Fachhochschule Düsseldorf

Fachbereich: Maschinenbau und Verfahrenstechnik

Diplomarbeit

Christian Berger

Matrikel-Nr.: 383 000

Objektorientierte Programmierung einer aeroakustischen Messdatenerfassung

Institut für Strömungsmaschinen IFS

Prof. Dr. Ing. F. Kameier

Dipl. Ing. C. Haukap

Düsseldorf, im Dezember 2001

Vorwort

Diese Diplomarbeit entstand an der Fachhochschule Düsseldorf am Institut für Strömungsmaschinen.
Ich möchte mich an dieser Stelle bei meinem sehr geehrten Prof. Dr. Ing. F. Kameier für die Anregung zu dieser Diplomarbeit bedanken.
Besonders möchte ich mich für die Betreuung, die mir während der gesamten Arbeit gewährt wurde, bei Prof. Dr. Ing. F. Kameier, Dipl. Ing. C. Haukap, sowie allen Mitarbeitern des Instituts für Strömungsmaschinen bedanken.
Ebenso möchte ich mich bei meiner Frau für die Unterstützung und vor allem für die Geduld bedanken.

Inhalt

1 Einleitung

Die vorliegende Diplomarbeit behandelt die objektorientierte Programmierung eines aeroakustischen Messdatenerfassungssystems. Dazu sollten zum einen die bekannten Methoden und Verfahren zur Prüfstandsmessung an Ventilatoren in ein Programm umgesetzt werden, zum anderen sollte das Programm auch in der Lage sein, zeitgleich zur Aufnahme der aerodynamischen Daten, Soundfiles an ausgewählten Messpunkten aufzuzeichnen. Die Software entstand in Zusammenarbeit mit der Firma **Pollrich Ventilator, Mönchengladbach** als Ersatz für ein bestehendes, aber veraltetes Messdatenerfassungssystem an einem Normprüfstand für Radialventilatoren.

Als Messgeräte stehen das Agilent 34970A Messdatenerfassungs- / Schaltsystem, mit den daran angeschlossenen Geräten zur Aufnahme von Druck und Temperatur, und die Soundkarte des PC's (TerraTec DMX Xfire 1024) zur Verfügung. Die Programmierung sollte entweder mit dem Softwarepaket DasyLAB oder LabVIEW erfolgen, die beide für die objektorientierte Programmierung von Messsoftware entwickelt wurden.

Mit den so gewonnen Daten soll zum einen die Kennlinie eines Radialventilators berechnet werden, zum anderen soll eine Datenbank geschaffen werden, die es ermöglicht auf Grundlage dieser Daten ein Kennfeld vorherzusagen oder einen Ventilator aerodynamisch auszulegen.

Bei der Programmierung und Dokumentation sollte besonderer Wert darauf gelegt werden, dass evtl. in Zukunft notwendige Änderungen oder Erweiterungen auch ohne die aktive Mithilfe des Programmierers durchgeführt werden können.

2 Grundlagen

In den folgenden Kapiteln werden zunächst die Grundlagen der aerodynamischen Berechnungen an Ventilatoren mit Fördermedium Luft zusammengetragen. Ein anderes Medium außer Luft wird nicht betrachtet.

2.1 feuchte Luft

Trockene Luft besteht zu 78,04 Mol-% aus Stickstoff, zu 21,04 Mol-% aus Sauerstoff, zu 0,93 Mol-% aus Argon und zu 0,03 Mol-% aus Kohlendioxyd. Die atmosphärische Luft kann man als Zweistoffgemisch betrachten, bestehend aus trockener Luft und Wasser, das in dampfförmiger und fester Form vorliegen kann. Man bezeichnet dieses Gemisch auch als ***feuchte Luft***. Die trockene Luft betrachtet man als einheitlichen Stoff. Da der Gesamtdruck bei Zustandsänderungen fast immer in der Nähe des Atmosphärendrucks liegt, kann man die feuchte Luft aus trockener Luft und Wasserdampf als ein Gemisch idealer Gase ansehen **[1]**.

Diese Betrachtung ist für Prüfstandsmessungen an allen Ventilatoren wichtig, die das Medium Luft fördern, da sich mit der Luftfeuchte auch die Dichte ändert und u.a. der Volumenstrom von der Dichte abhängt, etc..

<u>Berechnung der Dichte von feuchter Luft:</u>

allgemein gilt die Zustandsgleichung für ideale Gase:

$$p \cdot V = m \cdot R \cdot T \qquad \text{oder} \qquad p = \rho \cdot R \cdot T \tag{1}$$

wobei die Gaskonstante für die feuchte Luft wie folgt zu bestimmen ist **[2]**:

$$R_{fL} = \frac{R_L}{1 - 0{,}377681 \cdot \frac{p_D \cdot \varphi}{p}} \qquad (p_D \text{ ist der Dampfdruck des Wassers}) \tag{2}$$

In Gleichung (2) ist darauf zu achten, dass sich der Druck p in der Regel aus dem barometrischen Druck und dem statischen Druck der Strömung zusammensetzt, z.B. am Austritt des Ventilators ist $p = \Delta p_A + p_B$.

Der zur Lufttemperatur gehörende Dampfdruck p_D des Wassers kann anhand der Dampfdruckkurve ermittelt werden. Diese Kurve ist z.B. in *Bohl, W.: Technische Strömungslehre*, **[3]** auf den Seiten 269 bis 280 in tabellarischer Form gegeben und kann durch ein Polynom 5. Grades mit sehr guter Näherung approximiert werden:

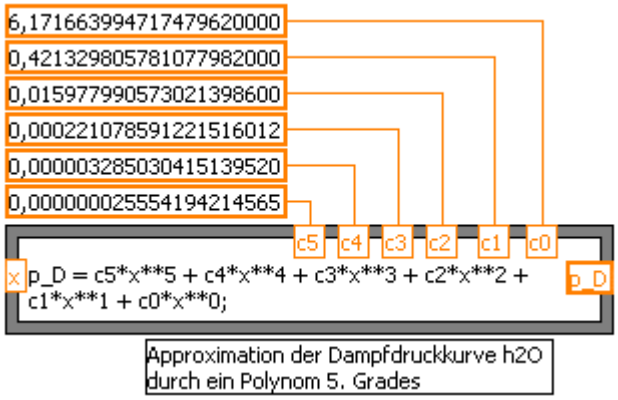


Bild 1: Auszug aus dem LabVIEW Code zur Berechnung des Dampfdruckes bei einer gegebenen Temperatur (x) in °C

Somit lässt sich die Dichte der feuchten Luft wie folgt bestimmten **[5]**:

$$\rho = \frac{p}{R_{fL} \cdot \left(t_{Luft} + 273{,}15\,K\right)} \tag{3}$$

Die Berücksichtigung von feuchter Luft als Fördermedium erfolgt, weil der Prüfstand der Fa. Pollrich Ventilator dafür eingerichtet ist. Wie die folgenden Grafiken zeigen, ergibt sich in dem für Niederdruckventilatoren üblichen Temperaturbereich eine Differenz zwischen dem nach Gleichung (3) berechneten Wert der Dichte und dem mit dem idealen Gasgesetz ermittelten Wert von ca. 1%. In Anbetracht der Tatsache, dass der Ventilator als hydraulische Strömungsmaschine angesehen wird und dabei das Medium (vereinfachender Weise) als inkompressibel betrachtet wird, ergibt sich aus dieser Berechnungsmethode keine technisch relevante Verbesserung der Genauigkeit erzielt.

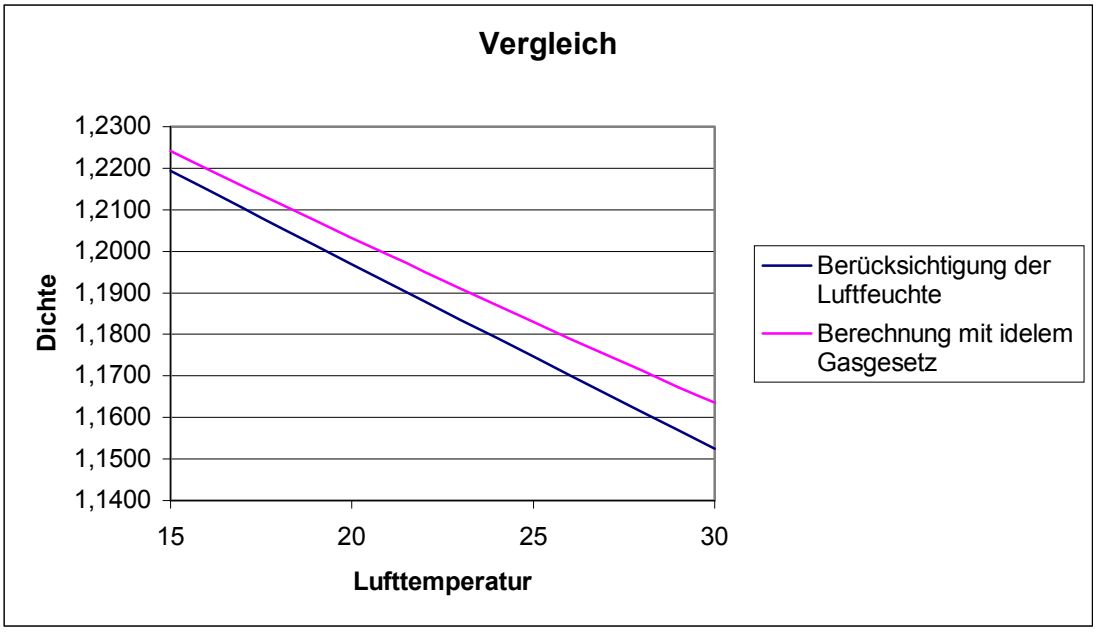


Bild 2: Vergleich zwischen dem nach Gleichung (3) berechneten Wert der Dichte und dem mit dem idealen Gasgesetz ermittelten Wert

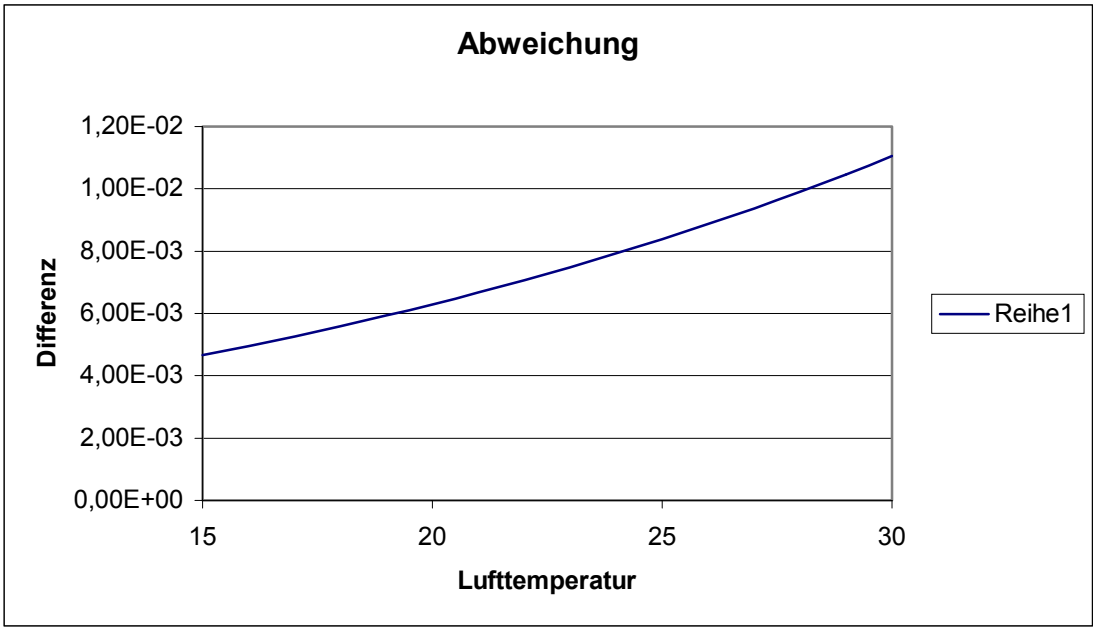


Bild 3: Differenz zwischen dem nach Gleichung (3) berechneten Wert der Dichte und dem mit dem idealen Gasgesetz ermittelten Wert

2.2 *Durchflussmessung von Fluiden mit Drosselgeräten*

Üblicherweise erfolgt die Messung des Durchflusses bei Ventilatorprüfständen mit den Drosselgeräten Blende oder Düse. Die genauen Berechnungsverfahren sind in der Norm DIN EN ISO 5167-1 **[4]** festgelegt. Die derzeit gültige Fassung ist die Ausgabe von 1998, die die Ausgabe von 1995 in der Berechnung des Durchflusskoeffizienten C bei der Blendenmessung korrigiert. Um eine Kompatibilität zu älteren Messergebnissen und älteren Programmen sicherzustellen wurden in dem neu erstellten Programm beide Versionen berücksichtigt und werden darum auch an dieser Stelle behandelt.

2.2.1 Blendenmessung

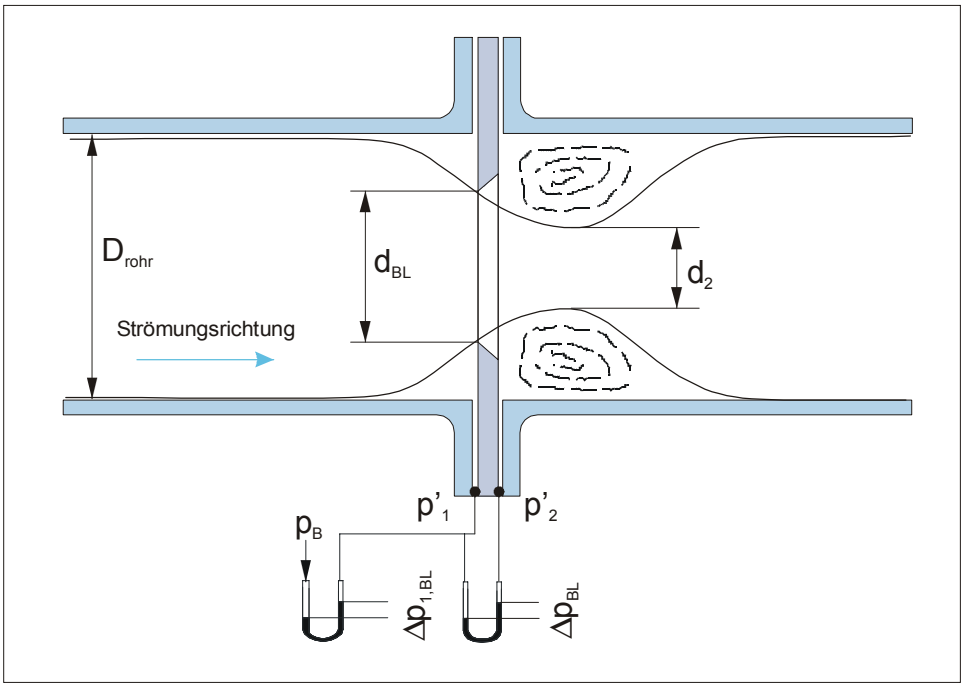


Bild 4: Prinzipskizze Blendenmessung

Bild 2 zeigt eine Prinzipskizze der Blende, des Strömungsverlaufs und der Drücke. Die Blende als unstetige Querschnittsverengung bewirkt eine starke Strahlkontraktion. Der Druck p_1' liegt über dem statischen Druckniveau in der Rohrleitung, die eingebaute Blende hat eine Druckabnahme und einen Druckverlust zur Folge : $p_1' > p_2'$. Zur Ermittlung des Massen- bzw. Volumenstroms wird die Druckänderung $p_1' - p_2'$ („Wirkdruckdifferenz") gemessen **[6]**.

Die rechnerische Ermittelung des Massenstroms erfolgt durch iteratives Lösen des folgenden Gleichungssystems:

$C = f(\beta; Re_D; L_1; L'_2)$ $\quad Re_D = f(q_m; D_{Rohr}; \eta)$ $\quad q_m = f(C; \beta; \varepsilon; d_{BL}; \rho_{BL}; \Delta p_{BL})$

<u>Bestimmung des Massenstroms mittels einer Messblende</u>
<u>gem. DIN EN ISO 5167-1 von 1995</u>

Durchflusskoeffizient C

$$C = 0{,}5959 + 0{,}0314 \cdot \beta^{2,1} - 0{,}1840 \cdot \beta^{8} + 0{,}0029 \cdot \beta^{2,5} \cdot \left(\frac{10^6}{Re_D}\right)^{0,75}$$
$$+ 0{,}0900 \cdot L_1 \cdot \frac{\beta^4}{(1-\beta^4)} - 0{,}0337 \cdot L'_2 \cdot \beta^3$$

Hierbei bedeuten:

$\beta = \frac{D_{BL}}{D_{Rohr}}$	Durchmesserverhältnis von Blende zu Rohr (< 1 !)
Re_D	Reynoldszahl bezogen auf den Rohrdurchmesser
$L_1 = \frac{l_1}{D_{Rohr}}$	das Verhältnis des Abstandes der Druckentnahme im Einlauf von der Blendenstirnseite zum Rohrdurchmesser
$L'_2 = \frac{l'_2}{D_{Rohr}}$	das Verhältnis des Abstandes der Druckentnahme im Auslauf von der Blendenrückseite zum Rohrdurchmesser (L'_2 bedeutet den Bezug auf den Abstand von der Blendenrückseite an, L_2 würde sich auf den Abstand von der Blendenstirnseite beziehen)

Unterscheidung von D–, $\frac{D}{2}$ –, Flansch– und Eck–Druckentnahmen:

<u>Eck – Druckentnahme:</u>

$L_1 = L'_2 = 0$

<u>D– und $\frac{D}{2}$ – Druckentnahme:</u>

$L_1 = 1$, $L'_2 = 0{,}47$

<u>Flansch – Druckentnahme:</u>

$L_1 = L'_2 = \frac{25{,}4}{D_{Rohr}}$

Expansionszahl ε

$$\varepsilon = 1 - \left(0{,}41 + 0{,}35 \cdot \beta^4\right) \cdot \frac{\Delta p_{BL}}{\kappa \cdot (p_B + \Delta p_{1,BL})}$$

Die Expansionszahl ε berücksichtigt die Kompressibilität des Fluids. Bei langsamen Strömungen oder bei geringen Druckerhöhungen kann die Strömung jedoch als inkompressibel betrachtet werden, d.h. ε = 1.

Massenstrom q_m

$$q_m = \frac{C}{\sqrt{1-\beta^4}} \cdot \varepsilon \cdot \frac{\pi}{4} \cdot d_{BL}^2 \cdot \sqrt{2 \cdot \rho_{BL} \cdot \Delta p_{BL}}$$

Reynoldszahl Re_D

$$Re_D = \frac{4 \cdot q_m}{\pi \cdot D_{Rohr} \cdot \eta}$$

<u>Bestimmung des Massenstroms mittels einer Messblende gem.</u>

<u>DIN EN ISO 5167-1 von 1998</u>

Seit der Änderung der Norm im April 1998 wird der Durchflusskoeffizient C nicht mehr durch die Stolz-Gleichung bestimmt, sondern durch die Reader-Harris/Gallangher-Gleichung **[4]**. Diese lautet:

Durchflusskoeffizient C

$$C = 0{,}5961 + 0{,}0261 \cdot \beta^2 - 0{,}216 \cdot \beta^8 + 0{,}000521 \cdot \beta^{2,5} \cdot \left(\frac{10^6 \cdot \beta}{Re_D}\right)^{0,7} +$$

$$+ \left(0{,}0188 + 0{,}0063 \cdot A\right) \cdot \beta^{3,5} \cdot \left(\frac{10^6}{Re_D}\right)^{0,3} + \left(0{,}043 + 0{,}080 \cdot e^{-10 \cdot L_1} - 0{,}123 \cdot e^{-7 \cdot L_1}\right) \cdot$$

$$\cdot \left(1 - 0{,}11 \cdot A\right) \cdot \frac{\beta^4}{1-\beta^4} - 0{,}031 \cdot \left(M'_2 - 0{,}8 \cdot M'^{1,1}_2\right) \cdot \beta^{1,3}$$

Hierbei bedeuten:

$\beta = \frac{D_{BL}}{D_{Rohr}}$ Durchmesserverhältnis von Blende zu Rohr (< 1 !)

Re_D Reynoldszahl bezogen auf den Rohrdurchmesser

A $= \left(\frac{19000 \cdot \beta}{Re_D} \right)^{0,8}$ (nicht die Fläche !!)

M'_2 $= \frac{2 \cdot L'_2}{1 - \beta}$

$L_1 = \frac{l_1}{D_{Rohr}}$ das Verhältnis des Abstandes der Druckentnahme im Einlauf von der Blendenstirnseite zum Rohrdurchmesser

$L'_2 = \frac{l'_2}{D_{Rohr}}$ das Verhältnis des Abstandes der Druckentnahme im Auslauf von der Blendenrückseite zum Rohrdurchmesser (L'_2 bedeutet den Bezug auf den Abstand von der Blendenrückseite an, L_2 würde sich auf den Abstand von der Blendenstirnseite beziehen)

Unterscheidung von D–, $\frac{D}{2}$ –, Flansch– und Eck– Druckentnahmen:

Eck – Druckentnahme:

$L_1 = L'_2 = 0$

D– und $\frac{D}{2}$ – Druckentnahme:

$L_1 = 1$, $L'_2 = 0{,}47$

Flansch – Druckentnahme:

$L_1 = L'_2 = \frac{25{,}4}{D_{Rohr}}$

Die übrige Berechnung hat sich nicht geändert und wird wie in EN ISO 5167-1 : 1995 durchgeführt.

2.2.2 Messung mit Normdüse

Das genaue Berechnungsverfahren zur Bestimmung eines Massestroms mittels Einlaufdüse ist ebenfalls in der DIN EN ISO 5167-1 festgelegt **[4]**. Hier gilt die Ausgabe von 1995.

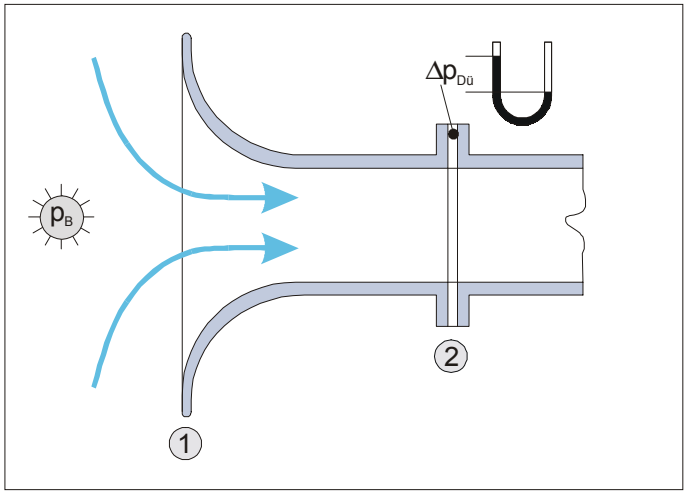


Bild 5: Prinzipskizze Düsenmessung

Gemäß Norm ist wie bei der Blendenmessung eine Iteration[1] notwendig, um die voneinander abhängigen Größen Re, C und q_m zu bestimmten. Die Berechnung von q_m und Re erfolgt analog zur Berechnung bei der Blendenmessung, lediglich die Gleichung zur Bestimmung des Durchflusskoeffizienten C ist eine andere:

Durchflusskoeffizient C bei ISA Düsen:

$$C = 0{,}9900 - 0{,}2262 \cdot \beta^{4,1} - \left(0{,}00175 \cdot \beta^{2} - 0{,}0033 \cdot \beta^{4,15}\right) \cdot \left(\frac{10^6}{Re_D}\right)^{1,15}$$

Durchflusskoeffizient C bei Langradiusdüsen:

$$C = 0{,}9965 - 0{,}00653 \cdot \beta^{0,5} \cdot \left(\frac{10^6}{Re_D}\right)^{0,5}$$

[1] sofern es sich um eine in eine Rohrleitung eingebaute Düse handelt !

Es ist jedoch möglich und üblich den Volumenstrom bei bekannter Dichte (der Umgebungsluft) durch eine Normdüse mit bekannter Durchflusszahl α zu ermitteln, wodurch man sich den Aufwand für eine iterative Berechnung sparen kann.
Die Berechnungsgleichung lautet:

$$\dot{V} = \alpha \cdot A_{Dü} \cdot \sqrt{\frac{2}{\rho} \cdot (p_1 - p_2)} = \alpha \cdot A_{Dü} \cdot \sqrt{\frac{2}{\rho} \cdot \Delta p_{Dü}}$$

2.2.3 Dynamische und kinematische Viskosität

Zur Bestimmung der Reynoldszahl wird die dynamische bzw. kinematische Viskosität eines Fluids benötigt. Allgemein lässt sich schreiben:

$$Re = \frac{c \cdot L}{\nu},$$

wobei c eine Strömungsgeschwindigkeit, L eine charakteristische Länge und ν die kinematische Viskosität sind.

Die kinematische Viskosität ν hängt mit der dynamischen Viskosität über die Dichte zusammen:

$$\nu = \frac{\eta}{\rho}$$

Bei Luft hängt die dynamische Viskosität η im Bereich zwischen 0 und 5 bar hauptsächlich von der Temperatur ab. Zur Berechnung der dynamischen Viskosität kann das Sutherland Law **[3]** verwendet werden:

$$\eta \approx \eta_0 \cdot \frac{T_0 + T_S}{T + T_S} \cdot \left(\frac{T}{T_0}\right)^{\frac{3}{2}}$$

für Luft gilt: $\eta_0 = 1{,}710 \cdot 10^{-5}\ \mathrm{Pa \cdot s}$

und $T_S = 122\ \mathrm{K}$, $T_0 = 273{,}15\ \mathrm{K}$

2.3 Ventilatorprüfstände

2.3.1 Druckseitiger Ventilatorprüfstand

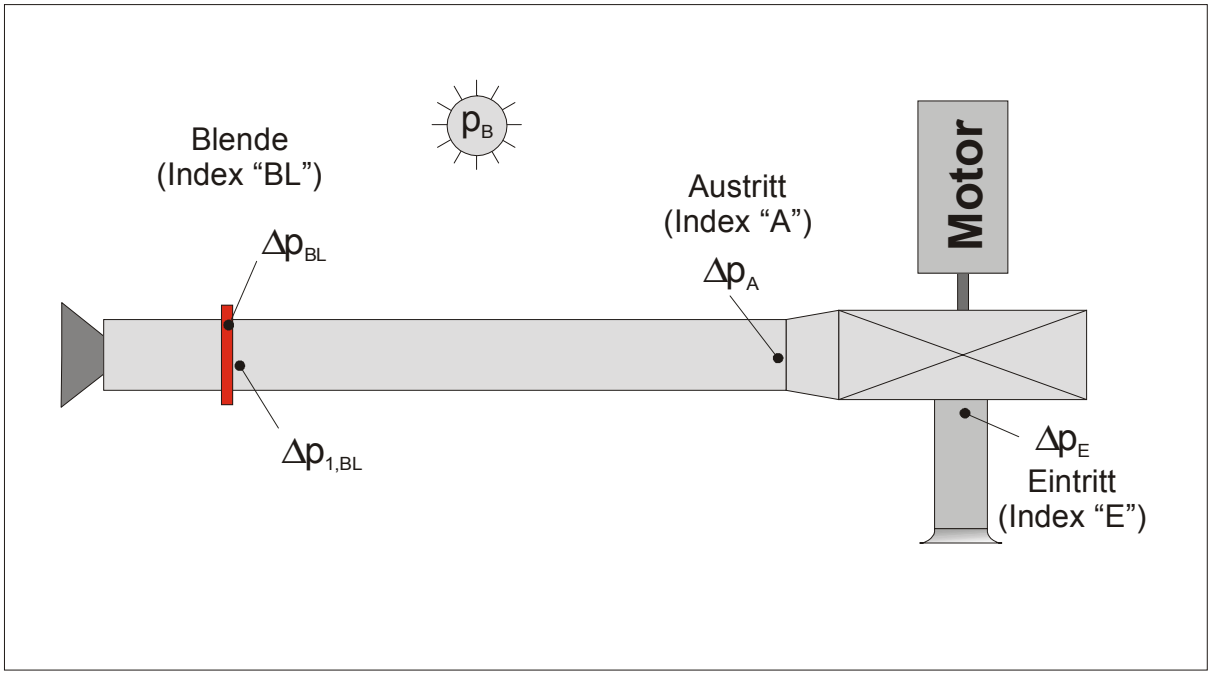


Bild 6: Prinzipskizze druckseitiger Ventilatorprüfstand

Bei dieser Konfiguration wird der Massenstrom im druckseitigen Kanal mit einer Blendenmessung bestimmt. Unter der Voraussetzung, dass das System dicht ist und keine Masse verloren geht, bleibt die Masse zwischen Eintritt und Austritt gleich (Massenerhalt).

