Hochschule Karlsruhe - Technik und Wirtschaft
Elektro- und Informationstechnik
Messtechnik-Labor

Messwerterfassung mit Digitalmultimeter

K. Wolfrum, Th. Münch, Ch. Thiele

8. Oktober 2020

Inhaltsverzeichnis

1 Lernziele

- Umgang mit einem Digitalmultimeter
  - Vornehmen der Grundeinstellungen
  - Nullpunktkorrektur
  - Vornehmen verschiedener Einstellungen über die serielle Schnittstelle
  - Messung von Strömen, Spannungen und Widerständen
- Umgang mit der seriellen Schnittstelle RS232
  - Einstellen der Schnittstelle
  - Kennenlernen der prinzipiellen Vorgehensweise
  - Manuelles Auslesen von Daten
- Verwendung des Jupyter-Notebook und Python
  - Kennenlernen der Software und einzelner Funktionen
  - Ansprechen der seriellen Schnittstelle
  - Verwendung des Jupyter-Notebook zum automatisierten Auslesen und Auswerten von Messdaten über die serielle Schnittstelle
- Widerstandsmessung
  - Vergleichen von strom- und spannungsrichtigen Messschaltungen
- Messabweichungen und Fehlerrechung
  - Berechnung und Beurteilung der Kennwerte Mittelwert, Median und der Standardabweichung, sowie Erfahren des Einflusses der Messwertanzahl auf diese Kennwerte
  - Beurteilung und Erfahren der Einflussnahme der systematischen und statistischen Messfehler

2 Grundlagen

2.1 Einführung RS232-Schnittstelle

Die serielle Schnittstelle dient zum Datenaustausch zwischen Computer und einem Peripheriegerät. Eine RS-232 Schnittstelle arbeitet (bit-)seriell mit je einer Datenleitung für beide Übertragungsrichtungen. Das heißt, die Bits werden nacheinander auf einer Leitung übertragen, wobei in der Regel 8 Datenbits von einem Startbit und mindestens einem Stopbit eingerahmt werden. Weitere moderne serielle Schnittstellen sind z. B. Ethernet, USB, Firewire, CAN-Bus oder RS-485. In diesem Versuch soll nun der grundlegende Umgang mit dieser Schnittstelle gezeigt werden. Für diesen Versuch wird ein USB-Seriell Adapter verwendet, wie in Abbildung 1 dargestellt.

Abbildung 1: USB-Serial Adapter

2.2 Virtueller COM-Port

Um mit der Schnittstelle arbeiten zu können muss ein virtueller COM-Port angelegt werden. Dieser ist notwendig, da mit einem USB-Seriell Adapter gearbeitet wird. Somit wird aus einem USB Anschluss ein (virtueller) serieller Anschluss.

2.3 Tischmultimeter Escort 3136A

Das Escort 3136A Tischmultimeter ist ein einfach zu bedienendes Messgerät und vielseitig einsetzbar. Abbildungen 2, 3 zeigen Front- und Rückansicht des Geräts. Die Hauptfunktionen sind:

- 1 μV Auflösung bei VDC Messung
- Berechnung der RMS-Werte („Echter“ Effektivwert) für VAC + VDC bzw. AAC + ADC
- Widerstandsmessung mittels Zweileitermethode
- Frequenzmessung bis zu 1 MHz
- Fernsteuerung über RS-232 Schnittstelle

