Projektarbeit 1. Lehrjahr

Henning Meyer
Erik Bauer

9. April 2019

Inhaltsverzeichnis

1 Analyse und Vorüberlegungen

1.1 IST-Analyse

1.1.1 Situation des Auftraggebers

Der Auftraggeber für dieses Projekt ist die technische Berufsschule Rostock. Die Schule besitzt mehrere Laborräume, in denen Computer mit gängiger Peripherie wie Maus und Tastatur vorhanden sind. Diese werden in gegebenen Unterrichtsstunden den Schülern zum Arbeiten zur Verfügung gestellt. Auf den Computern befinden sich u.a. Entwicklungsumgebungen für verschiedene Programmiersprachen, darunter die für dieses Projekt relevante Software „Microsoft Visual Studio“ in verschiedenen Versionen. Die Rechner verfügen ebenfalls über einen Internetzugang für Recherchezwecke, Anbindung an das lokale Schulnetzwerk mit Speicherplatz für jeden Schüler, sowie Schutzsoftware, um die Funktionalität der lokalen PCs zu gewährleisten.

1.1.2 Situation des Auftragsnehmers

Der Auftragsnehmer ist in Form eines Projektteams bestehend aus zwei Schülern der Klasse FIN81 gegeben. Beide absolvieren das erste Lehrjahr der Ausbildung zum Fachinformatiker für Anwendungsentwicklung. Es bestehen auf beiden Seiten Vorkenntnisse in der Programmierung in unterschiedlichen Programmiersprachen.

Zum Arbeiten besitzen die Teammitglieder eigene Computer, es stehen ihnen aber ggf. auch Rechner in ihren Arbeitsplätzen und in der Berufsschule zur Verfügung.

1.2 SOLL-Analyse

Als Anforderung an das Projektteam wurden Kriterien für ein Programm, sowie eine dazugehörige schriftliche Ausarbeitung gegeben. Die gewünschte Anwendung ist hierbei durch Muss-/ und Wunsch-Funktionalitäten beschrieben, konkrete Einschränkungen an die Umsetzung sind dabei aber nicht gemacht worden. Die Hauptaufgabe des Programms liegt in der Visualisierung von physikalischen Messdaten und deren Zusammenhänge in Form von dynamisch erstellten Graphen. Außerdem muss es zu Präsentationszwecken auf den Schulrechnern lauffähig sein.

1.3 Problemstellung

Es ist eine Beschreibung einer elektrotechnischen Schaltung gegeben. Die mathematischen Zusammenhänge zwischen den physikalischen Größen, sind vom Programm automatisch als Graph darzustellen. Speziell soll der Graph in einem begrenzten Maße ohne Nutzung fremden Quellcodes gezeichnet werden. Des Weiteren sind die Graphen nur als Näherungskurve darzustellen.

1.3.1 Begriffsdefinitionen

Spannung Die elektrische Spannung mit dem Formelzeichen U und der Einheit Volt (V), beschreibt die Eigenschaft, Ladung zu bewegen. Sie ist Ursache für das Fließen eines elektrischen Stroms innerhalb eines elektrisch leitfähigen Elementes.

Stromstärke Die elektrische Stromstärke besitzt das Formelzeichen I und die Einheit Ampere (A). Sie bemisst den elektrischen Strom im Bezug auf eine gegebene Fläche, zum Beispiel der Querschnittfläche eines Leiters oder Kondensators.

Widerstand Der elektrische Widerstand, welcher durch das Formelzeichen R und die Einheit Ohm (Ω) gekennzeichnet wird, gibt an, welche elektrische Spannung notwendig ist, um eine bestimmte elektrische Stromstärke durch einen elektrischen Leiter zu transportieren.

Leistung Das Formelzeichen P und die Einheit Watt (W) beschreiben die Leistung im generellen Sinne. Die elektrische Leistung beschreibt die Arbeit, welche ein elektrischer Strom in einer bestimmten Zeit verrichtet bzw. die Energie, welche in dieser Zeitspanne umgesetzt wird.

Spannungsanpassung Unter Spannungsanpassung versteht man die Optimierung der Beziehung zwischen elektrischer Spannungsquelle und elektrischem Verbraucher einer elektrischen Schaltung zugunsten der elektrischen Spannung. Es soll die maximale elektrische Spannung am Verbraucher erreicht werden. Dies geschieht, wenn der Widerstand außerhalb der Spannungsquelle größer ist, als innerhalb der Spannungsquelle.

Stromanapassung Unter Stromanpassung versteht man die Optimierung einer elektrischen Schaltung zugunsten des elektrischen Stroms, wobei die Beziehung zwischen elektrischer Spannungsquelle und elektrischen Verbraucher angepasst wird. Ziel ist, die maximale Stromstärke zu erreichen. Dazu muss der Widerstand innerhalb der Schaltung den Widerstand außerhalb der Schaltung überwiegen.