Es werden folgende Größen gemessen bzw. ermittelt:

- Umgebungsdruck p_B
- Temperatur der Umgebungsluft t_{Luft} in °C
- Luftfeuchte φ
- die Motorleistung P_{Mot}, z.B. über Drehzahl und Drehmoment
- Druckdifferenz Δp_E und Temperatur t_E am Eintritt
- Druckdifferenz Δp_A und Temperatur t_A am Austritt
- Druckdifferenz gegen Umgebungsdruck unmittelbar vor der Blende $\Delta p_{1,BL}$
- Temperatur des Mediums unmittelbar vor der Blende $t_{1,BL}$
- Wirkdruckdifferenz Δp_{BL}

Da sich die Temperaturen und die statischen Drücke (Δp_E; Δp_A und $\Delta p_{1,BL}$) zwischen Ein- und Austritt ändern, ist die Dichte des Mediums an jedem Punkt verschieden. Die Dichten lassen sich wie folgt berechnen:

Dichte an der Blende

$$\rho_{BL} = \frac{p_B + \Delta p_{1,BL}}{\dfrac{R_L}{1 - 0{,}377861 \cdot \dfrac{p_D \cdot \varphi}{p_B + \Delta p_{1,BL}}} \cdot (t_{1,BL} + 273{,}15)}$$

Dichte am Ventilatoraustritt

$$\rho_A = \frac{p_B + \Delta p_A}{\dfrac{R_L}{1 - 0{,}377861 \cdot \dfrac{p_D \cdot \varphi}{p_B + \Delta p_A}} \cdot (t_A + 273{,}15)}$$

Hinweis: wenn die Wärmeübertragung vom Fluid auf die Rohrwand zwischen Austritt und Blende vernachlässigt wird, so kann man die Lufttemperatur vor der Blende gleich der Lufttemperatur am Austritt setzen und spart sich somit den Aufwand für eine Temperaturmessung.

Dichte am Ventilatoreintritt

$$\rho_E = \frac{p_B + \Delta p_E}{\dfrac{R_L}{1 - 0{,}377861 \cdot \dfrac{p_D \cdot \varphi}{p_B + \Delta p_E}} \cdot (t_E + 273{,}15)} \qquad (\Delta_{pE} < 0)$$

Hinweis: Um den messtechnischen Aufwand möglichst klein zu halten ist es möglich, die Umgebungstemperatur (= t_E)für die Dauer der Messung als konstant anzusehen. Die Temperatur kann z.B. vor Beginn der Messung mit einem Quecksilberthermometer ermittelt werden.

Wenn kein Ansaugrohr vorhanden ist, spricht man von einem frei ansaugenden Ventilator. In diesem Fall ist der Druck am Eintritt gleich dem Umgebungsdruck, die Temperatur am Eintritt gleich der Temperatur der Umgebungsluft. Die Größen Druckdifferenz Δp_E und Temperatur t_E am Eintritt entfallen somit. Den in den Ventilator eintretenden Volumenstrom kann man in diesem Fall mit Hilfe der Kontinuitätsgleichung bestimmen:

Bekannt: Massenstrom (am Ventilatoraustritt) q_m

Dichte am Eintritt = Dichte der Umgebungsluft $\rho_E = \rho_B$

Kontinuitätsgleichung:

$$q_{mE} = q_{mA} = \int_{A_E} \rho_E \cdot c_E \cdot dA_E = \int_{A_A} \rho_A \cdot c_A \cdot dA_A$$

$$\dot{V}_E = \frac{q_m}{\rho_E}$$

Wird mit diesem Volumenstrom der Wirkungsgrad der Maschine bestimmt, so spricht man von einem auf den Ansaugzustand bezogenen Wirkungsgrad. Die Gleichung zur Berechnung der Dichte lautet in diesem Fall:

$$\rho_E = \frac{p_B}{\dfrac{R_L}{1 - 0{,}377861 \cdot \dfrac{p_D \cdot \varphi}{p_B}} \cdot (t_{Umbg} + 273{,}15)}$$

2.3.2 Saugseitiger Ventilatorprüfstand

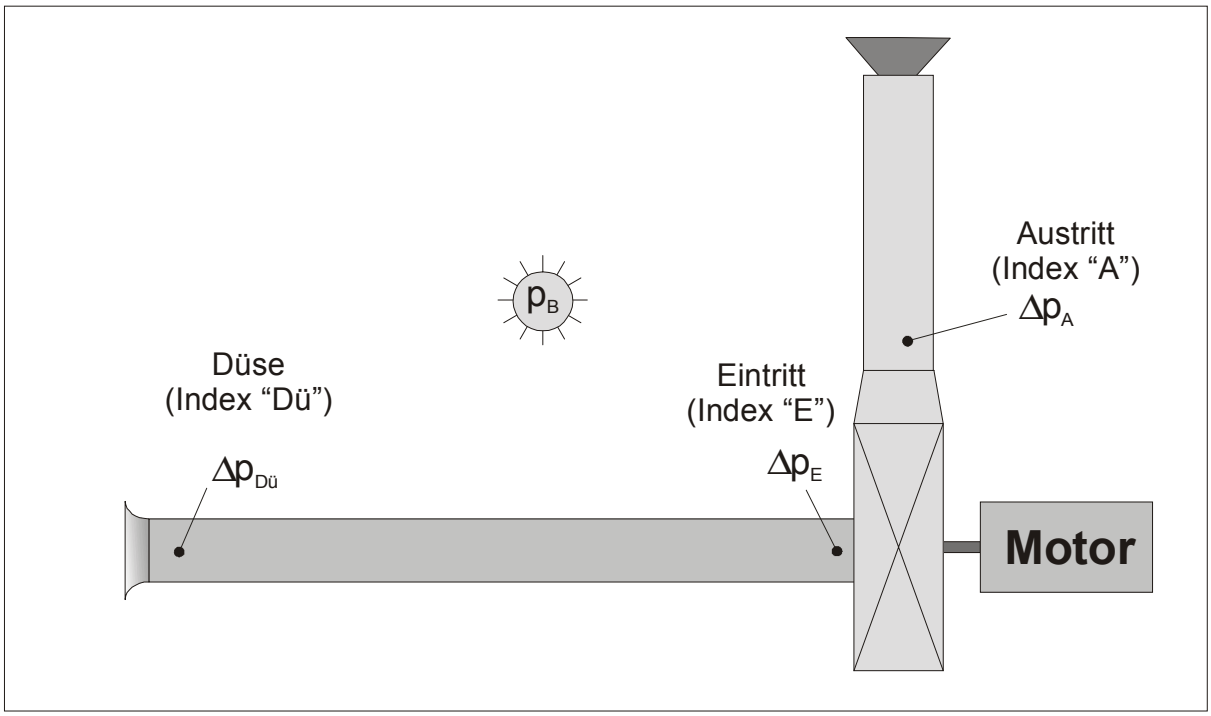


Bild 7: Prinzipskizze saugseitiger Ventilatorprüfstand

Bei dieser Konfiguration wird der Massenstrom auf der Saugseite ermittelt. Dies ist sowohl mit einer Blende als auch mit einer Düse möglich. Da die Verwendung der

Blendenmessung schon im Kapitel „Druckseitiger Ventilatorprüfstand" beschrieben wurde und bei der saugseitigen Blendenmessung die gleichen Gesetzmäßigkeiten gelten wie bei der druckseitigen Blendenmessung, wird hier nur noch auf die Düsenmessung eingegangen.

Es werden folgende Größen gemessen bzw. ermittelt:

- Umgebungsdruck p_B
- Temperatur der Umgebungsluft t_{Luft} in °C
- Luftfeuchte φ
- die Motorleistung P_{Mot}, z.B. über Drehzahl und Drehmoment
- Druckdifferenz Δp_E und Temperatur t_E am Eintritt
- Druckdifferenz Δp_A und Temperatur t_A am Austritt
- Wirkdruckdifferenz $\Delta p_{DÜ}$

Ist auf der Druckseite weder ein Kanal noch eine Drossel vorhanden, so spricht man von einem frei ausblasenden Ventilatorprüfstand. Hier gibt es keine Möglichkeit die Größen Druck und Temperatur am Austritt zuverlässig zu messen. Daher muss der Volumenstrom am Austritt mit Hilfe der Kontinuitätsgleichung bestimmt werden.

2.4 aerodynamische Kenngrößen

Die in den nachfolgenden Kapiteln beschriebenen Berechnungen und Verfahren benötigt man um die Leistung und die Güte eines Ventilators beurteilen zu können.

2.4.1 Totaldruckerhöhung, spez. Stufenarbeit und Wirkungsgrad

Bei einer thermodynamischen Betrachtung einer Strömungsmaschine handelt es sich um ein kontinuierlich durchströmtes System, für das die allgemeinen thermodynamischen Gesetze gelten. Eine Wärme und Leistungszufuhr müssen demnach zu einer Energieerhöhung des Fluids führen, die sich durch die Totalenthalpie h_t ausdrücken lässt:

$$P + \dot{Q} = \int_{A_A} h_{tA} \cdot \rho_A \cdot c_{mA} \cdot dA_A - \int_{A_E} h_{tE} \cdot \rho_E \cdot c_{mE} \cdot dA_E$$

Die spezifische Totalenthalpie erfasst die ganze, an das Fluid gebundene, Energie:

$$h_t = h + \frac{1}{2}c^2 + gz$$

In der spez. Enthalpie h sind zusätzlich zur inneren Energie die Verschiebungsarbeiten am Ein- und Austritt eingeschlossen. Unter der Voraussetzung, dass sich die Maschine an einem stationären Betriebspunkt befindet und instationäre Einflüsse nicht auftreten, kann man folgende Leistungsbilanz aufstellen:

$$\frac{P + \dot{Q}}{\dot{m}} = h_A - h_E + \frac{1}{2} \cdot \left(c_A^2 - c_E^2\right) + g \cdot (z_A - z_E)$$

Die kalorischen Zustandsgrößen Enthalpie h und Entropie s sind mit den thermischen Größen Temperatur T, Druck p und spez. Volumen v durch die Hauptgleichung von Gibbs verknüpft:

$$h_A - h_E = \int_E^A v\,dp + \int_E^A T\,ds \qquad \textbf{[1]}$$

Tritt nun bei einer Strömungsmaschine keine äußere Wärmezufuhr auf, kann man den Anteil der Wärme an der Gesamtenthalpie vernachlässigen und nur noch die

Druckänderung betrachten **[1]**. In diesem Fall spricht man von einer hydraulischen Strömungsmaschine, das Fördermedium wird dabei als inkompressibel angesehen.

Diese Betrachtungsweise ist natürlich nur eine inkompressible Näherung für Strömungen kompressibler Fluide. Nach Schade / Kunz gilt für die Dichte in einem strömenden, kompressiblen Fluid folgender Zusammenhang **[16]** :

$$\rho = \rho_0 \cdot \left(1 + \frac{\kappa - 1}{2} \cdot M^2\right)^{-\frac{1}{\kappa - 1}}$$

mit ρ_0 = Dichte im Ruhezustand,
κ = Isentropenexponent
und M = Machzahl

Aus dieser Gleichung lässt sich ableiten, dass die Dichteänderung in einer Gasstömung um so kleiner ist, je kleiner die Machzahl ist. Bei einer Auswertung dieser Gleichung gelangt man für Luft bei einer Temperatur von 20°C zu folgendem Ergebnis:

c	M	$(r_0 - r) / r_0$
30 m/s	0,087	0,4 %
50m/s	0,145	1,0 %
100 m/s	0,291	4,2 %
150 m/s	0,436	9,5 %

Aus dieser Tabelle wird ersichtlich, dass die Abweichung bei einer inkompressiblen Näherung der Dichte zur tatsächlichen Dichte bei Strömungsgeschwindigkeiten bis 50 m/s relativ gering ist. Üblicherweise berechnet man Luftströmungen bis zu einer Machzahl kleiner 0,3 inkompressibel.

Unter dieser Voraussetzung lässt sich auch die Bernoulli-Gleichung in der Form für stationäre Strömungen reibungsbehafteter, inkompressibler Fluide im Schwerefeld, längs einer Stromlinie für die Berechnungen an Ventilatoren verwenden.

Bei hydraulischen Strömungsmaschinen gilt:

mit: $$\rho = \frac{\rho_E + \rho_A}{2}$$

Bernoulli Gleichung: $$\frac{c_E^2}{2} + \frac{p_E}{\rho} + \frac{\Delta p}{\rho} = \frac{c_A^2}{2} + \frac{p_A}{\rho}$$

allgemein für den Ventilator: $$\frac{\Delta p}{\rho} = \frac{c_A^2 - c_E^2}{2} + \frac{p_A - p_E}{\rho}$$

oder als Druckerhöhung: $$\Delta p = p_A - p_E + \frac{1}{2} \cdot \rho \cdot \left(c_A^2 - c_E^2\right)$$

Die Druckerhöhung ist gleich der Summe aus der Differenz von statischem Druck an Aus- und Eintritt und der Differenz von dynamischem Druck an Aus- und Eintritt.

Die Druckerhöhung Δp wird in diesem Zusammenhang als Totaldruckerhöhung Δp_t bezeichnet und beschreibt die dem Fluid von der Strömungsmaschine zugeführte Arbeit pro Volumeneinheit:

$[\Delta p]$ = 1 Pa = $1\frac{N}{m^2}$

[W] = 1 J = 1 Ws = 1 Nm

1 Pa = $1\frac{N}{m^2} \cdot \frac{m}{m} = 1\frac{Nm}{m^3} = 1\frac{J}{m^3}$

$$\Rightarrow [\Delta p_t] = 1\ \text{Pa} = 1\ \frac{J}{m^3}$$

Bezieht man die Totaldruckerhöhung auf die Masse, so erhält man die spezifische Stufenarbeit:

$$Y = \frac{\Delta p_t}{\rho} \qquad [Y] = \frac{[\Delta p_t]}{[\rho]} = 1\frac{Pa}{\frac{kg}{m^3}} = 1\frac{\frac{J}{m^3}}{\frac{kg}{m^3}} = \mathbf{1\frac{J}{kg}} = 1\frac{Nm}{kg} = 1\frac{\frac{kg \cdot m}{s^2} \cdot m}{kg} = \mathbf{1\frac{m^2}{s^2}}$$

Sonderfälle in der Berechnung ergeben sich bei frei ansaugenden oder frei ausblasenden Maschinen:

Totaldruckerhöhung bei frei ansaugenden Maschinen:

$$\Delta p_t = \Delta p_A + \frac{1}{2} \cdot \rho_A \cdot c_A^2$$

Totaldruckerhöhung bei frei ausblasenden Maschinen:

$$\Delta p_{fa} = \Delta p_E - \frac{1}{2} \cdot \rho_E \cdot c_E^2$$

Bestimmung der Strömungsgeschwindigkeit im Rohr:

$$c = \frac{q_m}{\rho \cdot \frac{1}{4} \cdot \pi \cdot D_{Rohr}^2}$$

Bestimmung der Strömungsgeschwindigkeit, allgemein:

$$c = \frac{q_m}{\rho \cdot A} = \frac{q_v}{A}$$ (Volumenstrom durch Fläche)

Der Wirkungsgrad einer Maschine ist allgemein definiert als der Quotient aus Nutzen und Aufwand. Der Aufwand ist in diesem Falle die hineingesteckte mechanische Arbeit, mit der die Maschine angetrieben wird. Der Nutzen einer Strömungsmaschine sind die Totaldruckerhöhung und der geförderte Volumenstrom. Somit ergibt sich der Wirkungsgrad einer Strömungsmaschine zu:

$$\eta = \frac{q_V \cdot \Delta p_t}{P_M}$$

oder genauer als der auf den Ansaugzustand bezogener Wirkungsgrad:

$$\eta = \frac{q_{VE} \cdot \Delta p_t}{P_M}$$

2.4.2 dimensionslose Kennzahlen

Bei den dimensionslosen Kennzahlen einer Strömungsmaschine handelt es sich um durch Normierung gewonnene, dimensionslos gemachte Größen aus der Druckerhöhung, der Leistung und dem Volumenstrom. Sinn und Zweck dieser Kennzahlen ist es, das gemessene Verhalten einer Maschine bei einem bestimmten Betriebspunkt auf andere Betriebspunkte oder geometrisch ähnliche Maschinen extrapolieren zu können.

Die dimensionslosen Kennzahlen φ, ψ, und λ sind Kenngrößen für das Betriebsverhalten einer Strömungsmaschine bei vorgegebenem D_{Vent} und vorgegebener Drehzahl n **[5]**.

Aus dem Druck, bzw. der Druckerhöhung wird die Druckzahl ψ:

$$\psi = \frac{2 \cdot Y}{\pi^2 \cdot \left(\frac{n}{60}\right)^2 \cdot D_{Vent}^2} \qquad \textbf{[6]}$$

Bei der Berechnung ist zu beachten, dass SI-Einheiten verwendet werden. Darum ist die üblicherweise in 1/min gemessene Drehzahl der Maschine umzurechnen. Diese Umrechnung wurde in den hier angegebenen Gleichungen bereits berücksichtigt.

Aus dem Volumenstrom wir die Lieferzahl φ:

$$\varphi = \frac{4 \cdot q_V}{\pi^2 \cdot D_{Vent}^3 \cdot \frac{n}{60}} \qquad \textbf{[6]}$$

Aus der Leistung wird die Leistungszahl λ:

$$\lambda = \frac{P_M \cdot 8}{D_{Vent}^5 \cdot \left(\frac{n}{60}\right)^3 \cdot \pi^4 \cdot \rho} \qquad \textbf{[6]}$$

Die dimensionslosen Kennzahlen σ, und δ sind Kenngrößen für die Typisierung, bzw. Auslegung bei vorgegebenem $\dot{V}$ bzw. Y. **[5]**

Durchmesserzahl δ:

$$\delta = D_{Vent} \sqrt[4]{\frac{2 \cdot Y}{\dot{V}^2}} \cdot \sqrt{\frac{\pi}{4}} = \frac{\psi^{\frac{1}{4}}}{\varphi^{\frac{1}{2}}} \qquad \textbf{[5]}$$

Laufzahl σ (auch Schnelllaufzahl):

$$\sigma = n \cdot \sqrt[4]{\frac{\dot{V}^2}{(2 \cdot Y)^3}} \cdot \left\{ 2 \cdot \sqrt{\pi} \right\} = \frac{\varphi^{\frac{1}{2}}}{\psi^{\frac{3}{4}}} \qquad \textbf{[5]}$$

Die Kennzahlen δ und σ finden in dem s.g. Cordier-Diagramm Verwendung, das Aufschluss über die Bauformen von einstufigen Turbomaschinen gibt.

3 Programmierung der Messdatenerfassung

3.1 LabVIEW oder DASYLab ?

Zu programmieren ist ein aeroakustisches Messdatenerfassungssystem, d.h. Gleich- und Wechselgrößen sind möglichst gleichzeitig zu erfassen.

DASYLab ist ein einfach zu bedienende, Windows-basierte grafische Messdatenerfassungssoftware, mit der ein Bediener ohne weitreichende Programmierkenntnisse in der Lage ist, schnell und einfach per „Drag and Drop" von Bedienelementen eine Applikation zusammenzustellen.

Im Gegensatz dazu ist LabVIEW eine grafische Programmierumgebung für die Mess- und Automatisierungstechnik, die einfache grafische Entwicklung mit einer leistungsstarken, flexiblen Programmiersprache verbindet. Zwar ist LabVIEW eine weitaus komplexere Umgebung als DASYLab, der Bediener hat auch die Möglichkeit komplexe Programme zu erstellen – auf einem Niveau wie es mit DASYLab nicht möglich ist – und hat die volle Kontrolle über „seinen" Code. Darüber hinaus ist der Programmierer in LabVIEW nicht an vorgegebene Objekte gebunden, sondern hat die Möglichkeit, sich eigene Objekte, die auch in einer anderen Sprache (z.B. C oder C++) erstellt werden können, mit den gewünschten Eigenschaften zu programmieren.

Letzteres war auch für die Entscheidung für LabVIEW maßgeblich, da im Vordergrund stand, eine flexible, erweiterbare und wartbare Software zu erstellen, die ggf. auch unabhängig von der Programmierplattform lauffähig ist. Zwar war der Einarbeitungsaufwand erheblich größer als bei DASYLab, diesen Nachteil machte LabVIEW aber schnell durch eine sehr komfortable, intuitive Bedienung und den Umstand, dass es sich um ein Programm handelt, nicht um eine an eine bestimmte Software gebundene Applikation, wieder wett. Als Beispiel für die Einschränkungen von DASYLab in der Version 5.6 sei die fehlende Möglichkeit der Programmierung einer Iterationsschleife genannt.

3.2 Grundlegende Programmstrukturen in LabVIEW

Ein LabVIEW Programm setzt sich immer aus 2 Komponenten zusammen: Der Bedienoberfläche, auf der der Benutzer Eingaben vornehmen kann und Ergebnisse angezeigt bekommt und dem Diagramm, das den Quellcode enthält.
LabVIEW ist eine multitasking – multithreading Entwicklungsumgebung[2], die nach dem Datenfluss-Modell arbeitet . D.h., zwei im Diagramm parallel angeordnete Strukturen werden gleichzeitig ausgeführt, wenn Daten zur Verarbeitung vorliegen. Hierzu ein kleines Beispiel:

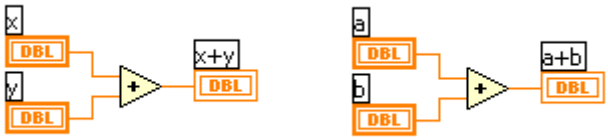


Bild 8: Parallele Strukturen in einem LabVIEW Diagramm

Dieses einfache Programm berechnet die Summen aus x und y, sowie a und b. Die Strukturen sind parallel im Diagramm angeordnet, und werden darum beim Starten des Programms gleichzeitig abgearbeitet. Im Gegensatz dazu werden die Berechnungen in dem Bild 9: nacheinander ausgeführt, da zum Startzeitpunkt das Berechnungsergebnis der Summe x+y noch nicht vorliegt, die 2. Berechnung also noch keine Daten hat.

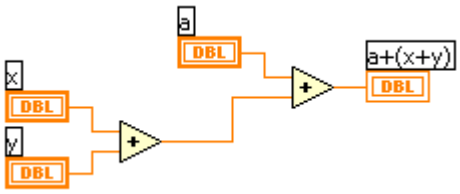


Bild 9: nacheinander ausgeführte Berechnungen

[2] Multitasking ist auf PCs mit nur **einer** CPU als quasiparalleles Abarbeiten von Befehlen mit Hilfe eines Zeitscheibenmodells zu verstehen und kann nicht mit echtem Multitasking, wie es nur auf Parallelrechnern möglich ist, verglichen werden. So kann es durchaus vorkommen, dass zeitkritische hardwarenahe Prozesse nicht wie erwartet parallel, sondern nacheinander ausgeführt werden.

Eine Nichtbeachtung dieser grundlegenden Dinge führt in den meisten Fällen – insbesondere beim Verwenden von lokalen Variablen –zu überraschenden, nicht gewollten Ergebnissen.

3.2.1 Sequenz und While-Schleife

Zu den wichtigsten Strukturen in einem LabVIEW Programm gehören die Sequenz und die While-Schleife.

In einer While-Schleife wird der enthaltene Code so lange ausgeführt, bis die Abbruchbedingung eintritt, in dem Beispiel unten wird so lange die Summe aus a und b gebildet, bis die STOPP-Taste gedrückt wird:

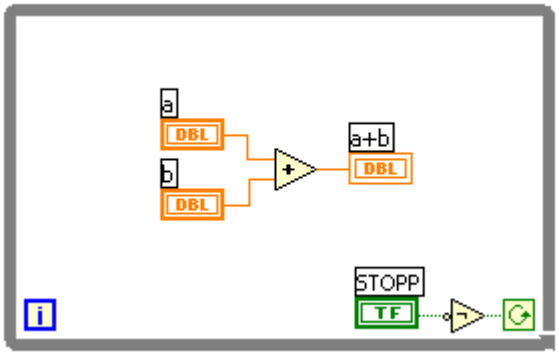


Bild 10: While-Schleife

In einer Sequenz werden die Anweisungen nacheinander, Schritt für Schritt ausgeführt:

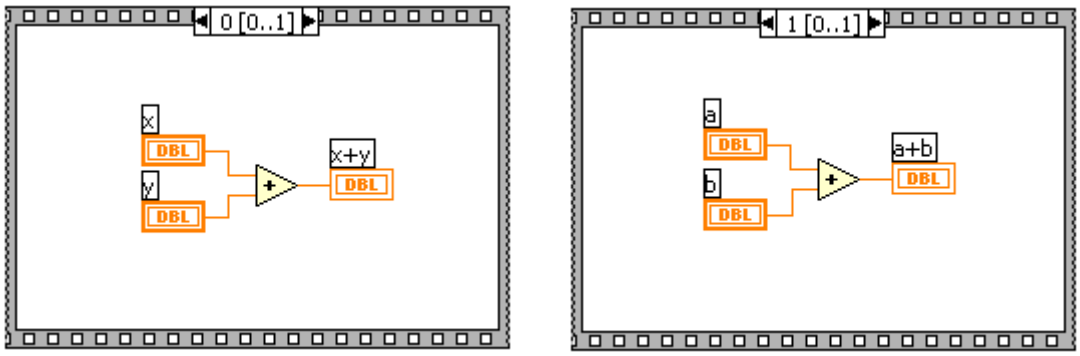


Bild 11: Sequenz mit 2 Schritten

Im ersten Schritt (Index 0) wird die Summe aus x und y berechnet, im 2. Schritt (Index 1) wird die Summe aus a und b berechnet.

3.2.2 Call by Value – Call by Reference

Wie auch in anderen Hochsprachen gibt es in LabVIEW die Möglichkeit, Unterprogramme als Funktion zu behandeln und mittels Call by Reference oder Call by Value aufzurufen.

Die Methode Call by Value bedeutet, dass das Hauptprogramm an eine Funktion (= das Unterprogramm) Werte übergibt, und vom Unterprogramm die berechneten Ergebnisse zurückbekommt. Zur Verdeutlichung soll das folgende Beispiel dienen:

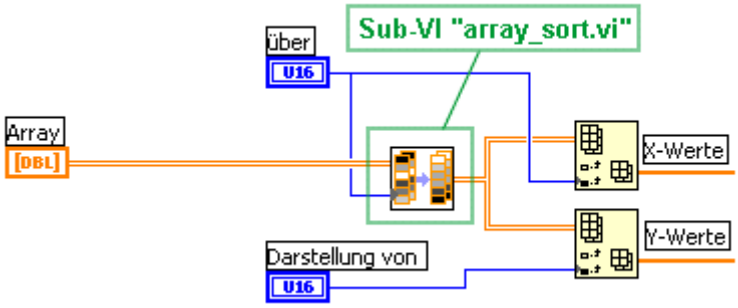


Bild 12: Beispiel für Call by Value

Der in Bild 12: dargestellte Quellcode stammt aus dem Auswerteprogramm. In „Array" sind die im Hauptprogramm gemessenem Werte gespeichert. Damit die Werte weiterverarbeitet werden können, müssen diese zunächst sortiert werden. Dazu dient das Unterprogramm „array-sort.vi". Der Aufruf dieses Unterprogrammes erfolgt per Call by Value, da die Funktion (= das Unterprogramm) als Sub-VI direkt in den Quellcode mit eingebunden ist. Die in der Variablen „Array" enthaltenen Werte werden an das Sub-VI übergeben und somit einer weiteren, im Sub-VI vorhandenen, Variablen zugewiesen. Mit der Kopie der Daten wird dann das Ergebnis berechnet und an die übergeordnete Programmstruktur zurückgegeben. Für diese Operation wird also im Speicher des Rechners eine Kopie der Daten erstellt. Der Vorteil dieser Methode liegt darin, dass die Originaldaten (also die, die in der Variablen „Array" enthalten sind) nicht manipuliert werden. Der Nachteil liegt genau darin, dass ein 2. Datensatz erstellt wird und das Erstellen dieses Datensatzes Rechenzeit beansprucht.

Im Gegensatz dazu werden bei der Methode Call by Reference keine Daten übergeben, sondern Verweise (References) auf die Daten. Die Gefahr bei dieser Methode liegt darin, dass keine Kopie der Daten erstellt wird, sondern die

Originaldaten selbst manipuliert werden. Dies bedeutet jedoch einen Geschwindigkeitsvorteil, weil kein 2. Datensatz erstellt wird und somit auch keine Rechenzeit dafür in Anspruch genommen werden muss.

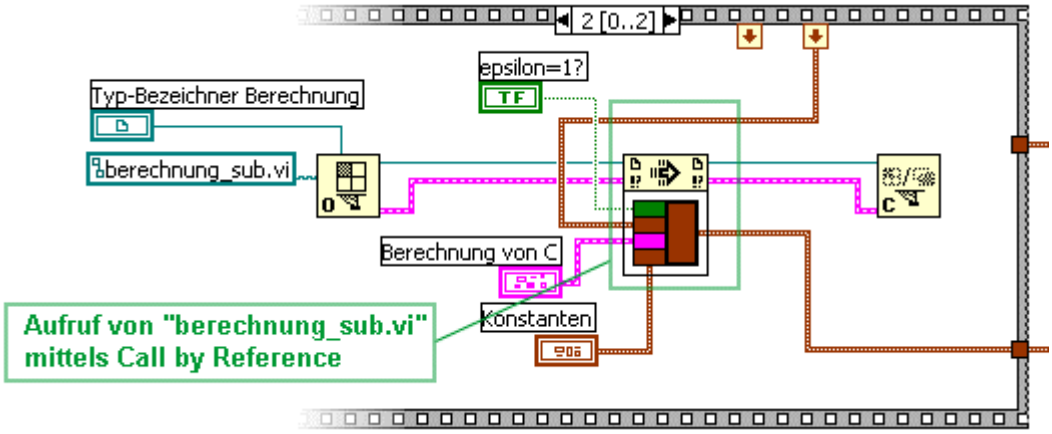


Bild 13: Beispiel für Call by Reference

Der in Bild 13: dargestellte Quellcode stammt aus dem Hauptprogramm[3]. Hier wird das Sub-VI „berechnung_sub.vi" per Referenz aufgerufen. Die Daten, die in die Funktion hineingeführt werden, werden verändert, was an dieser Stelle aber auch zulässig ist, da bereits eine Kopie dieser Daten erstellt wurde und diese Daten auch nicht mehr für eine weitere Operation benötigt werden.

