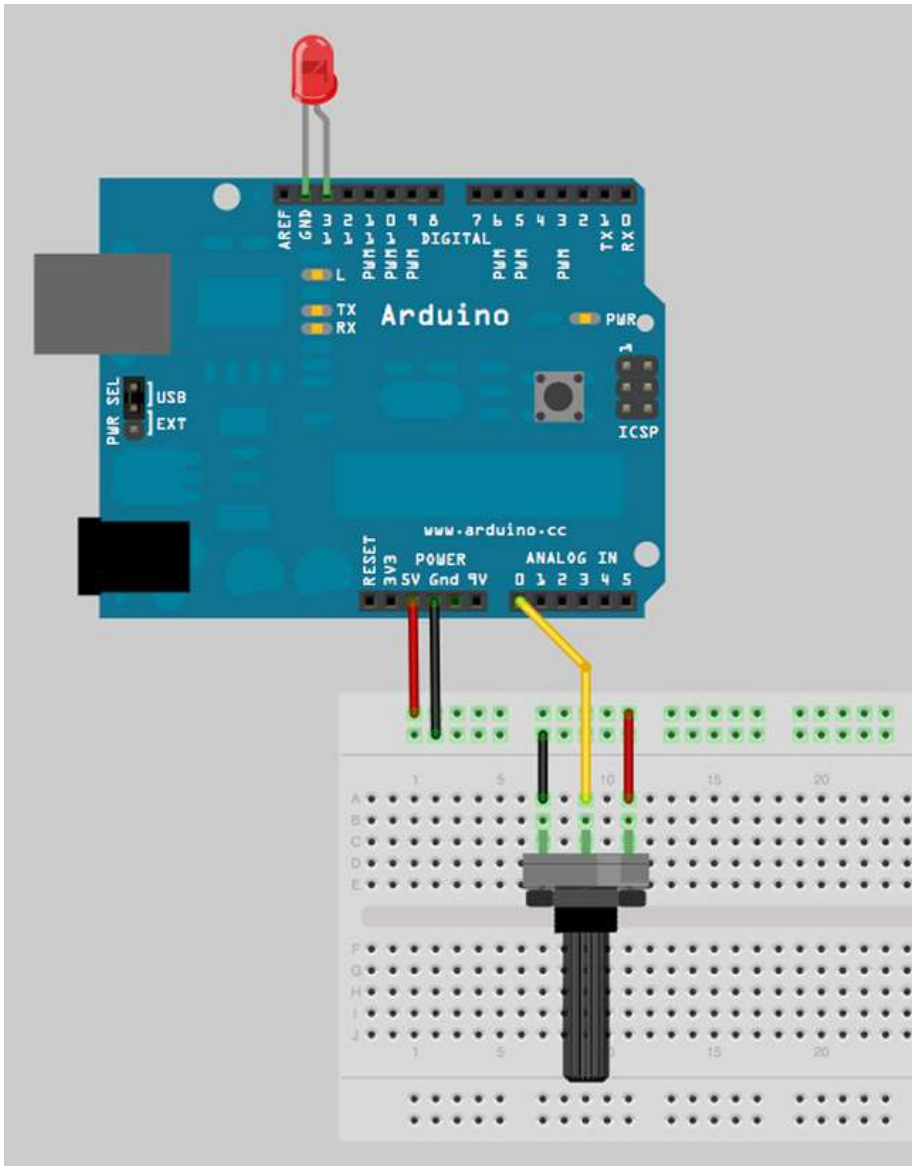
Analoge Eingänge



Quelle: http://www.arduino-tutorial.de/2010/06/analog-in/

Im Gegensatz zu digitalen Signalen, die entweder HIGH oder LOW sind, liefern analoge Sensoren auch Zwischenwerte.

Analoge Sensoren sind z.B. druckempfindliche Widerstände (FSR), Lichtsensoren oder auch Temperaturfühler.

Im Beispiel ist ein Potentiometer ans Arduino Board angeschlossen.

Die beiden äußeren Beine werden mit dem GND und dem 5V+ verbunden, das mittlere mit einem Analog Input. Das Arduino-Board kann nun das Verhältnis der Widerstände zu einander ermitteln und liefert durch den Befehl analogRead(Pin); Werte zwischen 0 und 1023.

```
int sensorPin = 0;
int ledPin = 13;
int sensorValue = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {  // Blink - Takt nach Analog Wert
  sensorValue = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);
  delay(sensorValue);
  digitalWrite(ledPin, LOW);
  delay(sensorValue);
}
```

Analoge Eingänge

Der **ATmega8** verfügt an Port C über 8 analoge Eingänge, die zur Messung von Spannungen verwendet werden können. Unter Arduino können diese Pins mit **A0 .. A7** angesprochen werden.

