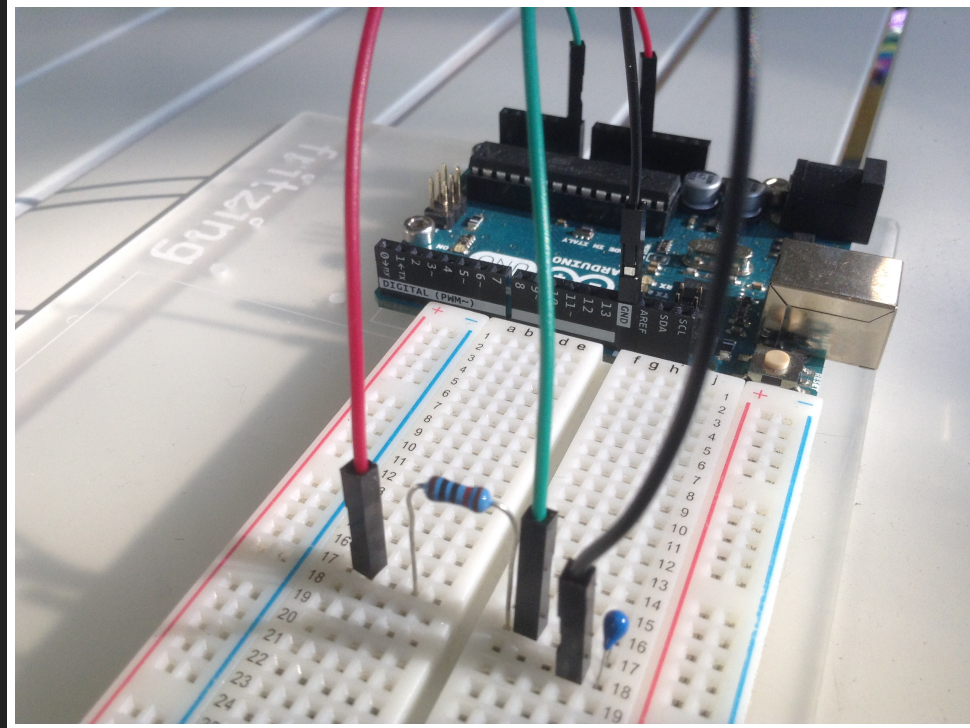


ECHTDATEN MIT ARDUINO

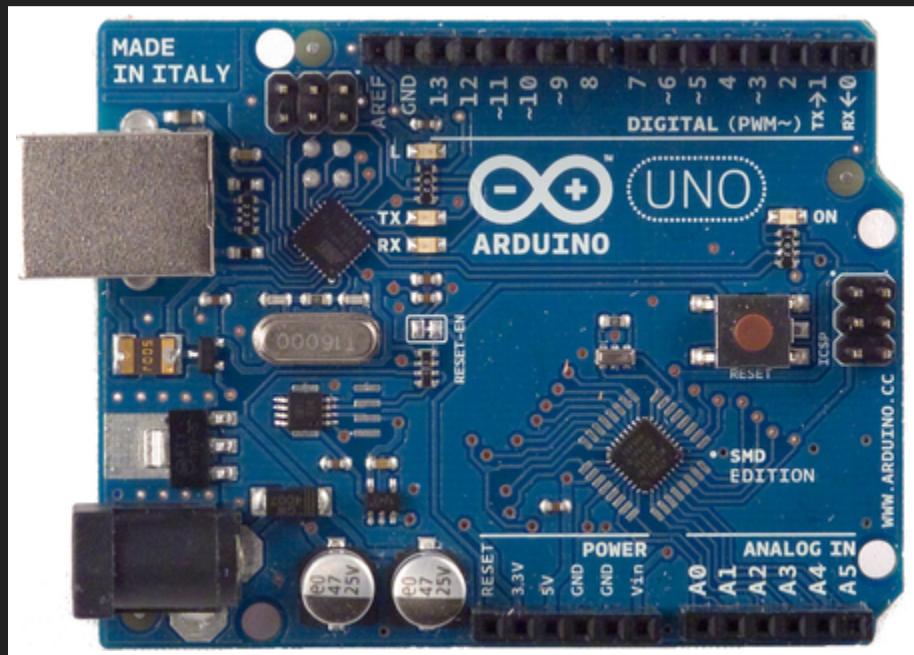


Paola Maneggia

Tag der Informatiklehrerinnen und -lehrer, LMU.