3.2.3 Cluster

Ein Cluster in LabVIEW lässt sich am einfachsten als „Bündel" von Daten beschreiben, die nicht zwangsläufig von einander abhängig sind. Sie eignen sich hervorragend, um eine größere Anzahl von einzelnen Elementen zu **einem** übersichtlichen Element zusammenzufassen und somit die Daten zu strukturieren. In Bild 13: ist u.a. das Symbol des Clusters „Konstanten" zu sehen, der die zur aerodynamischen Berechnung benötigten Konstanten (z.B. die Gaskonstante oder den Laufraddurchmesser, etc) beinhaltet.

Das Praktische an Clustern ist, dass alle darin enthaltenen Objekte einen Namen haben und der Programmierer mit dem Objekt „nach Namen aufschlüsseln" die

[3] Anmerkung: dieser Code ist nicht in der Version 9.3d zu finden, sondern stammt aus einer älteren Version.

Möglichkeit hat, definiert auf einen einzelnen Wert zurückzugreifen oder mit dem Objekt „nach Namen bündeln" auch einen einzelnen Wert schreiben kann. Sinnvoll angewendet, d.h., es werden Daten, die zu einer Gruppe gehören (z.B. Berechnungsergebnisse oder Konstanten), zusammengefasst, erleichtern Cluster die Programmierung erheblich, leisten einen wichtigen Beitrag zur Vermeidung von unübersichtlichem „Kabelwirrwarr" und machen somit den Quellcode leichter lesbar und verständlicher.

3.2.4 Sub-VIs

Ein Sub-VI ist – wie der Name schon sagt – ein Unterprogramm und kann für verschiedene Zwecke eingesetzt werden. Die Programmierung eines Sub-VIs bietet sich z.B. an, wenn man eine ganz bestimmte Funktion oder Routine in einem Programm öfter benötigt. Zwar wäre es theoretisch möglich, den Code mittels Copy & Paste so oft wie gewünscht zu vervielfältigen, dies führt aber schnell zu einem unübersichtlichen und schlecht strukturierten Programm. Der Einsatz von Sub-VIs ist in diesem Zusammenhang wesentlich eleganter und ergonomischer, da zum einen der Code des Hauptprogramms vergleichsweise klein und überschaubar bleibt, zum anderen sich Änderungen am Sub-VI sofort auf alle Stellen im Code auswirken.

Weiterhin ist der Einsatz von Sub-VIs auch sinnvoll, um den Code des Hauptprogramms zu strukturieren und komplexe Funktionen „auszulagern". Dies hilft dem Programmierer auch beim Debugging, da es bei einem nicht gegliederten Quellcode sehr schwer werden kann, die komplexen Zusammenhänge zu verstehen und die Datenflüsse nachzuvollziehen und so Fehlerquellen aufzuspüren.

Die einfachste Möglichkeit ein Sub-VI zu erstellen ist, den entsprechenden Code im Diagramm-Fenster zu markieren und im Menü den Befehl „Bearbeiten – Sub-Vi erstellen" auszuwählen:

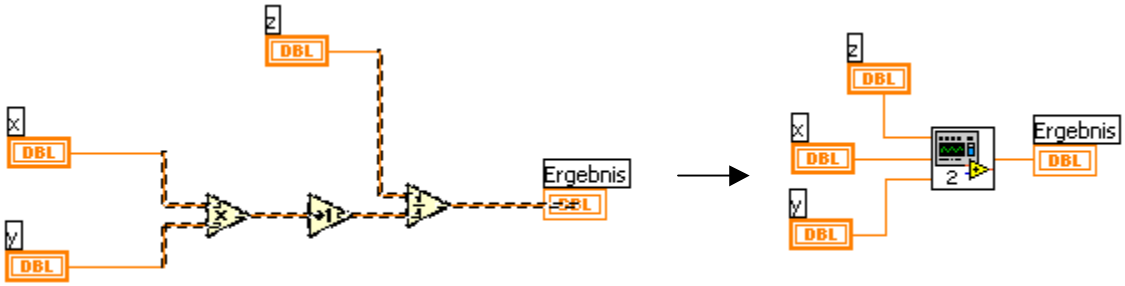


Bild 14: Sub-VI erzeugen

LabVIEW erzeugt dann selbständig ein neues VI, das die nötigen Anschlüsse enthält, mit denen das VI als Sub-VI in ein anderes eingebunden werden kann. Das Sub-VI wird als eigenständige Datei abgespeichert, ist auch ohne das übergeordnete VI lauffähig und kann ohne weiteres auch in anderen VIs verwendet werden.

Alternativ dazu kann auch jedes bereits bestehende VI so bearbeitet werden, dass es als Sub-VI eingesetzt werden kann. Dazu müssen Anschlüsse definiert werden:

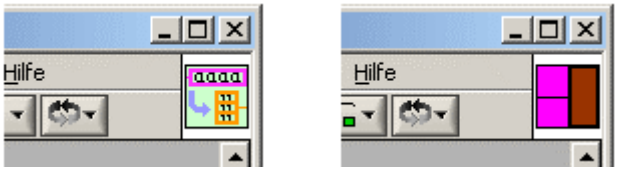


Bild 15: VI-Symbol und Anzeige der Anschlüsse des VIs

Normalerweise hat ein VI keine Anschlüsse durch die ein Datentransfer zu anderen VIs möglich ist. Diese müssen erst hinzugefügt werden. Dazu klickt man in dem Frontpanel-Fenster rechts oben auf das VI-Symbol, klickt mit der rechten Maustaste und wählt „Anschlüsse anzeigen". Daraufhin verschwindet das VI-Symbol und die Anschlüsse sind sichtbar. Sind keine Anschlüsse definiert, so sieht man nur ein leeres, weißes Kästchen. Wiederum mit einem Klick mit der rechten Maustaste gelangt man in das Auswahlmenü, mit dem man Anschlüsse hinzufügen, löschen und neu anordnen kann. Wenn alle nötigen Anschlussfelder vorhanden sind, so kann man mit dem „Kabeltrommel – Werkzeug" die Anschlüsse zuordnen, in dem man zuerst auf das Bedienelement auf dem Frontpanel klickt und anschließend auf den Anschluss, der zugewiesen werden soll. Wenn vorher kein Anschluss zugewiesen

war, dann ändert sich die Farbe des Anschlusses von weiß zu der Farbe, die dem Datentyp des Anzeige- oder Bedienelements entspricht.

3.3 Überblick über das Messdatenerfassungssystem

Dieses Kapitel soll einen Überblick über die Programmstruktur geben. Dabei wird ganz bewusst nicht auf alle Einzelheiten eingegangen und der komplette Quellcode erklärt, sondern nur die grobe Struktur erläutert. Dies soll dem Leser helfen, sich im Quellcode zurechtzufinden, ohne dass er die jeweils genaue Funktion des Codes kennen muss. Eine detaillierte Erklärung des Codes ist in Kapitel 4 zu finden.

3.3.1 Einschränkungen

Auch wenn während der Programmierung Wert auf eine möglichst multifunktionale Verwendbarkeit gelegt wurde, so bleibt die Software dennoch immer eine für einen Einzelfall erstellte Spezialanwendung, die sich ohne Änderungen am Quellcode nicht für andere Zwecke einsetzen lässt. Diese Restriktionen gelten ganz besonders für die Verwendung der Messhardware:

Ein wesentlicher Bestandteil des Programms ist die Steuerung des Agilent 34970A Messdatenerfassungs-/Schaltsystems. Ohne dieses Gerät ist das Programm – ohne sehr große Veränderungen – nicht lauffähig (es ist aber wahrscheinlich, dass sich Geräte ähnlicher Bauart des o.g. Herstellers ohne weiteres verwenden lassen).

Ebenso ist der Anwender bei der Wahl der Schnittstelle auf die an seinem Rechner verfügbaren COM-Ports beschränkt. Andere Schnittstellen, z.B. GPIB, wurden nicht implementiert.

Bauartbedingt durch das Agilent 34970A Messdatenerfassungs-/Schaltsystems ist es erforderlich, dass die Messgeräte Spannungen als Ausgangssignal liefern. Eine Messung von Strömen ist mit der zur Verfügung stehenden Messkarte ohnehin nur auf 2 Kanälen möglich.

3.3.2 Benutzeroberfläche

Das Frontpanel des Messdatenerfassungssystems gliedert sich in 5 Bereiche:

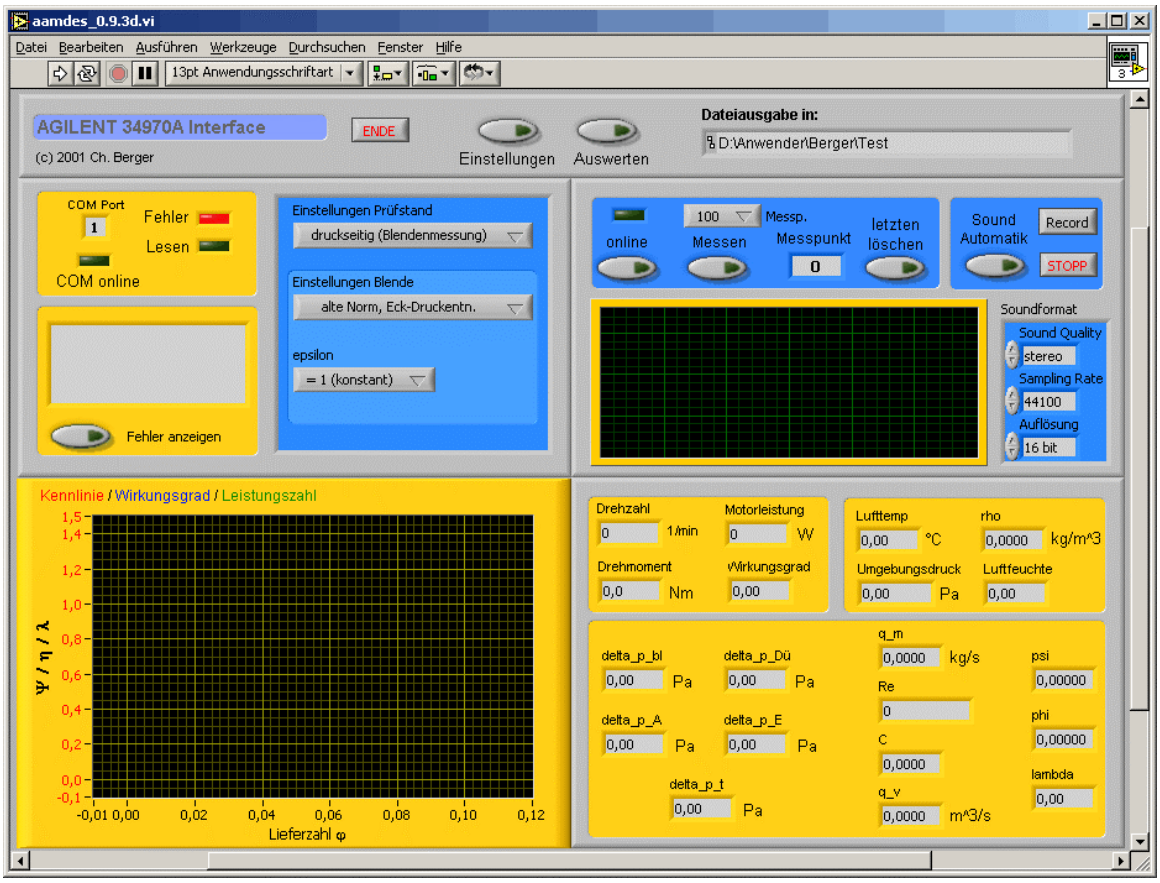


Bild 16: Frontpanel des Messdatenerfassungssystems

Im obersten Abschnitt befinden sich die Bedienelemente zum Beenden des Programms, zum Anzeigen der Einstellungen, zum Starten des Auswerteprogramms, sowie die Anzeige für den Arbeitspfad und den Dateinamen der Datei, in dem die Daten gespeichert werden.

Im linken oberen Bereich befinden sich die Anzeigeelemente, die den Status der RS232 Schnittstelle anzeigen: Die Nummer des verwendeten COM-Ports wird als Zahl angezeigt. die LED „COM online" leuchtet, wenn die Initialisierung des COM-Ports in der Startphase des Programms ohne Fehler durchgeführt worden ist. Sollte die Initialisierung fehlgeschlagen sein, so bleibt die LED „COM online" aus, stattdessen leuchtet die LED „Fehler" rot. Das Programm muss in diesem Fall neu gestartet werden. Die LED „Lesen" blinkt in der Betriebsart „online" oder „Messen" auf, wenn Daten von der RS232 Schnittstelle gelesen werden. Wenn keine Messung

durchgeführt wird (sich das Programm also im „Leerlauf" befindet), leuchtet die LED dauerhaft.
Das Bedienelement „Fehler anzeigen" dient dazu – wie der Name schon sagt – Fehler aus dem Fehlerspeicher der Multiplexers auszulesen und anzuzeigen. Der Schalter sollte nur betätigt werden, wenn die Schalter „Messen" und „online" aus sind, sonst kann der Multiplexer die ankommenden Befehle nicht verarbeiten und bricht mit einer Fehlermeldung ab. Die ausgelesenen Fehlermeldungen werden nach dem Abruf aus dem Fehlerspeicher gelöscht.
Im linken oberen Bereich befinden sich auch die Menüs mit denen der Benutzer die Prüfstandskonfiguration auswählen kann. Die Anwahl der Konfiguration „ε wird berechnet" macht nur bei Wahl der Konfiguration „Blendenmessung" Sinn.

Im rechten oberen Bereich befinden sich die Bedienelemente mit denen der Benutzer die Messdatenerfassung und die Sound-Aufnahme steuern kann. Durch Betätigen des Schalters „online" wechselt das Programm in die Betriebsart „online Messung", d.h., die Berechnungen werden über nur 10 vom Multiplexer gemittelte Messpunkte durchgeführt. Bei Betätigung des Schalters „Messung" wechselt das Programm für **einen** Zyklus in die Betriebsart „Messung", d.h., die Berechnungen werden über eine bestimmte Anzahl gemittelter Messpunkte (größer 30) durchgeführt. Die Betriebsart „Messung" hat Vorrang vor der Betriebsart „online Messung", d.h., wenn „online" angewählt ist und der Schalter „Messen" gedrückt wird, dann schließt die Software den laufenden Programmzyklus ab ,wechselt in die Betriebsart „Messung" und nach Abschluss der Messung wieder zurück in die Betriebsart „online Messung". Die aktuelle Nummer des Messpunktes wird nach jedem Durchlauf eines Zyklus der Betriebsart „Messung" um eins erhöht. Der Benutzer hat aber auch die Möglichkeit jeweils den letzten Messpunkt zu überschreiben um die Messung zu wiederholen.
Der Schalter „Sound Automatik" dient zur Aufzeichnung der Sounddaten. Die beiden Schalter rechts daneben „Record" und „Stopp" können betätigt werden, man sollte aber davon absehen, da dies unter Umständen dazu führen kann, dass die Steuerung des Programmes „aus dem Takt" gerät, weil sich 2 parallel abzuarbeitende While-Schleifen gegenseitig blockieren. Wenn der Schalter „Sound Automatik" gedrückt ist, dann nimmt die Software (nur) in der Betriebsart „Messung" für die Messdauer eine *.wav-Datei auf und speichert diese im Arbeitsverzeichnis. Die Informationen über

das verwendete Sound-Format werden in dem kleinen Feld darunter angezeigt. Stereo bedeutet in diesem Fall, dass auf 2 Kanälen Audiodaten aufgenommen werden, z.B. mit 2 Mono-Mikrofonen an 2 unterschiedlichen Orten.
In dem blau umrandeten Display wird online ein FFT der aufgenommen Audiodaten angezeigt. Diese FFT ist jedoch nicht skaliert und liefert somit nur Informationen über die Frequenzanteile, nicht jedoch über den Pegel. Das Display wird vertikal durch dünne grüne Linien in 6 Fehler aufgeteilt, die Anzeige beginnt bei 0 Hz und endet bei 3000 Hz[4].

In dem großen Display links unten werden die ermittelten Daten der online-Messung als farbige Kreise und die Daten der gespeicherten Messwerte als farbige Kästchen dargestellt. Es werden jeweils nur die gerade aktuellen Werte der online Messung angezeigt, jedoch alle bereits in der Betriebsart „Messung" aufgenommenen (und gespeicherten) Werte. Zur besseren Unterscheidung wurden folgende Farbzuordnungen gewählt: rot = psi über phi, blau = eta über phi, grün = lambda über phi (der Wirkungsgrad ist mit 100 zu multiplizieren um den Wert in % zu erhalten).

Im rechten unteren Abschnitt werden die Ergebnisse der Berechnung und – soweit nötig – auch die gemessenen Werte als Zahlen mit Einheiten (soweit vorhanden) dargestellt. Die bei der jeweiligen Prüfstandkonfiguration nicht benötigten oder nicht aufgenommen Werte bleiben dauerhaft 0, z.B. werden bei dem druckseitigen Ventilatorprüfstand kein Wirkdruck an der Düse und keine Druckdifferenz am Eintritt gemessen.

[4] diese obere Grenze ist nicht endgültig und kann (bis zur ½ Sampling-Frequenz) erweitert oder verkleinert werden.

3.3.3 Überblick

Diese beiden Strukturen Sequenz und While-Schleife wurden in Kapitel 3.2.1 explizit erwähnt, weil das Hauptprogramm ganz oberflächlich gesehen aus einer Sequenz und einer While-Schleife besteht:

Bild 17: stark vereinfachte Darstellung des Programmablaufes

Die äußerste Struktur ist eine Sequenz aus 4 Schritten. In Schritt 1 (Index 0) wird das Hauptprogramm gestartet, das (mit Hilfe eines Unterprogramms) die Einstellungen und die Kalibrierung, etc. abfragt. In Schritt 2 (Index 1) werden der COM-Port geöffnet, die Einstellungen geladen, der Multiplexer in die gewünschte Betriebsart versetzt, die Sub-VI's referenziert, Variablen – soweit notwendig – zu null gesetzt und die Dateien für die Speicherung erstellt. In Schritt 3 (Index 2) befindet sich das eigentliche Hauptprogramm, das in einer While-Schleife läuft und so lange wiederholt wird, bis der Stopp-Button gedrückt wird. In Schritt 4 (Index 3) werden der COM-Port und die referenzierten Unterprogramme wieder geschlossen. Diese Reihenfolge ist wichtig, um das Programm „sauber" zu starten und zu beenden. Wird dies nicht beachtet, so treten u.a. Laufzeitfehler bei der Initialisierung der RS232-Schnittstelle auf.

3.3.4 Struktogramm der Messdatenerfassung

Der Kern des Programms besteht aus der Messdatenerfassung und der Berechnung der aerodynamischen Größen. Hier stehen 2 „Betriebsarten" zur Verfügung, die sich nur dadurch unterscheiden, dass in der Betriebsart „Online" über nur 10 Messpunkte gemittelt wird, während in der Betriebsart „Messung" über eine bestimmte Anzahl von Messpunkten (größer 30 und kleiner 300) gemittelt wird.

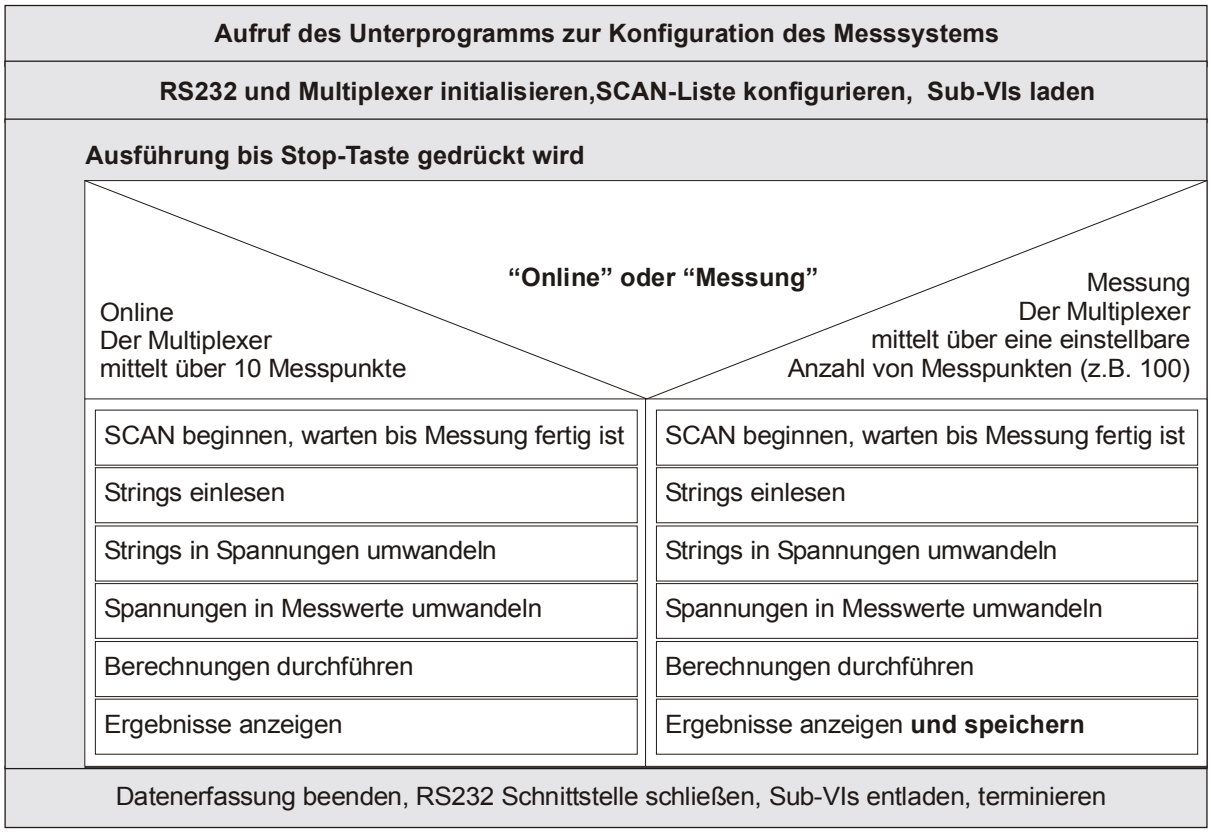


Bild 18: Struktogramm

Unter „Messpunkt" ist in diesem Zusammenhang der Wert der gemessenen Größe zu einem bestimmten Zeitpunkt zu verstehen. Die Zeitintervalle zwischen zwei Messpunkten sind abhängig von der Abtastrate des Multiplexers. Diese ist wiederum abhängig von der verwendeten Messkarte und liegt in der hier verwendeten Konfiguration bei 60 Messungen pro Sekunde pro gemessenem Kanal. Dies entspricht einer Abtastrate von ca. 3 Hz, wenn alle 20 verfügbaren Kanäle abgetastet werden.

Da die Mittelung der gemessenen Einzelwerte (= Messpunkt) im Multiplexer vorgenommen wird (damit nur ein Wert übertragen werden muss), benötigt der Multiplexer eine bestimmte Zeit um die vorgegebene Anzahl von Werten zu messen und den Mittelwert zu bestimmen. In dieser Zeit zwischen dem Start des Scan-Vorganges und der Rückgabe der Werte durch den Multiplexer, muss das Programm warten.

Da das Agilent 34970A Messdatenerfassungs-/Schaltsystem die gemittelten Messwerte als Strings an die Schnittstelle überträgt, müssen diese Strings im nächsten Schritt in Zahlen umgewandelt werden. Dies geschieht in dem Unterprogramm „string2num_sub.vi". Die nun vorliegenden Werte sind Spannungen, die mit Hilfe des Unterprogramms „kalib_sub.vi" in Messgrößen umgewandelt werden. Anschließend werden in dem Unterprogramm „berechnung_sub.vi" die nötigen aerodynamischen Berechnungen durchgeführt. Nach Abschluss dieser Operationen zeigt das Hauptprogramm die ermittelten Ergebnisse auf der Benutzeroberfläche an.

In diesem Struktogramm ist die Funktion „Fehler anzeigen" nicht enthalten, da diese nicht zur eigentlichen Funktion des Programms – dem Erfassen und Auswerten von aerodynamischen Messdaten – gehört, sondern lediglich ein Hilfsmittel für den User ist, bei Störungen am Agilent Multiplexer, die Fehler auszulesen und den Fehlerspeicher zu löschen.

3.4 Steuerung eines Agilent Multiplexers

Das Agilent 34970 A Messdatenerfassungs- / Schaltsystem ist ein Multifunktionsgerät, das mit entsprechenden Einschüben für Steckkarten auf der Rückseite für den jeweiligen Zweck ausgerüstet werden kann. In der hier verwendeten Konfiguration war das Gerät mit einer 20 Kanal Multiplexer-Karte bestückt, mit der Scannen und direktes Messen einer Temperatur, von Spannungen Widerständen, Frequenzen und von Strömen, letzteres jedoch nur auf 2 Kanälen, möglich ist. Die Kommunikation zwischen Messgerät und PC ist sowohl über die RS232 Schnittstelle, als auch über den GPIB-Bus möglich. Die Entscheidung für die RS232 Schnittstelle beruhte hauptsächlich darauf, dass praktisch jeder PC – insbesondere auch mobile Geräte (Notebooks und Laptops) – über diese Schnittstelle verfügen, die notwendigen Treiber bereits in das Betriebssystem integriert sind, und kein zusätzlicher Aufwand für Einsteckkarten, o.Ä. notwendig ist.

3.4.1 Einstellungen zum Betrieb an einer RS232 Schnittstelle

In der im Gerät einprogrammierten Grundkonfiguration für die Kommunikation über die RS232 Schnittstelle sind folgende Einstellungen festgelegt:

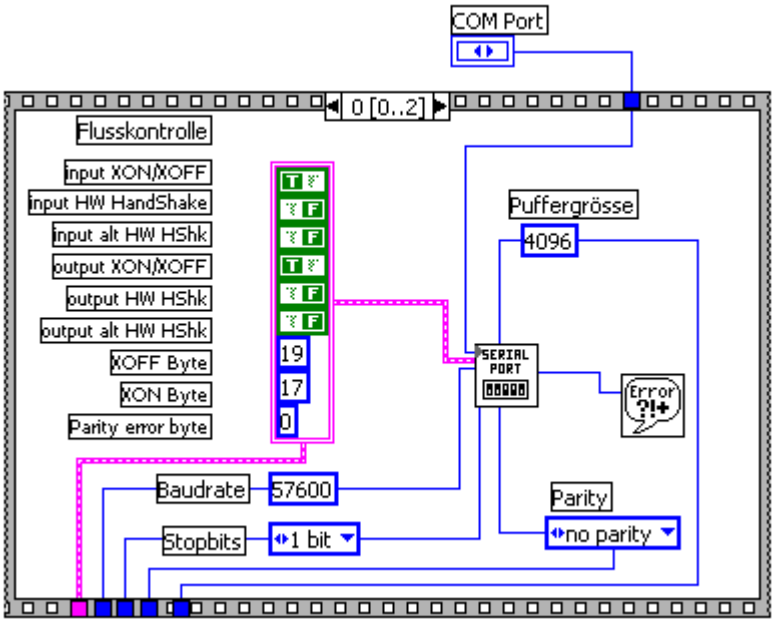


Bild 19: Einstellungen der RS232 Schnittstelle

Baudrate	56700	
Stopbits	1 bit	
Parität	keine Parität	
Datenbits	8	
input XON / XOFF	ja	= Datenfluß-steuerung XON/XOFF
input Hardware Handshake	nein	
input alt Hardware Handshake	nein	
output XON / XOFF	ja	
output Hardware Handshake	nein	
output alt Hardware Handshake	nein	
XOFF Byte	19	
XON Byte	17	
Parity Error Byte	0	

Da eine Änderung der Einstellungen für den Betrieb keine Vorteile bringt, wurden die Einstellungen so übernommen. Des weiteren fällt das Gerät bei einem Reset auf diese werkseitigen Grundeinstellungen zurück, d.h., bei einem evtl. auftretenden Fehler ist das Gerät nach einem Reset sofort wieder ansprechbar, ohne dass die COM Einstellungen neu initialisiert werden müssen.[5]

3.4.2 Externe Programmierung mit SCPI

Die zur externen Steuerung des Agilent 43970A Messdatenerfassungs-/Schaltsystems verwendete Skriptsprache heißt SCPI. Die Befehle werden über die Schnittstelle als Strings gesendet, das Gerät wertet die empfangenen Befehle aus und wechselt daraufhin in den gewünschten Betriebsmodus oder führt die entsprechenden Aktionen aus.

Mit Ausnahme des Speichers für die SCAN-Liste verfügt das Agilent 34970A Messdatenerfassungs- / Schaltsystem aber über keinen Programmspeicher, d.h., das Gerät reagiert jeweils nur auf den aktuell gesendeten Befehl. Eine Abfolge von Befehlen (=Programm) muss also in der Steuerungssoftware implementiert sein. Es handelt sich also nicht wie im Handbuch beschrieben um eine externe Programmierung, sondern um eine externe Steuerung, mit einer speziell dafür entwickelten Software, die den Ablauf steuert.

