D I P L O M A R B E I T

Interaktives Trainingsprogramm für die Laborübung „Gleichstrommaschine“

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

O. Univ. Prof. Dipl.-Ing. Dr. techn. Manfred Schrödl

und Betreuung durch

Dipl.-Ing. Andreas Eilenberger

E 372
Institut für elektrische Antriebe und Maschinen

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Amer Granov
Matr.-Nr. 9127549
Thaliastraße 149/24
1160 Wien

Mein besonderer Dank gilt o. Univ. Prof. Dipl.-Ing. Dr. techn. Manfred Schrödl der mich während der ganzen Arbeit unterstützt hat und Verständnis für meine Situation und die lange Dauer der Ausarbeitung gezeigt hat.

Mein Dank gebührt auch den Assistenten Dipl. Ing. Andreas Eilenberger und Dipl. Ing. Wolfgang Staffler die mich bei der Ausarbeitung unterstützt haben und am Aussehen des Applets stark mitgewirkt haben. Ich danke auch dem Dipl. Ing. Dragan Veljković der die ursprüngliche Idee für diese Arbeit gehabt hat.

Ich möchte mich bei meinen Freunden Dipl. Ing. Samir Ćerimović und Dipl. Ing. Almir Talić bedanken. Durch ihren Druck auf mich und ihre Hilfe ist diese Arbeit erst möglich geworden.

Ich bedanke mich besonders bei meiner Familie die mich immer unterstützt hat.

Vor allem danke ich Gott, dass ich diese Arbeit überhaupt habe machen können.

Kurzfassung

Die vorliegende Arbeit beschreibt das interaktive Trainingsprogramm für die Laborübung „Gleichstrommaschine“. Diese Laborübung ist eine Pflichtlehrveranstaltung im Studium Elektrotechnik, angeboten vom Institut für elektrische Antriebe und Maschinen an der TU Wien. Die Aufgabe der Diplomarbeit war ein Computerprogramm zu entwerfen, das zur Vorbereitung der Laborübung dienen soll. Die betreuenden Assistenten haben beobachtet, dass die Studenten viel Zeit während der Übung nur an Kennenlernen von Arbeitsplatz verlieren. Daher sollte ein Programm entwickelt werden, das den Studierenden möglichst realitätsnahe den Arbeitsplatz zeigt und ihnen die Möglichkeit bietet, die Übung virtuell durchzuführen.
Das Programm ist als ein Java Applet realisiert. Im ersten Kapitel dieser Diplomarbeit werden die theoretischen Grundlagen der elektrischen Maschinen erklärt. Besondere Aufmerksamkeit wurde auf die Gleichstrommaschine gelegt. Ihre Betriebsarten und Kennlinien wurden dabei ausführlich beschrieben.

Das zweite Kapitel befasst sich mit der Übungsdurchführung an sich. Zuerst werden die einzelnen Punkte der Übung sowie ihre Applet Ausführung erklärt. Zusätzlich sind auch die theoretischen Grundlagen der Teilübungen gegeben.

In dem darauf folgenden Kapitel wird dann die Sourcecode des Applets vorgestellt. Alle Klassen des Programms sind auf vier Bibliotheken (Packages) aufgeteilt. Die einzelnen Klassen werden in entsprechenden Unterkapiteln ausführlich behandelt und mit Sourcecode Abschnitten illustriert.

Kapitel vier illustriert die Bedienung des Applets. Es wird sowohl erklärt welche Dateien der Betreuer erstellen und konfigurieren muss, als auch welche Aktionen der Student unternehmen soll, damit der Applet wunschgemäß funktioniert.

Letztendlich werden die Hilfe-Dateien, die aus dem Applet aufrufbar sind, angeführt. Das ganze Hilfe-System ist auf HTML basiert. Für jede Teilübung wurden zwei HTML-Dateien zur Verfügung gestellt: Die Kurzversion, die in der Übungsmaske direkt angezeigt wird, und die erweiterte Help-Datei, die aus dem Applet aufgerufen und extern angezeigt wird.

Diese Diplomarbeit stellt ein Java-Applet vor, der die Studenten für die Durchführung der Laborübung „Gleichstrommaschine“ vorbereitet. Er bietet eine virtuelle Übungsdurchführung, einschließlich des Eingangstests an. Durch das Senden des Protokolls an den Betreuer per Email, ermöglicht der Applet eine Einsicht in die Vorbereitung des Studenten für die Übungsdurchführung im Labor. Darüber hinaus ist der Applet durch die gewählte Struktur leicht modifizierbar, was für die künftigen Inhaltsänderungen der Laborübung von der besonderen Bedeutung ist. Somit stellt er ein sehr effizientes Lernwerkzeug dar.

Summary

This diploma thesis describes the interactive training-program for the laboratory tutorial “DC machine”. This laboratory tutorial is a compulsory course at the electric engineering studies, given at the Institute of Electrical Drives and Machines at the Vienna University of Technology. The aim of the diploma thesis was to develop a computer program that is going to serve as a preparation for this laboratory tutorial. As teaching assistants have observed that students lose a noticeable amount of time in order to get used to the workspace environment, a need has arisen to create a computer simulation program which is able, in a most realistic way, to represent the given workspace and make it possible to conduct the tutorial virtually.

The program is realized as java-applet. In the first chapter theoretical bases of electrical machines are explained. Special attention is dedicated to DC machine, whereby its operating modes and characteristics are presented in details.

The second chapter addresses the conducting of the tutorial itself, its individual parts as well as their applet implementation. Additionally, the theoretical basics for the parts of the tutorial are provided.

The following chapter presents the source code of the applet. The classes of the program are classified in four libraries (packages). Particular classes are described in corresponding subsections, illustrated with source code.

The fourth chapter handles operating of the applet. On one hand, there is a description of files to be created and configured by the teaching assistant, on the other hand necessary actions of the student in order for the applet to operate as predicted.

Finally, help-files that can be called from the applet are presented. The entire help-system is based on HTML. For each part of the tutorial there are two HTML files at disposal: shortened version that appears directly in the applet mask and detailed help-file that can be called from within the applet and appears externally.

This diploma thesis presents a java-applet that helps students to prepare themselves for the laboratory tutorial “DC machine”. It enables virtual performance of the tutorial including the introductory test for the students. The applet also sends the protocol via e-mail to the assistant, enabling the insight of the student's preparation work for performing the tutorial in the laboratory. Moreover, the chosen structure of the applet enables its simple adaptations, what is of great importance in case of further modifications of the tutorial. This makes the Applet a very efficient learning tool for students as well as useful teaching tool for assistants.

Inhalt

Abbildungsverzeichnis

Abkürzungsverzeichnis

FEG	Fremderregter Generator
GSM	Gleichstrommaschine
GUI	Graphical user interface
NSG	Nebenschlussgenerator
NZ	Neutrale Zone
URL	uniform resource locator
VG	Verbundgenerator

Aufgabestellung

Die Aufgabe dieser Diplomarbeit ist es ein Java Applet zu entwerfen der zur Vorbereitung der Laborübung "Antreibe" im Studium der Elektrotechnik an der TU Wien dienen soll. Die betreuenden Assistenten haben beobachtet, dass die Studenten viel Zeit während der Übung nur an Kennenlernen von Arbeitsplatz verlieren. Dieser Applet soll daher den Studierenden den Arbeitsplatz zeigen und die Möglichkeit bieten die durchzuführenden Messungen auch realitätsnah zu simulieren.

Die Wahl der Programmiersprache fiel auf Java weil sie eine hohe und mächtige Sprache ist und weil sie durch Applets eine Internetplattform bietet, die es ermöglicht ein breit verfügbares Programm zu erstellen. Es wurde auch überlegt das Ganze in C/C++ zu lösen und dann als Download oder auf einer CD zur Verfügung zu stellen. Da aber die erste Idee ein internetbasiertes Training war, ist man bei Java geblieben. Außerdem hatte ich durch meinen Beruf schon eine große Erfahrung mit C++ sammeln können, daher war die Java Lösung bevorzugte Wahl, um diese Sprache endlich mal richtig kennen zu lernen.

1 Theoretische Grundlagen

Das folgende ist größten Teiles von dem Skriptum Grundlagen der Elektrotechnik III von der TU Berlin übernommen.

Alle elektrischen Maschinen kann man unter dem Sammelbegriff elektromagnetische Energiewandler zusammenfassen. Als Motoren dienen sie der Umwandlung von elektrischer in mechanische Energie, indem sie die im Netz vorhandene elektrische Energie (Spannung x Strom x Zeit) zur Erzeugung mechanischer Energie (Drehmoment) an ihrer Welle verwenden, die dann zur Fortbewegung von Fahrzeugen oder zum Betreiben von Werkzeugmaschinen genutzt wird.
Bei den Generatoren verläuft der Vorgang umgekehrt; sie nehmen mechanische Energie auf, d.h. sie müssen angetrieben werden, z.B. durch einen Dieselmotor, eine Dampf-, Wind- oder Wasserturbine, und wandeln diese Energie in elektrische Energie um, die in Form von Spannung und Strom zur Verfügung steht, solange der Generator angetrieben wird.
Elektrische Maschinen werden als ruhende (Transformatoren) oder rotierende elektrische Maschinen klassifiziert. Diese Diplomarbeit befasst sich mit der Gleichstrommaschine (GM) die eine rotierende als Stromwender-/Kommutatormaschine gebaute elektrische Maschine ist. Das Hauptanwendungsgebiet der GM ist als Antrieb für Straßenbahn oder U-Bahn aber auch als Regel- bzw. Stellantrieb.

1.1 Theoretische Grundlagen elektrischer Maschinen

Elektrische Maschinen sind von ihrem physikalischen Grundprinzip elektromechanische Energiewandler. Für den Umwandlungsprozess benötigen Elektrische Maschinen immer ein magnetisches Feld (einen oder mehrere magnetische Kreise). Ohne Magnetfeld gibt es keine Spannungsinduktion in einem bewegten elektrischen Leiter bzw. keine Krafterzeugung durch einen elektrischen Strom. Das sind die beiden Grundfunktionen aller elektrischen Maschinen. Elektrische Maschinen bestehen somit aus elektrischen, magnetischen und mechanischen Kreisen (Subsystemen), deren Wechselwirkung und gegenseitige Verknüpfung über bestimmte Grundgesetze mathematisch beschrieben werden können.

1.1.1 Entstehen einer Spannung (Generatorprinzip)

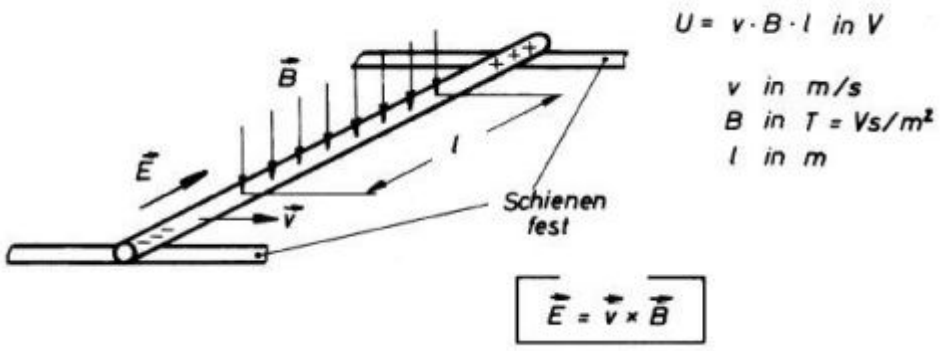


Bild 1.1: Generatorprinzip

Wird ein Leiter mit der Geschwindigkeit $\vec{v}$ auf zwei ruhenden, elektrisch leitenden Schienen bewegt und wirke auf die Länge *l* dieses bewegten Leiters dauernd ein homogenes Magnetfeld der Stärke B ein so wird eine elektrische Feldstärke induziert deren Vektor sich darstellen lässt als:

$$\vec{E} = \vec{v}\, x\, \vec{B}$$

E [V/m]; v [m/s]; B [T] oder [Vs/m^2]

Unter der Annahme, dass $\vec{E}, \vec{v}$ und $\vec{B}$ senkrecht zueinander stehen, ergibt sich für die induzierte Spannung (Skalar):

$$U = v \cdot B \cdot l$$

U in V; v in m/s; B in T; *l* in m

Es handelt sich um die induzierte Spannung in einem Einzelleiter.

1.1.2 Entstehen einer Kraft (Motorprinzip)

Im Bild 1.2 ruht ein Leiter in einem homogenen Magnetfeld der Stärke $\vec{B}$, das auf der Länge *l* wirksam ist. Schickt man durch den Leiter einen gerichteten elektrischen Strom I von der Stromdichte $\vec{G}$ so wird längs der Länge *l* auf die Leitereinheit eine spezifische Kraft ausgeübt, deren Vektor sich darstellen lässt als:

$$\frac{\vec{F}}{l} = I\,(\vec{e}\, x\, \vec{B})$$

$$\frac{F}{l}\left[\frac{N}{m}\right]; I\,[A]; B\,[T]\; oder\; \left[\frac{Vs}{m^2}\right]; \vec{e}\; Einheitsvektor$$

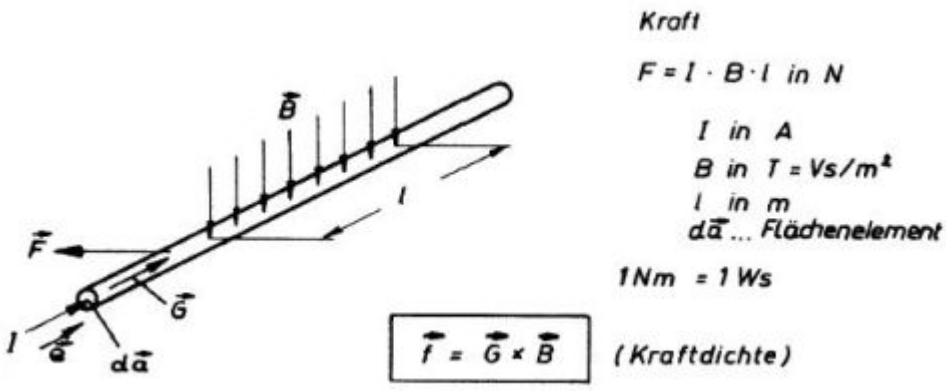


Bild 1.2: Motorenprinzip

Wenn wie im vorliegenden Fall $\frac{F}{l}$, I und B senkrecht aufeinander stehen, folgt für die entwickelte Gesamtkraft (Skalar) auf den Einzelleiter:

$$F = I \cdot B \cdot l$$

F in N; I in A; B in T; *l* in m.

1.2 Gleichstrommaschine

Der Energiewandler Gleichstrommaschine ist entwicklungsgeschichtlich die älteste elektrische Maschine. Die Gleichstrommaschine hat wegen ihrer guten Regeleigenschaften immer noch eine beachtliche Bedeutung im Bereich der Traktion und der Servoantriebe. In der Automobiltechnik gibt es ebenfalls viele Gleichstrommotoren in unterschiedlichen Bauformen.

1.2.1 Aufbau der Gleichstrommaschine

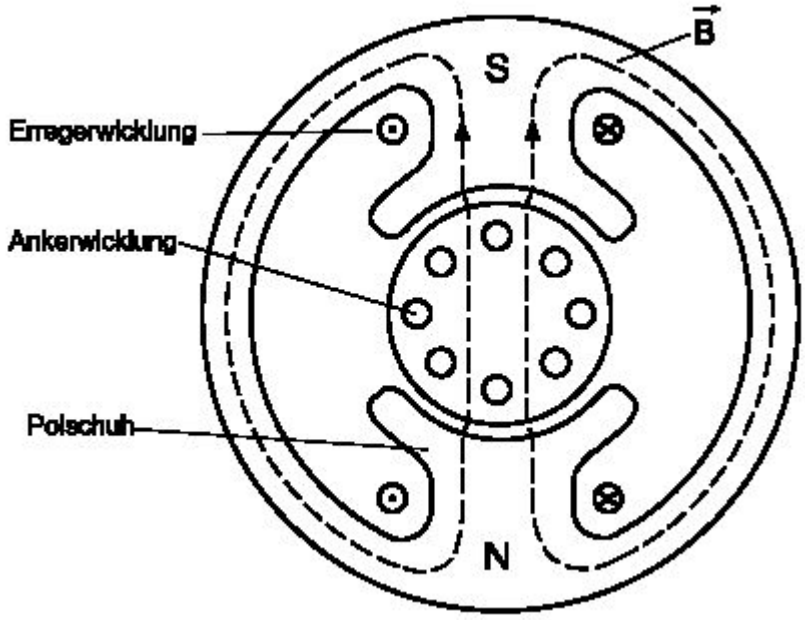


Bild 1.3: Aufbau der Gleichstrommaschine

Das Bild 1.3 zeigt den grundsätzlichen Aufbau einer GSM. Die Erregerwicklung befindet sich bei GSM üblicherweise im Ständer (Außenpolmaschine). Der Läufer (Rotor) trägt die Wicklung, in die bei Bewegung eine Spannung induziert wird (Ankerwicklung). Die Erregerwicklung erzeugt ein zeitlich und räumlich konstantes Feld. In diesem Feld dreht sich die Ankerwicklung. Der Rotor trägt eine Reihe von Wicklungen, die über den Kommutator (Stromwender) mit der speisenden Gleichspannungsquelle verbunden sind.

Zur Erläuterung der Funktion betrachten wir zunächst eine einzige Windungsschleife in einem homogenen, zeitlich konstanten Magnetfeld. Die induzierte Spannung u_{12} lässt sich folgendermaßen berechnen:

$$u_{12} = \frac{d\phi}{dt}$$

mit $\phi = \phi_0 \cdot \cos(2\pi nt)$ und $\phi_0 = -B \cdot A$ folgt

$$u_{12} = 2\pi nBA\sin(2\pi nt) = \omega BA\sin(\omega t)$$

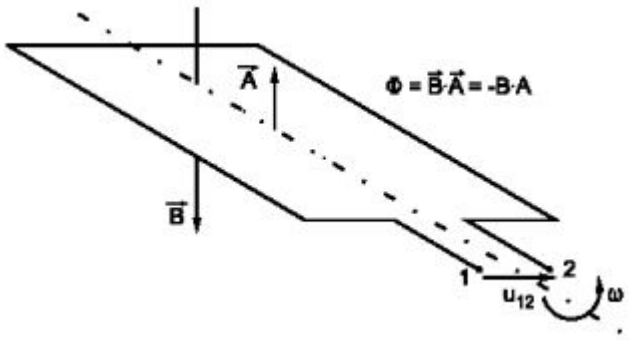


Bild 1.4: Ankerwicklung im homogenen, zeitlich konstanten Magnetfeld

Bild 1.5 und 1.6 zeigen eine zweipolige Maschine sowie die Feldverläufe.

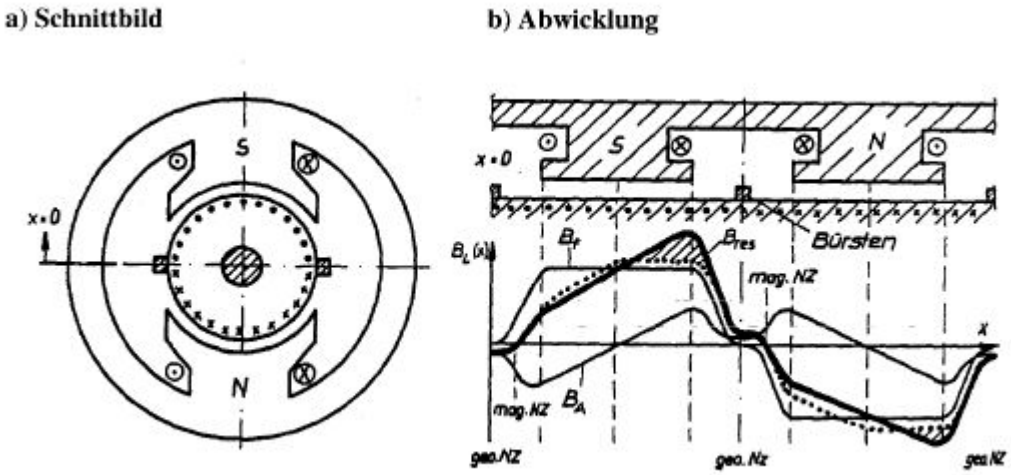


Bild 1.5: Gleichstrommaschine ohne Wendepole

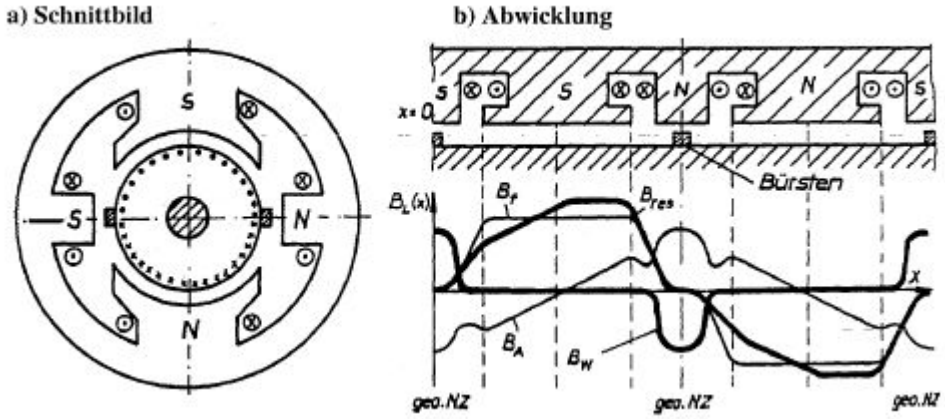


Bild 1.6 Gleichstrommaschine mit Wendepolen

1.2.2 Der Kommutator

Ein Kommutator für eine Windungsschleife besteht aus zwei voneinander isolierten Ringhälften, auf denen Bürsten schleifen (Bild 1.7). Wechselt die in der Schleife induzierte Spannung ihr Vorzeichen, wechseln auch die Bürsten auf die jeweils andere Ringhälfte. An den Bürsten ist eine pulsierende Gleichspannung abgreifbar. Durch mehrere räumlich versetzte Windungsschleifen (Wicklungen) erhält man eine Glättung des Spannungsverlaufs.

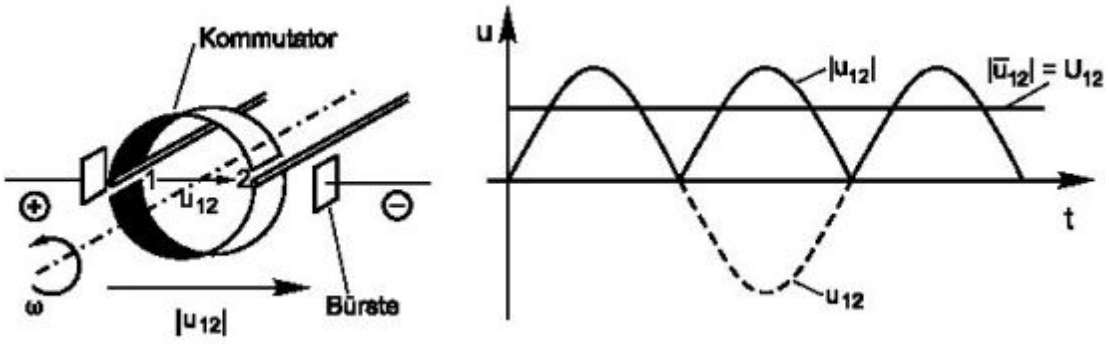


Bild 1.7: Das Kommutatorprinzip

1.2.3 Grundgleichungen der GSM

Wie schon im Kapitel 1.1 gezeigt gilt im Allgemeinen: Bewegt sich ein Leiter in einem Magnetfeld, dann entsteht in diesem Leiter eine elektrische Feldstärke.

$$-\vec{E} = \vec{v} \times \vec{B}$$

Im Luftspalt einer Maschine stehen B, v sowie der Leiter senkrecht aufeinander. Für einen Ankerleiter der Länge l ergibt sich:

$$U_q = -E \cdot l = l \cdot B \cdot v \cdot \sin 90°$$

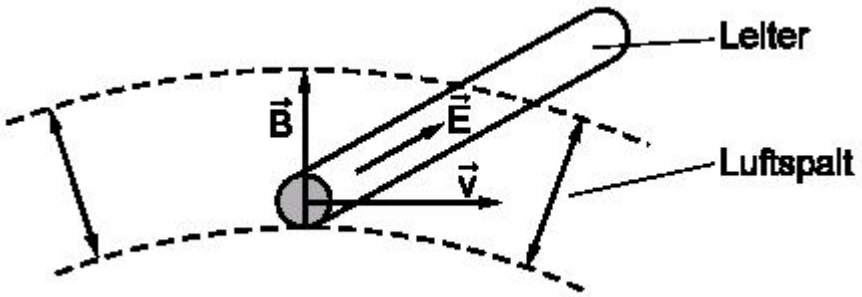


Bild 1.8: Zur Berechnung der induzierten Spannung

Die an den äußeren Klemmen abgreifbare Spannung entspricht bei leerlaufender Maschine ($I = 0$) der inneren Quellenspannung u_q entsprechend der Leiterzahl Z.

$$U_q = Z \cdot B \cdot l \cdot v$$

Wir führen anstelle der Flussdichte B den magnetischen Fluss Φ ein (Bild 1.9),

$$B = \frac{\Phi}{A} = \frac{\Phi}{b_p \cdot l}$$

und weiter anstelle der Umfangsgeschwindigkeit v die Drehzahl n

$$v = d \cdot \pi \cdot n$$

Setzen wir die Größen in die Gleichung für u_q ein, so erhalten wir nach entsprechendem Kürzen als Ergebnis die Größengleichung

$$\begin{aligned} U_q &= Z \cdot \frac{\Phi}{b_p \cdot l} \cdot l \cdot d \cdot \pi \cdot n = Z \cdot \frac{r}{b_p} \cdot 2 \cdot \pi \cdot n \cdot \Phi \\ &= k_1 \cdot n \cdot \Phi = Z \cdot \frac{p}{\pi} \cdot \omega \cdot \Phi \\ &= k_2 \cdot \omega \cdot \Phi = k_2 \cdot 2 \cdot \pi \cdot n \cdot \Phi \end{aligned} \qquad (1.1)$$

mit

$\cdots Z \rightarrow$ *Zahl der in Reihe liegenden Leiter*

$\cdots (r\,\pi)/b_p \rightarrow$ *Polpaarzahl p bei Polbedeckung* $\alpha = 1$

$\cdots 2\,\pi\,n \rightarrow$ *Winkelgeschwindigkeit* $\omega\,([n] = s^{-1})$

$\cdots \Phi \rightarrow$ *Erregerfluss* $([\Phi] = Vs = Wb)$

$\cdots \alpha \rightarrow$ *Polbedeckung* $\alpha = b_p/\tau$ *mit* τ *als Polteilung*

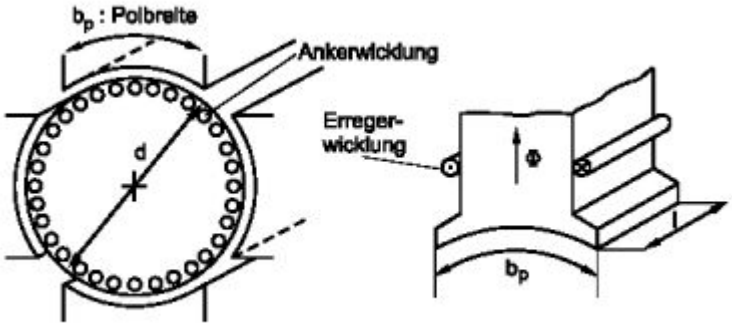


Bild 1.9: Zur Berechnung der Maschinenkonstanten

Für eine gegebene Maschine sind also Φ und n die betrieblichen Einflußgrößen. Die Maschinenkonstante C_{Masch} ist durch die Konstruktion der Maschine festgelegt. Sie kann durch Messungen bestimmt werden. Die Formel (1.1) gilt für jede Betriebsart der Maschine. Eine Quellenspannung u_q wird bei Generator- und Motorbetrieb induziert. Die an den Bürsten abgreifbare Gleichspannung ist durch die Frequenz $f_a = 0$ gekennzeichnet. (Bild: 1.10). Im Innern der Wicklung tritt dagegen eine Wechselspannung $f_i = p \cdot n$ auf. Die innere Frequenz kann in weiten Grenzen beliebige Werte annehmen. Dadurch ergibt sich bei der Gleichstrommaschine eine große Freizügigkeit bezüglich der Drehzahl.