AVCC

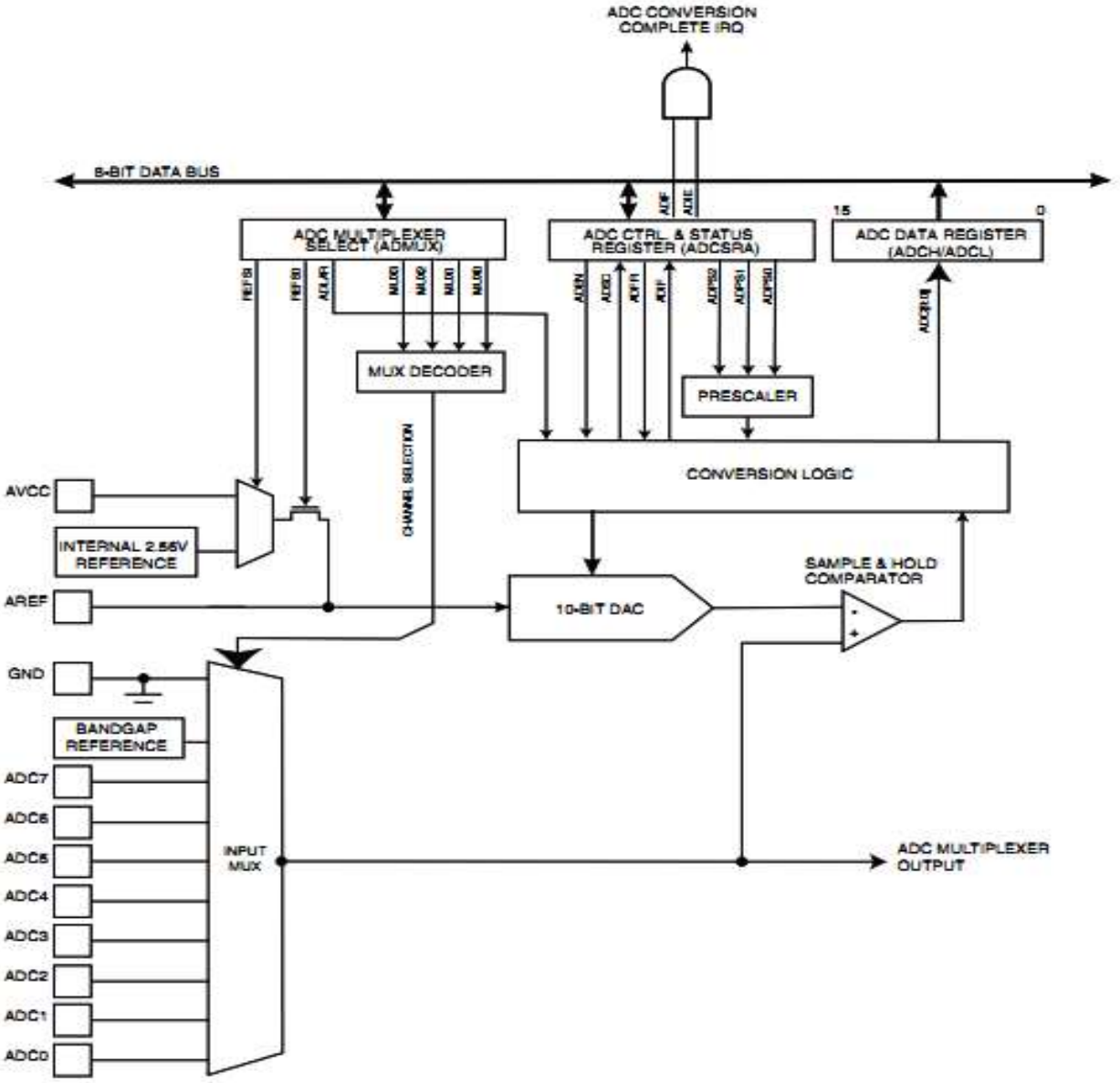
AREF

AGND

A0

…

A7



Bei der Messung wird die **Eingangsspannung** mit einer **Referenzspannung** verglichen und ausgewertet. Standardmässig wird hierzu die Versorgungsspannung an **AVCC** (Pin 20) angeschlossen. Durch ein **Tiefpassfilter** bestehend aus einer Induktivität von 10 uH oder einem Widerstand von 47 k und einem Kondensator von 100 nF werden Störimpulse unterdrückt. Intern wird die Spannung an VCC auf **AREF** geschaltet. AREF wird ebenfalls durch einen Kondensator von 100 nF gefiltert.

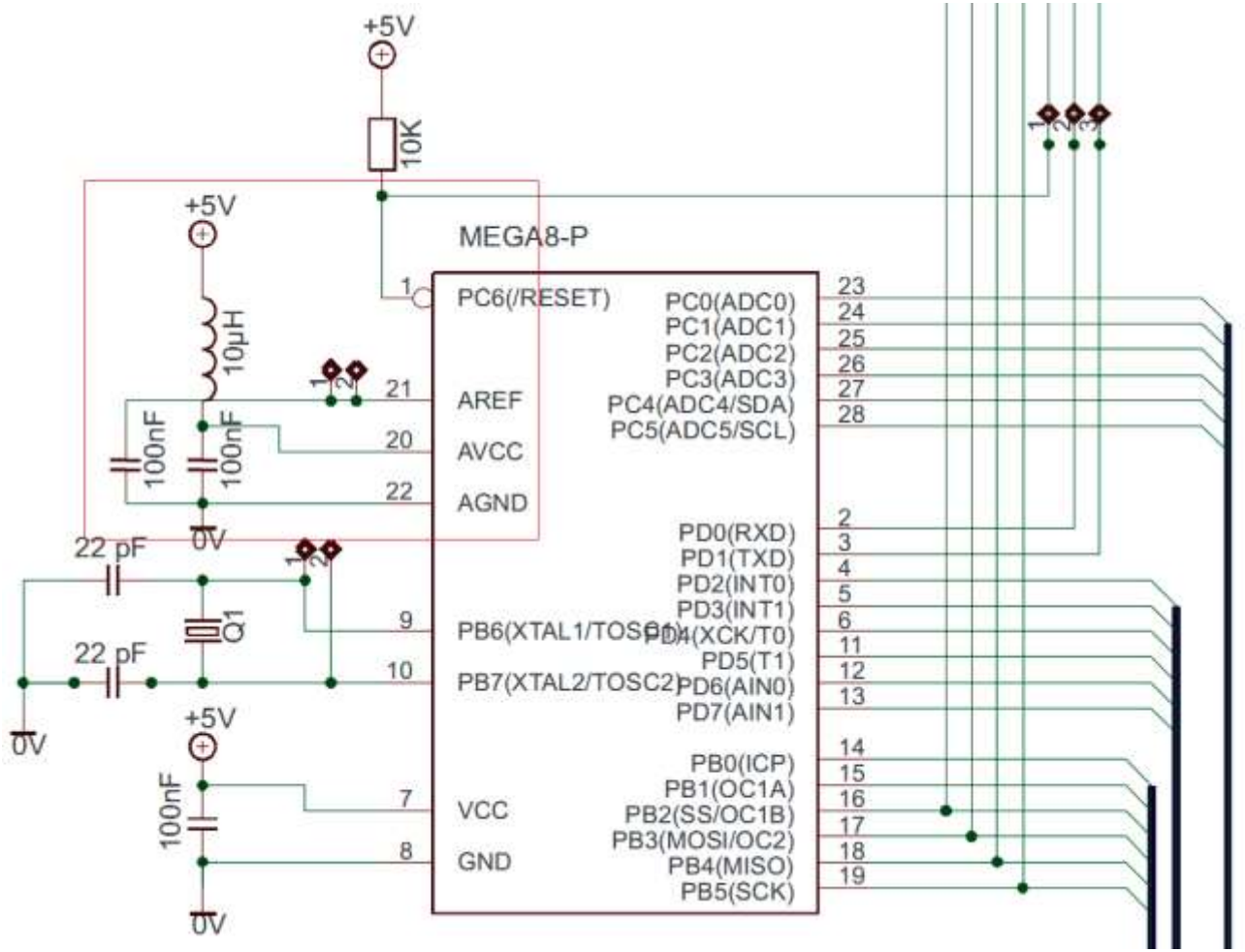
Es könnte auch eine andere Referenzspannung von <= 5 V an AREF angeschlossen werden.

Darüber hinaus kann per Software eine i**nterne Referenzspannung** von 2.56 V (1.1 V beim ATMega 168 und 328) zugeschaltet werden.

Dies erfolgt durch den Arduinobefehl

analogReference(INTERNAL);

Ohne diesen Befehl wird standardmässig die Versorgungsspannung an AVCC verwendet.



analogReference()

Es können optional folgende Referenzspannungen eingestellt werden:

Standart: 5 V (bei 5V Arduino Boards) oder 3,3 V (on 3,3V Arduino Boards)
Dies ist die Grundeinstellung und muss nicht angegeben werden.

INTERNAL: interne 1,1 V Referenz bei ATmega168 / ATmega328 und
2,56 volts bei ATmega8 (nicht verfügbar bei *Arduino Mega*)

INTERNAL1V1: 1,1V reference (nur *Arduino Mega*)

INTERNAL2V56: 2,56V reference (nur *Arduino Mega)*

EXTERNAL: externe Spannung 0 .. 5 V am AREF Pin

Syntax: analogReference(type)

Auch auf die Analog - Pins können Pullup Widerstände aufgeschaltet werden:

digitalWrite(A0, HIGH); // Setze Pullup bei Analog Pin 0, wenn Pin ist INPUT

Der ATMega8 hat verfügt über **10 Bit Analogwandler**. Dies bedeutet, dass eine Eingangspannung zwischen 0 und der Referenzspannung (meist 5 V) auf **0 .. 1023** umgesetzt wird.