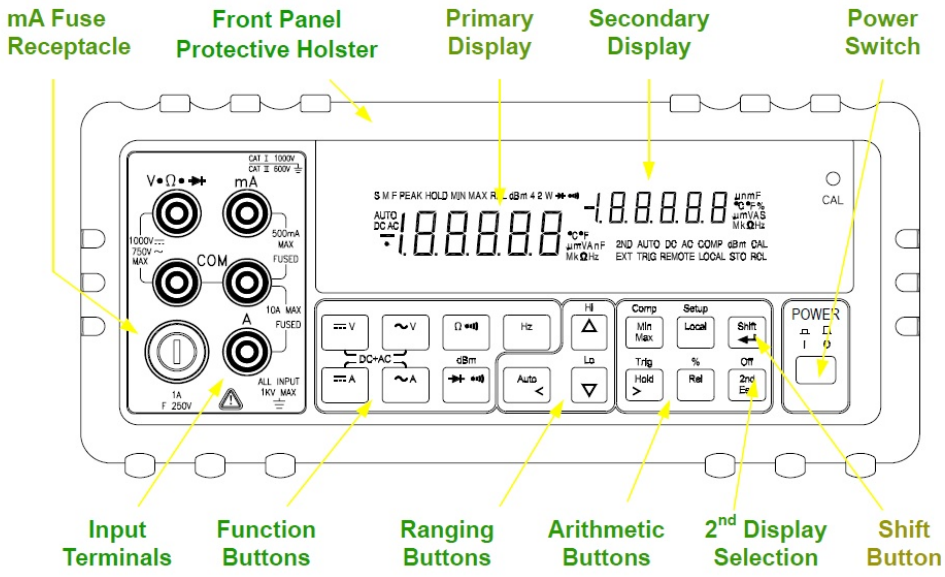


Abbildung 2: Tischmultimeter Vorderseite

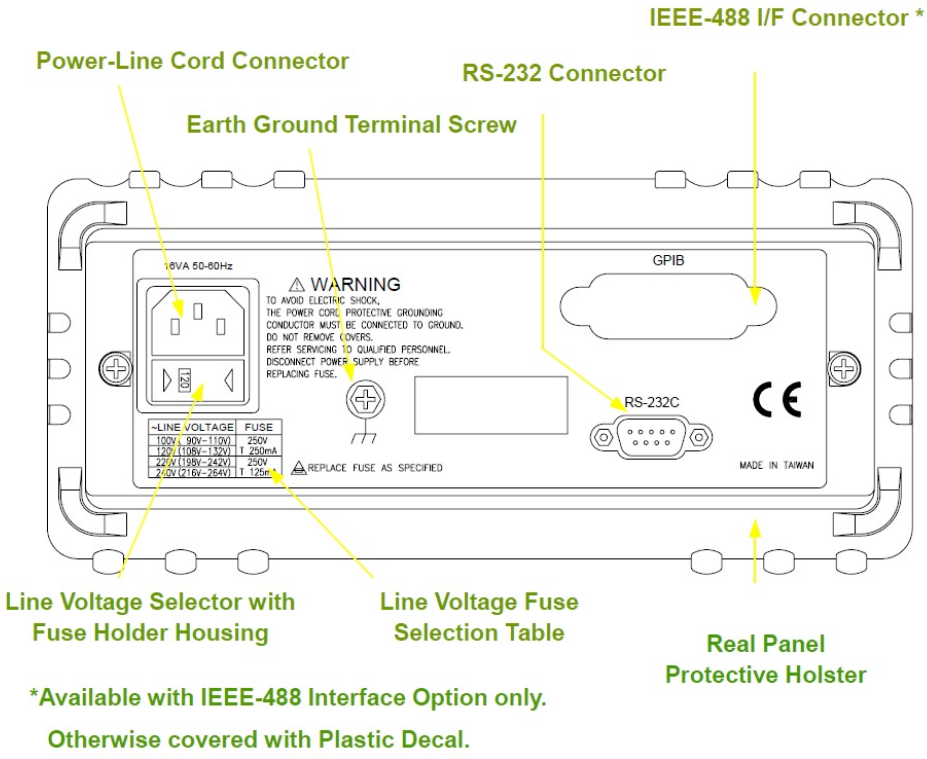


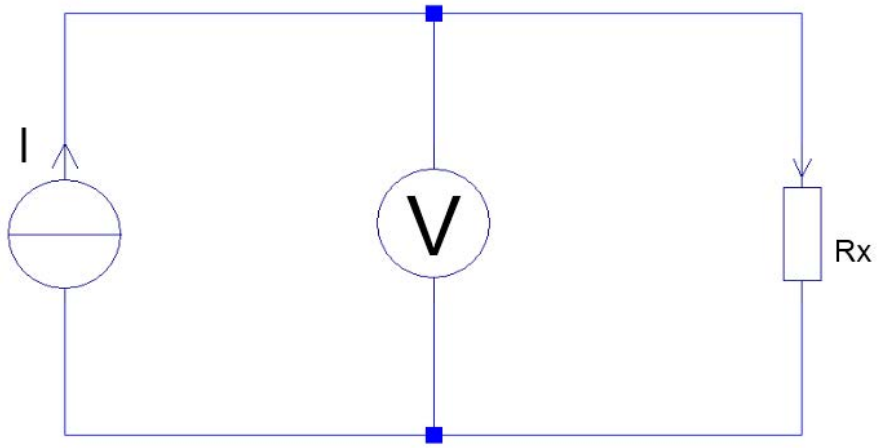
Abbildung 3: Tischmultimeter Rückseite

2.4 Widerstandsmessung mit der Zweileitermethode

Prinzipiell gibt es für die direkte Messung mittels der Zweileitermethode drei Möglichkeiten:

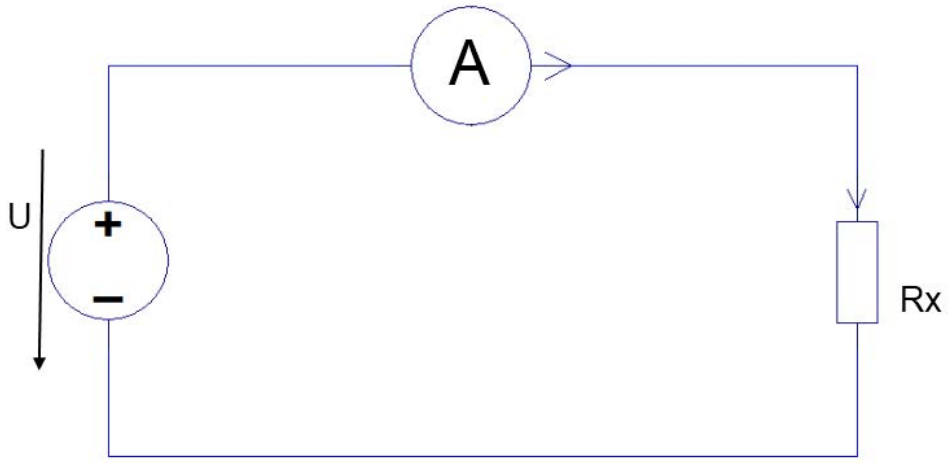
1. **Widerstand an einer idealen Stromquelle**

- Der zu messende Widerstand Rx wird von einem konstanten Strom durchflossen, der Spannungsabfall am Widerstand ist proportional zum Widerstandswert.