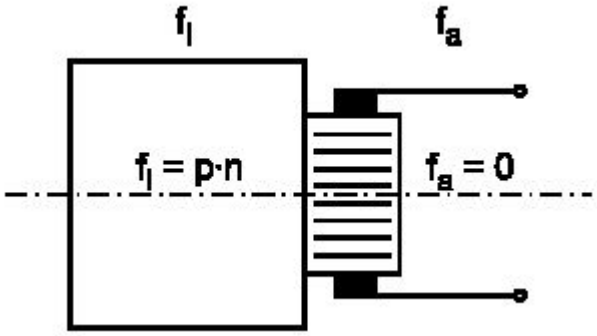


Bild 1.10: Kommutator als mechanischer Frequenzwandler

1.2.4 Leistung und Drehmoment

Sobald die Maschine belastet ist, also ein Ankerstrom I_A fließt, ergibt sich eine Kraftwirkung und damit ein inneres Drehmoment. Das gilt für jede Betriebsweise. Ein einfacher Weg zur Bestimmung des inneren Drehmomentes M_I ergibt sich über die innere Leistung:

$$P_i = U_q \cdot I_A$$

Die Gleichung für das Drehmoment lautet:

$$M_i = \frac{P_i}{\omega} = \frac{U_q \cdot I_A}{2 \cdot \pi \cdot n} \quad (1.2)$$

Setzt man die Gleichung (1.1) in (1.2) ein, so folgt:

$$M_i = Z \cdot \frac{p}{\pi} \cdot \frac{2 \cdot \pi \cdot n}{2 \cdot \pi \cdot n} \cdot I_A \cdot \Phi = Z \cdot \frac{p}{\pi} \cdot I_A \cdot \Phi$$

Die Faktoren für die Maschine stellen wieder eine Konstante dar:

$$M_i = k_2 \cdot \Phi \cdot I_A$$
$$k_1 = k_2 \cdot 2 \cdot \pi$$

1.2.5 Ankerrückwirkung

Der Ankerstrom I (Strombelag auf dem Rotor) ist für das Entstehen einer Schubkraft bzw. eines Drehmoments M_d unerlässlich; sein eigenes Magnetfeld hingegen ist im höchsten Maße unerwünscht. Dieses hat gerade im Bürstenbereich, wo eine feldfreie Zone gewünscht wird, seinen Höchstwert, und es verzerrt das Erregerfeld im Sinne einer Flussschwächung. Bild 1.11 zeigt die Feldverteilung für einen stromführenden Rotor.

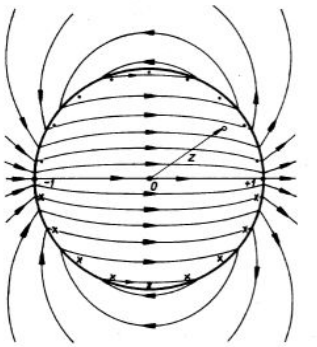


Bild 1.11: Feldverteilung an einem stromdurchflossenen Rotor

Wirkung des Ankerfelds	Gegenmaßnahme
1) Entstehen eines unerwünschten starken Feldes in der Wendezone	Wendepole (Wicklungen vom Ankerstrom durchflossen)
2) Flußschwächung durch Feldverzerrung und Sättigung der Polschuhkanten	Einbau von Kompound- und Kompensationswicklungen (vom Ankerstrom durchflossen)

Die Wirkung der Kompensationswicklung ist anhand von Bild 1.12 zu erkennen, wenn durch die Kompensationswicklung der Ankerstrom fließt. Ohne diese Wicklung erhält man die Feldverteilung gemäß Bild 1.13.

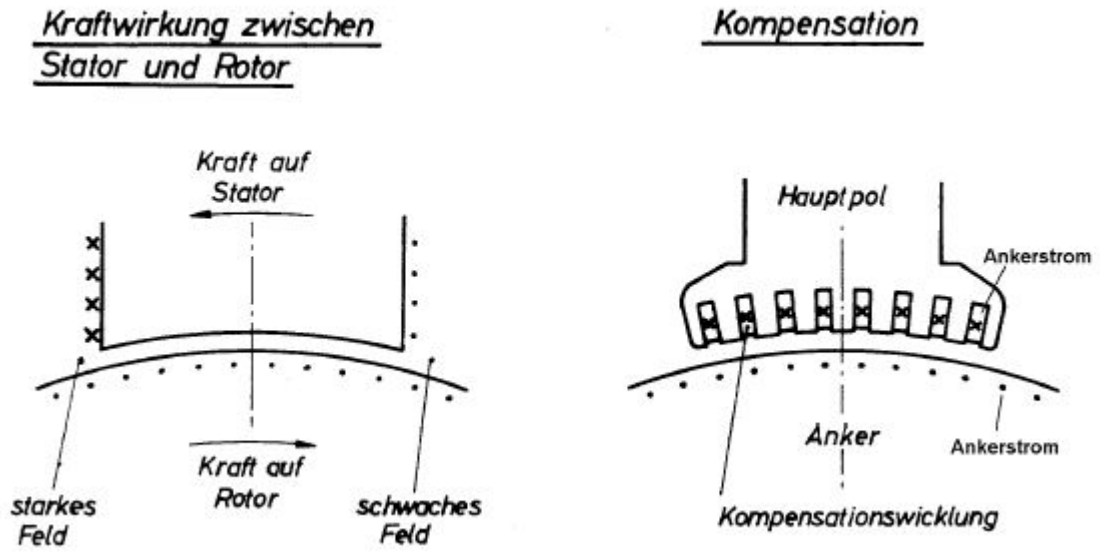


Bild 1.12: Wirkung von Kompensationswicklung

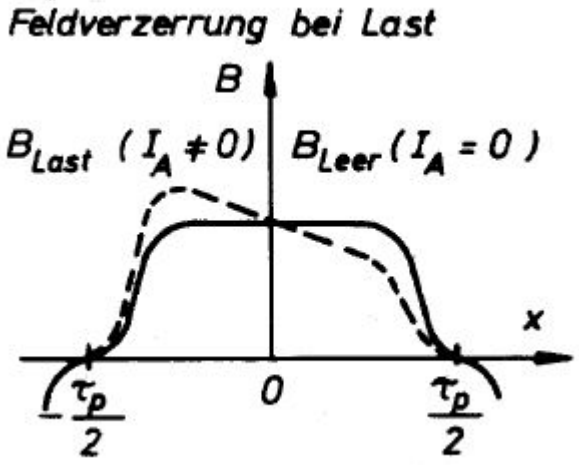


Bild 1.13: Feldverlauf der unkompensierten Gleichstrommaschine

1.3 Gleichungen und Kennlinien der Gleichstrommaschine

Das Betriebsverhalten der Gleichstrommaschine lässt sich durch wenige Gleichungen weitgehend beschreiben. Bei der Herleitung der „Spannungsgleichung“ und der „Drehmomentgleichung“ geht man am besten von der induzierten Spannung in einem Einzelleiter bzw. der Kraft auf den Einzelleiter aus.
Die induzierte Spannung für einen Einzelleiter des Ankers beträgt:

$$U_i = v \cdot B \cdot l$$

Mit $\Phi = B \cdot \tau \cdot l$, $\tau = \pi D / 2\mathrm{p}$, $v = \omega D / 2$ und $\omega = 2\pi n$ folgt, wenn die wirksame spannungsbildende Leiterzahl z/2a beträgt

$$U_i = \frac{z}{2\mathrm{a}} \cdot \frac{2\mathrm{p}}{2\pi} \cdot \omega \cdot \Phi = \frac{z}{2\mathrm{a}} \cdot 2\mathrm{p} \cdot n \cdot \Phi$$

$$U_i = k_1 \cdot n \cdot \Phi$$

Für den vollständigen Ankerkreis gilt

$$U = \underbrace{k_1 \cdot n \cdot \Phi + IR + \Delta U_{Bü}}_{\text{stat. Betr.}} + \underbrace{\frac{d}{dt}(L \cdot I)}_{\text{dyn. Betr.}}$$

Die Kraft auf den Einzelleiter im Anker beträgt

$$F = \frac{I}{2\mathrm{a}} \cdot B \cdot l$$

Somit ist das Drehmoment folgendermaßen gegeben

$$M = z \cdot F \cdot D/2 = \frac{z}{2\mathrm{a}} \cdot \frac{D}{2} \cdot I \cdot B \cdot l$$

$$M = \frac{z}{2\mathrm{a}} \cdot \frac{2\mathrm{p}}{2\pi} \cdot I \cdot \Phi$$ und schließlich

1.3.1 Gleichstrommaschine als Motor

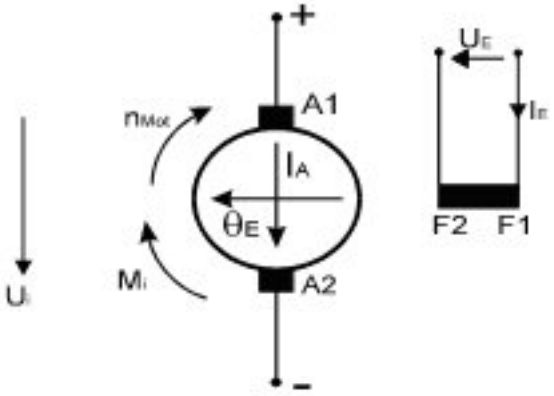


Bild 1.14: Gleichstrommaschine im Motorbetrieb

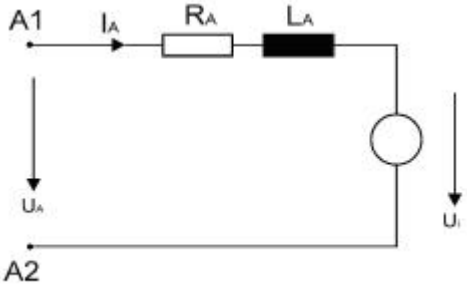


Bild 1.15: Ersatzschaltbild des Ankerkreises im Motorbetrieb

Im stationären Fall gelten für den Gleichstrommotor die Spannungsgleichung

$$U_A = U_i + I_A \cdot R_A + \Delta U_{Bü}$$

$$U_i = k_1 \cdot n \cdot \phi$$

$$\Delta U_{Bü} = 2\mathrm{V}$$

und die Momentengleichung

$$M_i = \frac{P_i}{\omega} = \frac{U_i \cdot I_A}{2\pi \cdot n} = k_2 \cdot \phi \cdot I_A$$

mit den Maschinenkonstanten

$$k_1 = \frac{z \cdot p}{a}$$

$$k_2 = \frac{1}{2\pi} z \cdot \frac{p}{a}$$

Dabei ist:
[z] - Gesamtleiterzahl
[2p] - Anzahl der Pole
[2a] - Anzahl der parallelen Ankerwicklungszweige

Für die Schaltung der Erregerwicklung (Feldwicklung) mit der Ankerwicklung gibt es mehrere Möglichkeiten wodurch die Charakteristiken der Maschine bestimmt werden können.

1.3.1.1 Fremderregter Gleichstrommotor

Beim fremderregten Gleichstrommotor wird der Erregerstrom von einer zweiten Gleichspannungsquelle U_E geliefert und kann daher unabhängig von Laststrom oder Drehzahl eingestellt werden.

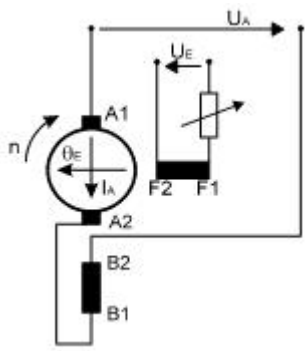


Bild 1.16: Schaltung des fremderregten Gleichstrommotors

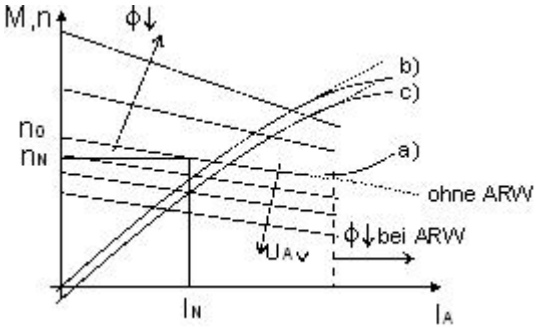


Bild 1.17: Kennlinien des fremderregten Gleichstrommotors
a) Drehzahlverlauf
b) Momentenverlauf - inneres Moment
c) Momentenverlauf - äußeres Moment

1.3.1.2 Gleichstrom-Nebenschlußmotor

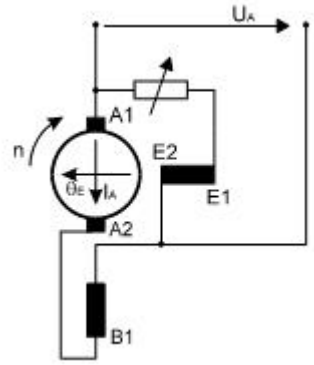


Bild 1.18 Schaltung des Nebenschlußmotors

Beim Nebenschlußmotor wird die Erregerwicklung parallel mit der Ankerwicklung geschaltet. Die Kennlinien sind gleich wie beim fremderregten Motor.

1.3.1.3 Gleichstrom-Reihenschlußmotor

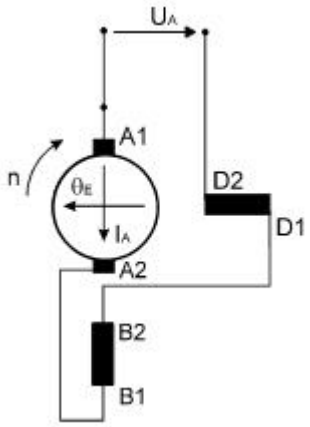


Bild 1.19 Schaltung des Reihenschlußmotors

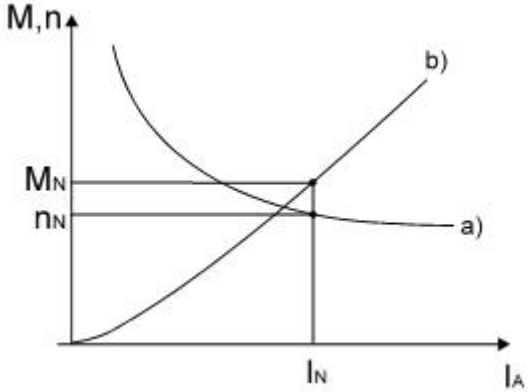


Bild 1.20 Kennlinien des Reihenschlußmotors
a) Drehzahlverlauf
b) Momentenverlauf - inneres Moment

Beim Reihenschlußmotor werden die Erregerwicklung und die Ankerwicklung in der Reihenschaltung miteinander geschaltet. Hauptfeld ist hier proportional dem Laststrom. Reihenschlußmotor ist daher nicht leerlauffest ($I_A \rightarrow 0 \rightarrow \varphi_H \rightarrow 0 \rightarrow n \uparrow$). Verwendung als Antriebsmotor für Fahrzeuge (z.B. Straßenbahn).

1.3.1.4 Gleichstrom-Verbundmotor

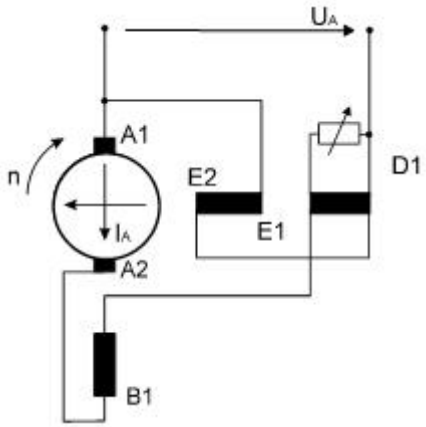


Bild 1.21 Schaltung des Verbundmotors

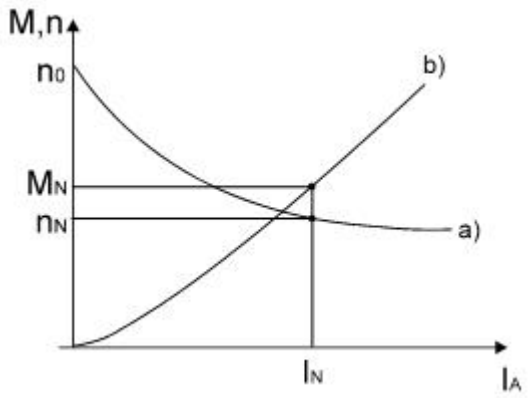


Bild 1.22 Kennlinien des Verbundmotors
a) Drehzahlverlauf
b) inneres Moment

Die Erregung wird aus zwei Wicklungen aufgebaut, z.B. Reihenschlußwicklung und Nebenschlußwicklung.

1.3.2 Gleichstrommaschine als Generator

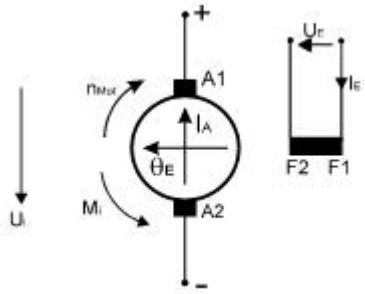


Bild 1.23:Gleichstrommaschine im Generatorbetrieb

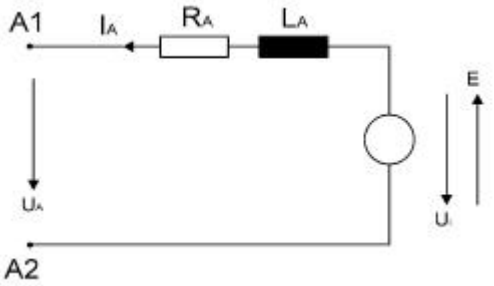


Bild 1.24:Ersatzschaltbild des Ankerkreises im Generatorbetrieb

Für den Gleichstromgenerator gilt stationär:

$$U_A = U_i - I_A \cdot R_A - \Delta U_{Bü} \quad .$$

U_i und $\Delta U_{Bü}$ sind dabei gleich wie beim Gleichstrommotor.

1.3.2.1 Fremderregter Gleichstromgenerator

Beim fremderregten Gleichstromgenerator wird der Erregerstrom von einer zweiten Gleichspannungsquelle geliefert und ist daher unabhängig von Laststrom oder Drehzahl einstellbar.

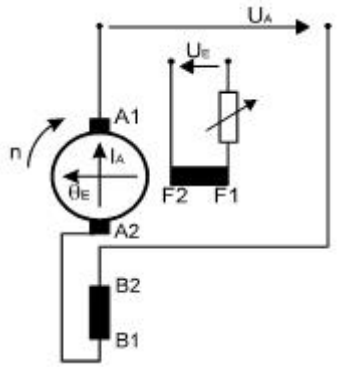


Bild 1.25: Schaltbild des fremderregten Generators

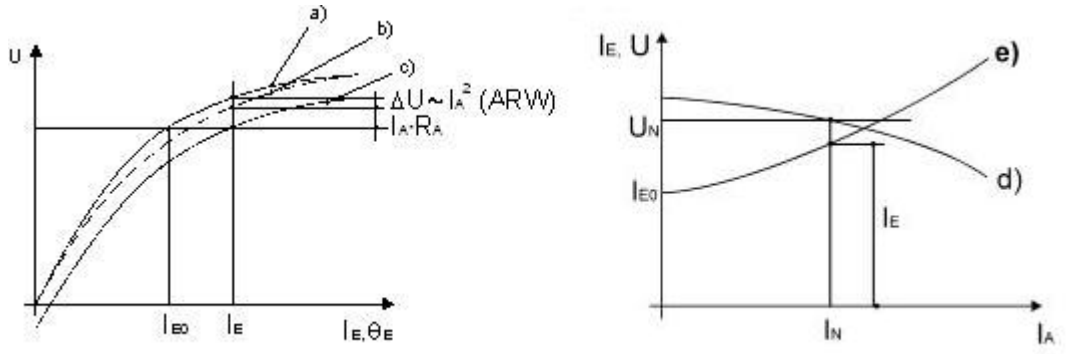


Bild 1.26: Kennlinien des fremderregten Gleichstromgenerators

a) Leerlaufkennlinie $U0(I_E)$, $I_A = 0$, $n = konst$

b) Innere Kennlinie $Ui(I_E)$, $I_A = konst$, $n = konst$

c) Belastungskennlinie $U_A(I_E)$, $I_A = konst$, $n = konst$

d) Äußere Kennlinie $U_A(I_A)$, $I_E = konst$, $n = konst$

e) Regulierkennlinie $I_E(I_A)$, $U = konst$, $n = konst$

1.3.2.2 Gleichstrom-Nebenschlußgenerator

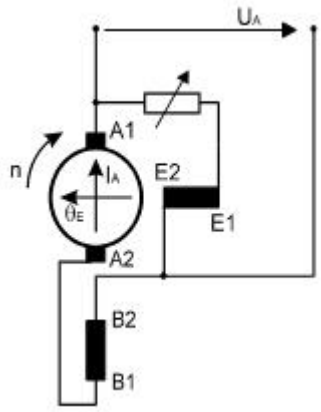
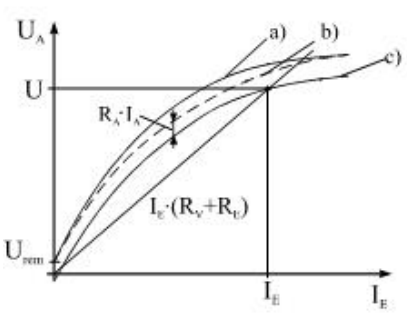


Bild 1.27:Schaltung des Nebenschlußgenerators



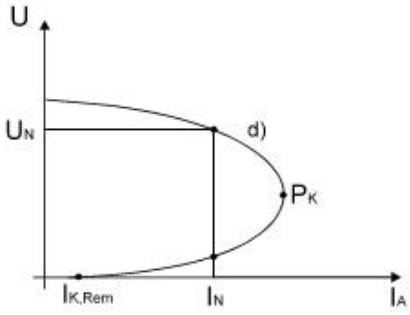


Bild 1.28: Kennlinien des Nebenschlußgenerators

a) Leerlaufkennlinie U0(IE), I_A = 0, n = konst

b) Innere Kennlinie Ui(IE), I_A = konst, n = konst

c) Belastungskennlinie U_A(IE), I_A = konst, n = konst

d) Äußere Kennlinie $U_A(I_A)$, $R_E + R_V$ = konst, n = konst

Beim Nebenschlussgenerator wird der Erregerstrom von dem Generator selbst geliefert und mittels Vorwiderstand RV eingestellt. Der Anlauf ohne externe Spannung ist nur mit Hilfe der Remanenz und eines geeigneten Vorwiderstandes möglich.

1.3.2.3 Gleichstrom-Verbundgenerator

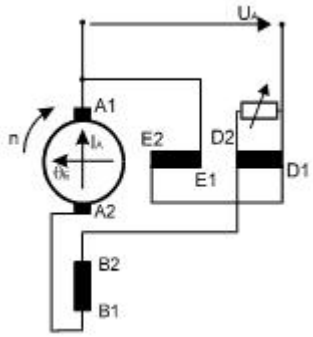


Bild 1.29: Schaltung des Verbundgenerators

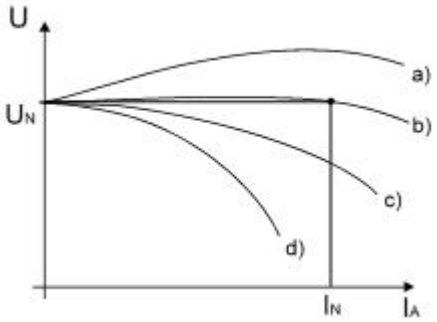


Bild 1.30: Äußere Kennlinie des Verbundgenerators

a) Überkompoundierung

b) Kompoundierung

c) Unterkompoundierung

d) Gegenkompoundierung

Die beim Verbundgenerator benutzte Reihenschlußwicklung erhöht das Hauptfeld proportional zum Ankerstrom, wodurch die Ankerrückwirkung reduziert wird.

1.3.2.4 Gleichstrom-Reihenschlußgenerator

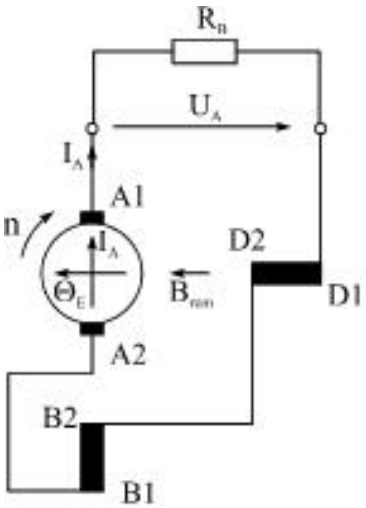


Bild 1.31: Schaltbild des Reihenschlußgenerators

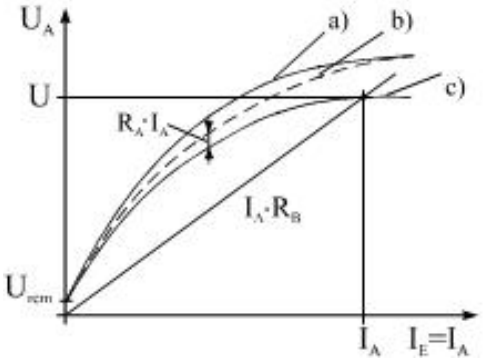


Bild 1.32: Kennlinien des Reihenschlußgenerators
a) Leerlaufkennlinie $U_0(I_E)$, $I = 0$, $n = konst.$
b) Innere Kennlinie $U_i(I)$, $n = konst$
c) Äußere Kennlinie $U(I)$, $n = konst$

Bei dem Reihenschlußgenerator ist das Hauptfeld proportional dem Ankerstrom. Diese Art der Schaltung wird als Bremsvorrichtung bei Straßenbahnen angewendet.

1.4 Schrage Motor

Bei der Übung wird ein Schrage Motor als Antrieb für den Generator verwendet. Deswegen wird hier auch eine kurze theoretische Abhandlung über Schrage Motor gehalten.

