Hans-Joachim Berndt

Messen, Steuern und Regeln

mit

LibreOffice

Makros für serielle Schnittstellen

INHALT

1 EINLEITUNG

LibreOffice als Plattform bietet die Möglichkeit unabhängig von Betriebssystemen Dokumente, Tabellenkalkulationen und mehr zu erstellen. Dinge lassen sich damit auch automatisieren. Das Softwarepaket ist frei kopierbar und der Quelltext ist offen. LibreOffice liegt zurzeit in der Version 6 für einige Plattformen vor, seine Verwandten sind OpenOffice und dessen Vorgänger.

Abbildung 1-1: Start von LibreOffice auf Linux oder Windows

Diese Office-Version ist für folgende Plattformen verfügbar:

Linux
Windows
macOS
Android

In diesem Buch sind Anwendungen für Linux Rasberry Pi OS (Raspberry Pi Zero W), Android 9 (Smartphone), Windows 8 und 10 (Tablets) aufgeführt. Dabei unterscheiden sich die Installationen geringfügig, so dass es ab und zu zu kleineren Abweichungen kommen kann. Da die Versionen laufend aktualisiert werden, kann hierzu keine allgemeine Aussage gemacht werden. Dass hier der kleinste Linux-Rechner Raspberry Pi Zero W und ein Windows 10 Tablet in einer Reihe stehen, zeigt die Unabhängigkeit dieses Pakets von Betriebssystem, Version und möglichen Distributionen. Auch die Methoden auf externe Daten Zugriff zu nehmen sind überwiegend einheitlich und unabhängig. Alle Ausführungen gelten im Prinzip für alle Plattformen, wobei dies für Android noch eingeschränkt gilt.

Dieses Buch zeigt Anwendungen der Sprachen Basic und Python, um gewisse Ziele im Kontext des Titels zu erreichen. Die Vermittlung und Einführung dieser Sprachen ist nicht vorgesehen oder beabsichtigt. Auch wird an keiner Stelle der Anspruch erhoben diese Sprachen mit allen ihren Möglichkeiten zu verwenden, vielmehr dient Python in diesem Zusammenhang meist lediglich als Brücke zwischen Betriebssystem und dem integrierten Basic. Dennoch kann die Auseinandersetzung mit diesen Themen möglicherweise als Kristallisationspunkt zur Anwendung der Sprache Python für Basic-Programmierer dienen.

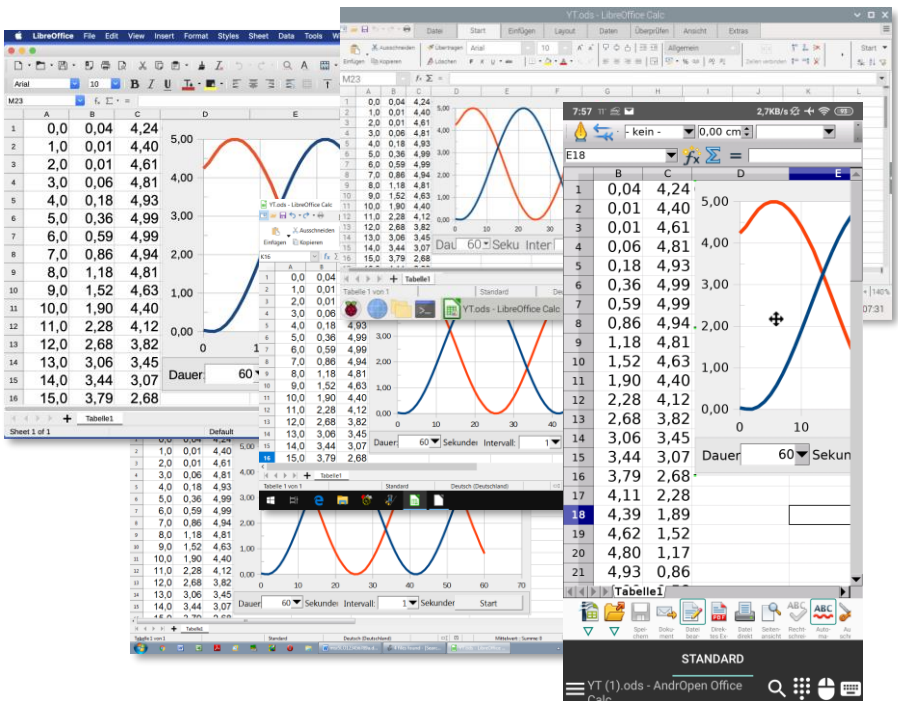

Abbildung 1-2: Plattformunabhängig, Basic, Python, Open Source…

2.1 Erstes Makro

In der vorigen Abbildung 2-1 wird ein neues Makro in das Dokument eingefügt, um einen ersten Test zu starten. Unter dem Menüpunkt *Extras/Makros Verwalten/LibreOffice Basic...* erreicht man den dargestellten Dialog. Mit dem Klick auf Standard vom Dokument *Unbenannt1* erzeugt die Schaltfläche *Neu* ein leeres Basic-Modul an dieser Stelle im Editor. Einen ersten Programmlauf erhält man, indem in die Zeile 4 ein

```
Print „Hallo Welt"
```

