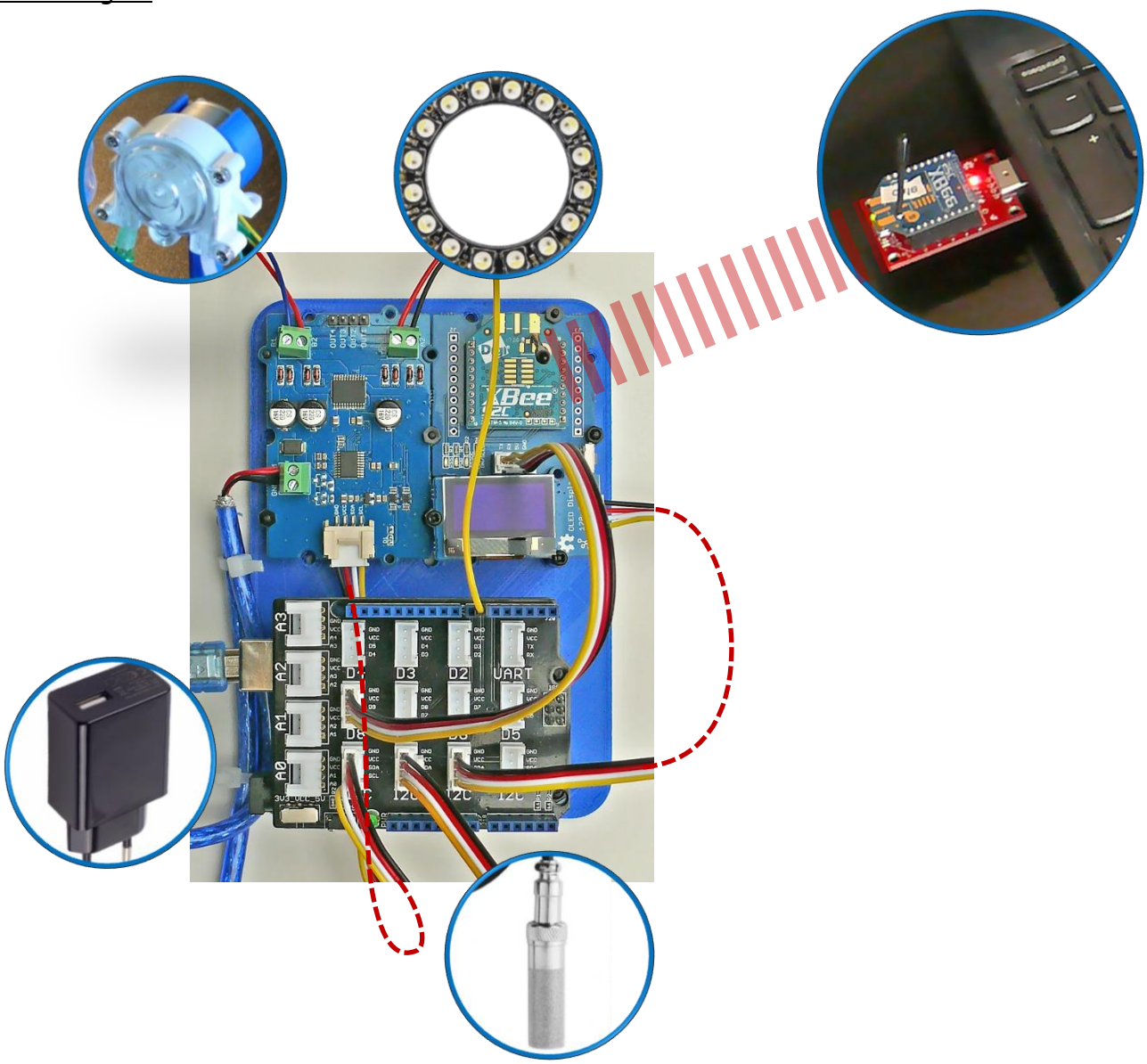
Arduino Bibliotheken:

Einige Komponenten wie der SHT20-Feuchtigkeitssensor oder der Motortreiber benötigen Bibliotheken, damit sie mit der Arduino IDE ordnungsgemäß funktionieren. Wie eine Bibliothek importiert wird, wird hier beschrieben: 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