---

[5] Anmerkung: wären andere Einstellungen für den Betrieb gewählt worden, so müsste das Gerät nach einem Reset erst mit den Standardeinstellungen angesprochen werden und dann wieder auf die neuen Einstellungen initialisiert werden.

Beim Senden der Befehle muss darauf geachtet werden, dass das Gerät nach dem Erhalt eines Befehls, der eine bestimmte Zeit in Anspruch nimmt (z.B. Scannen von 100 Messpunkten), keine weiteren Befehle annimmt, bis die aktuelle Operation abgeschlossen ist. Werden in der Zeit, in der das Gerät nicht empfangsbereit ist, dennoch Befehle gesendet, so ignoriert der Multiplexer diese Anweisung und bricht u.U. die aktuelle Operation mit der Fehlermeldung „Query Interrupted" ab. Dieser Umstand wirkt sich z.B. im Hauptprogramm in so fern aus, dass die Routine, die die Daten vom COM-Port einliest, so lange warten muß (ohne auf den COM-Port zuzugreifen), bis der Multiplexer die zu sendenden Daten in den Eingangspuffer geschrieben hat.

3.4.3 Übersicht über die verwendeten SCPI - Befehle, sowie Erläuterung

Die SCPI - Befehle an sich sind leicht verständlich, sie setzen sich aus jeweils 3-4 Anfangsbuchstaben der englischen Wörter für Messen, Scannen, Berechnen, etc. zusammen. Grundsätzlich gilt: SCPI-Befehle sind zeilenweise zu senden, d.h., ein Befehl muß mit einem Carriage-Return (=nächste Zeile) beendet werden.

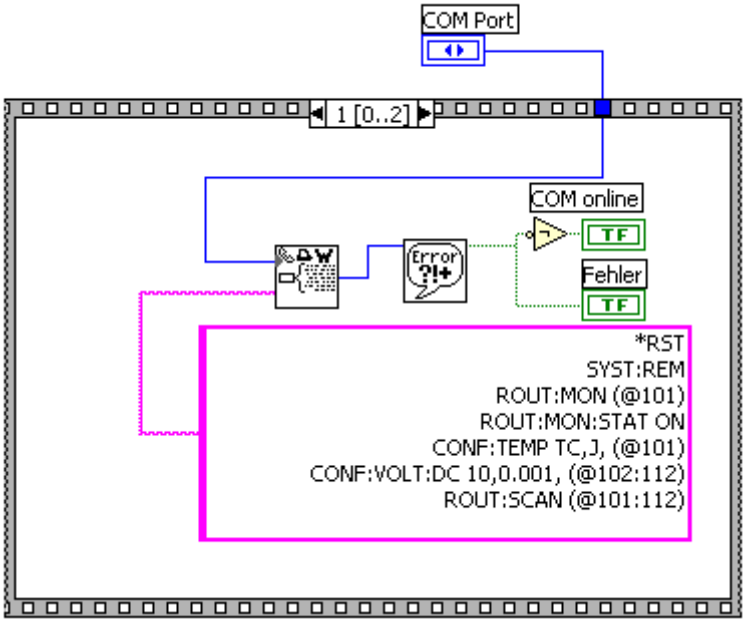


Bild 20: Initialisierung des Multiplexers mittels SCPI Befehlen

Eine Zusammenstellung der verwendeten Befehle mit Erklärung **[11]**:

***RST**	**R**e**S**e**T**	Setzt das Gerät in den werkseitig eingestellten Ausgangszustand zurück
SYST:REM	**SYST**em**:REM**ote	Versetzt das Gerät in die extern gesteuerte Betriebsart für den RS232 Betrieb. Alle Tasten der Frontplatte außer der Taste „local" werden bei dieser Betriebsart deaktiviert
CONF:VOLT	**CONFigure:VOLT**	Konfiguriert einen Kanal oder eine SCAN-Liste für Spannungsmessung. Mögliche Optionen: **DC** oder **AC**, die Angabe des Bereiches und der Genauigkeit (**10,0.001**) ist optional.
CONF:TEMP	**CONF**igure**: TEMP**erature	Konfiguriert einen Kanal für Temperaturmessung. Im Lieferumfang des Multiplexers ist ein Thermoelement (**T**hermo**C**ouple) des Typs **J** enthalten.
ROUT	**ROUT**e	Befehl für das Konfigurieren von zu beobachtenden / zu messenden Kanälen. Die gemessenen Werte werden im nicht flüchtigen Speicher des Gerätes abgelegt und stehen dort zur weiteren Verwendung zur Verfügung.
FETC	**FETC**h	Befehl um die im nicht flüchtigen Speicher des Multiplexers abgelegten Werte auszulesen (der Speicher wird danach geleert)
CALC:AVER	**CALC**ulate**: AVER**age	es wird der Mittelwert, der im nicht flüchtigen Speicher des Gerätes abgelegten Werte für die angesprochenen Kanäle berechnet und an die Schnittstelle gesendet

ROUT:MON	**ROUT**e:**MON**itor	Konfigurieren des Monitors = online Anzeige der gemessenen Werte auf dem Display des Multiplexers. Mögliche Optionen : **STAT**us **ON** oder **OFF** = Monitor für einen bestimmten Kanal ein- oder ausschalten. Die werkseitige Einstellung ist zunächst OFF. Wird das Monitoring eingeschaltet, so kann der Kanal mit dem Drehknopf auf dem Frontpanel des Multiplexers ausgewählt werden
ROUT:SCAN	**ROUT**e:**SCAN**nen	mit diesem Befehl wird die SCAN-Liste festgelegt. z.B. bedeutet, „(@101:112), dass die Kanäle 101 bis 112 in die SCAN-Liste aufgenommen werden.
TRIG	**TRIG**ger	Befehl zur Konfiguration des Triggers.
TRIG:TIM	**TRIG**ger:**TIM**e	Befehl zur Messung über ein bestimmtes Zeitinterval
TRIG:COUN	**TRIG**ger:**COUN**t	Befehl zur Messung über eine bestimmte Anzahl von Messpunkten
INIT		Startet den Trigger. Wenn keine Parameter angegeben wurden, die das Gerät dazu veranlassen auf z.B. eine bestimmte Uhrzeit zu warten, und eine SCAN-Liste konfiguriert ist, dann startet das Gerät sofort.

Aus Bild 20: wird auch die Messkonfiguration ersichtlich: Die SCAN-Liste umfasst 12 Kanäle (101 bis 112), von denen 11 Kanäle (102 bis 112) für eine Gleichspannungsmessung im Bereich von 0 bis 10 V und einer Auflösung von 0.001 V konfiguriert sind und einer (Kanal 101) für eine Temperaturmessung mit einem Thermoelement Typ J. Es handelt sich dabei um das im Lieferumfang enthaltene Thermoelement. Der Monitor wird bei der Initialisierung auf Kanal 101 gesetzt, so dass der Multiplexer standardmäßig die Lufttemperatur anzeigt.

3.4.4 Synchronisierung

Wie schon in Kapitel 3.3.4 erwähnt wurde, muss das Programm in der Zeit bis der Multiplexer die Daten an den Eingangspuffer der RS232 Schnittstelle sendet, in einen Wartezustand versetzt werden, da der Multiplexer sonst die laufende Messung mit einer Fehlermeldung abbricht. Zwar existiert in LabVIEW das Objekt „Warten (ms)", das die Ausführung des Programms für eine bestimmte (wählbare) Zeit anhält, dieses Objekt hat aber den Nachteil, dass in dieser Zeit das Frontpanel nicht auf Benutzereingaben reagiert und dadurch der Eindruck entsteht, das Programm funktioniere nicht richtig oder sei abgestürzt.

Aus diesem Grund wurde ein Sub-VI entwickelt, das ebenfalls für eine bestimmte Zeitspanne „wartet", jedoch die Ausführung des übrigen Codes nicht unterbricht. Da dieses VI als Sub-VI mit niedriger Priorität ausgeführt wird, ist gewährleistet, dass die wichtigen Tasks die nötige Rechenzeit zur Verfügung haben.

Bild 21: Sub-VI „timewait.vi"

Der Quellcode wird in Kapitel 4.2.11 erläutert.

3.5 Einstellungen

Das Messdatenerfassungsprogramm in den frühen Versionen kam mit relativ wenigen Einstellungen aus. Im Laufe der Entwicklung mussten aber immer mehr Parameter berücksichtigt werden, so dass es schließlich nötig wurde, diese aus dem Hauptprogramm auszulagern, da dieses sonst zu unübersichtlich geworden wäre.

Zur Verwaltung der Parameter wurde das Sub-VI „config_sub.vi" geschrieben:

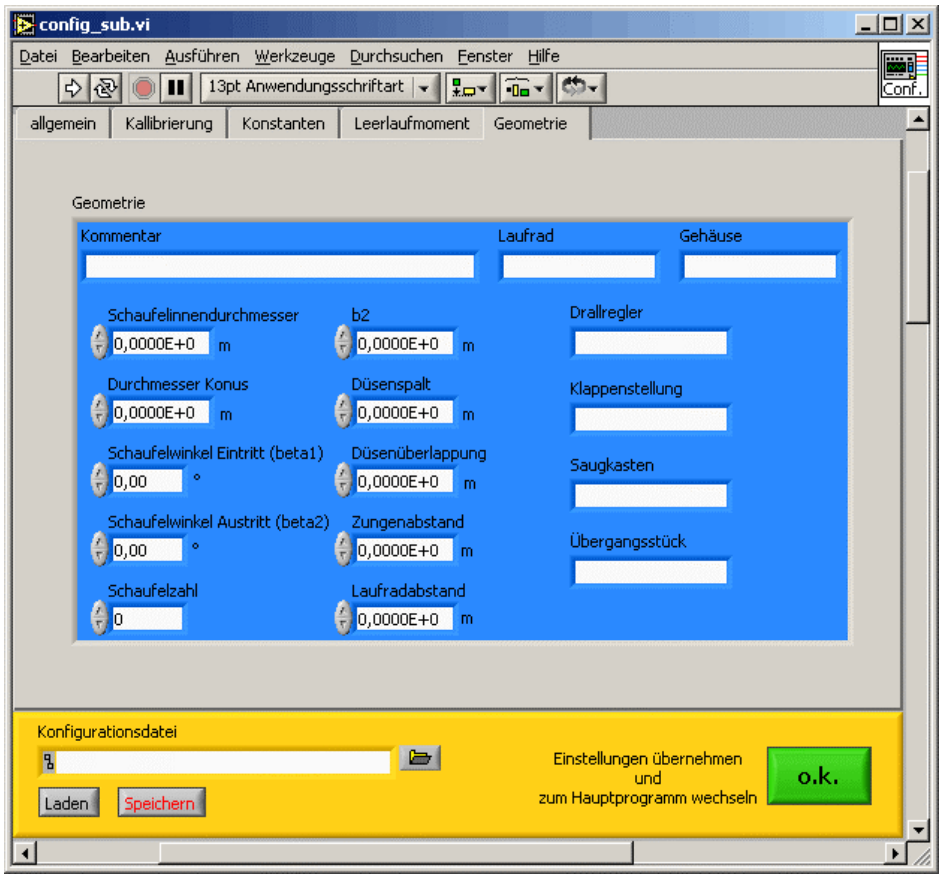


Bild 22: Frontpanel des Sub-VI „config_sub.vi"

Dieses Sub-VI wird beim Start des Messdatenerfassungssystems automatisch geöffnet und bleibt so lange im Vordergrund, bis der Button „o.k." gedrückt wird. In den einzelnen Registern kann der Benutzer die notwendigen Einstellungen vornehmen :

allgemein

- COM-Port
- Einstellungen zur Umwandlung der Strings in Zahlen
- Arbeitspfad (für die Dateiausgabe)

- Ordnungsmerkmal (Bestandteil des Dateinamens)
- Trennzeichen – entweder ein Leerzeichen (=formatierter Text) oder ein Semikolon (= *.csv – Datei)

Kalibrierung	In diesem Feld werden die Faktoren und Offsets für die einzelnen Kanäle eingegeben.
Konstanten	Stoffwerte des Fördermediums, sowie Angaben zur Geometrie des Laufrades, der Rohrleitung und des Messgerätes
Leerlaufmoment	Koeffizienten für das Polynom des Leerlaufmoments
Geometrie	Weitere Angaben zur Geometrie des Prüfstandes. Diese Angaben werden zur Berechnung in diesem Programm aber nicht benötigt.

Darüber hinaus hat der Benutzer die Möglichkeit die eingegebenen Daten in einer *.ini – Datei zu speichern und die gespeicherten Werte wieder einzulesen. Dieses Feature ist z.B. beim Neustart des Programms nützlich – man erspart sich den Aufwand alle Parameter neu eingeben zu müssen. Dem User steht die Wahl des Dateinamens frei, sinnvoller weise wählt man z.B. jedoch einen Namen, der zum Ordnungsnamen des Messdatenfiles passt, oder Aufschluss über die Prüfstandskonfiguration gibt.

3.6 Verarbeitung der eingelesenen Messgrößen

3.6.1 Umwandlung der Strings in Zahlen

Die am COM-Port eingelesenen Daten liegen als Strings vor. Der Multiplexer verwendet dabei standardmäßig das wissenschaftliche Zahlenformat mit 8 Nachkommastellen, d.h., eine gemessene Spannung von 23,5 Volt wird angegeben als 2.35000000E+1. Damit diese Spannungen zu Messgrößen umgewandelt werden können, müssen die Strings in Zahlen umgewandelt werden. Dies geschieht in dem Unterprogramm „string2num_sub.vi".

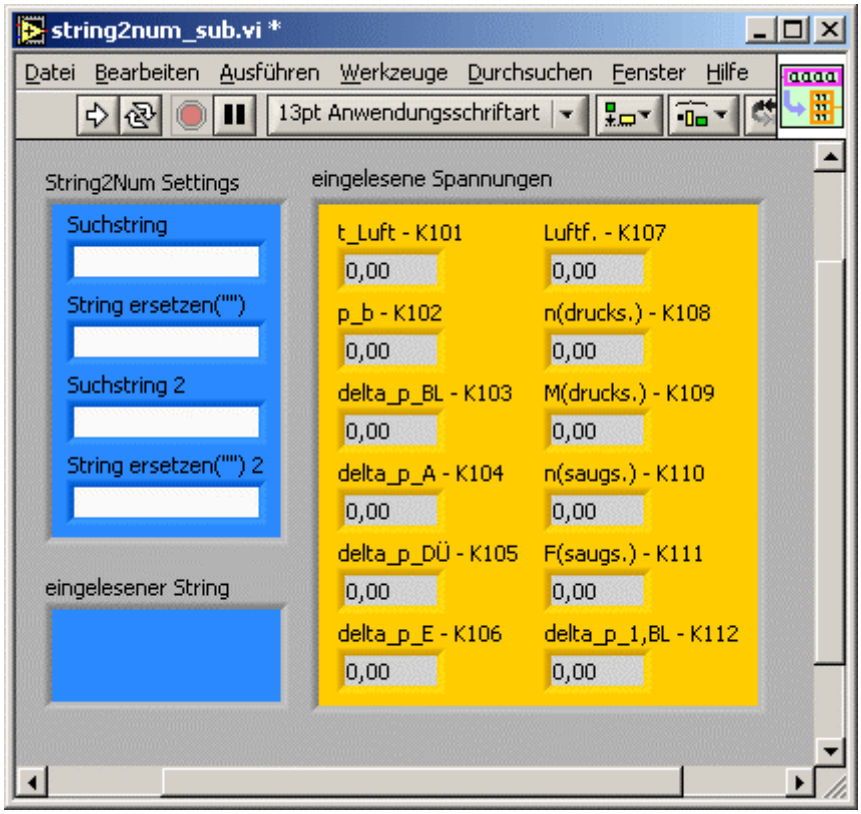


Bild 23: Sub-VI „string2num_sub.vi"

Das Hauptprogramm übergibt an dieses Sub-VI den eingelesenen String und die Einstellungen zur Umwandlung und erhält nach Abarbeitung des Sub-VIs Zahlenwerte („eingelesene Spannungen") zurück.

Während der Programmierung ist aufgefallen, dass mit Einstellungen der verwendeten LabVIEW-Objekte Probleme auftraten, wenn der Rechner gewechselt

wurde. Diese Probleme sind darauf zurückzuführen, dass es möglich ist, in den Systemeinstellungen der jeweiligen Plattform das Zeichen für den Dezimalpunkt zwischen Komma und Punkt umzustellen und der vom Multiplexer gesendete String Kommas als Trennzeichen zwischen den einzelnen Werten verwendet. Dies führt unter Umständen zu einer falschen Zuordnung oder dazu, dass Nachkommastellen ignoriert werden und die weitere Berechnung somit falsch ist. Als Abhilfe wurden die „String2Num Settings" eingeführt, mit denen der Benutzer die Möglichkeit hat vor der Umwandlung das Trennzeichen und den „falschen" Dezimalpunkt zu filtern und durch andere, „richtige" Zeichen zu ersetzen.

Unter Umständen könnte man auf dieses Unterprogramm auch verzichten. Allerdings müsste dann sicher gestellt sein, daß das System auf die entsprechende Ländereinstellung (amerikanisch) konfiguriert ist. Gerade dieses kann aber eventuell wiederum bei anderen Anwendungen zu Problemen führen und wird darum vom Bediener des PCs nicht gewünscht. Somit wäre das Programm auf so einem System nicht mehr lauffähig, was bedeuten würde, dass man extra für diese Anwendung einen eigenen PC zur Verfügung stellen müsste. Die Filterung der Zeichen im Programm ist somit die elegantere und flexiblere Lösung.

3.6.2 Umwandlung der eingelesenen Spannungen in Messgrößen

Um aus den eingelesenen Spannungen Messgrößen zu machen, muss die Spannung – wenn von einem linearen Übertragungsverhalten des Sensors ausgegangen wird – durch Multiplikation mit einem Faktor und Addition eines Summanden in eine Messgröße umgewandelt werden. Diese Aufgabe übernimmt das Sub-Vi „kalib_sub.vi“:

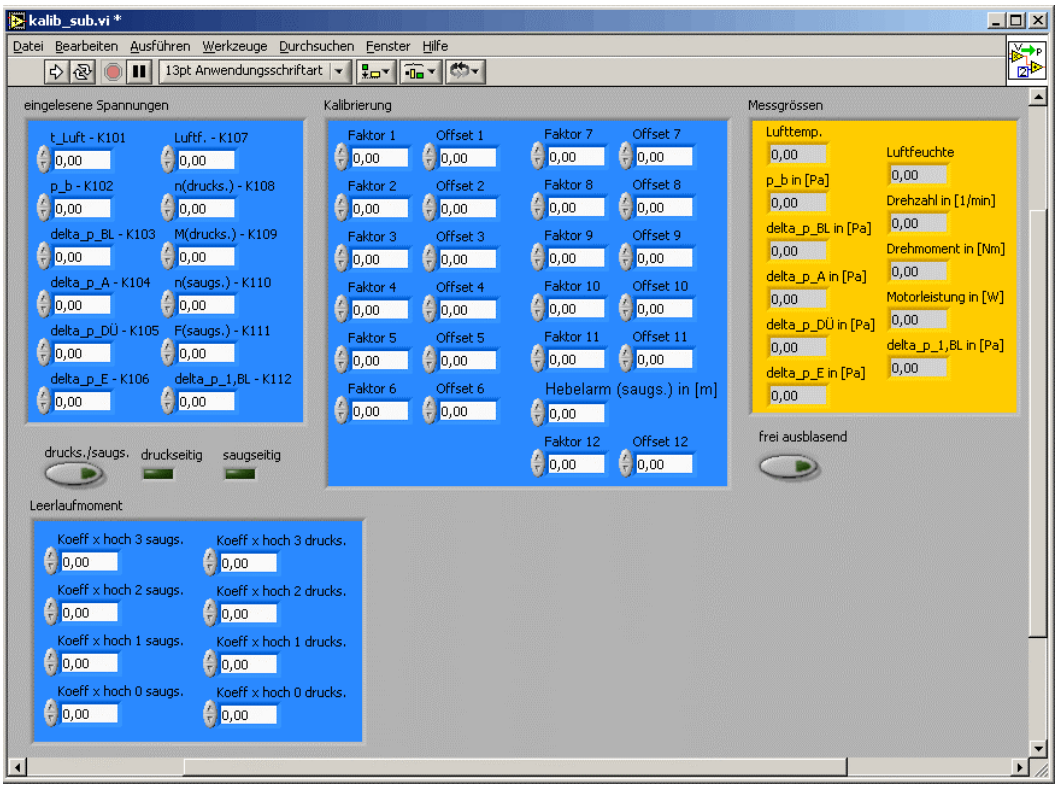


Bild 24: Sub_VI „kalib_sub.vi“ (verkleinerte Darstellung)

Die Eingangsvariablen (blau) sind: die eingelesenen Spannungen, die Kalibrierung (bestehend aus einen Faktor und einem Offset pro Kanal), die Koeffizienten für das Polynom für des Leerlaufmoments und die Schalter, mit denen man zwischen druckseitigem- und saugseitigem sowie frei ausblasendem Kanalprüfstand auswählen kann.

Bei Auswahl von „saugseitigem Prüfstand“ werden die Größen Blendenwirkdruck und Differenzdruck am Eintritt zu –1 Pa gesetzt, die Eingänge für Drehzahl und Drehmoment druckseitig werden ignoriert.

Bei Auswahl von „duckseitigem Prüfstand" wird die Größe Düsenwirkdruck zu –1 Pa gesetzt, die Eingänge für Drehzahl und Drehmoment saugseitig werden ignoriert. Wenn zusätzlich „frei ausblasend" angewählt ist, dann wird auch noch der Differenzdruck am Austritt auf –1 Pa gesetzt.
Aus Drehmoment und Drehzahl (druckseitig) bzw. Drehzahl und Kraft mal Hebelarm (saugseitig) wird in diesem VI die an den Ventilator abgegebene Motorleistung berechnet. Dabei wird die Lagerreibung durch ein Polynom 3. Grades berücksichtigt. Die Koeffizienten dieses Polynoms müssen in einem Leerlaufversuch ermittelt werden.
Zur automatischen Bestimmung der Werte für die Kalibrierung steht das VI „kalib.vi" zur Verfügung. Hier ist darauf zu achten, dass Größen, die vom Messgerät in einer anderen Einheit als Pa ausgegeben werden, umgerechnet werden müssen, da das Hauptprogramm und die weiteren Unterprogramme nur mit der Einheit Pa bei Druck-Größen rechnen. Das gleiche gilt natürlich, wenn die Kalibrierung manuell (z.B. aus einem Kalibrierschein) eingegeben wird!

3.6.3 aerodynamische Berechnungen

Die aerodynamischen Berechnungen sind alle in dem Sub-VI „berechnung_sub.vi" zusammengefasst. Die verwendeten Berechnungsgleichungen sind in Kapitel 2 zu finden.

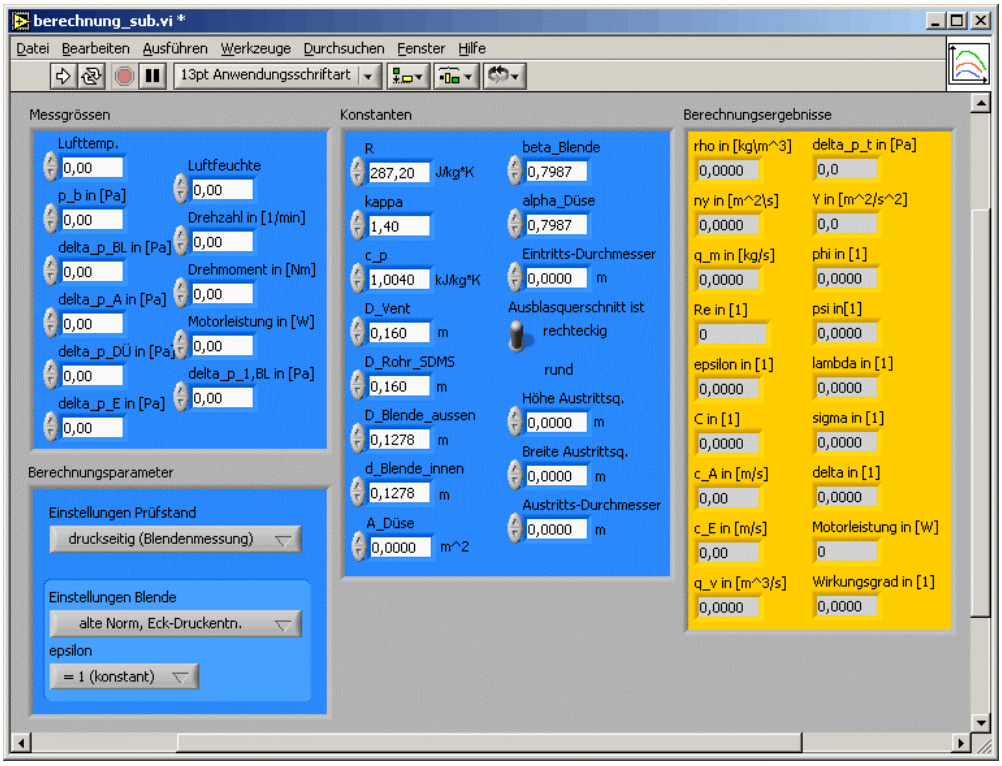


Bild 25: Sub-VI „berechnung_sub.vi" (verkleinerte Darstellung)

Die Eingangsgrößen dieses VIs sind die Cluster „Messgrößen", „Konstanten" und „Berechnungsparameter". Die Ausgangsgröße ist der Cluster „Berechnungsergebnisse".

In dem Cluster „Berechnungsparameter" werden an das Sub-VI Informationen über die Prüfstandkonfiguration übergeben. Hier wird eingestellt, ob es sich um einen druckseitigen Prüfstand (mit Blendenmessung), um einen saugseitigen Prüfstand (mit Messdüse zur Volumenstrombestimmung) oder um einen saugseitigen, frei ausblasenden Prüfstand (mit Messdüse) handelt. Des weiteren hat der Benutzer die Möglichkeit zwischen folgenden Berechnungsverfahren bei der Blendenmessung auszuwählen:

- Berechnung von C nach DIN EN ISO 5167-1 vom November 1995 (nur Eck-Druckentnahme)

- Berechnung von C nach DIN EN ISO 5167-1/A1 vom Februar 1998 für
    - Eck-Druckentnahme
    - Flansch-Druckentnahme
    - und D oder D/2 Druckentnahme
- Berechnung mit konstantem $\varepsilon = 1$ (Expansionszahl),
  Verzicht auf die Bestimmung des Differenzdrucks zum Umgebungsdruck vor der Blende ($\Delta p_{1,BL}$)
- Berechnung von ε mit $\Delta p_{1,BL}$, d.h., die Kompressibilität des Mediums wird berücksichtigt.

Die an das Berechnungsprogramm übergebenen Konstanten sind zum einen konstante Stoffwerte, zum anderen Angaben zur Geometrie:

Bezeichnung	Wert und Einheit	Beschreibung
R	287,2 $\mathrm{J}\frac{\mathrm{kg}}{\mathrm{K}}$	Gaskonstante der trockenen Luft
kappa	1,4	Isentropenexponent
c_p	1,004 $\frac{\mathrm{kJ}}{\mathrm{kg}\cdot\mathrm{K}}$	spezifische Wärmekapazität der Luft
D_Vent	variabel, m (Meter)	Durchmesser des Ventilatorlaufrades
D_Rohr_SDMS	variabel, m	Durchmesser des Rohres an der **s**tatischen **D**ruck**m**ess**s**telle
D_Blende_aussen	variabel, m	Außendurchmesser der Blende, entspricht meistens dem Rohrdurchmesser
d_Blende_innen	variabel, m	Durchmesser der Blendenöffnung
A_Düse	variabel, m²	Fläche der Düsenöffnung
beta_Blende	variabel	Durchmesserverhältnis Düse
alpha_Düse	variabel	Durchflusszahl der Düse
Eintritts-Durchmesser	variabel, m	Durchmesser der Eintrittsöffnung
Austritts-querschnitt	rund oder rechteckig	dementsprechend sind Höhe und Breite oder der Durchmesser anzugeben

Die Eingangsgröße „Messgrößen" und die Ausgangsgröße „Berechnungsergebnisse" bedürfen keiner ausführlichen Erläuterung.

3.6.4 Speicherung der Daten

Zur Speicherung der Messdaten stehen 2 Dateiformate zur Auswahl:

- formatierter Text mit einem Leerzeichen zwischen Zahlenwerten (*.prn – Datei)
- formatierter Text mit einem Semikolon als Trennzeichen zwischen den Zahlenwerten (*.csv – Datei)

Dies bedeutet, dass sämtliche Zahlenwerte, vor der Ausgabe in die Datei, in Strings umgewandelt werden.

Beide Formate sind transparent, da man zum einen die Möglichkeit hat die Datenfiles mit einem Text-Editor einzusehen (und auch zu verstehen !), zum anderen sind die Daten in den Files dokumentiert, d.h., vor jeder Größe steht der Name.