eingefügt wird. Damit besteht das erste Hauptmakro (Main) im Dokument aus der Ausgabe der bekannten Zeichenfolge. In LibreOffice Basic ist die Print-Ausgabe ein Meldungsfenster.

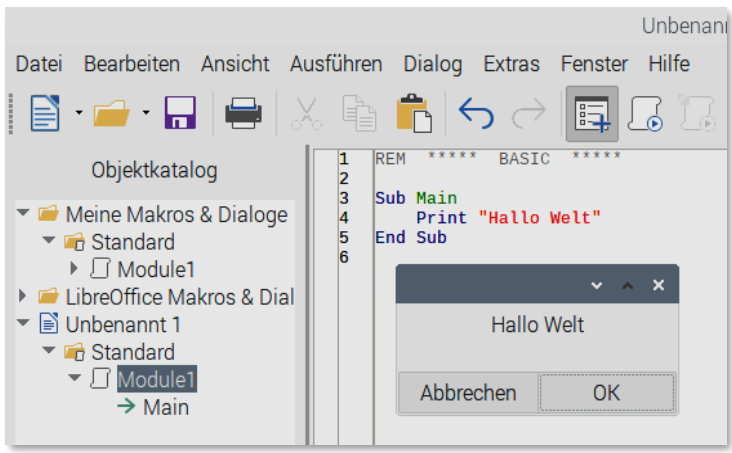

Abbildung 2-2: Erstes Makro im Dokument mit Quelltext und Ergebnis

2.2 Makros Aufzeichnen

LibreOffice Basic kann Makros aufzeichnen. Das ist nützlich, wenn man beim Arbeiten mit Office die Dinge automatisieren will, aber keinerlei Programmierkenntnisse besitzt. Die Aufzeichnung generiert Quelltext, der am Ende in ein Modul als Klartext eingefügt wird. Da dieses Office und sein Basic zwar stark an VBA erinnern, aber doch intern völlig anders aufgebaut sind, kann die Aufzeichnung der schnellere Weg sein eigene Makros zu erstellen, indem Aufzeichnungen nur etwas den eigenen Wünschen angepasst werden. Die Möglichkeit der Aufzeichnung ist in LibreOffice in den Voreinstellungen deaktiviert. Um das zu ändern, muss unter dem Menüpunkt *Extras/Optionen/Erweitert* der Haken bei *Makroaufzeichnung ermöglichen (eingeschränkt)* gesetzt sein. Nun erscheint diese Möglichkeit unter dem Menü *Extras/Makros/Makros Aufzeichnen* als erster Eintrag des klassischen Menüs.

Das erste aufgezeichnete Makro soll eine Zeichenfolge in das Dokument schreiben und anschließend durch Betätigung der Eingabetaste in eine neue Zeile springen, so dass bei mehrmaligem Aufruf der Aufzeichnung mehrere Zeilen untereinander erscheinen. Der Start der Aufzeichnung erfolgt vom Dokument aus über das Menü. Es erscheint ein kleines Fenster mit weißer Schrift auf hellgrauem Untergrund. Durch geschicktes Drehen des Bildschirms kann man erahnen, was der Programmierer als Text vorgesehen hat. Sind die Worte mit der Eingabetaste eingegeben, kann durch Bestätigung der Schaltfläche im PopUp-Fenster die Aufzeichnung beendet werden. Basic fragt dann, ob das Modul überschrieben werden soll. Das Ergebnis sind einige Zeilen Quelltext, die hier klein gelistet sind, damit Zeilenumbrüche das Listing weniger zerstören.