Spezielle Code-Schnipsel für Temperatursensor finden Sie hier:
http://wiki.seeedstudio.com/Grove-Temperature_Sensor_V1.2/

Verbindungen:



8b Theorie der LED-Growlights und Bodenfeuchtigkeitsmessung

LED growlighting

Grundlage für die Verwendung von LEDs für den Pflanzenbau ist die Theorie der PAR, "photosynthetisch aktive Strahlung": Pflanzen verwenden Lichtphotonen für chemische Reaktionen, um Zucker aus Kohlendioxid zu bauen; diese chemischen Reaktionen treten mit Chlorophyllpigmenten in den Chloroplasten jeder Pflanzenzelle auf.

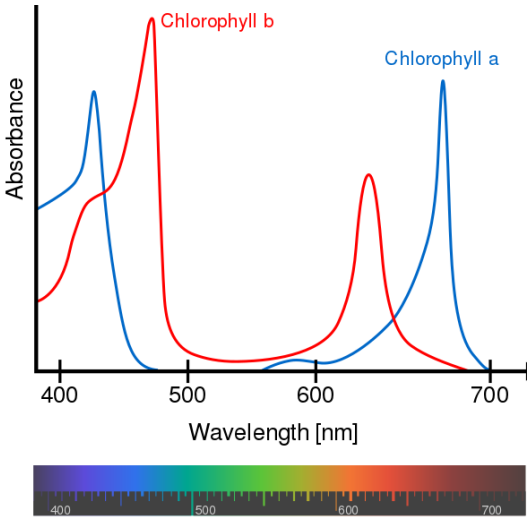


Figure 7: Das Absorptionsspektrum von Chlorophyll a und Chlorophyll b Pigmente. https://en.wikipedia.org/wiki/Chlorophyll_b

Chlorophyll, wenn es mit Sonnenlicht bestrahlt wird, absorbiert rotes und blaues Licht. Die grünen Farbteile werden nicht direkt für den photosynthetischen Prozess absorbiert. Daher sind Pflanzen grün.

LED-Licht für den Pflanzenbau muss hauptsächlich blaues und rotes Licht aus dem Chlorophyll-Absorptionsspektrum liefern. Deshalb verwenden wir die "R" und die "B" Anteile von Neopixel RGBW-High Power LEDs. Der grüne Teil der LED wird von der Pflanze nicht verwendet.

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

Aber eine Pflanze nutzt auch andere Teile des kontinuierlichen Sonnenlichtspektrums; der photosynthetische Prozess ist komplex und ein Gebiet aktueller Forschung: Grünes Licht taucht tiefer in ein Blatt ein und macht den photosynthetischen Prozess effizienter, da es die Absorptionsrate von Chlorophyll beeinflusst. Daher sind kleine Mengen an kontinuierlichem grün- bis gelbem Spektrum notwendig, da es die Absorptionsrate von Chlorophyll beeinflusst. Daher sind kleine Mengen an kontinuierlichem grün- bis gelbem Spektrum wichtig. https://academic.oup.com/pcp/article/50/4/684/1908367

Darüber hinaus wird der Pflanzenanbau von Pflanzenhormonen beeinflusst, die auch auf Sonnenlicht reagieren und meist ein kontinuierliches Sonnenlichtspektrum benötigen.
Phytochrome reagieren beispielsweise auf Infrarot-Beleuchtung.
https://en.wikipedia.org/wiki/Plant_hormone
https://en.wikipedia.org/wiki/Phytochrome

Daher darf sich ein optimales Growlight nicht auf rote und blaue Teile des Spektrums beschränken, sondern benötigt auch einen "weißen" LED-Teil, der ein warmweißes Dauersprectrum erzeugt, um sekundäre photosynthetische Systeme und Pflanzenhormone zu beeinflussen. Deshalb verwenden wir die 4500K-RGBW LED von Adafruit Industries.

