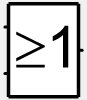
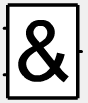
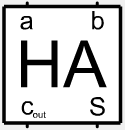
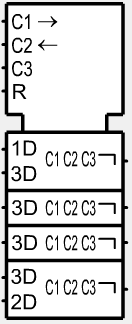
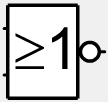
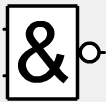
# Logik Simulator



Projektteam: Moritz Wahlenmeier, Bastian Schäfer &   
. Stephen Heisser

Auftraggeber: Markus Greiner



# Inhaltsverzeichnis

* [Logik Simulator 1](#_Toc10554222)
* [Inhaltsverzeichnis 2](#_Toc10554223)
* [Allgemeine Informationen 3](#_Toc10554224)
* [Projektbeschreibung 4](#_Toc10554225)
* [Funktionsweise (kurz) 5](#_Toc10554226)
* [Optimierungsmöglichkeiten 6](#_Toc10554227)
* [Funktionsweise (Lang) 7](#_Toc10554228)
* [Zustand 10](#_Toc10554229)
* [Wie es vorgefunden wurde: 10](#_Toc10554230)
* [Momentaner Zustand: 11](#_Toc10554231)
* [Zeitplan 12](#_Toc10554232)
* [Geplantes Material 12](#_Toc10554233)
* [Probleme 13](#_Toc10554234)

# Allgemeine Informationen

**Team:**

Moritz Wahlenmeier   
 moritz.wahlenmeier@yahoo.de

Bastian Schäfer  
 schaefer2001@web.de

Stephen Heisser  
 st.heisser@gmail.com

**Auftraggeber:**

Herr Markus Greiner  
greiner@gds2.de  
Lehrer Gottlieb Daimler Schule 2  
Böblinger Str. 73, 71065 Sindelfingen

# Projektbeschreibung

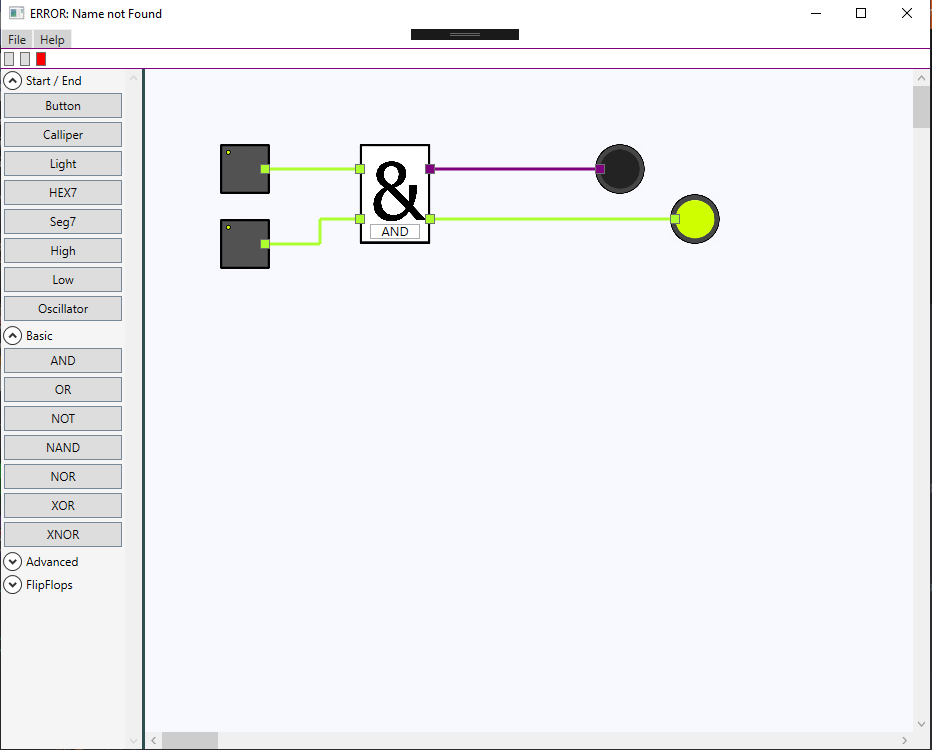
Das Ziel ist das ablösen des alten, auf Flash basierendem, Programms „Logiflash“, durch eine auf Windows Presentation Foundation (WPF) basierende Desktopanwendung.   
Da „Logiflash“ alte Symbole beinhaltet und in vielen Situationen nicht mehr von Browsern unterstützt wird. Des Weiteren ist das Programm aktuell nicht sehr benutzerfreundlich. „Logiflash“ wird aktuell im Praktikum Informatik Unterricht im ersten Schuljahr des 2BKI benutzt und soll gegebenenfalls durch das neue Programm abgelöst werden.