Als Schrage Maschine bzw. Schrage Motor wird der läufergespeiste Drehstrom-Nebenschluss-Kommutatormotor bezeichnet. Sie ist eine Drehstrom-Kommutatormaschine deren Läufe sowohl eine Schleifriingwicklung als auch eine Kommutatorwicklung trägt. Die Schleifringwicklung liegt am Speisenden Primärnetz, und die Ständerwicklung ist über die Kommutatorwicklung kurzgeschlossen. Letzte prägt der Ständerwicklung eine durch Bürstenverstellung veränderbare, nahezu lastunabhängige Zusatzspannung auf und wird deshalb auch Regelwicklung genannt. Der Bürstensatz ist ein Sechsbürstensatz, an den die Ständerwicklung in offener Schaltung angeschlossen ist.

Diese Maschine arbeitet somit wie ein Dreiphasen-Induktionsmotor, dessen Läufer über dem konstruktivintegrieren Frequenzwandler eine lastunabhängige schlupffrequente Zusatzspannung zur Drehzahlstellung und zur Verbesserung des Leistungsfaktors aufgeprägt erhält. Die Drehzahl-Drehmoment-Kennlinien ähneln denen des Dreiphasen-Induktionsmotors, insbesondere bezüglich des Nebenschlussverhaltens im normalen Arbeitsbereich.

Sie lassen sich durch Betätigen einer Bürßtenverstelleinrichtung in einem gewissen Bereich (Drehzahlstellbereich bis 1:3 und darüber) verschieben, sodass eine wirksame Drehzahlstellung möglich ist.

2 Übungsdurchführung

In diesem Teil werden teilweise die Abschnitte aus dem Skriptum zur Übung Labor Elektrische Antriebe von der TU Wien zitiert.

Die Übung "Messungen an Gleichstrommaschinen" ist in 6 Teilen aufgeteilt. Diese Teilübungen sollen der Einführung im Prozess der Identifikation der Maschine dienen. Als Objekt an dem die Messungen durchgeführt werden steht ein Generator zur Verfügung der dann auf verschiedene Weise verbunden wird um verschiedene Maschinen darzustellen.
Die durchzuführenden Teilübungen sind:

- Einstellung der Bürsten in die neutrale Zone
- Fremderregter Generator
- Nebenschlussgenerator
- Prinzip der Selbsterregung. (Nebenschlussgenerator)
- Verbundgenerator

Neben den theoretischen Abhandlungen für jede Teilübung wird hier aber vor allem die Übungsdurchführung im Applet gezeigt.

Hier möchten wir den Ablauf der Übungsvorbereitung im Applet zeigen. Es werden einzelne Punkte der Übung dargestellt und erklärt. Zusätzlich wird für jede Teilübung auch die theoretische Grundlage gegeben.

Applet verfolgt den normalen Übungsverlauf in Labor. Angefangen wird mit dem Eingangstest, dann kommt die Einstellung der Bürsten in die neutrale Zone, dann Messungen am fremderregten Generator, Nebenschlussgenerator und abschließen am Verbundgenerator. Am Ende wird im Abschluss E-Mail mit den Ergebnissen der durchgeführten Messungen dem Betreuer gesendet.

2.1 Allgemeine Bedienungsgrundlagen

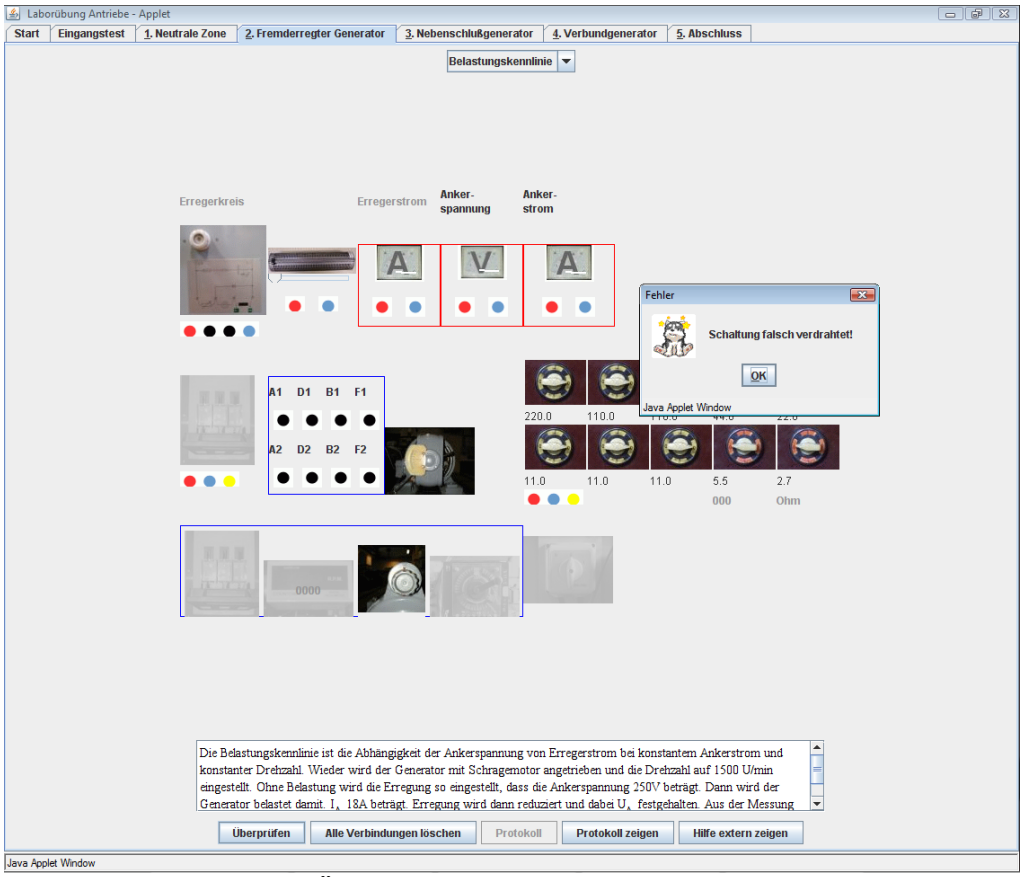


Bild 2.1: Beispiel einer Übungsmaske

Die Verbindung wird erstellt indem man auf die Klemmen mit der linken Maustaste klickt. Zuerst wird die Klemme vom Anfangs- und dann vom Endelement geklickt. Die Klemme selbst ändert ihr Aussehen von  zu  während sie aktiviert ist. Mit dem links Klick auf die Endklemme wird die Verbindung erstellt und es erscheint der Linie die zwei Klemmen verbindet .

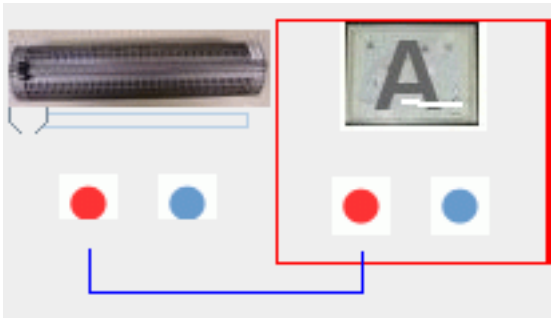


Bild 2.2:Verbindung

Ob die Schaltung richtig verbunden ist überprüft man indem man auf die Schaltfläche „Überprüfen“ klickt.

Falls die Schaltung nicht richtig verbunden ist gibt es zwei Möglichkeiten die Verbindungen zu löschen. Eine Möglichkeit ist wie bei der Erstellung der Verbindung die Klemmen einzeln zu klicken nur mit der rechten Maustaste. Die zweite Möglichkeit ist auf die Schaltfläche „Alle Verbindungen löschen“ zu klicken womit alle Verbindungen in der Maske gelöscht werden.

Beim Klick auf die Schaltfläche „Protokoll“ werden die aktuellen Werte von den Elementen die einen Wert haben gespeichert. Die Schaltfläche wird erst dann aktiviert wenn die Schaltung richtig verbunden ist. Mit dem Klick auf die Schaltfläche „Protokoll zeigen“ lässt sich das Protokoll in einem neuen Fenster anzeigen.

Mit dem Klick auf die Schaltfläche „Hilfe extern zeigen“ wird in einem neuen Fenster die Hilfe für die Übung angezeigt.

2.2 Start

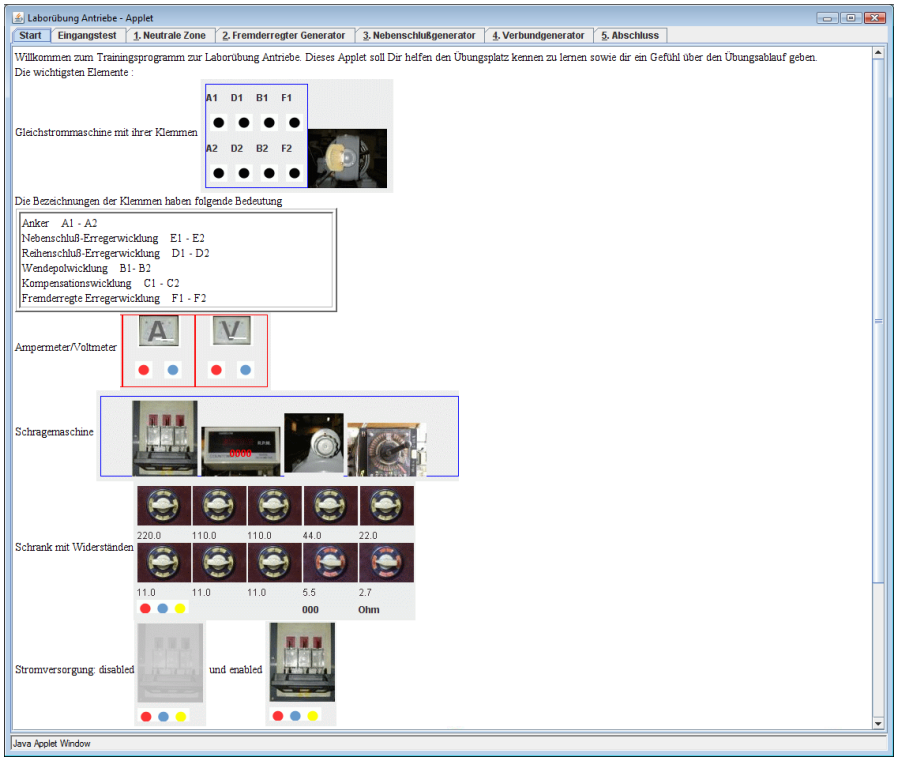


Bild 2.3: Appletstartfenster

Nach dem Aufruf der Internetadresse wo Applet abgelegt ist erscheint zuerst eine Hilfe (html) wo die wichtigsten Elemente der Übung angezeigt werden sowie die Beschreibung wie eine Verbindung zu machen ist.

2.3 Eingangstest

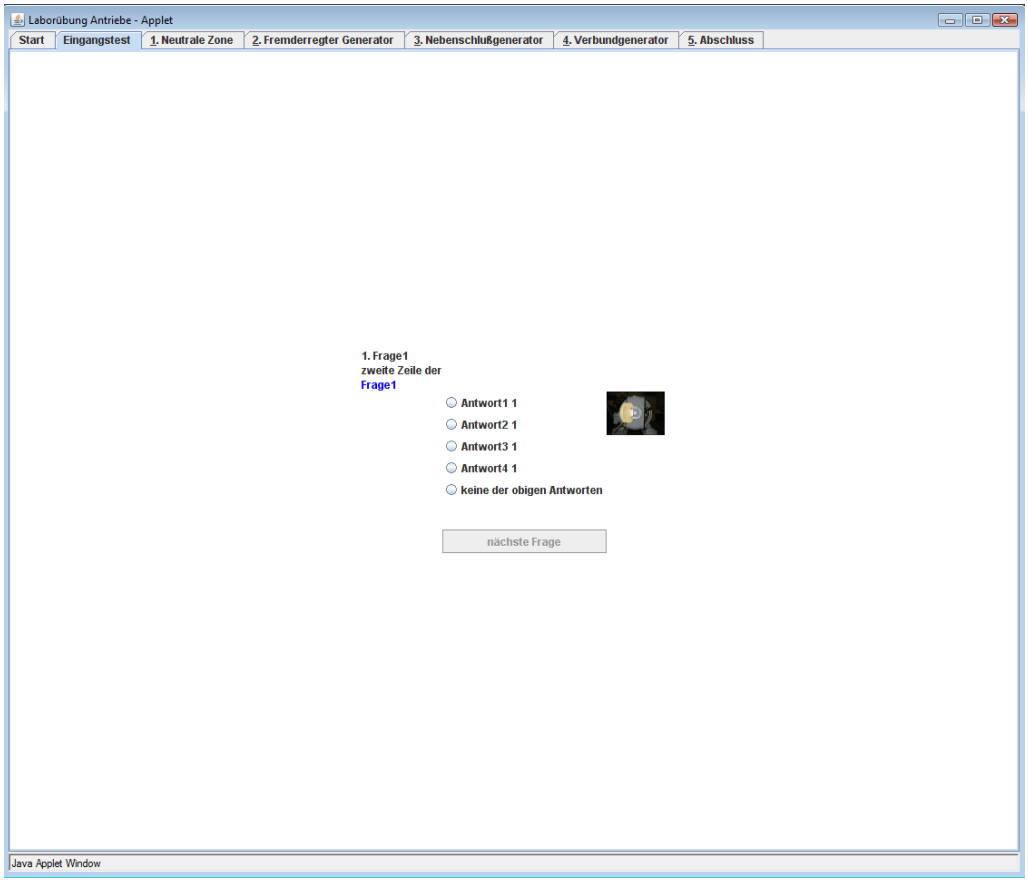


Bild 2.4: Eingangstestfenster

Bei dem Eingangstest werden die Fragen für die theoretische Vorbereitung zur Laborübung angezeigt. Links oben ist die Frage. Darunter befinden sich fünf Antwortmöglichkeiten und ganz rechts ist der Platz für ein eventuell vorhandenes Bild zur Frage. Ganz unten befindet sich die Schaltfläche „nächste Frage“. Bei dem Klick auf die Schaltfläche wird die ausgewählte Antwort in das Protokoll aufgenommen und es wird zur nächsten Frage übergegangen. Es ist nicht möglich zu der Frage zurückzukehren ohne den Applet neu zu starten.

2.4 Einstellung der Bürsten in die neutrale Zone

Bei dieser Messung ist das Ziel die neutrale Zone der Maschine zu bestimmen.
Die neutrale Zone ist ein Bereich wo es zu keiner magnetischen Kopplung zwischen Stator und Rotor kommt, bzw. ist diese zu schwach um ein genügend großes Drehmoment zu erzeugen damit die Drehung des Rotors möglich ist. Um die neutrale Zone zu bestimmen gibt es zwei Wege die in der Übung verfolgt werden: Momentenmethode und ballistische Methode.

2.4.1 Momentenmethode

2.4.1.1 Applet Ausführung

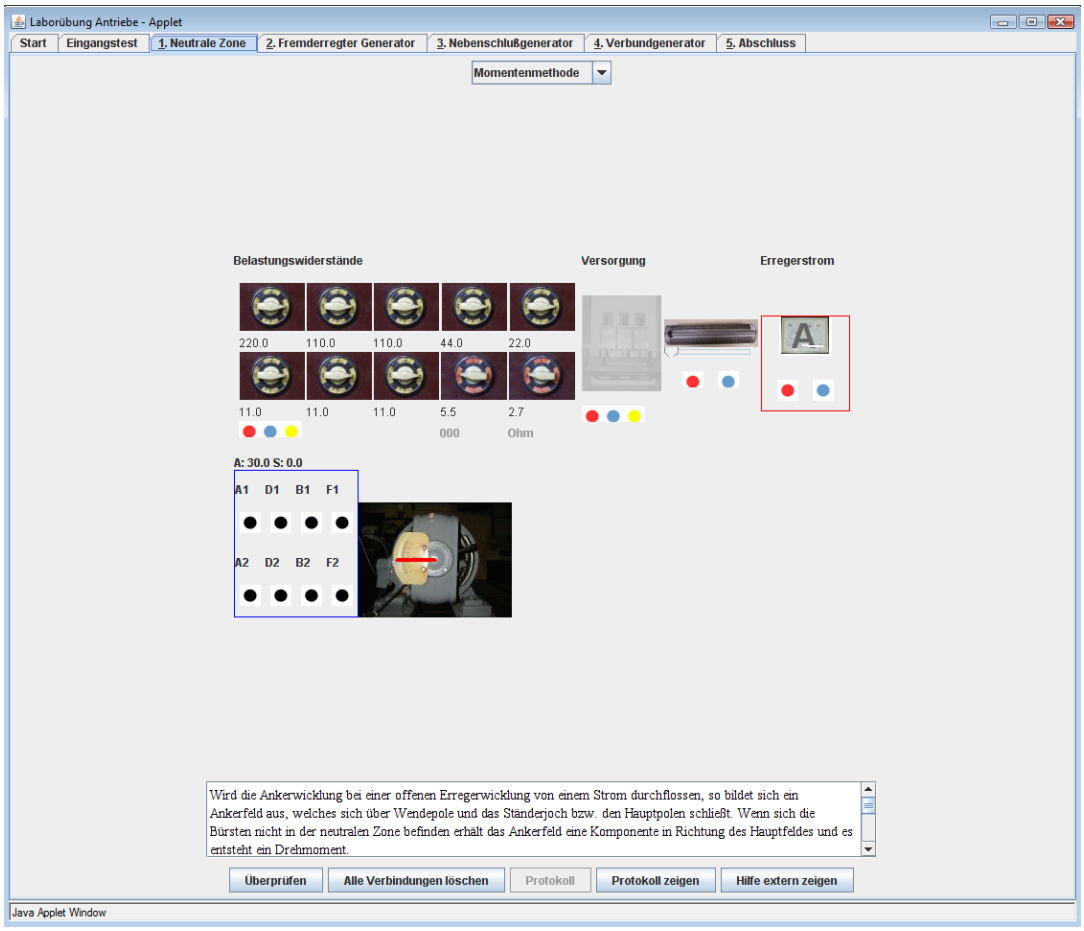


Bild 2.5: Fenster zu „Neutrale Zone-Momentenmethode“

In der Maske für die Momenten Methode werden der Widerstandschrank, die Maschine, ein Vorwiderstand, die Versorgung und ein Amperemeter angezeigt. Wie bei allen anderen Masken im Applet sind die Elemente deaktiviert und werden erst dann aktiv wenn die Schaltung richtig verbunden ist.

Die Maschine wird mit einem Hebel angezeigt. Dieser Hebel lässt sich mit der Maus bewegen. Klickt man auf die Maschine mit der linken Maustaste und bewegt die Maus mit der Maustaste gedrückt so bewegt sich der Hebel mit der Maus. Andere Möglichkeit den Hebel in kleineren Schritten zu bewegen ist mit der linken Maustaste oberhalb bzw. unterhalb des Hebels zu klicken.
Mit dem Klick auf die Versorgung wird die Schaltung mit dem Strom versorgt.

2.4.1.2 Theorie

Wird die Ankerwicklung bei einer offenen Erregerwicklung von einem Strom durchflossen, so bildet sich ein Ankerfeld aus, welches sich über Wendepole und das Ständerjoch bzw. den Hauptpolen schließt. Wenn sich die Bürsten nicht in der neutralen Zone befinden erhält das Ankerfeld eine Komponente in Richtung des Hauptfeldes und es entsteht ein Drehmoment.

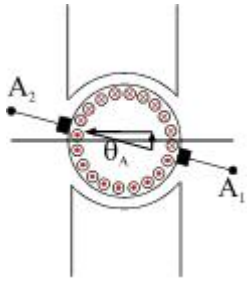


Bild 2.6: Zum Bestimmen der neutralen Zone

Damit eine Drehung des Rotors möglich ist, muss zuerst das Reibungsmoment überwunden werden. Deshalb werden Versuche für beide Drehrichtungen der Maschine gemacht. Die neutrale Zone wird dann durch Mittelung der beiden Ergebnisse ermittelt.
Die ganze Messung ist einmal mit und einmal ohne Kompensationswicklungen durchzuführen.
Bei der Messung wird der Nennstrom der Maschine (18.3A) über Vorwiderstand eingestellt. Danach werden die Bürsten gedreht bis sich der Rotor zu drehen beginnt. Dieser Punkt ist für beide Drehrichtungen festzuhalten.

2.4.2 Ballistische Methode

2.4.2.1 Applet Ausführung

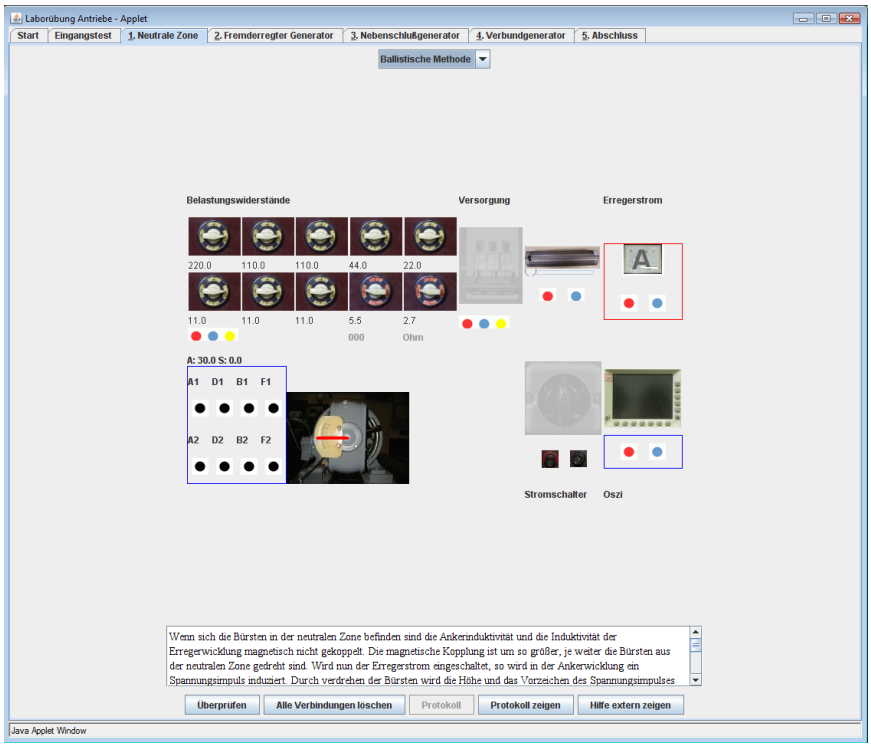


Bild 2.7: Fenster zu „Neutralen Zone-Ballistische Methode“

Bei der ballistischen Methode wird, nach dem die Schaltung richtig verbunden ist, durch den Klick auf den Stromschalter ein Impuls erzeugt. Abhängig davon wie die Bürsten eingestellt sind (Einstellung erfolgt wie bei der Momentenmethode) wird ein Aufschlag am Oszilloskop angezeigt.

2.4.2.2 Theorie

Wenn sich die Bürsten in der neutralen Zone befinden sind die Ankerinduktivität und die Induktivität der Erregerwicklung magnetisch nicht gekoppelt, da die Richtungen von Ankerfluss, und Hauptfluss, normal aufeinander sind. Die magnetische Kopplung ist umso größer, je weiter die Bürsten aus der neutralen Zone gedreht sind. Wird nun der Erregerstrom eingeschaltet, so wird in der Ankerwicklung ein Spannungsimpuls induziert. Durch verdrehen der Bürsten wird die Höhe und das Vorzeichen des Spannungsimpulses verändert. Wenn der Spannungsimpuls minimal wird, ist die neutrale Zone erreicht.

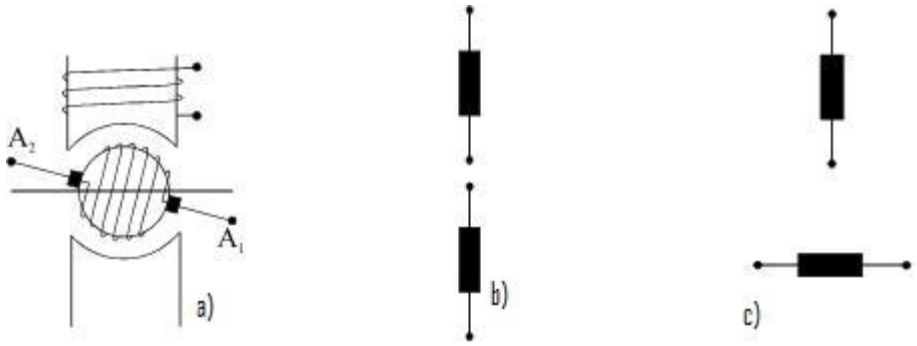


Bild 2.8: Zum Bestimmen der neutralen Zone
a) Kopplung zwischen Anker- und Erregerwicklung
b) maximale Kopplung, c) minimale Kopplung

Um diesen Impuls zu messen wird ein Oszilloskop an die Maschine angeschlossen. Der Erregerstrom wird über einen Schalter am Erregerkreis eingespeist. Der Schalter wird nur kurzzeitig ein- und dann wieder ausgeschaltet. Dabei ist am Oszilloskop der induzierte Spannungsimpuls des Ankerkreises sichtbar. Die Bürsten werden so lange gedreht bis der Impuls minimal wird. Die Messung ist wieder in beiden Drehrichtungen durchzuführen.

2.5 Fremderregter Generator

Beim fremderregten Gleichstromgenerator wird der Erregerstrom von einer zweiten Gleichspannungsquelle geliefert und ist daher unabhängig von Laststrom oder Drehzahl einstellbar. Dabei werden in dieser Teilübung die Leerlaufkennlinie, Belastungskennlinie, äußere Kennlinie und Regulierkennlinie gemessen.
Bei der Übung wird die Drehzahl des Generators durch eine zweite Maschine (Schragemotor) reguliert.