Bodenfeuchtesensorik

In der Regel wird Feuchtigkeit als Prozentsatz der Luftfeuchtigkeit gemessen. Daher ist ein Temperatur- und Feuchtigkeitssensor erforderlich. Die Luftfeuchtigkeit selbst kann auf unterschiedliche Weise gemessen werden, und eine der häufigsten Methoden ist die kapazitive Messung. Der Sensor selbst ist ein Kondensator, dessen Kapazität mit der Absorption / Desorption von Wasser verändert wird.

Die Kapazität C eines Kondensator hängt ab von der Plattenfläche A, dem Plattenabstand d und den dielektrischen Eigenschaften des Mediums zwischen den Platten, mit einer Permittivität ε_R:

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

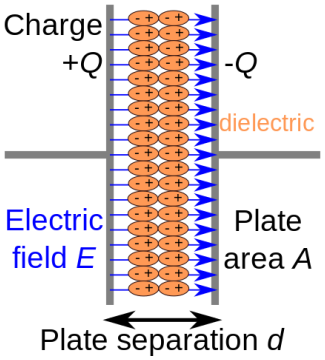


Figure 9:Kondensatorprinzip https://en.wikipedia.org/wiki/Capacitor

Während Abstand und Größe der Platten sich nicht mit der Feuchtigkeit verändern, tut es die Permittivität. In der Regel wird die Permittivität eines gegebenen Stoffes mit der Permittivität des perfekten Vakuums verglichen und "relative Permittivität" genannt. Hier sind einige wichtige Werte für die relative Permittivität:

Medium	*Rel.permittivity*
VaKuum	*1*
Luft	*1.0006*
Wasser	*80*
Tockener Mineralboden	*5*

Infolgedessen nimmt die Kapazität des Bodens umso mehr zu, je mehr Wasser er enthält. Der wässrige, nasse Boden hat eine deutlich höhere Permittivität als sein trockenes Pendant. In der Regel wird es als sogenannter "volumetrischer Bodenwassergehalt", SWC, gemessen. Es ist definiert als:

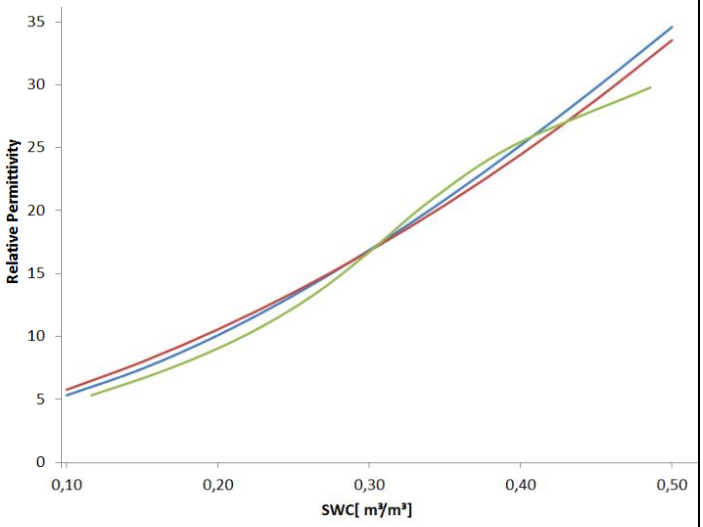


Figure 10: Porretta, Bianchi "Profiles of relative permittivity and electrical conductivity from unsaturated soil water content models", ANNALS OF GEOPHYSICS, 59, 3, 2016, G0320

$$SWC = \frac{Volumen\ an\ Wasser}{Volumen\ an\ Boden}$$

Als RC-Schaltung wird eine elektronische Schaltung zur Messung dieser Kapazitätsänderungen aufgebaut. Je nach Kapazität verfügt eine RC-Schaltung über eine charakteristische Zeitkonstante, die von einem Mikrocontroller gemessen werden kann. Je größer die Kapazität, desto größer die Zeitkonstante. Zusammenfassend wird die Luftfeuchtigkeit auf diese Weise gemessen:

$$steigende\ Feuchtigkeit \xrightarrow{führt\ zu} steigender\ Kapazität \xrightarrow{führt\ zu} steigender\ Zeitkonstante$$

Vergleich SHT20 im Gehäuse und Bodenfeuchtesensor:

SHT 20	Soil moisture sensor
	
Misst Luftfeuchtigkeit im wasserdichten Gehäuse und gleichzeitig die Temperatur.	Misst direkt die relative Permittivität des Bodens. Zusätzliche Temperaturmessung ist notwendig.
Kommuniziert via I2C mit dem Arduino Microkontroller	Erzeugt ein analoges Signal, das mit dem ADC des Arduino-Mikrocontrollers digitalisiert werden muss.
Darf nicht vollständig in den Boden gelegt werden, da Luftstrom notwendig ist	Muss so tief wie möglich im Boden gehalten werden
Kosten etwa 20 Euro pro Stück	Kosten etwa 5 Euro pro Stück
Achtung: Da dieser Sensor eine relative Luftfeuchtigkeit als Ausgang erzeugt, machen die Messwerte für die Bodenverhältnisse keinen Sinn. Meistens erzeugt der Sensor Werte über 100%, da die Luft im Inneren des wasserdichten Gehäuses mit Bodenfeuchtigkeit gesättigt ist. Sie müssen die rohen 16-Bit-Werte verwenden, die nicht in die nicht sinnvollen Luftfeuchtigkeitswerte konvertiert werden.	ACHTUNG: Da dieser Sensor die relative Permittivität des Bodens misst, ist es absolut notwendig, dass der Sensor in perfekter Berührung mit dem Boden ist, ohne Luftschicht zwischen beiden Oberflächen. Darüber hinaus driften die Sensorwerte, da sich bei der Bewässerung das Volumen des Bodens ändert und damit auch die Deckfläche des Sensors. Darüber hinaus verändern wachsende Pflanzen auch die Zulassungswerte.

8c Arduino Quellcode Beispiel

```
#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 mit Quellcode

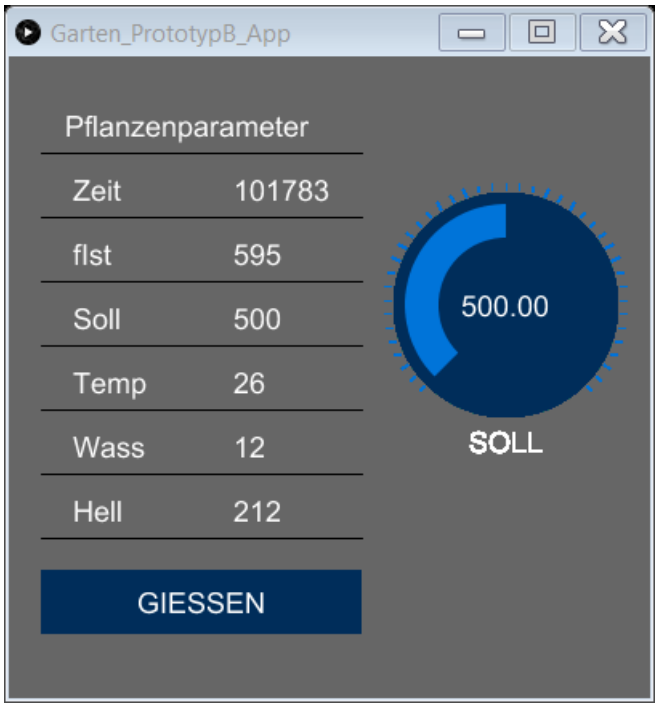


Figure 11: Screenshot App interface

Ein typisches IoT-Gerät verfügt in der Regel über eine App-Schnittstelle, die es dem Benutzer ermöglicht, sein verbundenes Gerät zu überwachen und fernzusteuern. Eine (relativ) leichte Möglichkeit, dies mit Studenten zu erreichen, besteht darin, serielle Daten aus dem XBee zu lesen, und die Werte auf eine grafische Benutzeroberfläche auszugeben.

Ausserdem sind die Arduino IDE und die Processing-IDE eng miteinander verwandt, da die Arduino IDE eine Processing-"Fork".

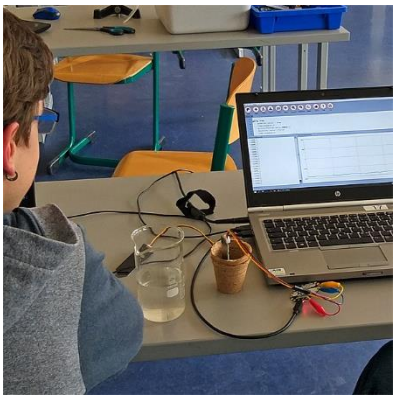
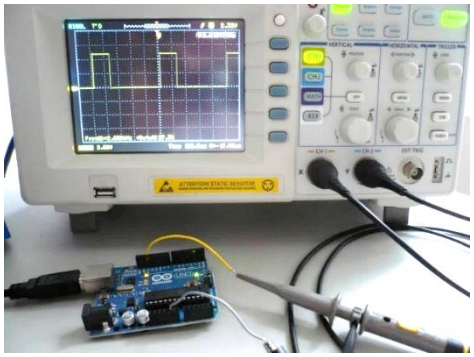
Für den Knopf "GIESSEN" und den Rundknopf "SOLL" wird die "ControlP5"-Bibliothek verwendet, die einfach aus der Verarbeitungs-IDE heraus installiert werden kann.

ControlP5 siehe: https://code.google.com/archive/p/controlp5/downloads

8c. Beispiel Processing Quellcode