Hierzu muss das Simulationsprogramms zum Erstellen und Testen von logischen Schaltungen erstellt werden.

Als Unterstützung dazu, dass das Projekt von mehreren Personen gleichzeitig bearbeitet werden kann und jederzeit gedownloadet werden kann, wird das Programm „Github Desktop“ verwendet.

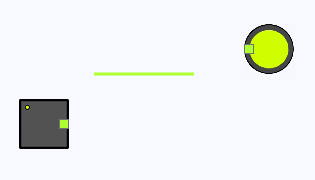
# Funktionsweise (Benutzeroberfläche)

In der unteren rechten Leiste befindet sich eine Liste mit allen Logik Gattern als auch Anfangs- und Endbausteine wie Buttons und Lampen, welche per Drag und Drop auf die Arbeitsfläche gezogen werden können. Um diese zu verbinden klickt man auf die Output Fläche (Quadrate rechts an Baustein) und zieht ein Kabel auf die Input Fläche (Quadrate links an Baustein). Kabel können nur vertikal oder horizontal gezogen werden. Auf jedes vertikale folgt ein horizontales Kabel und andersrum. Schaltungen können unter dem Menüpunkt File oder durch Shortcuts in der zweiten Zeile gespeichert, geöffnet oder gedruckt werden. Durch drücken des roten Icons in der zweiten Zeile wird der Lösch-Modus aktiviert oder deaktiviert. Im Lösch-Modus werden Gatter und Kabel durch Anklicken gelöscht.

  
logisch 0: lila logisch1: grün

Funktionsweise (ausführlich)

# Optimierungsmöglichkeiten

****

Beim Verschieben der Bausteine bleiben verbundene Kabel an ihrer vorherigen Position die logische Verbindung bleibt aber weiterhin bestehen.

Optimiert: Algorithmus setzt Kabelende /Anfang an neue Position und baut je nach Verschiebung benötigte vertikale/horizontale Kabel ein (Aufwendig).

Es fehlen noch einige Bilder für die FF und komplexeren Schaltungen, die noch angefertigt werden müssen und manche bereits existierende Bilder müssen noch angepasst werden.

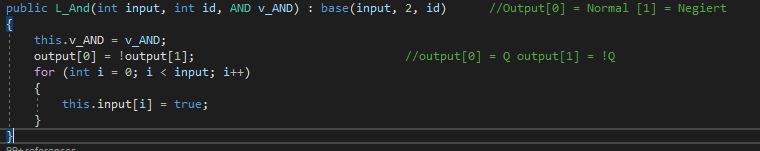
# Funktionsweise (Code)

* Logikgatter:

Alle Logikgatter müssen soweit verstanden sein, dass man sie mit C#  
 programmieren kann. Dabei muss darauf geachtet werden, dass manche Logikgatter auch mit unterschiedlich vielen Eingängen funktionieren müssen.

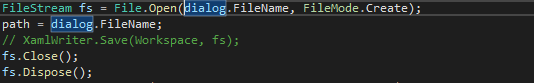
z.B.:

Ein UND/AND Gatter kann von 2 bis 8 Eingänge haben.

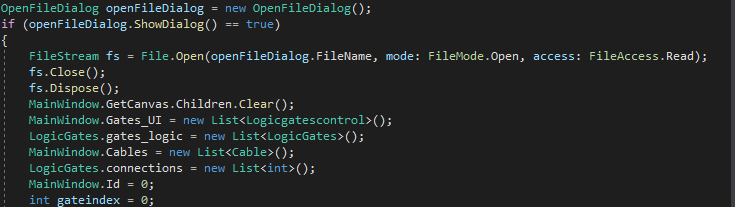


* Speichern/Laden

Beim Speichern (hier mit FileStream) müssen alle Listen ausgelesen werden und dann in einer Datei gespeichert werden.

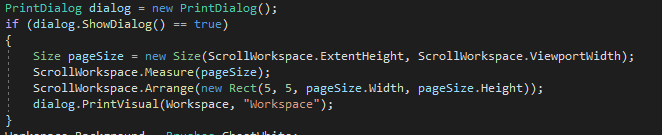


Beim Laden (was hier mit Filestream erledigt wird) müssen alle Listen zuerst geleert werden, damit die alten Gatter nicht die neuen Gatter beeinflussen.



