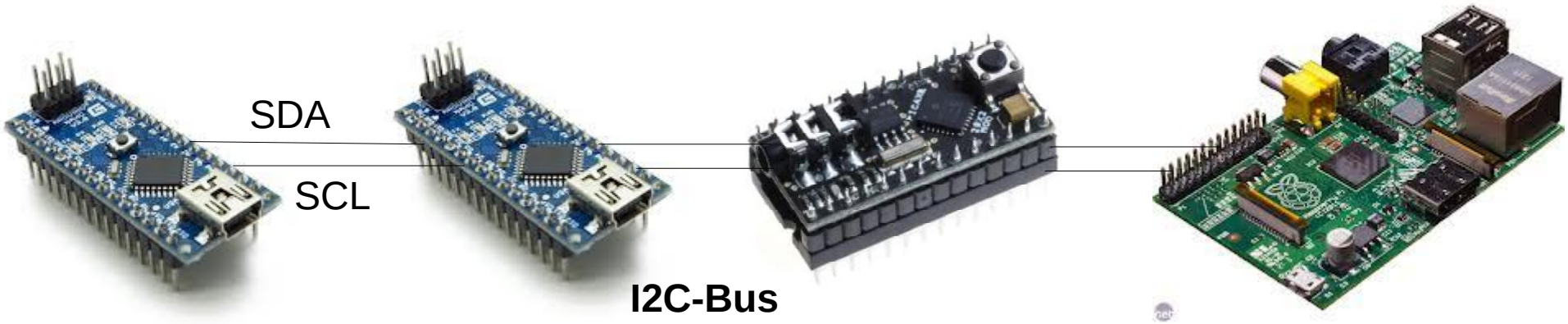
Experiment Arduino als I2C-Master und -Slave

A.Schultze, DK4AQ



I2C-Bus

Der I2C-Bus ist gut für die Kopplung verschiedener Mikrorechner geeignet. Es können Systeme mit Prozessoren verschiedener Hersteller gekoppelt werden.

Geht man von der Entwicklung eigener Leiterplatten aus, so sind die Preise der benötigten Komponenten für z.B. ein minimales Arduino-Systems gering. Es lohnt daher auch die Verwendung eines Arduino als „intelligentes Spezial-Pheripherie-IC“. Der Slave bekommt Steuerinformationen und kann dann intelligente Aufgaben im Kontakt mit seiner Umgebung übernehmen.

Aufgabe:

- Es soll ein Arduino Nano als Master Text und einen hochzählenden Zahlenwert übertragen
- Ein Arduino Nano soll als Slave mit einer eigenen Slave-Adresse diese Informationen empfangen und ausdrucken

Das Beispiel stammt mit kleinen Anpassungen aus
http://arduino.cc/en/Tutorial/MasterWriter

