Party Licht Steuerung – Programmentwurf für Lichttechniker mit LabView

Studienarbeit

für die Prüfung zum
Bachelor of Engineering

im Studiengang TIT08I
an der Dualen Hochschule Baden-Württemberg Mosbach

von

Tim Berger

17. Juni 2011

Bearbeitungszeitraum:	6. Theoriephase
Matrikelnummer:	115435
Ausbildungsfirma:	Kurtz Holding GmbH & Co. Beteiligungs KG
Gutachter der DHBW Mosbach:	Prof. Dr. Wolfgang Funk

Zusammenfassung

Abstract

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

LabView LV

LJ Light Jockey (dt. Lichttechniker)

NI National Instruments

1 Einleitung

Diese Studienarbeit dokumentiert den Programmentwurf einer Party-Lichtsteuerung, vom Design der Anwendung über die Implementierung bis hin zur Bereitstellung einer lauffähigen Applikation. Die Programmcode wird mit LabVIEW von National Instruments entwickelt.

1.1 Aufgabenstellung

Es ist ein Programm für Lichttechniker zu entwickeln, mit dem eine viel zahl von Scheinwerfer angesteuert werden kann.

1.2 Anforderungen

Der Light Jockey (LJ) stellt für verschiedene Lichtkanäle Intensität und Farbe ein. Für eine Gruppe von Lichkanälen (Set) kann eine Wartezeit, Überblendungszeit, Nachlaufzeit und Name eingestellt werden. Wählt der LJ die Schaltfläche zum aufnehmen, öffnet sich ein Fenster in dem die gewünschten Parameter übergeben werden. Nach der Bestätigung durch ein Klick auf die OK-Schaltfläche wird das erstellte Set hinten an die Queue angefügt.
Hat der Bediener einige Sets angelegt, wird mit der Abspiel-Schaltfläche das aufgenommene Programm durchlaufen. Ein Set das abgespielt wird wartet die angegebene Zeit, dann wird die Farbe bis zur Intensität über die Überblendungszeit hochgefahren. Jetzt beginnt die Nachlaufzeit, ist diese verstrichen wird mit dem nächsten Set aus der Queue fortgefahren. Der Bediener kann jeder Zeit eine abspielende Queue mit der Stopp-Schaltfläche anhalten.
Über die Menüleiste kann der Bediener mit dem Menüpunkt „Datei“ die Queue speichern und laden. In beiden Fällen öffnet sich eine Dialogbox in dem nach Speicher- bzw. Ladepfad gefragt wird.

1.3 Aufbau der Arbeit

Bla Bla Blaaa

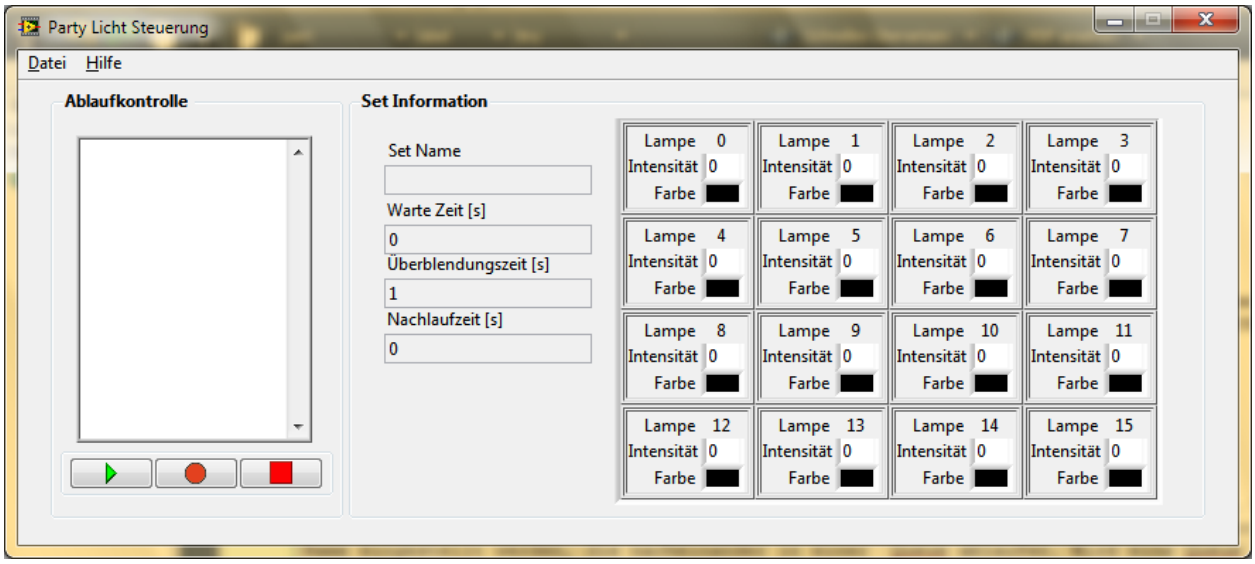

Abbildung 1: Bedienoberfläche für die Party-Lichtsteuerung

2 LabVIEW als Programmiersprache

LabVIEW ist ein grafisches Programmiersystem von National Instruments. Das Akronym steht für „Laboratory Virtual Instrumentation Engineering Workbench“.
Die Programmierung erfolgt in der graphischen Programmiersprache „G“. LabVIEW-Programme werden als Virtuelle Instrumente (VIs) bezeichnet. [NI11a]
Sie bestehen aus drei Komponenten:

Frontpanel Das User-Interface, über welches der Anwender mit dem VI interagiert.

Blockdiagramm Stellt den Programmcode des VIs dar.

Anschluss Dient zur Anbindung an weitere VIs. Bestimmt Übergabe und Rückgabe Werte.

In LabVIEW liegt die Ausführung von VIs dem Datenflussmodell zugrunde. Ein Blockdiagrammknoten (Bsp. Addition) wird ausgeführt, sobald all seine Eingänge belegt sind. Ist die Ausführung eines Knotens abgeschlossen, werden die Daten an die Ausgabeanschlüsse übergeben und die Ausgabedaten dann an den nächsten Knoten im Datenflussdiagramm weitergeleitet. [**?**] Die unter LabVIEW erstellten Blockdiagramme werden von einem grafischen Compiler in optimierten Maschinencode übersetzt. Dadurch ist die Performance vergleichbar mit der anderer Hochsprachen wie C oder Pascal. [NI11b]
Abbildung 2 zeigt eine kleine Demonstration. Es wird aus den Eingängen A und B ein Ausgang C berechnet. Die Formel wird im Blockdiagramm abgebildet. Sie lautet:

$$C = \frac{A + B^2}{4}$$

Des Weiteren findet die Berechnung in einer While-Schleife statt. Die Abbruchbedingung ist die Betätigung der Stopp-Schaltfläche.

2.1 Entwurfsmuster - Design Pattern

Zur Entwicklung einer umfangreichen Applikation ist es unerlässlich mit Entwurfsmustern zu arbeiten. Sie helfen nicht nur dem Entwickler den Überblick nicht zu verlieren sondern machen es auch für Außenstehende einfacher den Code zu lesen und modifizieren.
LabView bietet neun verschiedene Entwurfsmuster. Für welches man sich entscheidet hängt von folgenden Kriterien ab:

- Gibt es eine feste Reihenfolge / Sequenzen von Befehlen?

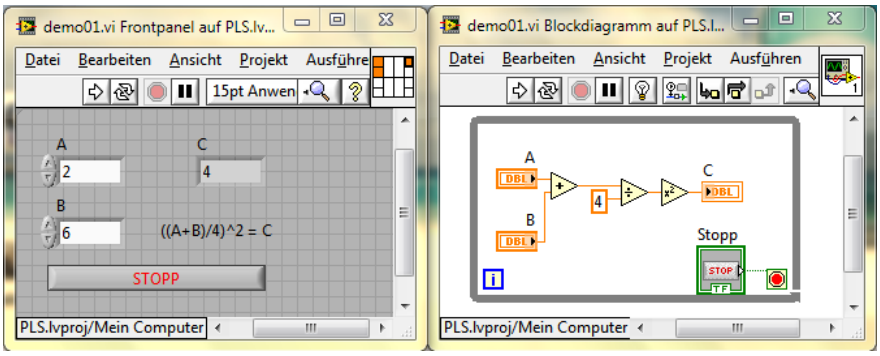

Abbildung 2: VI Demonstration: Links Frontpanel, Rechts Blockdiagramm

- Muss das Programm mit einem User-Interface agieren?
- Ist die Datenverarbeitung intensiv?
- Gibt es parallele Operationen?

Im folgenden gehe ich auf einige Entwurfsmuster ein, die für meine Problemstellung infrage kommen könnten. Das sind: der Zustandsautomat, Master/Slave-Entwurfsmuster, Ereignisbehandler für Benutzeroberfläche und das Erzeuger/Verbraucher-Entwurfsmuster. Später im XXX Kapitel, nach der Programmanalyse, werde ich auf meine Wahl des Entwurfsmuster eingehen.

2.1.1 Zustandsautomat

Mit jedem Zustand wird ein bestimmter Blockdiagramm Ausschnitt ausgeführt und ermittelt, zu welchem Zustand weitergesprungen wird. In einer While-Schleife wird eine Case-Struktur ausgeführt.