Damit man tatsächliche Spannungswerte erhält, muss das Messergebnis mit einem Faktor von **5 / 1024 = 0,004883** multipliziert werden.

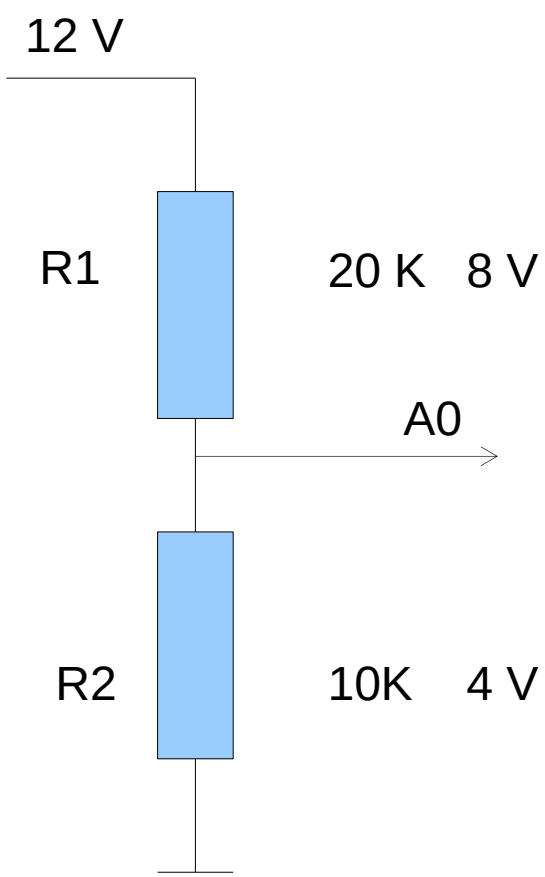
Achtung: Je nach Beschaltung des Spannungsreglers 7805 kann die Versorgungs-spannung zwischen 4.8 und 5.2 schwanken. Auf diese Spannung wird dann allerdings
sehr genau ausgeregelt.

Den Faktor müsste man deshalb bei einer Versorgungsspannung von z.B. 4.8 V korregieren auf 4.8 / 1024 = 0,004688.

Beachten sollte man auch, dass eine Multiplikation weniger Rechenzeit benötigt als eine Division.

0 .. 5 V = 0 .. 1023

Spannungsteiler



Häufig sollen größere Spannungen als 5 V gemessen werden. Hierzu könnte man einen Spannungsteiler benutzen. Für z.B. 12 V könnte man einen Spannungsteiler von 1 : 3 verwenden:

I = U / R = 12 / 30000 = 0,0004 A

U1 = 20000 x 0,0004 A = 8 V

U2 = 10000 x 0,0004 A = 4 V

Faktor = 4 / 12 = 0.3333

Analog = Messwert x (5 /1024) / 0.3333

Mit folgendem Befehl können unter Arduino Analogeingänge gelesen werden:

```
float Messung;
Messung = analogRead(A0);
```

Das Ergebnis wird hier in der Variable Ergebnis gespeichert.

Diese wurde als float (Fließkomma – Zahl) deklariert um
später mit Nachkommerstellen rechnen zu können.

Statt A0 kann man auch A1 .. A7 einsetzen der den Pin über eine Variable übergeben, z.B.

```
int sensorPin = A0;     // Eingang A0 = Port C0
```

Beispiel Messwerte über LCD und Serielle Schnittstelle ausgeben

```
#include <LiquidCrystal.h>                    // Binde LCD Bibliothek ein
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);          // Pins für LCD

int sensorPin = A0;                           // Eingang A0 = Port C0
float messung;                                // float Variable für Ergebnis

void setup()
{
  Serial.begin(115200);                       // Baudrate einrichten
  lcd.begin(16, 2);                           // LCD 2 x 16 Zeichen
}

void loop()
{
  messung = analogRead(sensorPin);            // lese Analogeingang A0
  messung = messung * 5 / 1023;               // Berechne Spannung

  lcd.clear();                                // Ausgabe auf LCD
  lcd.print(Ergebnis);
  lcd.print(" Volt");

  delay(1000);                                // Warte 1 Sekunde
  Serial.println(Ergebnis);                   // Messwerte über Serial
ausgeben
}
```

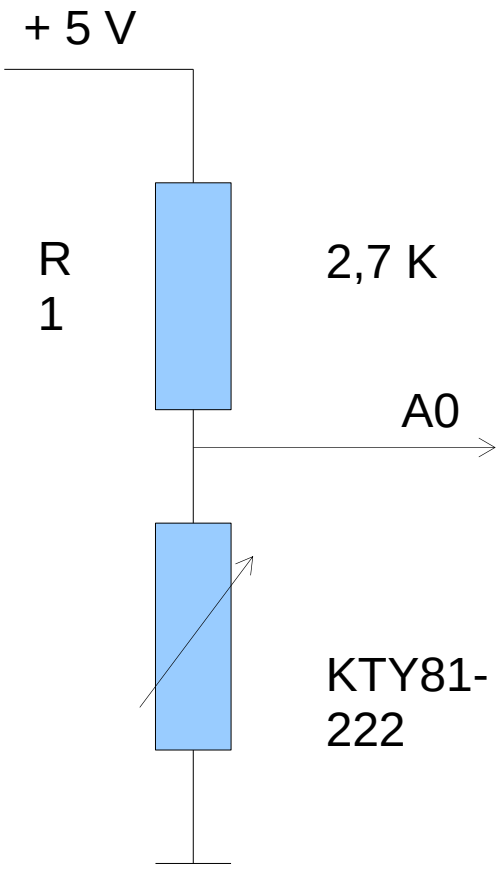
Temperaturmessung mit KTY81 - 222

Der KTY ist ein preiswerter Temperaturfühler (Ebay 2,50 €).
Laut Datenblatt hat dieser folgende Kennwerte:

Grad C	**Widerstand in Ohm**		
	Min	Typ.	Max
0	1619	1646	1673
20	1920	1941	1963
25	2000	2020	2040
40	2239	2267	2295

Wenn man hiermit einen Spannungsteiler aufbaut,
lassen sich hiermit sehr einfach Temperaturen messen.

