HTL anichstraße

Arduino-Voltmeter
mit
automatischer Messbereichswahl

PBE

Betreuer: Gruber

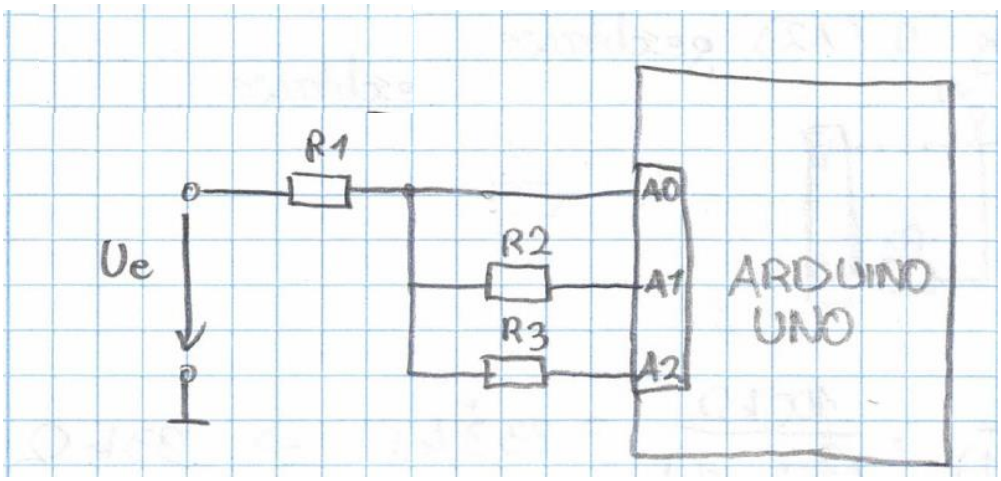
Datum: 17.04.2020

Tobias König

3AHEL 2019/20

1. Idee

Es soll mithilfe eines Arduinos ein Programm entwickelt werden, damit auch Spannungen über 5V an einem analogen Eingang des Arduinos gemessen werden können. Dazu wird der Arduino wie folgt beschaltet:



Die Schaltung erlaubt es einem, je nachdem, ob der analoge Eingang A1 oder A2 des Arduinos als Input oder Output mit 0V benutzt wird, einen Spannungsteiler für bestimmte Messbereiche zu steuern.

Die Analogen Ein-/Ausgänge funktionieren als Schalter gegen GND.

Spannungsteiler über R1 und R2:

- A1 wird auf „OUTPUT“ und 0V gesetzt und
- A2 auf „INPUT“

Spannungsteiler über R1 und R3:

- A2 wird auf „OUTPUT“ und 0V gesetzt und
- A1 auf „INPUT“

Spannungsteiler über R1 und R2||R3:

- **Wird verwendet, um den Messbereich zu bestimmen**
- A2 wird auf „OUTPUT“ und 0V gesetzt
- A1 wird auf „OUTPUT“ und 0V gesetzt

2. Ausführung

2.1 Widerstandsbestimmung

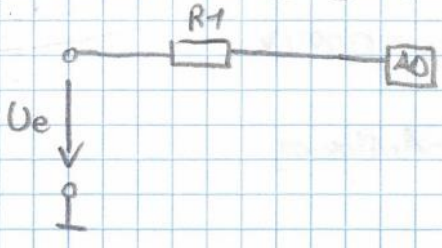
geg: $R1 = 100\,k\Omega$ (weil zuhause)

Messbereiche: $U_{MB1} = 5V$

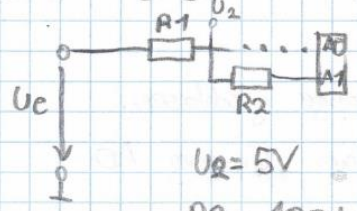
$U_{MB2} = 10V$

$U_{MB3} = 20V$ (mehr geht nicht, da NG-3000 max 20V)

MB = 5 V: beide Eingänge „offen"