```
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));
  }
}
```

9. Unterrichts-plan, Schritt für Schritt Beschreibung der Aktivität / Inhalte	**Lektionen 1 & 2 (90min):** Es werden Beispiele vorgestellt: Staubsauger-Roboter mit App-Fernbedienungen, internetbasierte Wetterstationen, Aktivitätstracker mit App-Kommunikation und nicht intelligente landwirtschaftliche Betriebe. Die Kursteilnehmer sollten untersuchen, wie diese Geräte funktionieren und welche Komponenten benötigt werden: ein Mikrocontroller-basiertes System steuert und koordiniert angeschlossene Sensoren/Aktoren. Darüber hinaus kommuniziert und koordiniert es mit anderen Systemen ähnlicher Art über drahtlose Kommunikationsnetze. Benötigte Teile: Sensoren, Schauspieler, Kommunikationsgeräte. Möglichkeiten/Gefahren werden diskutiert: Wo macht IoT Sinn und wo nicht? **Lektionen 3&4 (90 min)** Die Studierenden sollten die Komponenten für die Überwachung und Optimierung des Pflanzenwachstums planen. Danach können sie beginnen, eine solche Maschine von Grund auf neu zu bauen, indem sie die mitgelieferten Teile verwenden. Nach Abschluss können grundlegende Programmiertechniken vermittelt werden, um den Schülern die Möglichkeit zu geben, ihre eigenen Experimente durchzuführen. **Lektionen 5&6 (90 min)** Theorie der Bodenfeuchtemessung und Pflanzenbeleuchtung müssen gelehrt werden. Die Schüler können SWC mit ihren individuellen Sensoren messen, um Kalibrierkurven zu erstellen. Die Schüler sollten ihre experimentellen Ergebnisse vergleichen, um zu erkennen, dass jede Schülergruppe ihre eigenen Messwerte hat, die sich erheblich unterscheiden.  *Figure 12: experimenting with soil sensors* **Lektionen 7&8 (90 min):** Die Schüler sollten mit dem Motortreiber und der Programmierung und Steuerung der peristaltischen Pumpen beginnen. Die Theorie der H-Brücken muss unterrichtet werden und zusätzlich sollten die Grundlagen der I2C-Kommunikation zwischen elektronischen Gerätekomponenten erklärt werden. Die Schüler sollten die I2C-Kommunikation mit Oszilloskopen messen. Das Konzept der PWM (Pulse width modulation) muss eingeführt werden. Darüber hinaus können die Studierenden die Kommunikation zwischen Arduino und den Neopixel LEDs über das Oszilloskop messen.  *Figure 13: PWM Messung am Oszi*

Lektionen 9&10 (90 min):

Aufbau eines Kommunikationsnetzes: XBee-Module und serielle Kommunikation (UART). Studierende sollten XCTU Software als Ausgangspunkt für ihre Kommunikationsexperimente in drahtlosen Netzwerken nutzen.

ACHTUNG: XBees sollten vom Lehrer paarweise vorkonfiguriert werden, da sonst viel Zeit verloren geht, wenn man lernt, wie man mit den vielen verschiedenen Konfigurationsmöglichkeiten umgeht. Grundsätzlich werden XBees mit drei verschiedenen benutzerbasierten Werten definiert, hier in rot:

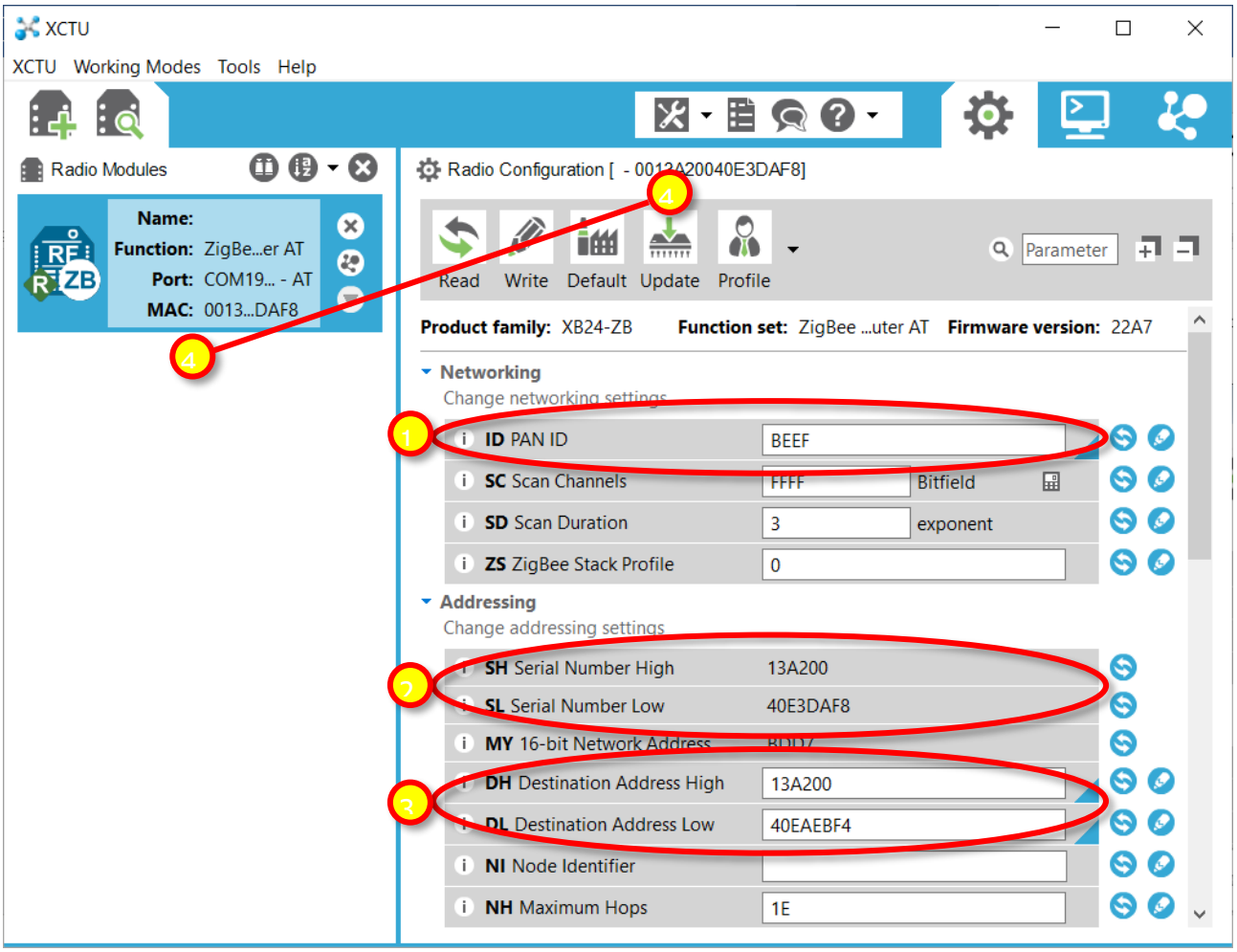
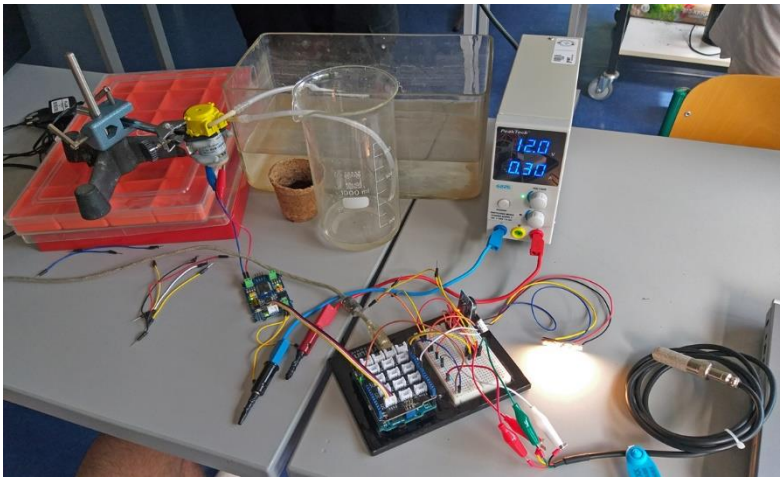
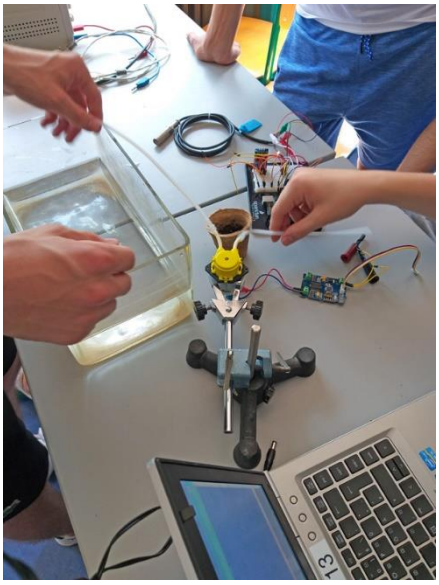