2.9 Messwerte in Dialog, Tabelle und Diagramm

Am Ende dieses Kapitels über Writer-Makros folgt das Einfügen der Messwerte aus einem Dialog heraus in ein Dokument. Dieser Dialog zeigt die einlaufenden Messwerte in Echtzeit, die sich auf Knopf- oder Tastendruck übernehmen lassen. Der Zustand eines Markierungsfeldes (Checkbox) entscheidet darüber, ob die Daten untereinander als Text, oder nebeneinander in eine vorhandene zweispaltige Tabelle einzutragen sind. Die Tabelle ist dann mit einem Diagramm verknüpft, so dass die eingehenden Messdaten auch dort quasi in Echtzeit dargestellt werden, falls die Rechnerkapazität ausreicht. Zusätzlich enthält der Dialog ein Kombinationsfeld, welches die verschiedenen Schnittstellen anzeigt.

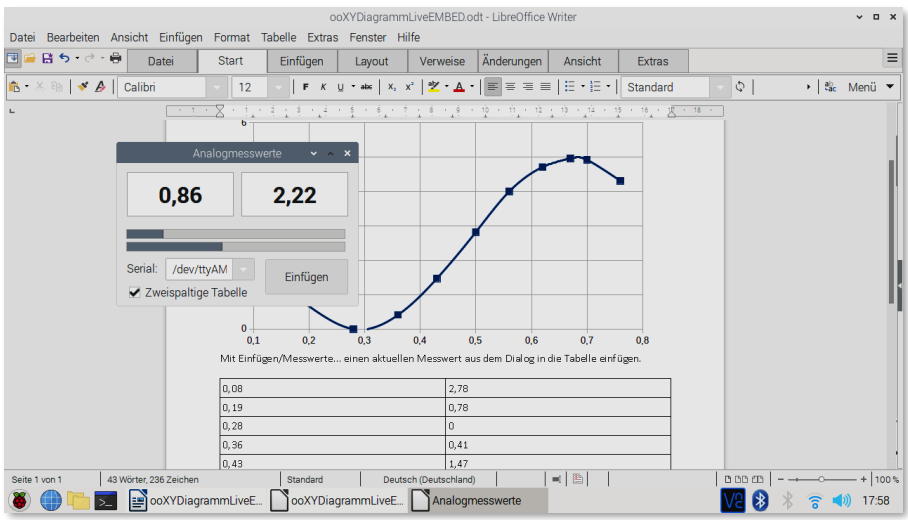

Abbildung 2-11: Dynamischer Dialog Tabelle mit Diagramm (Raspberry Pi OS)

Die Schnittstellenauswahl ist hier nur Platzhalter und soll zeigen, wie sich Dialogelemente ansprechen lassen. Der Dialog an sich ist eine Erweiterung des einfachen Dialogs aus Abschnitt 2.3.

3.2 Zeitroutinen und Geschwindigkeit

Bei Messungen spielt Zeit oft eine entscheidende Rolle, da viele Prozesse von der Zeit abhängig sind und diese Größe dann in Diagrammen auf der X-Achse erscheint. Die abhängige Größe ist dann auf der Y-Achse aufgetragen. Dies ist im einfachsten Fall eine simulierte Schwingung, wie in Abschnitt 2.5, die mit der Funktion *AIN* aufgerufen werden kann. Zur Erstellung von Zeitdiagrammen sind in diesem Kontext Zeiten im Millisekundenbereich erforderlich. Die drei Zeitroutinen *Delay*, *TimeInit* und *TimeRead* für Verzögerungen und Zeitmessungen bilden den Grundstock. Diese Routinen greifen meist auf den rechnerinternen Ticker zurück, der je nach System eine Auflösung im Millisekundenbereich aufweist. Solche Routinen lassen sich in dem jeweiligen Basic bequem nachbilden, so dass der Quelltext zwischen den „Office-Basics“ sogar kompatibel bleibt.

Tabelle 3-2: Zeitfunktionen für kurze Messzeiten

Delay	Verzögerung um eine gewisse Anzahl von Millisekunden
TimeInit	Zeitinitialisierung, vergleichbar mit einem Reset
TimeRead	liefert die verstrichene Zeit seit *TimeInit* in ms
Wait	Eingebaute Verzögerung im Millisekunden-Bereich
Delay2	Verzögerung auf Basis von *Timer*

Unter LibreOffice Basic u. a. gibt es zwei Aufrufe zur Kurzzeitbestimmung. Das maschinennahe *getsystemticks* liefert die Ticks des Rechners und die Routine *Timer* die Sekunden seit Mitternacht. Mit diesen beiden Aufrufen lassen sich diese drei obigen Zeitroutinen nachbilden. *Delay 1000* wartet zum Beispiel eine Sekunde. Beim Aufruf wird die aktuelle Zeit in Millisekunden in einer Variablen *t0* gespeichert. Dann läuft eine *While*-Schleife solange die aktuelle Zeit kleiner als die Summe von gespeicherter Zeit und Wartezeit ist. Eine solche einfache Schleife würde bei langen Zeiten den Rechner blockieren. Mit der *DoEvents*-Anweisung reagiert der Rechner weiter auf Eingaben von Maus und Tastatur. Auch