Leistungsanpassung Als Leistungsanpassung wird die Optimierung einer elektrischen Schaltung zugunsten der elektrischen Leistung bezeichnet. Der Widerstand der Spannungsquelle muss dabei dem Widerstand am Verbraucher gleichen, um die maximale Leistung zu erreichen.

1.3.2 Betrachtung mathematischer Zusammenhänge

Da die mathematischen Zusammenhänge klar definiert sind, lassen sich über die im Programm zu erwartenden Ergebnisse einige Vorüberlegungen treffen. Sofern zum Berechnen der Werte und zum Zeichnen des Graphen geeignete Algorithmen verwendet werden, sollten die produzierten Ausgaben den definierten Beziehungen entsprechen.

Gegeben sind eine Spannungsquelle U_0 von 10V, sowie ein veränderlicher Lastwiderstand R_L im Verbraucher L. Zu berechnen sind nun die Stromstärke I_L, die Stromspannung U_L, sowie die elektrische Leistung P_L in Abhängigkeit von R_L.

Die zu erwartenden Veränderungen der unterschiedlichen physikalischen Größen in Abhängigkeit von R_L lassen sich mit Hilfe ihrer Berechnungsformeln herleiten:

Berechnung von I_L:

$$I_L = \frac{U_0}{R_L}$$

$$I_L = \frac{10\text{V}}{R_L}$$

Berechnung von U_L:

$$U_L = R_L \cdot I_L$$

$$U_L = R_L \cdot \frac{10\text{V}}{R_L}$$

$$U_L = 10\text{V}$$

Berechnung von P_L:

$$P_L = U_L \cdot I_L$$

$$P_L = 10\text{V} \cdot \frac{10\text{V}}{R_L}$$

$$P_L = \frac{100\text{V}}{R_L}$$

1.3.3 Wertetabelle und grafische Darstellung im Programm

1.3.4 Leistungsanpassung am praktischen Beispiel

2 Lösungsfindung und Umsetzung

2.1 Betrachtung von Lösungsvarianten

Nach Analyse der gegebenen Anforderungen entschied das Projektteam, dass eine Entwicklungsumgebung, welche das Erstellen graphischer Oberflächen unterstützt, angemessen zur Lösung ist. Außerdem sollte die verwendete Programmiersprache eine einfache Schnittstelle oder Möglichkeit bieten, vom Programmierer definierte Grafiken auf dem Bildschirm anzuzeigen. Hierbei sollten speziell einzelne Punkte oder gerade Linien darstellbar sein, damit die Zeichnung von Kurven gut implementierbar ist. Als mögliche Programmiersprachen kamen *„Python“*, unter Nutzung von *„TKinter“*, *„Java“* und *„C#“*, sowie *„Javascript“* in Frage. Zusätzlich zur Seite der Problemstellung, wurde dann die Seite der Vorkenntnisse in Betracht gezogen. Mit den gefundenen Kriterien und den Überschneidungen vorhandener Entwicklungserfahrungen, sind sowohl *„Javascript“* als auch *„C#“* in die engere Auswahl gekommen. Nach dem abwägen möglicher Entwicklungsvorteile und Kompatibilitätskriterien, hat sich das Team letztlich für die Nutzung der Sprache *„C#“*, unterstützt durch die Entwicklungsumgebung *„Microsoft Visual Studio“*, entschieden.

2.2 Umsetzung der gewählten Variante

Das Programm ist im Kern modular gestaltet, d.h. jeder funktional eigene Bereich ist durch eine eigene Klasse abgetrennt. Die Arbeitsaufteilung wurde dementsprechend auch nach Modulen aufgeteilt. Die Erstellung der Wertetabelle wurde von Herr Meyer umgesetzt. Die Visualisierung des Graphen hat Herr Bauer übernommen.

Die Oberfläche des Programms wurde während der funktionalen Entwicklungsphase nur sporadisch gestaltet. Überlegungen zum generellen Layout haben zwar schon früh stattgefunden, aber da die einzelnen Bausteine möglichst unabhängig vom Aussehen des Programms funktionieren sollten, wurde der Fokus erst später auf das Design gesetzt. Nach einigen Besprechungen, wurde das fertige Layout der graphischen Oberfläche des Programms festgelegt, um eine Orientierung für die Formfaktoren der Bausteine zu bekommen.