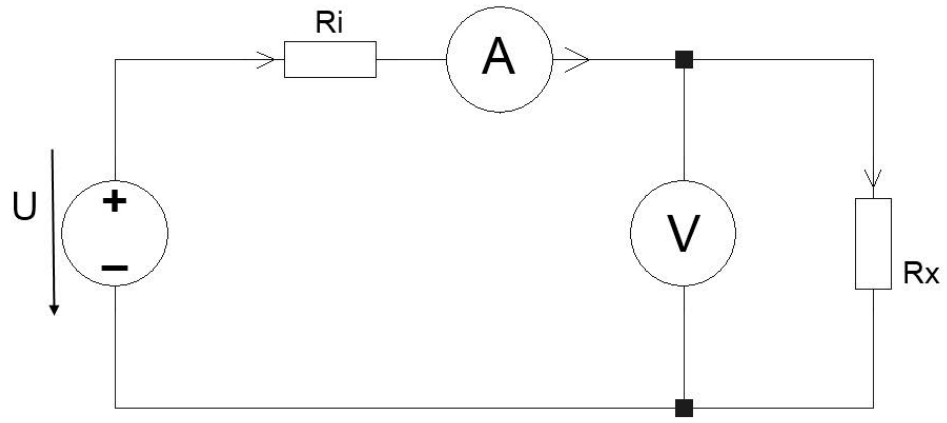
2. **Widerstand an einer idealen Spannungsquelle**

- Der zu messende Widerstand wird an eine konstante Spannung gelegt. Der Strom durch den Widerstand ist proportional zum Leitwert (G=1/Rx).



3. **Widerstand an einer realen Spannungsquelle**

- Der zu messende Widerstand wird an eine reale Spannungsquelle mit Quellenspannung und Innenwiderstand angeschlossen. Aus der Messung des Stroms durch den Widerstand sowie der am Widerstand abfallenden Spannung kann der Widerstandswert bestimmt werden.



3 Versuchsvorbereitung

3.1 Bedienung der Software Jupyter-Notebook

Jupyter-Notebook startet einen Server, die Bedienung erfolgt über einen Browser. Am übersichtlichsten ist es, wenn Jupyter-Notebook über die Kommandozeile im Anaconda Propmpt gestartet wird. Ermitteln Sie zunchst den Speicherort des Ordners der Datei “Versuch_DMM.ipynb” die sich in einem Unterordner von “DMM” auf dem Desktop befindet (Rechtsklick auf das Symbol -> Eigenschaften).

Öffnen Sie einen Anaconda Prompt, z. B. durch Eintippen von “Anaconda Prompt” im Windows Startfenster. In der Kommandozeile wechseln Sie durch Eingabe des Befehls “cd ” in das Verzeichnis, in dem sich das Jupyternotebook “Versuch_DMM.ipynb” befindet.

Durch Eingabe von jupyter-notebook Versuch_DMM.ipynb wird schließlich der jupyter-notebook Server gestartet und das aufgerufene Notebook geöffnet.

Fenster werden mit “Shift Return” ausgeführt, bei Codefenstern wird der darin befindliche Python-Code abgearbeitet. Wenn es bei längerem Herumprobieren zu Ungereimtheiten kommt, sollte im Zweifelsfall der Kernel neu gestartet werden (Kernel -> Restart & Clear Output).

Dieses Notebook ist dazu da, mit Ihren Beobachtungen, Messungen und Kommentaren ergänzt zu werden. Nach Beendigung des Versuchs haben Sie dann bereits ein fertiges Versuchsprotokoll, das nur noch von einem Betreuer durgesehen werden muss.

3.2 Bedienung des digitalen Tischmultimeters Escort 3136A

Bereiten Sie anhand der Bedienungsanleitung des digitalen Multimeters Escort 3136A vor, wie Sie

- Spannungs-, Strom- und Widerstandsmessung am Multimeter einstellen,
- eine Nullpunktkorrektur vornehmen, um einen Nullpunktfehler (Offset) zu kompensieren,
- die Vorgabewerte für die serielle Schnittstelle am Multimeter einstellen (Baud-Rate, Parity, Data-Bit, Stop-Bit, Echo und Printer-Only).

Lesen Sie den Abschnitt in der Bedienungsanleitung des Multimeters und notieren Sie sich die für die verschiedenen Versuche benötigten KEY, SET und QUERY-Befehle. Hinweis: Wird im Display des Multimeters „Remote“ angezeigt, so befindet sich das Multimeter im Fernbedienungsmodus. In diesem Modus sind einige Tasten am Multimeter ohne Funktion. Durch Drücken der Taste „Local“ am Multimeter können Sie den Fernbedienungsmodus verlassen.