2.5.1 Leerlaufkennlinie

2.5.1.1 Applet Ausführung

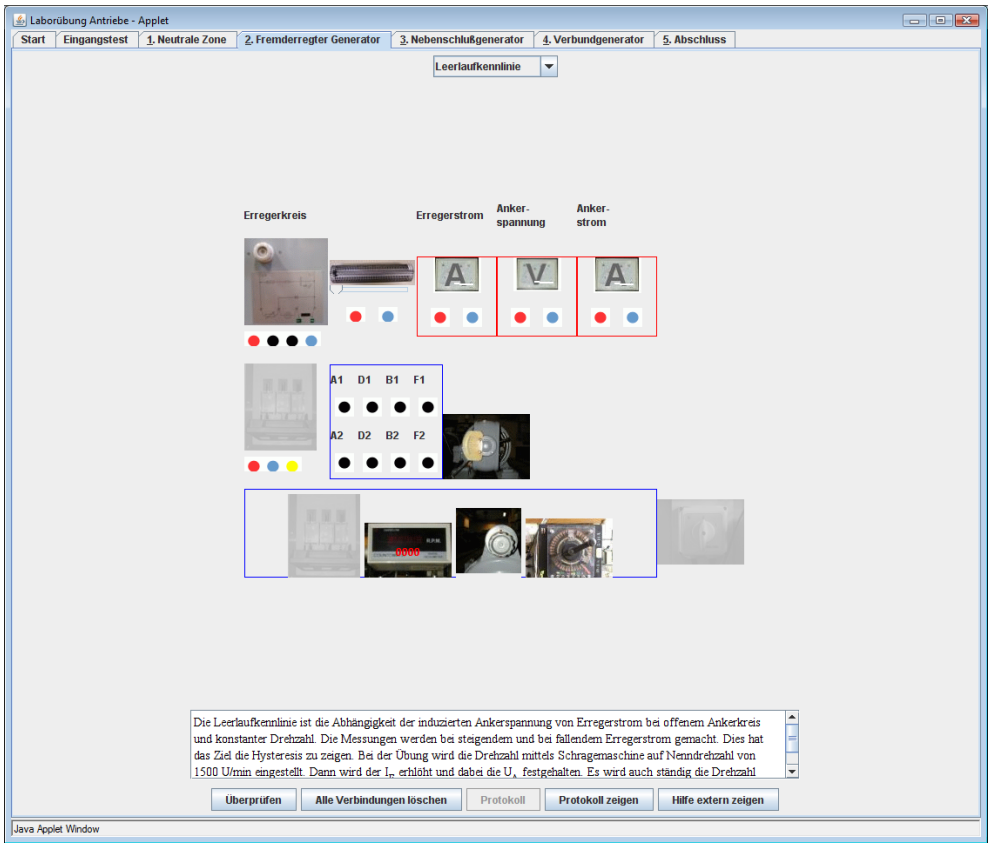


Bild 2.9: Fenster zu „Fremderregter Generator-Leerlauf Kennlinie“

In der Maske für die Leerlaufkennlinie ist die Gleichstrommaschine ohne Hebel angezeigt, sonst ist die Maschine gleich wie bei deren Ermittlung der neutralen Zone. In der Maske ist als neues Element der Schrage Motor angezeigt. Bei dem Schrage Motor sind keine Klemmen, da er schon fest verbunden ist. Dafür sind aber folgende Elemente angezeigt: der Schalter zur Stromversorgung, der Einschalteschalter, der Rundschalter, der Umdrehungszähler und letztlich der Motor selbst.

Bild 2.10: Schrage Machine

Nachdem die Schaltung richtig verbunden ist lassen sich die Stromschalter betätigen (klicken). Erst wenn alle drei richtig betätigt sind zeigt der Zähler die Nenndrehzahl von 1500 Umdrehungen je Minute.

2.5.1.2 Theorie

Die Leerlaufkennlinie ist die Abhängigkeit der induzierten Ankerspannung von Erregerstrom bei offenem Ankerkreis und konstanter Drehzahl. $U_A = U_i = f(I_E), I_A = 0, n = konst.$
Die Messungen werden bei steigendem und bei fallendem Erregerstrom gemacht. Dies hat das Ziel die Hysteresis zu zeigen. Bei der Übung wird die Drehzahl mittels Schragemaschine auf Nenndrehzahl von 1500 U/min eingestellt. Dann wird der I_E erhöht und dabei die U_A festgehalten. Die Drehzahl wird auch ständig nachreguliert.

2.5.2 Belastungskennlinie

Die Belastungskennlinie ist die Abhängigkeit der Ankerspannung von Erregerstrom bei konstantem Ankerstrom und konstanter Drehzahl. $U_A = f(I_E), I_A = konst., n = konst.$
Wieder wird der Generator mit Schragemotor angetrieben und die Drehzahl auf 1500 U/min eingestellt. Ohne Belastung wird die Erregung so eingestellt, dass die Ankerspannung 250V beträgt. Dann wird der Generator belastet damit sich $I_A = 18\text{A}$ ergibt. Die Erregung wird dann reduziert und dabei U_A festgehalten. Aus der Messung wird dann die innere Kennlinie $U_i = f(I_E)$ nach der Formel $U_A = U_i - R_A \cdot I_A$ berechnet.

2.5.2.1 *Applet Ausführung*

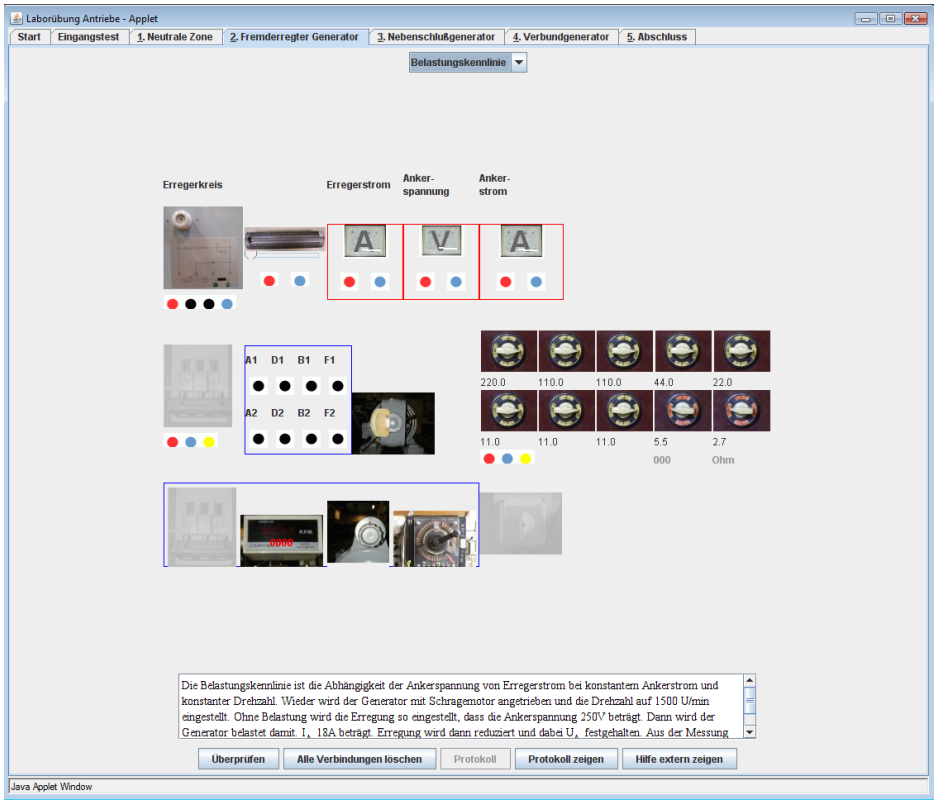


Bild 2.11: Fenster zu „Fremderregter Generator-Belastungskennlinie“

2.5.3 Äußere Kennlinie

Die äußere Kennlinie ist die Abhängigkeit der Ankerspannung vom Ankerstrom bei konstantem Erregerstrom. $U_A = f(I_A), I_E = konst., n = konst.$

Der Generator wird angetrieben und n auf Nenndrehzahl von 1500 U/min eingestellt. Ohne Belastung wird die Ankerspannung auf 250V eingestellt. Dann wird die Maschine auf $I_A = 18\text{A}$ belastet. Die Belastung wird danach reduziert und die Kennlinie aufgenommen.

2.5.4 Regulierkennlinie

Die Regulierkennlinie ist die Abhängigkeit des Erregerstroms von Ankerstrom bei konstanter Ankerspannung und konstanter Drehzahl. $I_E = f(I_A), U_A = konst., n = konst.$

Der Generator wird angetrieben und n auf 1500 U/min eingestellt. Ohne Belastung wird die Ankerspannung auf 250V eingestellt. Dann wird die Maschine belastet damit sich $I_A = 18\text{A}$ ergibt. Die Belastung wird reduziert. Gleichzeitig wird I_E nachgestellt damit U_A konstant bleibt. Die Messung von I_E ergibt dann die Kennlinie.

2.6 Nebenschlussgenerator

Beim Nebenschlussgenerator wird die Erregung nicht von einer äußeren Quelle genommen sonder wird vom Generator selbst induzierte Spannung verwendet. In der Übung wird nur die äußere Kennlinie aufgenommen.

2.6.1 Äußere Kennlinie

$U_A = f(I_A), I_E = konst., n = konst.$

I_E wird über den Vorwiderstand R_V so eingestellt, dass bei $I_A = 0$ die Ankerspannung U_A 250V beträgt. Die Drehzahl wird auf 1500 U/min eingestellt.

2.6.1.1 Applet Ausführung

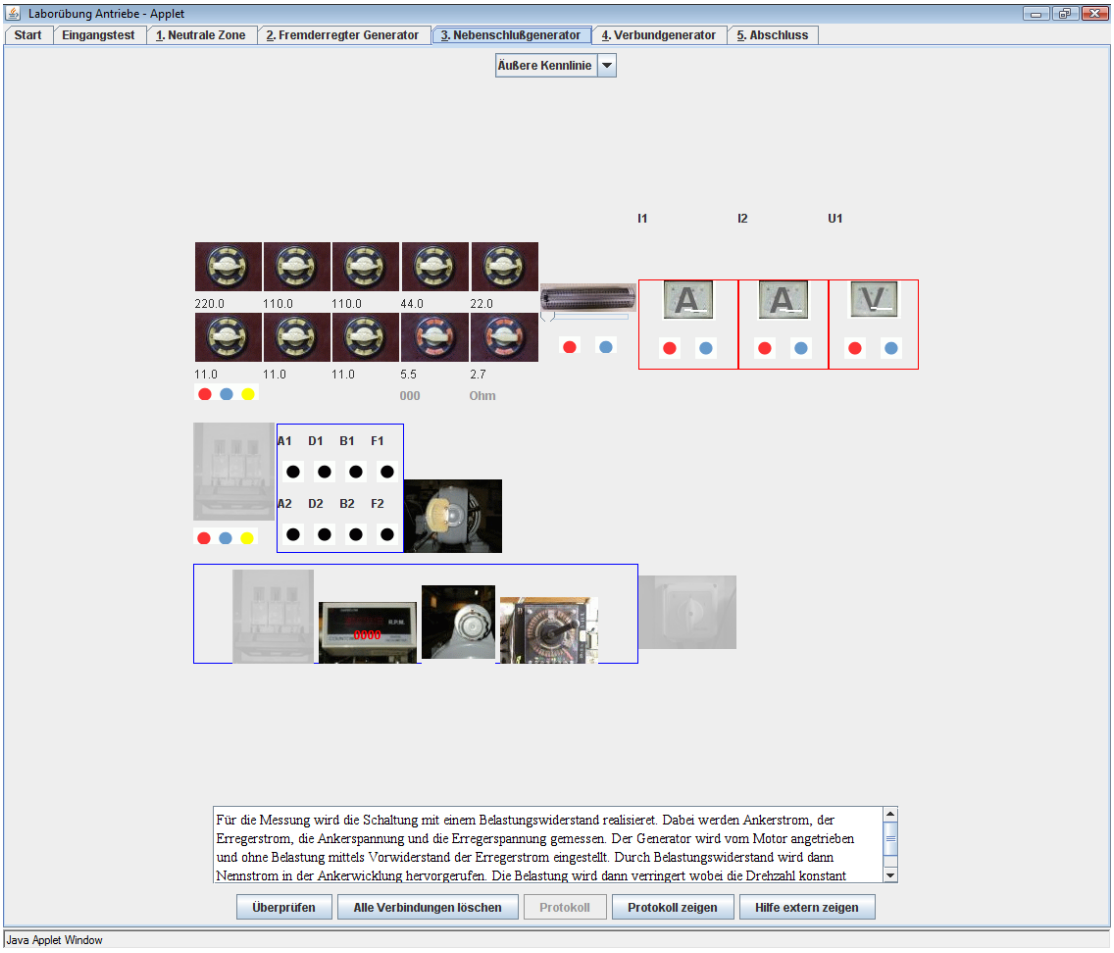


Bild 2.12: Fenster zu „Nebenschlußgenerator“

2.7 Verbundgenerator

Beim Gleichstrom – Verbundgenerator, was ja eigentlich ein Nebenschlussgenerator mit Kompoundwicklung ist, wird eine Reihenschlusswicklung zusätzlich zu einer Nebenschlusswicklung in Serie geschaltet. Diese hat einfach den Zweck das Hauptfeld zu verstärken, gesteuert durch den Ankerstrom I_A, und dadurch die Ankerrückwirkung zu minimieren.
Die Maske beim Verbundgenerator ist gleich wie beim Nebenschlussgenerator.

2.8 Abschluss

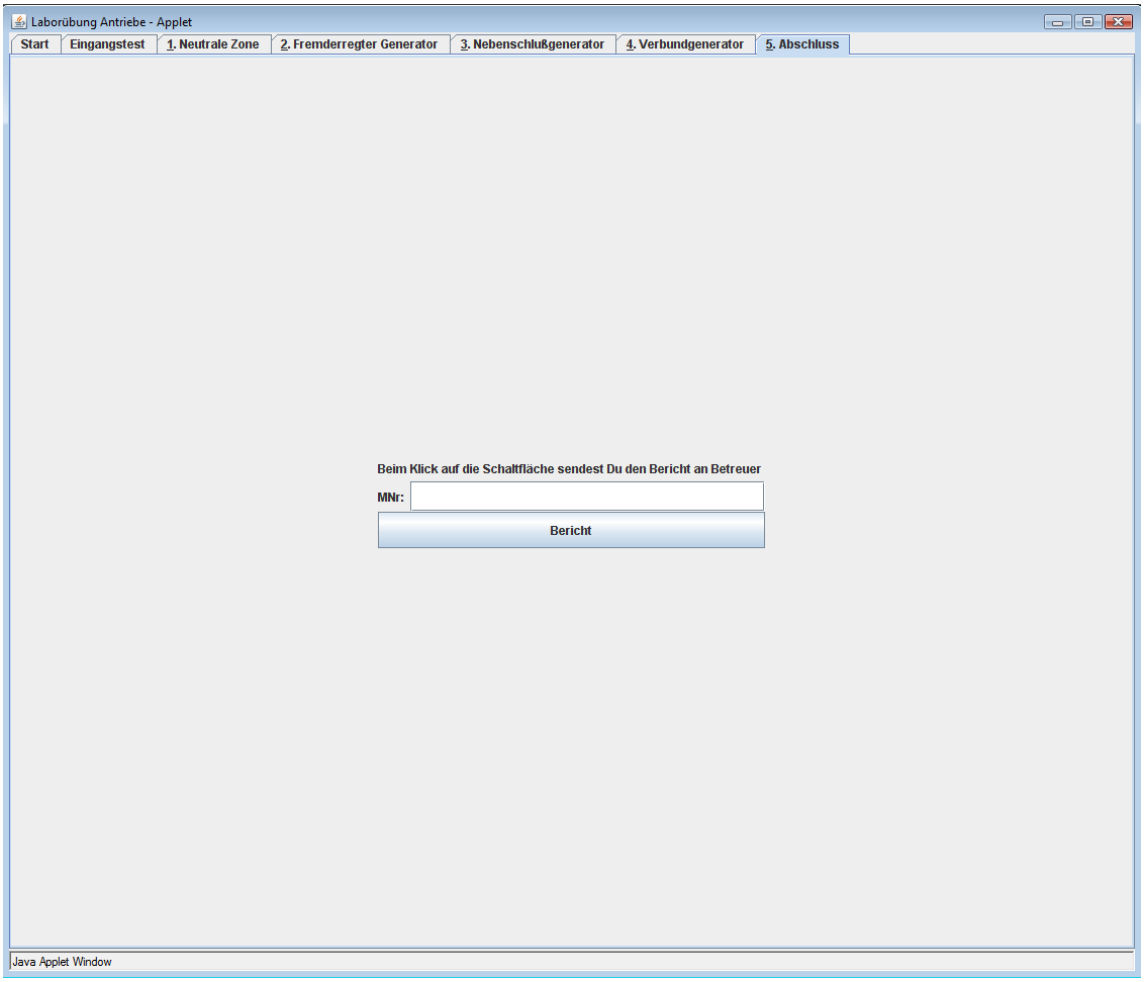


Bild 2.13: Abschlußfenster

Nachdem alle Punkte der Übung durchgemacht sind, hat der Student die Möglichkeit die Ergebnisse seiner Messungen via E-Mail an den Betreuer zu senden. In dem E-Mail sind die Antworten des Eingangstests sowie die Werte der Elemente in einzelnen Teilübungen enthalten.

3 Programmierung

Notation

Für die Notation der Variablen wird die so genannte „Ungarische Notation“ verwendet. Obwohl sie in Java nicht so wie in C++ verbreitet ist. Die Ungarische Notation weist jedem Variabelnamen ein oder zwei Buchstaben zu die den Typ der Variable anzeigen. So wird einem Double ein 'd' vorangestellt (z.B. double dEineVariable) einem Integer ein 'n', einem String ein 's' usw.... Die Ausnahme von dieser Regel (keine Regel ohne Ausnahme) sind die Objekte der Klassen.
Im ganzen Dokument wird für Teile von Sourcecode folgende Nomenklatur verwendet

```
public class GSMLaborMain extends JApplet {
```

. . . Bedeutet eine Unterbrechung im Sourcecode.

3.1 Applet Struktur

Damit das Applet in der gewünschten Flexibilität funktioniert ist eine gewisse Dateienstruktur notwendig.

- root-Verzeichnis
  beinhaltet:
  - die .class Datei der Hauptklassse mit dem Einstiegspunkt des Applets.
  - die .jar Datei
  - fragen.txt – Fragen für Eingangstest
  - index.html
  - welcome.html – kurze Einleitung

Außerdem werden folgende Unterverzeichnisse angelegt

- CL
  beinhaltet *.cl Dateien in denen die Zeichenfolgen gespeichert sind die zu realisierenden Verbindungen der Schaltung darstellen.
- Help
  beinhaltet *.html-Dateien des Hilfesystems.
- qS
  beinhaltet die Bilddateien die für Eingangstest benötigt werden.
- pics
  beinhaltet die Bilder aller Elemente des Applets

3.2 Klassenbibliothek

Die Klassen sind in vier Bibliotheken, in Java *package* gennant, aufgeteilt: „default package“, „messPlatz“, „panels“ und „ueTools“.

3.2.1 default package

Im default package sind die Klassen enthalten die die Panels für einzelne Teilübungen erstellen. Hier ist auch die zentrale Klasse des Applets GSMLaborMain die alle Panels des Applets lädt.

3.2.1.1 GSMLaborMain

GSMLaborMain ist die Hauptklasse des Applets. In ihr werden die Größe und das Aussehen des Applets definiert. Für das Layout des Applets wurde TabbedLayout gewählt. Dadurch wurde einiges an Platz gespart was nicht unwichtig ist da am Anfang die Entscheidung getroffen wurde die Größe des Applets auf 800x600 Pixel zu beschränken um sicher zu gehen, dass es auf (fast) allen aktuellen PCs gut darstellbar ist. Später, im Laufe der Entwicklung, ist stattdessen das Applet auf den ganzen Bildschirm erweitert worden.

```
public class GSMLaborMain extends JApplet {
...
      private JFrame f;
...
      static List<String> protocolString;
      static List<String> protocolStringET;

      public void init() {
            protocolString  = new ArrayList<String>();
            protocolStringET  = new ArrayList<String>();

            f = new JFrame("Laborübung Antriebe - Applet");
            JTabbedPane tabbedPane = new JTabbedPane();
...
            JComponent panelNZ = new NZPanel(this);
            tabbedPane.addTab("1. Neutrale Zone", panelNZ);
            tabbedPane.setMnemonicAt(2, KeyEvent.VK_1);
...
            add(tabbedPane);
            setVisible(true);
   }
}
```

Die Klasse GSMLaborMain ist von der standard Java-Klasse JApplet abgeleitet. Beim Aufruf der Klasse, was gleich bedeutend mit dem Aufruf des Applets ist, wird die Funktion `init()` ausgeführt. Die Funktion kreiert ein Objekt der Klasse `JTabbedPane`

```
JTabbedPane tabbedPane = new JTabbedPane();
```

das am Ende dem Panel zugefügt und dadurch GUI erstellt wird.
Dem `tabbedPane` werden die Objekte der Klassen, die einzelne Teilübungen darstellen, zugefügt.

```
JComponent panelNZ = new NZPanel(this);
tabbedPane.addTab("1. Neutrale Zone", panelNZ);
```

Das Applet wird im eigenen Fenster ausgeführt. Dieses Fenster erstreckt sich über den ganzen Bildschirm.

```
f = new JFrame("Laborübung Antriebe - Applet");
```

Das Dimensionieren vom Frame erfolgt in der Funktion `start()`.

```
public synchronized void start() {
      Dimension ss = Toolkit.getDefaultToolkit().getScreenSize();
      f.setBounds(0, 0, ss.width, ss.height);

      f.setVisible(true);
}
```

Die Klasse definiert auch zwei Strings die zum Protokollieren der Übung und des Eingangstests dienen.

```
static List<String> protocolString;
static List<String> protocolStringET;
```

protocolStringET steht für die Ergebnisse des Eingangstests und *protocolString* für die Resultate der Übungen.

Diese zwei Listen werden in den Funktionen `addProt()` und `addProtET()` gefüllt. Der Inhalt der Listen wird mit der Funktion `getProt()` abgerufen. Der Parameter `bET` bestimmt ob auch der Protokollstring für den Eingangstest zurückgegeben wird.

```
public void addProt(String s) {
public void addProtET(String s) {
public  List<String> getProt(boolean bET)
```

3.2.1.2 UEPanel

UEPanel ist die Basisklasse von der alle Klassen die Teilübungen repräsentieren abgeleitet werden. Sie ist von JPanel abgeleitet und implementiert `ActionListener` Interface. Dieses Interface ermöglicht die Reaktion beim Klicken auf Schaltflächen. Außerdem implementiert UEPanel `ItemListener` Interface zur Auswahl der Messungen bei Teilübungen.

```
abstract public class UEPanel extends JPanel implements ItemListener,
ActionListener, HyperlinkListener {
...
      JPanel cards = new JPanel();
      JPanel commandPanel = new JPanel();
      JPanel cobPanel = new JPanel();

      JButton pbCheck, pbStart, pbExplorerEx;

      JComboBox cob;

      UEMessage p;

      public UEPanel(JApplet parent) {
            this.parent = parent;
            setLayout(new BorderLayout());

            p = new UEMessage(this);

            createPanel(this);
      }
...
}
```

UEPanel ist eine abstrakte Klasse was bedeutet, dass keine Objekte dieser Klasse kreiert werden können. Sie dient nur als Basisklasse von der andere Klassen abgeleitet werden und zu diesem Zweck deklariert die „member“ Funktionen, GUI-Elemente und Panels für die abgeleiteten Klassen.
`JPanel cards` ist der Kontainer der mit GUI-Elementen befüllt wird.
In UEPanel sind abstrakte Funktionen deklariert die dann in abgeleiteten Klassen definiert werden.

- `abstract public void createCobPanel();`
  erstellt die `ComboBox` mit der die Messung ausgewählt wird. Jede abgeleitete Klasse befüllt die `ComboBox` mit entsprechenden Strings.
- `abstract public void createCards();`
  erstellt die Panels für die Übung
- `abstract protected void externHelp();`
  öffnet die entsprechende HTML Hilfe-Datei im externen Fenster
- `abstract protected boolean checkConnections();`
  überprüft ob alles richtig verbunden ist

Die Funktion `checkConnections()` in abgeleiteten Klassen ruft die overloaded UEPanel Funktion `protected boolean checkConnections(ParentPanel pnl, String s)` auf. Diese Funktion ruft ihrerseits die Funktion `getConnectionList()` auf, die .cl-Datei mit dem namen 's' liest und ihr Inhalt als eine Liste zurück gibt. Dann wird der Inhalt dieser Liste mit dem Inhalt der Liste, die die ParentPanel-Memberfunktion `getConnsS()`, bzw. `getRevConnsS()` zurückgibt, verglichen.
Die Funktion `itemStateChanged(ItemEvent evt)` wechselt die Messung was durch die Auswahl eines Punktes im ComboBox erfolgt.

Die nächste wichtige Funktion ist `public void createCommandPanel()`. Diese Funktion wird von allen abgeleiteten Klassen aufgerufen. Sie erstellt die Schaltflächen zur Steuerung vom Übungsablauf.

3.2.1.3 ETPanel

ETPanel ist die Klasse die den Eingangstest simuliert.

```
public class ETPanel extends JPanel implements ActionListener {
      JApplet parent;

      List<String>qAa = new ArrayList<String>();
      int i = 0;
      int n = 0;
      int nq = 0;

      JButton pbNext = new JButton("nächste Frage");
...
}
```

Die Liste qAa wird von der Datei „fragen.txt“ gefüllt. Die Struktur von „fragen.txt“ ist in Abschnitt 4.1 erklärt.

Das Lesen der Datei „fragen.txt“ und Ausfüllen der Liste qAa erfolgt im Konstrukt der Kasse im try/catch Block Die Datei wird mit der Klasse DataInputStream aus dem java.io Package gelesen. Damit sie die Datei lesen kann muß, zuerst eine URL Verbindung zur Datei hergestellt werden. Das erledigen die Klassen URL und URLConnection von java.net Package. Wird die Datei erfolgreich geöffnet, wird sie von DataInputStream zeilenweise gelesen und die gelesenen Strings in die Liste qAa geladen.

```
public ETPanel(JApplet parent) {
...
      try {
            URL url;
            URLConnection urlC;
            DataInputStream dis;

            url = new URL(parent.getCodeBase().toString() + "fragen.txt");
            urlC = url.openConnection();
            urlC.setDoInput(true);
            urlC.setUseCaches(false);

            dis = new DataInputStream(urlC.getInputStream());

            String s;
            while ((s = dis.readLine()) != null) {
                  qAa.add(s);
            }
            dis.close();
            nq = qAa.size()/8;
      }
      catch(IOException e) {
            System.err.println(e.toString());
      }

      createPanel(this);
}
```

Durch das Klicken der Schaltfläche `pbNext` wird die ausgewählte Antwort übernommen. Das passiert in der Funktion `actionPerformed(ActionEvent e)`. Zuerst wird mittels der Funktion `getSelection()` ermittelt welche Antwort ausgewählt wurde. Dann wird diese Antwort in das Protokollstring geschrieben und anschließend wird überprüft ob der Benutzer zum Ende des Tests gekommen ist.

```
public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
      if(src == pbNext) {
            JRadioButton bA = getSelection(ga);
            String sa = bA.getName();
            String pS = sa + "\n" ; //+ "====" + "\n";
            ((GSMLaborMain)parent).addProtET(pS);
            i++;
            if(i>nq-1) {
                  i=nq-1;
                  UEMessage msg = new UEMessage(this);
                  msg.setTitle("");
                  msg.setMessage("Eingangstest abgeschlossen");
                  msg.createAndShow();
            }
            recreatePanel();
      }
}
...
```

Beim Klicken auf eines der Radiobuttons wird anhand der qAa Liste überprüft ob die richtige Antwort gegeben wurde. Daraufhin werden die Begründung der richtigen Antwort sowie die Meldung, ob die Antwort falsch oder richtig war, im entsprechenden Label eingegeben. Durch das Klicken vom Radiobutton wird auch die Schaltfläche pbNext aktiviert.
Die Meldung ob die Antwort richtig oder falsch ist wird deswegen ausgegeben, weil ursprünglich vorgesehen war, dass der Test nur zum Üben dienen soll und das Protokollieren der Antworten nicht vorgesehen war. Erst später wurde in der Entwicklung von Assistenten der Wunsch geäußert den Test auch zu protokollieren und letztendlich als E-Mail an Betreuer zu senden. Da zum Zeitpunkt des Schreibens dieser Arbeit nicht geklärt wurde ob das Senden von E-Mail möglich ist, ist die Ausgabe der Meldung ob die Antwort richtig oder falsch ist, geblieben. Die Problematik des Sendens von E-Mail wird in Abschnitt 3.2.4.1erklärt.

```
      if(src == a1 || src == a2 || src == a3 || src == a4 || src == a5) {
            JRadioButton bA = (JRadioButton)src;
            String sa = bA.getName();

            if (0 == qAa.get(n+5).compareTo(sa)) {
                  a.setForeground(Color.black);
                  a.setText("Richtig");
                  aB.setText(qAa.get(n+6));
            }
            else {
                  a.setForeground(Color.red);
                  a.setText("leider falsch!");
                  aB.setText(qAa.get(n+6));
            }
            pbNext.setEnabled(true);

            repaint();
      }
      repaint();
}
```

3.2.1.4 FEGPanel

Fremderregter Generator wird durch diese Klasse dargestellt. Die Erklärung der Funktionen gilt auch für die andere von UEPanel abgeleiteten Klassen.

```
public class FEGPanel extends UEPanel {
...
final static String LLKPANE = "Leerlaufkennlinie";
final static String BKPANE = "Belastungskennlinie";
final static String AKPANE = "Äußere Kennlinie";
final static String RKPANE = "Regulierkennlinie";

FEGLLKPanel llkPanel;
UEHelpPanel helpPanelLLK;
...
}
```

Hiermit werden die Strings definiert die dann zur Bestimmung des anzuzeigenden GUI und des Hilfepanels sowie bei `checkConnections()` benutzt werden.

```
public void createCobPanel() {
      String cobItems [] = {LLKPANE, BKPANE, AKPANE, RKPANE};
      cob = new JComboBox(cobItems);
      cob.setEditable(false);
      cob.addItemListener(this);
      cobPanel.add(cob);
}
```

Die Funktion `createCobPanel()` erstellt ComboBox zur Auswahl der Messung. Die Funktion `addItemListener()` ermöglicht der Checkbox auf Mausklicks zu reagieren.

```
public void createCards() {
...
      // Leerlaufkennlinie
      JPanel cardLLK = new JPanel();
      cardLLK.setLayout(new BorderLayout());
      llkPanel = new FEGLLKPanel(this.parent);
      helpPanelLLK = new UEHelpPanel();
      createHelpPanel(helpPanelLLK, LLKPANE + "_B");

      cardLLK.add(llkPanel, BorderLayout.CENTER);
      cardLLK.add(helpPanelLLK, BorderLayout.PAGE_END);
...
      cards.setLayout(new CardLayout());
      cards.add(cardLLK, LLKPANE);
      cards.add(cardBK, BKPANE);
      cards.add(cardAK, AKPANE);
      cards.add(cardRK, RKPANE);
}
```

Die Funktion `createCards()` ist für die Erstellung der entsprechenden GUIs sowie der richtigen Hilfepanels zuständig. Die GUI wird eigentlich in der Klasse FEGLLKPanel (Panel für Leerlaufkennlinie das in Package panels definiert wird. Entsprechend werden die GUIs für die andere Übungsteile definiert) erstellt.