MB = 10V: Eingang 1 (A1) geschlossen



$U_2 = 5V$

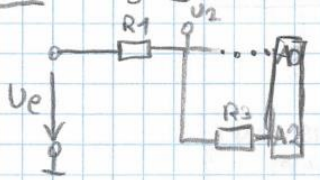
$R2 = 100\,k\Omega$

$$\frac{U_e}{U_2} = \frac{R1+R2}{R2}$$

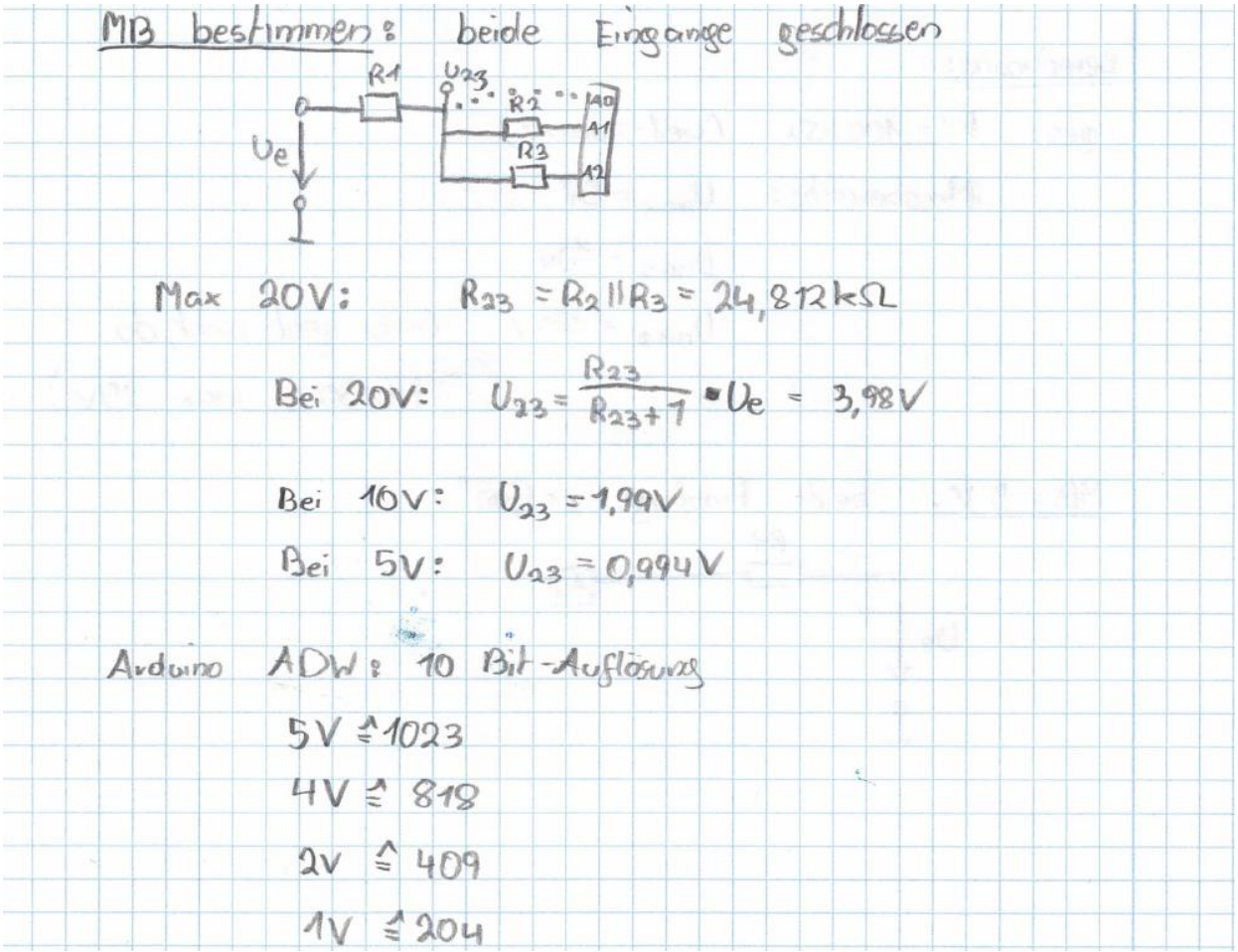
$$R2\left(\frac{U_e}{U_2}-1\right) = R1$$

$$R2 = \frac{R1}{\left(\frac{U_e}{U_2}-1\right)}$$

MB = 20V: Eingang 2 (A2) geschlossen



$$R3 = \frac{R1}{\left(\frac{U_e}{U_2}-1\right)} = \frac{100\,k\Omega}{\left(\frac{20V}{5V}-1\right)} = 33{,}\dot{3}\,k\Omega \Rightarrow 33k\Omega \text{ (E12)}$$