1. Juli 2016

ARDUINO - WOCHE 1

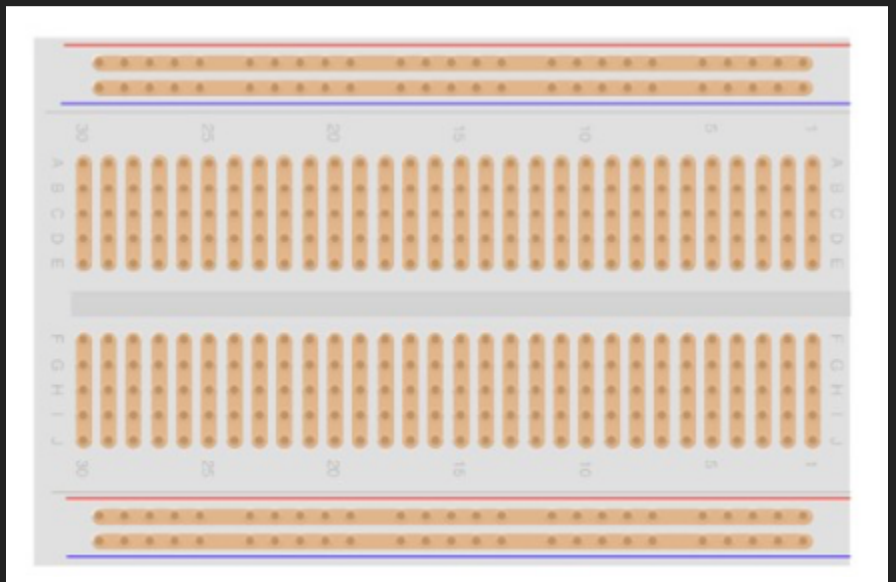
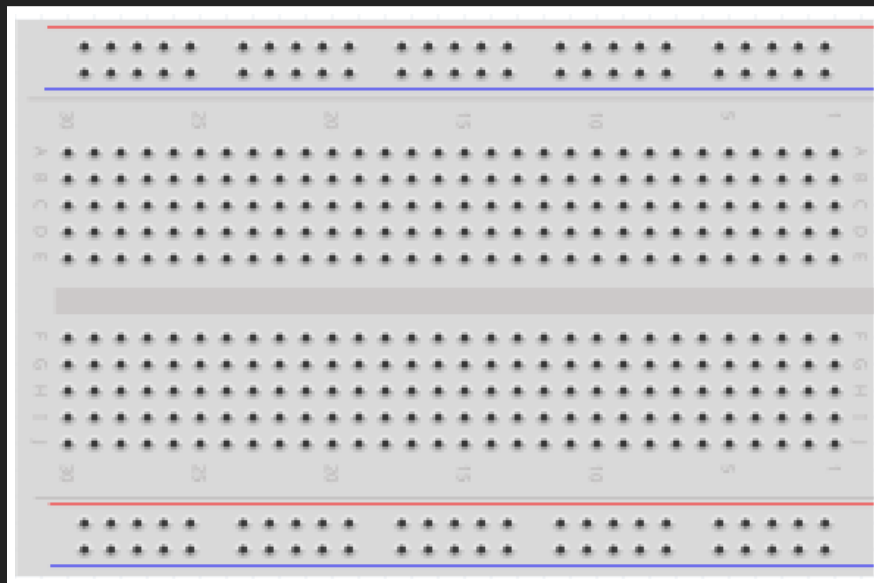


```
sketch_may25a $
void setup() {
  // put your setup code here, to run once:
}

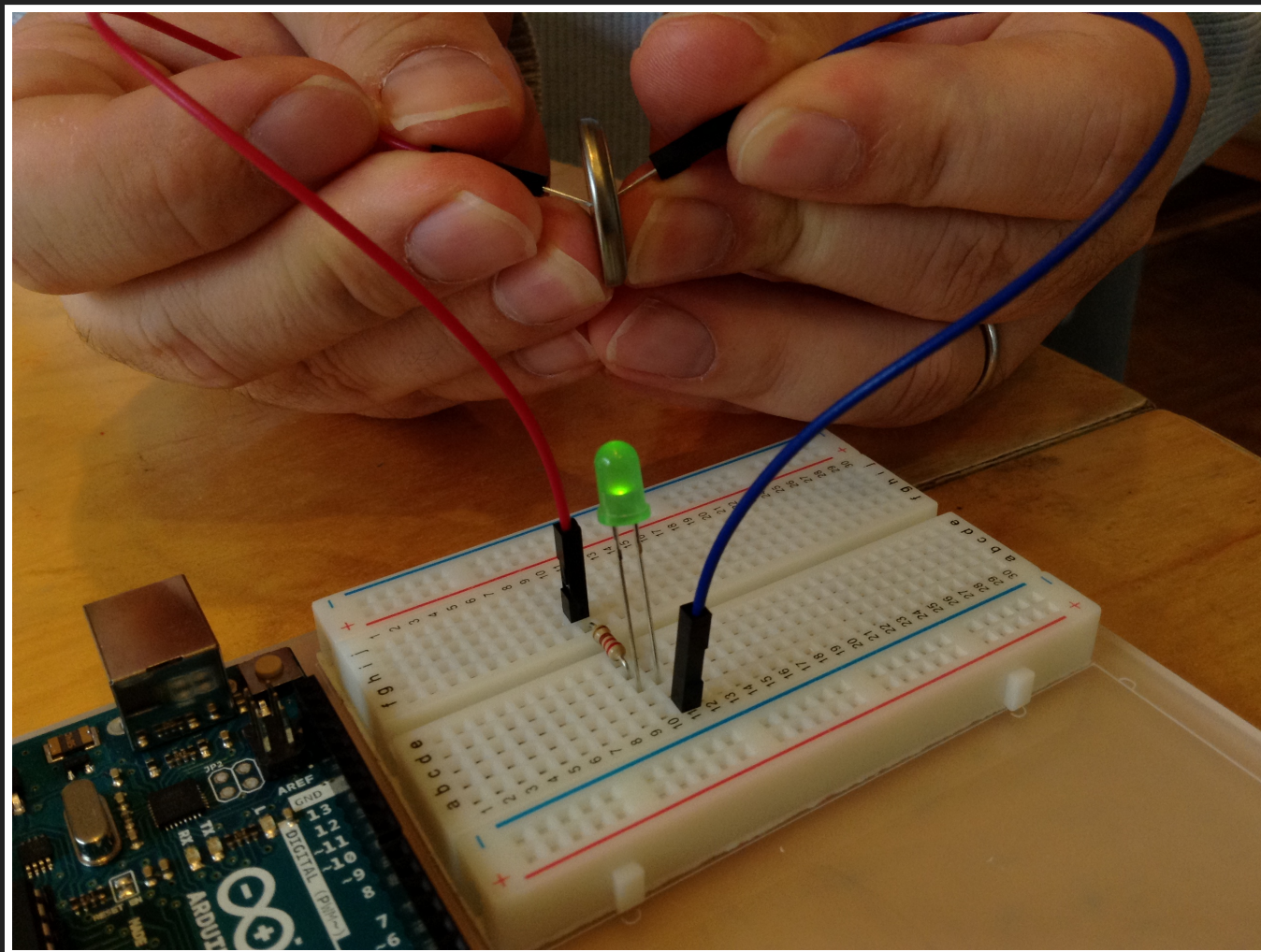
void loop() {
  // put your main code here, to run repeatedly:
}
```

11 Arduino/Genuino Uno auf /dev/cu.usbmodem1411

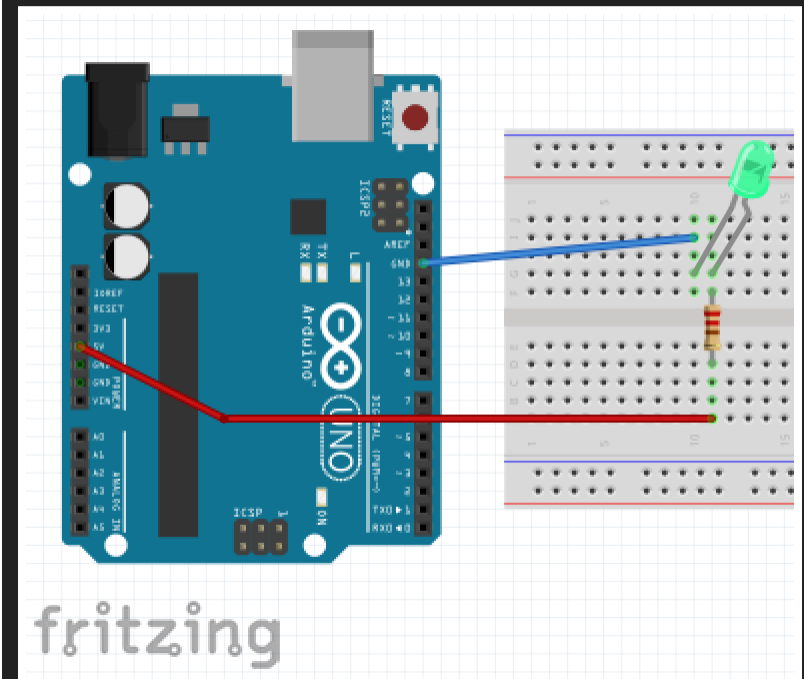
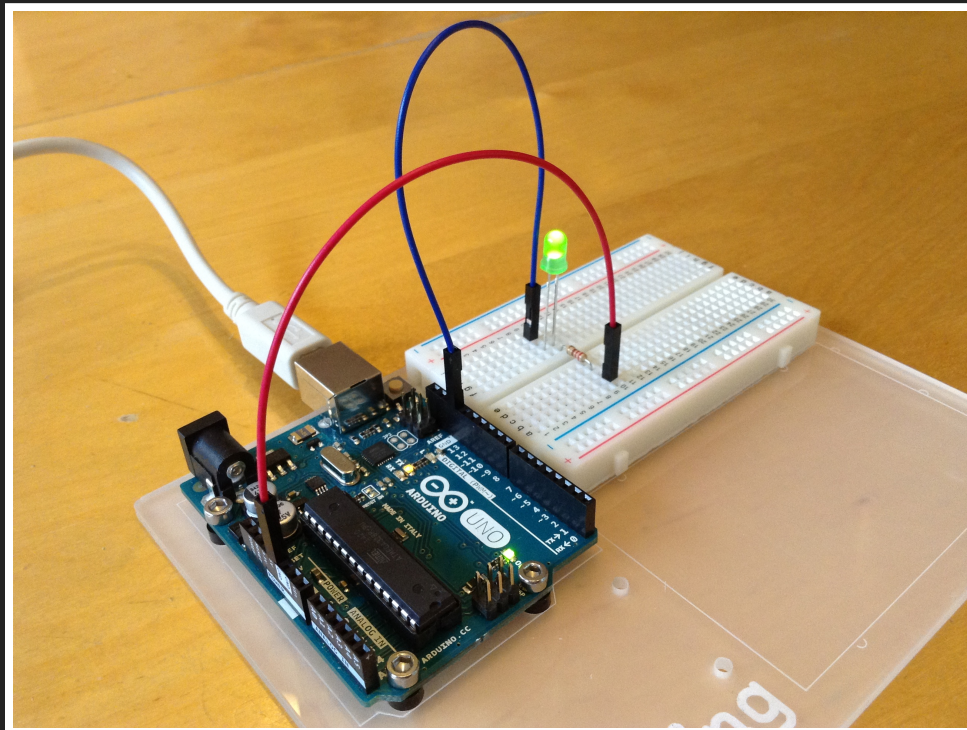
STECKBRETT