4.3 LibreOffice Basic ruft Python

Python Makros lassen sich auch von Basic aus aufrufen. Damit eröffnet sich die Möglichkeit externe Funktionen aus Python-Bibliotheken in LibreOffice mittels Basic zu steuern. Dieses System ist mit dem DLL-Prinzip unter Windows vergleichbar. Eine Python-Funktion kann mit der *return* Anweisung Werte zurück liefern, die sich dann in Basic weiter verwenden lassen, als kämen sie von einer entsprechenden Basic-Funktion.

Zunächst soll ein fester Zahlenwert von einer Python-Bibliothek, also einer Datei mit der Endung *py*, mittels Python-Funktion *def* als Makro abgerufen werden. Die Bibliothek ist sehr klein und besteht aus einer einzigen Funktion bzw. einem Makro mit zwei Zeilen.

```
def ZweiMalNeun(*args):
    return 2*9
```

Es sind keine weiteren Bibliotheken notwendig. Der Funktionsname ist *ZweiMalNeun* und der Rückgabewert ist das Ergebnis der Berechnung von zweimal neun. Das Argument **args* ist ein Platzhalter für mögliche Argumente, die hier nicht benötigt werden. Wenn die Position des Doppelpunktes und die vier Leerzeichen zur Einrückung stimmen, sollte Python keine Fehler melden. Eine Datei mit dem Namen *neu.py* kann diesen Programmtext bzw. dieses Skript im Benutzerverzeichnis enthalten.

Nun folgen die Vorbereitungen in LibreOffice Basic. Dazu ist es erforderlich, dass eines der Office Programme ausgeführt wird und der Basic-Editor unter *Extras/Makros/Makros bearbeiten/LibreOffice Basic* Makros entgegen nehmen kann. Um eine Python-Funktion bzw. Makro in der Datei *neu.py* mit Namen aufzurufen, sind folgende Zeilen in einem neuen Basic *Module* notwendig.

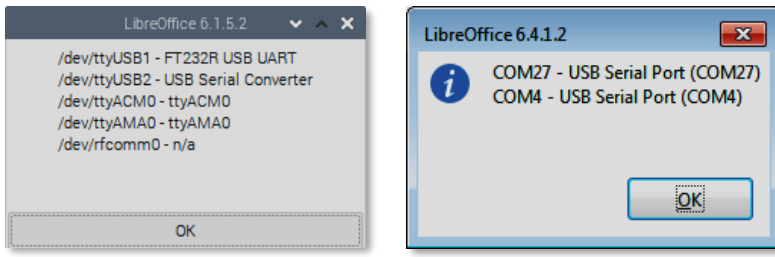

Abbildung 5-3: Aktuell verfügbare serielle Schnittstellen verschiedener Plattformen

```
Sub TestCOM
 MsgBox Replace(getcoms,chr(9),chr(13)+chr(10))
End Sub
```

Listing 5-3: Auflistung der verfügbaren seriellen Schnittstellen

Mit Hilfe von Formularsteuerelementen kann in einem Tabellenblatt eine komfortable Schnittstellenauswahl erfolgen. Dies ist dann nützlich, wenn Geräte an den Schnittstellen wechseln und manche Betriebssysteme immer wieder neue Bezeichnungen generieren. Da es möglich ist eine Schnittstelle geöffnet zu lassen, sind drei Bedienelemente ausreichend. In einer Listenauswahl trägt ein Makro die aktuell verfügbaren Namen der Schnittstellen ein. Dieses Makro ist mit einer Schaltfläche *Aktualisieren* verknüpft. Bei Selektion des Geräts in der Liste kann dann über eine Schaltfläche *Öffnen* die Schnittstelle geöffnet werden. Mit der *Print*-Anweisung erfolgt kurz vor dem Aufruf von Python noch die Anzeige des kompletten Parameterstrings für *Opencom*, damit eventuell noch ein Abbruch erfolgen kann. Das Listenfeld ist entsprechend den Zellenangaben im Listing verknüpft.