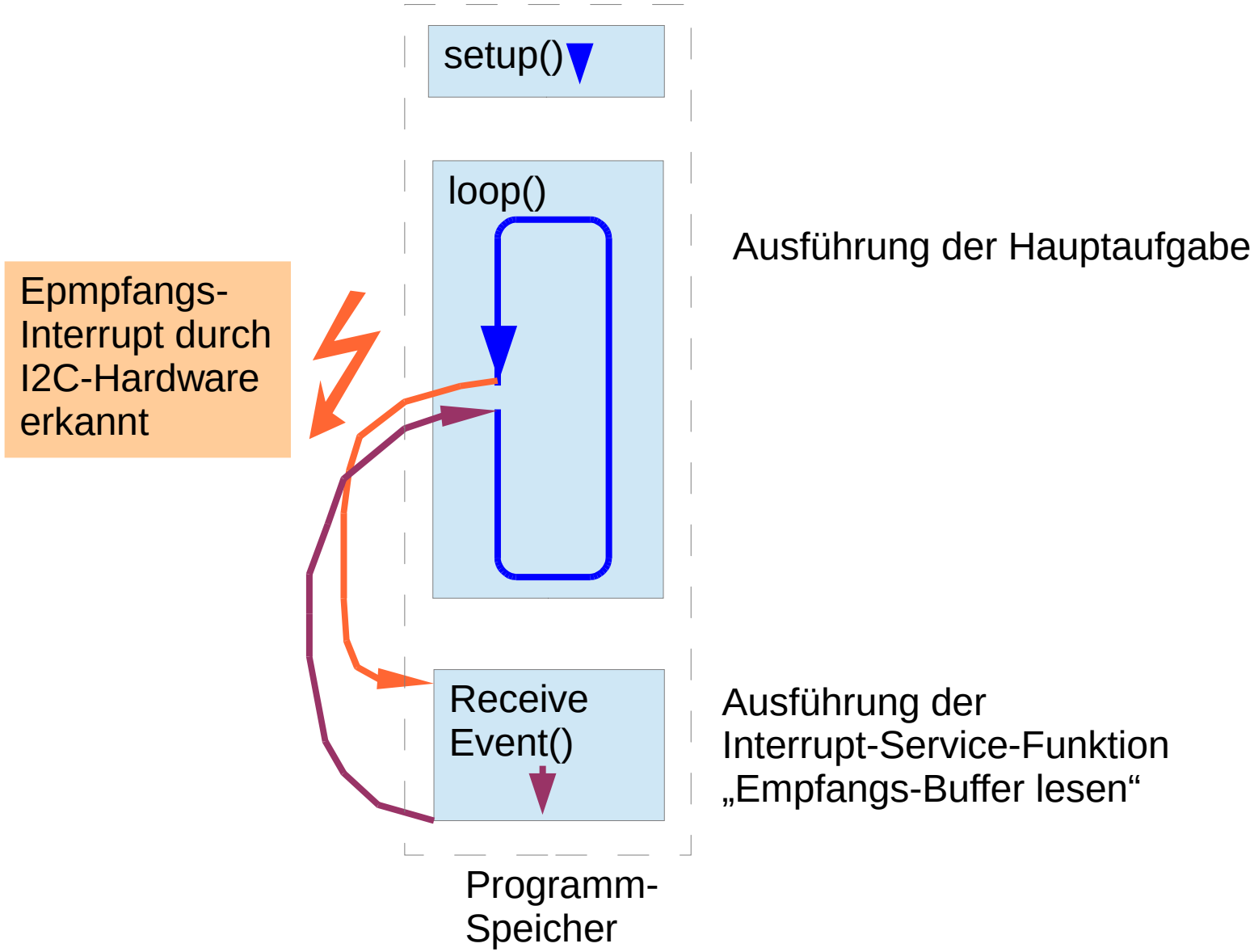
Da wir für die bisherigen Experimente immer Master programmiert haben, beschäftigen wir uns hier mit Arduino als Slave.

Ein Slave muss per Definition immer auf dem Bus horchen ob der Master etwas sendet. Dazu beobachtet er den Start-Vorgang und empfängt mindestens ein Byte, um herauszufinden, ob die eigene Slave-Adresse angesprochen wird. (Es können ja mehrere Slaves auf dem Bus sein)