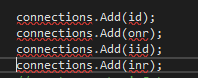
* Drucken

Der Canvas muss vom Programm ausgelesen werden und dann ausgedruckt werden.



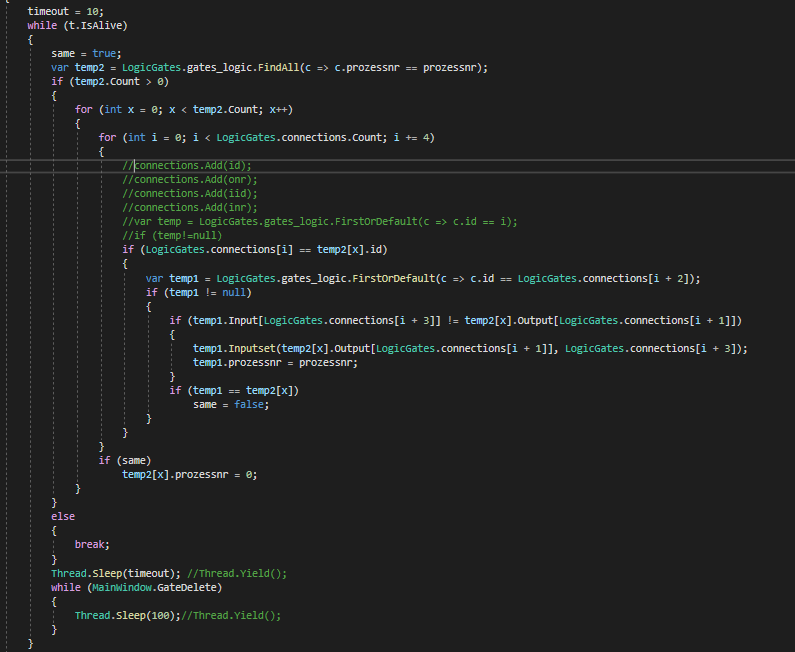
* Verbindungen:

Verbindungen arbeiten mit der einzigartigen Id jedes Logikgatters und jeder Verbindungslinie. Wird eine Verbindung eingeleitet, so wird in einer Liste die Id des Ausgangsgatts gespeichert, welcher Ausgang, die Id des Eingangsgatters und welchen Eingang die Verbindung hat.



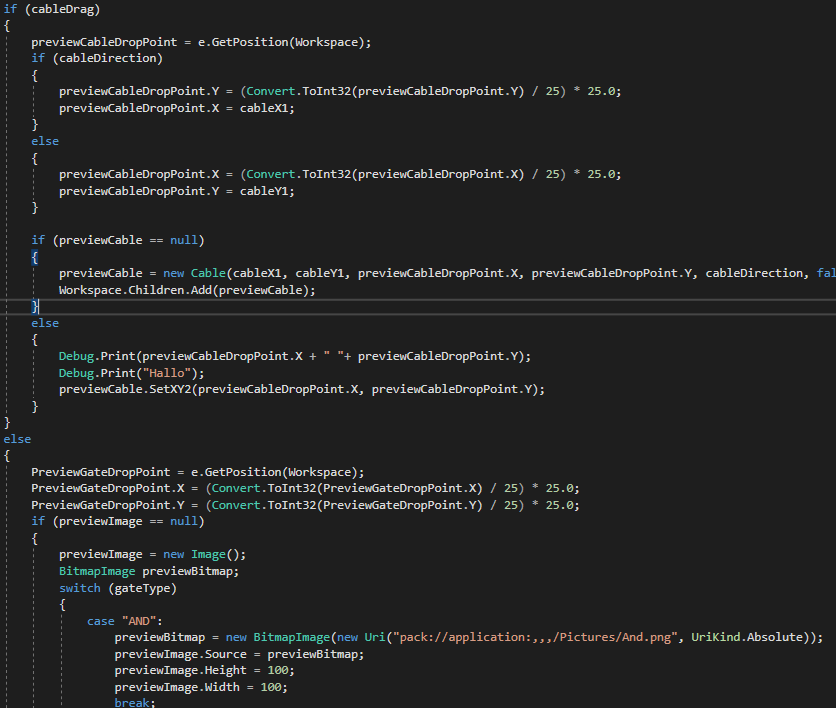
* Prozesstoken:

Wenn ein Taster oder ähnliches ein Signal abgibt wird ein „Prozesstoken“ erstellt der innerhalb eines neuen Threads die Liste mit den Verbindungslinien durchgeht und überprüft ob diese durch das Signal verändert wird. Falls nichts mehr verändert wird, wird der Thread beendet.