4 Versuchsdurchführung

4.1 Ansprechen der Schnittstelle

4.1.1 Grundeinstellung des digitalen Multimeters

Für den ersten Versuchsteil wird nur ein Multimeter benötigt. Damit Python über die Schnittstelle einwandfrei kommunizieren kann, müssen die Grundeinstellungen des Messgeräts überprüft werden. Hierzu sind folgende manuelle Eingaben auf den Tasten des Geräts notwendig: Öffnen Sie das Menü des Multimeters.

- Shift –> Setup
- RS232 auswählen und mit Shift/Enter bestätigen
- mit 5-Taste folgende Einstellungen anwählen:
  - baud = 9600
  - parity = none
  - data = 8 Bit
  - stop = 1 Bit
  - echo = off
  - print = off

Ändern der Werte unter Verwendung von:

- mit / . Wert ändern
- Bestätigen mit Shift/Enter
- Menü verlassen mit ESC

Überprüfen Sie die Einstellungen in beiden Multimetern. Deaktivieren Sie vor allem die akustische Tastatur-Quittierung durch Abschalten im Menüpunkt „Beep –> Off“.

4.2 Arbeiten mit der Schnittstelle in Jupyter-Notebook mit Python

4.2.1 COM-Ports mithilfe des Gerätemanagers ermitteln

Öffnen Sie den Geräte-Manager von Windows und finden Sie unter **Anschlüsse (COM & LPT)** die virtuellen COM-Ports, an denen die beiden Digitalmultimeter angeschlossen sind.

Alternativ kann dazu auch Python verwendet werden. Verwenden Sie für den ersten Versuchsteil nur ein Multimeter.

[ ]:
```
# Namen der Schnittstellen werden in ports.txt abgelegt
import os

# Python-Befehl wird an das Betriebssystem übergeben und das Resultat über␣
 ↪Pipeline in Datei geschrieben
os.system("python -m serial.tools.list_ports > ports.txt")

# Liste der Ports wird initialisiert
port = []

f = open("ports.txt")
portList = f.readlines()
#print(portList)
print("available serial ports")
for i in range(len(portList)) :
    tmpPort = portList[i]
    port.append(tmpPort[0:6])
    print(port[i])
```

Ordnen Sie die gefundenen Schnittstellen den Digitalmultimetern zu.

4.3 Kommunikation zwischen PC und DMM

4.3.1 Module importieren und Parameter einstellen

[ ]:
```
#import dmm
import serial
import time
```

Zunächst muss die Kommunikation über die serielle Scnnittstelle initialisiert und die Schnittstelle zum Lesen und Schreiben geöffnet werden.

[ ]:
```
timeoutVal = 1              # timeout für Zugriff auf serielle Schnittstell 1␣
 ↪Sekunde
portName   = ''             # zutreffende Portnummer eintragen

dmm = serial.Serial(portName, timeout = timeoutVal)
```

Bei einer Fehlermeldung an dieser Stelle war die Schnittstelle vermutlich schon geöffnet. Ist keine Fehlermeldung aufgetreten, führen Sie den obigen Befehl einfach ein weiteres Mal aus. Daher ist ein sogenanntes Exception-Handling empfehlenswert:

[ ]:
```
timeoutVal = 1              # timeout für Zugriff auf serielle Schnittstell 1␣
 ↪Sekunde
portName   = ''             # zutreffende Portnummer eintragen

try :
    dmm = serial.Serial(portName, timeout = timeoutVal)

except :
    print('exception handler:')
    print('port ' + portName + ' already in use')
    print('closing port ' + portName)
    print('opening port ' + portName)
    dmm.close()
    dmm = serial.Serial(portName, timeout = timeoutVal)

print("DMM connected to", dmm.name)
```

Es können weitere Eigenschaften des Objekts ausgegeben werden:

[ ]:
```
print('dmm = ', dmm)
print('dmm.baudrate = ', dmm.baudrate)
```

Nachdem die Schnittstelle erfolgreich geöffnet wurde, kann die Kommunikation mit dem Gerät erfolgen. Dabei ist zu beachten, dass jeder Befehl, der zum Gerät geschickt wird, von diesem durch mindestens eine Rückmeldung quittiert wird.

Rückmeldungen gemäß Table 6-2 der DMM-Anleitung

Table 6-2. RS-232 Return Prompts

Prompts	Description
* >	The meter is reset to power-up initialisation status.
= >	A command is executed and no errors are detected.
! >	A command error is detected.
? >	A parameter error is detected.
# >	The local key is pressed.
S >	The set up function is under executing.
@ >	No numeric reading is available.
W>	When the setting value HI<LO or LO>HI.
E>	Execution error, or disallow to execute. For example, the K13 will not be allowed after LLO.

- **Return result**

After the meter executes a query command the return of the result will be in the following format:

<RESULT> + <CR> <LF> + <PROMPT> + <CR><LF>

Alle Funktionen des Geräts könne über die serielle Schnittstelle gesteurt werden. Besonders einfach sind hierbei die sogenannten Key Commands, die die Betätigung der Tasten an der Frontplatte nachbilden. D. h. das Gerät reagiert auf den Key Command so, als wäre per Hand die zugehörige Taste gedrückt worden.