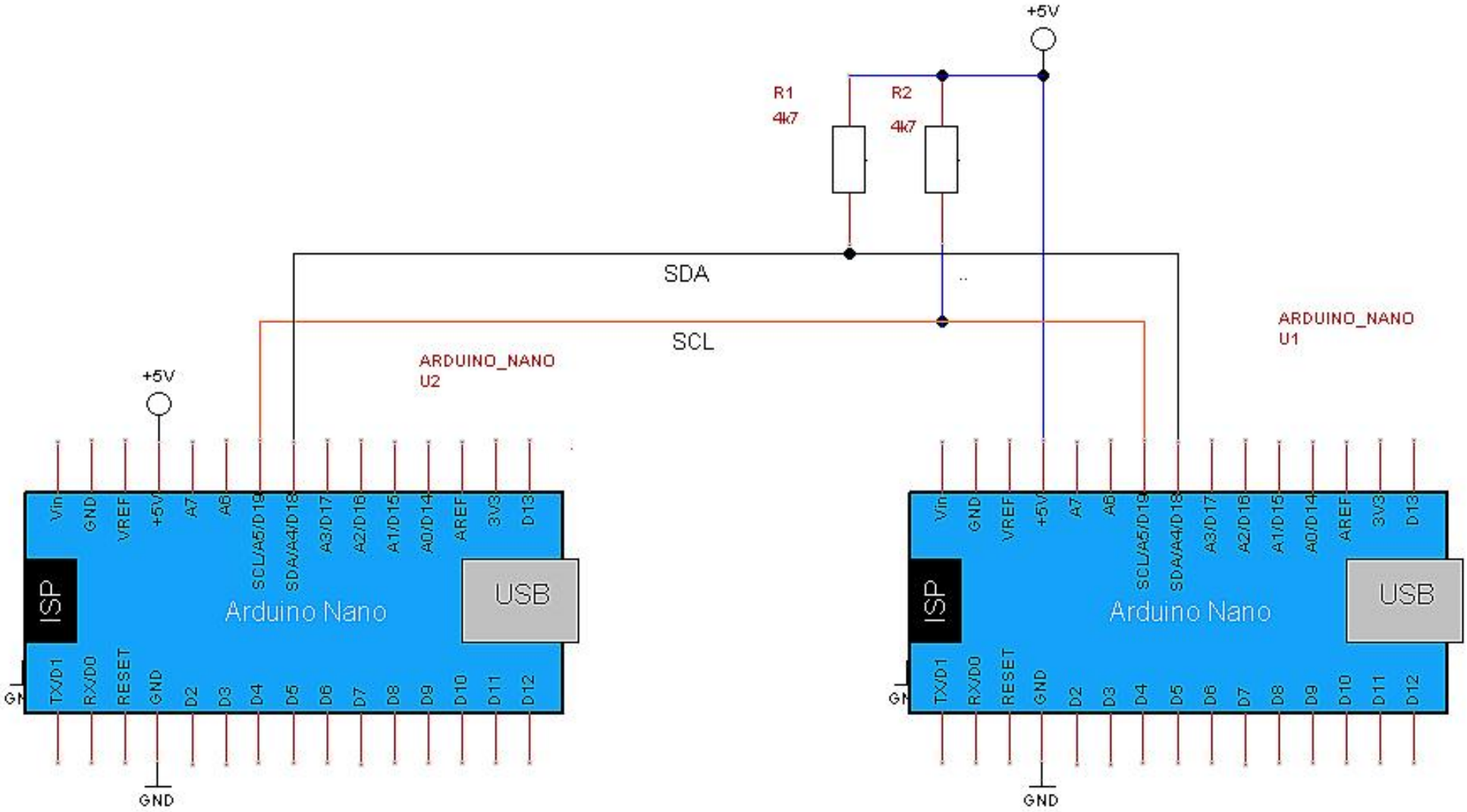
Nun soll ein Slave ja in der Regel auch noch andere Dinge „nebenbei“ tun. Daher wird der Empfang durch einen Interrupt aus der I2C-Hardware gestartet. Interrupt-Funktionen laufen nur dann los, wenn der zugehörige Interrupt eintrifft. Dann rettet der Prozessor schnellstens alle Register, um später dort weitermachen zu können. Es wird die Interrupt-Service-Funktion abgearbeitet und evtl. Werte in globalen Variablen gespeichert. Danach werden die alten Daten in die Prozessorregister gefüllt und die Hauptaufgabe weiter bearbeitet.

Interrupt-Service-Funktionen sollen möglichst kurz sein. Wo es möglich ist, werden kurz Werte gespeichert und die Interrupt-Service-Routine springt ins Hauptprogramm zurück.

