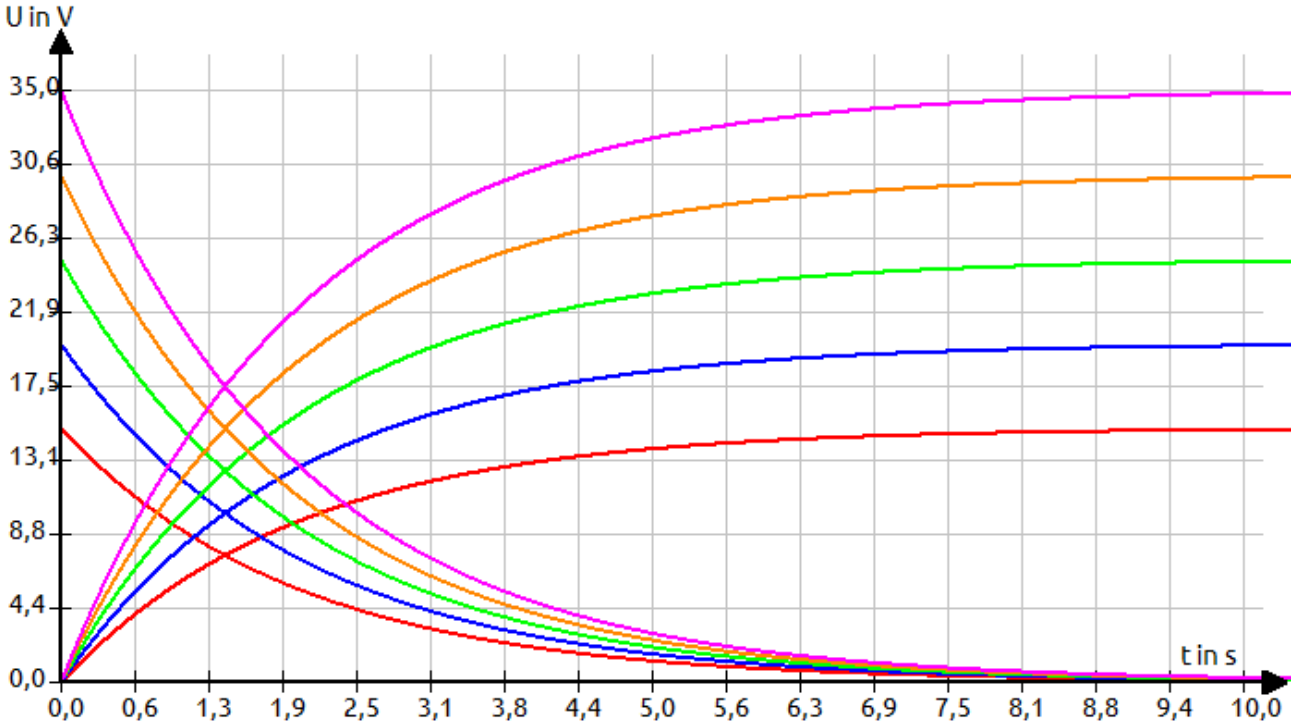
Projektdokumentation

Capacitux



Inhaltsverzeichnis

Anforderungsanalyse

(Pflicht-)Vorgaben

Die Aufgabe für das Projekt war es, eine Software zu entwickeln, die Kurvenverläufe eines elektrischen Bauteils unserer Wahl unter Berücksichtigung der physikalischen Parameter berechnen und ausgeben kann. Für die Darstellung eines solchen Diagramms soll allerdings keine Bibliothek oder vorgefertigte Komponente benutzt werden, sondern das Rendering soll selbst programmiert werden. Die Sprache und die Entwicklungsumgebung (IDE) – falls benötigt – waren nicht vorgegeben.
Das Projekt soll am Ende das fertige und ausführbare Programm, den Source-Code, eine Dokumentation (die Sie gerade lesen), ein Benutzerhandbuch und ein abschließendes Testprotokoll enthalten und entweder als Datenträger oder Download zur Verfügung stehen. Bei der Fertigstellung soll es beim Benutzer auf einem Windows 7 – 64 Bit benutzbar sein. Das Programm darf im Übrigen frei um Zusatzfunktionen erweitert werden, wenn möglich und sinnvoll.