Die Key Commands sind in Table 6-3 der DMM-Anleitung aufgeführt.

Table 6-3. Descriptions for Key Commands

Command	Equivalent Keystroke on the front panel
K1	Press Vdc key
K2	Press Adc key
K3	Press Vac key
K4	Press Aac key
K5	Press Ω key
K6	Press Diode key
K7	Press Hz key
K8	Press AUTO key
K9	Press △ key
K10	Press ▽ key
K11	Press MinMax key
K12	Press Hold key
K13 *1	Press Local key
K14	Press REL key
K15	Press Shift key
K16	Press 2nd key.
K17	Press Vdc and Vac keys simultaneously
K18	Press Adc and Aac keys simultaneously
K19	Press Shift and △ keys on the front panel. (Increasing the intensity of VFD display)
K20	Press Shift and ▽ keys on the front panel. (Decreasing the intensity of VFD display)

Note: The K13 will be disabled after LLO command. For K15 then K13, it is always disabled.

Wichtig für die korrekte Kommunikation ist, dass jeder Befehl korrekt terminiert wird. Hierzu dienen die Zeichen CR (carriage return) und LF (line feed). Dabei ist zu beachten, dass bei Schreibbefehlen in Python automatisch am Ende ein LF gesendet wird. Daher ist nur noch ein CR zu ergänzen, das in Python durch c̊odiert wird.

```
\r
```

4.3.2 Beispiel Reset-Befehl (Multimeter zurücksetzen)

Von den weiteren Befehle zum Steuern des Multimeters soll zunächst der Reset-Befehl verwendet werden (Table 6-9 sowie S. 63 ff.) Nach dem Reset-Befehl wird dieser zunächst quittiert und anschließend der durchgeführte Reset bestätigt.

[ ]:
```
# Befehl an Gerät
numBytes = dmm.write(b'RST\r')                # \r entspricht CR, \n wird␣
 ↪automatisch gesendet
print("Zahl der übertragenen Bytes: ", numBytes)
print(dmm.readline())
time.sleep(2)                                 # kurz warten, da die Ausführung␣
 ↪etwas dauert
print(dmm.readline())
print(dmm.readline())
```

Vergleichen Sie die empfangenen Quittierungen mit Table 6-2 und erläutern Sie die Antwort des Geräts.

4.3.3 Firmware-Version auslesen

Ermitteln Sie aus der Anleitung den erforderlichen Befehl zum Auslesen der Firmware-Version und ergänzen Sie die folgenden Codezeilen.

[ ]:
```
# Befehl an Gerät
numBytes = dmm.write(b'RV\r')
print("Zahl der übertragenen Bytes: ", numBytes)
print(dmm.readline())
print(dmm.readline())
print(dmm.readline())
print(dmm.readline())
```

Vor dem nächsten Abschnitt muss die Schnittstelle wieder geschlossen werden.

[ ]:
```
dmm.close()
```

4.4 Widerstandsmessung mit Zweileitermethode

4.4.1 Verbindung herstellen

Tragen Sie für diese Messung den Port des Multimeters ein, welches Sie für die **Widerstandsmessung** verwenden wollen.

[ ]:
```
timeoutVal = 1
port = ''

try :
    resistance=serial.Serial(port, timeout = timeoutVal)
except :
    resistance.close()
    resistance=serial.Serial(port, timeout = timeoutVal)
print("Voltmeter connected to ", resistance.name)
```

4.4.2 Widerstandsmessung einstellen

Finden Sie in der Bedienungsanleitung des Multimeters den Befehl zur Einstellung des Messmodus **Widerstandsmessung** mit **automatischer** Messbereichseinstellung.

Tragen Sie die den Befehl zur Umschaltung auf Widerstandsmessung zwischen den Hochkommas ein:

[ ]:
```
# Befehl an Gerät
numBytes = resistance.write(b'')
print("Zahl der übertragenen Bytes: ", numBytes)
```

[ ]:
```
# Rückmeldung vom Gerät
resistance.readline()
```

4.4.3 Nullpunktkorrektur

[ ]:
```
# Befehl an Gerät
numBytes = resistance.write(b'')
print("Zahl der übertragenen Bytes:", numBytes)
```

[ ]:
```
# Rückmeldung vom Gerät
resistance.readline()
```

4.4.4 Einlesen der primären Anzeige

Beim Einlesen von Anzeigewerten gibt das Gerät zunächst den Zahlenwert zurück und anschließend die Quittierung. Ermitteln Sie den Befehl zum Auslesen der primären Anzeige.

[ ]:
```
resistance.write(b'')
resValue = resistance.readline()
print(resValue)

ackValue = resistance.readline()
print(ackValue)

# serieller Puffer wird geleert um zu sehen, ob noch weitere Meldungen kommen
while len(ackValue) > 3 :
        ackValue=resistance.readline()
        print("length of ackValue string = ", len(ackValue), ", ackValue string␣
 ↪= ", ackValue)
```

Was passiert, wenn nach Abschicken eines Lesebefehls der serielle Puffer nicht vollständig geleert wird? Probieren Sie es aus, indem Sie nach einem Lesebefehl nur einmal readline() ausführen und dann nochmal einen Lesebefehl an das Gerät schicken.