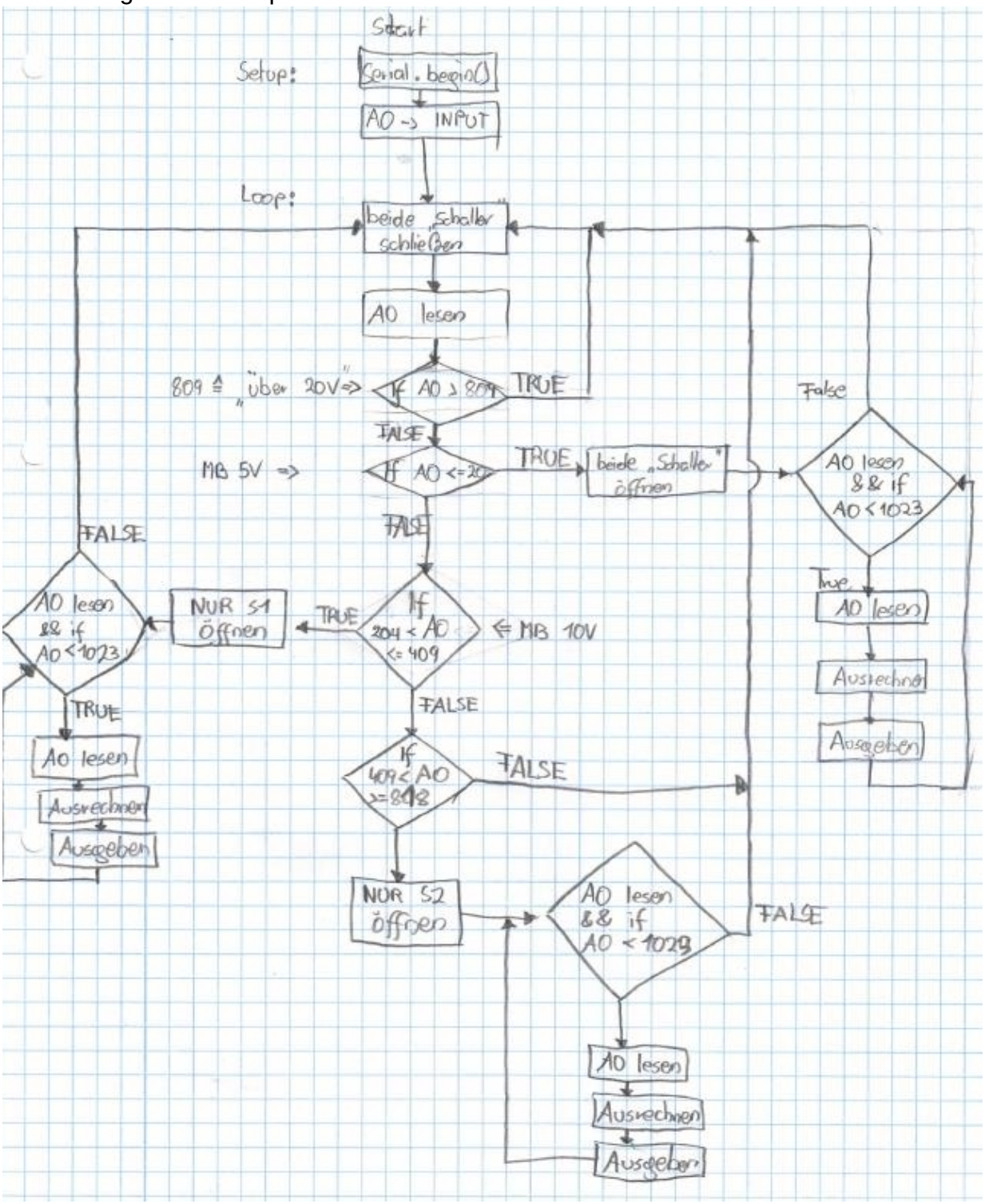
Ergebnis:

- R1 = 100 kΩ
- R2 = 100 kΩ
- R3 = 33 kΩ

Messbereiche:

- 0V…5V
- 0V…10V
- 0V…20V

2.2 Programmablaufplan:



2.3 Programm

```
/* Arduino-Voltmeter mit automatischer Messbereichserkennung

 * PBE-Gruber
 * 17.04.2020
 * Author: Tobias König
*/
byte inputpin = A0;
byte s1 = A1;
byte s2 = A2;
float u;

void setup() {
  pinMode(inputpin, INPUT);
  Serial.begin(9600);
}

void loop()
{
  //Abfrage Messbereich
  //"Schalter" schließen
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);

  analogWrite(s1, 0);
  analogWrite(s2, 0);
  float in = analogRead(inputpin); //Obwohl analogRead() immer Integer
zurück gibt, wird float verwendet, um in der Rechnung Gleikommazahlen als
Ergebnis zu bekommen.
  Serial.println(in);
  delay(1000);

  if (in > 809) //über 20V -> Schutz
  {
    Serial.println("!!!MB > 20V!!!");
  }

  else if (in <= 204)  //MB = 5V
  {
    pinMode(s1, INPUT);
    pinMode(s2, INPUT);
    while (analogRead(inputpin) < 1023)
    {
      Serial.print("MB = 5V | ");
      in = analogRead(inputpin);
      u = in * 5 / 1024;
      Serial.print("U = ");
      Serial.print(u);
      Serial.println("V");
      Serial.print(in);
      delay(1000); //damit nicht alles "durchrauscht"
    }
  }

  else if (in > 204 && in <= 409)  //MB = 10V
  {
    pinMode(s1, OUTPUT);
    analogWrite(s1, 0);
    pinMode(s2, INPUT);

    while (analogRead(inputpin) < 1023 && analogRead(inputpin)>= 526)
    {
```

```
      Serial.println("MB = 10V | ");
      in = analogRead(inputpin);
      u = in * 10 / 1024;
      Serial.print("U = ");
      Serial.print(u);
      Serial.println("V");
      delay(1000);//damit nicht alles "durchrauscht"
    }
  }

  else if (in > 409 && in <= 818)  //MB = 20V
  {
    pinMode(s1, INPUT);
    pinMode(s2, OUTPUT);
    analogWrite(s2, 0);

    while (analogRead(inputpin) < 1023 && analogRead(inputpin)>= 526)
    {
      Serial.print("MB = 20V | ");
      in = analogRead(inputpin);
      u = in * 20 / 1024;
      Serial.print("U = ");
      Serial.print(u);
      Serial.println("V");
      Serial.print(in);
      delay(1000);//damit nicht alles "durchrauscht"
    }
  }
}
```

2.4 Beschreibung

Am Anfang werden beide Eingänge auf GND gezogen und der 10-Bit-Wert der Spannung am Eingang A0 gelesen. Dieser stimmt jedoch nicht dem am Eingang wirklich liegenden Spannungswert überein, da die Widerstände R2 und R3 parallel geschalten werden, und somit der Spannungsteiler belastet wird.

In Punkt 2.1 wurde jedoch dieser belastete Spannungsteiler ausgerechnet und somit kann der aktuelle Spannungswert am Eingang bestimmt werden.

Liegt am Eingang eine Spannung unterhalb der 5V-Grenze, werden beide Eingänge auf „INPUT“ gestellt, wodurch ein hoher Innenwiderstand herrscht. In diesem Fall wird das Signal 1:1 am Eingang des Arduinos gemessen. Nach Umrechnen des 10-Bit-Wertes wird das Signal über den Seriellen Monitor solange ausgegeben und mehrmals gelesen, bis sich das Eingangssignal über die 5V bewegt.

Dann wird auf den zweiten Messbereich (10V) geschalten, und nur der Eingang A1 auf GND gelegt. Der berechnete Spannungsteiler über R1 und R2 sorgt dafür, dass am Eingang des Arduinos nur die Hälfte der ursprünglichen Spannung anliegt.

Liegt der Wert wieder außerhalb der 10V-Grenze, wird in den 20V-Messbereich geschalten. Dort liegt A2 auf GND.

Im Fall, dass die Spannung die Hälfte der maximalen Messbereichsspannung beträgt, wird wieder einen Messbereich zurückgeschalten.

2.5 Auswertung

Das Prinzip des Programms funktioniert soweit, nur liegen durch die Widerstandstoleranzen die Werte nicht genau auf den berechneten „Bit“-Werten, sondern weichen ab. Das bedeutet, dass die Messbereiche nicht im richtigen Moment umschalten, das Problem kann aber durch genaueres Auswerten der Eingangswerte bei den bestimmten Spannungen leicht behoben werden.

Um negative Spannungen zu messen:

2 OPVs, einer mit der Verstärkung -1 und einer, des checkt, ob die Spannung negativ ist…?