Man beachte, dass die Sensoren relativ hohe Streuungen
der Widerstände im Bereich + / - 20 Ohm haben.



```
//Temperaturmessung mit KTY81-222
// +5V  <Widerstand 2.7 K> <...> <KTY81> <GND>
// <PC0>

#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

int sensorPin = A0;     // Analog Pin Port C0

//Sonderzeichen für Grad
byte grad[8] = {
  B00000,
  B01110,
  B10001,
  B10001,
  B01110,
  B00000,
  B00000,
};


void setup() {
  lcd.begin(16, 2);
  lcd.createChar(0, grad); //Sonderzeichen °
  lcd.print("Temperatur");
  delay(2000);
}


void loop() {
  float sensorValue = analogRead(sensorPin);

  //4o°  C = 2267
  // 0°  C = 1646
  // Differenz = 621
  // 1 °  Differenz = 621 /40 = 15.525

  // Berechne KTY Widerstand
  float ukty = sensorValue * 5 /1024;
  float u2_7k = 5 - ukty;
  float ikty = u2_7k / 2700; //I = U / R
  float rkty =  ukty / ikty;

  // Addiere Korrekturfaktor Bauteile Tolleranz
  rkty = rkty + 18;

  // Berechne Temperatur
  float gradc;
  if (rkty == 2267)  //4o °  C
  {
    gradc = 40;
  }

  if (rkty < 2267)
  {
    float tmp = 2267 - rkty;
    tmp = tmp /15.525;  //1°  = 15.255 Diff
    gradc = 40 - tmp;
  }

  // Runden
  int int_gradc = (int) (gradc+0.5);

  lcd.clear();
  lcd.print(int_gradc);
  lcd.print(" ");

  lcd.write(0);  //Sonderzeichen °
  lcd.print("C Temperatur");

  lcd.setCursor(0, 1);
  int int_rkty = (int) (rkty+0.5);
  lcd.print(int_rkty);
  lcd.print(" Ohm");

   delay(1000);
}
```

Die Spannung am Sensor wird berechnet:

float ukty = sensorValue * 5 /1024;

Die Spannung am Festwiderstand von 2.7 K ist dann

float u2_7k = 5 – ukty;

Hieraus lässt sich der Strom durch den Spannungsteiler berechnen:

float ikty = u2_7k / 2700; **// I = U / R**

Mit dem Strom kann man den Widerstand des Sensors berechnen:

float rkty = ukty / ikty; **// R = U / I**

Zusätzlich kann man einen empirischen Wert für die Bautteil-tolleranzen hinzufügen:

rkty = rkty + 18;

Aus dem Datenblatt kann die Widerstandsänderung je Grad C errechnen:

40° C = 2267 Ohm
0° C = 1646 Ohm

Differenz = 621Ohm

1 ° Differenz = 621 /40 = 15.525

Wenn man weiß, dass 40° C - 2267 Ohm entspricht, kann man aus dem gemessenen Widerstand die Temperatur berechnen:

```
if (rkty < 2267)
  {
    float tmp = 2267 - rkty;
    tmp = tmp /15.525;                    // 1°  = 15.255 Diff
    gradc = 40 – tmp;
}
```

Zur Ausgabe von Messwerten über die serielle Schnittstelle benötigt man nur wenige Befehle. Die Pins sind mit **RX** und **TX** gekennzeichnet.

Die Schnittstelle arbeitet mit 5 V TTL Pegel - PC Schnittstellen hingegen mit +/- 12 V.

Schnittstelle initialisieren:

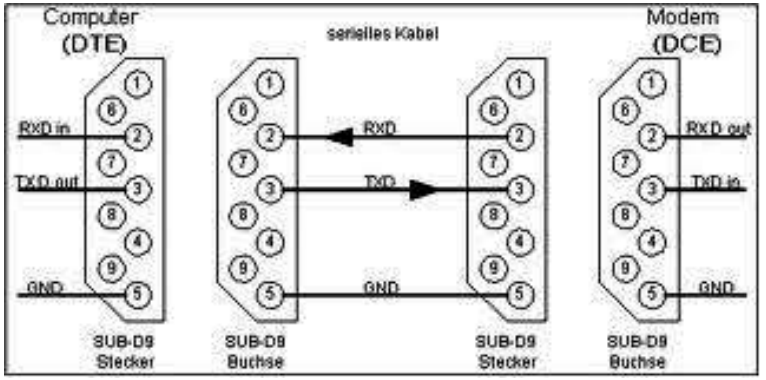
```
void setup()
{
  ...
Serial.begin(115200);                 // Statt 115200 kann man auch Werte
}                                     // wie z.B. 9600, 14400, 19200 eingeben
```

Die Ausgabe erfolgt mit

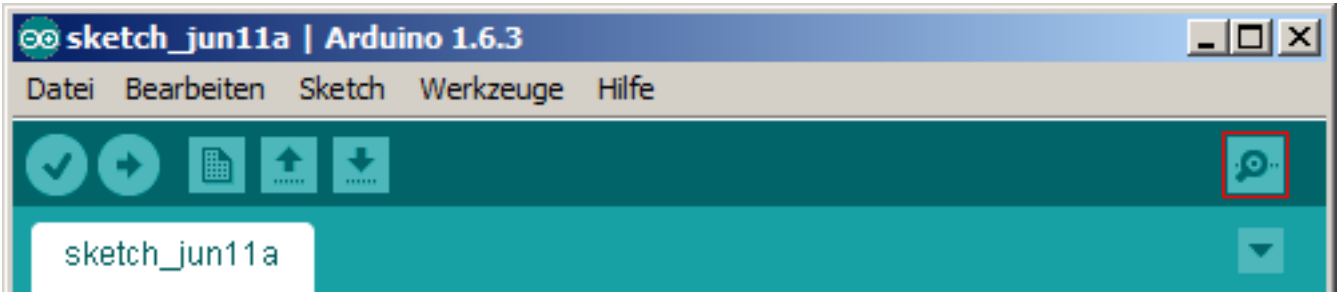
```
Serial.println(Ergebnis);             // Mit anschließendes Return
Serial.print(Ergebnis);               // Ohne abschließendes Return
```

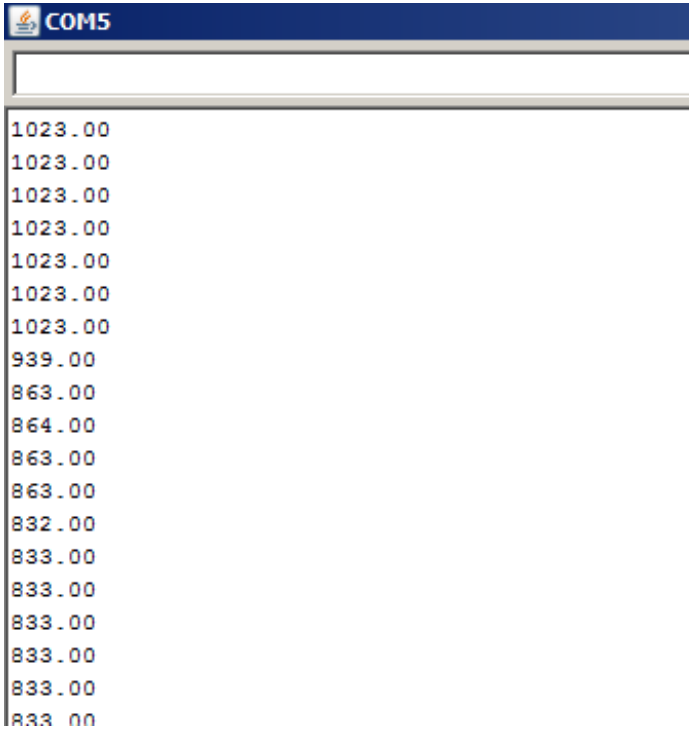
Weitere Befehle für die serielle Schnittstelle sind:

Serial.write(12);	**Sende als Binärwert 12**
Serial.available();	**Anzahl der Bytes im Eingangspuffer**
Serial.read();	**Lese Byte aus Eingangsbuffer;**
serial.end():	**Beende serielle Kommunikation**