Für das Anzeigen von verschiedenen Teilen der Übung wird CardLayout verwendet. Durch dieses Standardlayout von Java werden verschiedene Panels, die Teile der Übung darstellen, mit Strings verbunden. So ein Panel setzt sich aus dem Panel der den Messplatz repräsentiert und dem Panel der die Hilfe anzeigt zusammen.

Das Hilfepanel wird durch die Funktion `createHelpPanel()` erstellt.

Eine wichtige Funktion ist auch `externHelp()`. In dieser Funktion wird die gleichnamige Funktion der Basisklasse mit dem entsprechenden String als Parameter aufgerufen. Der String wird aufgrund der Auswahl von Combobox konstruiert.

```
protected void externHelp() {
      int nP = cob.getSelectedIndex();
      switch (nP) {
            case 0: // Leerlaufkennlinie
            {
            String s = "FEG"+LLKPANE+".html";
                  super.externHelp(s);
            }
            break;
…
      }
}
```

Die Funktion `saveValues()` muss überschrieben werden. Diese Funktion erstellt den richtigen Protokollstring für diese Teilübung.

```
protected void saveValues() {
      int nP = cob.getSelectedIndex();
      switch (nP) {
            case 0:
                  super.saveValues(llkPanel, "Fremderregter Generator === "
+ LLKPANE);
                  break;
      ...
      }
}
```

Genau wie bei den vorigen zwei Funktionen werden auch bei den Funktionen `createHelpPanel()`, `checkConnections()` und `delConnections() U` durch die Abfrage der Auswahl in der Combobox der richtige String konstruiert und die entsprechende Funktion von UEPanel ausgeführt.

3.2.1.5 WelcomePanel

Diese Klasse dient zur Anzeige von Hilfedateien wenn auf die Schaufläche „Hilfe extern zeigen“ geklickt wurde. Diese Klasse zeigt auch am Anfang die Willkommen-Hilfedatei woher auch der Name kommt. Da die Hilfedatei HTML Dateien sind implementiert `WelcomePanel` das Interface `HyperlinkListener` um das Navigieren durch HTML Dateien zu ermöglichen.

```
public class WelcomePanel extends JPanel implements HyperlinkListener {
      JApplet parent;

      private String content;

      JEditorPane welcome = new JEditorPane();
...
}
```

`WelcomePanel` definiert den String `content` welcher auf die Hilfedatei zeigt und den `JEditorPane welcome` welcher die Datei anzeigt.

```
public WelcomePanel(JApplet parent, String s) {
      this.parent = parent;
      setLayout(new BorderLayout());
      setBackground(Color.white);
      content = s;
      createPanel(this);
}
```

Bei der Konstruktion eines Objekts wird der String `s` angegeben, welcher auf die HTML Dateien zeigt. In der Funktion `createPanel()` wird dann der Panel welcome mit dem Inhalt der Datei befüllt.

```
public void createPanel(Container pane) {
      welcome.addHyperlinkListener(this);
      welcome.setEditable(false);
      try {
            welcome.setPage(parent.getCodeBase() + content);
      }
      catch (IOException e) {
            System.err.println("Attempted to read a bad URL");
      }
      JScrollPane sPane = new JScrollPane(welcome);


sPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
      sPane.setPreferredSize(new Dimension(750, 500));
      add(sPane);
}
```

Die Funktion `hyperlinkUpdate()` erkennt das Klicken auf einen Link in der angezeigten Datei und zeigt dann den Inhalt dieses Links an.

```
public void hyperlinkUpdate(HyperlinkEvent e) {
      if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
            try {
                  URL urlString = e.getURL();
                  welcome.setPage(urlString);
                  welcome.repaint();
            }
            catch (IOException ex) {
                  System.err.println("Attempted to read a bad URL:
"+e.getURL());
            }
      }
}
```

3.2.1.6 EndPanel

Das ist der letzte Punkt des Trainingsprogramms. Dieses Panel soll am Ende vom Training dem Studenten ermöglichen seine Arbeit dem Betreuer mittels E-Mail zu senden. Das Panel setzt sich aus einem Textfeld und einer Schaltfläche zusammen. Das Textfeld ist für die Matrikelnummer vorgesehen, die dann zusammen mit dem Rest des Protokolls in einem E-Mail dem Betreuer nach dem Klicken der Schaltfläche „Bericht“ gesendet wird.

```
public class EndPanel  extends JPanel implements ActionListener {
      JApplet parent;
      JButton pbBericht = new JButton("Bericht");
      JLabel lText = new JLabel("Beim Klick auf die " +
                        "Schaltfläche sendest Du den Bericht an Betreuer");
      JTextField  mNr = new JTextField(7);
…
}
```

Nach dem Klick auf die Schaltfläche Bericht wird zuerst der Text der Nachricht erstellt. Dieser Text besteht aus den Protokollstrings vom Eingangstests und der Übung. Den Strings wird die Matrikelnummer vorgestellt. `String msg = createMsg();`. Danach wird ein Objekt der Klasse UEMail erstellt der dann die Kontrolle über das E-Mail übernimmt.

```
UEMail m = new UEMail();
String [] recipients = {"betreuer@server.com"};
try {
      m.postMail(recipients, "test labor", msg,
„eMNr@student.tuwien.ac.at“);
}
catch (MessagingException me) {
      System.err.println(me.toString());
      JOptionPane.showMessageDialog(this, me.toString());
}
```

Das Senden von E-Mail erfolgt mit der Funktion. `postMail()` die im `try/catch` Block ausgeführt wird. Falls die Funktion fehlschlägt, erzeugt sie eine `MessagingException`. Beim Aufruf der Funktion werden der Empfänger, der Betreff, die Nachricht und der Absender definiert. Die Adresse des Empfängers und des Absenders müssen von dem Netzwerkadministrator bestimmt werden, bzw. mit ihm abgesprochen werden.
In der jetzigen Version wird vor dem Erstellen der Nachricht auch das Protokoll angezeigt. Dieses Protokoll kann der Student dann anschauen oder ausdrücken.
Das Senden von E-Mail wird in der Diskussion der Klasse `UEMail` genauer beschrieben.

3.2.2 panels

In diesem package werden die Panels für die einzelnen Teilübungen erstellt. Die Panels für die Teilübungen werden von der Klasse ParentPanel abgeleitet. Die Klasse die nicht von ParentPanel abgeleitet wird ist die UEHelpPanel, die direkt von JPanel abgeleitet wird und der Anzeige der Hilfe im Hilfefenster bei der Teilübung dient.

Die Bilder der Panels mit der Bezeichnung der Elemente sind im Anhang angegeben.

3.2.2.1 ParentPanel

ParentPanel ist die Basisklasse für alle Klassen die zur Anzeige der einzelnen Teilübungen verwendet werden. Sie stellt die Grundfunktionen zur Verfügung die von abgeleiteten Klassen übernommen werden. Die Klasse ist eine abstrakte Klasse die nur als Basisklasse für die Klassen der Teilübungen dient.

```
abstract public class ParentPanel extends JPanel implements MouseListener,
     MouseMotionListener, ChangeListener {
           JApplet parent;
           List<Connection> conns = new ArrayList<Connection>();
           List<String> sConnsL = new ArrayList<String>();
           boolean bOK = false;

           abstract public void createPanel();
     ...
     }
```

In der ParentPanel-Klasse sind MouseListener, MouseMotionListener und ChangeListener Interfaces implementiert. Diese sind notwendig um auf die Bewegungen der Maus und auf Änderungen des Zustands eines der Elemente im Panel zu reagieren.
Die Klasse definiert zwei Listen die conns und die sConnsL, die die Verbindungen und ihre String-Präsentation bewahren. Die boolean Variable bOK steuert das Aktivsein der Elemente am Messplatz. Die Funktion `createPanel()` muss bei allen abgeleiteten Klassen neu geschrieben werden.

```
public List<String> getConnsS() {
           return sConnsL;
      }
      public List<String> getRevConnsS() {
           List<String> retList = new ArrayList<String>();
           for (Connection c : conns) {
                 retList.add(c.getRevConnString());
           }
           return retList;
      }
      public List<Connection> getConns() {
           return conns;
      }
```

Die Funktionen `getConnsS()`, `getRevConnsS()` und `getConns()` sind für die Abfrage der Verbindungen und ihrer String-Repräsentation vorgesehen.

3.2.2.2 FEGLLKPanel

Die Klasse FDP wird bei der Messungen der Leerlaufkennlinie am fremderregten Generator angezeigt.

```
public class FEGLLKPanel extends ParentPanel {
...
```

Zuerst werden die anzuzeigenden Elemente deklariert.

```
public FEGLLKPanel(JApplet prnt) {
      super(prnt);
      setOpaque(true);
      createPanel();
      addMouseMotionListener(this);
}
public void paint(Graphics g) {
      super.paint(g);
      //paint all connections
      for (Connection cl : conns)
            cl.paintConn(false);
}
```

Im Konstruktor wird die Funktion `createPanel()` aufgerufen die dann die Erstellung und Positionierung der Elemente in der Maske macht. Die Funktion `paint()` geht die Liste der Verbindungen durch und zeichnet sie erneut.

```
...
public void createPanel() {
...
      setLayout(new GridBagLayout());
      GridBagConstraints c = new GridBagConstraints();
      c.fill = GridBagConstraints.HORIZONTAL;
      c.ipadx = 2; c.ipady = 2;
...
      c.gridx=1; c.gridy=2; c.gridwidth = 4;
      add(m, c);
      c.gridx=0; c.gridy=3; c.gridwidth = 5;
      add(sm, c);
      c.gridx=5; c.gridy=3; c.gridwidth = 1;
      add(smS, c);

      setOpaque(true);

      setVisible(true);
}
```

In der Funktion `createPanel()` wird `GridBagLayout` als Layout für das Panel definiert. Diese Art von Layout ermöglicht die genaue Positionierung der Elemente in der Maske. Es ist auch möglich zu definieren wie viele Plätze ein Element einnimmt.

Der Unterschied zwischen dieser Klasse und anderen Klassen in dem package panels, die Teile der Übung darstellen, ist nur in Elementen die angezeigt werden und in ihrer Positionierung. Diese Klassen werden hier nur Listenweise angeführt:

`FEGAKPanel` - Messung der äußeren Kennlinie am fremderregten Generator
`FEGBKPanel` - Messung der Belastungskennlinie am fremderregten Generator
`FEGRKPanel` - Messung der Reguliererkennlinie am fremderregten Generator
`NSGAKPanel` – Nebenschlussgenerator / Äußere Kennlinie
`NSGERPanel` – Nebenschlussgenerator / Entregung
`NSGSEPanel` – Nebenschlussgenerator / Selbsterregung
`NZBMPanel` - Neutrale Zone / ballistische Methode
`NZMMPanel` - Neutrale Zone / Momentenmethode
`VGAKPanel` – Verbundgenerator / Äußere Kennlinie
`VGERPanel` – Verbundgenerator / Entregung
`VGSEPanel` – Verbundgenerator / Selbsterregung

3.2.2.3 UEHelpPanel

Diese Klasse dient der Anzeige von Hilfe im Fenster der gerate bearbeiteten Teilübung. Das entsprechende UEHelpPanel wird von der Klasse aus dem default Package kreiert.

```
helpPanelLLK = new UEHelpPanel();
createHelpPanel(helpPanelLLK, LLKPANE + "_B");
```

Dafür wird die Funktion `createHelpPanel()` der Klasse UEPanel aufgerufen. Diese Funktion ruft ihrerseits die Funktion `createPanel()` von UEHelpPanel auf.

```
public void createPanel (JApplet parent, String s) {
      String urlString = "./Help/" + s + ".html";

      help1 = new JEditorPane();
      help1.setContentType("text/html");
      help1.setEditable(false);

      try {
            help1.setPage(parent.getCodeBase() + urlString);
      }
      catch (IOException e) {
            System.err.println("Attempted to read a bad URL:"+urlString);
      }
      sPane = new JScrollPane(help1);

sPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
      sPane.setPreferredSize(new Dimension(700, 80));

      add(sPane);

      repaint();

}
```

Die Funktion `createPanel()` bekommt als Parameter den String der HTML Datei die angezeigt werden sollte. Der Inhalt der HTML Datei wird dann in einem JScrollPane Objekt, welches ein weiteres Objekt der Klasse JEditorPane beinhaltet, angezeigt. Das JEditorPanel-Objekt bekommt seinen Inhalt durch den Aufruf der Funktion `setPage()`.

3.2.3 messPlatz

Package messPlatz beinhaltet alle Klasse, welche Elemente des Messplatzes beschreiben.

3.2.3.1 MPObject

MPObject ist die Basisklasse die die Elemente vom Messplatz beschreibt. Von ihr werden alle anderen Klassen die Elemente des Messplatzes, wie Widerstände, Ampermeter usw. beschreiben, abgeleitet. Sie beinhaltet einen Vector der festhält mit welchen anderen Elementen dieser verdrahtet ist sowie die Funktion die Verknüpfungen herstellt.

```
abstract public class MPObject extends JPanel {
      public List<MPObject> connWith = new ArrayList<MPObject>();

      boolean bHasValue = false;
      double dVal = 0.0;

      abstract public void createPanel();
...
}
```

Die Klasse MPObject ist eine abstrakte Klasse mit der abstrakten Funktion `createPanel()`. Dadurch wird jede abgeleitete Klasse gezwungen für ihre Darstellung am Bildschirm selbst zu sorgen. Die Klasse beinhaltet auch zwei Variablen: `bHasValue` und `dVal`. Diese zwei Variablen determinieren ob das Objekt einen Wert hat (z.B. Stromstärke) und wie groß dieser Wert ist. Mit Funktionen `getValue()` und `hasValue()` kann auf diese Werte zurückgegriffen werden.

```
public double getValue() {
      if (bHasValue)
            return dVal;
      else
            return 0.0;
}
public boolean hasValue() {
      return bHasValue;
}
```

Die Liste `public List<MPObject> connWith = new ArrayList<MPObject>();` beinhaltet alle anderen MPObject Objekte mit denen dieses Objekt verbunden ist. Die Herstellung bzw. das Löschen einer Verbindung erfolgt mit den Funktionen `connectWith()` und `disconnectFrom()`. Mit diesen Funktionen wird der Liste `connWith` ein neues Objekt hinzugefügt bzw. aus der Liste entfernt.

```
public boolean connectWith(MPObject obj) {
      if (!connWith.contains(obj)) {
            connWith.add(obj);
            return true;
      }
      return false;
}
public boolean disconnectFrom(MPObject obj) {
      if (!connWith.isEmpty() && connWith.contains(obj)) {
            connWith.remove(obj);
            return true;
      }
      return false;
}
```

Beide Funktionen geben eine Variable vom Typ boolean zurück und zwar true, wenn die Operation erfolgreich war bzw. false, falls die Operation fehlgeschlagen ist.

3.2.3.2 Connection

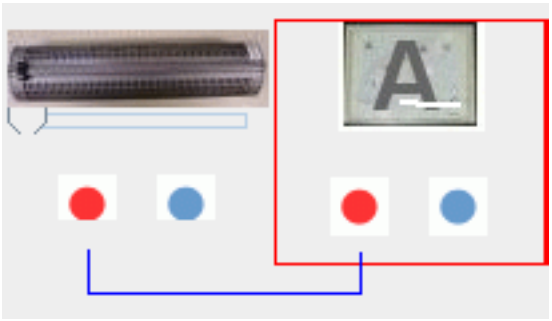


Bild 3.1: Connection

Die Verbindungen am Messplatz werden mit der Klasse Connection dargestellt. Die Klasse beinhaltet zwei Objekte der Klasse MPObject die verbunden werden sowie die Methode diese Verbindung zu zeichnen. Connection überschreibt auch die Methode "Object.equals()" um das Erstellen von Listen mit den Elementen der Klasse Connection zu ermöglichen bzw. sie in der Liste zu finden.

```
public class Connection  {
      MPObject start, end;
...
      String sConn;
      static int maxY = 0;
...
}
```

String `sConn` hält die Zeichenrepräsentation der Verbindung. `static int` *maxY* ist die maximale Y Koordinate der Verbindung. Sie wird bei dem Zeichnen der Verbindungen benötigt.

```
public boolean equals(Object o) {
      Connection oc = (Connection)o;
      if (oc.start == this.start && oc.end == this.end)
            return true;
      else if (oc.start == this.end && oc.end == this.start)
            return true;
      return false;
}
```

Die Funktion equals() vergleicht zwei Verbindungen und gibt true zurück wenn ihr Anfangs- und Endobjekt gleich sind. Dabei wird überprüft, ob das Startobjekt einer Verbindung gleichzeitig das Start- oder Endobjekt der anderen Verbindung ist.

```
public void setStart(MPObject o) {
      start = o;
      bCStartSetManual = false;
      sConn = start.getParent().getParent().getName() + "::" +
      int ym = start.getParent().getParent().getY();
      maxY = maxY < ym ? ym : maxY;
}
```

Die Funktionen `setStart()` und `setEnd()` bestimmen Anfang und Ende einer Verbindung. Sie bestimmen auch den Wert von maxY der für das Zeichnen ausschlaggebend ist.

```
public void paintConn(boolean bErase) {
      int startX, startY, endX, endY;
      if (bErase)
            nrOfConns--;
      else {
            if (connNr == NF)
                  connNr = nrOfConns++;
      }
...
```

Die Funktion `paintConn()` bekommt als Parameter eine boolean Variable die bestimmt ob die Verbindung erstellt oder gelöscht wird. Gleich am Anfang werden auch die Variablen für die Koordinaten des Anfang- und des Endobjekts deklariert. Anschließend werden diese Koordinaten anhand der Positionen des Anfangs- und Endobjekts errechnet.

```
      if (bCStartSetManual) {
            startX = rCoordsS.x + rCoordsS.width/2;
            startY = rCoordsS.y + rCoordsS.height;
      }
      else {
            startX = start.getParent().getParent().getX() +
start.getParent().getX() + start.getX() + start.getWidth()/2;
            startY = start.getParent().getParent().getY() +
start.getParent().getY() + start.getY() + start.getHeight();
      }
...
```

(auf gleiche Weise auch für das Endobjekt)

```
...
      int yM = Math.max(startY, endY);
...
```

yM ist die niedrigste Y Koordinate der Verbindung.

```
...
      Graphics pg = bGraphicsGiven ? this.g : c.getGraphics();
...
      pg.drawLine(startX, startY+1, startX, yM+10+connNr*2);
      pg.drawLine(startX, yM+10+connNr*2, endX, yM+10+connNr*2);
      pg.drawLine(endX, yM+10+connNr*2, endX, endY+1);

}
```

Am Ende wird die Verbindung anhand von berechneten Koordinaten gezeichnet. Die Verbindungslinien werden für jede Verbindung vertikal versetzt.

3.2.3.3 Klemme

Bild 3.2: Klemme

Die Klasse Klemme ist die Klasse die dann eigentlich die Verbindungen erstellt.

Diese Klasse repräsentiert alle Verbindungsklemmen am Messplatz. Die Klasse implementiert außerdem das Interface `MouseListener`. In der Funktion `mouseClicked()` dieses Interfaces wird dann eine neue Verbindung hergestellt bzw. dieselbe gelöscht. Das Löschen der Verbindung U erfolgt mit der rechten Maustaste. Die Verbindung wird der Liste der Verbindungen im Panel zugefügt bzw. aus der Liste entfernt. Dabei wird auch dem Start- und dem Endelement jeweils das andere Element in die Liste `connWith` hinzugefügt bzw. aus der Liste entfernt.

```
public class Klemme extends MPObject implements MouseListener {
	static boolean bStart;
	static Klemme cStart, cEnd;
	static MPObject mpoStart;

	Image img, aImg;

	JLabel kl;

	public Klemme(Image img) {
		this.img = img;

		ImageFilter imgF = new MPImgFilter(60);
		Image srcI = img;
		ImageProducer imgP = new FilteredImageSource(srcI.getSource(),
imgF);
		aImg = createImage(imgP);

		createPanel();
	}
```

Im Konstruktor der Klasse werden zwei Bilder der Klemme erstellt: Image `img`, das die Klemme im inaktiven und `aImg`, welches die Klemme im aktiven Zustand darstellt.

Die wichtigste Funktion der Klasse Klemme ist die Funktion `mouseClicked()`.

```
public void mouseClicked(MouseEvent e) {
	MPObject mpo =(MPObject)this.getParent();
	Container pp = mpo.getParent();
	while (pp.getClass().getSuperclass() != ParentPanel.class)
		pp = pp.getParent();
	ParentPanel pp1 = (ParentPanel)pp;
	if (pp1.isOK()) {
		UEMessage msg = new UEMessage(pp1);
		msg.setTitle("Alarm");
		msg.setMessage("<html><h1 align=center>Halooo!</h1>Die
Schaltung
                                    ist schon richtig verdrahtet!</html>");
		msg.createAndShow();
		e.consume();
		return;
	}
```

Zuerst wird überprüft ob in dem Panel schon alles richtig verbunden ist. Die Kontrolle wird durchgeführt in dem alle Parents der Klemme durchsucht werden bis das ParentPanel erreicht wird. Dann wird der Status des Panels überprüft und entsprechende Meldung ausgegeben.

```
...
	bStart = !bStart;
	boolean bRemoveConn = false;
	if (e.getButton() == MouseEvent.BUTTON3) {
		bRemoveConn = true;
	}
```

Im vorigen Abschnitt wird überprüft ob die rechte Maustaste gedrückt wurde. Falls ja wird die Verbindung als „Zu Löschen“ gekennzeichnet indem `bRemoveConn = true` gesetzt wird. Die static Variable *`bStart`* wird negiert um Start und Ende der Verbindungen zu unterscheiden.

```
	if (bStart) {
		cStart = this;
		mpoStart = (MPObject)this.getParent();
	}
	else {
		cEnd = this;
		cStart.setIcon(cStart.getIcon());
		cStart.repaint();
	}
	kl.setIcon(new ImageIcon(bStart ? aImg : img));
	kl.repaint();
```

Die Objekte *`cStart`* und *`cEnd`* werden abhängig von der Variablen *`bStart`* gesetzt. Auch das Label, das das Bild der Klemme anzeigt wird abhängig von *`bStart`* gesetzt.

```
	if(!bStart) {
		if (this.equals(cStart))
			return;
```

Falls die gerade ausgewählte Klemme gleich der Anfangsklemme, ist wird die Funktion beendet.

```
            mpo =(MPObject)this.getParent();
            pp = mpo.getParent();
            while (pp.getClass().getSuperclass() != ParentPanel.class)
                  pp = pp.getParent();
            pp1 = (ParentPanel)pp;

            Connection c = new Connection(pp1);
            c.setStart(cStart);
            c.setEnd(cEnd);
```

Es werden wieder die Parentklassen abgefragt bis ParentPanel erreicht wird. Dann wird eine neue Verbindung für diesen ParentPanel erstellt.

```
            if (!bRemoveConn) {
                  if (!pp1.addConnection(c))
                        pp1.getParentA().showStatus("Add connection
failed");
                  else {
                        cStart.connectWith(cEnd);
                        cEnd.connectWith(cStart);
                  }
            }
            else {
                  if (!pp1.removeConnection(c))
                        pp1.getParentA().showStatus("Remove connection
failed");
                  else {
                        cStart.disconnectFrom(cEnd);
                        cEnd.disconnectFrom(cStart);
                  }
            }
```

Die Verbindung wird in Abhängigkeit von der Variable bRemoveConn erstellt oder gelöscht.

```
            pp1.getParentA().showStatus(c.getConnString());

            cStart = null;
            mpoStart = null;
      }
}
```

Die Herstellung bzw. Löschung der Verbindung geschieht nur wenn das Ende der Verbindung erreicht ist. Deswegen wird `cStart` zurückgesetzt, da nicht mehr benötigt.
Die Klasse Klemme wird in anderen Klassen verwendet die die Klemmen von verschiedenen Elementen des Messplatzes darstellen. Das sind die Klassen: `PMKlemmen`, `PMEKlemmen`, `MKlemmen` und `ErrKlemmen` und werden hier nicht detailliert behandelt. Es werden nur die wichtigsten Eigenschaften aufgezählt

- `PMKlemmen` ist eine zweipolige Klemme
- `PMEKlemmen` ist eine zweipolige Klemme mit Erdungsanschluss
- `MKlemmen` stellt die Klemmen der Maschine dar
- `ErrKlemmen` stellt die Klemmen vom Erregerkreis dar.