Figure 14: Screenshot XCTU User Interface

1: PAN ID muss gleich sein für beide XBees. Man sollte Hexadezimal-Buchstaben verwenden, zum Beispiel den sogenannten "Hexspeak": "BEEF", "CAFE", "F00D", "AFFE", etc..
2. Die "Serial Number High" muss kopiert werden von einem Xbee zum nächsten.
4. Ein XBee muss als 'Coordinator' und der andere als 'Router' konfiguriert werden. Achtung: In einigen Texten ist zu lesen, dass der zweite als 'Endpunkt' konfiguriert werden sollte. Das kann einige Probleme verursachen, da Endpunkt-XBees aus energiesparenden Gründen „schlafen" gehen.

Danach können die XBees auf den UART-Kommunikationsport des Arduino schreiben/lesen.

	Lektionen 11 bis Ende (270 min): **Freestyle Programmierung! And happy SmartFarming 😊**   

10. Feedback	Am Ende der Lektion sollten die Kursteilnehmer über fundierte Kenntnisse darüber verfügen, wie IoT-Prinzipien funktionieren und wie mit dem Internet verbundene Maschinen kommunizieren. Sie haben selbst Möglichkeiten und Grenzen dieser aktuellen Technologie erlebt. Während des Unterrichts wurden wichtige Aspekte der Elektronik, Informatik und Konstruktionsgrundlagen vermittelt. Darüber hinaus wurden biologische Aspekte des Pflanzenanbaus vermittelt.
11. Bewertung & Evaluation	Die Schüler führen ihr Arbeitstagebuch, das vom Lehrer überprüft werden kann. Die Schüler können auch die Ergebnisse ihrer Experimente präsentieren. Darüber hinaus muss am Ende des Unterrichts ein Standard-Test durchgeführt werden.