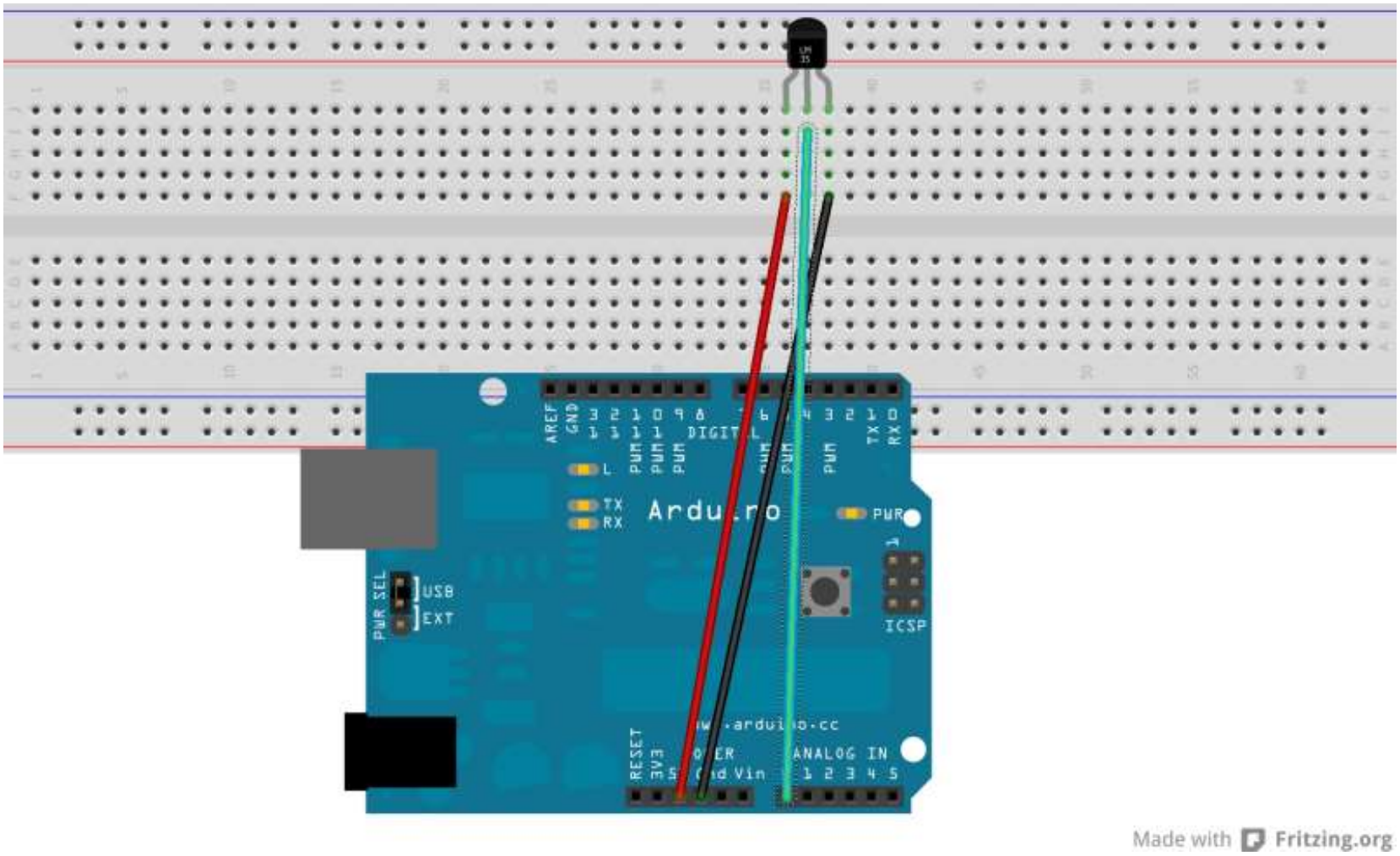


Mit dem rechten Icon der Arduino IDE kann man ein Terminal öffnen. Hier werden die empfangenen Daten angezeigt. Zuvor muss man rechts unten die Baudrate einstellen.





Temperaturmessung mit LM35



Der LM35 ist ein Temperatursensor mit höherer Genauigkeit.
Achtung nicht Plus und Minus verwechseln!

Temperaturmessung mit LM35

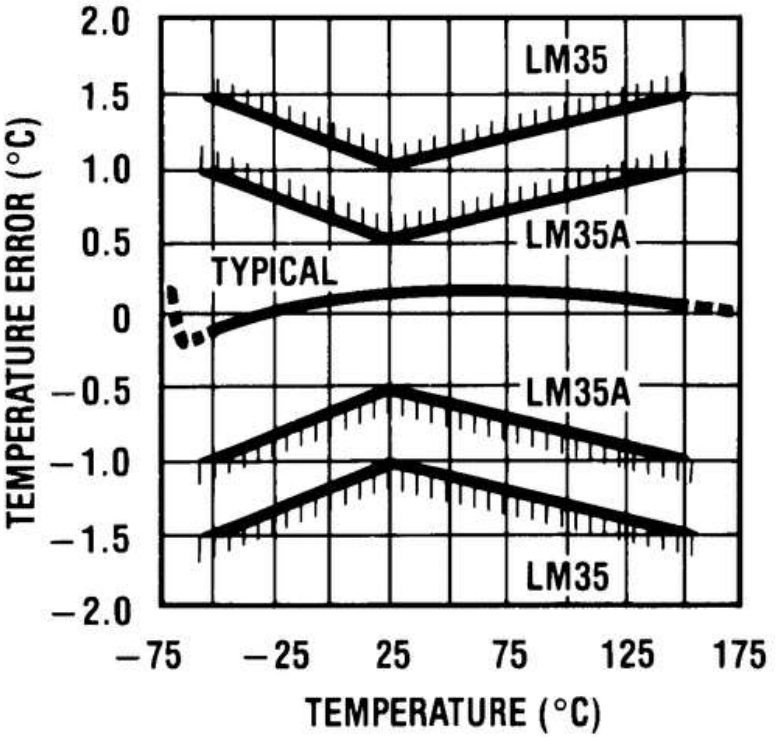
```
int LM35 = A0;
float SensorValue = 0;
float temperatur = 0;
float temp[5];
void setup() {
Serial.begin(9600);
}

void loop() {
SensorValue = analogRead(LM35);
temp[1] = (5.0 * analogRead(LM35) * 100.0) / 1024;
delay(1000);                                  // 1 Sekunde Pause zwischen den Messungen machen
temp[2] = (5.0 * analogRead(LM35) * 100.0) / 1024;
delay(1000);
temp[3] = (5.0 * analogRead(LM35) * 100.0) / 1024;
delay(1000);
temp[4] = (5.0 * analogRead(LM35) * 100.0) / 1024;
delay(1000);
temp[5] = (5.0 * analogRead(LM35) * 100.0) / 1024;
temperatur = (temp[1] + temp[2] + temp[3] + temp[4] + temp[5])/5;       // Mittelwert aus 5 Messungen bilden
//Serial.print(SensorValue, DEC);
//Serial.print(„ –> „);
Serial.print(temperatur, 1);                  // Ausgabe der Temperatur mit einer Nachkommastelle
Serial.println(„Grad Celsius“);
}
```