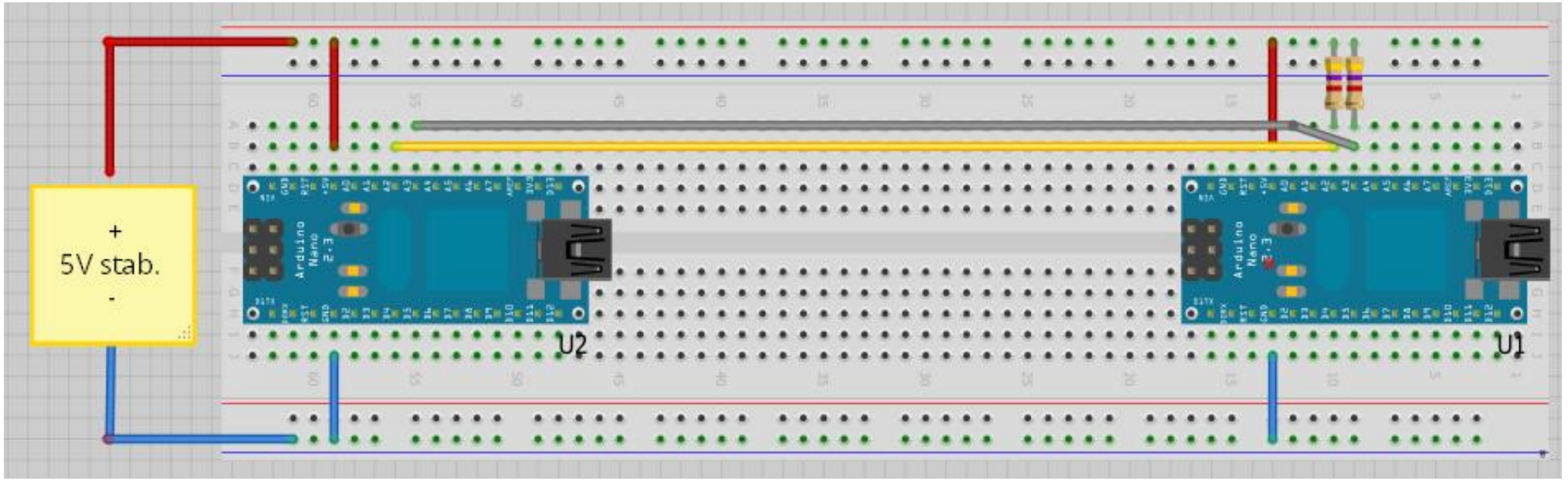
Programmablauf bei Eintreffen eines Interrupts