Das *.prn – Format wurde gewählt um die Daten in eine Form zu bringen, die auch von anderen Hochsprachen (z.B. C oder C++) möglichst problemlos eingelesen werden kann.
Das *.csv – Format wurde gewählt, damit die Daten möglichst einfach in eine Tabellenkalkulation eingelesen werden können. Zumindest mit Microsoft Excel funktioniert dies ohne weitere Konvertierung[6], wobei die als Strings gespeicherten Zahlenwerte von Excel automatisch als Zahlen interpretiert werden. Dies ermöglicht eine schnelle Weiterverarbeitung der Daten, ohne dass diese extra konvertiert werden müssen.

[6] wenn beim Öffnen das *.csv – Format ausgewählt wurde

3.7 Aufzeichnung akustischer Daten

Auf Wunsch des Anwenders (durch Betätigen des Schalters „Sound Automatik") wird für die Dauer eines Messzyklus mit der Soundkarte des PC und einem oder zwei daran angeschlossenen Mikrofonen eine Audio-Datei aufgezeichnet und im *.wav-Format (= Zeitsignal) auf die Festplatte gespeichert.

Der Anwender kann die Sampling Rate und die Auflösung wählen. Für eine möglichst genaue Aufzeichnung der Daten wird die unten dargestellte Einstellung empfohlen:

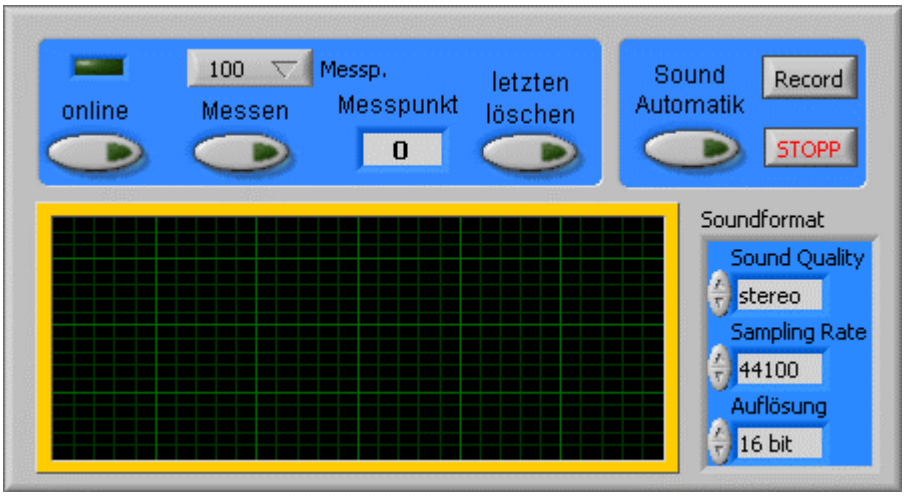


Bild 26: Bedienelemente zur Sound-Aufnahme

Die Datei wird in dem angegebenen Arbeitsverzeichnis mit folgendem Dateinamen gespeichert: `[Ordnungsname]`**`mp`**`[Nummer des Messpunktes]`**`-sound.wav`**

Während der Aufzeichnung (d.h. auch während der Messung) wird eine online FFT der eingelesenen Daten durchgeführt. Die Ergebnisse der FFT werden jedoch (noch) nicht weiterverarbeitet.

4 Erläuterung des Quellcodes

Nachdem in Kapitel 3 die Funktion des Programms erläutert wurde, soll nun in diesem Kapitel auf den Quellcode eingegangen werden. Es wird bewusst darauf verzichtet jedes LabVIEW Objekt einzeln zu erklären, viel mehr sollen der Zusammenhang zwischen den Objekten und die wichtigsten Abläufe und Routinen ausführlich erklärt werden. Für die Beschreibung der einzelnen verwendeten Objekte ist auch die in LabVIEW integrierte Kontexthilfe deutlich besser geeignet als ein langer Text, da dort die benötigten Informationen praktisch per Mausklick zur Verfügung stehen.

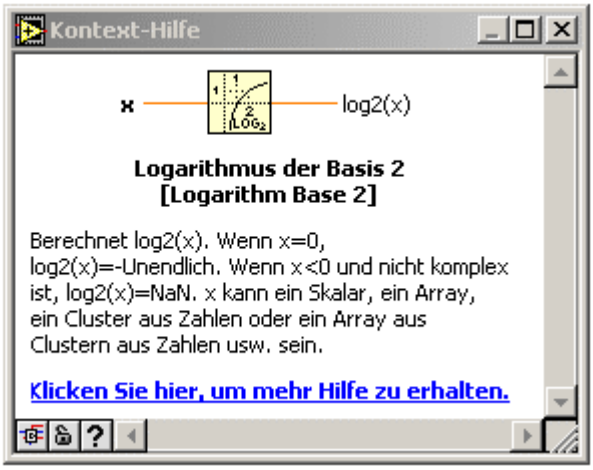


Bild 27: LabVIEW Kontexthilfe zum Objekt „Logarithmus zur Basis 2"

4.1 Liste der zum Programm gehörenden Dateien

Die folgende Tabelle dient als Übersicht über alle zum Messdatenerfassungssystem gehörenden Dateien und Unterprogramme und beschreibt kurz die Funktion:

Dateiname	Typ	Beschreibung
2kanalfft_flattop_sub.vi	Sub-VI	berechnet online die FFT eines Zeitsignals unter Verwendung der Flat-Top-Fensterfunktion
2kanalfft_sub.vi	Sub-VI	berechnet online die FFT eines Zeitsignals unter Verwendung der Hanning-Fensterfunktion
aamdes_0.9.3d.vi	VI	Hauptprogramm des Messdatenerfassungssystems
aamdes_0.9.3sim.vi	VI	Hauptprogramm, jedoch wird hier ein der Prüfstand „simuliert", d.h. man kann mit dem Programm ohne angeschlossene Messhardware (Multiplexer) „messen"
agilent_steuerung_sub.vi	VI	stellt die Steuerstrings in SCPI für den Multiplexer zur Verfügung und errechnet die notwendigen Wartezeiten (Synchronisierung)
array_sort.vi	Sub-VI	Sortiert ein 2D Array nach einer wählbaren Spalte oder Zeile
auswertung.vi	Sub-VI	wie der Name schon sagt, werden mit diesem VI die gemessenen Ergebnisse ausgewertet. Das Frontpanel wird bei Aufruf geöffnet, so dass der User die nötigen Eingaben vornehmen kann. Dieses VI ermöglicht das Betrachten der verschiedenen gemessenen Größen in einem XY-Diagramm, dazu werden die Koeffizienten eines an diese Kurve angepassten Polynoms (wählbaren Grades) ausgegeben.
berechnung_sub.vi	Sub-VI	in diesem VI werden alle nötigen aerodynamischen Berechnungen durchgeführt
config_sub.vi	Sub-VI	Verwaltungsprogramm für alle Parameter und Konstanten

geo_read_sub.vi	Sub-VI	liest einen bestimmten Teilbereich aus einer Konfigurationsdatei
kalib.vi	VI	Hilfsprogramm zum Erstellen einer Kalibrierkurve
kalib_global.vi	globale Variable	Globale Variable, in der die Kalibrierwerte gespeichert werden
kalib_sub.vi	Sub-VI	Unterprogramm in dem, aus den eingelesenen Spannungen mit Hilfe der Kalibrierung, die Messwerte erzeugt werden
simulator.ini	config. Datei	Konfigurationsdatei für das Hauptprogramm in dem die nötigen Einstellungen für den Simulationsbetrieb (aamdes_0.9.4sim.vi) gespeichert sind
simulator.vi	Sub-VI	Hilfsprogramm, das den Prüfstand simuliert und einen String ausgibt, den der Multiplexer bei der Messung senden würde
speicher_array.vi	globale Variable	Globale Variable in der die in der Betriebsart "Messung" aufgenommenen Werte gespeichert und zur Weiterverarbeitung (in auswertung.vi) bereitgestellt werden
standard.ini	config. Datei	Konfigurationsdatei mit Standardeinträgen. Dient als Vorlage zur Erstellung eigener Konfigurationsdateien.
string2num_sub.vi	Sub-VI	wandelt die eingelesenen Strings in Zahlenwerte um
testfhd01.ini	config. Datei	Konfigurationsdatei für den Betrieb des Messdatenerfassungssystems am Ventilatorprüfstand an der FH Düsseldorf
timewait.vi	Sub-VI	Sub-Vi, das – in einer Sequenz eingesetzt – für eine bestimmte (wählbare Zeit) wartet und somit den Sprung in den darauffolgenden Rahmen verzögert – OHNE die Ausführung des übrigen Codes zu beeinflussen.
umwandlung_csv_sub.vi	Sub-VI	Umwandlung von Zahlen in formatierten Text.

4.2 Quellcode

Der Quellcode ist in manchen Fällen zu groß, um als Grafik in diesen Text eingefügt zu werden. Darum werden hier nur Fragmente dargestellt. Es ist empfehlenswert, den Quellcode beim Lesen dieses Textes zu öffnen, um die hier beschriebenen Schritte nachvollziehen zu können.

4.2.1 Hauptprogramm

In der Datei „aamdes_0.9.3d.vi" ist der Code für das Hauptprogramm enthalten. Wie schon in Kapitel 3.3.3 beschrieben besteht die grobe Struktur dieses Programms aus einer Sequenz mit 4 Rahmen und einer while-Schleife in Rahmen 3, die das eigentliche Messprogramm enthält, und so lange ausgeführt wird, bis der User das Programm mit dem „Ende" Button beendet.
Der erste Rahmen (Index 0) ist noch relativ einfach zu verstehen. Er enthält nur das Sub-VI „config_sub.vi", dessen Frontpanel (vgl. Kapitel 3.5) beim Aufruf geöffnet wird, damit der Benutzer die nötigen Eingaben vornehmen kann.

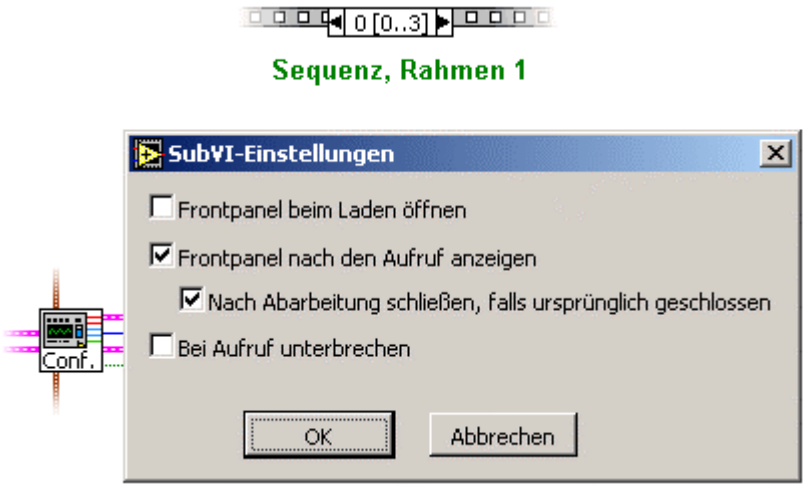


Bild 28: Sub-VI „config_sub.vi" im Rahmen 1 der äußeren Sequenz des Hauptprogramms

Dieses Sub-Vi übergibt nach der Abarbeitung die benötigten Parameter an das Hauptprogramm, wo sie (versteckt) angezeigt werden und mittels lokaler Sequenzvariablen an die nachfolgenden Rahmen übergeben werden.

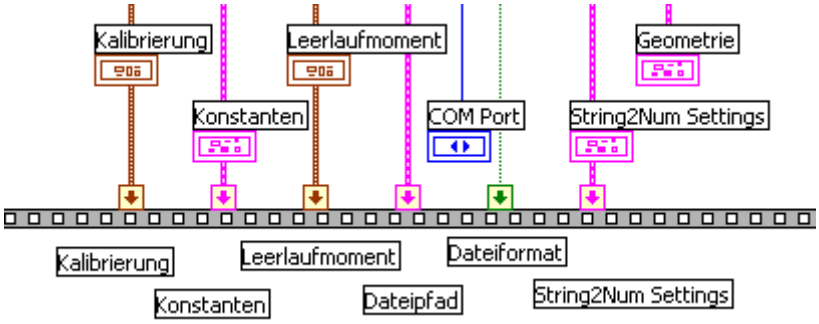


Bild 29: Anzeige der Parameter und Übergabe mittels lokaler Sequenzvariable

Im zweiten Rahmen der Sequenz befindet sich schon deutlich mehr Code. In den oberen 2/3 ist die Routine zu finden, mit dem der Kopf der Ausgabedatei erstellt wird. Dazu werden alle Konstanten und die Kalibrierung in Strings umgewandelt und mit Namen versehen:

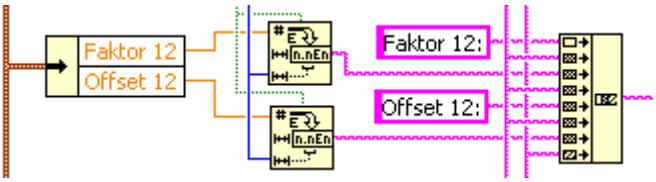


Bild 30: Erstellen der Strings für den Kopf der Ausgabedatei

In Bild 30: ist der Code zur Umwandlung des Kalibrierfaktors 12 und des dazugehörenden Offset 12 dargestellt. Dazu wird der Cluster „Kalibrierung" nach Namen aufgeschlüsselt, und die einzelnen Werte werden jeweils an ein Objekt übergeben, das die Zahl in einen String in Exponential-Format umwandelt. Danach werden die einzelnen Strings zu einem einzigen (längeren) String verknüpft. Die Exponentialnotation wurde gewählt, um sicherzustellen, dass auch bei kleinen Faktoren, Offsets oder Konstanten keine (wichtigen) Nachkommastellen verloren gehen. Diese Operation wird für die ganze Kalibrierung und alle Konstanten durchgeführt. Anschließend werden alle diese (schon etwas größeren) Einzelstrings nochmals zu einem einzigen String zusammengefasst und in eine Text-Datei geschrieben.

Das Ergebnis bei Verwendung des Semikolons als Trennzeichen sieht folgendermaßen aus:

	A	B	C	D	E
35					
36	Messp.	Lufttemp [°C]	p_b [Pa]	delta_p_BL [Pa]	delta_p_A [Pa]
37	0	2,13E+01	1,01E+05	4,66E+00	2,58E+03
38	1	2,13E+01	1,01E+05	4,03E+01	2,66E+03
39	2	2,13E+01	1,01E+05	1,32E+02	2,73E+03
40	3	2,13E+01	1,01E+05	2,62E+02	2,76E+03
41	4	2,13E+01	1,01E+05	4,35E+02	2,75E+03
42	5	2,13E+01	1,01E+05	6,77E+02	2,71E+03

Bild 31: Ausschnitt aus der Ausgabedatei - Betrachtung in Excel

Anhand der Tabelle in Bild 31: kann man auch die Struktur des Speicherarrays erkennen.

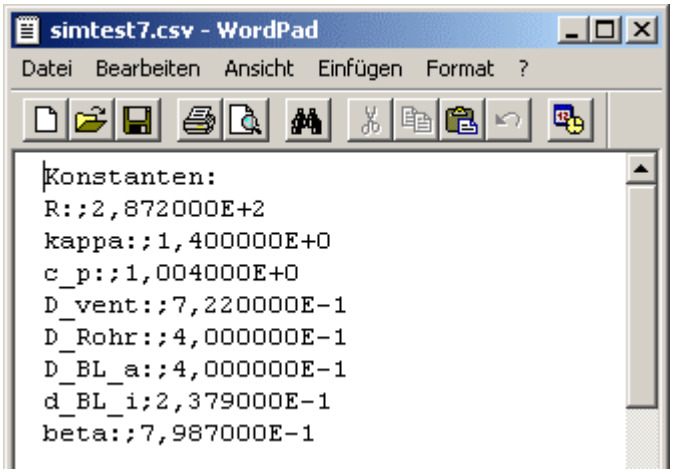


Bild 32: Ausschnitt aus der Ausgabedatei – Betrachtung mit Wordpad

Ebenfalls in Rahmen 2 befindet sich der Code mit dem die RS232 Schnittstelle angesprochen wird (vergleiche Kapitel 3.4.1) und einige Größen bzw. Bedienelemente zu Null gesetzt werden:

Bild 33: die Variablen „Messpunkt" und „Speicher Array" werden zu Null gesetzt

In Bild 33: ist die Initialisierung der Variablen „Messpunkt" und „Speicher Array" dargestellt. Das Anzeigeelement „Messpunkt" wird dazu nicht direkt angesprochen, sondern es wird eine lokale Variable mit dem Attribut „Schreiben" verwendet, da das Anzeigelement selbst nur in dem Sequenzrahmen direkt verbunden werden kann, in dem es eingebaut ist. Da die Anzeige des Messpunktes während der Messung erfolgen soll (also in Rahmen 3), kann es nicht in Rahmen 2 vorhanden sein. Es wäre zwar möglich eine lokale Sequenzvariable zu verwenden, was jedoch viel umständlicher wäre als die hier gezeigte Lösung.
Das Element „Speicher Array" ist eine globale Variable. Globale Variablen werden wie ein VI erstellt, beinhalten jedoch keinen Code – nur ein Anzeigeelement. Der Vorteil globaler Variablen ist, dass sie auch in anderen VIs ohne spezielle Übergabe zur Verfügung stehen, es reicht das Symbol in dem jeweiligen VI zu platzieren. Es handelt sich also um eine einfache Methode, um Werte von einem VI an ein anderes zu übertragen.
Von dieser Möglichkeit wird hier Gebrauch gemacht, weil die Daten, die das Hauptprogramm misst, in dem Sub-VI „auswertung.vi" zur Berechnung der Polynomkoeffizienten und zur Anzeige der Kurven benötigt werden. Das besondere daran ist, dass die gemessenen Werte auch nach dem Ende des Hauptprogramms noch in der globalen Variablen vorhanden sind[7], es wäre also durchaus möglich das Hauptprogramm zu beenden, anschließend das Auswerteprogramm zu starten und mit den Werten der letzten Messung den Versuch auszuwerten. Darüber hinaus stehen die Daten auch jedem anderen VI zur Verfügung. Davon kann man z.B. Gebrauch machen, wenn man „einfach mal einen Blick" auf die gemessenen Werte werfen will. Das dazu benötigte VI ist schnell zu programmieren:

Bild 34: Inhalt der globalen Variablen „Speicher Array" anzeigen

Des weiteren werden in Rahmen 2 die Sub-VIs „string2num_sub.vi", „kalib_sub_vi" und „berechnung_sub.vi" referenziert.

[7] zumindest so lange bis erneut gemessen wird, weil die Daten dann überschrieben werden !

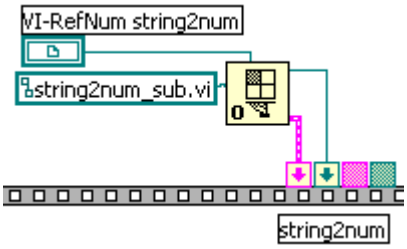


Bild 35: VI-Referenz auf „string2num_sub.vi"

Die Funktion und der Sinn von Call by Reference wurden bereits in Kapitel 3.2.2 beschrieben. Das benötigte Bedienelement (VI-RefNum string2num)

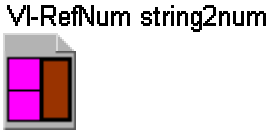


Bild 36: VI – RefNum Bedienelement (normalerweise ausgeblendet)

wird ausgeblendet, da der Benutzer zum einen keine Eingaben vornehmen soll, zum anderen auch gar keine Eingaben vornehmen kann. Dennoch wird es benötigt, um das Objekt „VI-Referenz öffnen" anzusteuern. Dies geschieht aber im Quellcode bei der Programmierung durch Auswahl des VI-Servers[8] und nicht im normalen Betrieb.

In Rahmen 3 befindet sich das eigentliche Messprogramm. Wie bereits in Kapitel 3.3.4 beschrieben, besteht dieses aus einer großen while-Schleife, die so lange wiederholt wird, bis der Benutzer den Button „Ende" betätigt. Der Ablauf des Messprogramms soll nun anhand eines Schleifendurchlaufes erklärt werden.

Der Schleifendurchlauf beginnt mit einer aus 8 Rahmen bestehenden Sequenz, die den Multiplexer mit den durch das Sub-VI „agilent_steuerung_sub.vi" bereitgestellten Steuerstrings steuert und die gesendeten Daten aus dem Eingangspuffer der RS232 Schnittstelle ausliest. Diese Sequenz befindet sich in Rahmen 3 oben links:

[8] die Auswahl erreicht man mittels Rechtsklick

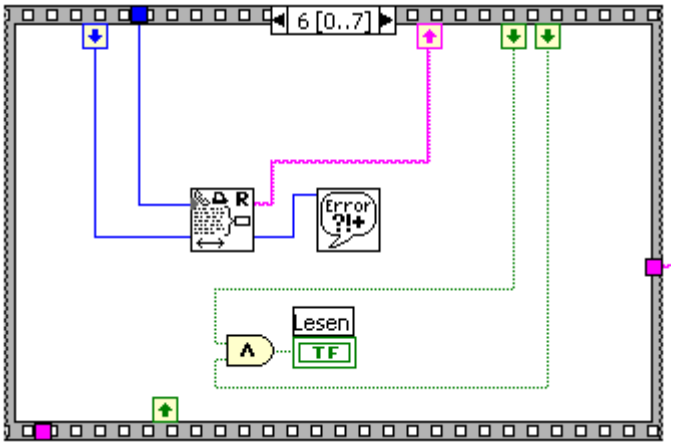


Bild 37: Steuerungs- / Einlesesequenz – Rahmen 7: Auslesen der Strings aus dem Puffer

Struktogramm dieser Sequenz:

“Record” und “Stopp” auf “false” setzen		
Steuerstring 1 an den Multiplexer senden	Sound Automatik: ein (true)	Sound Automatik: aus (false)
	Aufnahme starten	*tue nichts*
Aufruf Sub-VI “timewait.vi” = warte für eine bestimmte Zeit		
Steuerstring 2 an den Multiplexer senden		
Aufruf Sub-VI “timewait.vi” = warte für eine bestimmte Zeit		
Anzahl der Zeichen im Eingangspuffer ermitteln	Aufnahme beenden	
diese Anzahl Zeichen aus dem Eingangspuffer einlesen		
Zeichenkette ausgeben		

Bild 38: Struktogramm der Steuerungs- / Einlesesequenz

Beim Betrachten der Steuer- / Einlesesequenz losgelöst von dem Sub-VI „agilent_steuerung_sub.vi“ mag es eventuell unlogisch erscheinen, dass die Wartezeiten von der Fehlerabfrage und der Messung addiert, sowie die Steuerstrings aus Fehlerabfrage und Messung verknüpft werden. Da jedoch Steuerstrings bzw. Wartezeit für den jeweils nicht angewählten Zustand leer bzw. Null sind, war dieser

Weg die einfachste Methode ohne aufwendige Programmierung von Case-Strukturen, o.Ä. das gewünschte Ziel zu erreichen.

Als nächstes durchlaufen die eingelesenen Daten nun eine Art „Weiche". Wenn der Benutzer den Button „Fehler anzeigen" gewählt hat, dann werden die eingelesenen Strings ohne weitere Verarbeitung in der Anzeige angezeigt, an das weitere Programm wird ein leerer String übergeben. Ist der Button „Fehler anzeigen" aus, d.h. die empfangenen Daten sind Messwerte, dann werden diese Daten zur Weiterverarbeitung an die nachfolgende Struktur übergeben.

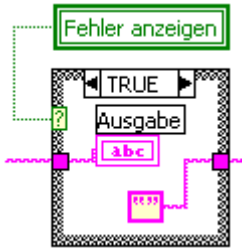


Bild 39: „Datenweiche" im Hauptprogramm

Nach passieren der „Weiche" werden die Daten an eine weitere, aus 3 Rahmen bestehende Sequenz übergeben. In dieser Sequenz wird mit Hilfe der referenzierten Sub-VIs der Hauptteil der Rechenarbeit geleistet.

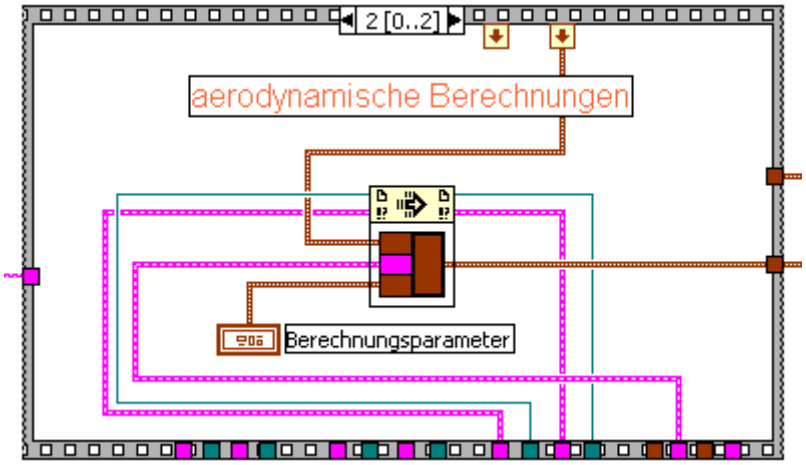


Bild 40: Sequenz mit referenzierten Sub-VIs zur Berechnung

In Rahmen 1 werden die Strings in Zahlen umgewandelt, in Rahmen 2 werden mit Hilfe der Kalibrierung aus den Spannungen die Messwerte errechnet und in Rahmen 3 werden anschließend die aerodynamischen Berechnungen durchgeführt.

Da die Sub-VIs „berechnung_sub.vi" und „kalib_sub.vi" die Ergebnisse als jeweils einen Cluster zurückgeben, werden die Cluster nach Namen aufgeschlüsselt, bevor die einzelnen Werte an die entsprechenden Anzeigeelemente übergeben werden.

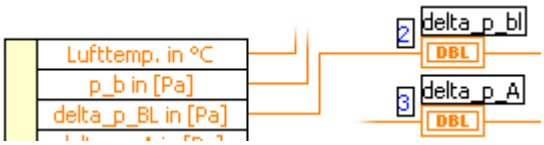


Bild 41: nach Namen aufgeschlüsselter Cluster mit Anzeigeelementen

Die kleinen blauen Zahlen sind die Spaltenindices des Arrays, in das die Werte gespeichert werden. Diese sind nicht mit dem Anzeigeelement verbunden, sondern einfach ein „von Hand" eingefügter Text. Beim Ändern des Quellcodes ist es sehr empfehlenswert, die Anzeigeelemente zusammen mit dieser Markierung zu verschieben. Das Zusammenfügen evtl. getrennter Verbindungen kann ohne dieses Hilfsmittel sehr kompliziert werden, da diese Nummerierung strikt eingehalten werden muss. Falls dies aus irgendeinem Grund nicht möglich sein sollte, müssen die Dateiausgabe und das Auswerteprogramm angepasst werden, da diese mit dieser Reihenfolge arbeiten.

Nachdem die gemessenen und errechneten Größen angezeigt wurden, werden diese zu einem eindimensionalen Array zusammengefügt und anschließend als neue Zeile in das Speicher-Array und in die Ausgabe-Datei eingefügt. Der Quellcode der Struktur, die die Daten in das Array schreibt ist nicht trivial und wird darum in Kapitel 4.3.2 detailliert erläutert. An dieser Stelle wird darum nicht weiter darauf eingegangen. Das Schreiben in die Datei ist dagegen wieder recht einfach. Die Daten werden in einen String umgewandelt und als neue Zeile in eine Textdatei geschrieben. Da auch hier das Trennzeichen (Leerzeichen oder Strichpunkt) beachtet werden muss und der Code viel Platz beansprucht, wurde die Umwandlung in das Sub-VI „umwandlung_csv_sub.vi" ausgelagert. Die dort ausgeführten Operationen sind denen der Erstellung des Dateikopfes in Rahmen 2 sehr ähnlich.

Parallel zum Speichern im Array und in der Datei läuft die grafische Anzeige der Punkte auf dem Frontpanel. Da das verwendete Anzeigobjekt (XY-Grafik) als Eingangsgröße einen Array aus Clusterarrays erwartet, müssen die Daten vor der Übergabe an das Objekt noch in die entsprechende Form umgewandelt werden.

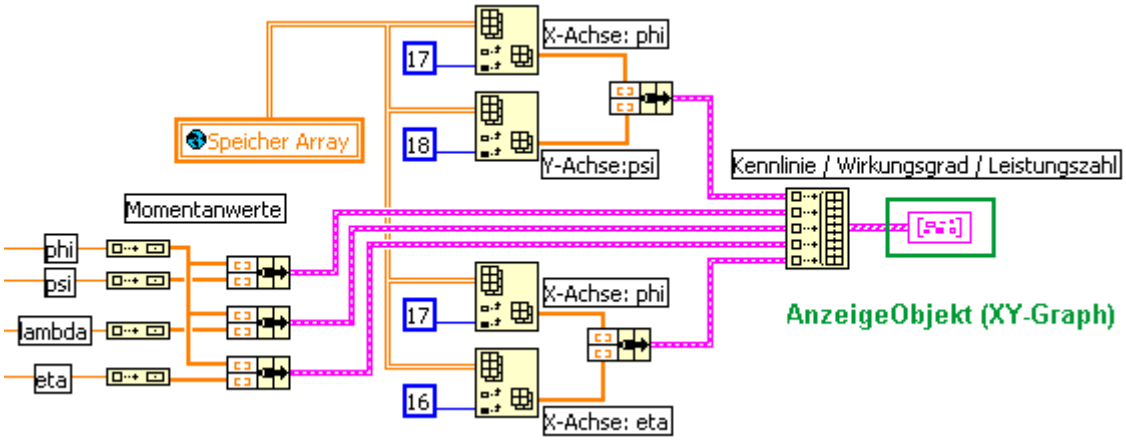


Bild 42: grafische Anzeige auf dem Frontpanel

Für die Anzeige der Momentanwerte Druckzahl über Lieferzahl, Leistungszahl über Lieferzahl und Wirkungsgrad über Lieferzahl werden die berechneten Größen zunächst in ein eindimensionales Array umgewandelt und anschließend in einen Cluster gebündelt. Dabei ist die Reihenfolge zu beachten: der Wert, der auf der X-Achse angezeigt werden soll, muss mit dem obersten Eingang des Objektes „Elemente bündeln" verbunden werden.