Anforderungen zusammengefasst:

*** mind. 3 Kurvenverläufe zu einem elektrischen Bauteil**

*** Darstellung soll sich den Eingabewerten anpassen**

*** Diagramm selbst rendern**

Eigene Anforderungen & Pläne

Unser Team hat sich recht schnell darauf einigen können, die Spannungsverläufe von Kondensatoren für den Auflade- und Entladevorgang als Basisbetrachtung für das Projekt zu wählen, da Kondensatoren momentan auch Teil des Schulstoffes sind, Dabei haben wir uns gleich zum Ziel gesetzt, die Möglichkeit zu bieten, theoretisch unendlich viele Bauteile miteinander zu vergleichen. Auch eine Wertetabelle wurde in Betracht gezogen.
Die Liste der eigenen Wunschkritieren war schnell gut gefüllt.

Optionale bzw. Wunsch-Kriterien:

* „unendlich viele“ Kondensatoren in einer Liste (Objektorientierung)
* selbstständiges, möglichst freies Zoomen durch den Nutzer
* Farbauswahl für die Kurven
* Möglichkeit, einzelne Kurven auszublenden
* Wertetabelle – in der man auch Werte direkt ändern kann
* Speichern und Laden der „Sitzung“
* Linux – Windows – kompatibel
* Export des Diagramms als Bild
* Auch andere Werte als nur Spannung anzeigen
* Einheiten umrechnen können (Präfixe)
* Diagramm passt sich Tab-Größe an

Soll-Ist-Vergleich

Als Entwicklungsumgebung kommt Lazarus zum Einsatz, da es opensource und damit frei erhältlich und plattformunabhängig ist und weil in der Firma sowieso Delphi benutzt wird, welches in vielen Belangen kompatibel zu Lazarus ist. Die Programmiersprache, in der das Programm geschrieben wurde, ist demnach Object Pascal. Diese ist leicht zu lernen und auch als Lernsprache konzipiert worden und somit für den didaktischen Gebrauch gut geeignet.
Das Zielsystem soll ein 64 Bit-Rechner mit Windows sein. Das Entwicklersystem ist ein 64 Bit-Linux. Die Einrichtung eines Crosscompilers erschien zu kompliziert, weshalb einfach das Programm im Nachhinein auf einem 64-Bit Windows kompiliert wurde.

Die Entwicklung findet zu hause oder auf Arbei statt. Zum Abgleich der Versionen wurde anfangs die Einrichtung eines SVN Repositories in Erwägung gezogen, der Einfachheit halber wurde dies aber durch die Einrichtung eines einfachen FTP-Servers beim Hauptentwickler ersetzt. alle wichtigen Versionen befinden sich auf dem Server. Zur Verständigung wurde Teamspeak-Server eingerichtet, mit dem ganz leicht und unkompliziert via IP-Telefonie kommuniziert werden kann. Beide Dienste befinden sich auf dem gleichen Server, welcher ebenfalls ein Linux-System ist.
Ansonsten fand die Kommunikation größtenteils über E-Mail statt. Die Planung und Durchführung ist aufgrund unterschiedlicher Arbeitszeiten und wohnlicher Bedinungen teils recht schwer gefallen.

Zum aktuellen Zeitpunkt erfüllt die Software alle Pflichtanforderungen und kann zum ihr zugedachten Zweck genutzt werden (wenn auch nicht im all zu professionellen Gebrauch). Es lassen sich (theoretisch) unendlich viele Kondensator-Spannungsverläufe hinzufügen, die sich alle entsprechend ihrer Eingabewerte vehalten. Und schließlich ist das Diagramm – wie eingangs gefordert – selbst gerendert worden und passt sich außerdem dem Anzeigebereich an. Das Programm verhält sich also so, wie es soll.
Zudem gibt es einige zusätzliche Funktionen, die z.b. den Komfort betreffen. Statt eines einzigen Kondensators kann eine ganze Liste von Kondensatoren erstellt werden deren Werte bearbeitet und gespeichert sowie geladen werden können. Um die Graphen im Diagramm unterscheiden zu können, kann man auch die Farbe des Graphen auswählen. Bei der Farbwahl gibt es keine Einschränkung wie vordefinierte Farben, damit eine möglichst gute Unterscheidung bei großer Kondensatoranzahl möglich ist. Für eine bessere Erkennbarkeit wurde außerdem die Möglichkeit geschaffen, Linien dick oder dünn anzeigen zu lassen.
Da es manchmal wichtig ist, nur gewisse Graphen zu betrachten, wurde eine Funktion hinzugefügt, die erlaubt, dass man einzelne Graphen ausblenden und zwischen Auflade- und Entladevorgang auswählen kann. Die einzelnen Kondensatoren werden sowohl in einer Liste, in der einzelne Exemplare aus- und abgewählt werden können, als auch in einer Tabelle aufgelistet. Die Tabelle zeigt alle Werte auf und rechnet außerdem die Einheiten automatisch um. Ebenfalls automatisch umgerechnet werden sollten die Eingaben der Werte im Eingabebereich. Wer 0,000002 V eingibt, soll am Ende 2 µF sehen. Dies war aber aufgrund interner String-Vergleich-Probleme nicht möglich, weshalb die entsprechenden Auswahlfelder vorerst ausgegraut wurden. Aus selbigem Grund war es nicht wirklich möglich, die Einheiten an der Skala umgerechnet anzuzeigen. Entsprechende (Hilfs-)funktionen finden sich aber (teils auskommentiert) im Quelltext wieder. Um diesen ganzen Problemen der Zahlendarstellung zu umgehen, wurde begonnen, einen eigenen Datentyp für einheiten- und präfixbehaftete Werte zu erstellen. Allerdings hat die Zeit nicht mehr gereicht, um den restlichen Quelltext dafür nochmal umzustellen.