SCHALTKREIS MIT BATTERIE

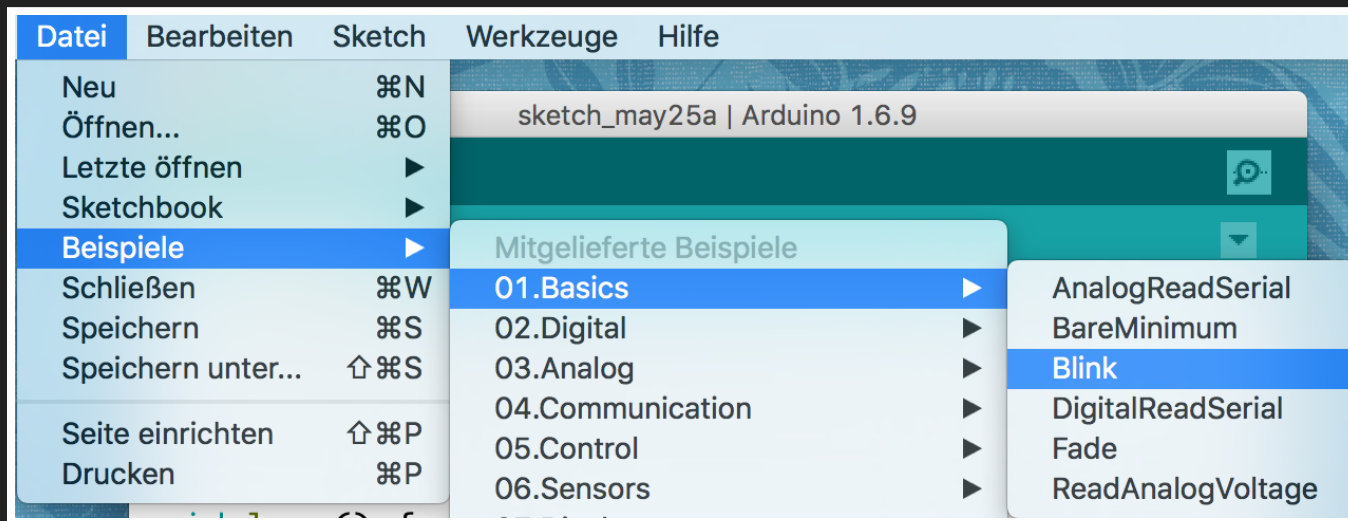


ARDUINO ALS BATTERIE



EINE LED BLINKEN LASSEN:

Stecke den roten Draht von 5V auf digital-Pin 13 um und öffne den Beispiel-Sketch: Datei > Beispiele > 0.1Basic > Blink

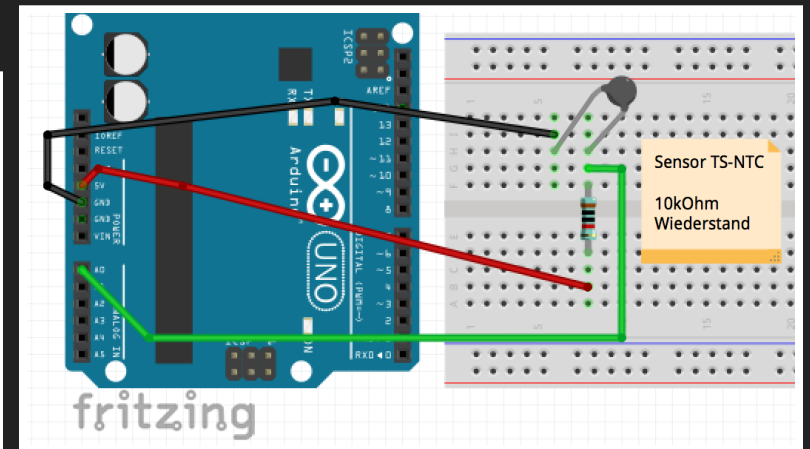
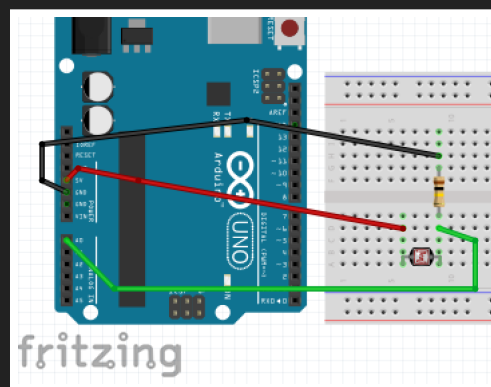
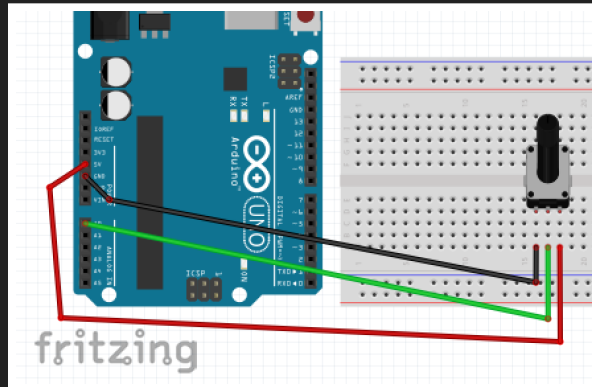


Lade das Programm hoch (zuerst Arduino mit USB-Kabel mit dem Rechner verbinden und im Menü "Werkzeuge" das entsprechende Board und Port auswählen)



WEITERE AUFGABEN

1. Zwei LED abwechselnd blinken lassen
2. Beispiele > 0.1 Basics > DigitalReadSerial: Liest das (digitale) Signal eines Buttons ein und gib es am seriellen Monitor aus
3. Benutze den vom Buttons eingelesenen Wert um zwei LED zu steuern (Grün aus, Rot an und umgekehrt beim Drücken des Buttons): Hier ist eine if-Anweisung nötig, eventuell eine Übersetzung Karol-Arduino mit den Schülern besprechen.
4. Beispiele > 0.1 Basics > AnalogReadSerial: Liest das (analoge) Signal eines Sensors (mit einem Potentiometer, einem Lichtsensor, einem Temperatursensor TS-NTC-103, ...)