3.2.3.4 Maschine

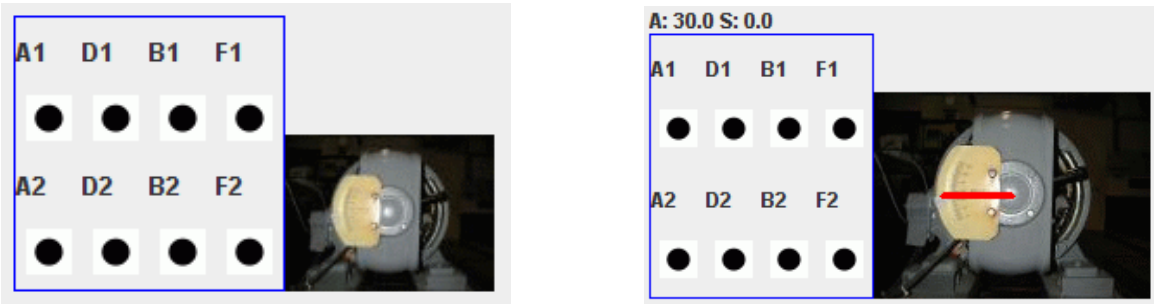


Bild 3.3: Maschine

Diese Klasse repräsentiert die Gleichstrommaschine am Messplatz. Die Klasse hat zwei Konstruktoren je nach dem ob die Maschine mit dem Hebel angezeigt wird. Sie implementiert zwei Interface: `MouseListener` und `MouseMotionListener`. Zum Zeichen der Maschine werden zwei Funktionen verwendet: `createPanel()` und `paintComponent()`.

```
public class Maschine extends MPObject implements MouseListener,
MouseMotionListener {
	MKlemmen kl;
	Image imgM, imgK;
	JLabel lSpd;
	boolean bPaintHebel;
	double dWinkel = Math.PI;

	int aktY = 0;
	int lastY = 0;
	boolean bPowerOn = false;
	boolean bSpdSetManual = false;
	double dRotSpeed = 0.0;
```

`JLabel lSpd` zeigt die Rotationsgeschwindigkeit der Maschine und den Winkel der Bürsten.

```
	public Maschine(Image imgM, Image imgK) {
		super();
		this.imgM = imgM;
		this.imgK = imgK;
		bPaintHebel = false;
		addMouseListener(this);
		addMouseMotionListener(this);
		createPanel();
	}
	public Maschine(Image imgM, Image imgK, boolean b) {
		super();
		this.imgM = imgM;
		this.imgK = imgK;
		bPaintHebel = b;
		addMouseListener(this);
		addMouseMotionListener(this);
		createPanel();
	}
```

Der erste der Konstruktoren zeichnet die Maschine ohne den Hebel und der Zweite mit dem Hebel.

Die Funktionen, die die Maschine zeichnen, werden entweder aus dem Konstruktoren aufgerufen oder von dem Javasystem, im Falle, dass die Komponente neu gezeichnet werden muss.

```
public void createPanel() {
	setLayout(new BorderLayout());
	if (imgK != null) {
		kl = new MKlemmen(imgK);
		kl.setBorder(BorderFactory.createLineBorder(Color.blue));
	}
	else
		kl = null;
	lSpd = new JLabel("A: " + getAngle() + " S: " + rotSpeed());
...
```

Die Funktion kreiert Klemmen der Maschine `kl` und das Label `lSpd`. Bei dem Erstellen von `lSpd` werden die Funktionen `getAngle()` und `rotSpeed()` verwendet. Da die Aufgabe dieser Diplomarbeit nicht Simulation der Gleichstrommaschine war, geben diese Funktionen provisorische Werte zurück.

```
	if (kl != null)
		add(kl, BorderLayout.LINE_START);
	if(bPaintHebel)
		add(lSpd, BorderLayout.NORTH);

	if(bPaintHebel)
		aktY = lastY = imgM.getHeight(this)/2;

	setVisible(true);
}
```

Falls `kl` vorhanden ist wird es gezeichnet. Genauso wird die `lSpd` in Abhängigkeit von `bPaintHebel` gezeichnet.
Die Funktion `paintComponent` wird vom System jedes Mal aufgerufen, wenn die Komponente neu gezeichnet werden muss.

```
public void paintComponent(Graphics g) {
	super.paintComponent(g);
	Point ptM;
	if (kl != null)
		ptM = new Point(kl.getWidth(), getSize().height -
imgM.getHeight(this));
	else
		ptM = new Point(0, 0);
	g.drawImage (imgM, ptM.x, ptM.y, this);
```

Zuerst werden die Koordinaten der Klemmen ermittelt und daraus der Punkt wo das Bild gezeichnet werden soll.

```
      if (bPaintHebel) {
            Graphics2D g2d = (Graphics2D)g.create();
            Point ptZ, ptE;
            int width = 4;
            g2d.setColor(Color.red);
            g2d.setStroke(new BasicStroke(width, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND));
```

Falls der Hebel gezeichnet werden soll, wird ein Objekt der Klasse `Graphics2D` erstellt, das das Zeichnen von dicken Linien ermöglicht. Die Linie, die den Hebel darstellt, wird nach der Berechnung von ptZ und ptE Punkten gezeichnet.

```
...
            g2d.drawLine(ptZ.x, ptZ.y, ptE.x, ptE.y);
      }

}
```

Die Funktion `mouseClicked()` übernimmt das Zeichnen vom Hebel in Schritten wenn die linke Maustaste gedrückt wird.

```
public void mouseClicked(MouseEvent e) {
      if (bPaintHebel) {
            e.consume(); // don't let this event go to other components
            int hi = e.getY() - getSize().height + imgM.getHeight(this);
            if (e.getButton() == MouseEvent.BUTTON1)
                  aktY += 1;
            else if (e.getButton() == MouseEvent.BUTTON3)
                  aktY -= 1;
...
            repaint();
      }
```

Der Klick mit der linken Maustaste zieht den Hebel nach unten und der Klick mit der rechten Maustaste nach oben.

```
      else {
            if (e.getButton() == MouseEvent.BUTTON1)
                  dRotSpeed -= 1;
            else if (e.getButton() == MouseEvent.BUTTON3)
                  dRotSpeed += 1;
      }
}
```

Wenn der Hebel nicht gezeichnet wird bewirkt ein Mausklicks die Änderung der Rotationsgeschwindigkeit der Maschine. Mit der linken Maustaste wird die Geschwindigkeit reduziert und mit der rechten Maustaste erhöht. Dieses wird bei dem Schrage Motor verwendet.
Die Position des Hebels lässt sich auch mit dem Ziehen der Maus ändern.

```
public void mouseDragged(MouseEvent e) {
      if (bPaintHebel) {
            e.consume(); // don't let this event go to other components
            if (e.getY() > lastY)
                  aktY += 4;
            else
                  aktY -= 4;
            if (aktY > getSize().height)
                  aktY = getSize().height;
            if (aktY < 0)
                  aktY = 0;
...
      }
}
```

Die Funktion `getAngle()` gibt die entsprechende Nummer der Skalierung an der Maschine zurück. Die Funktion `getElAngle()` gibt den entsprechenden elektrischen Winkel zurück. Die Funktion `rotSpeed()` errechnet die Rotationsgeschwindigkeit der Maschine in Abhängigkeit von dem Winkel der Bürsten. An dieser Stelle sei wieder betont, dass die Aufgabe nicht in der Simulation der Maschine lag und dass deswegen diese Funktion nur aus Vorbereitung für weitere Entwicklung des Programmes steht.

3.2.3.5 SchrageMaschine

Bild 3.4: SchrageMaschine

SchrageMaschine ist die Klasse die den Schrage Motor, der am Messplatz installiert ist und der den Generator steuert, darstellt. Die Klasse setzt sich aus einer Maschine, einem Zähler der die Drehzahl anzeigt, einem Schalter und einem Versorgungschalter zusammen.

```
public class SchrageMaschine extends MPObject implements MouseListener,
ActionListener {
...
      Image [] imgS;
      Timer t = new Timer(250, this);
...
}
```

Für das Anzeigen von dem Rotorschalter wird eine Reihe von Bildern `imgS`, deren Anzeige von dem Timer `t` gesteuert wird, verwendet. Der Timer wird initialisiert mit der Zeit von 250 Millisekunden für jede Aktion. Das Timer-Ereignis wird in der Funktion `actionPerformed()` behandelt und es wird das entsprechende Bild aus der Reihe `imgS` als Bild des Schalters

angezeigt. Mit der Funktion `setPower()` wird die Maschine, nach Überprüfung ob der Rundschalter und die Versorgung eingeschaltet sind, „unter Strom“ gesetzt.

```
public void mouseClicked(MouseEvent e) {
      if(e.getSource() == schalter) {
...
            t.start();
      }
      if(e.getComponent() == m) {
            if (bPowerOn) {
                  m.mouseClicked(e); // gibt dieses Event an die Maschine
weiter
                  zaehler.setText("" + m.getRotSpeed()); // aktualisiere
die Anzeige von dem zaehler
                  zaehler.repaint();
            }
      }
      if(e.getComponent() == u) {
            bVersorgungOn = !bVersorgungOn;
            setPower(u.isOn());
      }
...
}
```

Bei dieser Funktion wird überprüft was gedrückt wurde und dementsprechend werden verschiedene Aktionen durchgeführt.

3.2.3.6 Versorgung

Bild 3.5: Versorgung

Die Klasse Versorgung implementiert den Versorgungsschalter am Messplatz.

```
public class Versorgung extends MPObject  implements MouseListener {
      public final double dSpannung;
...
      PMEKlemmen kl;
      JLabel lImg;
...
```

Die Klasse definiert den Wert der Spannung die an den Klemmen `PMEKlemmen kl` freigegeben wird. `JLabel lImg` zeigt das Bild des Schalters an.

```
...
public Versorgung(Image iP, Image iM, Image iE, Image imgOn, Image imgOff,
                                double dSpannung) {
...
}
public Versorgung(Versorgung o) {
...
}
```

Es sind zwei Konstruktoren vorgesehen. Neben dem normalen Konstruktor, der die Bilder für die Klemmen und den Schalter im eingeschalteten und ausgeschalteten Zustand sowie die Spannung definiert, ist auch ein Kopierkonstruktor vorhanden, der die Parameter von dem zu kopierenden Objekt übernimmt. Die Funktion `mouseClicked()` übernimmt dann das Aus- bzw. Einschalten von der Versorgung. Es ist auch die Funktion `getSpannung()` definiert die den Wert der Spannung zurück gibt.

3.2.3.7 Widerstand

Diese Klasse repräsentiert Widerstände am Messplatz. Die Objekte der Klasse werden auf zwei Weisen erstellt. Die erste Möglichkeit ist das Objekt mit dem Bild und dem Wert des Widerstandes zu kreieren. Die zweite Variante ist zusätzlich zu definieren, dass der Wert des Widerstandes angezeigt wird.

```
public class Widerstand extends MPObject {
      Image img;
      public double dWert = 1.0;
      public boolean bOn = false;
      private boolean bShowValue = true;

      JLabel rImg;
...
}
```

Die Klasse Widerstand wird bei der Klasse WiderstandPult verwendet.

3.2.3.8 WiderstandPult

Bild 3.6: WiderstandPult

Ihre Hauptaufgabe ist es die Widerstände zu bedienen mit denen die Last der Maschine gestellt wird. Diese Klasse stellt den Widerstandschrank am Messplatz dar.

```
public class WiderstandPult extends MPObject implements MouseListener {
...
      Widerstand r0, r1, r2, r3, r4, r5, r6, r7, r8, r9;
      PMEKlemmen kl = null;
      JLabel lRGes = new JLabel("000");
      JLabel lOhm = new JLabel("Ohm");

      private double dRGes = 0.0;
      private double dRGesI = 0.0;
...
}
```

Die Klasse definiert zehn Objekte vom Typ `Widerstand`, ein Objekt von Typ `PMEKlemmen` sowie zwei Labels, die den Gesamtwert des Widerstands und die Einheit anzeigen. Die Klasse implementiert außerdem den Interface `MouseListener` womit das Ändern des Zustandes der einzelnen Widerstände ermöglicht wird.

```
public WiderstandPult(Container parent, Image imgOff, Image imgOn, Image
imgROff, Image imgROn, Image imgP, Image imgM, Image imgE) {
      this.imgOn = imgOn;
      this.imgOff = imgOff;
      this.imgROn = imgROn;
      this.imgROff = imgROff;
      this.imgP = imgP;
      this.imgM = imgM;
      this.imgE = imgE;
      dRGes = 0.0;
      this.parent = parent;
      this.bHasValue = true;

      createPanel();
      this.repaint();
}
```

Im Konstruktor der Klasse werden dem Objekt die Bilder für verschiedene Elemente der Klasse zugewiesen. Dadurch werden in der Funktion `createPanel()` verschiedene Elemente erstellt. Außerdem wird der Gesamtwiderstand mit `dRGes = 0.0` initialisiert.

```
public void createPanel() {
      setBackground(parent.getBackground());
      r0 = new Widerstand(this.imgOff, 220.0);
      r0.setPreferredSize(new Dimension(imgOff.getWidth(this),
imgOff.getHeight(this)+20));
      r0.addMouseListener(this);
...
```

Alle Widerstände werden auf die gleiche Weise mit dem Bild `imgOff` und dem entsprechenden Wert des Widerstands initialisiert. Durch die Funktion `addMouseListener()` wird das Ein- bzw. Ausschalten des Widerstands ermöglicht.

```
kl = new PMEKlemmen(imgP, imgM, imgE);
```

Auf ähnliche Weise werden auch die Klemmen erstellt.
`setLayout(new GridBagLayout());`
Für das Layout wird `GridBagLayout` verwendet damit die Widerstände räumlich ähnlich dem Schrank am Messplatz angezeigt werden.
Die Funktion `mouseClicked()` ändert nach dem Klick auf einen Widerstand dessen Zustand und berechnet den Gesamtwiderstand.

3.2.3.9 ErregerR

Bild 3.7: ErregerR

Die Klasse `ErregerR` stellt den Widerstand dar über welchen der Erregerstrom eingespeist wird.

Diese Klasse verwendet den Interface `ChangeListener` um die Änderungen am Schieber, der den Widerstand simuliert, zu erfassen. Dies erfolgt in der Funktion `stateChanged()`.

```
public void stateChanged(ChangeEvent e) {
      JSlider src = (JSlider)e.getSource();
      aWert = src.getValue();
      this.dVal = aWert;
      cl.stateChanged(e);
}
```

`aWert` gibt den aktuellen Wert vom Widerstand zurück. Der `slider` der den Wert zurück gibt wird in der Funktion `createPanel()` erstellt.

```
public void createPanel() {
      ImageIcon imgI = new ImageIcon(img);
      kl = new PMKlemmen(iP, iM);

      slider = new JSlider(0, nMaxWert, 0);
      slider.addChangeListener(this);
...
}
```

In der gleichen Funktion werden auch die Klemmen für den Widerstand erstellt.

3.2.3.10 MessWertAnzeige

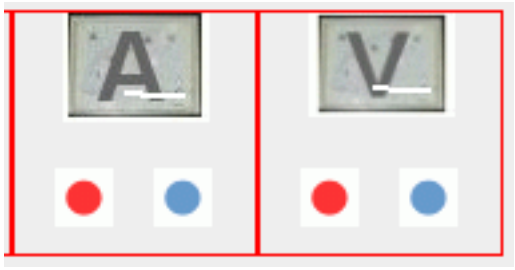


Bild 3.8: MessWertAnzeige

Die Klasse MessWertAnzeige repräsentiert die Amperemeter und die Voltmeter am Messplatz.

```
public class MessWertAnzeige extends MPObject {
...
      PMKlemmen klemmenPane;
      double dMessBereich = 200.0;
      protected double dMessWert = 0.0;
...
      protected String sMessWert = new String();
...
}
```

Die Klasse definiert die Variable für den Messbereich `dMessBereich` und die Variable für den aktuellen Messwert `dMessWert`. Auch wird die String-Variable `sMessWert` für das Zeichnen der Ausgabe des aktuellen Messwertes definiert. Die Variable `dMessWert` wird durch den Aufruf der Funktion `setWert()` gesetzt.

```
public void setWert(double dWert) {
      if (this.dMessBereich < dWert)
            this.dMessWert = this.dMessBereich;
      else
            this.dMessWert = dWert;

      sMessWert = Math.round(dWert*100)/100.0 + " ";
      this.dVal = dMessWert;
      repaint();
}
```

Die Funktionen `fx()` und `fy()` errechnen X und Y Koordinate vom Zeiger.

```
private double fx() {
      double x = Math.PI * (1 - dMessWert/(dMessBereich*2));
      return ptRoot.x + Math.cos(x)*length;
}
private double fy() {
      double y = Math.PI * (1 - dMessWert/(dMessBereich*2));
      return ptRoot.y - Math.sin(y)*length;
}
```

In der Funktion `paintComponent()` werden dann das Bild von Amperemeter/Voltmeter sowie der Zeiger und der aktuelle Wert gezeichnet.

```
public void paintComponent(Graphics g) {
      super.paintComponent(g);
      int img_x = getSize ().width/2 - img.getWidth (this)/2;
      int img_y = getSize ().height/2 - img.getHeight (this)/2;
...
      g.drawImage (img, img_x, 0, this); //Das Bild des
Ampermeter/Voltmeter Zeichnen

      Graphics2D g2d = (Graphics2D)g;
      int width = 2;
      g2d.setStroke(new BasicStroke(width));
      g2d.setColor(Color.white);
      g2d.drawLine(ptRoot.x, ptRoot.y, (int)fx(), (int)fy());
      g2d.setColor(Color.yellow);
      g2d.drawString(sMessWert, (int)fx()-20,(int)fy());
      width = 1;
      g2d.setStroke(new BasicStroke(width));
}
```

3.2.3.11 Restliche Klassen

Es werden auch die restlichen Klassen nur kurz dargestellt.

- `Schalter`

Bild 3.9: Schalter

Die Klasse `Schalter` repräsentiert den Schalter bei der Schrage Maschine. Die Klasse ist für jeden Schalter universell einsetzbar.

- `BMSchalter`

Bild 3.10: BMSchalter

Diese Klasse repräsentiert den Schalter, der bei der ballistischen Methode eingesetzt wird. Zusätzlich zu den Funktionen der Klasse `Schalter` inkludiert `BMSchalter` auch die zugehörigen Klemmen.

- `Oszi`

Bild 3.11: Oszi

Bei der Ermittlung der neutralen Zone mittels ballistischer Methode wird auch das Oszilloskop verwendet, welches durch diese Klasse gezeigt wird.

- `ErrKreis`

Bild 3.12: ErrKreis

`ErrKreis` stellt den Erregerkreis vom Messplatz dar. Beim Klick auf den `ErrKreis` wird ein größeres Schema der Verbindung angezeigt.

3.2.4 ueTools

Das Package ueTools beinhaltet Hilfsklassen die bestimmte Funktionen ausführen die nicht mit anderen Funktionen des Applets direkt in Verbindung stehen.

3.2.4.1 UEMail

`UEMail` ist die Klasse die das Senden von E-Mails ermöglicht.

```
public class UEMail {
      private static String SMTP_AUTH_USER = "user@server.ext";
      private static String SMTP_AUTH_PWD = "pasword";
      public void postMail( String recipients[ ], String subject, String
message,                      String from)
            throws MessagingException
      {
            boolean debug = false;
      //    Set the host smtp address
            Properties props = new Properties();
            props.put("mail.smtp.host", "myserver");
            props.put("mail.smtp.auth", "true");
            Authenticator auth = new SMTPAuthenticator();
      //    create some properties and get the default Session
            Session session = Session.getDefaultInstance(props, auth);
            //Session session = Session.getDefaultInstance(props,
null);//Diese Zeile ist für den Fall, dass die Authentifikation nicht nötig
ist.
            session.setDebug(debug);
      //     create a message
            Message msg = new MimeMessage(session);
      //     set the from and to address
            InternetAddress addressFrom = new InternetAddress(from);
            msg.setFrom(addressFrom);
            InternetAddress[] addressTo = new
InternetAddress[recipients.length];
            for (int i = 0; i < recipients.length; i++) {
                  addressTo[i] = new InternetAddress(recipients[i]);
            }
            msg.setRecipients(Message.RecipientType.TO, addressTo);
      //     Optional : You can also set your cyour custom headers in the
Email if you Want
            msg.addHeader("MyHeaderName", "myHeaderValue");
      //     Setting the Subject and Content Type
            msg.setSubject(subject);
            msg.setContent(message, "text/plain");
            Transport.send(msg);
      }

      private class SMTPAuthenticator extends javax.mail.Authenticator {
            public PasswordAuthentication getPasswordAuthentication() {
                  String username = SMTP_AUTH_USER;
                  String password = SMTP_AUTH_PWD;
                  return new PasswordAuthentication(username, password);
            }
      }
}
```

Das Ausführen der Funktion `postMail()` ist im Prinzip begrenzt auf das Setzen von verschiedenen Eigenschaften der Objekte aus der `javax.mail` und `javax.mail.internet` Packages und das Ausführen ihrer Funktionen.
Die eigentliche Schwierigkeit bei der Klasse `UEMail` sind die Sicherheitseinschränkungen beim Applet. Diese Einschränkungen erlauben dem Applet nur auf den Host zuzugreifen von dem er auch gestartet wurde. Deswegen ist es notwendig, dass sich das Applet und der Mailserver auf dem gleichen Rechner befinden.

3.2.4.2 UEMessage

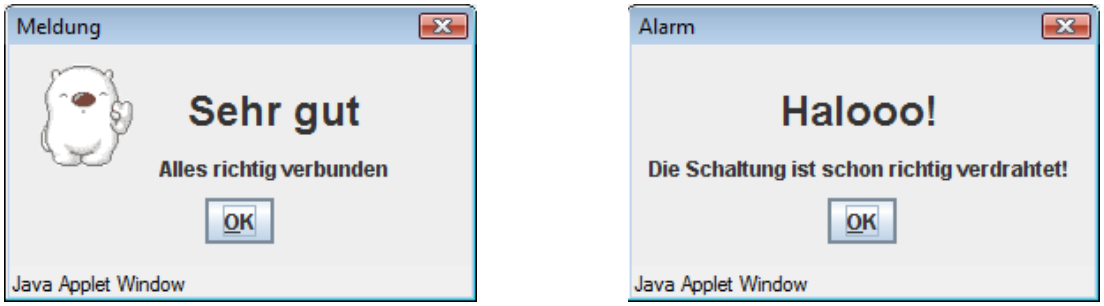


Bild 3.13: Meldungen

Die Klasse `UEMessage` dient zur Anzeige verschiedener Meldungen die ausgegeben werden während das Applet ausgeführt wird.

```
public class UEMessage extends JOptionPane {
      Container parent;
      String msg, title;
      ImageIcon img;
      boolean bSet = false;
...
```

Die Klasse definiert zwei Strings die die Meldung und den Titel im Fenster der Meldung definieren. Auch wird das Bild das mit der Meldung angezeigt wird definiert.

```
      public UEMessage(Container p) {
            super();
            this.parent = p;
      }
      public UEMessage(Container parent, String title, String msg,
ImageIcon i) {
            super();
            this.parent = parent;
            this.msg = msg;
            this.title = title;
            this.img = i;
      }
...
```

Es sind zwei Konstruktoren vorgesehen: in Einem wird nur der Kontainer der die Meldung ausgibt definiert und in dem Anderen werden auch die Nachricht, das Bild und der Titel sofort übergeben.

```
public void createAndShow() {
      if (this.img != null) {
            this.setIcon(img);
      }
      if (this.msg != null)
            this.setMessage(msg);
      JDialog d = this.createDialog(parent, title);
      d.setVisible(true);
      bSet = false;
}
```

Die Funktion `createAndShow()` erstellt und zeigt den Dialog mit seinen Parametern an. Die folgenden Funktionen `setTitle()`, `setMessage()` und `setIcon()` können verwendet werden um die verschiedenen Parameter einzeln zu setzen.

```
public void setTitle(String title) {
      this.title = title;
}
public void setMessage(String msg) {
      super.setMessage(msg);
      this.msg = msg;
}
public void setIcon(ImageIcon ii) {
      super.setIcon(ii);
      this.img = ii;
}
```

3.2.4.3 MPImgFilter

MPImgFilter ist die Klasse die beim Auswählen eines Elementes in Applet diesen dann schattiert anzeigt sofern diese Funktionalität bei dem Element vorgesehen ist.

```
public class MPImgFilter extends RGBImageFilter {
      int nDarkness;
      public MPImgFilter(int nDarkness) {
            canFilterIndexColorModel = true;
            this.nDarkness = nDarkness;
      }
```

Die Klasse ist abgeleitet von der Java Klasse `RGBImageFilter` und definiert durch den Parameter `nDarkness` wie stark das Objekt verdunkelt werden soll.

```
      public int filterRGB(int x, int y, int rgb) {
            int r = (rgb >> 16) & 0xff;
            int g = (rgb >> 8) & 0xff;
            int b = (rgb >> 0) & 0xff;

            if (nDarkness > 100)
                  nDarkness = 100;
            if (nDarkness < 0)
                  nDarkness = 0;
...
            return (rgb & 0xff000000) | (r << 16) | (g << 8) | (b << 0);
      }
}
```

Die Funktion `filterRGB()` muss überschrieben werden um die gewünschte Filterung zu erzielen. In unserem Fall gilt es nur das Bild etwas zu verdunkeln.

4 Applet Bedienung

Das Applet ist ein Javaprogramm, das in einem Internetbrowser ausgeführt wird. Damit erfüllt das Programmieren eines Applets zwei wichtige Vorgaben dieser Arbeit: Plattformunabhängigkeit und Verfügbarkeit für Studenten. Das Applet soll leicht bedienbar sein und den Studenten zur Vorbereitung der Laborübung „Messungen an der Gleichstrommaschine“ helfen.
Die Übung ist in mehrere Teilübungen aufgeteilt. Um diese darzustellen wurde für das Appletlayout das TabbedLayout gewählt. Für jeden Teil der Übung ist eine eigene Karte vorgesehen. Die Karten selbst sind bei Bedarf als CardLayout realisiert da unter einem Übungsteil mehrere Messungen vorgesehen sind. Im weiteren Verlauf werden die einzelnen Karteien (Tabs) beschrieben.

4.1 Eingangstest

Vor der Übung ist ein Zulassungstest vorgesehen bei welchem geprüft wird wie gut die Studenten für die Übung vorbereitet sind. Durch die Kartei „Eingangstest“ erhalten sie die Vorbereitungshilfe für diesen Test.
Die Kartei ist ein einfaches GridBagLayout welches die Frage und die möglichen Antworten von denen eine ausgewählt werden soll anzeigt. Es wird noch ein eventuell vorhandenes Bild zur Frage gezeigt. Nach dem Auswählen einer Antwort wird gezeigt ob diese richtig ist und dazu eine Erklärung die helfen soll die richtige Antwort zu finden. Die zwei Schaltflächen "vorherige Frage" und "nächste Frage" werden erst durch diese Aktion enabled.
Die Fragen und Antworten, sowie Verweise auf Bilder werden in einer externen Textdatei "fragen.txt", geschrieben.

<html>1. Frage1
zweite Zeile der Frage1</html>
Antwort1 1
Antwort2 1
Antwort3 1
Antwort4 1
kA
weil 1 ...
img1.gif
2. Frage2
Antwort1 2
Antwort2 2
Antwort3 2
Antwort4 2
3
weil 2 ...