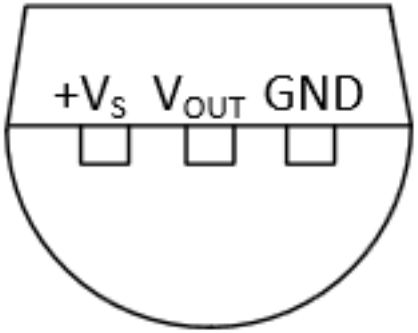
Quelle: http://www.gunnarherrmann.de/blog/temperatur-auslesen-arduino-lm35/

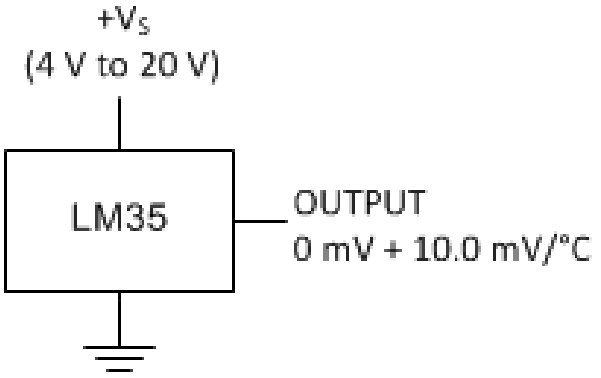
LM35 Daten

- Calibrated Directly in Celsius
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C
- RangeSuitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-µA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load
- Ebay 2,40 €

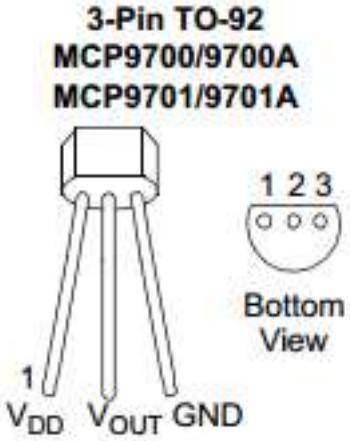


LP Package
3-Pin TO-92
(Bottom View)

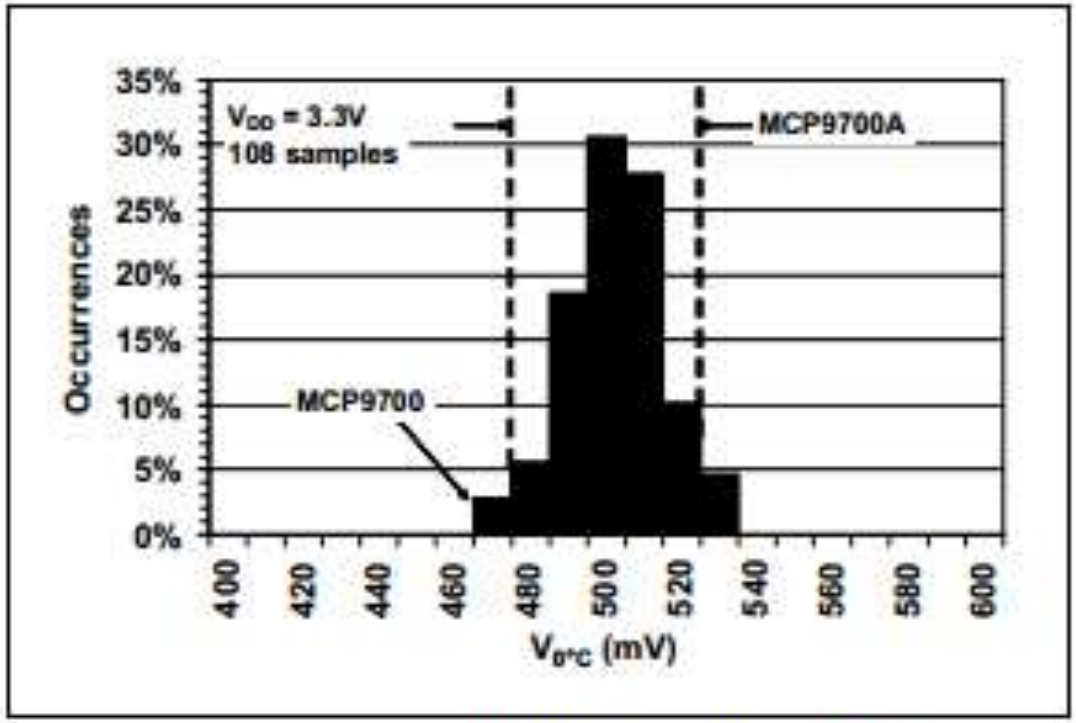




Temperatur-Messung mit dem MCP9700



Temperaturbereich	-40° C … 120° C
Hilfsspannung	2.2 .. 5,5 V DC
Genauigkeit	+ /- 2° C
Ausgang 0° C	ca. 500 mV
Spanne	10 mV / ° C



Output Voltage at 0°C

Beispiel für Messroutine mit dem MCP9700

```
// Routine um Temperatur zu bekommen
float getTemperatur()
{
float  Nullpunkt = 0.5;                    // Sensor liefert bei 0° C ca. 500 mV
float  Abgleich = 0.05;                    // Korrektur für Feinabgleich
int temperatur = analogRead(sensorPin);
   // Wandele Analogwert in Spannung
 temperatur = 5.0 / 1024.0 *  temperatur;
 // der Sensor liefert bei 0 Grad ca. 500 mV
 // Variable Nullpunkt = 0.5
 // der Sensor liefert pro °C 10 mV
 // die Spannungsdifferenz zum Nullpunkt ist die Temperaturspanne
 // der Sensor hat Fertigungstolleranzen von +/- 1 ° C
 // verändere den Nullwert etwas zum Kalbrieren mit Abgleich"
 Nullpunkt = 0.5 + Abgleich;

 temperatur = temperatur - Nullpunkt;

 // mit 100 multiplizieren für Gradzahlen
 temperatur = temperatur * 100.0;

 // Runden auf volle Gradzahl
 temperatur = round(temperatur);

// Sende Temperatur über Schnittstelle
 Serial.println(temperatur);
}
```

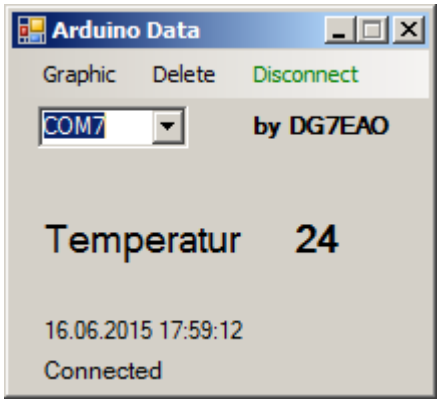
Komplettes Beispiel: Temperatur_9700.ino