DIE WICHTIGSTEN ARDUINO-BEFEHLE

Die wichtigsten Arduino-Befehle



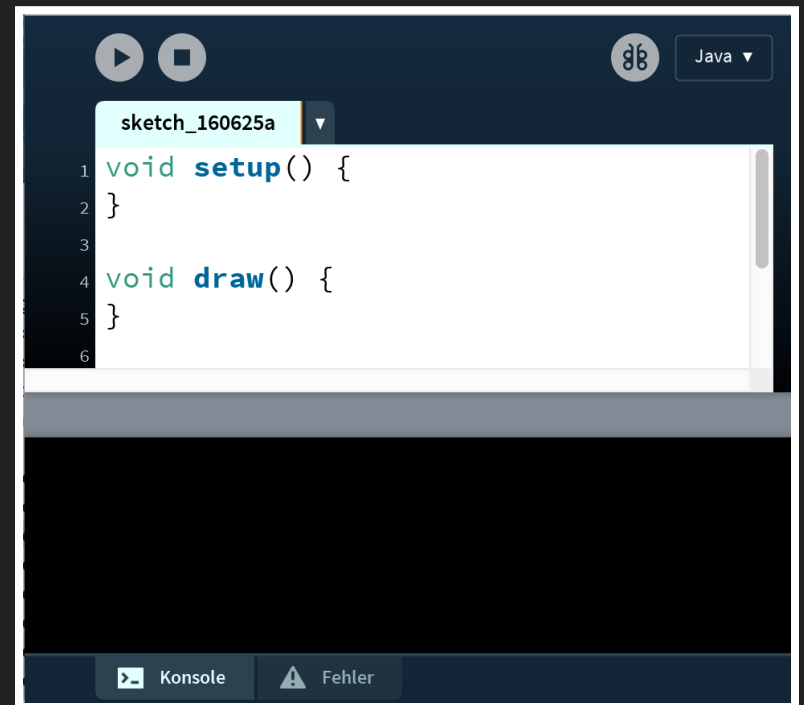
- Digital-Pins: 1, 2, ..., 13 (digital: die möglichen Zustände sind 0 und 1)
- Analog-Pins: A0, A1, A2, A3, A4, A5 (analog: Die möglichen Zustände sind alle Werte zwischen 0 und 1023)
- `pinMode(10, OUTPUT);` Bestimmt, dass Pin 10 als Ausgabe konfiguriert ist.
- `PinMode(A2, INPUT);` Bestimmt, dass Pin A2 als Eingabe konfiguriert ist.
- `analogRead(A0), digitalRead(13);` liest den Wert von Pin A0 bzw. 13.
- `analogWrite(A0, 512), digitalWrite(13, LOW);` schreiben auf den jeweiligen Pin
- Die gelesenen Werte dürfen verglichen oder in Berechnungen weiter benutzt werden:
 - `analogRead(A0) < 900` (ist entweder `true` (wahr) oder `false` (falsch), je nach gelesenen Wert.)
 - `digitalRead(13) == LOW` (ist der Wert am Pin 13 gleich 0? Ergebnis → `true` oder `false`)
 - `2*analogRead(A0) + 1000` (verdoppelt den gelesenen Wert und addiert 1000 dazu)
 - Die Vergleichsoperatoren sind (ähnlich wie in Rechnerblättern):
 - `<` `<=` `>` `>=`
 - `==` ist gleich
 - `!=` ist ungleich (in Rechnerblätter haben wir stattdessen `<>`)
(zum Beispiel `3 != 5` „drei ist ungleich 5“ → Wert `true` bedeutet: es ist wahr, dass 3 ungleich 5 ist)
- `//` startet einen Kommentar: alles was bis dem Zeilenvorschub folgt, wird vom Programm ignoriert.

Arduino braucht immer zwei Blöcke von Befehlen: `setup() { ... }` und `loop() { ... }`

- Im **setup**-Block wird die Anfangskonfiguration gemacht;
- Im **loop**-Block wird bestimmt, was unser Programm macht. Die darin angegebenen Befehle werden der Reihe nach unendlich oft wiederholt.

PROCESSING - WOCHE 2

Die Entwicklungsumgebung von Processing ist sehr ähnlich wie die von Arduino. Die voreingestellte Sprache ist Java. Processing kann u.a. die serielle Übertragung von Daten aus Arduino einlesen, Werte in graphische Darstellungen umwandeln und mit einer Datenbank kommunizieren.



ERSTE AUFGABE

Als wir letzte Woche die Schaltung mit dem Potentiometer gebaut haben, haben wir die von Arduino ausgelesenen Werte mit `Serial.print ()` auf dem Serial-Monitor angezeigt.

Diesmal lesen wir diese Werte mit Processing ein und stellen sie graphisch dar.

Dieses Beispiel ist in Arduino, Menü: Datei > Beispiele > 0.4 Communication > Graph.

Der Code für Arduino ist genau wie im Beispiel `AnalogReadSerial`. Der Code für Processing ist in dem Arduino-Sketch in einem Kommentar zu finden.

SCHRITTE

- Baue die Schaltung mit dem Potentiometer wieder auf
- Öffne in Arduino Beispiele > 0.4 Communication > Graph
- Starte Processing und kopiere den Code für Processing in einen leeren Sketch
- Lade eventuell das Programm auf Arduino neu hoch und starte danach Processing. Eine graphische Darstellung der eingelesene Daten wird in einem Fenster gezeichnet. Drehe den Potentiometer, um die Werte zu ändern und betrachte was mit der graphischen Darstellung passiert.

Graph | Arduino 1.6.9

Graph

```
/* Processing code for this example

// Graphing sketch

// This program takes ASCII-encoded strings
// from the serial port at 9600 baud and graphs
// range 0 to 1023, followed by a newline,

// Created 20 Apr 2005
// Updated 24 Nov 2015
// by Tom Igoe
// This example code is in the public domain

import processing.serial.*;

Serial myPort;          // The serial port
int xPos = 1;           // horizontal position
```

sketch_160626a | Processing 3.1.1

sketch_160626a

```
// Graphing sketch

// This program takes ASCII-encoded strings
// from the serial port at 9600 baud and graphs
// range 0 to 1023, followed by a newline,

// Created 20 Apr 2005
// Updated 24 Nov 2015
// by Tom Igoe
// This example code is in the public domain

import processing.serial.*;

Serial myPort;          // The serial port
int xPos = 1;           // horizontal position
float inByte = 0;

void setup () {
```


DER INDEX FÜR DEN PORT ANPASSEN:

sketch_160626a

```
20 void setup () {  
21   // set the window size:  
22   size(400, 300);  
23  
24   // List all the available serial ports  
25   // if using Processing 2.1 or later, use Serial.printArray()  
26   println(Serial.list());  
27  
28   // I know that the first port in the serial list on my mac  
29   // is always my Arduino, so I open Serial.list()[0].  
30   // Open whatever port is the one you're using.  
31   myPort = new Serial(this, Serial.list()[3], 9600);  
32
```

Index des Arduino-
Ports anpassen

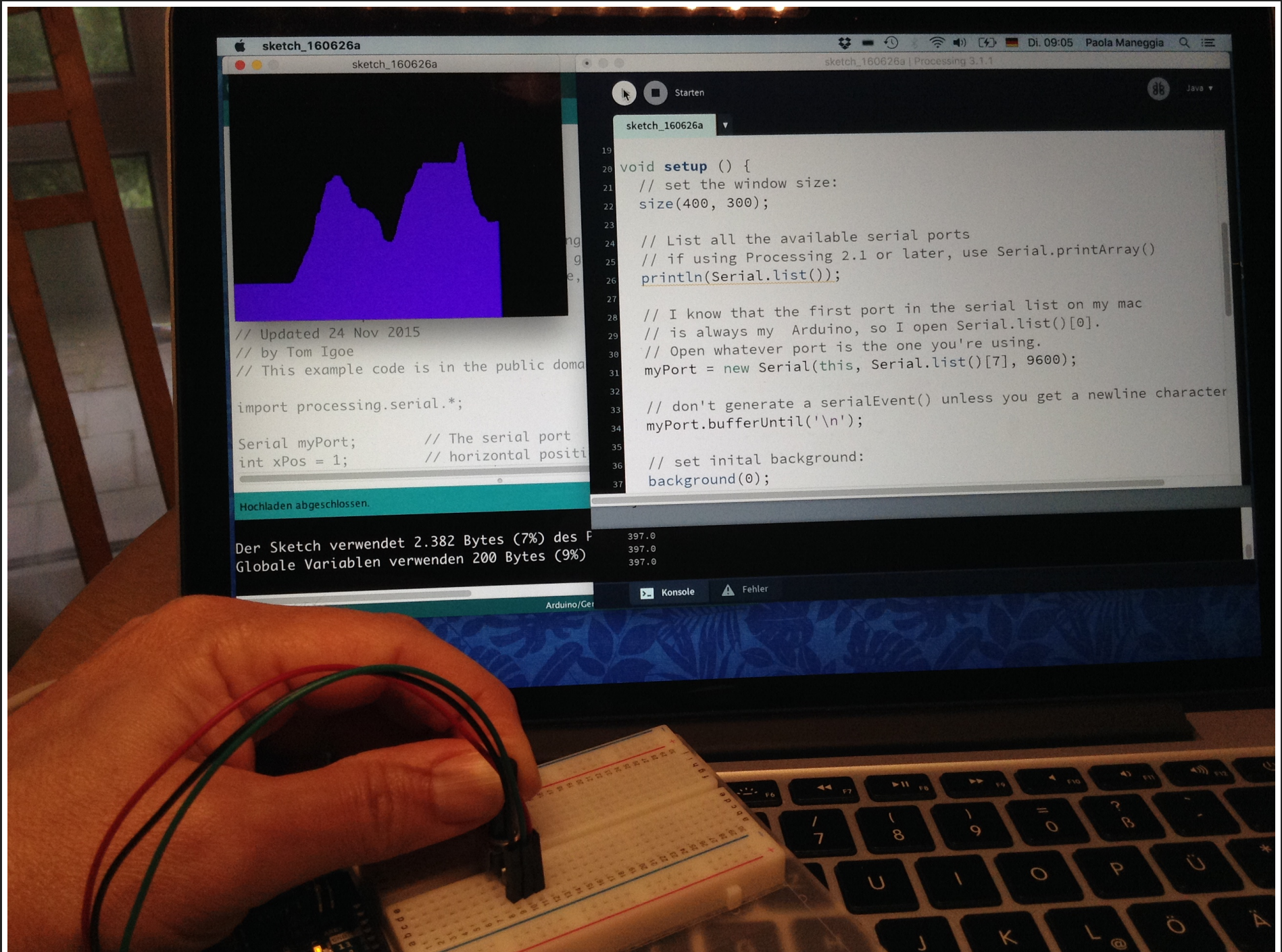
0 1 2
/dev/cu.Bluetooth-Incoming-Port /dev/cu.MathiassiPad-WirelessiAP /dev/cu.PaolasiPad-WirelessiAP
3 /dev/cu.usbmodem1421 /dev/tty.Bluetooth-Incoming-Port /dev/tty.MathiassiPad-WirelessiAP
/dev/tty.PaolasiPad-WirelessiAP /dev/tty.usbmodem1421