Die Routine *CellCom* trägt die gefundenen Verbindungen in das Tabellenblatt im angegebenen Bereich ein, womit das Listenfeld verknüpft ist. Die Auswahl kann dann in der Ausgabezelle M1 abgeholt werden. Diese Aufgabe übernimmt *OpenSelected* und sucht die korrekte Bezeichnung des Schnittstellennamens anhand eines Leerzeichens. Das funktioniert auf den hier benutzen Plattformen. Die *Print*-Anweisung zeigt das Ergebnis zur Überprüfung an.

```
Sub CellCom 'Eintrag der Schnittstellen in Tabelle
 RangeClearString "N1:N20"
```

```
End Sub

Sub gpsZeit
 opencom "COM11,9600,8,N,1"
 udpOpen "192.168.178.34",4712
 Do
  r = readline
  If Left(r,6)="$GPRMC" Then cell 1,1,Mid(r,8,6)
  udpsend r
 Loop Until False
End Sub
```

Listing 5-6: GPS-Daten in LibreOffice über USB

Die aktuelle Uhrzeit in UTC steckt unter anderem in dem Datenpaket, welches mit der Zeichenfolge *$GPRMC* eingeleitet wird. Ab der 8. Stelle steht dann die sechsstellige Uhrzeit. Die Routine *gpsZeit* überprüft diese Voraussetzungen und zeigt bei Erfolg das Ergebnis in Zelle A1 an. Die zwei UDP-Aufrufe kommen aus Abschnitt 7.3 und leiten die Messdaten weiter an ein Gerät mit der angegebenen IP und dem angegebenen Port. Verknüpft man diese Zelle mit einer mittleren Zelle des Tabellenblatts durch die Formel *=A1* und formatiert diese Zelle mit großer Schrift, so entsteht eine GPS-Uhr im Tabellenblatt.

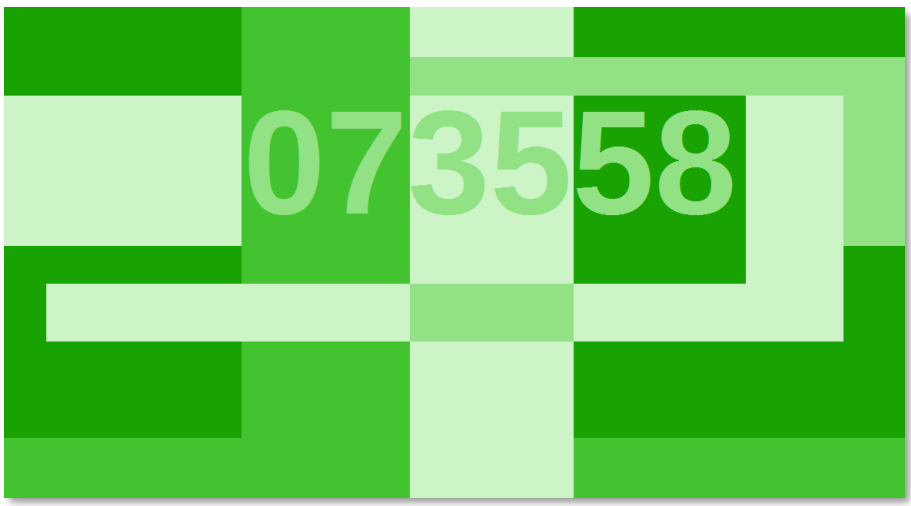

Abbildung 5-14: GPS-Uhrzeit in LibreOffice Calc live vom Satelliten

```
   Sendline "digitalWrite(13,0)"
   Delay(500)
 Wend
End Sub
```

Listing 5-10: Arduino Blink in LibreOffice Basic

Der Aufruf *Delay* kommt aus Kapitel 3 und kann durch das eingebaute *Wait* ersetzt werden. Eine Funktion *ai* zum Lesen eines Analogeingangs wäre dann:

```
Function ai(ein)
 SendLine "analogRead("+Str(ein)+")":ai = ReadLine
End Function

Sub arduino6xAnalog
 For Zeile = 1 To 20
  For Eingang = 0 To 5
   Cell Zeile,Eingang+1,ai(Eingang)
  Next Eingang
 Next Zeile
End Sub
```

Listing 5-11: Sechs Analogeingänge des Arduino im Tabellenblatt

Mit *arduino6xAnalog* werden 20 Messwerte aller 6 Analogeingänge in ein Tabellenblatt eingetragen, dabei sind ein Eingang an 5 und ein anderer an 3,3 Volt gelegt – die restlichen Eingänge sind nicht angeschlossen.
Die Bezeichnung *ai* für AnalogIn stammt aus ESPBASIC [4] und macht genau dies mit der Syntax *io(ai)*. Darum interpretiert der Arduino auch diese Befehle. Noch kürzer funktioniert der Aufruf mit nur *a0.* Die folgenden drei Kommandos sind alle gleichwertig.