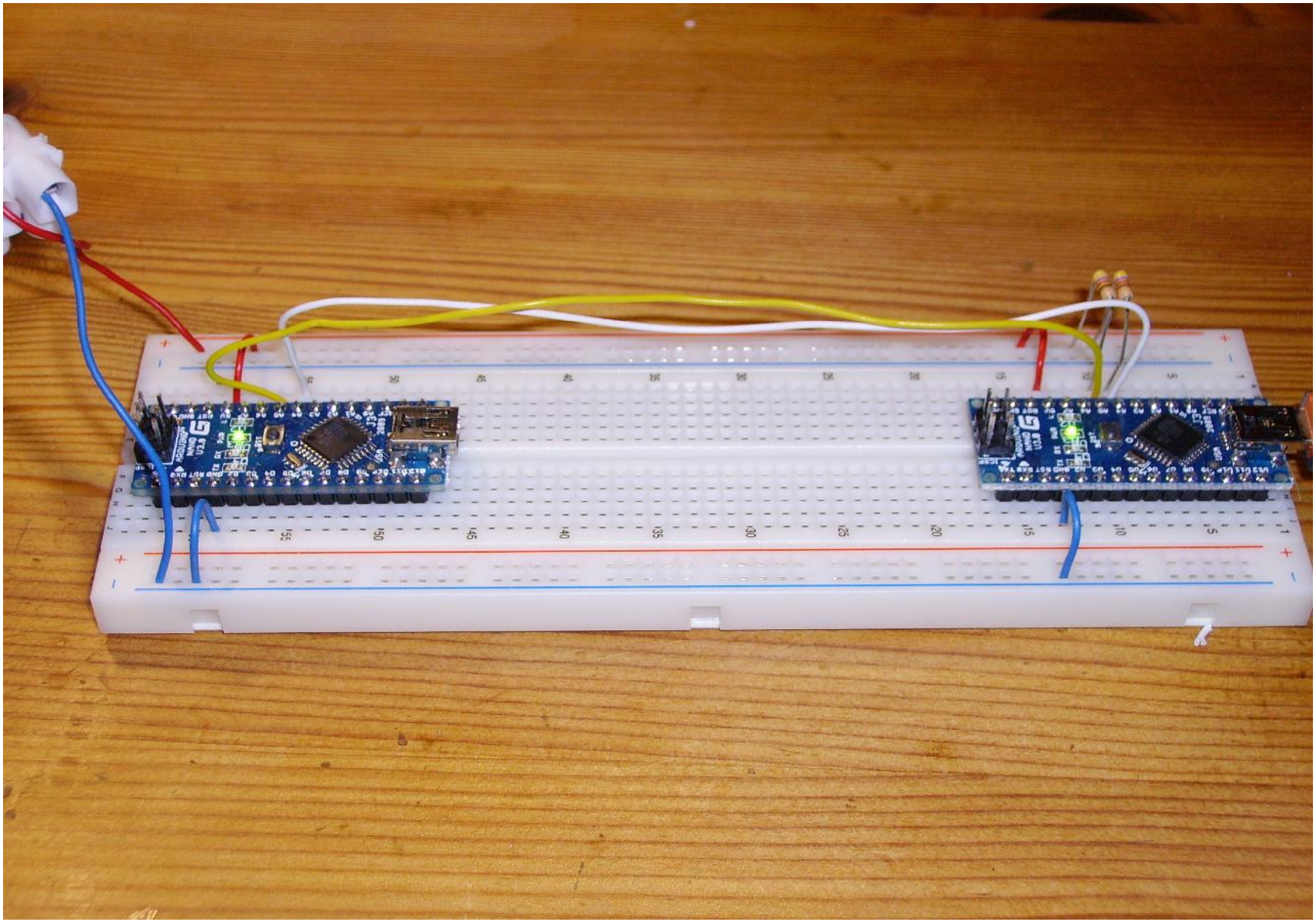
Schaltbild


+5V
R1
4k7
R2
4k7
SDA
SCL
ARDUINO_NANO
U2
ARDUINO_NANO
U1
+5V
ISP
Arduino Nano
USB
Vin
GND
VREF
+5V
A7
A6
SCL/A5/D19
SDA/A4/D18
A3/D17
A2/D16
A1/D15
A0/D14
AREF
3V3
D13
TX/D1
RX/D0
RESET
GND
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
GND

Aufbau-Vorschlag



Aufbau des Experiments



I2C_ArduinoMasterSlave_Exp51

Arduino Master (1)

```
void setup()
{
  Wire.begin(); // join i2c bus (address optional for master)
  Serial.begin(9600);
  Serial.println("Arduino I2C-Master");
}

byte x = 0;
boolean f = 0;

void loop()
{
  Serial.print("x is ");
  Serial.println(x);

  Wire.beginTransmission(4); // Start preparation telegram to device #4
  Wire.write("x is ");        // five bytes to buffer
  Wire.write(x);              // one byte to buffer
  f = Wire.endTransmission(); // transmit telefram and stop
```

I2C_ArduinoMasterSlave_Exp51

Arduino Master (2)

```
  if(f > 0)
  {
    Serial.println("Slave antwortet nicht, Fehler !");
  }

  x++;
  delay(500);
}
```

I2C_ArduinoMasterSlave_Exp51

ArduinoSlave(1)

```
#include <Wire.h>

void setup()
{
  Wire.begin(4);                // join i2c bus with address #4
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);           // start serial for output
  Serial.println("Arduino-Slave empfangsbereit");
}

void loop()
{
  delay(100);
}
```

I2C_ArduinoMasterSlave_Exp51

ArduinoSlave(2)
Interrupt Service Funktion

```
// function that executes whenever data is received from master
// this function is registered as an event, see setup()
void receiveEvent(int howMany)
{
  while(1 < Wire.available()) // loop through all but the last
  {
    char c = Wire.read(); // receive byte as a character
    Serial.print(c);         // print the character
  }
  int x = Wire.read();    // receive byte as an integer
  Serial.println(x);         // print the integer
}
```

Ausgabe Ser.Monitor Master - Slave

```
Arduino I2C-Master          Arduino-Slave empfangsbereit
x is 0                      x is 0
x is 1                      x is 1
x is 2                      x is 2
x is 3                      x is 3
x is 4                      x is 4
x is 5                      x is 5
x is 6                      x is 6
x is 7                      x is 7
x is 8                      x is 8
x is 9                      x is 9
x is 10                     x is 10
x is 11                     x is 11
x is 12                     x is 12
x is 13                     x is 13
x is 14                     x is 14
x is 15                     x is 15
x is 16                     x is 16
```