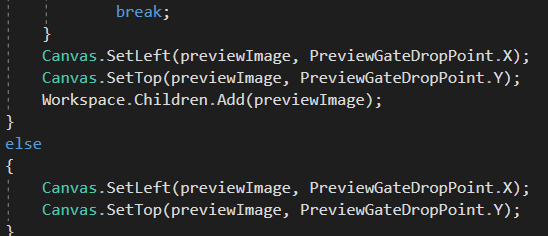


* Drag Drop

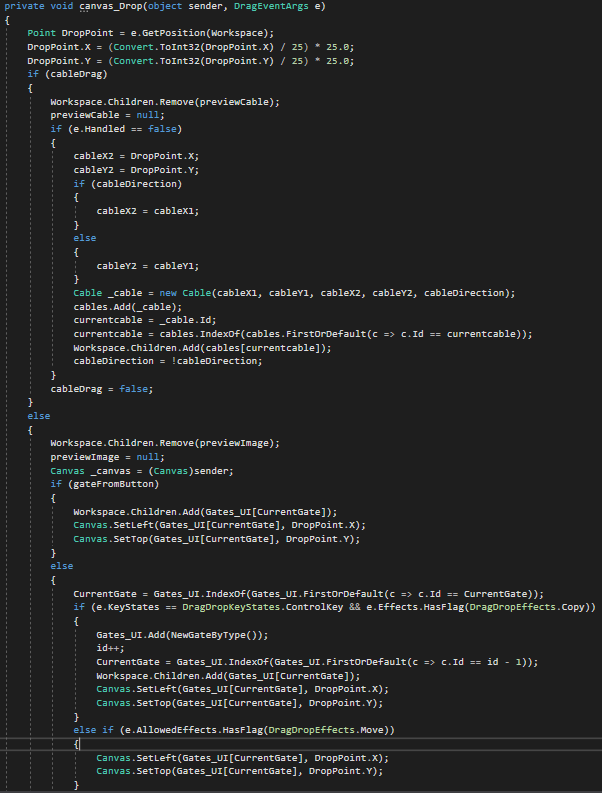
Wird ein Baustein oder Kabel über den Canvas gezogen wird bis zum Drop eine Preview generiert, dessen Position auf dem Canvas wiederholend aktualisiert wird, die zeigt wo das Element beim Drop positioniert wird



(restliche cases aus platz gründen ausgeschnitten )

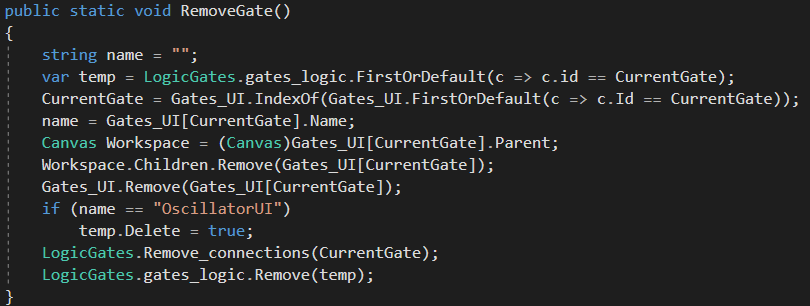
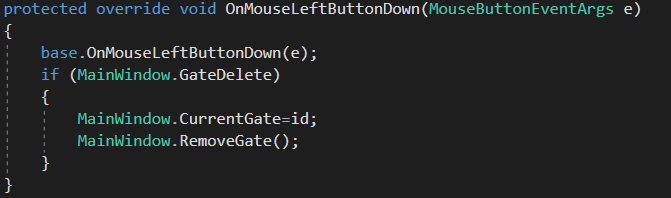


Beim Drop wird die Preview gelöscht und durch ein, je nachdem ob Steuerung gedrückt wird, neues oder bisher vorhandenes Element ersetzt