6.7 Kennlinien mit drei USB-Geräten

In diesem Kapitelabschnitt zur seriellen Schnittstelle erfolgt eine Überprüfung des Konzepts, indem drei Geräte an jeweils einer eigenen USB-Schnittstelle gleichzeitig eine automatisierte Messung durchführen. Die Steuerung erfolgt über LibreOffice Basic, was seinerseits auf Python zurückgreift, um plattformunabhängig Messung mit realer Hardware über das serielle Protokoll und deren Leitungen zu ermöglichen.

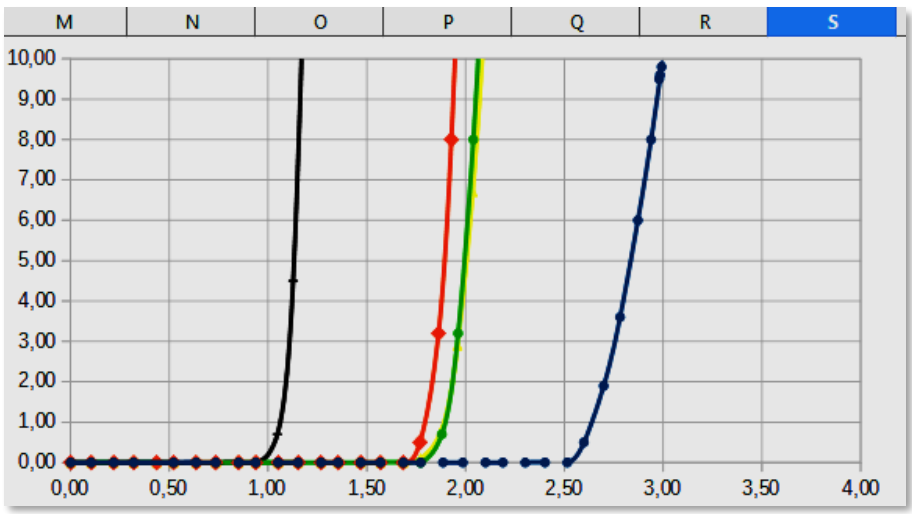

Abbildung 6-12: Ergebnis einer automatisierten Kennlinienaufnahme

Mit den zwei weiter oben behandelten Multimetern und der steuerbaren Spannungsquelle des vorigen Abschnitts entsteht ein Aufbau zur automatischen Kennlinienaufnahme. Da die verschiedenen Farben von LEDs durch unterschiedliche Materialien auch zu unterschiedlichen Durchlassspannungen führen, soll mit Volt- und Amperemeter das Bauelement klassisch messtechnisch untersucht werden. Dabei wird die Kennlinie bis zu einem Durchlassstrom von etwa 10 mA automatisch in einem Tabellenblatt mit Diagramm aufgenommen. Die Geräteliste steht somit fest.

6.9 Digitale Messbrücke mit automatischem Abgleich

Mit einem digitalen Potentiometer lässt sich eine unabgeglichene Messbrücke automatisch per Software abgleichen. Anhand des digital eingestellten Widerstandswertes im einfach zu berechnenden abgeglichenen Zustand kann dann auf die Änderung des zu untersuchenden Brückenwiderstandes geschlossen werden. Dazu ist es erforderlich die Brückenspannung zu erfassen, die je nach Zustand der Brücke auch negativ sein kann. Mit einem Digitalmultimeter nach Abschnitt 5.2 kann man diese Spannung komfortabel erfassen und auswerten.