Für die Anzeige der gespeicherten Werte wird das Speicher-Array indiziert, d.h., es werden neue eindimensionale Arrays gebildet, die die Werte der Spalte 18 (Index 17 – Lieferzahl), der Spalte 19 (Index 18 – Druckzahl) und der Spalte 17 (Index 16 – Wirkungsgrad) beinhalten. Mit diesen 3 Arrays werden die Cluster Druckzahl über Lieferzahl und Wirkungsgrad über Lieferzahl gebildet.

Diese 5 Cluster werden nun wiederum zu einem Array zusammengefügt und an das Anzeigeelement übergeben, so dass die jeweiligen Momentanwerte für Druckzahl über Lieferzahl, Leistungszahl über Lieferzahl und Wirkungsgrad über Lieferzahl sowie alle bereits gemessenen und gespeicherten Werte für Druckzahl über Lieferzahl und Wirkungsgrad über Lieferzahl zur Anzeige gebracht werden.

Mit der Anzeige der Werte auf dem Display und der Speicherung in dem Speicher-Array ist ein Zyklus aus Messen, Umwandeln, Verarbeiten, Anzeigen und Speichern abgeschlossen. Parallel dazu werden aber noch andere Strukturen verarbeitet:

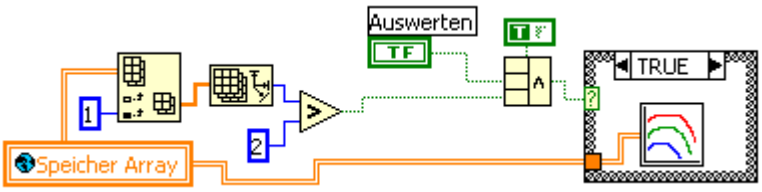


Bild 43: Starten des Sub-VIs „auswertung.vi" bei Betätigung des Schalters „Auswerten"

Zum einen wird das Sub-VI „auswertung.vi" gestartet (und dessen Frontpanel angezeigt), wenn der Benutzer den Schalter „Auswerten" betätigt und bereits mehr als 2 Messpunkte aufgenommen wurden (vorher macht eine Polynomanpassung auch keinen Sinn).

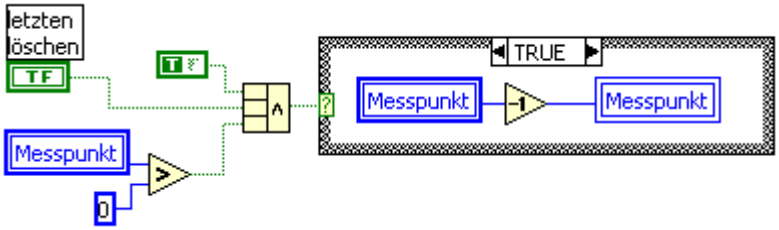


Bild 44: Code zum verringern des Zählers für die Messpunkte

Zum anderen wird bei Betätigen des Schalters „letzten löschen" der Wert der Variablen „Messpunkt" um eins verringert. Da dieser Wert als Zeilenindex für das Speicher-Array verwendet wird, bedeutet dies, dass die Werte des nächsten Messpunktes über die Werte des zuletzt gemessenen geschrieben werden. Dies funktioniert jedoch nur in dem Speicher-Array. Die Zeile in der Dateiausgabe muss nach Beendigung des Programms manuell gelöscht werden, da die Datei immer sofort nach dem Ende eines Messzyklus geschrieben wird. Wird nun ein Messpunkt gelöscht (überschrieben), dann erscheinen in der Ausgabedatei zwei (oder auch mehr) Zeilen mit gleicher Messpunktnummer. Die „richtigen" Werte stehen dann immer in der unteren Zeile. Es ist auch möglich, durch mehrmaliges Betätigen des

Schalters „letzten löschen" den Index um mehr als eins zu verringern. Die Dateiausgabe muss dann dementsprechend manuell korrigiert werden.
Last but not least befindet sich oben rechts in der while-Schleife eine Case-Struktur mit dem Code für die Aufzeichnung der Audiodaten. Dieser Teil des Programms wurde zunächst in einem eigenständigen VI entwickelt und erst nach erfolgreichen Tests in das Hauptprogramm eingefügt. Erklärungen zu dieser Struktur sind im Kapitel 4.3.3 zu finden.

Da sich der Code in dem „Case – true" der Case-Struktur befindet wird dieser Teil des Programms nur ausgeführt, wenn die programmierte Bedingung wahr ist. Dieser Fall tritt ein, wenn der Schalter „Sound Automatik" eingeschaltet ist **und** der Schalter „Messen" gedrückt wird. Da man nicht beide Schalter gleichzeitig betätigen kann, muss man, wenn man Sound aufnehmen möchte, zuerst den Schalter „Sound Automatik" betätigen und dann die Messung starten. Die Länge (in Sekunden) der Aufnahme ist abhängig von der gewählten Anzahl der zu mittelnden Messwerte, da der Sound immer über eine Messperiode (Multiplexer ansteuern - Messung starten – Warten - gemittelte Werte auslesen) aufgenommen wird.

Zusätzlich zur Speicherung der Zeitdaten in einer *.wav-Datei wird eine online FFT der aufgenommenen Signale berechnet. Dies ist möglich – bzw. dazu steht die notwendige Rechenzeit zur Verfügung – weil der Rest des Programms in dieser Zeit durch das Sub-VI „timewait.vi" in einen Wartezustand versetzt wird, der wenig CPU-Leistung beansprucht.

Der letzte Rahmen der Sequenz des Hauptprogramms wird erst dann erreicht, wenn die while-Schleife in Rahmen 3 terminiert weil der „Ende"-Schalter betätigt wurde. Dieser Rahmen dient nur dazu das Programm „sauber" zu beenden, d.h. die geöffneten VI-Referenzen und der COM-Port – Treiber werden mit Hilfe der entsprechenden Objekte wieder geschlossen.

4.2.2 2kanalfft_flattop_sub.vi und 2kanalfft_sub.vi

Die beiden VIs sind fast identisch, beide berechnen mit Hilfe des LabVIEW Objekts „Auto Power Spectrum" das Frequenzspektrum eines Zeitsignals für einen oder 2 Kanäle und geben dieses aus. Der Unterschied liegt in der verwendeten Fensterfunktion. Wie der Name schon andeutet verwendet das Sub-VI „2kanalfft_flattop_sub.vi" die Flat Top Fensterfunktion, das Sub-Vi „2kanalfft_sub.vi" verwendetet dagegen das Hanning Fenster.

Der Quellcode stammt aus einem Beispielprogramm (Sound Card Simple Spectrum Analyer.vi) , das in der LabVIEW Hilfe enthalten ist und fast unverändert übernommen werden konnte. Notwendige Anpassungen waren:

- Die Aufzeichnung der Sound-Daten erfolgt im Hauptprogramm, d.h., das Objekt „sound read" und damit zusammenhängende Objekte wurden entfernt
- es wurde eine Case-Struktur eingebaut, die bei der Berechnung die verschiedenen möglichen Formate (8 oder 16 bit, stereo oder mono) berücksichtigt
- Die grafische Ausgabe wird vom aufrufenden VI übernommen

4.2.3 agilent_steuerung_sub.vi

Das Sub-VI „agilent_steuerung_sub.vi" funktioniert relativ einfach: es gibt abhängig von den Eingangsgrößen – Messen mit Mittelung (true/false) , Online Messung (true/false), Fehler anzeigen (true/false) und Anzahl der Messpunkte (numerisch) – den gewünschten Steuer-String für den Agilent Multiplexer aus.
Die Steuer-Strings an sich sind fest einprogrammiert, es handelt sich bei diesem Sub-VI also im Prinzip um zwei ausgelagerte Case-Strukturen. Lediglich der String zur Steuerung der Betriebsart „Messung" wird dynamisch in Abhängigkeit von der Anzahl der Messpunkte generiert:

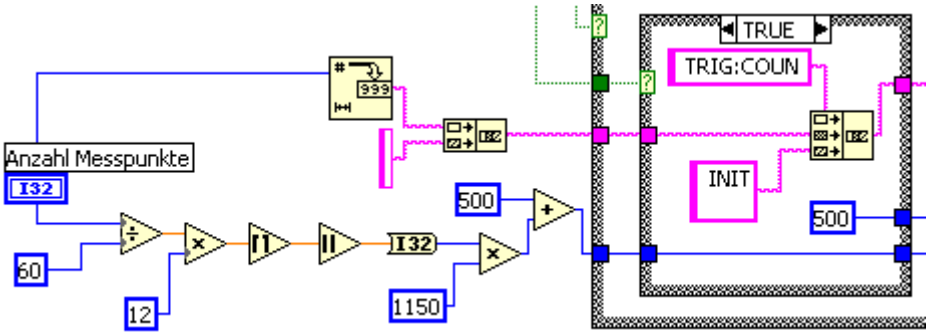


Bild 45: dynamisches erzeugen einen Strings und Berechnung der Messzeit

Die Messzeit wird in Abhängigkeit von der gewählten Anzahl der Messpunkte nach einer empirisch ermittelten Formel berechnet:

$$t_{Mess} = \left\| \left\lceil \frac{n}{60} \cdot z \right\rceil \right\| \cdot 1150 + 500 \qquad [t_{Mess}] = ms$$

mit n = Anzahl der Messpunkte

und z = Anzahl der angesprochenen Kanäle.

Diese Formel entstand durch folgende Überlegungen:

Der Multiplexer kann mit der verwendeten Messkarte maximal 60 Kanäle pro Sekunde abtasten. Für eine Messung über 12 Kanäle mit 100 Messpunkten benötigt der Multiplexer also mindestens 20 Sekunden. Diese Zahl muss mit 1000 multipliziert werden, da das Sub-VI „timewait.vi" in Millisekunden rechnet. In Versuchen mit dem Faktor 1000 stellte sich aber heraus, dass die dem Multiplexer zum Messen zur Verfügung gestellte Zeit zu kurz war. Darum wurde der Faktor nach und nach auf 1150 erhöht, bis ein zufriedenstellendes Ergebnis erreicht war. Eine halbe Sekunde benötigt der Multiplexer durchschnittlich, um den empfangenen String auszuwerten und die Messung auszulösen.

4.2.4 array_sort.vi

Das Sub-VI "array_sort.vi" wird verwendet um ein zweidimensionales Array nach einer Spalte der Größe nach aufsteigend zu sortieren. Die programmiertechnische Umsetzung ist relativ simpel:

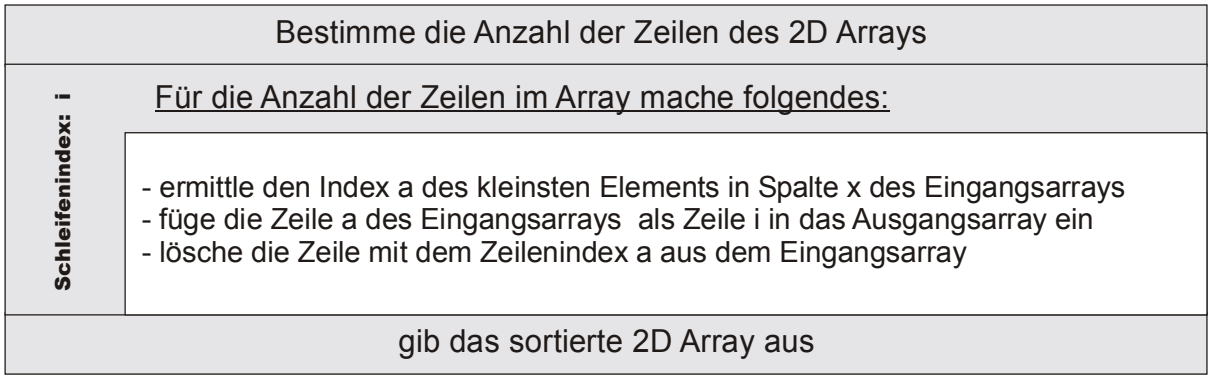


Bild 46: Struktogramm: 2D Array sortieren

Im Eingangsarray wird der Index des kleinsten Elements der Spalte, nach der die Sortierung erfolgen soll, gesucht. Die Zeile, in der sich dieses Element befindet wird als erste Zeile in das Ausgangsarray eingefügt, anschließend wird die Zeile aus dem Ausgangsarray gelöscht. Anschließend wird wieder (in dem nun verkleinerten Eingangsarray) das kleinste Element in der Spalte, nach der sortiert wird, gesucht, die Zeile, in der sich das Element befindet, als zweite Zeile in das Ausgangsarray eingefügt und aus dem Eingangsarray entfernt. Dieser Vorgang wird so lange wiederholt, bis das Eingangsarray aus Null Zeilen besteht. Das Ausgangsarray ist dann das sortierte Array.

4.2.5 auswertung.vi

Das Sub-VI "auswertung.vi" ist eines der wenigen Sub-VIs, bei denen der Benutzer auch mit dem Frontpanel arbeitet. Es dient dazu, die Messwerte als Kurven darzustellen, die Polynomkoeffizienten dieser Kurven zu ermitteln und das Maximum der Kurve zu ermitteln.

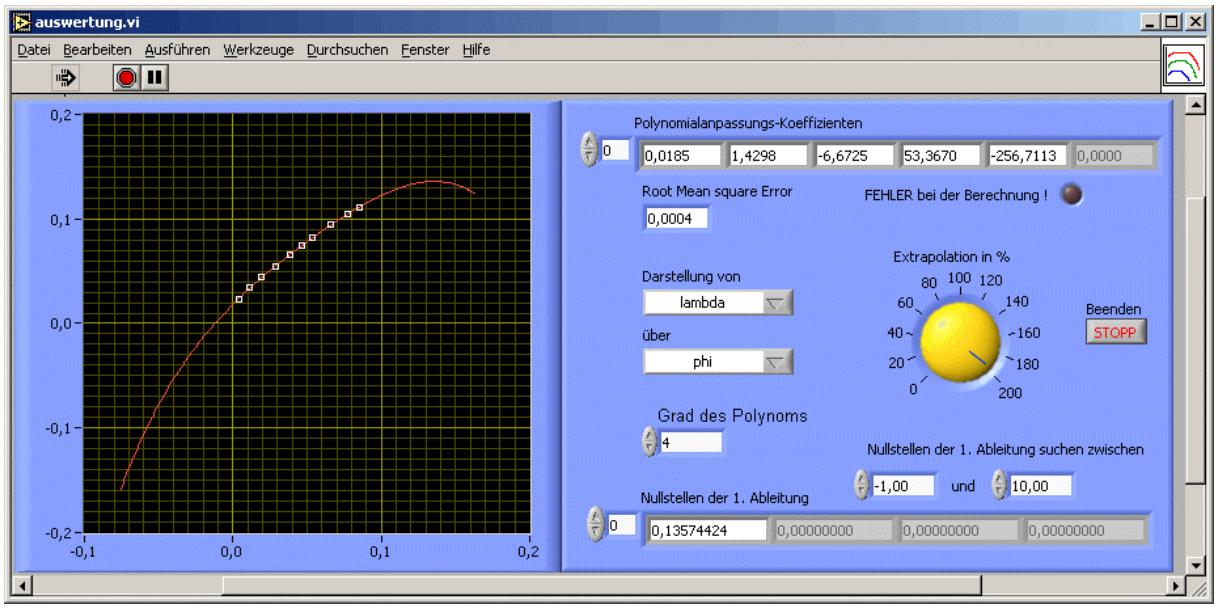


Bild 47: Frontpanel von „auswertung.vi"

Der User kann die Kurve, die er dargestellt haben möchte, mittels eines Menüs auswählen. Der Wertebereich (die Stützstellen) ist durch den kleinsten und den größten gemessenen Wert gegeben. Die Kurve kann aber mittels des gelben Drehreglers bis um maximal 100% extrapoliert werden. Die berechneten Polynomkoeffizienten werden automatisch angezeigt. Die Anzahl der Koeffizienten hängt vom gewählten Grad des Polynoms ab. Sind im Eingangsarray weniger Zeilen vorhanden als der gewählte Grad des Polynoms, so wird automatisch der höchste berechenbare Grad angezeigt. (Beispiel: bei 3 gemessenen Punkten [=3 Zeilen im Eingangsarray] kann kein Polynom 5. Grades angepasst werden. Wird nun der 5. Grad gewählt, werden die Koeffizienten für ein Polynom 3. Grades ausgegeben.)

Die Eingangsgröße dieses VIs ist die globale Variable „Speicher-Array", in der sich die vom Hauptprogramm gemessenen und berechneten Werte befinden. Da sich diese Werte nicht zwangsläufig in der richtigen Reihenfolge befinden [9], müssen diese zunächst mit dem Sub-Vi „array_sort.vi" sortiert werden. Die Spalte, nach der sortiert wird, ist die Abszisse des Graphen.

Das Objekt „allgemeine Polynomanpassung" spielt in diesem VI eine zentrale Rolle.

[9] die Werte, die in der X-Achse dargestellt werden, müssen streng monoton steigen, sonst kann kein Polynom angepasst werden !

Bild 48: LabVIEW-Objekt „allgemeine Polynomanpassung"

Mit diesem Objekt wird u.a. für 2 eindimensionale Arrays ein angepasstes Polynom wählbaren Grades berechnet. Die Polynomkoeffizienten werden wiederum als Array ausgegeben. Der Index (Spalte oder Zeile) des Elements ist gleich dem Exponenten im Polynom, d.h., der Koeffizient für den Term x^0 steht im Array an Position 0, der Koeffizient für den Term x^1 steht im Array an Position 1 und so weiter ...
Dank dieser Anordnung lässt sich die Berechnung der Stützstellen zur grafischen Darstellung dieses Polynoms auf 2 geschachtelte for-Schleifen reduzieren:

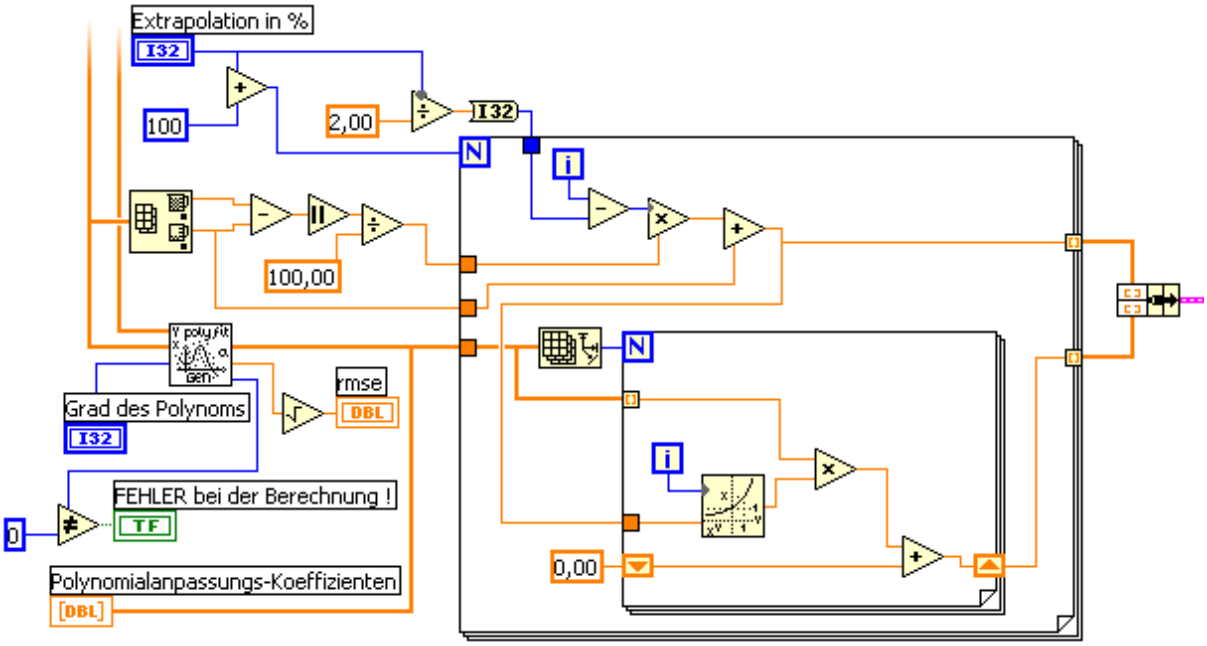


Bild 49: Berechnung eines Polynoms in 2 geschachtelten for-Schleifen

Das Praktische an dieser Methode ist, dass diese Lösung für jeden Grad des Polynoms funktioniert und keine Fallunterscheidungen programmiert werden müssen, da der Grad des Polynoms auch durch die Anzahl der berechneten Koeffizienten und somit durch die Länge des Arrays, in dem diese Koeffizienten ausgegeben werden, gegeben ist.

Die äußere for-Schleife dient zur Berechnung der Stützstellen. Dazu wird die Differenz zwischen größtem und kleinstem Element durch 100 geteilt, jeweils mit dem aktuellen Wert der Schleifenvariablen i multipliziert und zum kleinsten Element addiert, d.h., ohne Extrapolation wird die Kurve mit 100 Stützstellen berechnet.

In der inneren for-Schleife wird für ein festes x der Wert des Polynoms berechnet. Da die for-Schleife automatisch indizieren kann, d.h., in Schleifendurchlauf 0 wird das Arrayelement 0 verwendet, in Schleifendurchlauf 1 das Arrayelement 1, usw., kann man mit der Information über den Grad des Polynoms die programmiertechnische Lösung, wie oben dargestellt, auf eine einzige (sich für jeden x-Wert wiederholende) Operation reduzieren.

4.2.6 berechnung_sub.vi

In dem Sub-VI „berechnung_sub.vi" sind alle aerodynamischen Berechnungen zusammengefasst. Die verwendeten Berechnungsgleichungen sind im Kapitel Grundlagen beschrieben.

In diesem VI wurde hauptsächlich mit dem Objekt „Formelknoten" gearbeitet, da dieser zum einen den Code leicht lesbar macht, zum anderen die Programmierung komplizierter und langer Formeln wesentlich erleichtert.

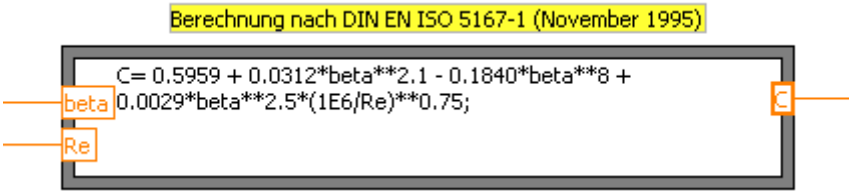


Bild 50: Berechnung des Durchflusskoeffizienten C in einem Formelknoten

Zum besseren Verständnis sind die einzelnen Bereiche im Quellcode selbst mit kleinen Texten dokumentiert.

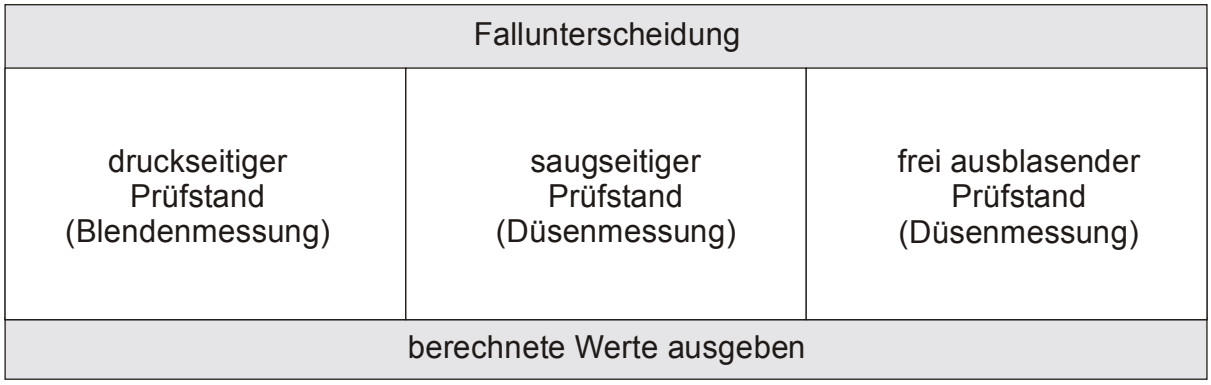


Bild 51: Struktogramm des Sub-VIs „berechnung_sub.vi"

Die äußerste Struktur dieses VIs ist eine Case-Struktur, mit der die verschiedenen Berechnungen für einen druckseitigen Ventilatorprüfstand, einen saugseitigen Ventilatorprüfstand und einen saugseitigen, frei ausblasenden Ventilatorprüfstand unterschieden werden. In allen 3 Fällen ist die Berechnung der Dichte der feuchten Luft und der kinematischen und dynamischen Viskosität des Mediums gleich. Unterschiede gibt es zum einen bei der Berechnung des Durchflusses (Iteration bei der Blende – fester Durchflussbeiwert bei der Düse) zum anderen bei der Berechnung der Totaldruckerhöhung Δp_t von frei ausblasenden Ventilatoren und Ventilatoren mit Rohranschluss. Die aerodynamischen Zusammenhänge sind aber in allen Fällen die gleichen.

Die explizite Erklärung jeder Formel würde den Rahmen dieser Arbeit sprengen, zudem sind die verwendeten Gleichungen bereits in der Literatur ([6], [8] und [9]) sehr gut dokumentiert. Stattdessen soll der Ablauf der Berechnungen für den saugseitigen- und druckseitigen Prüfstand mit jeweils einem Struktogramm verdeutlicht werden.