Es soll zum Abschluss dieses Versuchsteils der Kontaktwiderstand eines Relaiskontakts gemessen werden. Verdrahten Sie dazu die Relaisplatine so, dass der Relaiskontkt durch Betätigung eines Schalters geschlossen werden kann.

Messen Sie zunächst den Widerstand durch eine Zweileitermessung (Messbereich Ohm) mit und ohne Nullpunktkorrektur.

Protokollieren Sie die Messergebnisse hier:

Wenn genug Zeit vorhanden ist, sollten Sie noch ausprobieren, wie der Messbereich des DMM über die serielle Schnittstelle umgeschaltet werden kann (und natürlich hier mitprotokollieren).

[ ]:
```
# hier könnte der Code zum Umschalten des Messbereichs stehen
```

4.4.5 Verbindung schließen

Nach Beendigung der Messungen muss die Schnittstelle wieder ordnungsgemäß geschlossen werden.

[ ]:
```
# Befehl zum Schließen der Verbindung
resistance.close()
```

4.5 Messung sehr kleiner Widerstände mit Vierleitermessung

Der Kontaktwiderstand soll nun durch eine Vierleitermessung bestimmt werden. Verwenden Sie dazu eines der DMM als Voltmeter und eines als Amperemeter. Als Spannungsquelle für die Messung dient ein Labornetzteil mit eingestellter Strombegrenzung I = 0,1 A. Das Labornetzteil demzufolge als Stromquelle, wenn die Strombegrenzung einsetzt

Verwenden Sie die Relaisplatine (vgl. Abschnitt Halbautomatisierte Messung)

[ ]:
```
# hier muss die korrekte Portzuordnung eingetragen werden
voltPort = ''           # für Spannungsmessung
currPort = ''           # für Strommessung
```

Überprüfen Sie die Zuordnung, indem Sie die Verbindung zu den DMMs aufbauen und einen Reset durchführen.

[ ]:
```
timeoutVal = 1

port = voltPort

try :
    volt = serial.Serial(port, timeout = timeoutVal)
except :
    volt.close()
    volt = serial.Serial(port, timeout = timeoutVal)
print("Voltmeter connected to ", volt.name)
```

[ ]:
```
timeoutVal = 1

port = currPort

try :
    curr = serial.Serial(port, timeout = timeoutVal)
except :
    curr.close()
    curr = serial.Serial(port, timeout = timeoutVal)
print("Amperemeter connected to ", curr.name)
```

[ ]:
```
# Befehl an Gerät (Reset)
numBytes = curr.write(b'')
print("Zahl der übertragenen Bytes:", numBytes)
print(curr.readline())
time.sleep(2)
print(curr.readline())
print(curr.readline())
```

```
[ ]: # Befehl an Gerät (Reset)
numBytes = volt.write(b'')
print("Zahl der übertragenen Bytes:", numBytes)
print(volt.readline())
time.sleep(2)
print(volt.readline())
print(volt.readline())
```

```
[ ]: # Befehl an Gerät (Spannungsmessung)
numBytes = volt.write(b'')
print("Zahl der übertragenen Bytes:", numBytes)
print(volt.readline())
time.sleep(2)
print(volt.readline())
print(volt.readline())
```

```
[ ]: # Befehl an Gerät (Strommessung)
numBytes = curr.write(b'')
print("Zahl der übertragenen Bytes:", numBytes)
print(curr.readline())
time.sleep(2)
print(curr.readline())
print(curr.readline())
```

```
[ ]: # Messung des Stroms
numBytes = curr.write(b'')
current = curr.readline()
print('current=', current)

ackValue = curr.readline()
print(ackValue)
ackValue = curr.readline()
print(ackValue)
ackValue = curr.readline()
print(ackValue)
```

```
[ ]: # Messung der Spannung
numBytes = volt.write(b'')
voltage = volt.readline()
print('voltage=', voltage)

ackValue = volt.readline()
print(ackValue)
ackValue = volt.readline()
print(ackValue)
```

Zur Umwandlung des empfangenen Strings in eine Fließkommavariable gibt es mehrere Möglichkeiten. Da in vielen numerischen Projekten häufig das Paket numpy verwendet wird, verwenden

wir es hier auch, allerdings sparsam:

```
[ ]: #import numpy as np           # so wird das gesamte numpy Paket importiert
                                   # die Methode kann dann unter dem alias np␣
      ↪aufgerufen werden:   np.float(())

     #U = np.float(voltage)
     #I = np.float(current)

     from numpy import float  as np_float     # das ist ein sparsamerer Import, der␣
      ↪nur die Methode float importiert
                                              # und unter dem alias "np_float"␣
      ↪verfügbar macht

     U = np_float(voltage)
     I = np_float(current)
     R = U / I

     print('U =', U, 'I =', I, 'R =', R)
```

Führen Sie die Vierleitermessung sowohl in der spannungsrichtigen als auch in der stromrichtigen Variante durch und protokollieren Sie die Messergebnisse hier:

4.6 Halbautomatisierte Messung

Grundsätzlich können mit den oben kennengelernten Befehlssequenzen beliebig lange Messreihen durchgeführt werden. Um die Handhabung zu erleichtern, wurden die Befehlssequenzen in kleine Unterprogramme gepackt, die durch Python-Funktionsaufrufe gestartet werden können. Damit sind einfach halb- und vollautomatische Messungen durchführbar.