3. Frage3
Antwort1 3
Antwort2 3
Antwort3 3

Antwort4 3
3
weil 3 ...
img1R.gif

Ist für eine Frage kein Bild / Schema vorgesehen so wird in der Zeile nur ein Leerzeichen eingegeben. Für jede Frage sind 4 Antwortmöglichkeiten anzugeben. Danach kommt die Nummer der richtigen Antwort und in der nächsten Zeile die Begründung. Sollte keine der Antwortmöglichkeiten richtig sein muss in der Zeile kA geschrieben werden. Dann folgt der Name des Bildes das gezeigt werden soll. Das Bild muss in GIF Format hinterlegt sein und sich im Unterordner qS des root Ordners vom Applet befinden.
Die erste Frage ist hier mit HTML Tags versehen wodurch das Formatieren ermöglicht wird.
Die Liste qAa beinhaltet jetzt alle Daten die für die Darstellung des Eingangstests notwendig sind.
Das Erstellen der Datei „fragen.txt“ Liegt in Verantwortung von dem Betreuer der Übung.

4.2 Übungstabs

Bei allen Übungstabs ist die Bedienung gleich und zwar:

Zuerst werden alle Verbindungen hergestellt, danach wird mit dem Klick auf die Schaltfläche „Überprüfen“ überprüft ob die Verbindungen richtig sind. Falls alles richtig verbunden ist, werden alle Elemente aktiviert. Mit dem Betätigen verschiedener Elemente lässt sich ein Wert einstellen. Mit dem Klick auf die Schaltfläche „Protokoll“ werden die Werte allen Elementen die einen Wert haben in das Protokoll übernommen. Falls die Verbindungen nicht richtig waren kann man sie entweder einzeln oder mit dem Klick auf die Schaufläche „Alle Verbindungen löschen“ alle auf einmal löschen.

Die Verbindung wird erstellt indem man auf die Anfangs- und Endklemme mit der linken Maustaste klickt. Gelöscht werden die Verbindungen dann auf gleiche Weise, nur mit dem rechten Mausklick.

4.2.1 Verbindungsliste (CL)

Im CL Unterverzeichnis des Applets befinden sich die Dateien in denen die Verbindungen für die einzelnen Teilübungen definiert sind. Für jede Teilübung ist eine eigene CL-Datei vorhanden. Jede Datei ist nach der Teilübung benannt wie z.B. FEGAeussere_Kennlinie.cl für die Teilübung zum Messen der äußeren Kennlinien beim fremderregten Generator.

M::A1 - M::A2

U::P - M::B2

===

U::P - M::B2

M::D1 - M::B2

===

U::P - M::B2

Hier ist der Inhalt einer Datei angezeigt. Der Inhalt ist zwar nicht realitätsbezogen dient aber nur als Beispiel. Eine Verbindung schreibt man folgendermaßen:

Begonnen wird mit dem Namen des Startelementes gefolgt von :: (zwei :) und dem Namen der Klemme. Dann kommt ein Leerzeichen, dem ein – (Bindestrich) folgt, dann wieder ein Leerzeichen und dann das Endelement mit seiner Klemme. Die Namen der Elemente und die Klemmenbezeichnungen sind im Anhang angegeben. Eine Verbindung befindet sich in einer Zeile. Sollte es für eine Teilübung mehrere Möglichkeiten zur Verbindung der Elemente geben so werden sie mit === (drei =) in einer Zeile getrennt. Das Erstellen von CL-Dateien ist die Aufgabe des Betreuers.

4.3 Abschluss

Beim Abschluss werden nur ein Textfeld, in welchem der Student seine Matrikelnummer eingeben soll und eine Schaltfläche zum Absenden der E-Mail angezeigt.

5 Hilfe

Hilfe zur virtuellen Übungsdurchführung ist in HTML realisiert. Für jede Teilübung gibt es zwei HTML Dateien. Eine, *_B.html, wird im Hilfepanel des Applets angezeigt, die Andere, *.html, wird nach dem Klick auf die Schaltfläche „Hilfe extern zeigen“ in einem neuen Fenster geöffnet. (* stetht für internen Namen der Teilübung.) In der Datei *_B.html ist nur grob beschrieben was das Ziel der Teilübung ist. In der Datei *.html ist dann die Teilübung detailliert beschrieben, mit dem Schaltplan der zu realisierenden Schaltung. Alle html Dateien sind im Verzeichnis „Help“. Die Schaltpläne sind im Unterverzeichnis „Schema“ als gif Dateien abgelegt. Weitere Graphikdatein sind im Unterverzeichnis „graph“.

Das Erstellen von Hilfedateien sowie der Schemen und Grafiken ist wieder ein Teil der zur Bedienung seitens des Betreuers anfällt.

In Folgenden führen wir die Hilfedateien an: (Durch das Kopieren aus den HTML-Dateien sind manche Formatierungen verlorengegangen.)

5.1 FEGAeussere Kennlinie.html

$U_A = f(I_A)$, I_E = const. , $n = 1500\ \text{min}^{-1}$. Hier wird zunächst die Ankerspannung ohne Belastung, also Ankerstrom $I_A = 0$, eingestellt. Anschließend wird dieser langsam erhöht und die Ankerspannung U_A dabei aufgenommen. Der Erregerstrom neigt dazu sich zu ändern und muss daher bei jeder Veränderung des Ankerstromes auf den Wert von 0,55 A nachgestellt werden. Die Schaltung ist die gleiche wie für die Aufnahme der Belastungskennlinie.

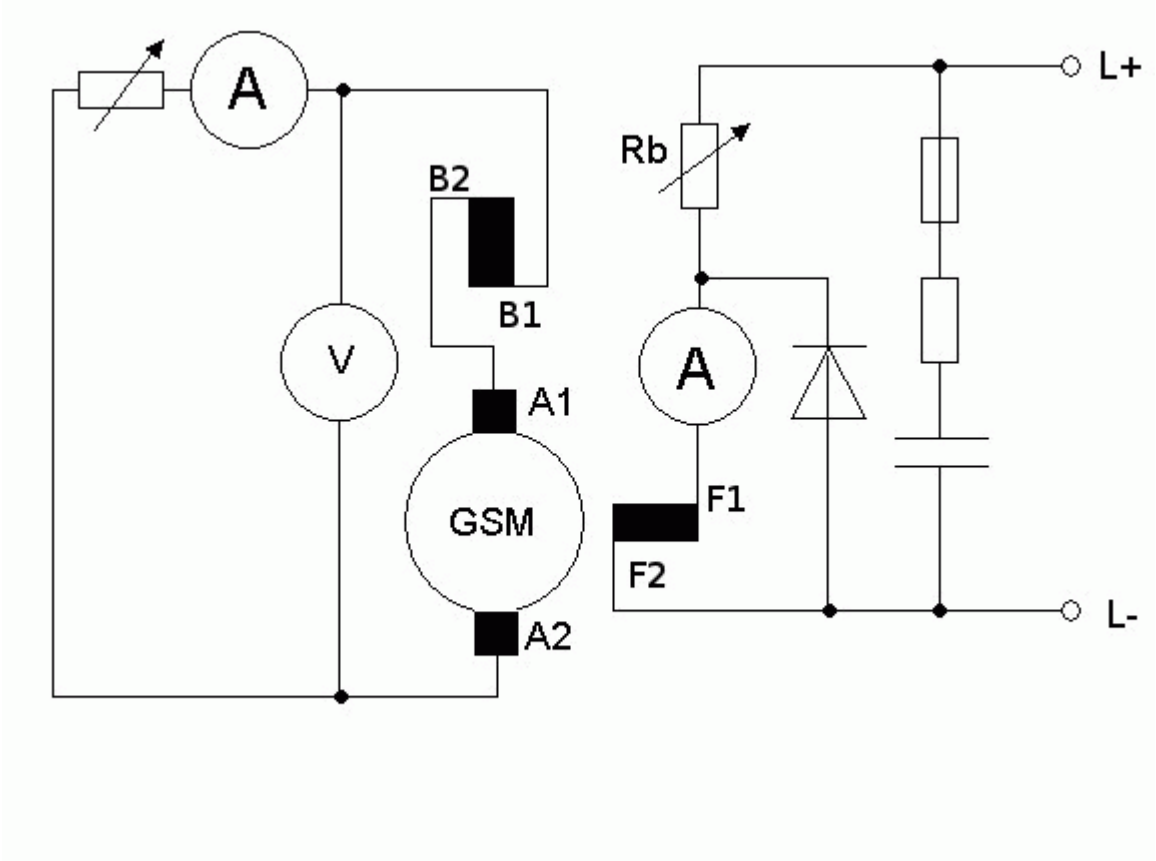


Bild.5.1: Fremderregter Generator

5.2 FEGAeussere_Kennlinie_B.html

Die äußere Kennlinie ist die Abhängigkeit der Ankerspannung vom Ankerstrom, bei konstantem Erregerstrom, $U_A = f(I_A)$, I_E = konst., n = konst.. Generator wird angetrieben und n auf Nenndrehzahl von 1500 U/min eingestellt. Ohne Belastung wird die Ankerspannung auf 250 V eingestellt. Dann wird die Maschine belastet auf. $I_A = 18$ A. Die Belastung wird danach reduziert und die Kennlinie aufgenommen.

5.3 FEGBelastungskennlinie.html

Zur Bestimmung der Belastungskennlinie ist die folgende Schaltung zu realisieren.

Hier erscheint das Bild.5.1 aus der Datei FEGAeussere Kennlinie.html.

Der Ankerkreis ist bei diesem Versuch zu schließen.Ferner wird der Erregerstrom I_E auf I_{E0}= 580 mA eingestellt um eine Ankerspannung von 250 V zu erhalten. Danach wird der Erregerstrom langsam verringert und dabei die U_A= f(I_E) – Kennlinie ermittelt. Die innere Spannung U_i wird dabei rechnerisch nach der Formel:

$U_i = U_A + I_A(R_A + R_W) + \Delta U_{Bürste} \approx U_A + I_A * 0{,}797\ \Omega$

ermittelt. Dabei ist $R_A + R_W$= 0,797 (Widerstand von Anker- und Wendepolwicklung bei 20°C). Die Bürstenspannung $\Delta U_{Bürste}$ beträgt 2 V, wird aber bei der Ermittlung der inneren Spannung vernachlässigt.

5.4 FEGBelastungskennlinie_B.html

Die Belastungskennlinie ist die Abhängigkeit der Ankerspannung vom Erregerstrom bei konstantem Ankerstrom und konstanter Drehzahl. Wieder wird der Generator mit dem Schragemotor angetrieben und die Drehzahl *n* auf 1500 U/min eingestellt. Ohne Belastung wird die Erregung so eingestellt, dass die Ankerspannung 250 V beträgt. Dann wird der Generator belastet damit. I_A= 18 A beträgt. Die Erregung wird dann reduziert und dabei U_A festgehalten. Aus der Messung wird dann die innere Kennlinie berechnet.

5.5 FEGLeerlaufkennlinie.html

Bei der Aufnahme der Leerlaufkennlinie ist folgende Schaltung zu realisieren. Der Gleichstromgenerator wird dabei von einem Schragemotor angetrieben. Auf der Schragemotorseite ist alles am Messplatz schon fest verdrahtet.

5.6 FEGLeerlaufkennlinie_B.html

Die Leerlaufkennlinie ist die Abhängigkeit der induzierten Ankerspannung vom Erregerstrom bei offenem Ankerkreis und konstanter Drehzahl. Die Messungen werden bei steigendem und bei fallendem Erregerstrom gemacht. Dies hat das Ziel die Hysteresis zu zeigen. Bei der Übung wird die Drehzahl mittels Schragemaschine auf Nenndrehzahl von 1500 U/min eingestellt. Dann wird der I_E erhöht und dabei die U_A festgehalten. Die Drehzahl wird auch ständig nachreguliert.

5.7 FEGLlkschaltung.html

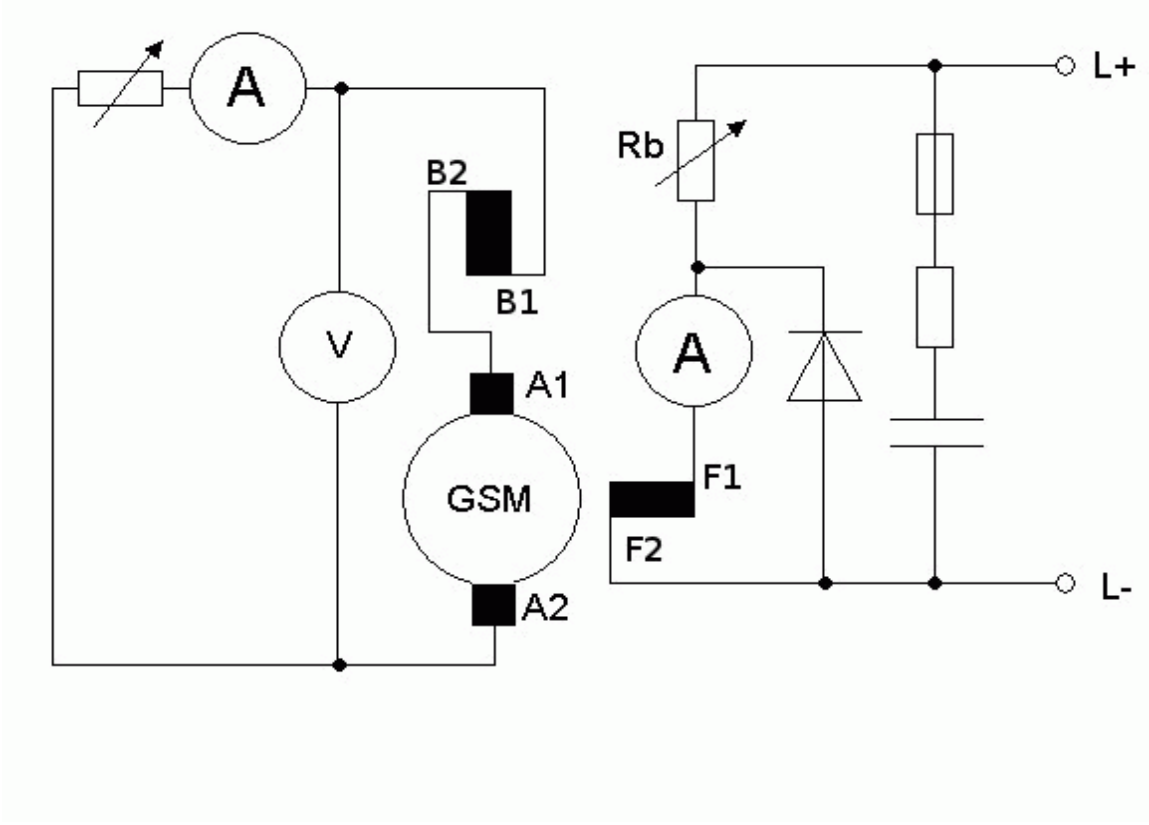


Bild.5.2: Schaltbild des fremderregten Generator

5.8 Schrage.html

Als Schrage Maschine bzw. Schrage Motor wird der läufergespeiste Drehstrom-Nebenschluss-Kommutatormotor bezeichnet. Sie ist eine Drehstrom-Kommutatormaschine deren Läufe sowohl eine Schleifriingwicklung als auch eine Kommutatorwicklung trägt. Die Schleifringwicklung liegt am Speisenden Primärnetz, und die Ständerwicklung ist über die Kommutatorwicklung kurzgeschlossen. Letzte prägt der Ständerwicklung eine durch Bürstenverstellung veränderbare, nahezu lastunabhängige Zusatzspannung auf und wird deshalb auch Regelwicklung genannt. Der Bürstensatz ist ein Sechsbürstensatz, an den die Ständerwicklung in offener Schaltung angeschlossen ist.

Diese Maschine arbeitet somit wie ein Dreiphasen-Induktionsmotor, dessen Läufer über dem konstruktivintegrieren Frequenzwandler eine lastunabhängige schlupffrequente Zusatzspannung zur Drehzahlstellung und zur Verbesserung des Leistungsfaktors aufgeprägt erhält. Die Drehzahl-Drehmoment-Kennlinien ähneln denen des Dreiphasen-Induktionsmotors, insbesondere bezüglich des Nebenschlussverhaltens im normalen Arbeitsbereich.

Sie lassen sich durch Betätigen einer Bürßtenverstelleinrichtung in einem gewissen Bereich (Drehzahlstellbereich bis 1:3 und darüber) verschieben, sodass eine wirksame Drehzahlstellung möglich ist.

5.9 FEGRegulierkennlinie.html

Die Regulierkennlinie ist die Abhängigkeit des Erregerstroms vom Ankerstrom bei konstanter Ankerspannung und konstanter Drehzahl.

Die Schaltung zur Aufnahme von der Regulierkennlinie ist gleich wie bei der Aufnahme der Belastungskennlinie.

Hier erscheint das Bild.5.1 aus der Datei FEGAeussere Kennlinie.html.

Der Generator wird angetrieben und n auf 1500 U/min eingestellt. Ohne Belastung wird die Ankerspannung auf 250 V eingestellt. Dann wird die Maschine belastet. Die Belastung wird reduziert. Gleichzeitig wird I_E nachgestellt damit U_A konstant bleibt. Die Messung von I_E ergibt dann die Kennlinie.

5.10 FEGRegulierkennlinie_B.html

Die Regulierkennlinie ist die Abhängigkeit des Erregerstroms vom Ankerstrom bei konstanter Ankerspannung und konstanter Drehzahl. Der Generator wird angetrieben und n auf 1500 U/min eingestellt. Ohne Belastung wird die Ankerspannung auf 250 V eingestellt. Dann wird die Maschine belastet und die Belastung schrittweise reduziert. Gleichzeitig wird I_E nachgestellt damit U_A konstant bleibt.

5.11 NSGAeussere Kennlinie.html

Beim Nebenschlussgenerator wird der Erregerstrom vom Generator selbst geliefert und mittels eines Vorwiderstandes R_V eingestellt. Der Anlauf ohne externe Spannung U_A ist nur mit Hilfe einer Remanenz möglich.

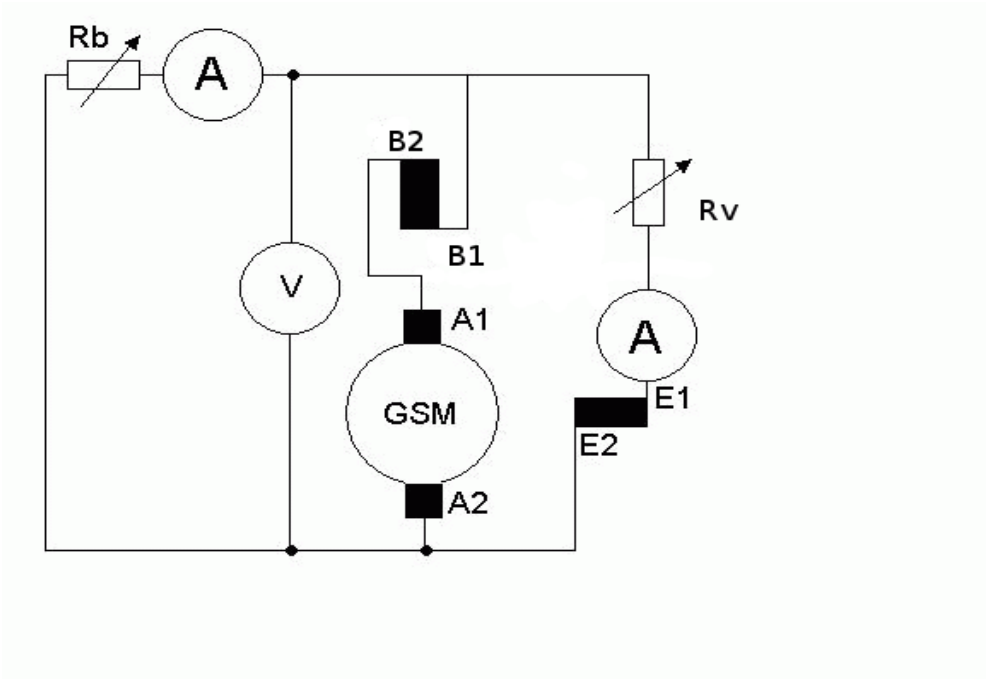


Bild.5.3: Nebenschlussgenerator

5.12 NSGAeussere_Kennlinie_B.html

Für die Messung wird die Schaltung mit einem Belastungswiderstand realisieret. Dabei werden der Ankerstrom, der Erregerstrom, die Ankerspannung und die Erregerspannung gemessen. Der Generator wird vom Motor angetrieben und der Erregerstrom ohne Belastung mittels Vorwiderstand eingestellt. Durch Belastungswiderstand wird dann Nennstrom in der Ankerwicklung hervorgerufen. Die Belastung wird dann verringert wobei die Drehzahl konstant gehalten wird.

5.13 NSGEntregung.html

Für die Messung wird die Schaltung mit einem Belastungswiderstand verwendet.

Hier erscheint das Bild.5.3 aus der Datei NSGAeussere Kennlinie.html.

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt, die dadurch erhöhte Ankerspannung erhöht wieder den Erregerstrom und es entsteht ein stabiles Gleichgewicht im Schnittpunkt der Kennlinie $U_i = f(I_E)$ und der Widerstandsgeraden $(R_E+R_V)*I_E = f(I_E)$.
Bei falscher Polung der Erregerwicklung wird das Feld geschwächt und es kommt zu einer Entregung!

5.14 NSGEntregung_B.html

Gemessen werden der Ankerstrom, der Erregerstrom, die Ankerspannung (am Belastungswiderstand) und die Erregerspannung. Der Generator wird vom Motor angetrieben und die Nenndrehzahl eingestellt.

5.15 NSGSelbsterregung.html

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt. Dadurch erhöht sich die Ankerspannung die dann den Erregerstrom erhöht.

Hier erscheint das Bild.5.3 aus der Datei NSGAeussere Kennlinie.html.

5.16 NSGSelbsterregung_B.html

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt. Dadurch erhöht sich die Ankerspannung die dann den Erregerstrom erhöht.

5.17 NZBallistische Methode.html

Wenn sich die Bürsten in der neutralen Zone befinden sind die Ankerinduktivität und die Induktivität der Erregerwicklung magnetisch nicht gekoppelt, da die Richtungen vom Ankerfluss, und vom Hauptfluss, normal aufeinander sind. Die magnetische Kopplung ist umso größer, je weiter die Bürsten aus der neutralen Zone gedreht sind. Wird nun der Erregerstrom eingeschaltet, so wird in der Ankerwicklung ein Spannungsimpuls induziert. Durch das Verdrehen der Bürsten werden die Höhe und das Vorzeichen des Spannungsimpulses verändert. Wenn der Spannungsimpuls minimal wird, ist die neutrale Zone erreicht.

Hier erscheint das Bild aus dem Abschnitt Übungsdurchführung/Einstellung der Bürsten in die neutrale Zone/Ballistische Methode/Theorie.

Um diesen Impuls zu messen wird ein Oszilloskop an die Maschine angeschlossen. Der Erregerstrom wird über einen Schalter am Erregerkreis eingespeist. Der Schalter wird nur kurzzeitig ein- und dann wieder ausgeschaltet. Dabei ist am Oszilloskop der induzierte Spannungsimpuls des Ankerkreises sichtbar. Die Bürsten werden so lange gedreht bis der Impuls minimal wird. Die Messung ist wieder in beiden Drehrichtungen durchzuführen.

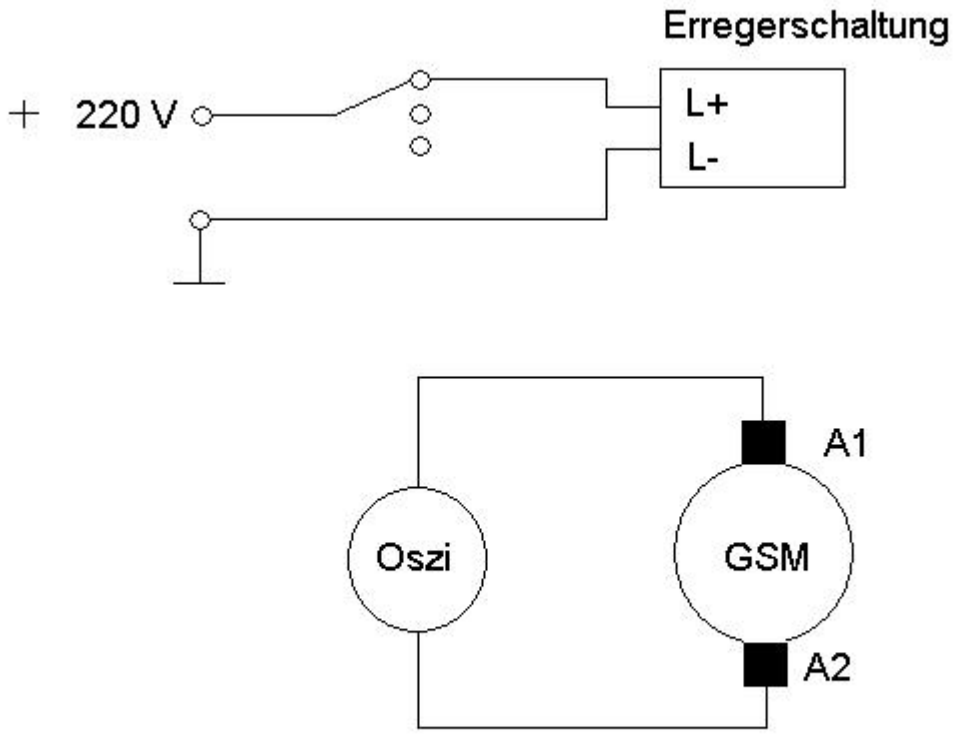


Bild 5.4: Ballistische Methode

5.18 NZBallistische_Methode_B.html

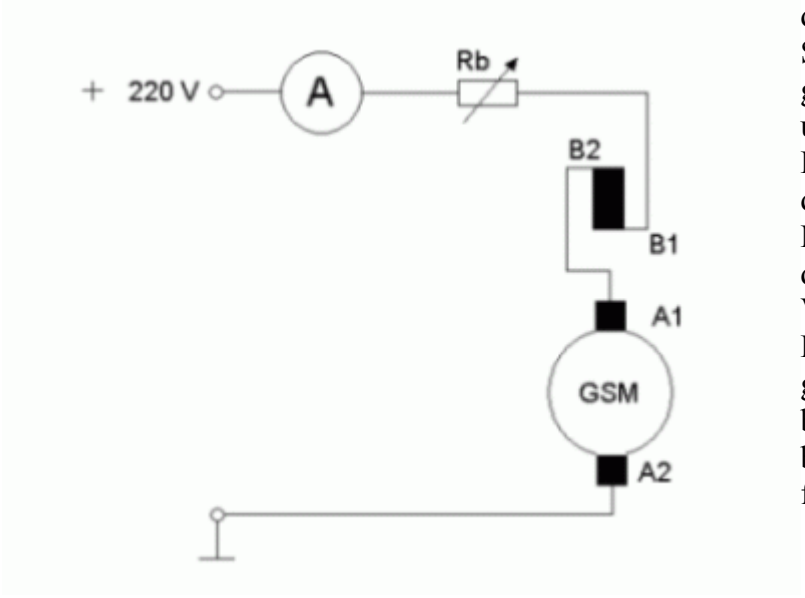
Wenn sich die Bürsten in der neutralen Zone befinden sind die Ankerinduktivität und die Induktivität der Erregerwicklung magnetisch nicht gekoppelt. Die magnetische Kopplung ist umso größer, je weiter die Bürsten aus der neutralen Zone gedreht sind. Wird nun der Erregerstrom eingeschaltet, so wird in der Ankerwicklung ein Spannungsimpuls induziert. Durch das Verdrehen der Bürsten werden die Höhe und das Vorzeichen des Spannungsimpulses verändert. Wenn der Spannungsimpuls minimal wird, ist die neutrale Zone erreicht. Um diesen Impuls zu messen wird ein Oszilloskop an die Maschine angeschlossen. Der Erregerstrom wird über einen Schalter am Erregerkreis eingespeist. Der Schalter wird nur kurzzeitig ein- und dann wieder ausgeschaltet. Dabei ist am Oszilloskop der induzierte Spannungsimpuls des Ankerkreises sichtbar. Die Bürsten werden so lange gedreht bis der Impuls minimal wird. Die Messung ist wieder in beiden Drehrichtungen durchzuführen.