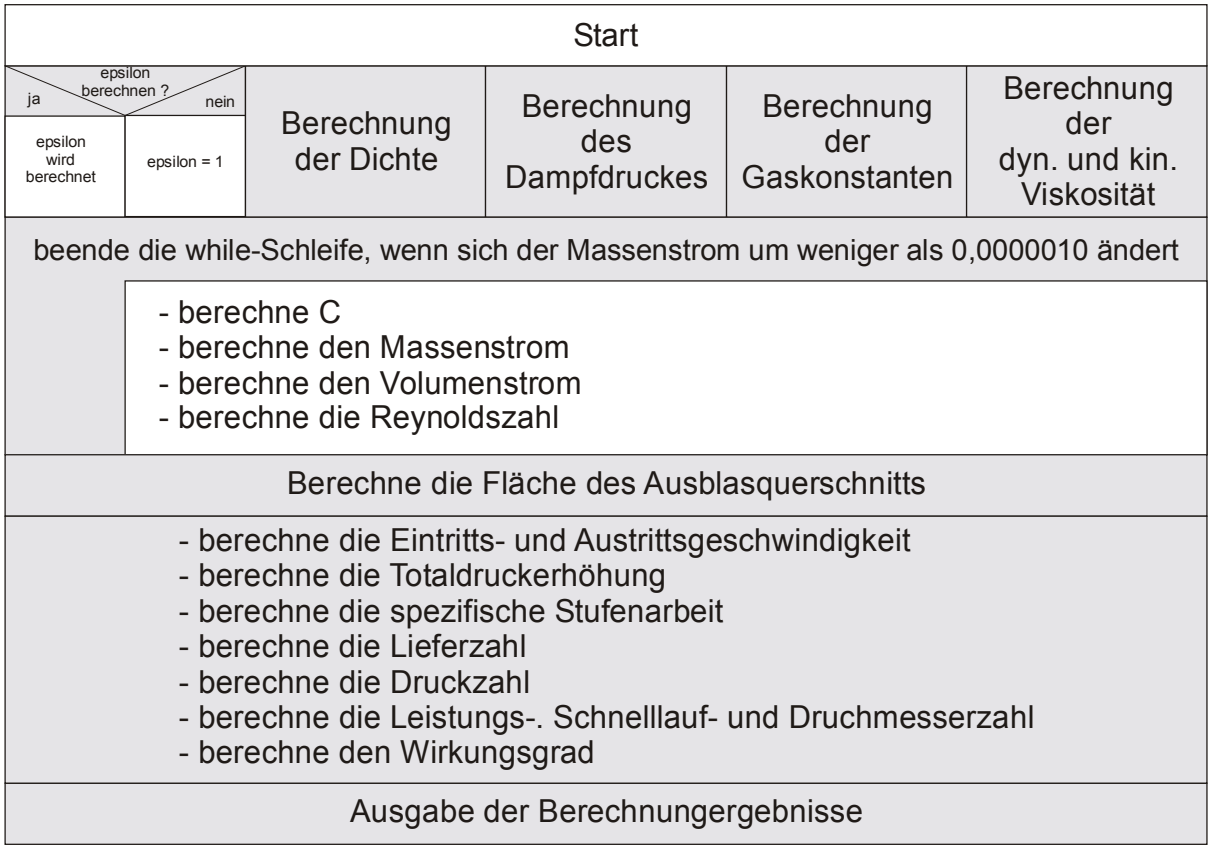


Bild 52: Struktogramm für die Berechnung beim druckseitigen Prüfstand

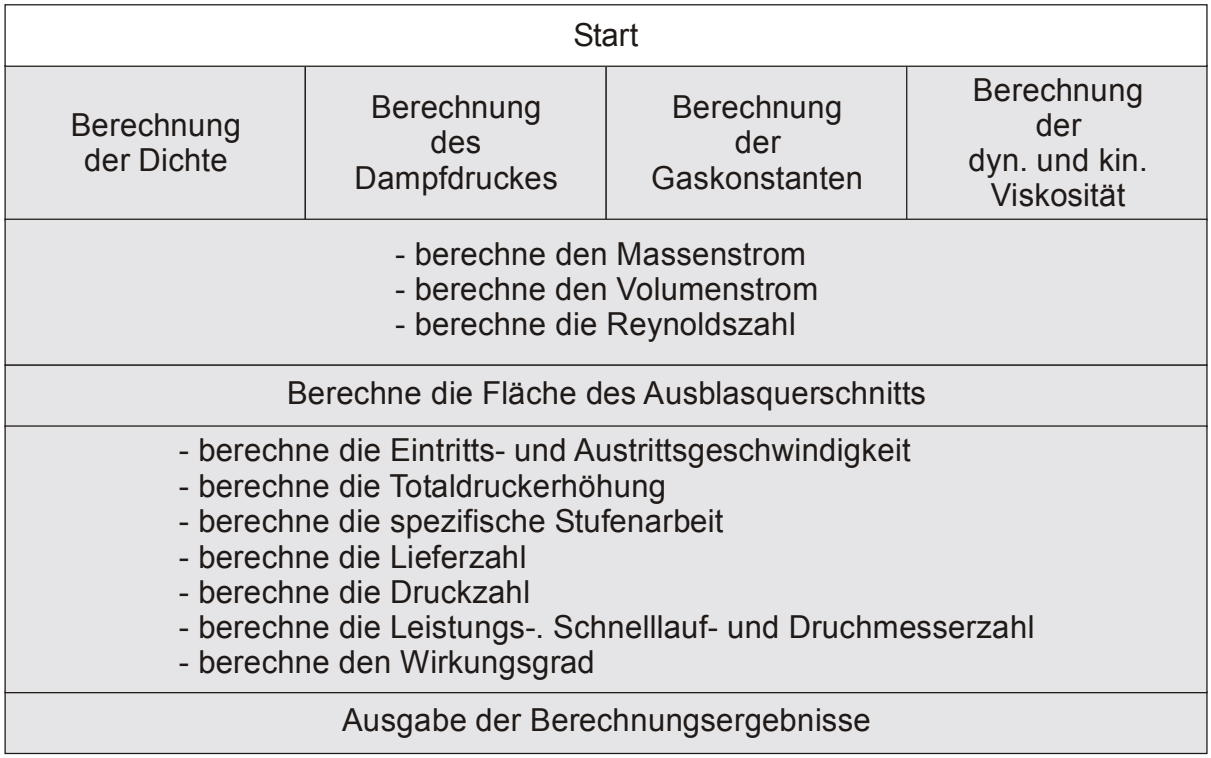


Bild 53: Struktogramm für die Berechnung beim saugseitigen Prüfstand

4.2.7 config_sub.vi

Das Sub-VI „config_sub.vi" ist ein weiteres Unterprogramm, bei dem der Benutzer aktiv auf dem Frontpanel Eingaben vornehmen kann und soll. Es dient zur Verwaltung und Speicherung der Einstellungen und wird beim Start des Hauptprogramms automatisch geöffnet.

Der Quellcode dieses VIs besteht hauptsächlich aus zwei Case Strukturen. In der einen befindet sich die Routine um die eingegebenen Daten in eine *.ini Datei zu schreiben, die andere liest gespeicherte Daten aus einer *.ini Datei aus. Die beiden am häufigsten verwendeten Objekte sind demnach auch:

Bild 54: LabVIEW Objekt „Schlüssel lesen"

und

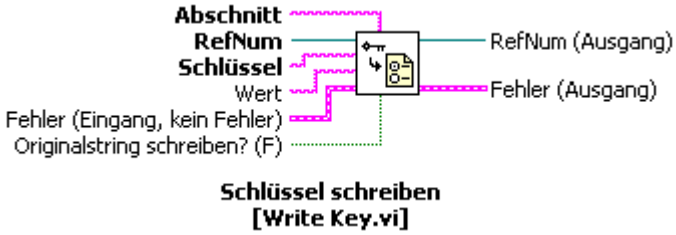
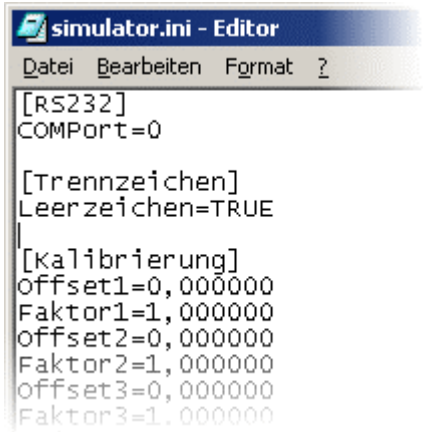


Bild 55: LabVIEW Objekt „Schlüssel schreiben"

Die Funktion dieser beiden Objekte lässt sich am einfachsten anhand der Struktur einer *.ini – Datei erklären:



Eine *.ini-Datei besteht aus sogenannten Abschnitten und aus Schlüsseln. Jeder Abschnittsname kommt in einer Datei nur ein mal vor und ist durch eckige Klammern gekennzeichnet. Dem Abschnittnamen folgen die Schlüssel, die den Wert nach dem Gleichheitszeichen repräsentieren. Ein Schlüsselname kann in einer Datei mehrmals vorkommen, jedoch nur einmal pro Abschnitt. Durch dieses System erhält jeder Wert, der in dieser Datei gespeichert ist, einen eindeutigen Namen. Beim Auslesen sucht man nun nach diesem Namen und erhält den Wert. Beim Schreiben gibt man den Abschnittsnamen und den Schlüssennamen vor und schreibt dann den Wert.

4.2.8 geo_read_sub.vi

Das Sub-VI „geo_read_sub.vi" wurde erstellt um den Quellcode des übergeordneten VIs („config_sub.vi") klein und übersichtlich zu halten. Der Code benötigt relativ viel Platz, weil der Cluster „Geometrie" verschiedene Datentypen beinhaltet, was dazu führt, dass keine Schleife zum Einlesen der Daten verwendet werden kann, sondern jeder Wert separat eingelesen werden muss.
Die Definition des Datentyps erfolgt bei dem Objekt „Schlüssel lesen" durch Angabe eines Standardwertes eines bestimmten Typs:

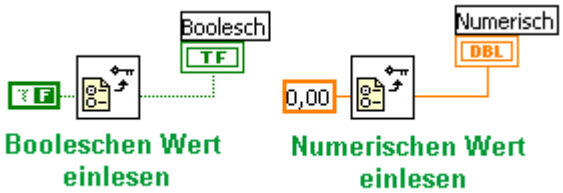


Bild 56: Schlüssel einlesen

Da die Übergabe der ausgelesenen Werte an den Ausgangscluster nicht sequentiell erfolgen kann, müssen alle Schlüssel parallel eingelesen und übergeben werden, was dazu führt, dass für jeden einzulesenden Wert ein eigenes Objekt mit der dazugehörigen Definition des Standardwertes verwendet werden muss. [10]

[10] dies ist sicher keine besonders elegante Lösung ...

4.2.9 simulator.vi

Das Sub-VI „simulator.vi" simuliert den Betrieb am Prüfstand. Dieses Unterprogramm ist in das VI „aamdes_0.9.3sim.vi" eingebaut und ersetzt dort die Strukturen zur Steuerung des Multiplexers und zum Einlesen der Daten von der RS232 Schnittstelle. Die Strings, die „normalerweise" von dem Gerät gesendet werden, werden im Simulatorbetrieb nun von dem Unterprogramm generiert. Die Steuerung des Sub-VI's erfolgt durch ein zusätzliches Bedienfeld im Hauptprogramm. Dort werden auch die benötigten Parameter eingestellt.

Bild 57: Bedienfelder für den Simulatorbetrieb

Für einen reibungslosen Betrieb sollten entweder die Konfigurationsdatei „simulator.ini" geladen werden, oder aber sinnvolle Einstellungen gewählt werden. Hierbei gilt es zu beachten, dass das Trennzeichen zwischen den Dezimalstrings immer ein „&" ist, alle Kalibrierfaktoren 1 und alles Offsets 0 sind. Der Hebelarm zur Berechnung des saugseitigen Drehmoments sollte mit 1 m angegeben werden.

Die Berechnungen, die dieses VI ausführt sind ist im Prinzip die Umkehrung des VI „berechnung_sub.vi". Mit Hilfe der dimensionslosen Kennlinie einer bereits verifizierten Messung werden die Größen Blendenwirkdruck, Druckdifferenz am Austritt, Düsenwirkdruck, Druckdifferenz am Eintritt und das Drehmoment (druck- und saugseitig) berechnet und als Strings ausgegeben. Diese Strings werden vom Hauptprogramm übernommen und verarbeitet. Wenn das Messdaten-erfassungssystem richtig arbeitet, dann stimmen vom Simulatorprogramm berechneten Größen mit den Ergebnissen des Hauptprogramms überein.

Der Drossel- / Entdrosselvorgang wird durch einen Schieberegler simuliert, mit dem man die Lieferzahl zwischen 0 und 100% eines festgelegten Intervalls bewegen kann. Bei der Festlegung des Intervalls sollte der Bereich gewählt werden, in dem diese Kennlinie tatsächlich gemessen wurde.

4.2.10 string2num_sub.vi

Das Sub-Vi „string2num_sub.vi" wandelt die eingelesenen Strings in Zahlen um. Das wichtigste Objekt in diesem Programm ist das Objekt Tabellenstring in Array:

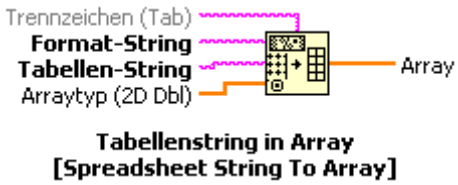


Bild 58: LabVIEW Objekt „Tabellenstring in Array"

Mit Hilfe dieses Objektes wird z.B. ein String der Form „1,123#2,123#3,123" in das eindimensionale Array (1,123 2,123 3,123) umgewandelt. Dank dieses Objektes spart man sich – bei richtiger Formatierung des Eingangsstrings – die Arbeit, den String in einzelne Teilstücke, die jeweils einen Zahlenwert repräsentieren, zu zerlegen und anschließend jeden String einzeln in eine Zahl umzuwandeln.

Unglücklicher weise ist das vom Multiplexer verwendete Trennzeichen ein Komma, das auf manchen Systemen als Dezimaltrennzeichen interpretiert wird.

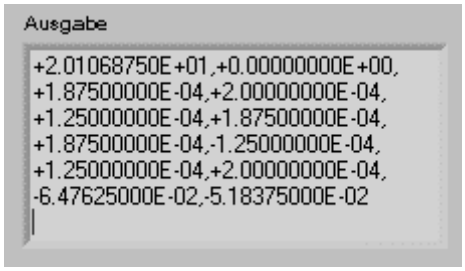


Bild 59: vom Multiplexer gesendete Strings in ihrer „Urform"

Das führt auf manchen Systemen dazu, dass die Strings nicht richtig interpretiert werden. Darum wird mit Hilfe der „String2Num Settings" das Komma aus dem String gefiltert und durch ein anderes Zeichen ersetzt (Standardeinstellung ist eine Raute „#"), das evtl. falsche Dezimaltrennzeichen (der Punkt) wird ebenfalls gefiltert und durch das richtige ersetzt. Mit diesem „aufbereiteten" String kann man das Objekt „Tabellenstring in Array" ohne Probleme einsetzen und den String in ein Array aus Zahlen umwandeln.

Die Werte im Array können mit dem Objekt „Array indizieren" aus der Tabelle extrahiert werden und dem Ausgangscluster „gemessene Spannungen" zugeführt werden:

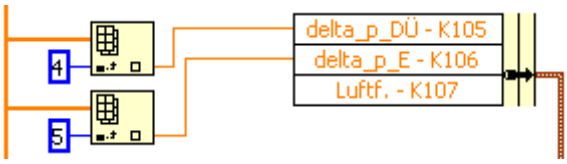


Bild 60: Array indizieren und Werte in den Ausgangscluster schreiben

4.2.11 timewait.vi

Das Sub-VI „timewait.vi" ist ein sehr kleines VI. Es hat lediglich den Zweck eine Struktur für einen bestimmten Zeitraum warten zu lassen. Der Quellcode besteht aus einer Sequenz mit 2 Rahmen und einer while-Schleife. In Rahmen 1 wird der aktuelle Timer-Wert eingelesen. In Rahmen 2 wird so lange der aktuelle Timer-Wert ausgelesen und mit dem Wert aus Rahmen 1 plus die Wartezeit in Millisekunden verglichen, bis dieser größer ist.

Bild 61: Quellcode von „timewait.vi"

Das besondere an diesem VI im Gegensatz zu den anderen Sub-VIs ist die Priorität, mit der es ausgeführt wird. Mit einem Mausklick auf Datei – VI-Einstellungen gelangt man in ein Menü, in dem man unter der Kategorie „Ausführung" wählen kann, ob die Bearbeitung dieses VI im Bezug auf das aufrufende VI eine niedrigere oder höhere Priorität hat:

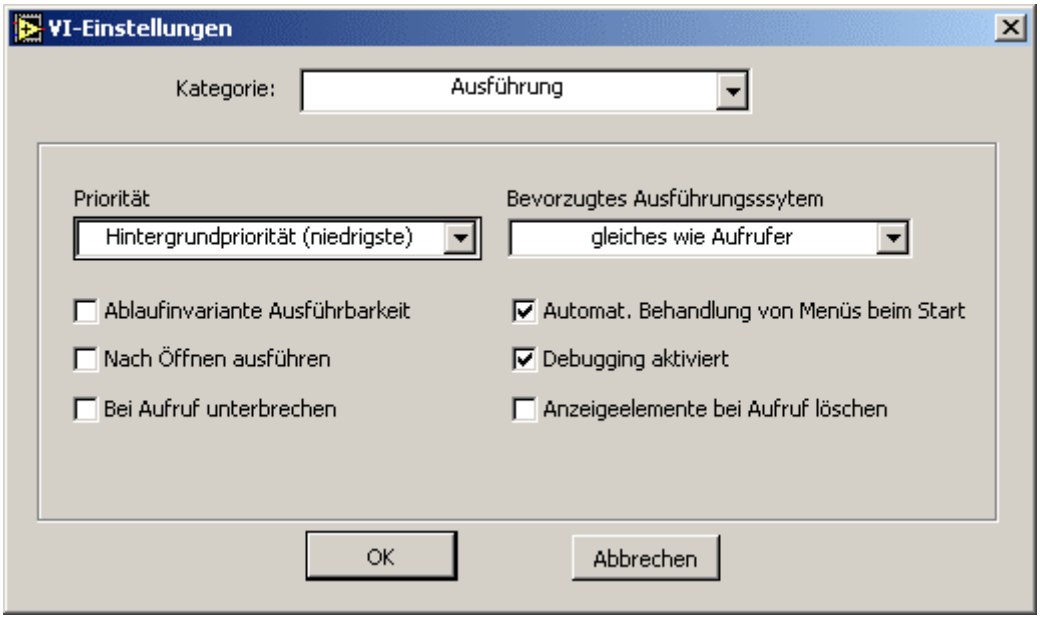


Bild 62: VI-Einstellungen

Dieses VI hat nun die niedrigste Priorität und macht damit Rechenzeit für wichtigere Unterprogramme wie z.B. die Aufzeichnung von Audiodaten und die online FFT sowie die Datenerfassung und Berechnung der Ergebnisse frei.

4.2.12 umwandlung_csv_sub.vi

Vor dem Schreiben in die Ausgabedatei werden die Zahlenwerte wieder in Strings umgewandelt. Dies geschieht in dem Sub-VI „umwandlung_csv_sub.vi".

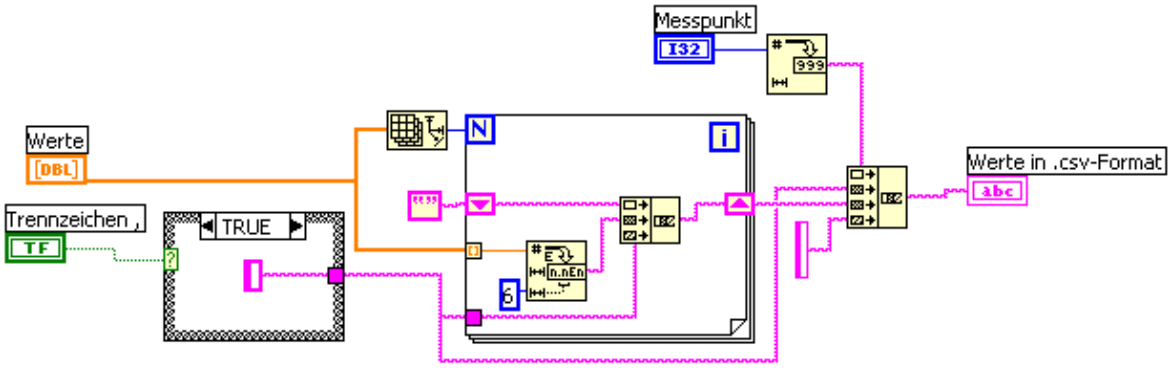


Bild 63: Quellcode „umwandlung_csv_sib.vi"

Die Werte, die auf dem Frontpanel zu sehen sind, werden als eindimensionales Array an dieses Unterprogramm übergeben. Weitere Eingangsgrößen sind der aktuelle Messpunkt, sowie das gewählte Trennzeichen. Die dem Array entnommen Werte

werden in einen String in Exponentialnotation mit 6 Nachkommastellen umgewandelt und zu einem einzigen – langen – String zusammengefügt. Zwischen den Werten wird das gewählte Trennzeichen gesetzt. Diesem langen String wird der ebenfalls in einen String umgewandelte aktuelle Messpunkt vorangestellt. Das abschließende Zeichen ist ein „Carriage Return". Damit wird sichergestellt, dass in der Dateiausgabe für jeden Messpunkt eine neue Zeile begonnen wird. Der fertige String wird dann wieder an das Hauptprogramm zurückgegeben, wo er in die Ausgabedatei geschrieben wird.

4.3 Hilfsprogramme zur Verdeutlichung von Abläufen

Viele komplexe Funktionen wurden nicht im Hauptprogramm selbst entwickelt, sondern zunächst als eigenständige Anwendung programmiert und getestet. Zwar werden diese Programme zum Teil im Messdatenerfassungssystem nicht mehr benötigt, sie eignen sich aber hervorragend, um die programmiertechnische Lösung nicht trivialer Zusammenhänge, losgelöst von dem sehr großen Hauptprogramm, zu betrachten und zu erklären.

4.3.1 iter2.vi

Das Programm „iter2.vi" berechnet den Wert der Folge

$$\left\{\left(1+\frac{1}{n}\right)^{n}\right\}$$

die bekanntlich gegen die natürliche Zahl e (≈ 2,71828) konvergiert. **[7]**

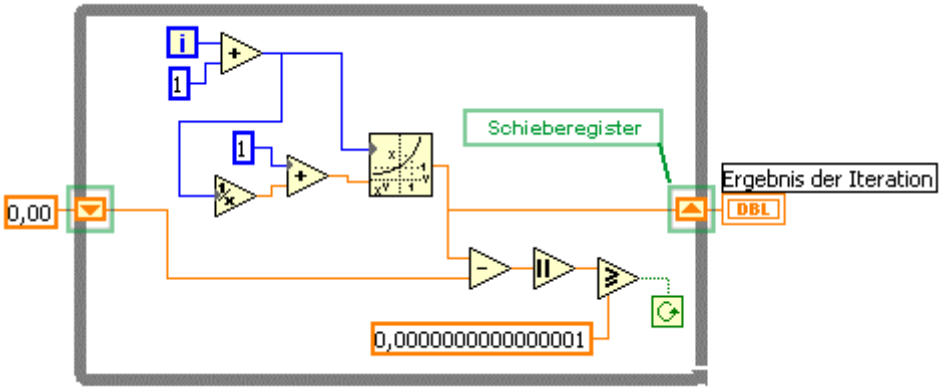


Bild 64: VI „iter2.vi"

Die Iteration läuft in einer while-Schleife ab, die beendet wird und das Ergebnis an das Anzeigeelement „Ergebnis der Iteration" sendet (vgl. Kap. 3.2 – „Datenfluss-Modell"), wenn die Differenz zwischen aktuellem Wert der Berechnung und dem Wert der Berechnung im letzten Schleifendurchlauf kleiner als 1,0 E-16 ist. Um diese Differenz bilden zu können, ist es notwendig, auf den aktuellen **und** den letzten Wert zurückzugreifen. Dies kann man durch ein Schieberegister bewerkstelligen – ein Element der while-Schleife, das dem Eingang (linke Seite) den Wert des Ausgangs (rechte Seite) des vorhergehenden Schleifendurchlaufes zuweist.

Nach diesem Prinzip läuft die Iteration in dem Sub-VI „berechnung_sub.vi" ab, mit der die Werte für den Durchflusskoeffizienten berechnet werden, damit Massenstrom und Reynoldszahl ermittelt werden können.

4.3.2 array.vi

Normalerweise ist das Einfügen von Spalten oder Zeilen in ein Array mit dem Objekt „in Array einfügen" leicht zu bewerkstelligen:

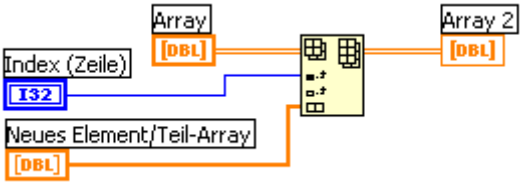


Bild 65: 1D Array als neue Zeile in ein 2D Array einfügen

Wenn man jedoch nicht direkt auf die Eingangs- und Ausgangsarrays zugreifen kann, und gezwungen ist mit lokalen Variablen zu arbeiten, bekommt man das Problem, dass am Eingang des Objekts immer Daten vorhanden sind und die Struktur mehrmals ausgeführt wird. Zwar gehen keine Daten verloren, die Struktur benötigt aber zusätzliche Rechenzeit, die evtl. an anderer Stelle fehlt, zudem ist das unkontrollierte wiederholte Schreiben der gleichen Werte an die gleiche Stelle auch kein Zeichen von sauberer Programmierung.

Um dieses Problem zu lösen, ist es notwendig einen definierten Zustand herzustellen, bei dem die Werte nur einmal geschrieben werden. Dies erreicht man folgendermaßen:

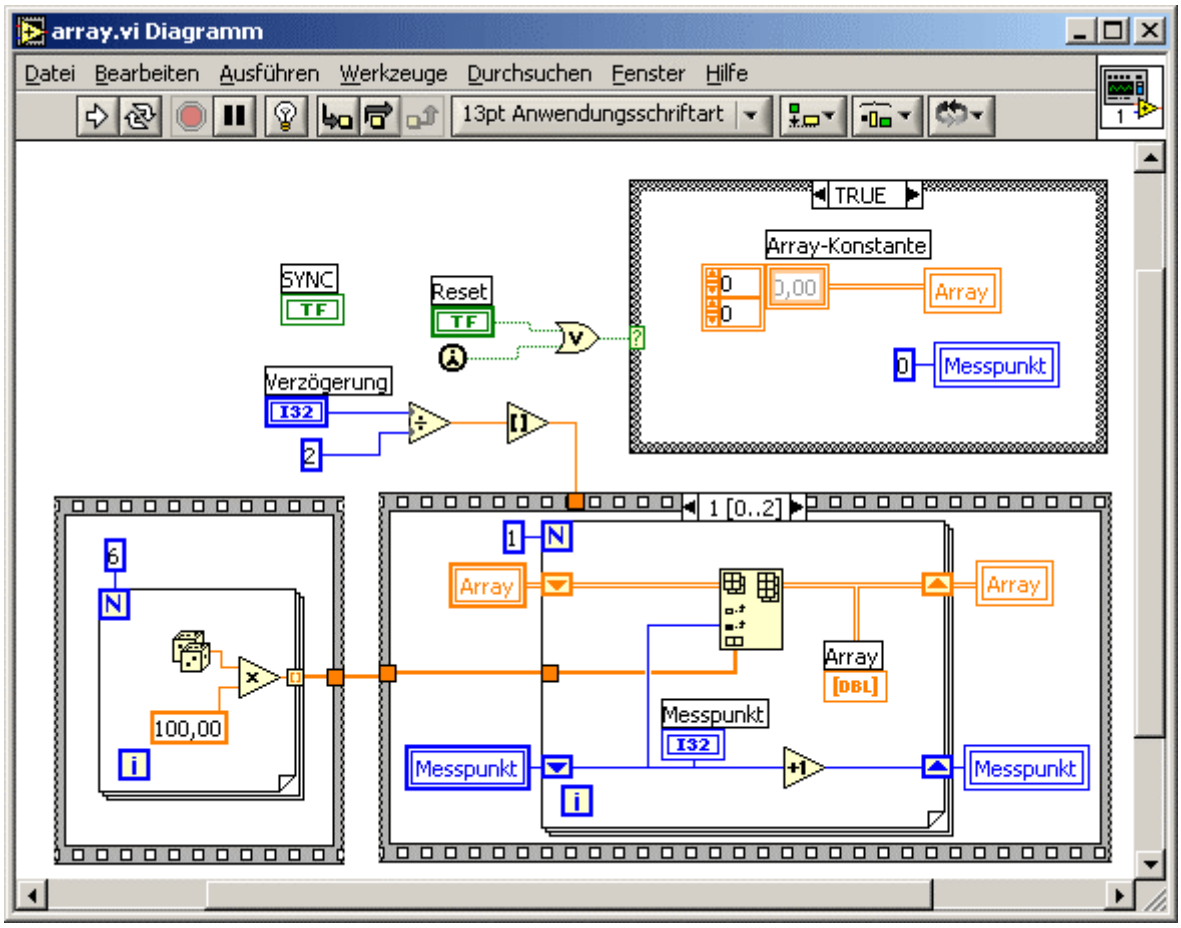


Bild 66: Quellcode „array.vi"

Der Kern dieses Programms befindet sich in der Sequenz rechts unten. Die Case Struktur rechts oben dient lediglich zum Initialisieren der Variablen beim Programmstart, die for-Schleife links unten erzeugt ein eindimensionales Array aus 6 Zufallszahlen im Intervall von 0 bis 100.

Rahmen 1 und 3 (Index 0 und 2) der Sequenz rechts unten beinhalten nur das bereits aus Kapitel 4.2.11 bekannte Sub-VI „timewait.vi" und haben nur den Zweck, die Ausführung dieses Beispielprogramms so zu verlangsamen, dass man den Änderungen auf dem Frontpanel folgen kann.

Den Eingangs beschriebenen definierten Zustand erreicht man durch die for-Schleife in Rahmen 2 der Sequenz, die genau einmal ausgeführt wird und dann wieder auf einen neuen Datensatz wartet, bevor sie erneut ausgeführt wird.

Die verwendeten Schieberegister sind im oben dargestellten Beispiel nicht unbedingt notwendig. Sie wurden aber verwendet, da es mit einer kleinen Änderung am Code auch möglich ist, die for-Schleife mehrmals auszuführen und einzelne Werte in ein Array einzufügen. Letzteres ist sinnvoll, wenn die Zeile oder Spalte eben nicht als 1 D Array vorliegt, sondern erst in dieser Schleife zu einer Zeile oder Spalte zusammengefügt werden soll.

4.3.3 soundtest2.vi

Der Quellcode des VI's „soundtest2.vi" stammt aus dem von der Hilfe zur Verfügung gestellten Beispielprogramm „Record Wave File.vi", das demonstriert, wie eine Windows Audiodatei aufgenommen wird. Für die Programmierung einer Audioaufnahme mittels der Soundkarte des PCs stehen bereits LabVIEW Objekte zur Verfügung. Die Zusammenstellung dieser Objekte zu einem lauffähigen Programm und die Objekte selbst werden nun im einzelnen vorgestellt:

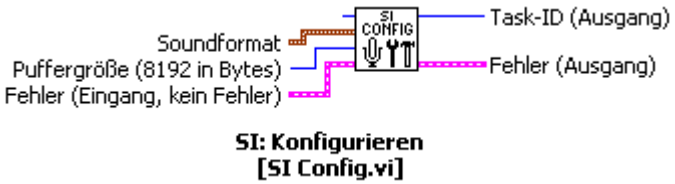


Bild 67: Sound Eingang konfigurieren

Zunächst muss der Soundeingang konfiguriert werden. Dazu benötigt das Objekt „SI Config.vi" Informationen über die verwendete Karte (oberster Eingang). Da jedoch üblicherweise pro PC nur eine Soundkarte vorhanden ist, kann dieser Eingang offen bleiben oder man verwendet die Standardeinstellung „0". Der Eingang „Soundformat" benötigt einen Cluster, in dem die Informationen über die Soundqualität (stereo oder mono), die Abtastrate (8; 11,025; 21,5; 44,1 kHz) und die Auflösung (8 oder 16 bit) enthalten sind. Die Puffergröße sollte zwischen 8192 Bytes (= 8 kB) bei einer mono – 8 bit Aufnahme und 65335 Bytes (= 64 kB) liegen. Von diesem Objekt wird auch die „Task-ID" erstellt, die alle anderen Objekte, die mit der Aufzeichnung von Audiodaten zu tun haben, als Eingangsgröße zwingend benötigen.