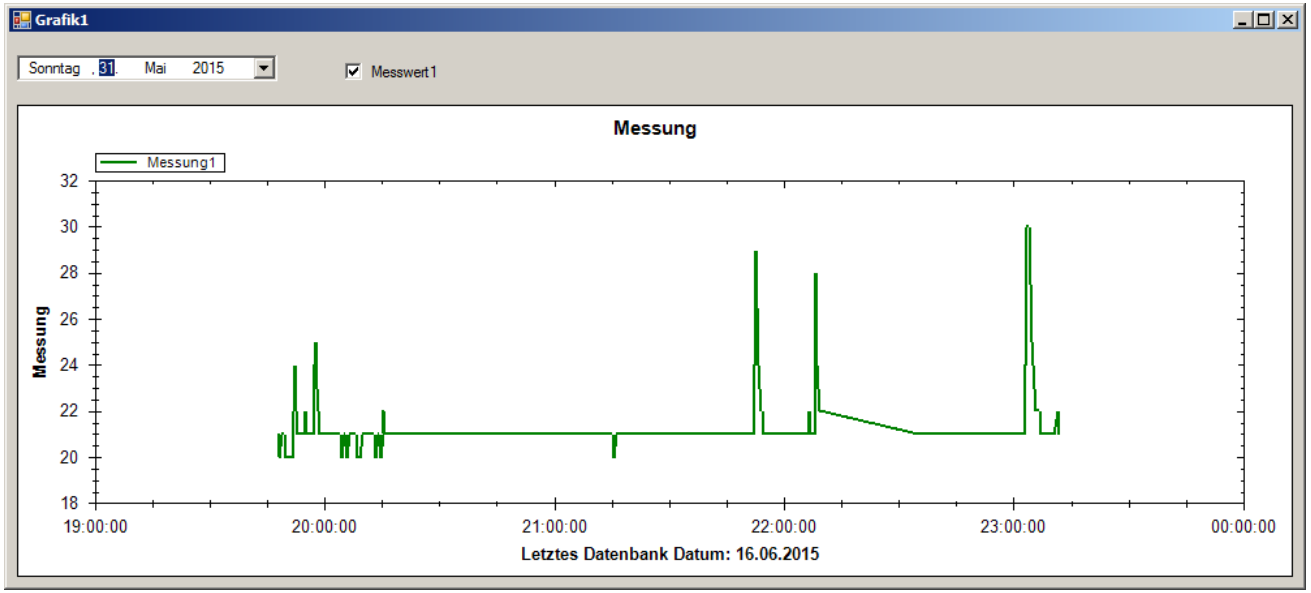
https://drive.google.com/file/d/0Bwp8f5BCaeDLZ1hYUkRnblB5aWs/view?usp=sharing

Komplettes Beispiel: Temperatur_9700.ino

https://drive.google.com/file/d/0Bwp8f5BCaeDLZ1hYUkRnblB5aWs/view?usp=sharing

- Gibt Temperatur über Board LED aus:
  - blinkt 5x schnell, dann langsames Blinken für 1. Stelle, z.B. 2 x für 2 von 23
  - blinkt 10 x schnell, dann langsames Blinken für 2. Stelle, z.B. 3 x für 3 von 23
  - dann längere Pause
- Ausgang Temperatur - Sensor an A0
- Grenzwertschalter an PIN12 mit Hysterese (Wertvorgabe in Sourcecode)
- Übergibt die Temperatur über Serielle Schnittstelle an Windows - Programm, dieses speichert Messwerte in Datenbank und zeichnet Verlaufskurve, braucht Dot.Net Framework => 4.0





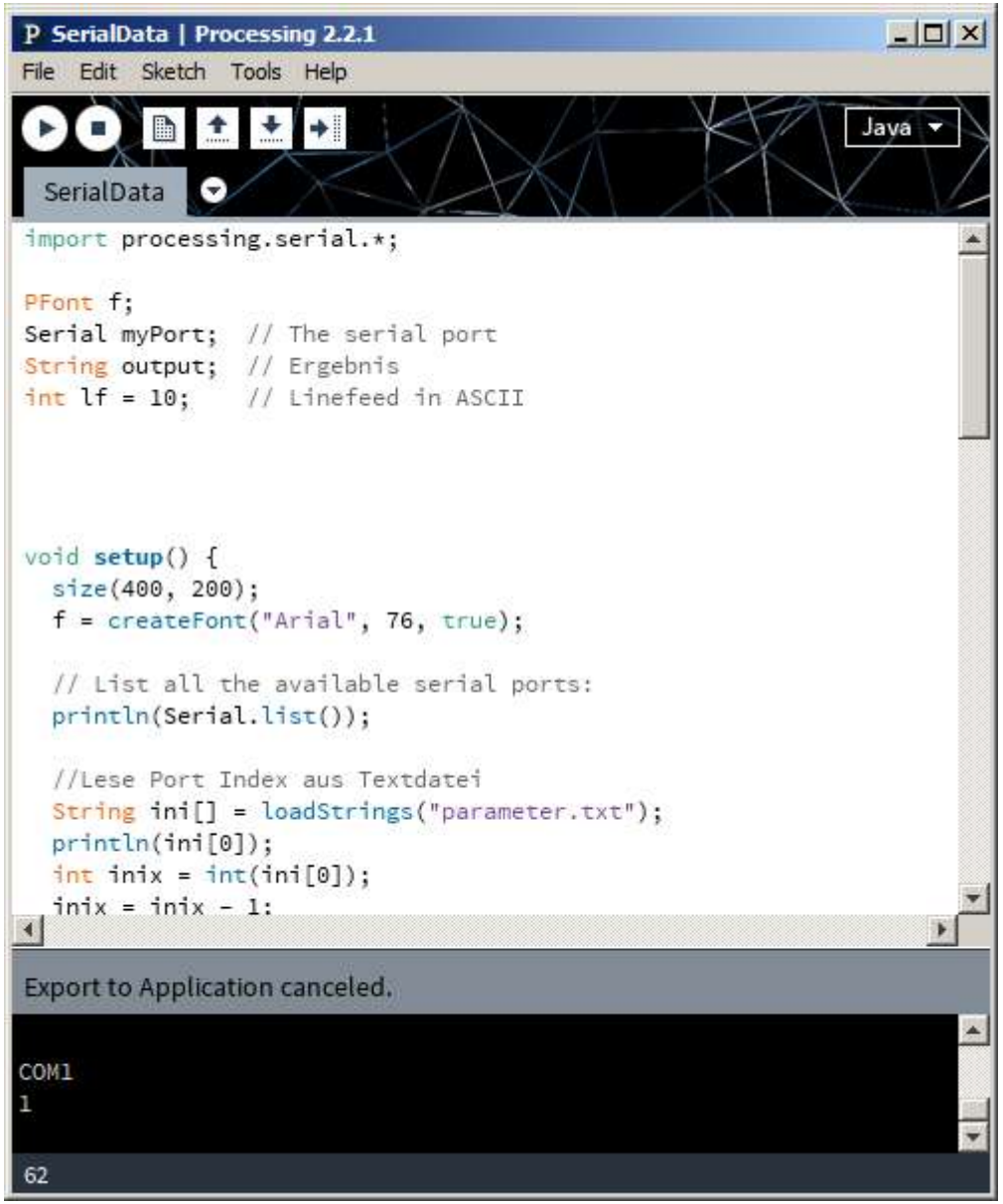
Windows Programm Arduino Data:

https://drive.google.com/file/d/0Bwp8f5BCaeDLT1BWZHBuWXVJUlE/view?usp=sharing

Es gibt unter dem Namen **Processing** eine Programmierumgebung mit der man mit einer ähnlichen Syntax wie bei Arduino, eigene Windows Programme erstellen kann. Mit den Beispieldateien findet man sich schnell zurecht. Hiermit kann man z.B. kleine Applets programmieren um serielle Daten auf dem PC anzuzeigen.