Konsole



Fehler



sketch_160626a

sketch_160626a

sketch_160626a | Processing 3.1.1

Starten

sketch_160626a

```
19
20 void setup () {
21   // set the window size:
22   size(400, 300);
23
24   // List all the available serial ports
25   // if using Processing 2.1 or later, use Serial.printArray()
26   println(Serial.list());
27
28   // I know that the first port in the serial list on my mac
29   // is always my Arduino, so I open Serial.list()[0].
30   // Open whatever port is the one you're using.
31   myPort = new Serial(this, Serial.list()[7], 9600);
32
33   // don't generate a serialEvent() unless you get a newline character
34   myPort.bufferUntil('\n');
35
36   // set initial background:
37   background(0);
```

// Updated 24 Nov 2015
// by Tom Igoe
// This example code is in the public domain

import processing.serial.*;

Serial myPort; // The serial port
int xPos = 1; // horizontal position

Hochladen abgeschlossen.

Der Sketch verwendet 2.382 Bytes (7%) des P
Globale Variablen verwenden 200 Bytes (9%)

397.0
397.0
397.0

Konsole

Fehler

Arduino/Ger

METHODE SERIALEVENT

```
void serialEvent (Serial myPort) {  
    // get the ASCII string:  
    String inString = myPort.readStringUntil('\n');  
  
    if (inString != null) {  
        // trim off any whitespace:  
        inString = trim(inString);  
        // convert to an int and map to the screen height:  
        inByte = float(inString);  
        println(inByte);  
        inByte = map(inByte, 0, 1023, 0, height);  
    }  
}
```

AUFGABEN

- Ändere die Farbe des Graphen (RGB-Kenntnis nötig - Eingabewerte in `stroke(. . .)`).
- Ändere den Graphen, so dass nur eine Kurve gezeichnet wird.
- Ändere die `SerialEvent()`-Methode, so dass in der Konsole vor jedem Wert auch ein Text "Wert: " erscheint.

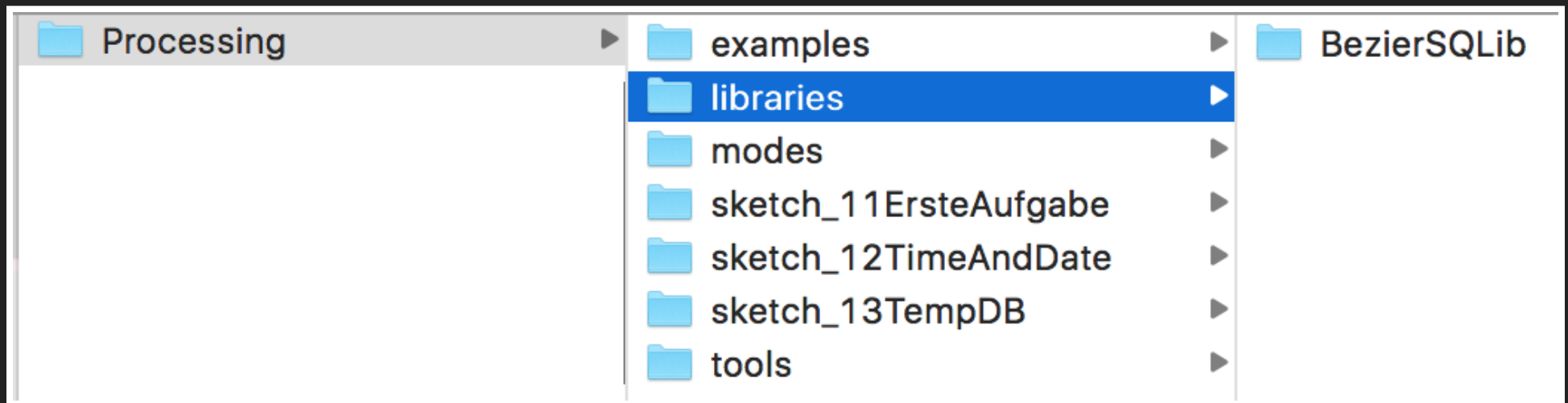
PROCESSING: VERBINDUNG MIT EINER DATENBANK - WOCHE 3

Für die Verbindung mit einer sqlite Datenbank habe ich eine praktische Bibliothek, **BezierSQLib** von Florian Jannet <http://fjenett.github.io/sql-library-processing/> verwendet.

Processing2 → Processing3 im Mai 2016!

In Processing3 sieht es immer noch so aus, als ob die Bibliothek nicht mehr kompatibel ist. Dies wird sich hoffentlich bald ändern. Mittlerweile finden Sie einen funktionierenden Build auf der TdI-Website: BezierSQLib.zip

Den Ordner **BezierSQLib** in den Ordner "Libraries" (ggf. diesen selbst erstellen) in das Sketchbook von Processing kopieren. Möglicherweise Processing beenden und neu starten.



SQLITE

- Öffne die Datei name-age.db in sqlite3 mit `.open name-age.db`

```
[sqlite> .table
age9b
[sqlite> .schema
CREATE TABLE age9b(name text, age integer);
[sqlite> select * from age9b;
Daniel|16
Pavao|15
Tobias|15
```

- Füge eine Zeile hinzu, z.B. Verena, 16, mit `INSERT INTO age9b VALUES ('Verena', 16);`
- Gib die Tabelle mit `SELECT * FROM age9b;` aus
- Eine knappe Zusammenfassung von sqlite-Befehlen ist auf der TdI2016-Website zu finden.

VERBINDUNG MIT EINER DATENBANK, ANFRAGE MIT AUSGABE IN DER KONSOLE UND UPDATE

Erzeuge einen Unterordner data des aktuellen Sketch-Ordners. Kopiere die Datei name-age.db in diesen.

```
import de.bezier.data.sql.*;

SQLite db;

void setup() {
    db = new SQLite( this, "name-age.db" ); // open database file
    db.connect();

    // update
    db.execute("insert into age9b values ('Paul', 15)");

    // read all in table age9b
    db.query( "SELECT * FROM age9b" );
    while (db.next()) {
        println(db.getString("name") + " " + db.getInt("age"));
    }
}
```

AUFGABE

Nach mehreren Tests hat wahrscheinlich fast jeder, wie auch ich, die Zeile "Paul, 15" mehrmals eingefügt. Ändere das Processing-Programm, so dass alle solche Zeilen aus der Tabelle gelöscht werden!