2.1.2 Master/Slave-Entwurfsmuster

Bei diesem Entwurfsmuster gibt es eine Master-Schleife und mindestens eine Slave-Schleife. Die Master Schleife wird immer ausgeführt. Sie benachrichtigt Slave-Schleifen, einen bestimmten Code auszuführen. Die Slave-Schleifen werden vollständig ausgeführt und warten dann auf die nächste Benachrichtigung.

2.1.3 Einfacher Ereignisbehandler für Benutzeroberfläche

Dieses Entwurfsmuster wird verwendet für die Verarbeitung von Ereignissen der Benutzeroberfläche. Die Vorlage eignet sich für Dialogfelder und andere Programmoberflächen.

Des Weiteren kann man benutzerdefinierte Ereignisse erzeugen und ausführen, die wie Ereignisse der Benutzeroberfläche behandelt werden.

2.1.4 Erzeuger/Verbraucher-Entwurfsmuster

Hier werden zwei separate While-Schleifen unabhängig voneinander ausgeführt: Die erste Schleife erzeugt Daten, während die zweite Schleife die Daten verarbeitet. Obwohl sie parallel ausgeführt werden, werden zwischen den Schleifen über Queues Daten ausgetauscht. Diese Vorlage bietet die Möglichkeit bei Benutzereingriffen asynchron Code auszuführen, ohne die Reaktionszeit der Benutzeroberfläche zu beeinträchtigen. So kann man durch die parallele Ausführung der Schleifen Leistungssteigerung des Programms erzielen.

3 Programm Analyse

3.1 Ablaufdiagramm

Mit Ablaufdiagrammen wird der Programmfluss illustriert. Mit deren Hilfe kann man eine Aufgabe in handhabbare Funktionen teilen. Abbildung 3 zeigt das Ablauf Diagramm für die Abspiel-Funktion in der Party-Licht-Steuerung. Ein Knoten repräsentiert eine Funktion.

3.2 Datenfluss Diagramm

Datenfluss Diagramme haben die Aufgabe zu zeigen, welchen Weg die Daten durch eine Applikation nehmen. Abbildung 4 zeigt das Datenfluss Diagramm für die Abspiel-Funktion in der Party-Licht-Steuerung. Die Knoten (Kreise) repräsentieren die Prozesse. Eine externe Entität ist das Licht Kontroll-System. Die Pfeile zeigen den Datenfluss.