http://processing.org/

Die Processing IDE hat einen ähnlichen Aufbau und Befehlssatz wie Arduino



SerialMonitor.pde - Processing Code für einen einfachen seriellen - Monitor

```
import processing.serial.*;

PFont f;
Serial myPort;                  // The serial port
String output;                  // Ergebnis
int lf = 10;                    // Linefeed in ASCII

void setup() {
  size(400, 200);
  f = createFont("Arial", 76, true);

  // List all the available serial ports:
  println(Serial.list());

  // Lese Port Index aus Textdatei
  String ini[] = loadStrings("parameter.txt");
  println(ini[0]);
  int inix = int(ini[0]);
  inix = inix - 1;

   myPort = new Serial(this, Serial.list()[inix], 115200);
}

void draw() {
  background(155);

  stroke(175);
  //line(width/2,0,width/2,height);

  textFont(f);
  fill(0);

  textAlign(CENTER);
  //text("123",width/2,140);
  //textAlign(LEFT);
  //text("This text is left aligned.",width/2,100);
  //textAlign(RIGHT);
  //text("This text is right aligned.",width/2,140);

  while (myPort.available () > 0) {
    output = myPort.readStringUntil(lf);
    // if (myString != null) {
    // println(myString);
    // }
  }


  // Ergebnis ausgeben
  output = "" + output;

  // Ausgabe wenn String > 2 Zeichen
  if (output.length() >2)
  {
    text(output, width/2, 80);
    delay(1000);
  }
}
```

Im Sketch Verzeichnis eine Datei **parameter.txt** erstellen.

Hier z.B. die Zahl 3 hinein schreiben, wenn die 3. Com Schnittstelle im System für die serielle Kommunikation benutzt wird.

SerialMonitor.pde - Sourcecode

Processing Sourcecode SerialMonitor:

https://drive.google.com/file/d/0Bwp8f5BCaeDLRGdNN1RFNXd5ZU0/view?usp=sharing

Serial Win Application mit Embedded Java Runtime (ca. 34 MB !)

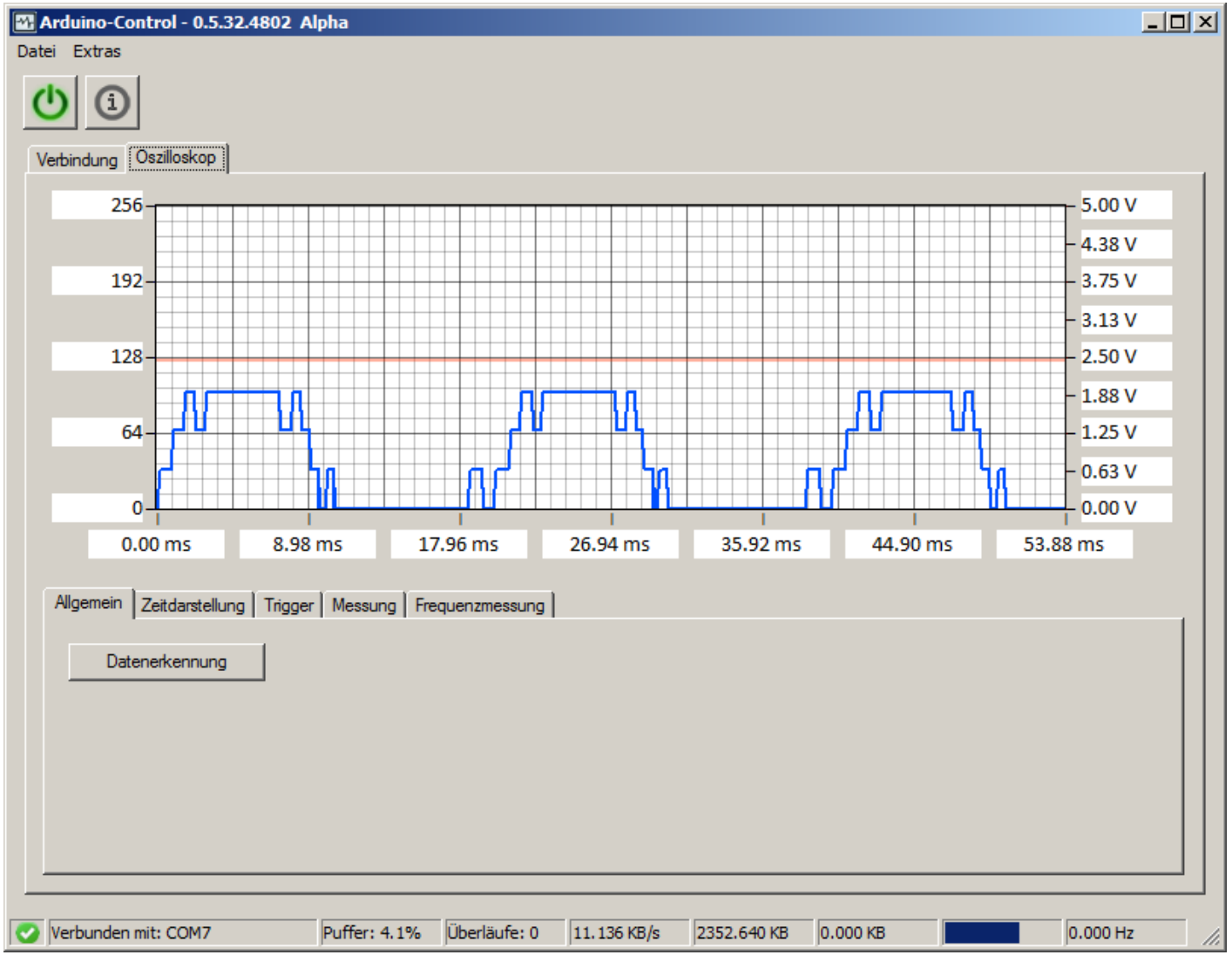
https://drive.google.com/file/d/0Bwp8f5BCaeDLZWNxZGhuNmp0ejA/view?usp=sharing

Arduino Oszilloskop

Quelle: http://forum.arduino.cc/index.php?topic=160703.60

Programm:

https://drive.google.com/file/d/0Bwp8f5BCaeDLQ2R4T1B6d2ZFZlE/view?usp=sharing



Arduino Oszilloskop

ArduinoScope.ino

```
// Definiert die Bitnamen (sbi -> SetBit = 1, cbi -> ClearBit = 0)
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif


void setup() {

   // setzt den ADC-Divisor von 128 Bit auf 16 Bit
   // die Genauigkeit sinkt etwas, aber die Geschwindigkeit steigt
  // um das 6,5-fache
   sbi(ADCSRA,ADPS2);
   cbi(ADCSRA,ADPS1);
   cbi(ADCSRA,ADPS0);

   Serial.begin(230400);
}

void loop() {

  Serial.write(analogRead(0)/4);

}
```