Lösung

```
db.execute("delete from age9b where  
name='Paul'");
```

ZEIT IN PROCESSING

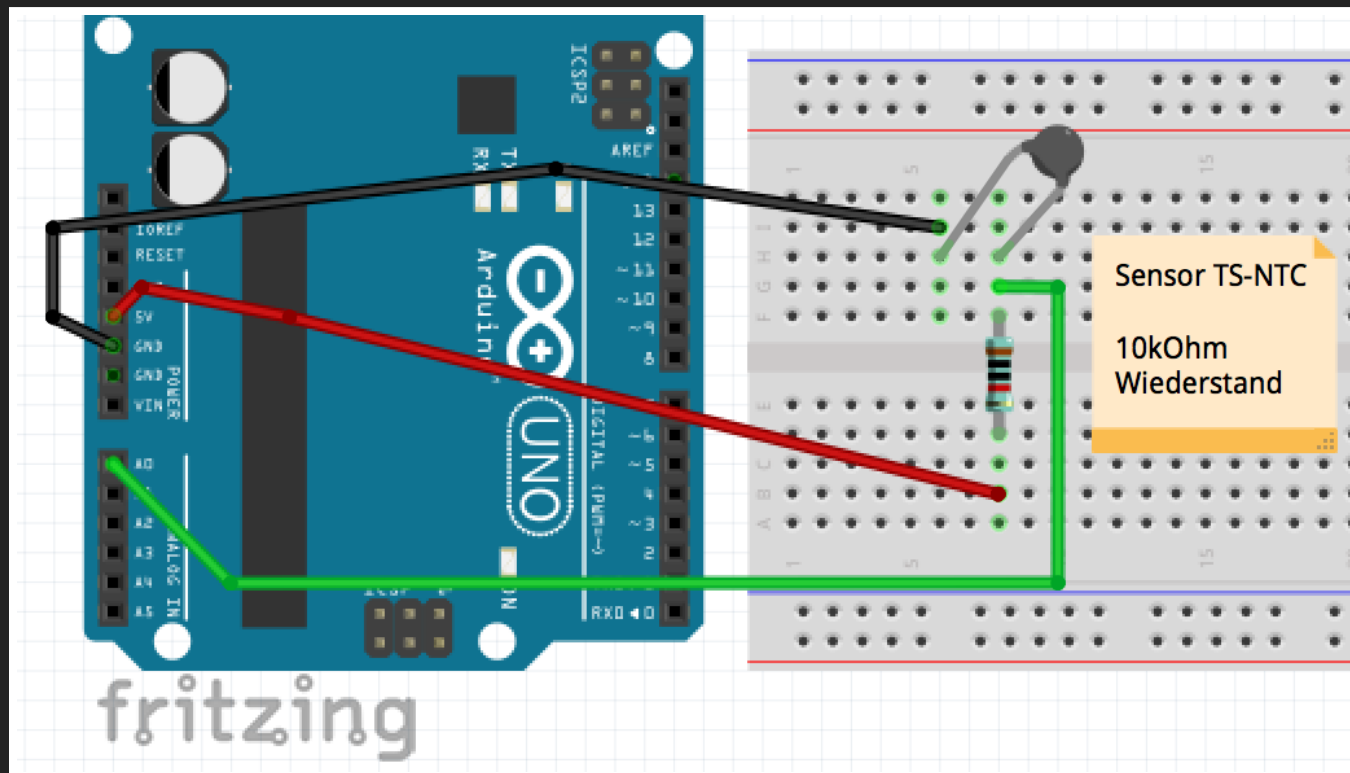
```
void setup() {  
    println("Heute ist der " + day() + "." + month() + "." + year());  
    // mehr über Processing Time & Day hier:  
    // https://processing.org/reference/day\_.html  
  
    // Aufgabe: Wie kannst du die Stunde, Minute und Sekunde abfragen?  
}  
  
void draw() { }
```

TEMPERATURDATENBANK - WOCHEN 4

ÜBERBLICK

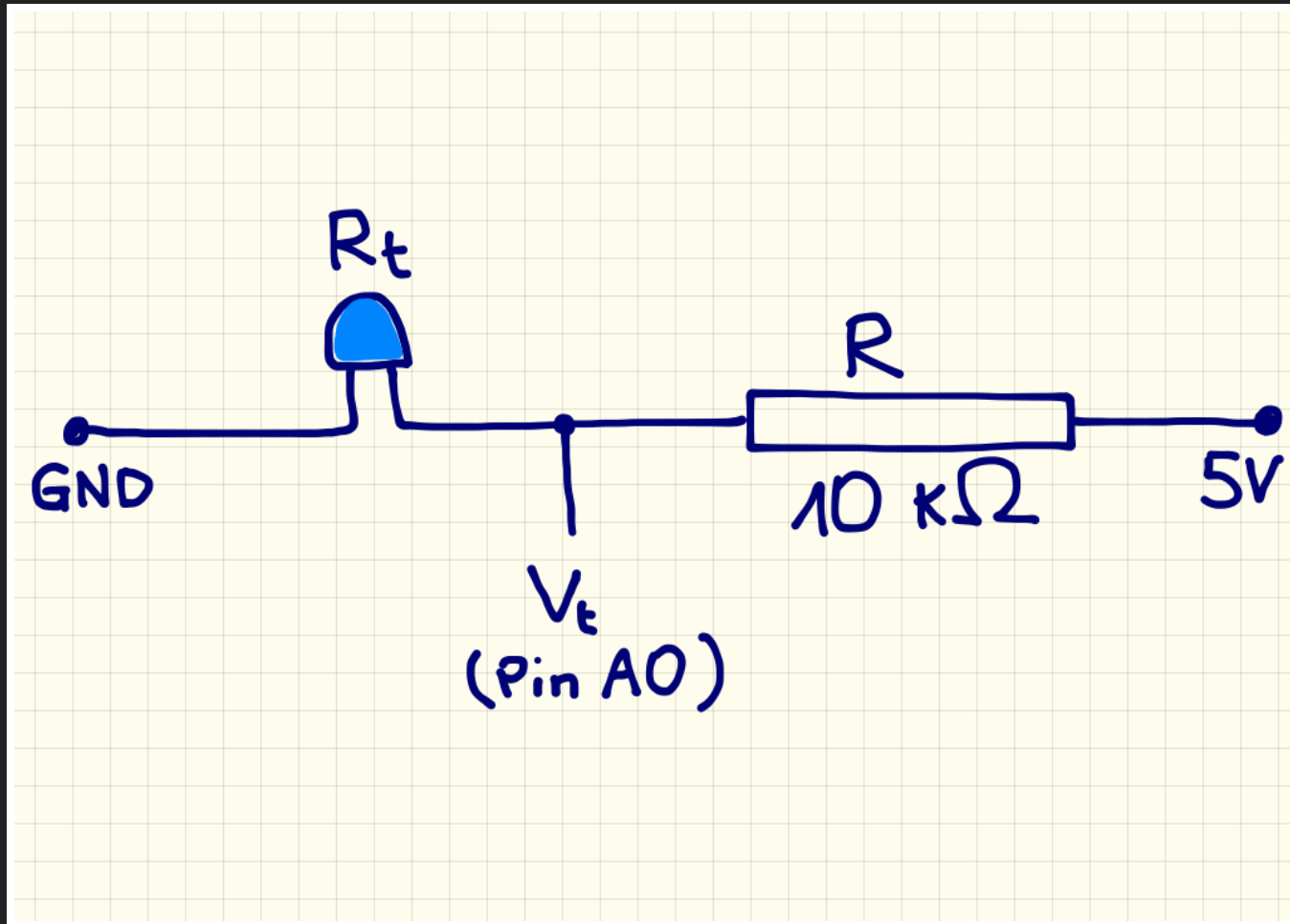
1. Arduino: Baue die Schaltung mit dem Temperatursensor auf.
2. Übersetzung Wert → Temperatur
3. Entwerfe die Tabelle für die Datenbank: Was muss gespeichert werden?
4. Erzeuge die gewünschte Tabelle mit sqlite3.
5. Processing: Methode SerialEvent, um die Werte einzulesen.
6. Processing: Verbindung mit einer sqlite-Datenbank, um Zeit und Messwert zu speichern.