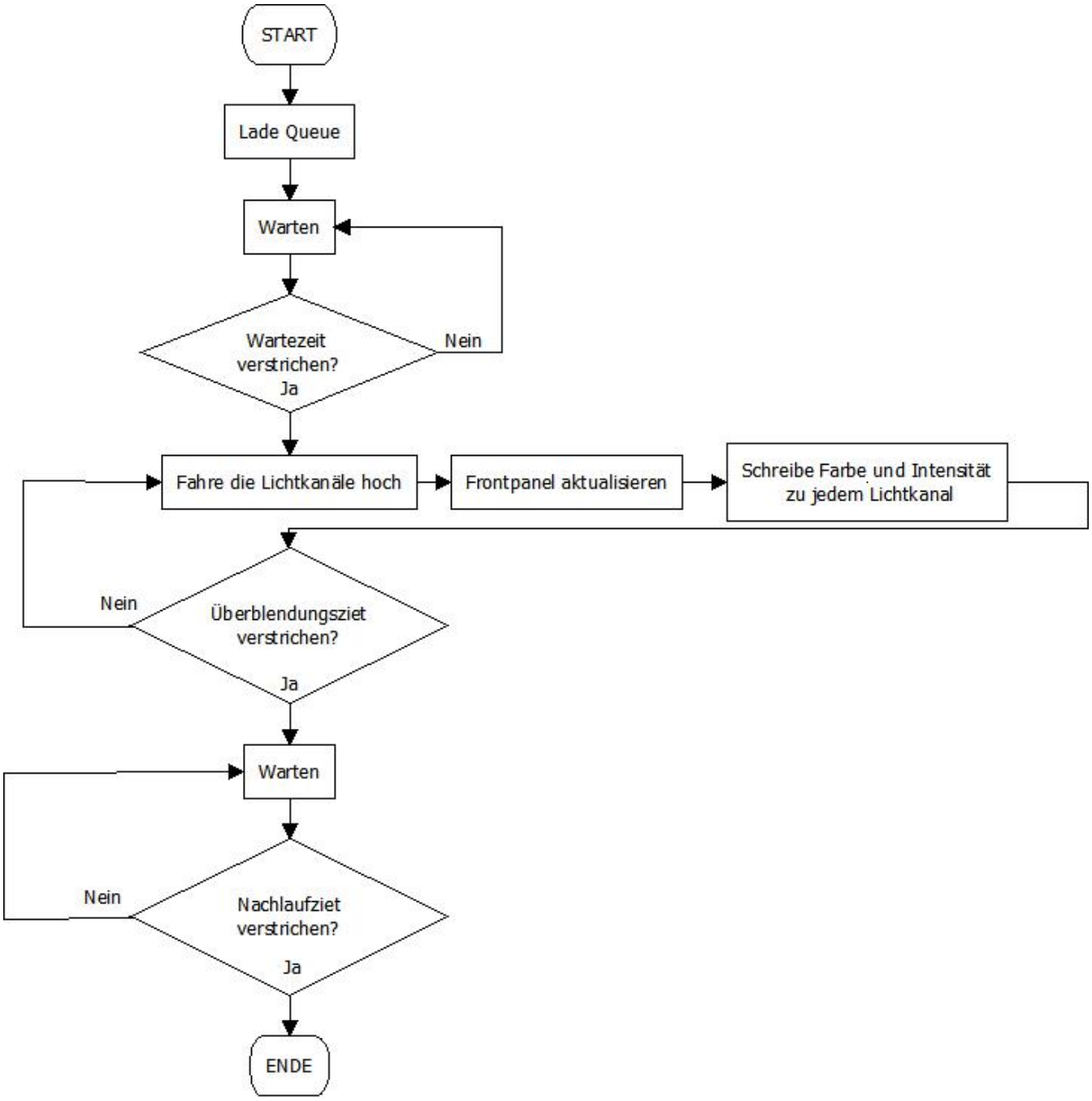

Abbildung 3: Ablaufdiagramm für die Abspiel-Funktion

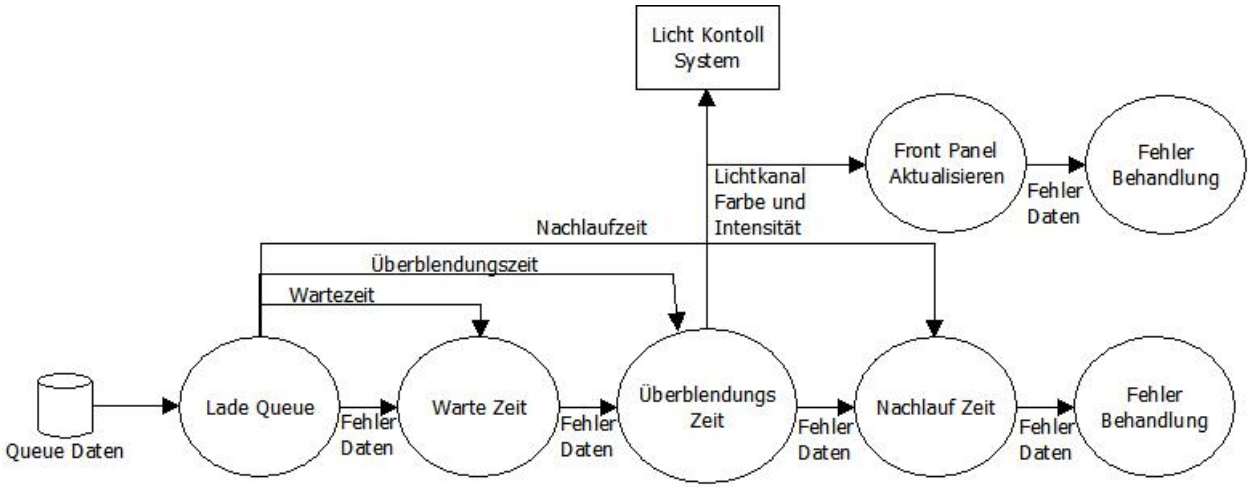

Abbildung 4: Datenfluss Diagramm für die Abspiel-Funktion

4 User Interface

5 Code Implementierung

5.1 Auswahl des Design Pattern

5.2 Timing

5.3 Auswahl der Datentypen

5.4 Init und Shutdown Funktion

5.5 Aufnahme-Funktion

5.6 Abspiel-Funktion

5.7 Stopp-Funktion

5.8 Speichern und Lade Funktion

5.9 Fehlerbehandlung

6 Testen

7 Anwendung

7.1 Stand-Alone Applikation

Literatur

[NI11a] NI: *Einführung in LabVIEW - Dreistündiger Einführungskurs*. http://www.ni.com/pdf/academic/d/labview_3_hrs.pdf, 2011. Zugriff am 2.6.2011 um 14:27 in Datei *„labview3hrs.pdf“*.

[NI11b] NI: *Wie funktioniert der Compiler von NI LabVIEW?* http://zone.ni.com/devzone/cda/tut/p/id/11936, 2011. Zugriff am 2.6.2011 um 14:27 in Datei *„Compiler LabVIEW - Developer Zone - National Instruments.pdf“*.

Anhang

A.1 Text

A.2 Text

A.3 Text

A.4 Text

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Party Licht Steuerung – Programmentwurf für Lichttechniker mit LabView

selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe.

Mosbach, den 17. Juni 2011

Tim Berger