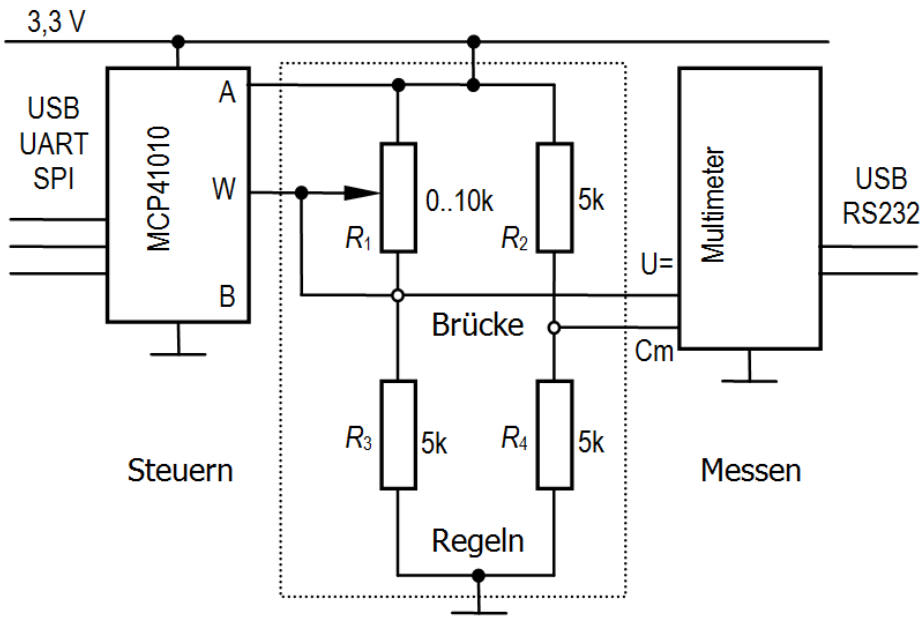

Abbildung 6-19: Automatische Messbrücke, schematisch

Wem Messbrücken bekannt sind, kann die folgenden allgemeinen Ausführungen dazu überspringen und gleich zum Aufbau übergehen.

8 USB DIREKT

Der Universal Serial Bus USB als Schnittstelle kam erstmals unter Windows 98 zum Einsatz und ist seit dieser Zeit allgegenwärtig. Inzwischen bei Version 3 angekommen ist er die Verbindung fast aller Peripheriegeräte zum PC. Festplatten, Kameras, Mäuse, Tastaturen, Audio, aber auch Geräte aus dem Bereich MSR wie Multimeter, Oszilloskop und Interfaces sind hier als Beispiele zu nennen. Für den Anwender einfach zu benutzen, gestaltet sich der Umgang auf Programmierebene eher schwierig und komplex. Für ein umfassendes Verständnis auf dieser Ebene sei auf entsprechende Literatur verwiesen.

Abbildung 8-1: USB-Tastatur als HID-Gerät und Ansteuerung der drei Status-LEDs

Dieses Kapitel enthält lediglich eine vereinfachte Beschreibung, die den direkten Umgang mit dieser komplexen Schnittstelle in LibreOffice Basic über Python verständlich machen soll. Die Vereinfachung ist so hoch, dass zwei Programmzeilen ausreichen, um zum Beispiel ein reines USB-Computerinterface mit zwei Analogeingängen, acht Digitaleingängen und acht Digitalausgängen komplett anzusprechen. Dieser Weg benutzt den direkten Zugriff auf die USB-Schicht des Betriebssystems unter Umgehung eventuell erforderlicher Gerätetreiber des Herstellers.

8.1 Kernel-Gerätetreiber

Mit dem Universal Serial Bus USB kam das Plug&Play-Prinzip von Geräten. Wird ein USB-Gerät mit einem PC verbunden, so sieht der PC an den Datenleitungen eine Pegeländerung und sendet eine Bytefolge fester Struktur an das unbekannte Gerät. Das Gerät antwortet nach den USB-Richtlinien mit 18 Bytes, die erste Informationen des Gerätes enthalten. Diese 18 Bytes enthalten immer zwei Zahlenwerte, die dem Hersteller mit einer *VendorID* und dem Produkt mit einer *ProductID* zugeordnet sind, die üblicherweise in hexadezimaler Schreibweise erscheinen. Anhand dieser beiden Zahlenwerte versucht das Betriebssystem den entsprechenden Treiber zu aktivieren oder zu installieren.

Einige USB-Geräte arbeiten nach den sogenannten HID-Interface-Richtlinien und benötigen keine speziellen Herstellertreiber. Es sind Human Interface Devices, wie Mäuse, Tastaturen, Gamepads und andere, für die das Betriebssystem jeweils vereinheitlichte Treiber bereithält. Nach der Verbindung eines solchen Geräts ist dieses sofort einsatzbereit. Auch mehrere gleiche Geräte dieser Art lassen sich parallel betreiben. Das Betriebssystem selber kümmert sich um entsprechende Ansteuerungen.