5.19 NZMomentenmethode.html

Wird die Ankerwicklung bei einer offenen Erregerwicklung von einem Strom durchflossen, so bildet sich ein Ankerfeld aus, welches sich über Wendepole und das Ständerjoch bzw. die Hauptpole schließt. Wenn sich die Bürsten nicht in der neutralen Zone befinden, erhält das Ankerfeld eine Komponente in Richtung des Hauptfeldes und es entsteht ein Drehmoment.

Hier erscheint das Bild aus dem Abschnitt Übungsdurchführung/Einstellung der Bürsten in die neutrale Zone/Momentenmethode/Theorie

Damit eine Drehung des Rotors möglich ist, muss zuerst das Reibungsmoment überwunden werden. Deshalb werden Versuche für beide Drehrichtungen der Maschine gemacht. Die neutrale Zone wird dann durch Mittelung der beiden Ergebnisse ermittelt. Um die Messung durchzuführen ist folgende Schaltung zu realisieren. Die ganze Messung ist einmal mit und einmal ohne Kompensationswicklungen durchzuführen. Bei der Messung wird der Nennstrom der Maschine (18.3 A) über Vorwiderstand eingestellt. Danach werden die Bürsten gedreht bis der Rotor zu drehen beginnt. Dieser Punkt ist für beide Drehrichtungen festzuhalten.



5.20 NZMomentenmethode_B.html

Wird die Ankerwicklung bei einer offenen Erregerwicklung von einem Strom durchflossen, so bildet sich ein Ankerfeld aus, welches sich über Wendepole und das Ständerjoch bzw. die Hauptpole schließt. Wenn sich die Bürsten nicht in der neutralen Zone befinden erhält das Ankerfeld eine Komponente in Richtung des Hauptfeldes und es entsteht ein Drehmoment. Damit eine Drehung des Rotors möglich ist, muss zuerst das Reibungsmoment überwunden werden. Deshalb werden Versuche für beide Drehrichtungen der Maschine gemacht. Die neutrale Zone wird dann durch Mittelung der beiden Ergebnisse ermittelt. Die ganze Messung ist einmal mit und einmal ohne Kompensationswicklungen durchzuführen. Bei der Messung wird der Nennstrom der Maschine (18.3 A) über Vorwiderstand eingestellt. Danach werden die Bürsten gedreht bis sich der Rotor zu drehen beginnt. Dieser Punkt ist für beide Drehrichtungen festzuhalten.

5.21 VGAeussere Kennlinie.html

Beim Gleichstrom – Verbundgenerator, was ja eigentlich ein Nebenschlussgenerator mit Kompoundwicklung ist, wird eine Reihenschlusswicklung zusätzlich zu einer Nebenschlusswicklung in Serie geschaltet. Diese hat den Zweck das Hauptfeld, gesteuert durch den Ankerstrom I_A zu verstärken, und dadurch die Ankerrückwirkung zu minimieren. Die zu realisierende Schaltung:

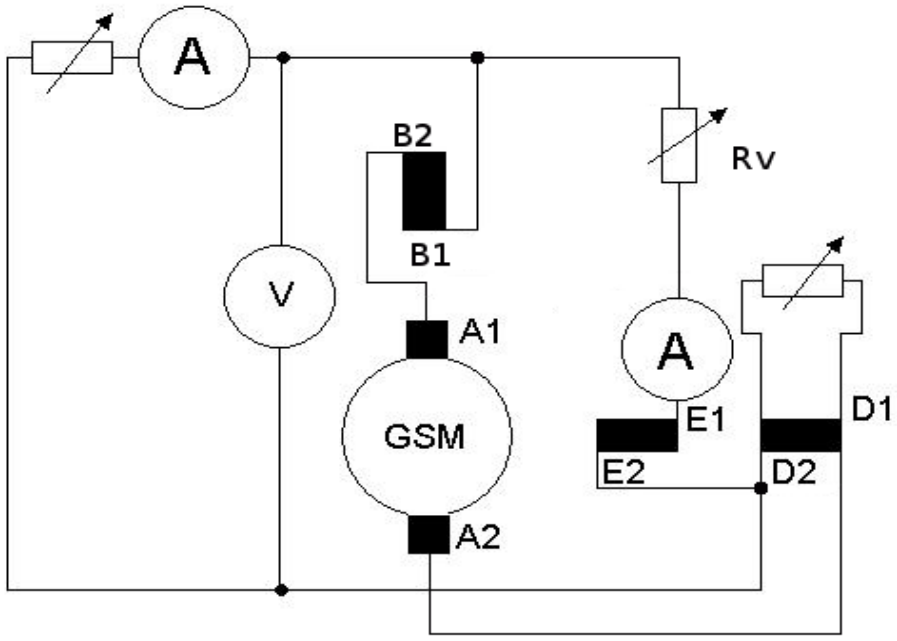


Bild 5.5: Verbundgenerator

5.22 VGAeussere_Kennlinie_B.html

Beim Gleichstrom – Verbundgenerator, was ja eigentlich ein Nebenschlussgenerator mit Kompoundwicklung ist, wird eine Reihenschlusswicklung zusätzlich zu einer Nebenschlusswicklung in Serie geschaltet. Diese hat den Zweck das Hauptfeld gesteuert durch den Ankerstrom I_A zu verstärken, und dadurch die Ankerrückwirkung zu minimieren.

5.23 VGEntregung.html

Für die Messung wird die Schaltung mit einem Belastungswiderstand verwendt.

Hier erscheint Bild 5.5 aus der Datei VGAeussere Kennlinie.html.

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt, die dadurch erhöhte Ankerspannung erhöht wieder den Erregerstrom und es entsteht ein stabiles Gleichgewicht im Schnittpunkt der Kennlinie $U_i = f(I_E)$ und der Widerstandsgeraden $(R_E+R_V)*I_E = f(I_E)$.
Bei falscher Polung der Erregerwicklung wird das Feld geschwächt und es kommt zu einer Entregung!

5.24 VGEntregung_B.html

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt, die dadurch erhöhte Ankerspannung erhöht wieder den Erregerstrom und es entsteht ein stabiles Gleichgewicht im Schnittpunkt der Kennlinie $U_i = f(I_E)$ und der Widerstandsgeraden $(R_E+R_V)*I_E = f(I_E)$.
Bei falscher Polung der Erregerwicklung wird das Feld geschwächt und es kommt zu einer Entregung!

5.25 VGSelbsterregung.html

Für die Messung wird die Schaltung mit einem Belastungswiderstand verwendt.

Hier erscheint Bild 5.5 aus der Datei VGAeussere Kennlinie.html.

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt, die dadurch erhöhte Ankerspannung erhöht wieder den Erregerstrom und es entsteht ein stabiles Gleichgewicht im Schnittpunkt der Kennlinie $U_i = f(I_E)$ und der Widerstandsgeraden $(R_E+R_V)*I_E = f(I_E)$.
Bei falscher Polung der Erregerwicklung wird das Feld geschwächt und es kommt zu einer Entregung!

5.26 VGSelbsterregung_B.html

Schaltet man die Erregerwicklung an die Klemmen des Ankerkreises, so wird sich, bei Vorhandensein einer kleinen Anfangsspannung durch Remanenz, ein zunächst sehr kleiner Erregerstrom einstellen. Durch diesen Erregerstrom wird das Magnetfeld verstärkt, die dadurch erhöhte Ankerspannung erhöht wieder den Erregerstrom und es entsteht ein stabiles Gleichgewicht im Schnittpunkt der Kennlinie $U_i = f(I_E)$ und der Widerstandsgeraden $(R_E+R_V)*I_E = f(I_E)$.
Bei falscher Polung der Erregerwicklung wird das Feld geschwächt und es kommt zu einer Entregung!

6 Verwendete Programme

- Entwicklung:
  - java 1.6.0_03
  - eclipse 3.2

- HTML (Hilfe System)
  - SeaMonkey 1.1.6

- Graphik:
  - IrfanView 3.92
  - gimp 2.4.2

7 Literaturverzeichnis

Skriptum:
Grundlagen der Elektrotechnik III
TU Berlin - Institut für Energie- und Automatisierungstechnik
R. Hanitsch

Skriptum:
Antriebe
TU Wien – Institut für Elektrische Maschinen und Antriebe
H. Kleinrath

Skriptum:
Skriptum zur Übung
Labor Elektrische Antriebe
TU Wien – Institut für Elektrische Antriebe und Maschinen

Web-Seiten:

Overview (Java 2 Platform SE v1.4.2)
http://java.sun.com/j2se/1.4.2/docs/api/overview-summary.html

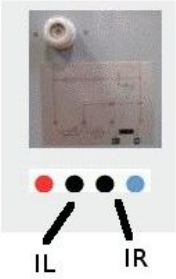
Java(TM) Programmierhandbuch und Referenz
http://www.dpunkt.de/java/index.html

Anhang

Verbindungen

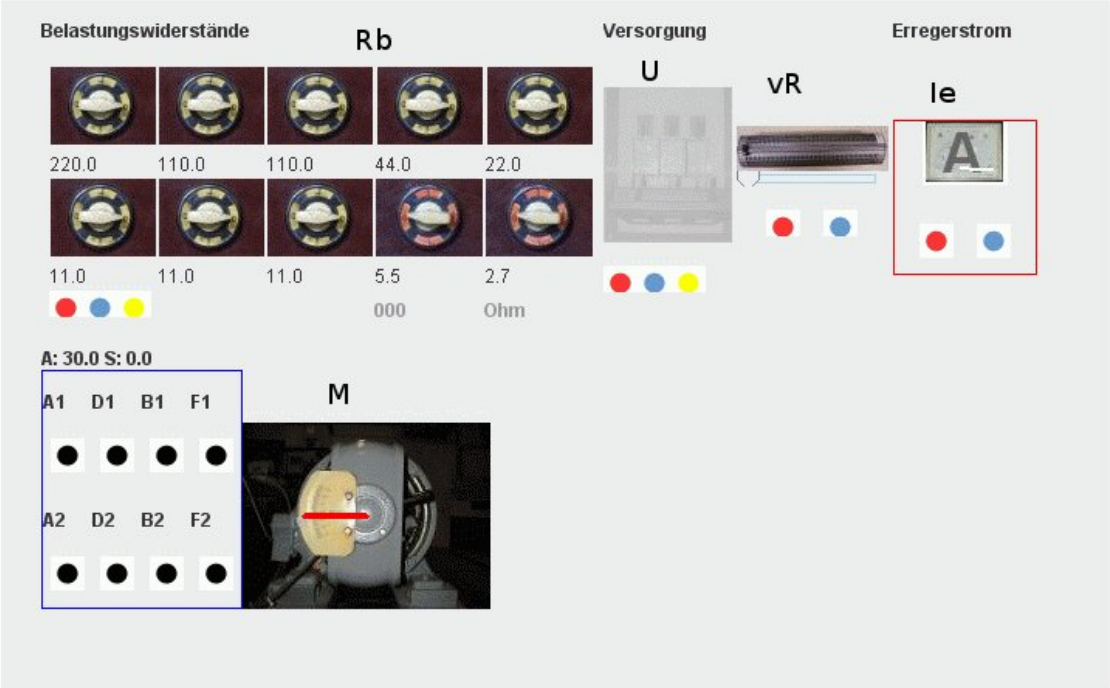
Die Verbindung lautet immer
ELEMENT_START::KLEMME – ELEMENT_ENDE::KLEMME
Bsp: U::P – M::B2

Bei Klemmen gilt;
ROT – P; BLAU – M; GELB – E
Erregerkreis wie im Bild unten

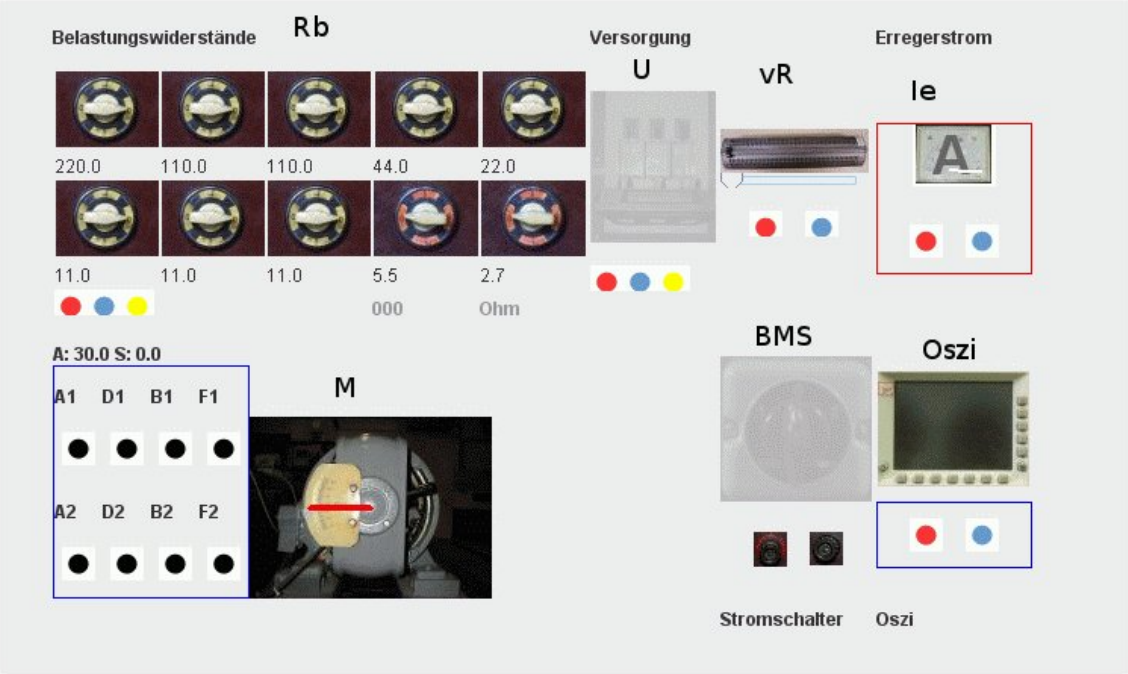


Bei der Maschine sind die Klemmen beschriftet.

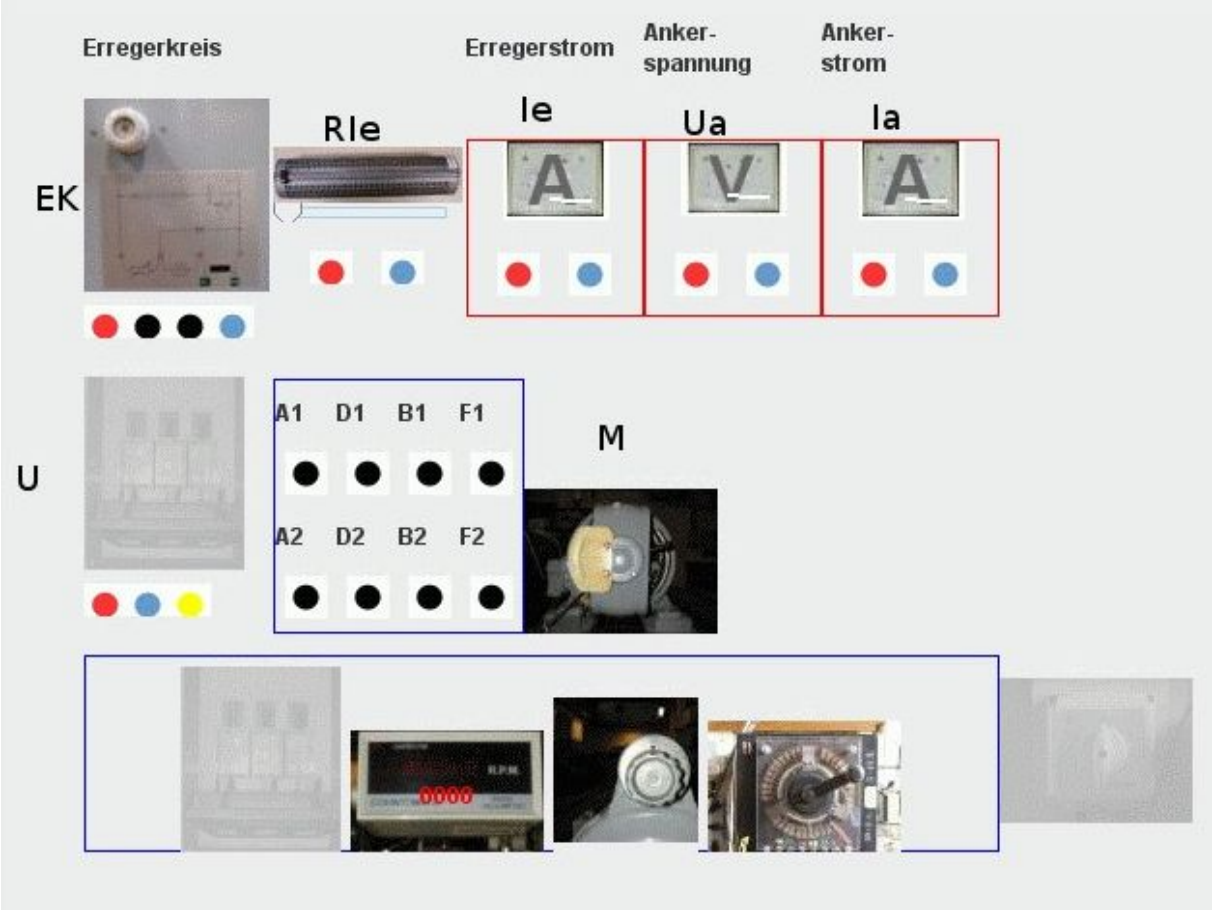
Neutrale Zone Momentenmethode



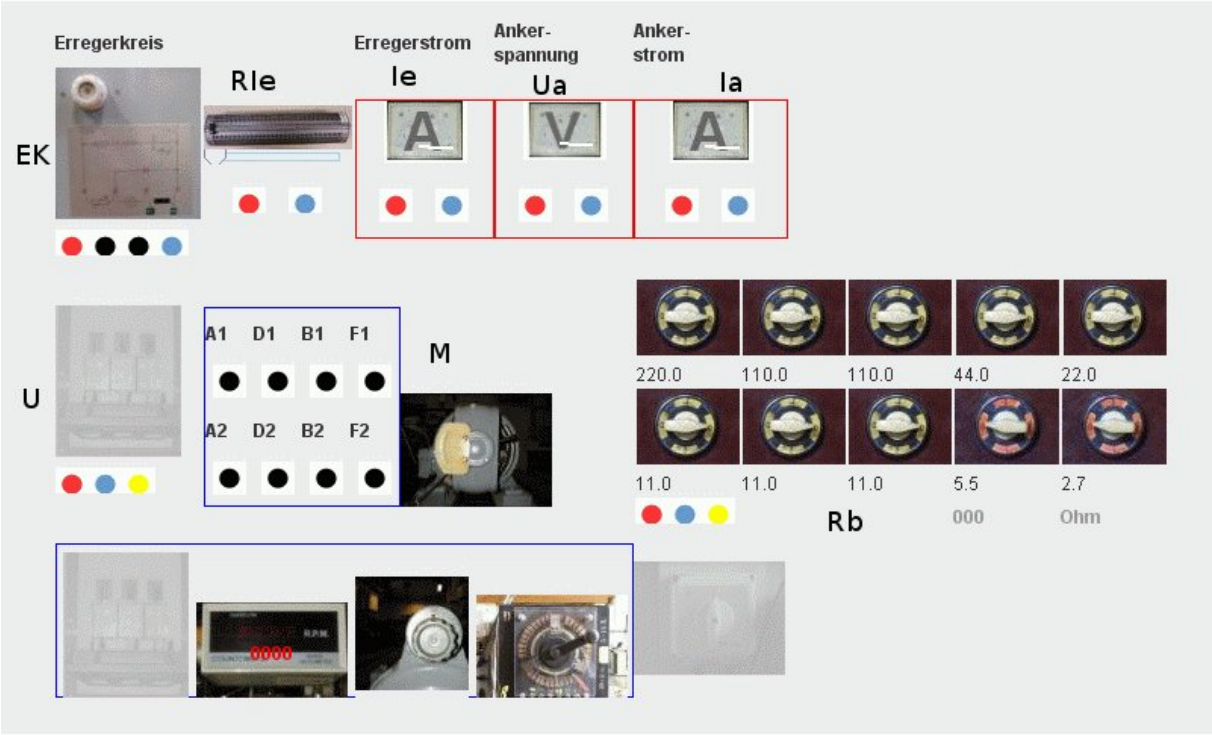
Neutrale Zone ballistische Methode



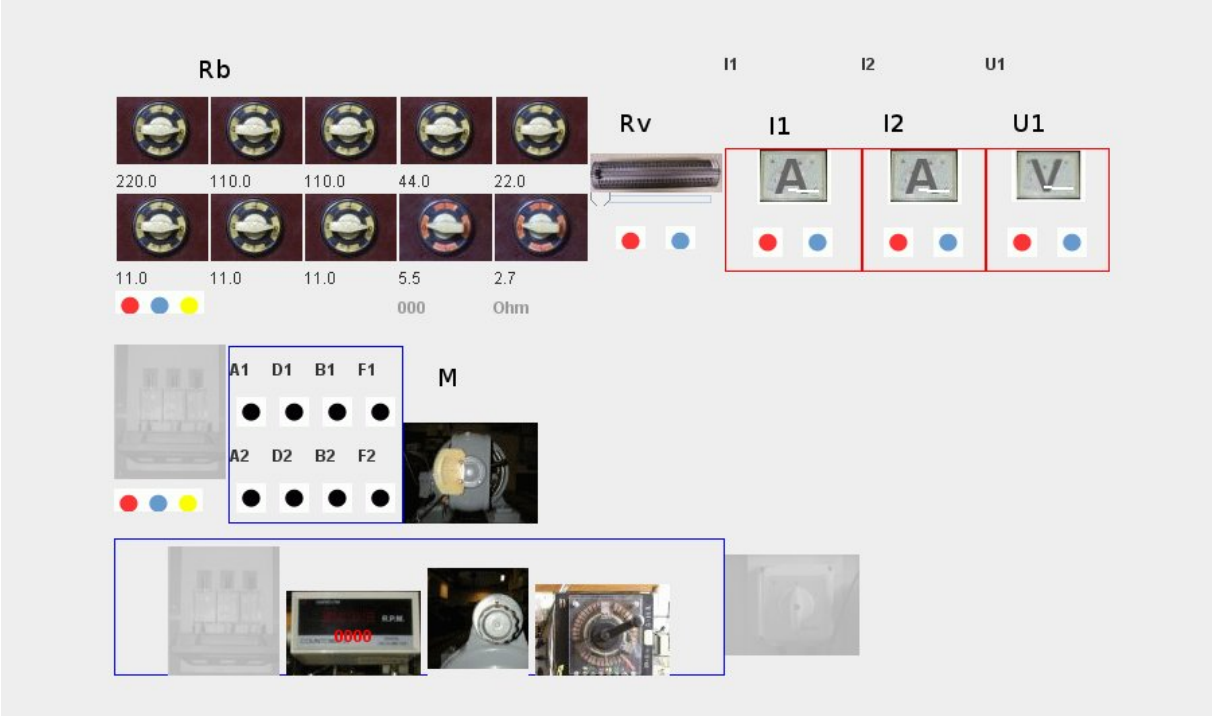
Fremderregter Generator Leerlaufkennlinie



Fremderregter Generator Belastungs-, Regulierkennlinie und äußere Kennlinie



Nebenschluss- und Verbundgenerator



Schragmaschine und dazugehörige Elemente die nicht verbunden werden sind nicht beschriftet..

Verwendete Bilder

Ampermeter.gif

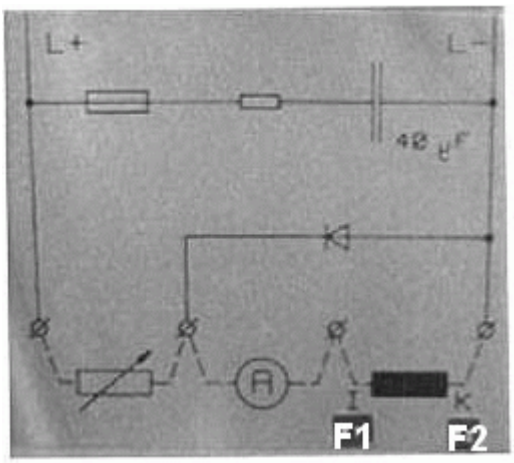
Ampermeter

Voltmeter.gif

Voltmeter

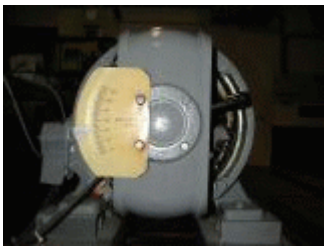
ErrKreis4.gif

Erregerkreis



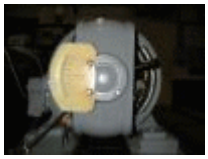
ErrKreis4S.gif

Schaltung des Erregerkreises



GMFront160.gif

Gleichstrommaschine gross



GMFront100.gif

Gleichstrommaschine klein

Klemme0.gif

Klemme

Klemme Erdung	MWAKlemmeE.gif
 Klemme minus	MWAKlemmeM.gif
 Klemme plus	MWAKlemmeP.gif
 Klemme für die neutrale Zone ballistische Methode minus	NZBMKlemmeM2.gif
 Klemme für die neutrale Zone ballistische Methode plus	NZBMKlemmeP2.gif
 Neutrale Zone ballistische Methode Schalter aus	NZBMSchalter0.gif
 Neutrale Zone ballistische Methode Schalter ein	NZBMSchalter1.gif
 Widerstand aus	Off.gif
 Widerstand ein	On.gif
 Widerstand aus	Roff.gif

Ron.gif

Widerstand ein

Ra1002.gif

Erreger Widerstand

Oszi0.gif

Oszilloskop

sm.gif

Schrage Maschine

SMOff.gif

Schrage Maschine Schalter aus

SMOn.gif

Schrage Maschine Schalter ein

smZaehler.gif

Schrage Maschine Drehzahlzähler

SourceOff2.gif

Stromversorgung aus

SourceOn2.gif

Stromversorgung ein

Bilder 1.gif bis 12.gif in Ordner SMS

Schrage Maschine rund Schalter