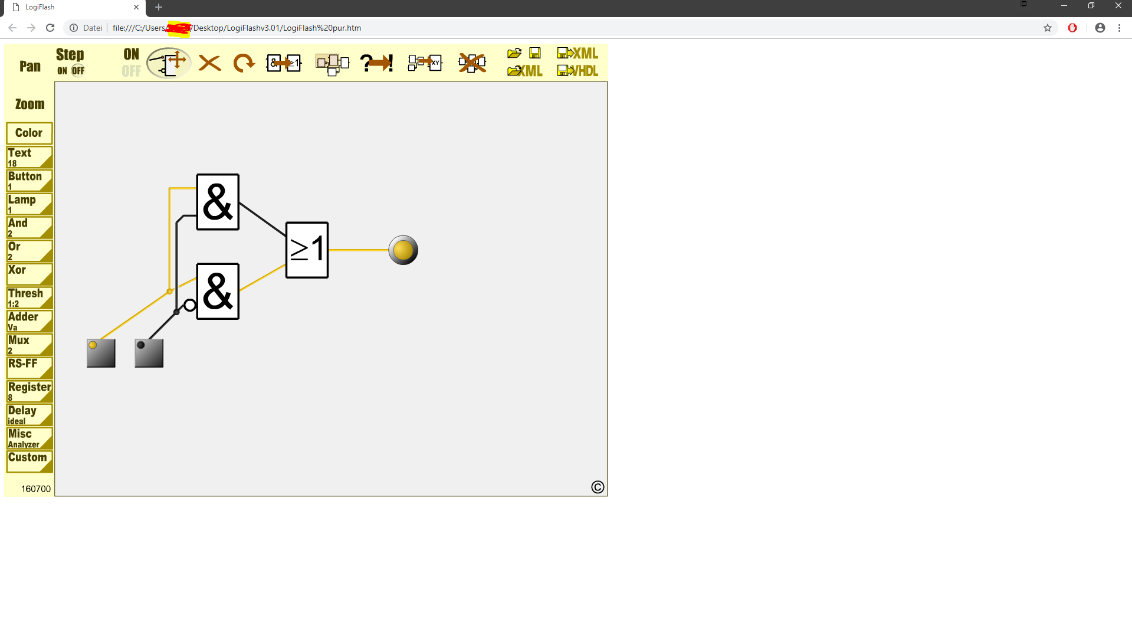
* Löschen

Wird ein Element während der Lösch-Modus aktiviert ist angeklickt wird dieses aus der Gatter Liste entfernt und die logischen Verbindungen gelöscht



# Zustand

## Wie es vorgefunden wurde:



Button2

Button1

* Nicht skalierend abhängig von Bildschirmgröße
* Linien ziehen nicht anwenderfreundlich
* Testfunktion änderbar im laufenden Betrieb
* Flash basierend (wird häufig nicht mehr von Browsern unterstützt)
* Veraltete Symbole
* Umständliches umstellen für Löschen von Bausteinen
* Text ist nicht an Bausteine gebunden und keine Pflicht
* Benötigt ein weiteres Programm, um Schaltungen auszudrucken

## Momentaner Zustand:

* Erreicht:
* Skalierend
* auf (WPF) basierende Desktopanwendung
* Tasten-kürzel
* Direktes Drucken aus dem Programm heraus
* Feste namens Zuordnung zu Bauteilen
* Nicht erreicht:

- Debugging Umgebung

- Kontextmenü

- Möglichkeit simple Programme auf einem Mikrokontroller zu testen

# Zeitplan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Wahlenmeier** | Zeit in h | **Schäfer** | Zeit in h | **Heisser** | Zeit in h |
| Programm Organisation | 2 | Grafikdesign | 50 | Logik Bausteine in C# | 29 |
| Programm in C für Mikrokontroller | 0 | Tastatur Kürzel | 0 | Verbindungslinien | 32 |
| Schnittstelle für Mikrokotroller zu C# | 0 | Englische und deutsche Bedienelemente | 0 | Abspeichern und öffnen von Programm | 28 |
| Drag & Drop | 54 |  |  | Fleißarbeit | 35 |
| Löschen / zoomen /  Toolbox | 8 |  |  |  |  |
| Fehlerdiagnose | 24 | Fehlerdiagnose | 11 | Fehlerdiagnose/Cleanup | 53 |
| Lern Zeit | 26 | Lern Zeit | 10 | Lern Zeit | 54 |
| PowerPoint | 2 | PowerPoint | 2 | PowerPoint | 2 |
|  |  |  |  |  |  |
| Gesamtzeit | 117 | Gesamtzeit | 73 | Gesamtzeit | 231 |

# Geplantes Material

|  |  |  |  |
| --- | --- | --- | --- |
| **Materialname** | **Anzahl** | **Preis in Euro €** | **Gestellt**  **von Team** |
| XC886 Mikrokontroller | 1 | 60 | X |
| USB-A zu USB-B Kabel | 1 | 1 | X |
|  |  |  |  |
| Gesamtpreis |  | 61 |  |

# Probleme

* Bugs
  + Neue Funktionen haben häufig alte Funktionen behindert
  + CPU Überlastung bei der zyklischen Abarbeitung des Programms
  + Bildschirm Größen unterschiede
  + Schul-PCs zu langsam
* Keine richtige Arbeitseinteilung