ARDUINO



Das Program ist in Beispiele > 0.1 Basics > AnalogReadSerial. Ändere `delay(2)` zu `delay(1000)`, ein Wert pro Sekunde einlesen ist mehr als genug fürs Testen!

ÜBERSETZUNG EINGELESENEN WERT \rightarrow TEMPERATUR



Wie viel Physik und Mathe hier mit den Schülern besprechen?

ÜBERSETZUNG EINGELESENEN WERT → TEMPERATUR

TS-NTC-103-Widerstandstabelle.pdf: Widerstand des Sensors je nach Temperatur. Für das Ergebnis von `AnalogRead()` gilt:

- Der Wert 1023 entspricht 5V
(sehr niedrige Temperatur, sehr hoher Widerstand)
- Der Wert 0 entspricht 0V (sehr hohe Temperatur, sehr niedriger Widerstand)
- Der Vorwiderstand ist $R=10\text{ k}\Omega$
- $R_t/(R_t + R) = V_t/5V$ (wegen des Ohmschen Gesetzes)
- $(V_t/5V) \cdot 1023 = R_t/(R_t + R) \cdot 1023$ ist der eingelesene Wert.
- Zuordnung Temperatur-Wert in einem Rechenblatt anhand der Tabelle des Erstellers.

ÜBERSETZUNG EINGELESENEN WERT → TEMPERATUR

- Die Umsetzung Wert-Temperatur im Processing kann beliebig je nach Geschmack und Kenntnissen der Schüler gemacht werden: if-else-if, map, lineare Approximation, noch genauere Approximation,...
- Im folgenden Code habe ich einfach eine lineare Approximation mit der Formel $Temp = (737 - Wert)/9$ benutzt. Sie ist für 25 Grad exakt und für das Messen der Zimmertemperatur genau genug.

ENTWURF EINER TABELLE FÜR DIE TEMPERATUREN

Name der Tabelle: t

Attribute: Datentyp

Temperatur: NUM

Jahr: INT

Monat: INT

Tag: INT

Stunde: INT

Min: INT

Sek: INT

IN SQLITE3

```
.open wetter.db
```

```
CREATE TABLE t ( Temperatur NUM, Jahr INT,  
  Monat INT, Tag INT, Stunde INT, Min INT,  
  Sek INT );
```

Einen Unterordner data im aktuellen Sketch-Ordner erzeugen und die gerade erzeugte Datei wetter.db dort speichern

PROCESSING: ZEILE IN DIE TABELLE EINFÜGEN

```
import processing.serial.*;
import de.bezier.data.sql.*;

Serial myPort;
SQLite db;

void setup() {
  db = new SQLite( this, "wetter.db" ); // open database file
  db.connect();

  //println(Serial.list());
  // Open whatever port is the one you're using.
  myPort = new Serial(this, Serial.list()[3], 9600);
  // skip to first newline character:
  myPort.bufferUntil('\n');
}
```

Teste das Programm!

Für andere Werte (Feuchtigkeit, Luftdruck) müssen die entsprechenden Sensoren in Arduino eingebaut, und die ausgelesenen Werte mit `Serial.printl(...)` ausgegeben werden. In Processing muss dann die übertragene Zeichenkette entsprechend zerlegt werden. Die Erweiterung der Tabelle und des Update-Befehls ist unkompliziert.

DATENAUSWERTUNG

Beispiele für SQL-Abfragen:

```
select Jahr, Monat, Tag from t where Temperatur > 35;
```

```
select Jahr, Monat from t group by Jahr, Monat  
having avg(Temperatur) > 20;
```

```
select max(Temperatur) from t;
```

```
select min(Temperatur), max(Temperatur), avg(Temperatur)  
from t where Jahr=2016 and Monat=6;
```

```
select max(Temperatur) - min(Temperatur), Jahr, Monat from t  
group by Jahr, Monat order by Jahr, Monat;
```

```
select avg(Temperatur), Jahr, Monat, Tag from t  
group by jahr, monat, tag order by 2, 3, 4;
```


SERVER UND CLIENT IN PROCESSING

Wenn wir Daten von mehreren Arduinos sammeln möchten, müssen diese von verschiedenen Clients an einen Server geschickt werden. Das ist auch in Processing möglich.

HTTP GET-REQUEST

```
GET /temperature/25.0 HTTP/1.1  
Host: localhost  
Leereile: \r\n\r\n
```

Der Client schickt einen GET-Request, z.B. mit der URL
`http://host:port/temperature/<temp_value>`

HTTP RESPONSE

```
HTTP/1.1 200 OK  
Content-Type: text/html  
  
<h1>Received</h1>
```

Der Server antwortet mit ein wenig HTML. Das ist nicht für das Programm nötig sondern hilft beim Testen im Browser:

Schicke einen GET-Request mit URL

`http://localhost:8888/temperature/45` und betrachte die Antwort im Browser.

EIN SERVER IN PROCESSING

```
// Ich bin ein Server.  
// Ich schreibe in eine sqlite-DB die Temperaturen, die mir ein Clier  
import de.bezier.data.sql.*;  
import processing.net.*;  
  
Server server;  
SQLite db;  
  
void setup() {  
    server = new Server(this, 8888); // this is a generic tcp-sever  
  
    db = new SQLite( this, "wetter.db" ); // open database file  
    db.connect();  
}  
  
void draw () {
```

EIN CLIENT IN PROCESSING, DER WERTE AUS ARDUINO ZUM SERVER SCHICKT

```
// Ich bin ein Client, der Daten von Arduino empfängt
// und weiter an den Server schickt.
import processing.serial.*;
import processing.net.Client;

Serial myPort;

void setup () {
    // Serial
    //println(Serial.list());
    // Open whatever port is the one you're using.
    myPort = new Serial(this, Serial.list()[1], 9600);

    // skip to first newline character
    myPort.bufferUntil('\n');
}
```

EXTRAS: FEUCHTIGKEITSSENSOR

GROVE TEMP AND HUMIDITY SENSOR

Website [SEED STUDIO Grove Temperature and Humidity Sensor](#)

Kopiere den Code für Arduino in einen neuen Sketch (der Pin muss nicht unbedingt A0 sein, ein Digitalpin funktioniert genauso gut, aber der Code muss entsprechend geändert werden)

Folge dem Link [Library](#) für die entsprechende Bibliothek in Github und lade sie herunter (Von Safari mit gedrückter ALT-Taste, um das Unzippen zu vermeiden):

The screenshot shows the GitHub interface for the repository 'Seeed-Studio / Grove_Temperature_And_Humidity_Sensor'. At the top, there are navigation links: Personal, Open source, Business, Explore, Pricing, Blog, and Support. A search bar and 'Sign in'/'Sign up' buttons are also present. The repository name is displayed with icons for Watch (6), Star (22), and Fork (35). Below this, there are tabs for Code, Issues (2), Pull requests (0), Pulse, and Graphs. The description states: 'Arduino library for the DHT series temperature&humidity sensors' with a link to the Seeed Studio depot. A progress bar shows 8 commits, 1 branch, 0 releases, and 7 contributors. A 'Clone or download' dropdown menu is open, showing options to 'Clone with HTTPS' (with a URL), 'Use SSH', 'Open in Desktop', and 'Download ZIP'. The file list includes 'examples/DHTtester', 'DHT.cpp', 'DHT.h', and 'README.md', all attributed to 'restructure'.

Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

Seeed-Studio / Grove_Temperature_And_Humidity_Sensor Watch 6 Star 22 Fork 35

Code Issues 2 Pull requests 0 Pulse Graphs

Arduino library for the DHT series temperature&humidity sensors <http://www.seeedstudio.com/depot/grove-temperaturehumidity-sensor-pro-p-838.html>

8 commits 1 branch 0 releases 7 contributors

Branch: master New pull request Find file Clone or download

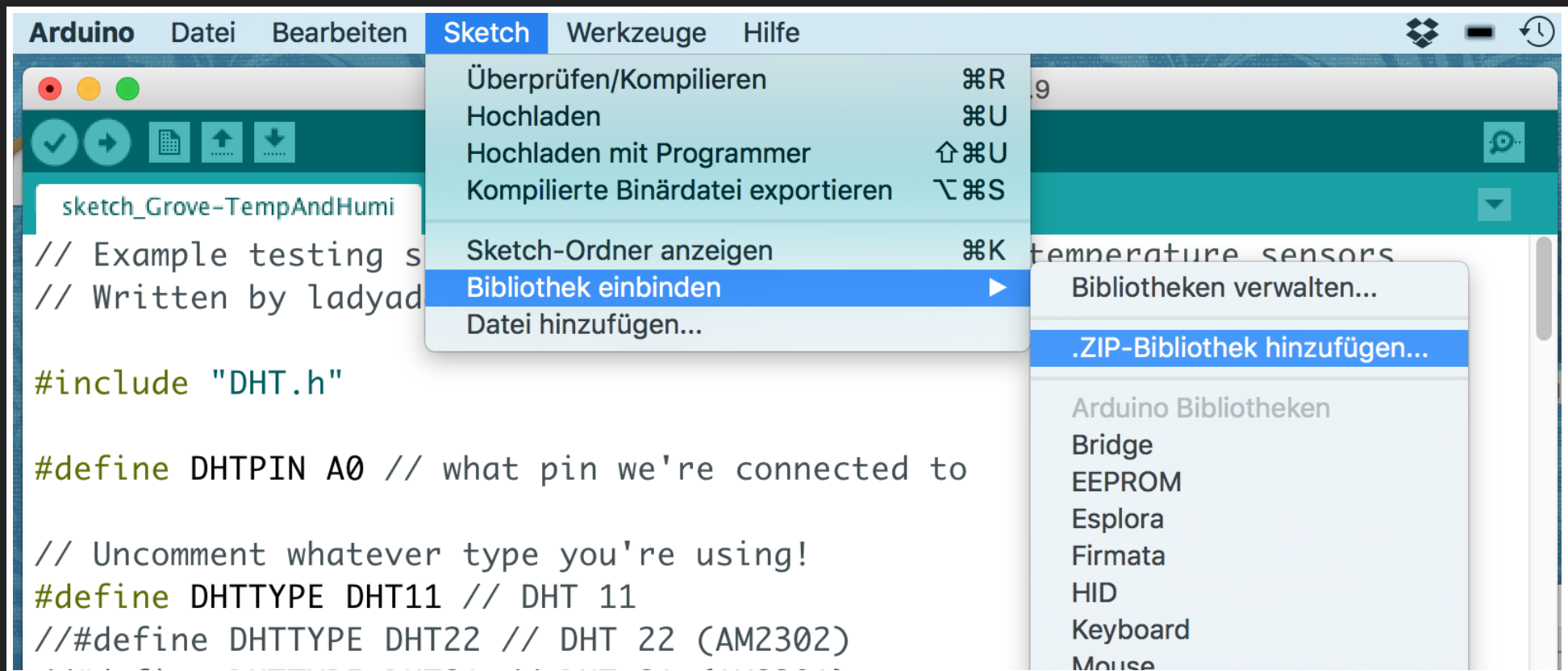
Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/Seeed-Studio/Grove_
Open in Desktop Download ZIP

xiongyihui restructure

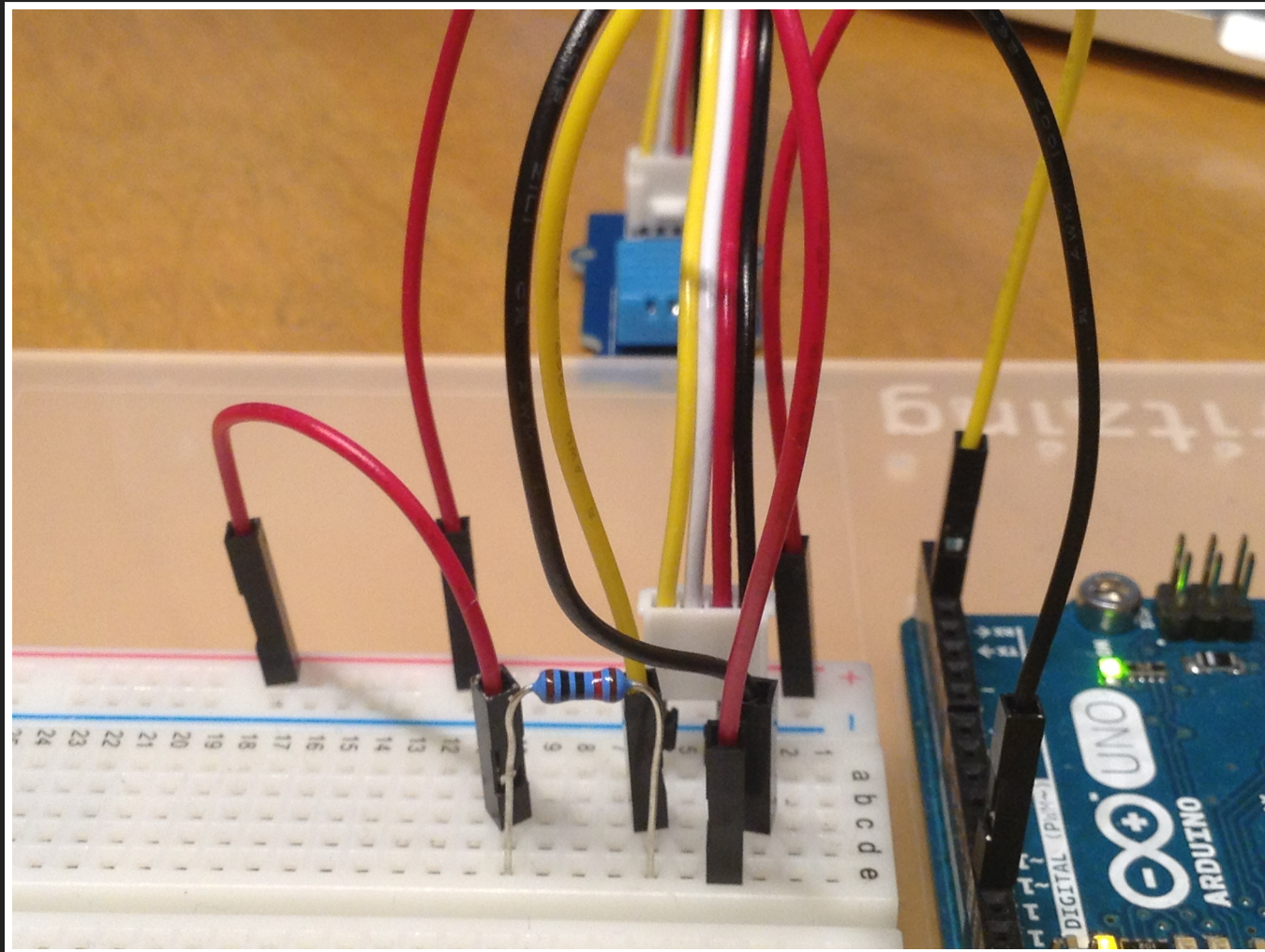
examples/DHTtester	restructure
DHT.cpp	restructure
DHT.h	restructure
README.md	restructure

a year ago

Binde die heruntergeladene Bibliothek in den Sketch ein mit dem Menü Sketch > Bibliothek einbinden:



Schaltkreis mit 10 kOhm Vorwiederstand



WEITERE IDEEN

- Ventilator
- Twitter-Anbindung
- Connect to OpenWeatherMap:
<http://openweathermap.org/stations>
- Raspberry Pi Weather Station:
<http://www.raspberrypiweather.com>
- Raspberry Pi Weather Station for Schools:
<https://www.raspberrypi.org/education/weather-station/>

DANKE!