4.6.1 Software

Für die Durchfürung der halbautomatisierten Messung dient das Modul dmm.py, das zahreiche hilfreiche Methoden bereitstellt. Ein Blick in den Quellcode ist zur Vorbereitung sinnvoll.

4.6.2 Anschluss und Einstellung der Multimeter

Legen Sie fest, welches der Multimeter als Voltmeter und welches als Amperemeter verwendet werden soll.

4.6.3 Aufbau der Messschaltung

Die Messschaltung soll als spannungsrichtige Vierleitermessung aufgebaut werden.

4.6.4 Versuchsplatine

Der zu vermessende Relaiskontakt befindet sich auf der Relaisplatine, die in Abb. 4 dargestellt ist. Der Schaltplan findet sich in Abb. 5.

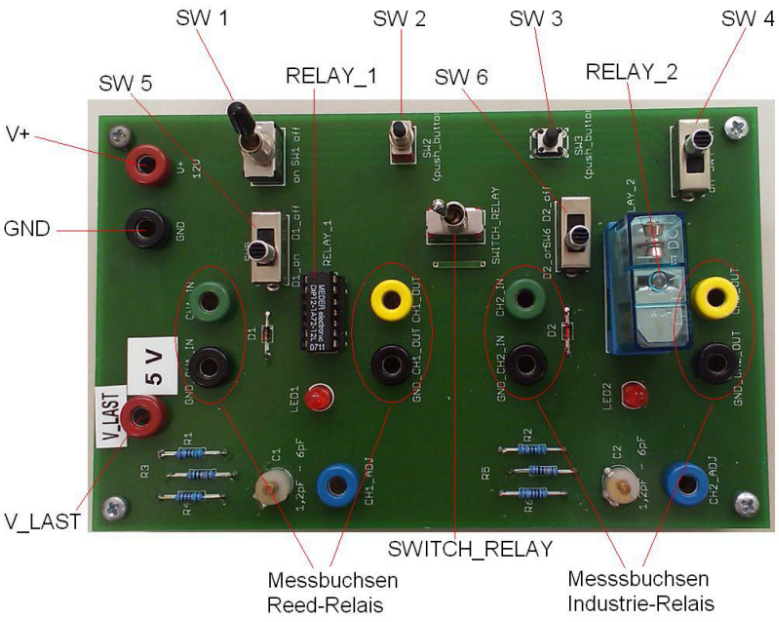


Abbildung 4: Verwendete Relaisplatine

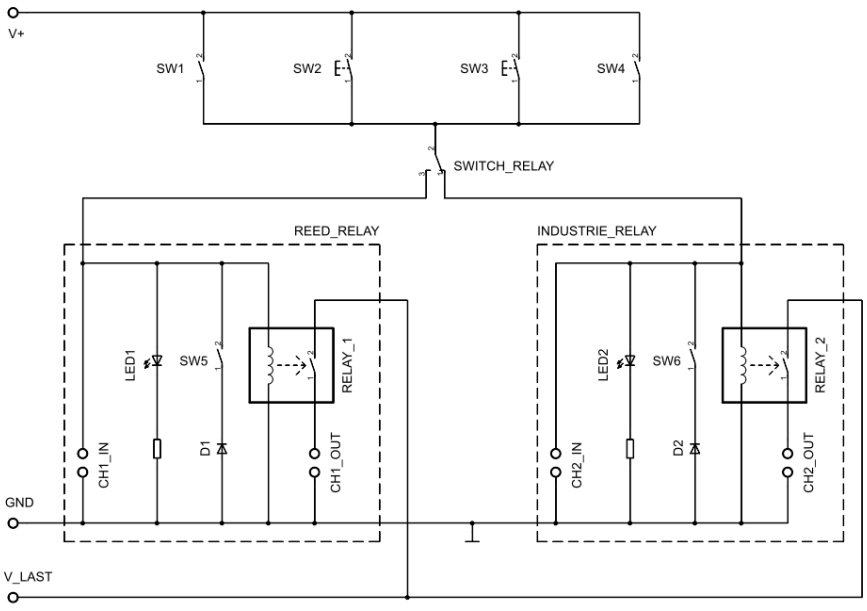


Abbildung 5: Schaltplan Laborplatine

Die Versorgungsspannung der Relaisplatine („Steuerspannung“) ist auf +12VDC einzustellen und an die Anschlüsse V+ und GND anzulegen (Abb. 8, oben links). Die Lastspannung ist auf +5VDC einzustellen und über die Anschlüsse V_Last und GND anzuschließen.