Damit das Programm, bzw. dessen Ergebnisse, auch in anderen Medien wie Präsentationen verwendet werden kann, können sowohl das Diagramm als auch die Tabelle (bzw. eigentlich nur ein Ausschnitt derer) als Bild exportiert werden.
Für den Komfort und die Zeitersparnis des Nutzers wurden Funktionen zum Aufrufen der Dokumentation, des Handbuchs und der Versionshistorie, sowie das Öffnen des Lazarus-Projekts mit dem installierten Standard-Programm aus dem Haupt-Programm heraus eingefügt. Auch mehrere Weblinks können vom Programm aus mit dem Standard-Browser geöffnet werden.

Ensteht zur Programmlaufzeit ein Fehler durch z.b. Eingaben des Nutzers oder soll eine Warnung ausgegeben werden, öffnen sich Dialog-Fenster, die entweder entsprechende Warnungen ausgeben oder einen Auswahl-Dialog bereithalten, über den über gewisse Aktionen wie die Terminierung des Programms oder die Löschung aller Werte bestimmt werden kann. Das bedeutet auch, dass Fehler, die zu Programmabstürzen führen könnten und auf Eingaben zurückzuführen sind, an den kritischen Stellen abgefangen werden. Dies soll dem Nutzer, der Auto-Save-Mechanismen oder Sitzungs-Speicherung und -Wiederherstellung kennt, den richtigen Umgang mit der Software ermöglichen, da auf etwaige Auto-Speicher-Funktionen verzichtet wurde.
Genaueres zum Thema Fehler und Fehlerbehandlung finden Sie vor allem im Fehlerprotokoll und dem Handbuch.

Ein Installationstool wurde nicht in Erwägung gezogen, da dies die Portabilität (z.b. für USB-Sticks) des Programms nur unnötig verkomplizieren würde. Außerdem soll das Programm sowieso keine weiteren (Konfigurations-)Dateien im System erstellen, welche dann im Dateisystem-Nirvana verloren gehen könnten. Ein einzelner Programm-Ordner für Installation + Eigene Dateien sorgt für eine möglichst hohe Integrität des Programms.

Begleitende Diagramme

Flussdiagramme o.Ä. wurden aufgrund der Fülle an Funktionen und Hilfsfunktionen nicht erstellt. Wichtige Erläuterungen finden sich, wie es sich gehört, im Quelltext bzw. auch im Benutzerhandbuch. Zudem enthält das Programm keinen besonders komplexen oder interessanten Algorithmus, den es lohnt grafisch darzustellen, sodass er dann tatsächlich erleuchtende Einsichten bringt.

Zeit- und Kostenanalyse

Lizenzkosten fallen für die Software keine an, da sie opensource sein soll. Zusätzlich wurde auf die Benutzung von opensource-Programmen oder zumindest Freeware geachtet, was wiederum sekundäre Lizenzkosten entfallen lässt. Allerdings müssen 2 Programmierer für die Zeit, in der sie das Projekt initiiert und betreut haben, bezahlt werden (In diesem Fall durch z.b. eine fiktive Organisation).

Projektdauer:	*ca. 4 Monate*
Arbeitstage mit 8h Arbeitszeit:	*2 Tage pro Woche* *32 Tage insgesamt*
Stundenlohn eines Programmierers:	*6€*
Bruttogehalt eines Programmieres:	*32 x 8 x 6€ = 1536€*
Gesamtgehaltskosten:	***3072€***