Um den Graphen und die Funktionskurven zu zeichnen wurden verschiedene Methoden implementiert und getestet. Anfangs bestand die Idee, die Graphen als Näherungskurven mittels der gegebenen Wertetabelle zu zeichnen, jedoch stellte sich eine geeignete Annäherungsprozedur als zu komplex für die gegebene Aufgabe heraus. Besonders, wenn erst wenige Werte eingetragen sind, ist eine Kurve nicht praktikabel anzunähern. Daher wird in der finalen Version des Programms eine Repräsentation der mathematischen Zusammenhängen der Werte, in Form von Polynomen, codiert und dargestellt.

Für die Wertetabelle fanden wir eine geeignete Struktur und konfigurierten diese, bis sie unseren Vorstellungen entsprach. Dem Nutzer ist es einfach möglich, neue Widerstands-Werte einzutragen bzw. vorhandene zu löschen. Die Werte in Abhängigkeit vom Widerstand werden dann automatisch berechnet.

3 Testen der Anwendung

3.1 Betrachtete Testfälle

3.2 Testprotokolle

4 Dokumentation

4.1 Benutzerhandbuch

4.1.1 Installation

Das Programm ist eine portable Anwendung, jedoch keine alleinstehende. Es wird zum Ausführen ein „dotNET Runtime Environment“ der Version 4.0 oder höher vorausgesetzt. Unterstützt wird eine 64-Bit Variante des Betriebssystems Windows ab Version „Windows 7“.

Bei gegebenen Voraussetzungen reicht ein Kopieren der Anwendungsdatei auf einen Rechner, gefolgt von einem Ausführen der Datei.

Es startet sich direkt, ohne weitere Installationsschritte, die Anwendungsoberfläche.

Zum Deinstallieren der Software, kann die Anwendungsdatei einfach gelöscht und gegebenenfalls aus dem Papierkorb entfernt werden. Konfigurationsdateien oder andere Programmausgaben in Form von Dateien werden nicht angelegt.

4.1.2 Nutzungshinweise

Zur Interaktion mit der Softwareoberfläche wird eine Maus und eine Tastatur vorausgesetzt. Wir, die Entwickler, geben keine Gewähr für die Vollständigkeit oder Korrektheit der Ergebnisse. Weiterhin verantworten wir keine Fehler, welche durch unangemessene Nutzung des Computers oder auch Überlastung der Hardwareressourcen hervorgerufen werden. Dies beinhaltet ebenfalls die Nutzung der Software auf Computern, welche nicht die im Abschnitt Installation genannten Mindestanforderungen erfüllen. Das Logo des Programms, ist das Logo der Berufsschule Technik Rostock und die Urheberrechte liegen beim Rechteinhaber. Der Quellcode wurde von den Projektmitgliedern verfasst und eine Vervielfältigung ist nur für nicht-kommerzielle und gegebenenfalls pädagogische Zwecke gestattet.

4.1.3 Umgang mit der Programmoberfläche

Im Startzustand finden sich im Programm auf der linken Seite ein leeren Graph und auf der rechten Seite eine leere Tabelle, sowie eine Aufreihung von Menüreitern. Das anfangs geöffnete Menü bietet verschiedene Auswahlkästchen und Funktionsknöpfe.

4.1.4 Entwicklerhandbuch

4.1.5 Programmstruktur

Der Programmeinstiegspunkt ist in der statischen Klasse `Program` mit der Funktion `Main`. Diese instanziiert ein Objekt der `BitgraphGUI`-Klasse, welche die eigentliche Darstellungs- und Berechnungslogik, in Form von Unterobjekten, kapselt. Die Struktur des Programms ist im groben geteilt in Menü, Graphen und Tabelle.

Das Menü in Form eines `TabPages`-Objektes, befindet sich direkt in der `BitgraphGUI`-Instanz, wobei die Inhalte desselben unter anderem von Kind-Objekten generiert werden. Es ist gegliedert in die Tabs „Optionen“, „Schaltung“, „Hilfe“ und „Über“.

4.1.6 Schnittstellen

Das Programm bietet keine externen Entwicklungsschnittstellen. Ein Erweitern des Programms ist nicht vorgesehen, die einzelnen Unterklassen sind jedoch so gestaltet, dass sie bei Bedarf leicht wiederverwendet werden können. Besonders die vorhandene Trennung der Berechnungs- und Darstellungslogik des Graphen gestaltet ein Ändern der Darstellungsmethode einfach, da lediglich die aufgerufenen Funktionen in einer eigenen Art implementiert werden müssten. Es wäre also als theoretische Verbesserung der Allgemeinheit des Programms, ein konzipieren von abstrakten Klassen bzw. Schnittstellen, zu den derzeit vorhandenen, denkbar.

5 Quellen

Die API-Dokumentation des „dotNET Framework“:
`https://docs.microsoft.com/de-de/dotnet/api/`