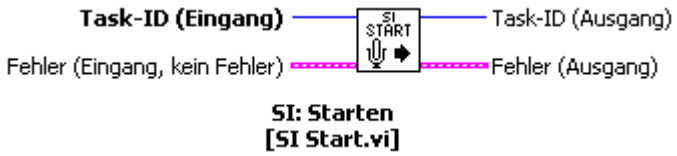


Bild 68: Aufnahme starten

Wenn die Konfiguration abgeschlossen ist, kann die Aufnahme mit dem Objekt „SI Start.vi" gestartet werden. Die Soundkarte beginnt nun Daten aufzunehmen und in den Puffer zu schreiben. Von dort müssen die Daten mit dem Objekt „SI Read.vi" ausgelesen werden.

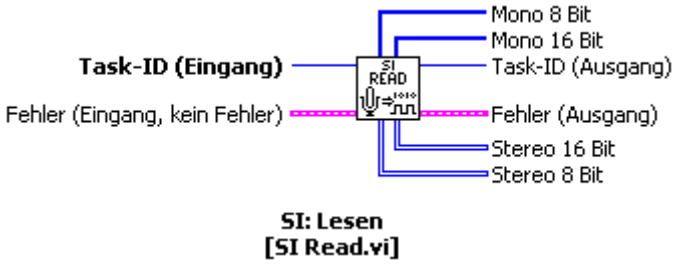


Bild 69: Audiodaten aus dem Puffer lesen

Dieses Objekt hat für die Formate mono 8 bit, mono 16 bit, stereo 8 bit und stereo 16 bit jeweils einen Ausgang. Daten sind aber nur an dem Ausgang vorhanden, der den durch „Soundeingang konfigurieren" gewählten Einstellungen entspricht. Die Audiodaten werden als Integer-Zahlenwerte werden in Arrays abgelegt. Sinnvollerweise legt man um das Einlese-Objekt eine while-Schleife mit einem Schieberegister, in das die eingelesenen Daten eingefügt werden. Die aufgezeichneten Audiodaten werden dann bis zum Ende der Schleife im Arbeitsspeicher des PCs abgelegt.

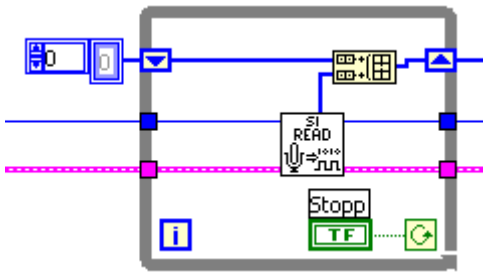


Bild 70: Audiodaten in einer while-Schleife einlesen

Diese Methode birgt aber auch die Gefahr, dass der Arbeitsspeicher des PCs „volläuft", da bei hoher Auflösung größere Datenmengen gesammelt werden:

Bei einer Abtastrate von 44,1 kHz (CD-Qualtität) und einer Auflösung von 16 bit stereo fallen in einer Sekunde 2 (2 Byte = 16 bit) mal 44100 mal 2 = 176,4 kByte Audiodaten an, was wiederum bedeutet, dass eine Aufnahme über ein Minute unter Umständen bis zu 10 Megabyte Speicher beansprucht. Bei Rechnern mit wenig RAM kann das unter Umständen zu „BlueScreens" führen.

Durch das Objekt „SI Stop.vi" wird die Aufzeichnung von Audiodaten wieder angehalten.

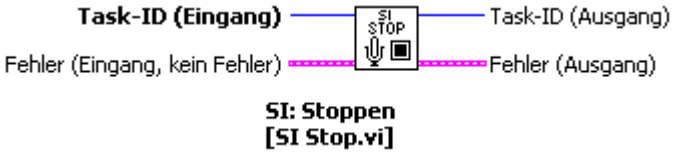


Bild 71: Aufnahme stoppen

Anschließend müssen die Soundkarte und damit alle Resourcen, die in diesem Prozess verwendet wurden, wieder freigegeben werden. Dies erfolgt durch das Objekt „SI Clear.vi":

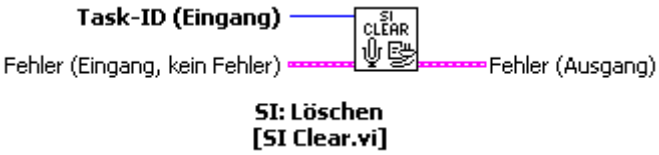


Bild 72: Soundkartentreiber schließen

Mit dem Objekt „SI File Write.vi" lassen sich die aufgezeichneten Daten nun auf die Festplatte in eine Datei speichern:

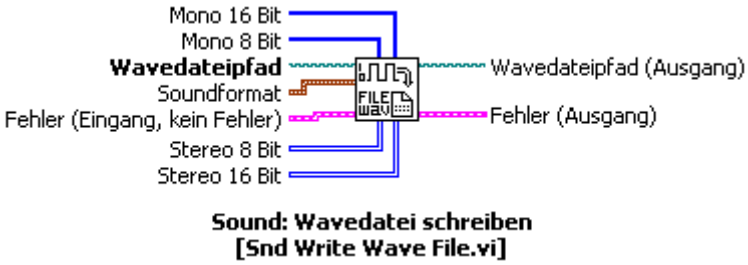


Bild 73: Audiodaten in ein *wav-File schreiben

5 Vergleich des PC-gestützten Systems mit herkömmlichen Messgeräten

Beim Vergleich der Genauigkeit des PC-gestützten Systems mit herkömmlichen Systemen muss man gedanklich zwischen Messgerät und Rechner trennen.

Werden nur externe Messgeräte verwendet (z.B. Multimeter, etc.), die mit dem Rechner über eine Schnittstelle verbunden sind, so können die Berechnungen im PC können als sehr genau betrachtet werden. Zwar macht auch ein PC beim Rechnen mit Fliesskommazahlen einen Fehler, dieser liegt aber im Bereich von mindestens 10^{-30} oder kleiner, ist also technisch in keinster Weise relevant.

Werden die Messungen im PC selbst vorgenommen (z.B. mit der Soundkarte oder Messkarte) so tritt auch im PC ein Messfehler auf. Wie bei allen Digitalen Messgeräten kommt es zu dem s.g. Differentialfehler, da die Auflösung bei der AD-Wandlung immer begrenzt ist. Darüber hinaus neigen AD-Wandlerkarten[11] zum Übersprechen der Kanäle.

5.1 Soundkarte als Messgerät

Eggert hat in seiner Diplomarbeit „Objektorientierte Programmierung eines 2-Kanal Frequenzanalysators unter DASYLab und LabVIEW" **[10]** bereits Untersuchungen zur Verwendbarkeit von Soundkarten als Messgerät für Schalluntersuchungen gemacht und stellt fest:

„Die Messungen zeigten, dass die getestete Soundkarte vom Typ DMX Xfire 1024 der Firma TerraTec ein sehr gutes lineares Übertragungsverhalten zeigt. [...] Die Untersuchung auf Phasentreue zeigte, dass bei der Soundkarte DMX Xfire 1024 eine maximale Phase von 0,18° bei etwa 22,3 kHz auftrat, was einen sehr guten Wert darstellt. Auch für die Amplitude und die Kohärenz ergaben sich sehr gute Werte.
Leider zeigte sich bei der Soundkarte DMS Xfire 1024 mit einem Abstand von 66 dB ein leichtes Übersprechen auf den zweiten Kanal."

---

[11] AD-Wandlerkarten können zudem nur Spannungen in einem beschränkten Spannungsbereich sampeln

In seiner Zusammenfassung schreibt er:

„Zusammenfassend lässt sich sagen, dass Soundkarten der mittleren bzw. oberen Preisklasse als Messkarten zu Zusammenarbeit mit einem Personalcomputer zu Messzwecken in Betracht kommen“.

Mit diesem Wissen kann man also durchaus davon ausgehen, dass die Weiterverarbeitung der aufgezeichneten Soundfiles korrekte Ergebnisse liefert. In eigenen Untersuchungen konnte auch festgestellt werden, dass – Dank des fast linearen Frequenzgangs der hier verwendeten Soundkarte – eine Kalibrierung mit Pistonfon und Kalibrator gute Resultate liefert.

5.2 Temperaturmessung

Das Agilent 34970A Messdatenerfassung- / Schaltsystem verfügt auf der verwendeten Messkarte (Typbezeichnung: HP 34901A, 20-Kanal-Anker-Multiplexer) über eine eingebaute Thermoelement Vergleichsstelle. Die Messung der Lufttemperatur wird mit dem im Lieferumfang enthaltenen Thermoelement Typ J durchgeführt.

Im Vergleich zu Präzisionsmessgeräten, die bis zu 0,1 Kelvin[12] auflösen können, ist ein Thermoelement mit einer Genauigkeit von ± 1,5 K relativ ungenau. Eine Verbesserung der Genauigkeit wäre z.B. durch die Benutzung von Platin-Widerstandsthermometern möglich. Hier kann man – wenn man Preis und Leistung abwägt – durchaus eine Auflösung von 0,1 K erreichen, allerdings muß bei Pt-Widerstandsthermometern auch die Zeitkonstante beachtet werden, da diese wesentlich träger sind als Thermoelemente.

Bei genauerer Betrachtung des Berechnungsmoduls und der Überlegung, was eine genauere Temperaturmessung denn für Auswirkungen auf die Genauigkeit der Ergebnisse hat, kommt man jedoch schnell zu der Auffassung, dass eine Erhöhung

[12] bei Luftströmungen ist die Bestimmung auf 0,1 K schon mit einem sehr großen Aufwand verbunden

der Genauigkeit der Temperaturmessung relativ wenig Einfluss auf die Genauigkeit der Resultate hat. Dies ist darauf zurückzuführen, dass die gesamte aerodynamische Berechnung für eine hydraulische Strömungsmaschine ausgeführt ist, bei der die Temperatur nur in die Berechnung der Dichte eingeht. Diese Berechnung wird auch nur ein mal pro Messzyklus ausgeführt. Des weiteren geht die Dichte in die meisten Gleichungen mit der Potenz 1 ein. Eine Verbesserung der Genauigkeit hätte also nur Auswirkungen auf technisch nicht mehr relevante Nachkommastellen und lohnt sich somit nicht.

Ein weiterer Messfehler entsteht durch die Iteration des Durchflusskoeffizienten mit dem im Berechnungsmodul der Massenstrom und die Reynoldszahl berechnet werden. Bedingt durch die Abbruchbedingung der while-Schleife, die terminiert, wenn sich der Massenstrom um weniger als 0,000001 ändert, ist auch der hierbei entstehende Fehler technisch nicht relevant.

5.3 Druckmessung

Die Genauigkeit der Druckmessung ist von 2 Faktoren abhängig: Zum einen ist die Güte des verwendeten Messgeräts sehr entscheidend. Zum anderen entstehen auch bei der Spannungsmessung mit dem internen Digitalmultimeter des Multiplexers Fehler.

Die verwendeten Druckmessgeräte der Firma Mecotec weisen je nach Ausführung eine Abweichung von ± 12,5 Pa (bei der Ausführung 0 bis 50 mbar, dies entspricht der angegebenen Abweichung von 0,25% vom Endausschlag, der für alle Typen gilt) bzw. sogar bis zu ± 250 Pa (bei der Ausführung 0 bis 1 bar) auf. Da alle Messgeräte die Spannung von 1 V bei Vollausschlag ausgeben, entsteht dadurch auch ein Fehler bei der ausgegebenen Spannung, der bei ±2,5 mV liegt. Zusammen mit dem Fehler des DMM, der bei der verwendeten Messkarte und dem gewählten Messbereich zwischen ± 0,6 und ± 0,8 mV **[11]** liegt, entstehen Ungenauigkeiten bei der Druckmessung im Bereich von ± 20 bis ±320 Pa.

Da aber diese Geräte Piezokristalle zur Umwandlung des Drucks in eine Spannung verwenden, kann man von einem linearen Zusammenhang zwischen gemessenem Druck und ausgegebener Spannung ausgehen. Wenn man diese Geräte in regelmäßigen Abständen mit einem sehr genauen Flüssigkeitsmanometer (Betz - Manometer) überprüft, erreicht man auch mit einem derartigen Messgerät Genauigkeiten kleiner 1 Pa, was ausreichend ist, um auch kleine Strömungsgeschwindigkeiten mit ausreichender Genauigkeit aufzunehmen.

Ein Einfluss der Umgebungstemperatur[13] auf die Genauigkeit der Mekotek Geräte ist nicht überprüft worden und spielt, wie bei allen Druckmessgeräten, nur für den unteren Druckmessbereich eine Rolle.

Die Genauigkeit der von der Fa. Pollrich Ventilator verwendeten Druckmessgeräte des Typs MKS Baratron wird vom Hersteller mit 0,15% vom Messwert angegeben. Bei einem zu messenden tatsächlichen Druck von 10 Pa bedeutet dies eine Messunsicherheit von 0,015 Pa. Dies ist in der Tat eine sehr hohe Genauigkeit. Ein Einfluss der Umgebungstemperatur macht sich nicht bemerkbar, da die Geräte auf eine Temperatur von 45°C aufgeheizt werden und diese Temperatur beibehalten wird.

[13] Der Einfluss der Gastemperatur wird hier nicht explizit berücksichtigt, da aufgrund des Aufbaus des Prüfstandes, die Gastemperatur am Messgerät gleich der Umgebungstemperatur ist

6 Erweiterung der Software

Für die Erweiterung der Software gibt es auf Grund der Komplexität keine universelle Lösung. Eine Änderung des Quellcodes wird nur zu bewerkstelligen sein, wenn man zum einen Erfahrung in der Programmierung von LabVIEW hat, zum anderen benötigt man auch Kenntnisse im Bereich Prüfstandsmessungen an Ventilatoren.

Anhand einiger Beispiele soll nun verdeutlicht werden, wie man eine notwendige Änderung durchführt und was dabei beachtet werden muss.

6.1 Beispiel 1: Austausch des Multiplexers

Annahme: der bisher verwendete Multiplexer kann aus einem nicht näher genannten Grund nicht mehr eingesetzt werden. Es steht ein anderes Gerät zur Verfügung, das ebenfalls an die RS232 Schnittstelle angeschlossen werden kann und mit SCPI - Befehlen gesteuert werden kann. Die gesendeten Daten sind ebenfalls Strings.

Lösung:

Dieses Problem betrifft hauptsächlich das Modul „agilent_steuerung_sub.vi". Zunächst müssen die Steuer-Strings gemäß dem Handbuch so angepasst werden, dass der Multiplexer wieder gesteuert werden kann. Die Zeit, die die Messkarte zur Aufnahme einer Messreihe benötigt muss ebenfalls berücksichtigt werden. Dazu ist in Abhängigkeit von der Abtastrate der Karte entweder die Formel (Bild 45:), mit der die Messzeit berechnet wird, anzupassen, oder ggf. auch neu zu ermitteln.

Sollte das Format des ausgegebenen Strings nicht dem bisher verwendeten Format ähnlich sein, so kann es unter Umständen notwendig sein auch die Routine zur Umwandlung der Strings in Zahlen abzuändern. Normalerweise – z.B. bei einem anderen Trennzeichen – sollte dies aber nicht notwendig werden.
Es ist durchaus denkbar, dass auch die Einstellungen des COM-Ports verändert werden müssen. Diese Änderung müsste dann direkt im Hauptprogramm vorgenommen werden.

6.2 Beispiel 2: Erweiterung der Sound Funktion

Annahme: Das bestehende Programm soll neben der Aufnahme auch die Momentanwerte von Schallleistung und Schalldruck anzeigen.

Lösung:
Am einfachsten lässt sich diese Aufgabenstellung realisieren, wenn man das Sub-VI „2kanalfft_sub.vi" anpasst, da dieses schon auf das Hauptprogramm abgestimmt ist und die benötigten Größen bereits in der richtigen Form an dieses Sub-VI übergeben werden. Die Berechnung von Schallleistung und Schalldruck kann direkt in diesem Unterprogramm erfolgen. Da das Sub-VI noch nicht über die benötigten Anschlüsse zur Übergabe der berechneten Werte an das Hauptprogramm verfügt, müssen diese hinzugefügt werden.

Ggf. kann es notwendig werden das LabVIEW-Objekt „Einheiten Skalieren"[14] zu entfernen und durch eine selbst geschriebene Logarithmierung zu ersetzen, da der in diesem Objekt verwendete Skalierungsfaktor nicht einstellbar ist, sondern in Abhängigkeit von den Eingangsgrößen fest einprogrammiert ist und auch gar nicht für die Berechnung von Schallgrößen vorgesehen ist. Zwar ist es möglich, das Objekt zu ändern, davon sollte aber abgesehen werden, da sich diese Änderung dann auf **ALLE** Berechnungen auswirkt, die mit diesem Objekt durchgeführt werden – auch solche, die gar nichts mit diesem Programm zu tun haben. Eine Anpassung bzw. Kalibrierung des Eingangssignals mit Hilfe eins Faktors ist mit diesem Objekt ebenfalls nicht möglich, was bei der Berechnung des von Schallgrößen natürlich zu falschen Ergebnissen führt.

Den zur Kalibrierung des Eingangssignals benötigten Faktor sollte man direkt in das Hauptprogramm einbauen – oder besser eine Routine schreiben, die den Benutzer zwingt, diesen Faktor bei jedem neuen Start des Programms mit einem Unterprogramm mit Hilfe von Kalibrator und/oder Pistonfon zu ermitteln. Der Aufwand, der betrieben werden müsste, wenn dieser Faktor in die Einstellungen mit

[14] es handelt sich hier um ein LabVIEW Objekt, dass genauso wie ein Sub-VI geöffnet und bearbeitet werden kann.

aufgenommen würde steht in keinem Verhältnis zum Nutzen, da dann alle Cluster – sowohl in dem Unterprogramm zur Erstellung der Einstellungen, als auch im Hauptprogramm und den davon abhängigen Sub-VIs, geändert werden müssten. Die Multiplikation des Eingangssignals mit dem Faktor sollte wiederum im Sub-Vi erfolgen. Der benötigte Eingang muss ebenfalls hinzugefügt werden.

Da es sich hier um Schall handelt, der von einer Maschine in einen geschlossenen Kanal eingestrahlt wird, muss die Bestimmung des Schalldruck- und Schallleistungspegels mit Hilfe des in der DIN EN 25130 beschriebenen Kanalverfahrens erfolgen. Da sich die Schallwellen im Kanal nicht gleichmäßig ausbreiten, sind auch stehende Wellen in radialer Richtung möglich. Der Schalldruck ist dann über den Kanalquerschnitt nicht mehr konstant, und muss räumlich gemittelt werden. **[13]** Dies erreicht man durch Messung an mindestens 3 gleichmäßig auf den Umfang verteilten Messstellen, oder durch Verwendung eines Drehkanals.

Der Schalldruckpegel ist der RMS-Wert des aufgenommenen kalibrierten Signals. Der Kanal-Schallleistungspegel L_W ergibt sich aus dem räumlichen Mittelwert des Schalldruckpegels L_P und der Querschnittsfläche A des Kanals, sowie einigen frequenzabhängigen Korrekturen. **[13]**

Der prinzipielle Zusammenhang zwischen L_W und L_P ist in folgender Gleichung gegeben:

$$L_W = L_p + 10 \cdot \log \frac{A}{A_0}$$

6.3 Hinzufügen einer Drehkanalsteuerung

Annahme: Die Audiodaten sollen mit einem Drehkanal aufgenommen werden. Die Steuerung des Drehkanals soll durch das Programm erfolgen, die Motorsteuerung verfügt über eine serielle Schnittstelle.

Lösung:
Da die Aufzeichnung von Audiodaten bereits fertig programmiert ist, beschränkt sich diese Erweiterung auf die Programmierung der Motorsteuerung des Drehkanals.

Die Steuerung des Motors über die Schnittstelle ist prinzipiell ähnlich zu der Steuerung des Multiplexers mit SCPI Befehlen. Die verwendete Schnittstelle muss initialisiert werden, im Betrieb müssen die entsprechenden Steuerbefehle über die Schnittstelle gesendet werden und beim Beenden des Programms muss diese wieder geschlossen werden. Es ist empfehlenswert, die Initialisierung, das Senden der Befehle und das Schließen dem Hauptprogramm zu überlassen. Dazu können die Sequenzen, die bereits den COM-Port zur Steuerung des Multiplexers öffnen und schließen, um den entsprechenden Code erweitert werden.

Da die Drehbewegung synchron zur Aufnahme der Audiodaten erfolgen soll, ist es zweckmäßig das Senden des Start- und des Stopp-Befehls in die Sequenz einzufügen, in der auch die Soundaufnahme gesteuert wird. Die nötigen Steuerbefehle kann ein Sub-Vi generieren.

6.4 Hinzufügen eines weiteren Kanals

Annahme: Es soll ein weiterer Kanal (Nr. 113) zur Temperaturmessung mit einem Widerstandsthermometer hinzugefügt werden. Der Temperatursensor liefert ein Einheitssignal zwischen 0 und 10 V. Es wird die Temperatur am Austritt bestimmt.

Dieser Fall ist der schwierigste, da man beim Hinzufügen einer neuen Messgröße fast alle Module abändern muss. Unter Umständen kann es sogar einfacher sein das Modul neu zu schreiben als den alten Code zu verändern.

Mit Hilfe der folgenden Checkliste soll ein Überblick gegeben werden, was beim Einfügen eines zusätzlichen Kanals alles beachtet werden muss:

Konfiguration (config_sub.vi)	• Kalibrierfaktor und –offset einfügen • Lese- und Schreibroutine für die Konfigurationsdatei anpassen • Ausgangscluster „Kalibrierung Ausgang" anpassen
Steuerung (agilent_steuerung_sub.vi)	• Steuerstring zur Mittelwertbildung um einen Kanal erweitern : **„CALC:AVER:AVER? (@101:112)"** wird zu **„CALC:AVER:AVER? (@101:113)"** • Berechnungsformel für die benötigte Messzeit anpassen (ggf. durch Testen)
Umwandlung in Zahlen (string2num_sub.vi)	• Ausgangscluster erweitern • Clusterkonstante für die Elementbezeichnung des Ausgansclusters um das entsprechende Element erweitern • eine zusätzliche Indizierung (Index 12) hinzufügen und mit dem Ausgangscluster verbinden
Umwandlung in Messgrößen (kalib_sub.vi)	• Eingangscluster „gemessene Spannungen" löschen • Ausgangscluster von „string2num_sub.vi" kopieren, einfügen und in Bedienelement umwandeln. • Ausgangscluster „Kalibrierung Ausgang" aus „config_sub.vi" kopieren, einfügen und in

	Bedienelement umwandeln • eine weitere „Kalibrierstruktur" – bestehend aus Messwert = Spannung mal Faktor plus Offset einfügen • Ausgangscluster „Messwerte" und Clusterkonstante für die Elementbezeichnung des Ausgansclusters um das entsprechende Element erweitern
Berechnung der aerodynamischen Kenngrößen (berechnung_sub.vi)	• Eingangscluster „Messwerte" löschen, Ausgangscluster „Messwerte" von „kalib_sub.vi" kopieren, einfügen und in Bedienelement umwandeln. • Ausgangscluster „Berechnungsergebnisse" und Clusterkonstante für die Elementbezeichnung des Ausgansclusters um das/die entsprechende/n Element/e erweitern • Berechnung anpassen.
Hauptprogramm (aamdes_0.9.3d.vi)	• das versteckte Anzeigeelement Kalibrierung löschen, Ausgangscluster „Kalibrierung Ausgang" von „config_sub.vi" kopieren und einfügen. Eigenschaftsknoten (sichtbar/unsichtbar) in Rahmen 2 wieder anschließen • das Frontpanel um die gewünschten Anzeigeelemente erweitern • SCPI-Steuer-Strings in Rahmen 2 um einen Kanal erweitern (Scan-Liste und Spannungsmessung) • die Routine zur Erstellung des Dateikopfes um den entsprechenden Kanal und die gewünschten Ergebnisse erweitern • Die Objekte „Cluster nach Namen aufschlüsseln „ in Rahmen 3 um die neuen Elemente in „Messwerte" und „Berechnungsergebnisse" erweitern • Das Objekt „Array erstellen" um die benötigte Anzahl erweitern, Index – Liste erweitern ! ggf. die Indices in der Struktur zur grafischen Anzeige ändern
Auswerung (auswertung_sub.vi)	• Anpassen der Menüs für die Auswahl von Abszisse und Ordinate gemäß der der im Hauptprogramm gewählten Indices

7 Zusammenfassung

In der vorliegenden Diplomarbeit wurde ein aeroakustisches Messdaten-erfassungssystem mit dem Softwarepaket LabVIEW programmiert. Mit der erstellten Software lassen sich Kennliniendaten von Radialventilatoren an einem Normprüfstand aufnehmen und berechnen. Die dabei gewonnenen Datensätze kann man als Datenbank betrachten, die die Grundlage zur Berechnung von Kennfeldern und zur Auslegung von Ventilatoren darstellt. Für die Speicherung der aerodynamischen Daten wurden 2 Datenformate gewählt, die sowohl für den Menschen (z.B. mit einem Texteditor) als auch für bestimmte Anwendungen leicht lesbar sind. Neben der Erfassung der aerodynamischen Daten verfügt das Programm auch über die Möglichkeit, akustische Daten mit der Soundkarte des PCs aufzuzeichnen und als Windows-Audio-File abzuspeichern.

Die Benutzeroberfläche des Programms ist intuitiv zu bedienen und zeigt alle gemessenen Werte in übersichtlicher Form an. Die notwendigen Einstellungen werden zentral verwaltet und können in einer Konfigurationsdatei gespeichert werden. Mit einer integrierten Auswerteroutine ist der Benutzer in der Lage die gemessenen Daten der gesamten Messreihe grafisch darzustellen.

Das verwendete Messdatenerfassungs- / Schaltsystem der Firma Agilent erwies sich im Betrieb als sehr zuverlässig und gut für die Aufgabe geeignet. Die Möglichkeit temporäre Daten in dem integrierten nicht flüchtigen Speicher abzulegen und nur Mittelwerte auszulesen entlastet den PC. Die so frei gehaltenen Systemressourcen können nun für andere, rechenintensive Operationen, wie z.B. eine real-time Fast Fourier Transformation der audio-Daten verwendet werden.

Zusammenfassend lässt sich sagen, dass ein Programm erstellt wurde, das in der bestehenden Form den Ansprüchen an eine normgerechte Prüfstandsmessung genügt und darüber hinaus durch den modularen Aufbau und die detaillierte Dokumentation des Quellcodes auch offen für zukünftige Erweiterungen oder Änderungen ist.

Literaturverzeichnis

[1] Dubbel, Taschenbuch für den Maschinenbau, 18. Auflage 1995, Springer Verlag Berlin

[2] Lessenich / Reinartz: Laborskript „Bestimmung der Luftfeuchte"

[3] Bohl, W.: Technische Strömungslehre, 11. Aufl. 1998, Vogel Verlag Würzburg

[4] DIN EN ISO 5167-1 von 1995 und 1998

[5] VDI-Richtlinie „Abnahme- und Leistungsversuche an Ventilatoren"

[6] Kameier, Vorlesungsskript „Strömungsmaschinen", FH Düsseldorf 2000

[7] Taschenbuch der Mathematik, 25. Auflage 1991, Verlag Nauka Moskau (Anmerkung des Verfassers: bekannt als „Bronstein")

[8] Bohl, W.: Strömungsmaschinen 1, 7. Aufl. 1998, Vogel Verlag Würzburg

[9] Bohl, W.: Strömungsmaschinen 2, 5. Aufl. 1995, Vogel Verlag Würzburg

[10] Eggert, R.: Diplomarbeit zum Thema „Objektorientierte Programmierung eines 2-Kanal Frequenzanalysators unter DASYLab und LabVIEW", Juni 2000, Institut für Strömungsmaschinen, Fachhochschule Düsseldorf

[11] Agilent Technologies: Benutzerhandbuch Agilent 34970A Messdatenerfassungs-/Schaltsystem, 1. Ausgabe 1997

[12] Jamal, R., Hangestedt, A.: LabVIEW, Das Grundlagenbuch, 3. Auflage 2001, Addison-Wesley Verlag München

[13] L. Bommes, J. Fricke, K. Klaes (Hrsg.) : Ventilatoren, 1. Auflage 1994, Vulkan Verlag Essen

[14] Brüel & Kjær; Schallintensität, Broschüre zur Erläuterung der Grundlagen von Schallintensitätsmessung in Theorie und Praxis, Revision April 1991, Bräel & Kjær, Nærum (Dänemark)

[15] Kameier, Vorlesungsskript „Strömungsakustik", FH Düsseldorf 2001

[16] Schade, H., Kunz, E.: Strömungslehre, mit einer Einführung in die Strömungsmesstechnik von Jorg-Diter Vagt, 2. Auflage 1989, DeGruyter Verlag Berlin, New York

Abbildungsverzeichnis