4.6.5 Messung des Kontaktwiderstands

Vor dem Fortfahren ist “Kernel -> Restart & Clear Output” empfehlenswert

[ ]:
```
import dmm                                              # Modul zum Steuern des DMM
 ↪HsKA-EIT
import serial
import time
import pandas as pd
import numpy as np
from IPython.display import clear_output
from termcolor import colored
import matplotlib.pyplot as plt


dmm.vPort = ''
dmm.cPort = ''

numSwitchEvents = 10
numMeasurements = 25
```

[ ]:
```
#vConn.close()
#cConn.close()
```

[ ]:
```
vConn = dmm.connectDMM(dmm.vPort)
cConn = dmm.connectDMM(dmm.cPort)
```

[ ]:
```
# Lesebefehl abschicken und Ergebnis absichtlich nicht abholen
vConn.write(b'')
cConn.write(b'')

# seriellen Eingangspuffer löschen
dmm.flushSerialInput(vConn)
dmm.flushSerialInput(cConn)
print("serial input buffer flushed")
```

[ ]:
```
val = dmm.readPrimary(vConn)
print("val = ",val)
val = dmm.readPrimary(cConn)
print("val = ",val)
```

[ ]:
```
vConn.write(b'')
time.sleep(1)
ack1 = vConn.readline()
time.sleep(1)
ack2 = vConn.readline()
print('ack1 = ', ack1)
print('ack2 = ', ack2)
```

[ ]:
```
vConn.close()
cConn.close()
```

Verdrahten Sie die Relais-Platine für spannungsrichtige Vierleitermessung. Stellen Sie dazu am Labornetzteil eine Strombegrenzung von ca. 100 mA ein. Die DMMs sind standardmäßig auf Auto Range geschaltet. Das ist für diesen Versuchsteil ungünstig. Es sollte vor der automatischen Messung jeweils manuell (an der Frontplatte des DMM) ein passender Messbereich eingestellt werden. Dazu muss das DMM vorübergehend wieder auf lokalen Betrieb geschaltet werden (Taste "Local").

Überprüfen Sie die Anzeigewerte, indem Sie das Relais anziehen lassen. Korrigieren Sie ggf. den Messbereich. Erst wenn alles richtig eingestellt ist, soll die automatische Messwertaufnahme gestartet werden.

[ ]:
```
# automatische Messwertaufnahme
result = dmm.measureRemote(numSwitchEvents, numMeasurements)
```

Die Messergebnisse sind in einem Pandas DataFrame abelegt und können über die Spalten- und Zeilenbezeichner adressiert werden. Dabei muss der Typ des Bezeichners beachtet werden. Hier ist der Spaltenbezeichner vom Typ String und der Zeilenbezeichner vom Typ Integer.

[ ]:
```
result
```

[ ]:
```
# Zugriff auf Spalte 'voltage_1' und Zeile 3
result['voltage_1'][3]
```

Jetzt werden die Widerstandswerte für jede Messreihe als Quotient aus Spannung und Strom berechnet und in einer Liste abgelegt.

[ ]:
```
# eventuelle frühere Instanzen von resistance werden gelöscht
#resistance.clear()
# resistance wird neu als Liste initialisiert
resistance = []
print(resistance)
```

[ ]:
```
for i in range(numSwitchEvents) :
    resistance.append(result['voltage_' + str(i+1)]/result['current_' +␣
 ↪str(i+1)])
print(resistance)
```

Für jede Messreihe wird jetzt der Mittelwert berechnet, es ergibt sich eine Liste der Mittelwerte.

[ ]:
```
resMean = []

for i in range(numSwitchEvents) :
    resMean.append(resistance[i].mean())
resMean
```

Analog für die Standardabweichungen:

[ ]:
```
resStd = []

for i in range(numSwitchEvents) :
    resStd.append(resistance[i].std())
resStd
```

[ ]:
```
x = []

for i in range(numSwitchEvents) :
    x.append(i)
x
```

[ ]:
```
# Mittelwert der Widerstandsmittelwerte
resMeanValue = np.mean(resMean)
resMedianValue = np.median(resMean)
print('mean = ', resMeanValue, 'median = ', resMedianValue)
```

4.7 Plotten der Messergebnisse und Standardabweichung

[ ]:
```
plt.plot(resMean, linestyle='None', marker='x')
plt.errorbar(x,y=resMean, yerr=resStd,linestyle='None')
plt.hlines(resMeanValue,x[0],x[numSwitchEvents-1],color='green',␣
 ↪label='Mittelwert')
plt.hlines(resMedianValue,x[0],x[numSwitchEvents-1],color='red', label='Median')
plt.show()
```

Kommentieren Sie die graphische Darstellung. Was lässt sich über den Kontaktwiderstand und das Messverfahren aussagen?

Für schnelle und besonders neugierige Studierende: Wiederholen Sie die halbautomatisierte Messung mit dem Kontakt des Reed-Relais. Reduzieren Sie die Strombegrenzung dazu aber auf ca. 20 mA. Wenn das mit dem Einstellknopf am Labornetzteil nicht gelingt, verwenden Sie einen Strombegrenzungswiderstand.