Bekannte Treiber bringen die meisten Betriebssysteme bereits mit, andere erfordern eine eigene Installation. Ein solcher Treiber für ein bestimmtes Gerät, der unter Windows die Endung **.sys* trägt, ist meist vom Hersteller mit Hilfe eines von Microsoft vorgeschriebenen Developmentkits aufwändig entwickelt und kann üblicherweise von der entsprechenden Internetseite geladen werden. Er ist der Mittler zwischen dem Windows-USB-System und den Geräteaufrufen.

```
Sub findkeyboard
 If Instr(info,"Keyboard") > 0 Then
  usb = list
  ids = Split(usb(0),":")
  vid = ids(0): pid=ids(1)
  If Not init(int(vid),int(pid)) Then
   MsgBox "Initialisierung gescheitert: Einmal Gerät ab-/anstecken?"
   End
  EndIf
  MsgBox "OK, Tastatur entdeckt und initialisiert!"
 Else
  MsgBox "Keine Tastatur vermutlich."
 End If
End Sub

Sub blink ' Keyboard LED-Zähler
 For i = 0 To 7
  leds i
  Wait 200
 Next
End Sub

Sub Tastatur
 While True
  m = keys
  For x = Lbound(m) To Ubound(m)
   Cell 1,x+1,m(x)
  Next
  Wait 30
 Wend
End Sub

Sub Notaus
 End
End Sub
```

Die Python-Bibliothek befindet sich im Anhang *Listing h)*.

9.4 Kurvendiskussion mit VBA

In der VBA-Syntax erscheinen die Makro Listings subjektiv kürzer und übersichtlicher. Das lädt dazu ein kleinere mathematische Spielereien zu testen. Eine Parabel und deren Ableitung kratzt ein wenig an der Differentialrechnung, wenn die erste Ableitung von x^2 zu $2x+1$ numerisch über die Steigung anschaulich ermittelt wird.

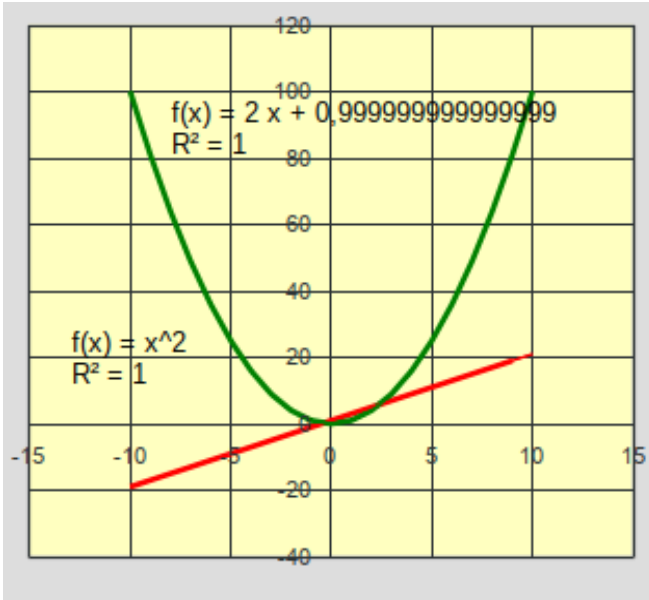

Abbildung 9-7: Mathematik im Tabellenblatt

Das Makro *Parabel* füllt die Spalten A und B eines Tabellenblatts mit VBA-Support mit den Zahlen von -10 bis + 10 in A und der Potenz in Spalte B. Mit dem Diagrammassistenten wird die Kurve der Parabel in einem PunktXY-Diagramm (Scatter) dargestellt. Im Makro *Ableitung* erfolgt die Berechnung der Steigung zwischen zwei Parabelpunkten und das Ergebnis landet in der dritten Spalte C. Durch das Editieren des Diagramms und dem Hinzufügen von Datenreihen in der Datenquelle ist es möglich die dritte Spalte der Ableitung im selben Diagramm einzutragen.

$y = x^2 \quad y' = 2x + 1$

```
Option VBASupport 1
Sub Parabel()
```

```
  "$cls"+LAN).invoke(array(),array(),array()):end sub

Sub OLEDreset
 ThisComponent.getScriptProvider().getScript(SCR+file+_
  "$reset"+LAN).invoke(array(),array(),array()):end sub

Sub MainOledTest
 OLEDreset
 While True
  OLEDcls
  OLEDtext 30, 2, "MSR LibreOffice"
  OLEDtext 72,16, Time()
  For x = 0 To 127 Step 10
   OLEDline 0,0,x,63
  Next x
 'OLEDrect 0,0,127,63
  OLEDline 0,63,120,63
  Wait 5000
 Wend
End Sub
```

Das Ergebnis ist eine kleine wiederholte grafische Animation mit einem kleinen Text auf einer kleinen Anzeige an einem Kleinstcomputer, der die aktuelle Uhrzeit als Zeichenkette neben der Grafik darstellt.

Abbildung 10-6: LibreOffice Basic schreibt und zeichnet auf ein Oled-Display

Mit dieser Uhr endet dieses Buch über MSR mit LibreOffice.